
FreeBSD Documentation

Release 10.1

Claudia Mane

July 12, 2015

1	&title;	3
1.1	What is FreeBSD?	3
1.2	Cutting edge features	3
1.3	Powerful Internet solutions	3
1.4	Advanced Embedded Platform	3
1.5	Run a huge number of applications	3
1.6	Easy to install	4
1.7	FreeBSD is <i>free</i>	4
1.8	Contributing to FreeBSD	4
2	&title;	5
2.1	Introduction	5
3	&title;	15
3.1	Experience the possibilities with FreeBSD	15
3.2	FreeBSD is a true open system with full source code.	15
3.3	FreeBSD runs thousands of applications.	15
3.4	FreeBSD is an operating system that will grow with your needs.	16
3.5	What experts have to say	16
4	&title;	17
4.1	BSD Daemon	17
4.2	“Powered by FreeBSD” Logos	19
4.3	Old Advertisement Banners	19
4.4	Graphics Use	19
4.5	Trademarks	20
5	&title;	21
6	&title;	23
6.1	Subversion	23
6.2	Other options	23
7	&title;	25
8	&title;	27
8.1	&os; offers many unique features.	27
9	&title;	31

9.1	Usage Guideline	31
9.2	Resource	31
9.3	Sample	32
10	&title;	37
10.1	Questions about FreeBSD...	37
10.2	Questions about the contents of this WWW server...	37
10.3	Snail mail, phone and fax	37
10.4	Who Is Responsible for What	37
11	&title;	39
11.1	White Papers	39
11.2	Presentations	39
11.3	Flyers	39
12	iSCSI	105
13	pfSense Tutorial	107
14	Bacula	109
15	FreeBSD/mips	111
16	Building self-contained PBIs from Ports (Automagically)	113
17	An Open Source Enterprise VPN Solution with OpenVPN and OpenBSD	115
18	“fininstall” - the new FreeBSD installer	117
19	Measured (almost) does Air Traffic Control	119
20	BSD licensed C++ compiler	121
21	Closing	123
22	Google SoC	125
23	A closer look at the ZFS file system	127
24	Interfacing embedded FreeBSD with U-Boot	129
25	Introduction to Debugging the FreeBSD Kernel	131
25.1	Introduction to Debugging the FreeBSD Kernel	131
26	DTrace for FreeBSD	133
27	X.org	135
28	What Not To Do When Writing Network Applications	137
29	Using FreeBSD to Promote Open Source Development Methods	139
30	SCTP - SCTP what it is and how to use it	141
31	Porting FreeBSD/ARM to Marvell Orion System-On-Chip	143
32	Opening session	145

33 &title;	151
33.1 Privacy Policy	151
33.2 What information is collected on our Site:	151
33.3 How the information collected is used:	152
33.4 How your information is protected:	152
33.5 When your information may be disclosed:	152
33.6 Updates to this privacy policy:	152
33.7 Sale of assets:	153
33.8 Third party links:	153
33.9 Contacting Us:	153
 34 &title;	 155
34.1 Go fix/close a PR!	155
 35 &title;	 157
35.1 Books	159
35.2 CDROMs	160
35.3 Magazines	177
 36 &title;	 179
36.1 Currently Supported Releases	179
36.2 Most Recent Release(s)	179
36.3 Future Releases	180
36.4 Prior Releases Which Have Reached End-Of-Life	180
 37 &title;	 185
37.1 RELEASE versions of FreeBSD	185
37.2 Snapshot versions of FreeBSD	186
37.3 Documentation for -CURRENT and -STABLE	186
37.4 Other Sites	186
 38 &title;	 187
 39 &title;	 189
39.1 What Are Snapshots?	189
39.2 Getting Snapshots	189
39.3 Things You Might Want to Know	189
 40 &title;	 191
40.1 Problem Reporting	191
 41 &title;	 193
41.1 Choosing an Architecture	193
41.2 Choosing an Image	193
41.3 &os; Deployment Statistics	193
41.4 &os; &rel.current;-RELEASE	194
41.5 &os; &rel2.current;-RELEASE	194
41.6 &os; &rel3.current;-RELEASE	194
41.7 Development Snapshots	195
41.8 &os; &rel.head;-CURRENT	195
41.9 &os; &rel.current;-STABLE	195
 42 Why you should use a BSD style license for your Open Source Project	 197
42.1 Introduction	197
42.2 Very Brief Open Source History	197

42.3	Unix from a BSD Licensing Perspective	197
42.4	The Current State of FreeBSD and BSD Licenses	198
42.5	The origins of the GPL	198
42.6	The origins of Linux and the LGPL	199
42.7	Open Source licenses and the Orphaning Problem	199
42.8	What a license cannot do	200
42.9	GPL Advantages and Disadvantages	200
42.10	BSD Advantages	201
42.11	Specific Recommendations for using a BSD license	201
42.12	Conclusion	202
42.13	Addenda	203
43	Abstract	205
43.1	Introduction	205
43.2	FreeBSD as a set of building blocks	206
43.3	Collaborating with FreeBSD	209
43.4	Conclusion	211
44	Committer's Guide	213
44.1	Administrative Details	213
44.2	OpenPGP Keys for OS	214
44.3	Kerberos and LDAP web Password for OS Cluster	215
44.4	Commit Bit Types	215
44.5	Subversion Primer	216
44.6	Setup, Conventions, and Traditions	233
44.7	Commit Log Messages	235
44.8	Preferred License for New Files	236
44.9	Keeping Track of Licenses Granted to the OS Project	237
44.10	Developer Relations	238
44.11	If in Doubt...	238
44.12	Bugzilla	239
44.13	Phabricator	239
44.14	Who's Who	239
44.15	SSH Quick-Start Guide	240
44.16	COVERITY Availability for OS Committers	240
44.17	The OS Committers' Big List of Rules	241
44.18	Support for Multiple Architectures	245
44.19	Ports Specific FAQ	247
44.20	Issues Specific to Developers Who Are Not Committers	253
44.21	Information About GA	254
44.22	Miscellaneous Questions	254
45	Contributing to FreeBSD	257
45.1	What Is Needed	257
45.2	How to Contribute	259
45.3	Contributing to ports	261
46	CUPS on FreeBSD	267
46.1	An Introduction to the Common Unix Printing System (CUPS)	267
46.2	Installing the CUPS Print Server	267
46.3	Configuring the CUPS Print Server	267
46.4	Configuring Printers on the CUPS Print Server	268
46.5	Configuring CUPS Clients	268
46.6	CUPS Troubleshooting	269

47 Explaining BSD	273
47.1 What is BSD?	273
47.2 What, a real UNIX?	273
47.3 Why is BSD not better known?	274
47.4 Comparing BSD and Linux	275
48 Filtering Bridges	279
48.1 Why use a filtering bridge?	279
48.2 How to Install	279
48.3 Final Preparation	280
48.4 Enabling the Bridge	281
48.5 Configuring The Firewall	281
48.6 Contributors	283
49 Fonts and FreeBSD	285
49.1 Introduction	285
49.2 Basic terminology	285
49.3 What font formats can I use?	285
49.4 Setting a virtual console to 80x60 line mode	286
49.5 Using type 1 fonts with X11	286
49.6 Using type 1 fonts with Ghostscript	288
49.7 Using type 1 fonts with Groff	289
49.8 Converting TrueType fonts to a groff/PostScript format for groff	291
49.9 Can TrueType fonts be used with other programs?	292
49.10 Where can additional fonts be obtained?	293
49.11 Additional questions	293
50 How to get best results from the FreeBSD-questions mailing list	295
50.1 Introduction	295
50.2 How to subscribe to FreeBSD-questions	295
50.3 How to unsubscribe from FreeBSD-questions	296
50.4 Should I ask <code>-questions</code> or <code>-hackers</code> ?	297
50.5 Before submitting a question	297
50.6 How to submit a question	297
50.7 How to follow up to a question	299
50.8 How to answer a question	299
51 Build Your Own OS Update Server	301
51.1 Acknowledgments	301
51.2 Introduction	301
51.3 Prerequisites	301
51.4 Configuration: Installation & Setup	302
51.5 Building Update Code	303
51.6 Building a Patch	307
51.7 Tips	309
52 Writing a GEOM Class	311
52.1 Introduction	311
52.2 Preliminaries	311
52.3 On FreeBSD Kernel Programming	313
52.4 On GEOM Programming	314
53 Implementing UFS Journaling on a Desktop PC	319
53.1 Introduction	319
53.2 Understanding Journaling in OS	320

53.3	Steps During the Installation of OS	320
53.4	Setting Up Journaling	321
53.5	Troubleshooting Journaling	324
53.6	Further Reading	325
54	Mirroring FreeBSD	327
54.1	Contact Information	327
54.2	Requirements for FreeBSD mirrors	327
54.3	How to Mirror FreeBSD	329
54.4	Where to mirror from	331
54.5	Official Mirrors	332
54.6	Some statistics from mirror sites	333
55	Independent Verification of IPsec Functionality in FreeBSD	335
55.1	The Problem	335
55.2	The Solution	335
55.3	The Experiment	336
55.4	Caveat	336
55.5	IPsec—Definition	336
55.6	Installing IPsec	336
55.7	src/sys/i386/conf/KERNELNAME	337
55.8	Maurer's Universal Statistical Test (for block size=8 bits)	337
56	LDAP Authentication	341
56.1	Preface	341
56.2	Configuring LDAP	341
56.3	Client Configuration	345
56.4	Security Considerations	348
56.5	Useful Aids	350
56.6	OpenSSL Certificates for LDAP	350
57	OS Support for Leap Seconds	351
57.1	Introduction	351
57.2	Default Leap Second Handling on OS	351
57.3	Cautions	351
57.4	Testing	352
57.5	Conclusion	352
58	LINUX emulation in OS	353
58.1	Introduction	353
58.2	A look inside...	353
58.3	Emulation	360
58.4	LINUX emulation layer -MD part	365
58.5	LINUX emulation layer -MI part	368
58.6	Conclusion	377
58.7	Literatures	378
59	FreeBSD Quickstart Guide for LINUX Users	379
59.1	Introduction	379
59.2	Default Shell	379
59.3	Packages and Ports: Adding Software in OS	379
59.4	System Startup	380
59.5	Network Configuration	381
59.6	Firewall	382
59.7	Updating OS	382

59.8	procs: Gone But Not Forgotten	383
59.9	Common Commands	383
59.10	Conclusion	383
60	Frequently Asked Questions About The OS Mailing Lists	385
60.1	Introduction	385
60.2	Mailing List Etiquette	386
60.3	Recurring Topics On The Mailing Lists	388
60.4	What Is A “Bikeshed”?	388
60.5	Acknowledgments	388
61	Introduction to NanoBSD	389
61.1	Introduction to NanoBSD	389
61.2	NanoBSD Howto	389
62	For People New to Both FreeBSD and UNIX	395
62.1	Logging in and Getting Out	395
62.2	Adding A User with Root Privileges	395
62.3	Looking Around	396
62.4	Getting Help and Information	397
62.5	Editing Text	398
62.6	Other Useful Commands	399
62.7	Next Steps	400
62.8	Your Working Environment	400
62.9	Other	401
62.10	Comments Welcome	401
63	Perforce in OS Development	403
63.1	Introduction	403
63.2	Getting Started	403
63.3	Clients	404
63.4	Syncing	405
63.5	Branches	406
63.6	Integrations	406
63.7	Submit	407
63.8	Editing	408
63.9	Changes, Descriptions, and History	408
63.10	Diffs	409
63.11	Adding and Removing Files	409
63.12	Working with Diffs	410
63.13	Renaming Files	410
63.14	Interactions Between OS Subversion and Perforce	411
63.15	Offline Operation	411
63.16	Notes for Google Summer of Code	411
64	Further Reading	413
64.1	Introduction	413
64.2	Terms and conventions	413
64.3	PAM Essentials	415
64.4	PAM Configuration	418
64.5	FreeBSD PAM Modules	420
64.6	PAM Application Programming	423
64.7	PAM Module Programming	423
64.8	Sample PAM Application	423
64.9	Sample PAM Module	423

64.10	Sample PAM Conversation Function	423
64.11	Further Reading	423
64.12	Papers	423
64.13	User Manuals	424
64.14	Related Web pages	424
65	OpenPGP Keys	425
65.1	Officers	425
66	Port Mentor Guidelines	427
66.1	Guideline for Mentor/Mentee Relationships	427
67	Problem Report Handling Guidelines	431
67.1	Introduction	431
67.2	Problem Report Life-cycle	431
67.3	Problem Report State	432
67.4	Types of Problem Reports	432
67.5	Further Reading	436
68	Writing OS Problem Reports	437
68.1	When to Submit a Problem Report	437
68.2	Preparations	438
68.3	Writing the Problem Report	439
68.4	Follow-up	445
68.5	If There Are Problems	445
68.6	Further Reading	445
69	Practical rc.d scripting in BSD	447
69.1	Introduction	447
69.2	Outlining the task	448
69.3	A dummy script	448
69.4	A configurable dummy script	450
69.5	Startup and shutdown of a simple daemon	451
69.6	Startup and shutdown of an advanced daemon	452
69.7	Connecting a script to the rc.d framework	455
69.8	Giving more flexibility to an rc.d script	458
69.9	Further reading	459
70	Using Greylist with OS	461
70.1	Basic Configuration	461
71	OS Release Engineering	465
71.1	Introduction	465
71.2	Release Process	466
71.3	Release Building	469
71.4	Distribution	471
71.5	Extensibility	472
71.6	Lessons Learned from OS 4.4	472
71.7	Future Directions	473
71.8	Acknowledgements	473
72	Remote Installation of the OS Operating System Without a Remote Console	475
72.1	Background	475
72.2	Introduction	475
72.3	Preparation - mfsBSD	476

72.4	Installation of the OS Operating System	477
72.5	ZFS	480
73	Serial and UART Tutorial	481
73.1	The UART: What it is and how it works	481
73.2	Configuring the <code>sio</code> driver	491
73.3	Configuring the <code>cy</code> driver	494
73.4	Configuring the <code>si</code> driver	495
74	OS and Solid State Devices	497
74.1	Solid State Disk Devices	497
74.2	Kernel Options	497
74.3	The <code>rc</code> Subsystem and Read-Only Filesystems	498
74.4	Building a File System from Scratch	498
74.5	System Strategies for Small and Read Only Environments	500
75	The <code>vinum</code> Volume Manager	503
75.1	Synopsis	503
75.2	Access Bottlenecks	503
75.3	Data Integrity	504
75.4	<code>vinum</code> Objects	505
75.5	Some Examples	506
75.6	Object Naming	509
75.7	Configuring <code>vinum</code>	510
75.8	Using <code>vinum</code> for the Root File System	511
76	Design elements of the OS VM system	515
76.1	Introduction	515
76.2	VM Objects	516
76.3	SWAP Layers	518
76.4	When to free a page	518
76.5	Pre-Faulting and Zeroing Optimizations	519
76.6	Page Table Optimizations	520
76.7	Page Coloring	520
76.8	Conclusion	521
76.9	Bonus QA session by Allen Briggs briggs@ninthwonder.com	521
77	OS Architecture Handbook	525
78	FreeBSD Bibliography	527
79	The Design and Implementation of the 4.4BSD Operating System	529
79.1	Design Overview of 4.4BSD	529
79.2	References	545
80	A project model for the FreeBSD Project	547
80.1	Foreword	547
80.2	Overview	548
80.3	Definitions	548
80.4	Organisational structure	549
80.5	Methodology model	550
80.6	Hats	552
80.7	Processes	556
80.8	Tools	562
80.9	Sub-projects	563

80.10 References	564
81 FreeBSD Developers' Handbook	565
82 Installation	567
82.1 Introduction	567
82.2 Documentation and Support	570
82.3 Installation	573
82.4 Hardware Compatibility	575
82.5 Troubleshooting	578
82.6 User Applications	581
82.7 Kernel Configuration	583
82.8 Disks, File Systems, and Boot Loaders	584
82.9 ZFS	589
82.10 System Administration	590
82.11 The X Window System and Virtual Consoles	595
82.12 Networking	599
82.13 Security	602
82.14 PPP	604
82.15 Serial Communications	610
82.16 Miscellaneous Questions	611
82.17 The OS Funnies	614
82.18 Advanced Topics	616
82.19 Acknowledgments	619
83 FreeBSD Documentation Project Primer for New Contributors	621
83.1 Preface	621
84 OS Porter's Handbook	623
85 Indices and tables	629

Contents:

&title;

]>

1.1 What is FreeBSD?

FreeBSD is an operating system for a variety of platforms which focuses on features, speed, and stability. It is derived from BSD, the version of &unix; developed at the University of California, Berkeley. It is developed and maintained by a large community.

1.2 Cutting edge features

FreeBSD offers advanced networking, performance, security and compatibility features today which are still missing in other operating systems, even some of the best commercial ones.

1.3 Powerful Internet solutions

FreeBSD makes an ideal Internet or Intranet server. It provides robust network services under the heaviest loads and uses memory efficiently to maintain good response times for thousands of simultaneous user processes.

1.4 Advanced Embedded Platform

FreeBSD brings advanced network operating system features to appliance and embedded platforms, from higher-end Intel-based appliances to Arm, PowerPC, and shortly MIPS hardware platforms. From mail and web appliances to routers, time servers, and wireless access points, vendors around the world rely on FreeBSD's integrated build and cross-build environments and advanced features as the foundation for their embedded products. And the Berkeley open source license lets them decide how many of their local changes they want to contribute back.

1.5 Run a huge number of applications

With over 24,000 ported libraries and applications, FreeBSD supports applications for desktop, server, appliance, and embedded environments.

1.6 Easy to install

FreeBSD can be installed from a variety of media including CD-ROM, DVD, or directly over the network using FTP or NFS. All you need are these directions.

1.7 FreeBSD is *free*

While you might expect an operating system with these features to sell for a high price, FreeBSD is available free of charge and comes with full source code. If you would like to purchase or download a copy to try out, more information is available.

1.8 Contributing to FreeBSD

It is easy to contribute to FreeBSD. All you need to do is find a part of FreeBSD which you think could be improved and make those changes (carefully and cleanly) and submit that back to the Project by means of a bug report or a committer, if you know one. This could be anything from documentation to artwork to source code. See the Contributing to FreeBSD article for more information.

Even if you are not a programmer, there are other ways to contribute to FreeBSD. The [FreeBSD Foundation](#) is a non-profit organization for which direct contributions are fully tax deductible. Please contact board@FreeBSDFoundation.org for more information or write to: The FreeBSD Foundation, P.O. Box 20247, Boulder, CO 80308, USA.

]>

2.1 Introduction

This page lists teams, groups and individuals within the FreeBSD project with designated project roles and areas of responsibility, along with brief descriptions and contact information.

- Project Management
 - *Core Team*
 - *Documentation Engineering Team*
 - *Port Management Team*
- Release Engineering
 - *Primary Release Engineering Team*
 - *Builders Release Engineering Team*
- Teams
 - *Donations Team*
 - *Marketing Team*
 - *Security Team*
 - *Vendor Relations*
- Secretaries
 - *Core Team Secretary*
 - *Port Management Team Secretary*
 - *Security Team Secretary*
- Internal Administration
 - *Accounts Team*
 - *Backup Administrators*
 - *Bugmeisters*
 - *Cluster Administrators*

- *DNS Administrators*
 - *Forum Administrators*
 - *GitHub Repository Mirror*
 - *Jenkins Continuous Integration Testing Administrators*
 - *FTP/WWW Mirror Site Coordinators*
 - *Perforce Repository Administrators*
 - *Phabricator Code Review Administration*
 - *Postmaster Team*
 - *Subversion Administrators*
 - *Webmaster Team*
-

2.1.1 FreeBSD Core Team <core@FreeBSD.org>

The FreeBSD Core Team constitutes the project’s “Board of Directors”, responsible for deciding the project’s overall goals and direction as well as managing specific areas of the FreeBSD project landscape. The Core Team is elected by the active developers in the project.

- &a.gavin.email;
- &a.theraven.email;
- &a.bapt.email;
- &a.emaste.email;
- &a.gnn.email;
- &a.hrs.email;
- &a.glebius.email;
- &a.rwatson.email;
- &a.peter.email;

2.1.2 FreeBSD Documentation Engineering Team <doceng@FreeBSD.org>

The FreeBSD Documentation Engineering Team is responsible for defining and following up documentation goals for the committers in the Documentation project. The doceng team charter describes the duties and responsibilities of the Documentation Engineering Team in greater detail.

- &a.gjb.email;
- &a.wblock.email;
- &a.blackend.email;
- &a.gabor.email;
- &a.hrs.email;

2.1.3 FreeBSD Port Management Team <portmgr@FreeBSD.org>

The primary responsibility of the FreeBSD Port Management Team is to ensure that the FreeBSD Ports Developer community provides a ports collection that is functional, stable, up-to-date and full-featured. Its secondary responsibility is to coordinate among the committers and developers who work on it. The portmgr team charter describes the duties and responsibilities of the Port Management Team in greater detail.

- &a.mat.email;
 - &a.antoine.email;
 - &a.bapt.email; (Core Team Liaison)
 - &a.bdrewery.email; (Release Engineering Team Liaison)
 - &a.erwin.email; (Cluster Administration Team Liaison)
 - &a.swills.email;
-

2.1.4 Primary Release Engineering Team <re@FreeBSD.org>

The Primary Release Engineering Team is responsible for setting and publishing release schedules for official project releases of FreeBSD, announcing code freezes and maintaining `releng/*` branches, among other things. The release engineering team charter describes the duties and responsibilities of the Primary Release Engineering Team in greater detail.

- &a.gjb.email; (Lead)
- &a.kib.email;
- &a.blackend.email;
- &a.delphij.email;
- &a.remko.email; (Security Team Liaison)
- &a.rodriqc.email;
- &a.hrs.email;
- &a.glebius.email;
- &a.marius.email;
- &a.rwatson.email;

2.1.5 Builders Release Engineering Team <re-builders@FreeBSD.org>

The builders release engineering team is responsible for building and packaging FreeBSD releases on the various supported platforms.

- &a.marcel.email;
 - &a.nyan.email;
 - &a.nwhitehorn.email;
-

2.1.6 Donations Team <donations@FreeBSD.org>

The FreeBSD Donations Team is responsible for responding to donations offers, establishing donation guidelines and procedures, and coordinating donation offers with the FreeBSD developer community. A more detailed description of the duties of the Donations Team is available on the FreeBSD Donations Liaison page.

- &a.gjb.email;
- &a.gahr.email;
- &a.pgollucci.email;
- &a.skreuzer.email;
- &a.obrien.email;
- &a.trhodes.email;
- &a.ds.email;
- &a.rwatson.email;

2.1.7 Marketing Team <marketing@FreeBSD.org>

Press contact, marketing, interviews, information.

- Steven Beedle <steven@zna.com>
- Anne Dickison <anne@freebsd.foundation.org>
- &a.deb.email;
- &a.dru.email;
- &a.gnn.email;
- &a.mwluccas.email;
- &a.imp.email;
- &a.kmoore.email;
- &a.murray.email;
- &a.matt.email;
- Jeremy C. Reed <reed@reedmedia.net>
- &a.randi.email;
- &a.rwatson.email;

2.1.8 Security Team <secteam@FreeBSD.org>

The FreeBSD Security Team (headed by the Security Officer) is responsible for keeping the community aware of bugs, exploits and security risks affecting the FreeBSD src and ports trees, and to promote and distribute information needed to safely run FreeBSD systems. Furthermore, it is responsible for resolving software bugs affecting the security of FreeBSD and issuing security advisories. The FreeBSD Security Officer Charter describes the duties and responsibilities of the Security Officer in greater detail.

- &a.benl.email;
- &a.cperciva.email; (Officer Emeritus)

- &a.csjp.email;
- &a.delphij.email; (Officer)
- &a.des.email; (Officer Emeritus)
- &a.gavin.email; (Core Team Liaison)
- &a.gjb.email; (Cluster Administrators Team Liaison)
- &a.glebius.email; (Officer Deputy)
- &a.imp.email; (Officer Emeritus)
- &a.jonathan.email;
- &a.philip.email;
- &a.qingli.email;
- &a.remko.email; (Secretary)
- &a.rwatson.email; (Officer Emeritus)
- &a.simon.email; (Officer Emeritus)
- &a.stas.email;
- &a.trasz.email;

2.1.9 Vendor Relations <vendor-relations@FreeBSD.org>

Vendor Relations is responsible for handling email from hardware and software vendors. Email sent to Vendor Relations is forwarded to the &os; Core Team in addition to the &os; Foundation.

2.1.10 Core Team Secretary <core-secretary@FreeBSD.org>

The &os; Core Team Secretary is a non-voting member of the Core Team, responsible for documenting the work done by core, keeping track of the core agenda, direct contact with non-core members sending mail to core and to be an the interface to the admin team for committer/account approval. The Core Team Secretary is also responsible for writing and sending out monthly status reports to the &os; Developer community, containing a summary of core's latest decisions and actions.

- &a.matthew.email;

2.1.11 Port Management Team Secretary <portmgr-secretary@FreeBSD.org>

The FreeBSD Port Management Team Secretary is a non-voting member of the Port Management Team, responsible for documenting the work done by portmgr, keeping track of voting procedures, and to be an interface to the other teams, especially the admin and Core teams. The Port Management Team Secretary is also responsible for writing and sending out monthly status reports to the FreeBSD Developer community, containing a summary of portmgr's latest decisions and actions.

- &a.culot.email;

2.1.12 Security Team Secretary <secteam-secretary@FreeBSD.org>

The FreeBSD Security Team Secretary will make sure someone responds to incoming emails towards the Security Team. He will acknowledge receipt and keep track of the progress within the Security Team. If needed the Secretary will contact members of the Security Team to let them provide an update on ongoing items. Currently the Security Team Secretary does not handle Security Officer Team items.

- &a.remko.email;
-

2.1.13 Accounts Team <accounts@>

The Accounts Team is responsible for setting up accounts for new committers in the project. Requests for new accounts will not be acted upon without the proper approval from the appropriate entity.

Email sent to the Accounts Team is currently forwarded to *Cluster Administration*.

2.1.14 Backups Administrators <backups@>

The Backups Administrators handle all backups on the FreeBSD cluster.

Email sent to the Backups Team is currently forwarded to *Cluster Administration*.

2.1.15 Bugmeisters <bugmeister@FreeBSD.org>

The Bugmeisters are responsible for ensuring that the maintenance database is in working order, that the entries are correctly categorised and that there are no invalid entries. They are also responsible for the problem report group.

- &a.eadler.email;
- &a.mva.email;
- &a.gavin.email;
- &a.koobs.email;
- &a.gonzo.email;

2.1.16 Cluster Administrators <admins@>

The Cluster Administrators consists of the people responsible for administrating the machines that the project relies on for its distributed work and communication to be synchronised. It consists mainly of those people who have physical access to the servers. Issues concerning the projects infrastructure or setting up new machines should be directed to the cluster administrators. This team is led by the lead cluster administrator whose duties and responsibilities are described in the cluster administration charter in greater detail.

- &a.dhw.email;
- &a.gavin.email;
- &a.gjb.email;
- &a.hiren.email;
- &a.peter.email; (Lead)
- &a.sbruno.email;

- &a.simon.email;
- &a.zi.email;

2.1.17 DNS Administrators <dnsadm@>

The DNS Administrators are responsible for managing DNS and related services.

E-mail to the DNS Administrators is currently forwarded to *Cluster Administration*.

2.1.18 &os; Forum Administrators <forum-admins@FreeBSD.org>

The Forum Administrators maintain the &os; Project's web forum site, located at <https://forums.freebsd.org/> and lead the group of moderators who work to ensure the relevance and quality of forum content.

- &a.brd.email;
- &a.danger.email;
- &a.dutchdaemon.email;
- &a.lme.email;
- &a.wblock.email;

2.1.19 Repository Automated Mirroring to GitHub Coordinators <github-automation@FreeBSD.org>

The GitHub Automation team oversees the export of &os; source code repository content to the read-only repository instances on GitHub

- &a.koobs.email;
- &a.mva.email;
- &a.robak.email;
- &a.rodrihc.email;
- &a.uqs.email;

2.1.20 Jenkins Continuous Integration Testing Administrators <jenkins-admin@FreeBSD.org>

The Jenkins Administrators run continuous build and integration tests against the HEAD revision of the &os; Source Code.

- Ahmed Kamal <email.ahmedkamal@gmail.com>
- &a.jmmv.email;
- &a.lwhsu.email;
- &a.rodrihc.email;
- &a.swills.email;

2.1.21 FTP/WWW Mirror Site Coordinators <mirror-admin@FreeBSD.org>

The FTP/WWW Mirror Site Coordinators coordinate all the FTP/WWW mirror site administrators to ensure that they are distributing current versions of the software, that they have the capacity to update themselves when major updates are in progress, and making it easy for the general public to find their closest FTP/WWW mirror.

E-mail to the Mirror Site Coordinators is currently forwarded to the *Cluster Administration* team with the addition of:

- &a.kuriyama.email;

2.1.22 Perforce Repository Administrators <perforce-admin@FreeBSD.org>

The Perforce Repository Administrators are responsible for administrating the FreeBSD perforce source repository and setting up new perforce accounts. All requests concerning new perforce accounts for non-committers should be directed to the perforce administrators.

- &a.gibbs.email;
- &a.gordon.email;
- &a.rwatson.email;
- &a.peter.email;

2.1.23 Phabricator Code Review Application Administrators <phabric-admin@FreeBSD.org;>

The Phabricator Administrators are responsible for maintaining the &os;'s instance of the Phabricator on-line code review tool located at <https://reviews.freebsd.org/>

- &a.eadler.email;
- &a.emaste.email;
- &a.lwhsu.email;
- &a.mat.email;
- &a.robak.email;
- &a.rpaulo.email;

2.1.24 Postmaster Team <postmaster@FreeBSD.org>

The Postmaster Team is responsible for mail being correctly delivered to the committers' email address, ensuring that the mailing lists work, and should take measures against possible disruptions of project mail services, such as having troll-, spam- and virus-filters.

- &a.flo.email; (Lead)
- &a.sahil.email;
- &a.dhw.email;
- &a.jadawin.email;
- &a.brd.email;

2.1.25 Subversion Administrators <svnadm@>

The FreeBSD Subversion team is responsible for maintaining the health of the Subversion Repositories.

Email to the Subversion Administration team is currently forwarded to *Cluster Administration*.

2.1.26 Webmaster Team <webmaster@FreeBSD.org>

The FreeBSD Webmaster Team is appointed by &os; Documentation Engineering Team, and responsible for keeping the main FreeBSD web sites up and running. This means web server configuration, CGI scripts, fulltext and mailing list search. Anything web related, technical stuff belongs to the scope of the Webmaster Team, excluding bugs in the documentation.

Email to the Webmaster Team is currently forwarded to the *Documentation Engineering* team with the addition of:

- &a.wosch.email;

&title;

]>

3.1 Experience the possibilities with FreeBSD

FreeBSD can handle nearly any task you would expect of a &unix; workstation, as well as many you might not expect:

3.2 FreeBSD is a true open system with full source code.

There is no doubt that so-called open systems are *the* requirement for today's computing applications. But no commercial vendor-supplied solution is more open than one which includes full source code to the entire operating system, including the kernel and all of the system daemons, programs, and utilities. You can modify any part of FreeBSD to suit your personal, organizational, or corporate needs.

With its generous licensing policy, you can use FreeBSD as the basis for any number of free *or commercial* applications.

3.3 FreeBSD runs thousands of applications.

Because FreeBSD is based on 4.4BSD, an industry-standard version of UNIX, it is easy to compile and run programs. FreeBSD also includes an extensive packages collection and ports collection that bring precompiled and easy-to-build software right to your desktop or enterprise server. There is also a growing number of commercial applications written for FreeBSD.

Here are some examples of the environments in which FreeBSD is used:

- **Internet services.** Many Internet Service Providers (ISPs) find FreeBSD ideal, running WWW, Usenet news, FTP, Email, and other services. Ready-to-run software like the [nginx](#) or [Apache](#) web server or the [ProFTPD](#) or [vsftpd](#) FTP server make it easy to set up a business or community-centered ISP. Of course, with FreeBSD's unbeatable networking, your users will enjoy high speed, reliable services.
- **X Window workstation.** From an inexpensive X terminal to an advanced X display, FreeBSD works quite well. Free X software ([X.Org](#)TM) comes with the system. [nVidia](#) offers native drivers for their high-performance graphics hardware, and the industry standard [Motif](#)® and [OpenGL](#)® libraries are supported. The [Xfce](#) and [LXDE](#) products provide a desktop environment. The [KDE](#) and [GNOME](#) desktop environments also enjoy full support and provide office suite functionality, with further good functionality available in the [LibreOffice](#), [OpenOffice.Org](#) and [TextMaker](#) products.

- **Networking.** From packet filtering to routing to name service, FreeBSD can turn any PC into a Internet firewall, email host, print server, PC/NFS server, and more.
- **Software development.** A suite of development tools comes with FreeBSD, including the GNU C/C++ compiler and debugger. The LLVM-based clang suite is also provided and will eventually replace the GNU suite. &java; and Tcl/Tk development are also possible for example, and more esoteric programming languages like Icon work just fine, too. And FreeBSD's shared libraries have always been easy to make and use. You can also choose from a wide range of popular and powerful editors, such as XEmacs and Vim.
- **Net surfing.** A real UNIX workstation makes a great Internet surfboard. FreeBSD versions of [Chromium](#), [Firefox](#) and [Opera](#) are available for serious web users. Surf the web, publish your own web pages, read Usenet news, and send and receive email with a FreeBSD system on your desktop.
- **Education and research.** FreeBSD makes an excellent research platform because it includes complete source code. Students and researchers of operating systems or other computer science fields can benefit greatly from such an open and well-documented system.
- **And much more.** Accounting, action games, MIS databases, scientific visualization, video conferencing, Internet relay chat (IRC), home automation, multiuser dungeons, bulletin board systems, image scanning, and more are all real uses for FreeBSD today.

3.4 FreeBSD is an operating system that will grow with your needs.

Though FreeBSD is free software, it is also *user supported* software. Any questions you have can be posted to hundreds of FreeBSD developers and users simply by e-mailing the freebsd-questions@FreeBSD.org mailing list.

FreeBSD also has a worldwide group of programmers and writers who fix bugs, add new features and document the system. Support for new devices or special features is an almost constant development process, and the team keeps a special eye out for problems which affect system stability. FreeBSD users are quite proud of not only how fast but how reliable their systems are.

3.5 What experts have to say . . .

“FreeBSD handles [our] heavy load quite well and it is nothing short of amazing. Salutations to the FreeBSD team.”

—Mark Hittinger, administrator of WinNet Communications, Inc.

&title;

]>

- *BSD Daemon*
- *“Powered by FreeBSD” Logos*
- *Old advertisement banners*
- *Graphics use*

This page contains miscellaneous FreeBSD “art”. Suggestions for additions can be sent to www@FreeBSD.org. Please note the *usage policy* for these graphics.

4.1 BSD Daemon



Created by &a.phk;

Source: /usr/share/examples/BSD_daemon/ on FreeBSD systems.





4.2 “Powered by FreeBSD” Logos



4.3 Old Advertisement Banners

4.4 Graphics Use

The “Powered by FreeBSD” logos above may be downloaded and displayed on personal or commercial home pages served by FreeBSD machines. Use of this logo or the likeness of the BSD Daemons for profitable gain requires the consent of [Brian Tao](#) (creator of the “power” logo) and [Marshall Kirk McKusick](#) (copyright holder for the BSD Daemon image).

&a.phk;’s rendering of the BSD Daemon is released under “THE BEER-WARE LICENSE”. See the [README](#) for more information.

4.5 Trademarks

The [FreeBSD Foundation](#) holds several FreeBSD related trademarks (among them the trademark for the term “FreeBSD” itself). For more information about these trademarks read the [FreeBSD Trademark Usage Terms and Conditions](#).

&title;

]>

The power, flexibility, and reliability of FreeBSD attract a wide variety of users and vendors. Below is a list of vendors offering commercial products, services, and/or consulting for FreeBSD.

- **Consulting Services** Whether you are just starting out with FreeBSD, or need to complete a large project, hiring consultants might be your answer. You can see the alphabetical list, or, if you prefer, view it by categories.
- **Hardware Vendors** Need specialized tools, looking to buy a new desktop, or to fill a cage with rack mount servers that come with FreeBSD pre-installed? These companies may have what you need!
- **Internet Service Providers** If you need webspace on a FreeBSD-based system or want your company connected to the internet, these companies may offer what you need!
- **Miscellaneous Vendors** Books and accessories you just cannot live without!
- **Software Vendors** From audio players, network drivers, to commercial databases, FreeBSD has a wide variety of industrial strength software available. You can see the alphabetical list, or, if you prefer, view it by categories.

If your company supports a FreeBSD related product, service, consulting, or support that should be added to this page, please fill out a [problem report](#) in category Documentation->Website. Submissions should contain a medium-sized paragraph in length, describing your company. Please note that the inclusion of vendors in our list does not signify our endorsement of their products or services by the FreeBSD Project.

&title;

]>

6.1 Subversion

Subversion (SVN for short) is the tool the &os; Project uses for keeping its sources under control. Every change (with an accompanying log message explaining its purpose) is stored. It can be easily viewed from the web interface mentioned below.

In June 2008, development of the base system migrated from CVS to Subversion. The **web interface** is available for browsing the repository.

In May 2012, the FreeBSD Documentation Project moved from CVS to Subversion. There is a **web interface** available for browsing the contents of the FreeBSD Documentation Project SVN repository.

In July 2012, the FreeBSD Ports tree moved from CVS to Subversion. There is a **web interface** for browsing the repository.

6.2 Other options

CTM if you are looking for very low overhead, batch-mode access (basically, patches through email).

&title;

l>



A wide variety of documentation is available for FreeBSD, on this web site, on other web sites, and available over the counter.

]>

8.1 &os; offers many unique features.

No matter what the application, an operating system should take advantage of every resource available. &os;’s focus on performance, networking, and storage combines with ease of system administration and comprehensive documentation to realize the full potential of any computer.

8.1.1 A complete operating system based on 4.4BSD.

&os;’s distinguished roots derive from the **BSD** software releases from the Computer Systems Research Group at the University of California, Berkeley. Over twenty years of work have been put into enhancing &os;, adding industry-leading scalability, network performance, management tools, file systems, and security features. As a result, &os; may be found across the Internet, in the operating system of core router products, running root name servers, hosting major web sites, and as the foundation for widely used desktop operating systems. This is only possible because of the diverse and world-wide membership of the volunteer &os; Project.

&os; 10.X introduced many new features and replaces many legacy tools with updated versions.

- **bhyve:** A new BSD licensed, legacy-free hypervisor has been imported to the &os; base system. It is currently able to run all supported versions of &os;, and with the help of the grub-bhyve port, OpenBSD and Linux.
- **KMS And New drm2 Video Drivers:** The new drm2 driver provides support for AMD GPUs up to the Radeon HD 6000 series and provides partial support for the Radeon HD 7000 family. &os; now also supports Kernel Mode Setting for AMD and Intel GPUs.
- **Capsicum Enabled By Default:** Capsicum has been enabled in the kernel by default, allowing sandboxing of several programs that work within the “capabilities mode”, such as:
 - tcpdump
 - dhclient
 - hast
 - rwhod
 - kdump
- **New Binary Packaging System:** &os; now uses pkg, a vastly improved package management system that supports multiple repositories, signed packages, and safe upgrades. The improved system is combined with

more frequent official package builds for all supported platforms and a new stable branch of the ports tree for better long term support.

- **Unmapped I/O:** The newly implemented concept of unmapped VMIO buffers eliminates the need to perform costly TLB shootdowns for buffer creation and reuse, reducing system CPU time by up to 25-30% on large SMP machines under heavy I/O load.

&os; 9.X brought many new features and performance enhancements with a special focus on desktop support and security.

- **OpenZFS:** &os; 9.2 includes OpenZFS v5000 (Feature Flags), including the feature flags:

- `async_destroy`
- `empty_bpobj`
- `lz4_compress`

which allow ZFS destroy operations to happen in the background, make snapshots consume less disk space, and offers a better compression algorithm for compressed datasets.

- **Capsicum Capability Mode:** Capsicum is a set of features for sandboxing support, using a capability model in which the capabilities are file descriptors. Two new kernel options `CAPABILITIES` and `CAPABILITY_MODE` have been added to the GENERIC kernel.
- **Hhook:** (Helper Hook) and `khel(9)` (Kernel Helpers) KPIs have been implemented. These are a superset of `pfil(9)` framework for more general use in the kernel. The `hhook(9)` KPI provides a way for kernel subsystems to export hook points that `khel(9)` modules can hook to provide enhanced or new functionality to the kernel. The `khel(9)` KPI provides a framework for managing `khel(9)` modules, which indirectly use the `hhook(9)` KPI to register their hook functions with hook points of interest within the kernel. These allow a structured way to dynamically extend the kernel at runtime in an ABI preserving manner.
- **Accounting API** has been implemented. It can keep per-process, per-jail, and per-login class resource accounting information. Note that this is neither built nor installed by default. To build and install this, specify the option `RACCT` in the kernel configuration file and rebuild the base system as described in the &os; Handbook.
- **Resource-limiting API** has been implemented. It works in conjunction with the `RACCT` resource accounting implementation and takes user-configurable actions based on the set of rules it maintains and the current resource usage. The `rcctl(8)` utility has been added to manage the rules in userland. Note that this is neither built nor installed by default.
- **USB** subsystem now supports USB packet filter. This allows capturing packets which go through each USB host. The architecture of the packet filter is similar to that of `bpf`. The userland program `usbdump(8)` has been added.
- **Infiniband support:** OFED (OpenFabrics Enterprise Distribution) version 1.5.3 has been imported into the base system.
- **TCP/IP network** stack now supports the `mod_cc(9)` pluggable congestion control framework. This allows TCP congestion control algorithms to be implemented as dynamically loadable kernel modules. Many kernel modules are available: `cc_chd(4)` for the CAIA-Hamilton-Delay algorithm, `cc_cubic(4)` for the CUBIC algorithm, `cc_hd(4)` for the Hamilton-Delay algorithm, `cc_htcp(4)` for the H-TCP algorithm, `cc_newreno(4)` for the NewReno algorithm, and `cc_vegas(4)` for the Vegas algorithm. The default algorithm can be set by a new `sysctl(8)` variable `net.inet.tcp.cc.algorithm`.
- **SU+J:** &os;'s Fast File System now supports soft updates with journaling. It introduces an intent log into a softupdates-enabled file system which eliminates the need for background `fsck(8)` even on unclean shutdowns.

&os; includes a number of other great features:

- **Firewalls:** The base system includes IPFW and IPFilter, as well as a modified version of the popular pf with improved SMP performance. IPFW also includes the dummynet feature, allowing network administrators to simulate adverse network conditions, including latency, jitter, packet loss and limited bandwidth.
- **Jails** are a light-weight alternative to virtualization. Allowing processes to be restricted to a namespace with access only to the file systems and network addresses assigned to that namespace. Jails are also Hierarchical, allowing jails-within-jails.
- **Linux emulation** provides a system call translation layer that allows unmodified Linux binaries to be run on &os; systems.
- **DTrace** provides a comprehensive framework for tracing and troubleshooting kernel and application performance issues while under live load.
- **The Ports Collection** is a set of more than 23,000 third party applications that can be easily installed and run on &os;. The ports architecture also allows for easy customization of the compile time options of many of the applications.
- **Network Virtualization:** A container (“vimage”) has been implemented, extending the &os; kernel to maintain multiple independent instances of networking state. Vimage facilities can be used independently to create fully virtualized network topologies, and jail(8) can directly take advantage of a fully virtualized network stack.

&title;

|>

- *Usage Guideline*
- *Resource*
- *Sample*

9.1 Usage Guideline

FreeBSD is a registered trademark of The FreeBSD Foundation. The FreeBSD logo and The Power to Serve are trademarks of The FreeBSD Foundation.

All images listed under the heading “Resource” are available for use under license from The FreeBSD Foundation.

For more information on how to obtain permission to use the logo, please refer to the FreeBSD Logo Usage Guidelines at [The FreeBSD Foundation](#).

9.2 Resource

9.2.1 Standard Logo (fullcolor)



9.2.2 Standard Logo (fullcolor for dark background)



9.2.3 Standard Logo (black and white)



9.2.4 Vector formats

Format: Adobe(r) Illustrator(r), SVG



9.3 Sample

NOTE: “freeBSD” text in these images were created based on draft version of logo contest. Correct version of this text should be rendered by one black color, not two colors and first “f” character should be spelled capitalized as “F”.


9.3.1 CD/DVD package



9.3.2 Postcard



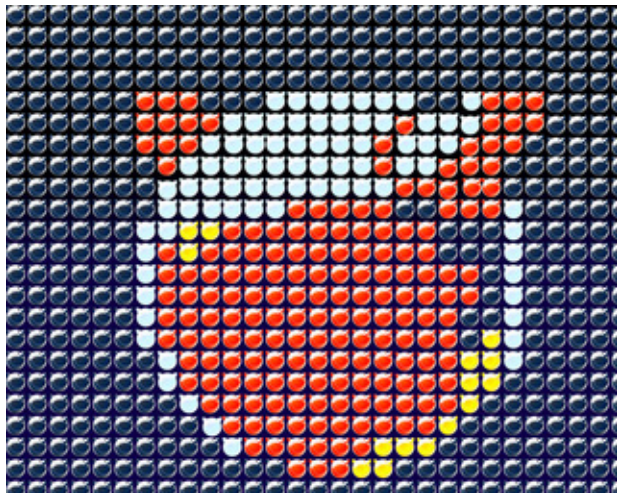


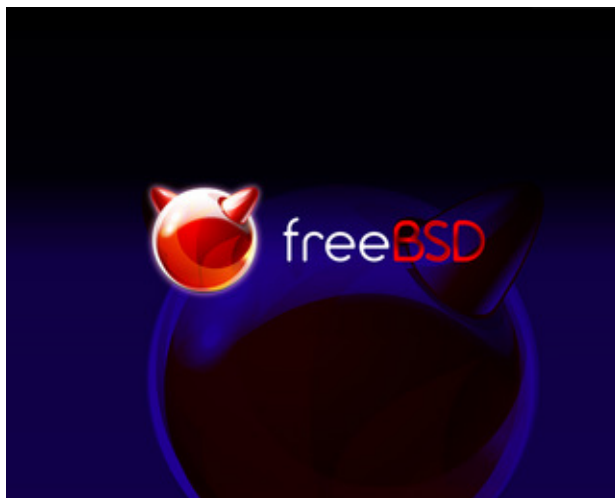
choose your way with  freeBSD



9.3.3 Wallpaper

Here are sample wallpapers.





&title;

]>

10.1 Questions about FreeBSD...

Questions regarding FreeBSD should be addressed to the FreeBSD Questions mailing list, freebsd-questions@FreeBSD.org.

Mailing lists are the primary support channel for FreeBSD users, with numerous mailing lists covering different topic areas. Several non-English mailing lists are also available.

10.2 Questions about the contents of this WWW server...

Questions or suggestions about our documentation (Handbook, FAQ, Books & Articles) should be addressed to the FreeBSD Documentation Project mailing list, freebsd-doc@FreeBSD.org.

10.3 Snail mail, phone and fax

For CDROM orders: [The FreeBSD Mall](#)

For commercial support: [The FreeBSD Mall](#)

10.4 Who Is Responsible for What

Marketing, Bugmeister, Security Officer, Postmaster, Webmaster etc.

]>

This page contains presentations, white papers, and other marketing materials for FreeBSD.

11.1 White Papers

Thinking of using FreeBSD in a project? Finding it hard to convince your boss, the CTO, the CEO? Read through these real life examples of FreeBSD successes with shipping products, then give them to the decision makers at your company.

- Explaining BSD
- Building Products with FreeBSD

11.2 Presentations

- 28 April 2005, [FreeBSD in the Enterprise, an Introduction for Linux Users](#), Murray Stokely (Open Source Forum, Moscow)
- 18 September 2002, [FreeBSD Release Engineering](#), Murray Stokely (Japan Unix Society)
- 10 June 2001, [FreeBSD for Linux Users](#), Nik Clayton

11.3 Flyers

- What is FreeBSD? ([PDF](#) | [PostScript](#))
- BSD Success Stories (27 pages) ([PDF](#)), O'Reilly.

Queue Portrait: Robert Watson

https://queue.acm.org/detail_video.cfm?id=2382552 George Neville-Neil, Queue's Kode Vicious, interviews Robert Watson to learn about Capsicum and other exciting research projects at Cambridge. 2012 interview research Robert Watson George Neville-Neil

BSDCan-2012 Photos - Friday

http://www.db.net/gallery/BSDCan/BSDCan_2012_day_1/ Photos taken during the Conference on Friday at BSDCan 2012 in Ottawa by Diane Bruce. 2012 bsdcan bsdcan2012 photos diane bruce

BSDCan-2012 Photos - Saturday

http://www.db.net/gallery/BSDCan/BSDCan_2012_day_2/ Photos taken during both the DevSummit and Conference on Saturday at BSDCan 2012 in Ottawa by Diane Bruce. 2012 bsdcan bsdcan2012 photos diane bruce

BSDCan-2012 Photos - Saturday

<https://plus.google.com/photos/117117406211143183805/albums/5742469737904181073?banner=pwa&authkey=COK7-ca5-N-TQ> Photos taken during both the DevSummit and Conference on Saturday at BSDCan 2012 in Ottawa by Benedict Reuschling. 2012 bsdcan bsdcan2012 photos benedict reuschling

BSDCan-2012 - Michael Dexter - An applied survey of BSD multiplicity and virtualization strategies from chroot to BHyVe

Ever since the University of California, Berkeley CSRG implemented the chroot(8) command and system call in its BSD operating system in 1982, the community-developed BSD Unix derivatives have set the standard for the introduction of plurality to the conventionally-singular Unix computing model. Today's system operators and developers have an array of BSD-licensed multiplicity strategies at their disposal that offer various degrees of both isolation and virtualization when introducing plurality. This paper will survey current and experimental BSD multiplicity strategies including chroot, FreeBSD jail, NetBSD/Xen, Amazon EC2, compatlinux, GXemul and SIMH, plus experimental strategies such as FreeBSD BHyVe, compatmach, Usermode NetBSD, Dragonfly BSD vkernel, OpenBSD sysjail and NetBSD mult. As an applied survey, this paper will both categorize each multiplicity strategy by the Unix environment to which it introduces plurality and demonstrate the usage of the utilities relating to each solution. <http://www.bsdcan.org/2012/schedule/events/291.en.html> 2012 bsdcan bsdcan2012 papers michael dexter <http://www.bsdcan.org/2011/schedule/events/291en.html> html html

BSDCan-2012 - Kirk McKusick - An Overview of Locking in the FreeBSD Kernel

The FreeBSD kernel uses seven different types of locks to ensure proper access to the resources that it manages. This talk describes the hierarchy of these locks from the low-level and simple to the high-level and full-featured. The functionality of each type of lock is described along with the problem domain for which it is intended. The talk concludes by describing the witness system within the FreeBSD kernel that tracks the usage of all the locks in the system and reports any possible deadlocks that might occur because of improper acquisition ordering of locks. <http://www.bsdcan.org/2012/schedule/events/306.en.html> 2012 bsdcan bsdcan2012 papers kirk mckusick http://www.bsdcan.org/2012/schedule/attachments/195_locking.pdf 27 Kb Slides pdf

BSDCan-2012 - Pawel Jakub Dawidek - auditdistd - Secure and reliable distribution of audit trail files

Security Event Audit is a facility to provide fine-grained, configurable logging of security-relevant events. Audit events are stored in trail files that can be used for postmortem analysis in case of system compromise. Once the system is compromised, an attacker has access to audit trail files and can modify or delete them. The auditdistd daemon's role is to distribute audit trail files to a remote system in a secure and reliable way. <http://www.bsdcan.org/2012/schedule/events/335.en.html> 2012 bsdcan bsdcan2012 papers pawel jakub dawidek http://www.bsdcan.org/2012/schedule/attachments/217_Auditdistd%20slides PDF =265.6 Kb 50 pages pdf

BSDCan-2012 - Ivan Voras - Bullet Cache - Balancing speed and usability in a cache server

Bullet Cache is an in-memory cache server inspired by memcached, but with a twist: a powerful record tagging and bulk query facility, configurable multithreading models and a dump / cache prewarm option. This talk will have two parts: a technical description of Bullet Cache's implementation with focus on programming techniques and optimizations, and a description of usage scenarios with the focus on how it can help real-world applications (not limited to Web applications). <http://www.bsdcan.org/2012/schedule/events/339.en.html> 2012 bsdcan bsdcan2012 papers ivan voras http://www.bsdcan.org/2012/schedule/attachments/198_BSDCan2012.pdf PDF =661.3 Kb 40 pages pdf

BSDCan-2012 - Benedict Reuschling - Google Code-In and FreeBSD

A summary of FreeBSD's participation in the 2011 contest. <http://www.bsdcan.org/2012/schedule/events/354.en.html> 2012 bsdcan bsdcan2012 papers benedict reuschling http://www.bsdcan.org/2012/schedule/attachments/213_FreeBSDGCIN2011Summary.pdf PDF =82 Kb 16 pages pdf

BSDCan-2012 Photos - Developers summit and conference

<http://gallery.keltia.net/v/voyages/conferences/bsdcant-2012/devsummit/> Photos taken during both the DevSummit and Conference on Saturday at BSDCan 2012 in Ottawa by Ollivier Robert. 2012 bsdcant bsdcant2012 photos ollivier robert

BSDCan-2011 - Brooks Davis - Improving System Management with ZFS

The Zetabyte File System (ZFS) is a modern file system which combines traditional file system features like a POSIX file system interface with RAID and volume management functionality. Features such as snapshot management and file share management are all managed within the ZFS interface. This management interface provides a number of opportunities to simplify system management. In the Technical Computing Services Sub-division of The Aerospace Corporation we are taking advantage of these features in a number of different ways. This talk presents some of the more interesting ones. <http://www.bsdcant.org/2011/schedule/events/233.en.html> 2011 bsdcant bsdcant2011 papers brooks davis http://www.bsdcant.org/2011/schedule/attachments/149_abstract.pdf PDF =40.4 Kb 2 pages pdf

BSDCan-2011 Photos - Saturday

Photos taken during the Conference on Saturday at BSDCan 2011 in Ottawa by Diane Bruce. http://www.db.net/gallery/BSDCan/BSDCan_2011_day_2/ 2011 bsdcant bsdcant2011 photos diane bruce

BSDCan-2011 Photos - Friday

Photos taken during the Conference on Friday at BSDCan 2011 in Ottawa by Diane Bruce. http://www.db.net/gallery/BSDCan/BSDCan_2011_day_1/ 2011 bsdcant bsdcant2011 photos diane bruce

BSDCan-2010 Photos - Saturday

Photos taken during the Conference on Saturday at BSDCan 2010 in Ottawa by Diane Bruce. http://www.db.net/gallery/BSDCan/BSDCan_2010_day_2/ 2010 bsdcant bsdcant2010 photos diane bruce

BSDCan-2010 Photos - Friday

Photos taken during the Conference on Friday at BSDCan 2010 in Ottawa by Diane Bruce. http://www.db.net/gallery/BSDCan/BSDCan_2010_day_1/ 2010 bsdcant bsdcant2010 photos diane bruce

BSDCan-2010 - Kris Moore - The PBI format re-implemented for FreeBSD and PC-BSD

The PBI format (Push Button Installer) has been the default package management system for PC-BSD going on 5+ years now. However as we looked to the future it became apparent that it was greatly needing an overhaul to both improve its functionality, and expand its usage outside the scope of just PC-BSD. Among the areas needing improvement were how it dealt with identical libraries between applications, the heavy requirements from being implemented in QT/KDE, and lack of a digital verification mechanism. <http://www.bsdcant.org/2011/schedule/events/215.en.html> 2010 bsdcant bsdcant2010 papers kris moore <http://www.bsdcant.org/2011/schedule/events/215.en.html> html html

A Few FreeBSD Core Team Members

Interview with a few of the FreeBSD Core Team members at BSDCan 2009: Robert Watson, Brooks Davis, Hiroki Sato, Philip Paeps, and George V. Neville-Neil. We talk about the recent 7.2 release, and what is coming for 8. <http://bsd.talk.blogspot.com/2009/05/bsd.talk173-few-freebsd-core-team.html> bsd.talk interview bsdcant freebsd core team robert watson brooks davis hiroki sato philip paeps george neville-neil <http://cisx1.uma.maine.edu/~wbackman/bsd.talk/> bsd.talk173.mp3 18 Mb 38 minutes MP3 version mp3 bsd.talk173.ogg 38 minutes Ogg version ogg

BSDCan 2009 with Dan Langille

Interview with Dan Langille. We talk about BSDCan 2009. More information at <http://www.bsdcant.org>. <http://bsd.talk.blogspot.com/2009/04/bsd.talk172-bsdcant-2009-with-dan.html> bsd.talk interview bsdcant dan langille <http://cisx1.uma.maine.edu/~wbackman/bsd.talk/> bsd.talk172.mp3 6 Mb 13 minutes MP3 version mp3 bsd.talk172.ogg 13 minutes Ogg version ogg

Andrew Doran from the NetBSD Project

Interview with Andrew Doran from the NetBSD Project. We talk about the upcoming 5.0 release. <http://bsd.talk.blogspot.com/2009/03/bsd.talk171-andrew-doran-from-netbsd.html> bsd.talk interview netbsd andrew do-

ran <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk171.mp3> 10 Mb 22 minutes MP3 version mp3 bsdtalk171.ogg 22 minutes Ogg version ogg

Marshall Kirk McKusick at DCBSDCon

A recording of Marshall Kirk McKusick's talk "A Narrative History of BSD" at DCBSDCon this past weekend.

You can get a much more complete history here: <http://www.mckusick.com/history/index.html>
<http://bsdtalk.blogspot.com/2009/02/bsdtalk170-marshall-kirk-mckusick-at.html> bsdtalk presentation bsdtalk history kirk mckusick <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk170.mp3> 26 Mb 55 minutes MP3 version mp3 bsdtalk170.ogg 55 minutes Ogg version ogg

Justin Sherrill of the DragonFlyBSD Digest

Interview with Justin Sherrill of the DragonFlyBSD Digest, which can be found at <http://www.shiningsilence.com/dbsdlog/> <http://bsdtalk.blogspot.com/2009/01/bsdtalk169-justin-sherrill-of.html> bsdtalk interview dragonflybsd justin sherrill <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk169.mp3> 10 Mb 22 minutes MP3 version mp3 bsdtalk169.ogg 22 minutes Ogg version ogg

Michael Lauth from iXsystems

Interview with Michael Lauth, CEO of iXsystems. We talk about his experiences with running a business using BSD. <http://bsdtalk.blogspot.com/2008/12/bsdtalk168-michael-lauth-from-ixsystems.html> bsdtalk interview ixsystems michael lauth <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk168.mp3> 8 Mb 17 minutes MP3 version mp3 bsdtalk168.ogg 17 minutes Ogg version ogg

DCBSDCon with Jason Dixon

I speak with Jason Dixon about DCBSDCon, which will take place in February 2009. For more info see www.dcbbsdcon.org <http://bsdtalk.blogspot.com/2008/12/bsdtalk167-dcbbsdcon-with-jason-dixon.html> bsdtalk interview dcbbsdcon dcbbsdcon2009 jason dixon <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk167.mp3> 5 Mb 10 minutes MP3 version mp3 bsdtalk167.ogg 10 minutes Ogg version ogg

Asterisk Open Source Community Director John Todd

An interview with Asterisk Open Source Community Director John Todd, who also happens to be a user of BSD. We talk about Asterisk on BSD, and his choice of OpenBSD for his systems. <http://bsdtalk.blogspot.com/2008/11/bsdtalk166-asterisk-open-source.html> bsdtalk interview john todd asterisk openbsd <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk166.mp3> 11 Mb 23 minutes MP3 version mp3 bsdtalk166.ogg 23 minutes Ogg version ogg

Julian Elischer

An interview with Julian Elischer at MeetBSD in California. We talk about his early days with BSD and his work using BSD at various companies. He is currently with IronPort, which was bought by Cisco. <http://bsdtalk.blogspot.com/2008/11/bsdtalk165-julian-elischer.html> bsdtalk interview julian elischer ironport <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk165.mp3> 16 Mb 35 minutes MP3 version mp3 bsdtalk165.ogg 16 minutes Ogg version ogg

At MeetBSD with some of the FreeBSD Core Team

A conversation with some of the FreeBSD Core Team at MeetBSD California 2008. I speak with Brooks Davis, Kris Kennaway, Robert Watson, Peter Wemm, and Philip Paeps about the recent core team election, FreeBSD 7.1 and 8, Developer Summits, and the move to Subversion. <http://bsdtalk.blogspot.com/2008/11/bsdtalk164-at-meetbsd-with-some-of.html> bsdtalk interview freebsd core team meetbsd2008 meetbsd robert watson brooks davis kris kennaway peter wemm philip paeps freebsd subversion <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk164.mp3> 18 Mb 38 minutes MP3 version mp3 bsdtalk164.ogg 38 minutes Ogg version ogg

A Tour of iXsystems

A brief description of my visit to iXsystems in California prior to MeetBSD 2008. <http://bsdtalk.blogspot.com/2008/11/bsdtalk163-tour-of-ixsystems.html> bsdtalk interview ixsystems <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk163.mp3> 4 Mb 8 minutes MP3 version mp3 bsdtalk163.ogg 8 minutes Ogg version ogg

BSD on a eeePC 900A

I look forward to attending MeetBSD this weekend.

A brief description of my first attempts to get BSD on a eeePC 900A. I try OpenBSD 4.4, DragonFlyBSD 2.0.1, PC-BSD 7.0.1, and FreeBSD 7. <http://bsdtalk.blogspot.com/2008/11/bsdtalk162-bsd-on-eeepc-900a.html> bsdtalk eeepc <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk162.mp3> 5 Mb 10 minutes MP3 version mp3 bsdtalk162.ogg 10 minutes Ogg version ogg

Live from NYCBSDCon Sunday

A copy of Sunday's live stream from NYCBSDCon 2008. <http://bsdtalk.blogspot.com/2008/10/bsdtalk161-live-from-nycbsdcon-sunday.html> bsdtalk nycbsdcon2008 nycbsdcon interview <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk161.mp3> 12 Mb 25 minutes MP3 version mp3 bsdtalk161.ogg 25 minutes Ogg version ogg

Live from NYCBSDCon Saturday

A copy of Saturday's live stream from NYCBSDCon 2008. I wander around during lunch talking to random people. Voices include Jason Dixon, Pawel Jakub Dawidek, Kris Moore, Matt Olander, George Neville-Neil, Phillip Coblenz, and Jason Wright. <http://bsdtalk.blogspot.com/2008/10/bsdtalk160-live-from-nycbsdcon-saturday.html> bsdtalk nycbsdcon2008 nycbsdcon interview jason dixon pawel jakub dawidek kris more matt olander george neville-neil phillip coblenz jason wright <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk160.mp3> 18 Mb 40 minutes MP3 version mp3 bsdtalk160.ogg 40 minutes Ogg version ogg

Kris Moore

Interview with Kris Moore. We talk about the recent release of PC-BSD 7.0. <http://bsdtalk.blogspot.com/2008/10/bsdtalk159-kris-moore.html> bsdtalk interview kris more pc-bsd <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk159.mp3> 6 Mb 12 minutes MP3 version mp3 bsdtalk159.ogg 12 minutes Ogg version ogg

Interview with Chess Griffin

Interview with Chess Griffin, host of the LinuxReality podcast. We talk about his use of Linux and recent exploration into the BSDs. <http://bsdtalk.blogspot.com/2008/09/bsdtalk158-interview-with-chess-griffin.html> bsdtalk interview chess griffin <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk158.mp3> 11 Mb 24 minutes MP3 version mp3 bsdtalk158.ogg 24 minutes Ogg version ogg

Questions for you

- Things have been very busy at the beginning of the school year, so I'm sorry that I haven't been producing as many shows as usual.
- Registration is open for NYCBSDCon and the list of speakers is available. Are you going?
- I plan on streaming live during the conference. Do you have any suggestions for live streaming software that is known to work well on the BSDs? Are there any live CDs like Dyne:bolic?
- I've come into possession of a Soekris 5501. What are your suggestions for soekris-friendly projects to test?

<http://bsdtalk.blogspot.com/2008/09/bsdtalk157-questions-for-you.html> bsdtalk <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk157.mp3> 3 Mb 6 minutes MP3 version mp3 bsdtalk157.ogg 6 minutes Ogg version ogg

NYCBSDCon Update with Isaac Levy and Steven Kreuzer

An update on NYCBSDCon 2008 with Isaac Levy and Steven Kreuzer. More information on the conference can be found at <http://www.nycbsdcon.org/> <http://bsdtalk.blogspot.com/2008/08/bsd-talk156-nycbsdcon-update-with-isaac.html> bsdtalk interview nycbug nycbsdcon nycbsdcon2008 isaac levy steven kreuzer <http://cisx1.uma.maine.edu/~wbackman/bsd-talk/bsd-talk156.mp3> 7 Mb 15 minutes MP3 version mp3 bsd-talk156.ogg 15 minutes Ogg version ogg

Martin Tournioj from DaemonForums.org

A brief interview with Martin Tournioj, one of the founders of DaemonForums.org. <http://bsd-talk.blogspot.com/2008/07/bsd-talk-155-martin-tournioj-from.html> bsdtalk interview daemonforums martin tournioj <http://cisx1.uma.maine.edu/~wbackman/bsd-talk/bsd-talk155.mp3> 3 Mb 7 minutes MP3 version mp3 bsd-talk155.ogg 7 minutes Ogg version ogg

Matthew Dillon

An interview with Matthew Dillon. He gives a fairly technical description of the HAMMER filesystem features that will make it in the DragonflyBSD 2.0 release. <http://bsd-talk.blogspot.com/2008/07/bsd-talk154-matthew-dillon.html> bsdtalk interview hammer matthew dillon <http://cisx1.uma.maine.edu/~wbackman/bsd-talk/bsd-talk154.mp3> 14 Mb 30 minutes MP3 version mp3 bsd-talk154.ogg 30 minutes Ogg version ogg

Michael W. Lucas

Interview with Michael W. Lucas at BSDCan 2008. We talk about some of his books and strategies for writing technical publications. <http://bsd-talk.blogspot.com/2008/06/bsd-talk153-michael-w-lucas.html> bsdtalk interview bsdcant2008 michael lucas <http://cisx1.uma.maine.edu/~wbackman/bsd-talk/bsd-talk153.mp3> 6 Mb 12 minutes MP3 version mp3 bsd-talk153.ogg 12 minutes Ogg version ogg

A Few FreeBSD Core Team Members

An interview with a few of the FreeBSD Core Team members: Warner Losh, George V. Neville-Neil, Murray Stokely, Hiroki Sato, Robert Watson, Brooks Davis, and Philip Paeps. The interview was recorded at BSDCan2008 in Ottawa, Canada. <http://bsd-talk.blogspot.com/2008/06/bsd-talk152-few-freebsd-core-team.html> bsdtalk interview bsdcant2008 freebsd core warner losh george neville-neil murray stokely hiroki sato robert watson brooks davis philip paeps <http://cisx1.uma.maine.edu/~wbackman/bsd-talk/bsd-talk152.mp3> 12 Mb 26 minutes MP3 version mp3 bsd-talk152.ogg 26 minutes Ogg version ogg

Sean Cody from Frantic Films VFX

Interview with Sean Cody at BSDCan2008. We talk about his use of BSD at a visual effects studio. <http://bsd-talk.blogspot.com/2008/05/bsd-talk151-sean-cody-from-frantic-films.html> bsdtalk interview bsdcant2008 frantic films sean cody <http://cisx1.uma.maine.edu/~wbackman/bsd-talk/bsd-talk151.mp3> 6 Mb 13 minutes MP3 version mp3 bsd-talk151.ogg 13 minutes Ogg version ogg

Alex Feldman from Sangoma

Interview at BSDCan2008 with Alex Feldman from Sangoma. <http://bsd-talk.blogspot.com/2008/05/bsd-talk150-alex-feldman-from-sangoma.html> bsdtalk interview sangoma alex feldman <http://cisx1.uma.maine.edu/~wbackman/bsd-talk/bsd-talk150.mp3> 4 Mb 9 minutes MP3 version mp3 bsd-talk150.ogg 9 minutes Ogg version ogg

Justin Gibbs from the FreeBSD Foundation

Interview with Justin Gibbs from the FreeBSD Foundation. <http://bsd-talk.blogspot.com/2008/05/bsd-talk149-justin-gibbs-from-freebsd.html> bsdtalk interview freebsd foundation justin gibbs <http://cisx1.uma.maine.edu/~wbackman/bsd-talk/bsd-talk149.mp3> 5 Mb 11 minutes MP3 version mp3 bsd-talk149.ogg 11 minutes Ogg version ogg

Jeremy White, Founder of CodeWeavers

Interview with Jeremy White, Founder of CodeWeavers. We talk about the recent availability of an experimental build of Crossover Games for BSD. <http://bsd-talk.blogspot.com/2008/05/bsd-talk148-jeremy-white-founder-of.html> bsdtalk interview freebsd codeweavers crossover jeremy white <http://cisx1.uma.maine.edu/~wbackman/bsd-talk/bsd-talk148.mp3> 7 Mb 16 minutes MP3 version mp3 bsd-talk148.ogg 16 minutes Ogg version ogg

FreeBSD Developer Alexander Motin

Interview with FreeBSD Developer Alexander Motin. We talk about mpd, the net-graph based Multi-link PPP Daemon. For more information, see <http://mpd.sourceforge.net/>. <http://bsdtalk.blogspot.com/2008/04/bsdtalk147-freebsd-developer-alexander.html> bsdtalk interview freebsd mpd alexander motin <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk147.mp3> 8 Mb 16 minutes MP3 version mp3 bsdtalk147.ogg 16 minutes Ogg version ogg

James Cornell

Another interview with Sysadmin James Cornell. We talk about BSD, OpenSolaris, and Linux on the desktop. <http://bsdtalk.blogspot.com/2008/04/bsdtalk146-james-cornell.html> bsdtalk interview desktop james cornell <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk146.mp3> 9 Mb 20 minutes MP3 version mp3 bsdtalk146.ogg 9 minutes Ogg version ogg

Adam Wright from No Starch Press

Intro: Some musings on the consistency and simplicity of BSD.

A brief interview with Adam Wright from No Starch Press, recorded by Micheal Dexter on behalf of BSDTalk. They talk about recent and future BSD books.

<http://bsdtalk.blogspot.com/2008/04/bsdtalk145-adam-wright-from-no-starch.html> bsdtalk interview books no starch press adam wright <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk145.mp3> 4 Mb 8 minutes MP3 version mp3 bsdtalk145.ogg 8 minutes Ogg version ogg

Dan Langille

Interview with Dan Langille. We talk about his new job with Afiliis, and BSDCan 2008. <http://bsdtalk.blogspot.com/2008/03/bsdtalk144-dan-langille.html> bsdtalk interview afiliis bsdcn2008 dan langille <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk144.mp3> 10 Mb 22 minutes MP3 version mp3 bsdtalk144.ogg 22 minutes Ogg version ogg

BSD Hobbiest Deborah Norling

Interview with Deborah Norling. We talk about her use of BSD on old hardware, accessibility on the BSDs, and Simh (<http://simh.trailing-edge.com>). <http://bsdtalk.blogspot.com/2008/03/bsdtalk143-bsd-hobbiest-deborah-norling.html> bsdtalk interview accessibility deborah norling <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk143.mp3> 10 Mb 23 minutes MP3 version mp3 bsdtalk143.ogg 23 minutes Ogg version ogg

FreeBSD Lead Release Engineer Ken Smith

Interview with FreeBSD Lead Release Engineer Ken Smith. <http://bsdtalk.blogspot.com/2008/02/bsdtalk142-freebsd-lead-release.html> bsdtalk interview freebsd release engineer ken smith <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk142.mp3> 7 Mb 16 minutes MP3 version mp3 bsdtalk142.ogg 16 minutes Ogg version ogg

PBI 4 with Kris Moore

Interview with PC-BSD founder Kris Moore about the new features in PBI 4. <http://bsdtalk.blogspot.com/2008/02/bsdtalk141-pbi4-with-kris-moore.html> bsdtalk interview pc-bsd kris moore <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk141.mp3> 5 Mb 10 minutes MP3 version mp3 bsdtalk141.ogg 10 minutes Ogg version ogg

The Mult Project with Kristaps Dzonsons

We talk about the Mult project, which is “an on-going research project to create a high-performance instance multiplicity system.” You can find more information at <http://mult.bsd.lv/>. He also gives a quick update on Sys-jail. <http://bsdtalk.blogspot.com/2008/02/bsdtalk140-mult-project-with-kristaps.html> bsdtalk interview multi project kristaps dzonsons <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk140.mp3> 14 Mb 30 minutes MP3 version mp3 bsdtalk140.ogg 30 minutes Ogg version ogg

Dru Lavigne

Interview with Dru Lavigne. We talk about her new book “The Best of FreeBSD Basics” and also get an update on some other projects including BSD Certification.

See the following links for more information:

- <https://register.bsdcertification.org/register/get-a-bsdcg-id>
- <http://reedmedia.net/books/freebsd-basics>
- <http://www.osbr.ca>

<http://bsdtalk.blogspot.com/2008/01/bsdtalk139-dru-lavigne.html> bsdtalk interview dru lavigne the best of freebsd basics <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk139.mp3 7 Mb 14 minutes MP3 version mp3 bsdtalk139.ogg 14 minutes Ogg version ogg

Central Syslog

Setting up a central syslog server.

- If you are concerned about the security of your logs, use a dedicated machine and lock it down.
- Keep clocks in sync.
- You may need to change log rotation schedule in `/etc/newsyslog.conf`. You can rotate based in size and/or time. This can be as much a policy decision as a hardware decision.
- On central log host, change `syslogd` flags to listen to network. Each BSD does this differently, so check the man pages. Also, check out the `-n` flag for busy environments.
- Make sure host firewall allows syslog traffic through.
- Be careful to limit syslog traffic to just the trusted network or hosts. FreeBSD man page refers to `syslogd` as a “remote disk filling service”.
- For heavy logging environments, it is important to have a dedicated network. A down `syslogd` server can create a lot of “ARP who-has” broadcasts.
- Most network devices such as printers and commercial firewalls support sending to a central syslog server. Take a look at “Snare” for Windows hosts.
- To send messages from a Unix host, specify the host name prepended with `@` instead of a file for logging in `/etc/syslog.conf`. For example, change `/var/log/xferlog` to `@loghost.mydomain.biz`. You can also copy and edit the line to have it log to both a local file and a remote host.

<http://bsdtalk.blogspot.com/2008/01/bsdtalk138-central-syslog.html> bsdtalk syslog <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk138.mp3 3 Mb 7 minutes MP3 version mp3 bsdtalk138.ogg 7 minutes Ogg version ogg

Open Community Camp with Marten Vijn

Interview with Marten Vijn about www.OpenCommunityCamp.org. <http://bsdtalk.blogspot.com/2008/01/bsdtalk137-open-community-camp-with.html> bsdtalk interview opencommunitycamp marten vijn <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk137.mp3 6 Mb 13 minutes MP3 version mp3 bsdtalk137.ogg 13 minutes Ogg version ogg

PF with Peter N. M. Hansteen

An interview with Peter N. M. Hansteen, recorded by Michael Dexter on behalf of BSDTalk. If you would like to learn more about the PF firewall, check out “The Book of PF” which is available at <http://nostarch.com/frameset.php?startat=pf> <http://bsdtalk.blogspot.com/2007/12/bsdtalk136-pf-with-peter-n-m-hansteen.html> bsdtalk interview pf michael dexter peter n m hansteen book of pf <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk136.mp3 7 Mb 16 minutes MP3 version mp3 bsdtalk136.ogg 15 minutes Ogg version ogg

Joerg Sonnenberger

Michael Dexter sent me an interview he recorded on behalf of BSDTalk with Joerg Sonnenberger at EuroBSDCon 2007. <http://bsdtalk.blogspot.com/2007/11/bsdtalk135-joerg-sonnenberger.html> bsdtalk interview eurobsdcon eurobsdcon2007 michael dexter joerg sonnenberger <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk135.mp3> 8 Mb 17 minutes MP3 version mp3 bsdtalk135.ogg 17 minutes Ogg version ogg

AsiaBSDCon Update with Hiroki Sato and George Neville-Neil

A quick update on AsiaBSDCon 2008 with Hiroki Sato and George Neville-Neil. More information at <http://2008.asiabsdcon.org/>. <http://bsdtalk.blogspot.com/2007/10/bsdtalk134-asiabsdcon-update-with.html> bsdtalk interview asiabsdcon hiroki sato george neville-neil <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk134.mp3> 5 Mb 10 minutes MP3 version mp3 bsdtalk134.ogg 10 minutes Ogg version ogg

OpenCon 2007 update from Marc Balmer

A short update on OpenCon 2007 with Marc Balmer. More information at <http://www.opencon.org/>. <http://bsdtalk.blogspot.com/2007/10/bsdtalk133-opencon-2007-update-from.html> bsdtalk interview opencon marc balmer <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk133.mp3> 3 Mb 7 minutes MP3 version mp3 bsdtalk133.ogg 7 minutes Ogg version ogg

Richard Stallman

Interview with Richard Stallman. <http://bsdtalk.blogspot.com/2007/10/bsdtalk132-richard-stallman.html> bsdtalk interview rms richard stallman <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk132.ogg> 16 Mb 28 minutes Ogg version ogg

PCC with Anders “Ragge” Magnusson

Interview with Anders “Ragge” Magnusson. We talk about his work on the Portable C Compiler. More information can be found at <http://pcc.ludd.ltu.se/>. <http://bsdtalk.blogspot.com/2007/10/bsdtalk131-pcc-with-anders-ragge.html> bsdtalk interview pcc ragge anders magnusson <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk131.mp3> 7 Mb 15 minutes MP3 version mp3 bsdtalk131.ogg 15 minutes Ogg version ogg

Network Stack Virtualization with Marko Zec

Michael Dexter sent me an interview he recorded on behalf of BSDTalk with Marko Zec at EuroBSDCon 2007. More information on the project at <http://imunes.tel.fer.hr/virtnet/>. <http://bsdtalk.blogspot.com/2007/10/bsdtalk130-network-stack-virtualization.html> bsdtalk interview stack virtualization marko zec <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk130.mp3> 8 Mb 16 minutes MP3 version mp3 bsdtalk130.ogg 16 minutes Ogg version ogg

BSDCertification Update with Dru Lavigne

Interview with Dru Lavigne. We talk about the progress of BSDCertification.org and also her new position with the Open Source Business Resource at <http://www.osbr.ca/>. <http://bsdtalk.blogspot.com/2007/09/bsdcertification-update-with.html> bsdtalk interview bsdcertification dru lavigne <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk129.mp3> 10 Mb 20 minutes MP3 version mp3 bsdtalk129.ogg 22 minutes Ogg version ogg

Sysjail Revisited with Michael Dexter

Interview with Michael Dexter. We talk about the new sysjail and the recent system call wrapper issues. <http://bsdtalk.blogspot.com/2007/09/bsdtalk128-sysjail-revisited-with.html> bsdtalk interview sysjail michael dexter <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk128.mp3> 10 Mb 22 minutes MP3 version mp3 bsdtalk128.ogg 22 minutes Ogg version ogg

Why I like the CLI

Why I like the CLI:

- Uses minimal resources. Less space, less memory, fewer dependencies.
- Transparency. GUI hides internals, limits options.

- Similar between Unix-like systems. GUI tools seem to change every week.
- Remote management. SSH rocks.
- Everything is text. Configs, devices, output. CLI is natural complement.
- Pipes and scripts. One time is hard, a thousand times is easy.
- Only need a few tools. Grep, sed, awk, vi, cron.
- Text config files. Easy to version, share, and comment.
- Requires reading skills instead of clicking skills.
- Much faster when you know what you are doing.

<http://bsdtalk.blogspot.com/2007/08/bsdtalk127-why-i-like-cli.html> bsdtalk cli will backman
<http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk127.mp3> 6 Mb 12 minutes MP3 version mp3 bsdtalk127.ogg
12 minutes Ogg version ogg

MidnightBSD founder Lucas Holt

Interview with MidnightBSD founder Lucas Holt. <http://bsdtalk.blogspot.com/2007/08/bsdtalk126-midnightbsd-founder-lucas.html> bsdtalk interview midnightbsd lucas holt <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk126.mp3> 7 Mb 15 minutes MP3 version mp3 bsdtalk126.ogg 15 minutes Ogg version ogg

Matthew Dillon

Interview with DragonflyBSD's Matthew Dillon. We talk about the 1.10 release and the design of a new filesystem. <http://bsdtalk.blogspot.com/2007/08/bsdtalk125-matthew-dillon.html> bsdtalk interview dragonflybsd matthew dillon <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk125.mp3> 10 Mb 20 minutes MP3 version mp3 bsdtalk125.ogg 20 minutes Ogg version ogg

PC-BSD Founder Kris Moore

Interview with PC-BSD Founder Kris Moore. We talk about the upcoming 1.4 release. <http://bsdtalk.blogspot.com/2007/08/bsdtalk124-pc-bsd-founder-kris-moore.html> bsdtalk interview pc-bsd kris moore <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk124.mp3> 6 Mb 12 minutes MP3 version mp3 bsdtalk124.ogg 12 minutes Ogg version ogg

William “whurley” Hurley, Chief Architect of Open Source Strategy at BMC Software, Inc.

Interview with William “whurley” Hurley, Chief Architect of Open Source Strategy at BMC Software, Inc. We talk about the BMC Developer Network. <http://bsdtalk.blogspot.com/2007/07/bsdtalk123-william-whurley-hurley-chief.html> bsdtalk interview bmc software whurley william hurley <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk123.mp3> 14 Mb 28 minutes MP3 version mp3 bsdtalk123.ogg 28 minutes Ogg version ogg

Embedding FreeBSD with M. Warner Losh

Interview with M. Warner Losh about embedding FreeBSD. <http://bsdtalk.blogspot.com/2007/07/bsdtalk122-embedding-freebsd-with-m.html> bsdtalk interview embedding freebsd m warner losh <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk122.mp3> 8 Mb 16 minutes MP3 version mp3 bsdtalk122.ogg 16 minutes Ogg version ogg

Fast IPsec with George Neville-Neil

Interview with George Neville-Neil about Fast IPsec. <http://bsdtalk.blogspot.com/2007/07/bsdtalk121-fast-ipsec-with-george.html> bsdtalk interview ipsec george neville-neil <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk121.mp3> 7 Mb 14 minutes MP3 version mp3 bsdtalk121.ogg 14 minutes Ogg version ogg

BSD Hacker Isaac “Ike” Levy

Interview with BSD Hacker Isaac “Ike” Levy. To hear more of Ike and other NYCBUG audio, visit <http://www.fetissof.org/public/nycbug/> <http://bsdtalk.blogspot.com/2007/07/bsdtalk120-bsd-hacker-isaac-ike->

[levy.html](#) bsdtalk interview nycbug isaac levy <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk120.mp3 13 Mb 26 minutes MP3 version mp3 bsdtalk120.ogg 26 minutes Ogg version ogg

Playing with IPv6

I ramble on about how I have been experimenting with IPv6. For more details, see <http://cisx1.uma.maine.edu/~wbackman/cis341/resources/ipv6-test-lab.html>. <http://bsdtalk.blogspot.com/2007/07/bsdtalk119-playing-with-ipv6.html> bsdtalk ipv6 <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk119.mp3 8 Mb 15 minutes MP3 version mp3 bsdtalk119.ogg 15 minutes Ogg version ogg

Sidsel Jensen from EuroBSDCon

Interview with Sidsel Jensen from www.eurobsdcon.org. <http://bsdtalk.blogspot.com/2007/06/bsdtalk118-sidsel-jensen-from.html> bsdtalk interview eurobsdcon eurobsdcon2007 sidsel jensen <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk118.mp3 5 Mb 9 minutes MP3 version mp3 bsdtalk118.ogg 9 minutes Ogg version ogg

One Time Passwords

- Important when you don't trust the computer you are using, such as a library computer or internet kiosk.
- Available by default in Free/Net/Open BSD.
- FreeBSD uses OPIE, Net/Open use S/Key.
- One time passwords are based on your pass phrase, a non-repeating sequence number, and a seed.
- Initial setup should be done directly on the server.
- “skeyinit” for Net/Open, “opiepasswd -c” for FreeBSD.
- Enter a pass phrase that is not your regular account password.
- Find your current sequence number and seed with “opieinfo” or “skeyinfo”, for example: “497 pc5246”.
- Generate a list of the next 10 passwords and write them down, using “opiekey -n 10 497 pc5246” or “skey -n 10 497 pc5246”.
- When you log in from a remote machine that might have a keystroke logger, you can now use a one time password instead of your regular password.
- For OpenBSD, log in as account:skey, for example “bob:skey”, which will cause the system to present the s/key challenge.
- For NetBSD, the system will always present you with the s/key challenge if it is configured for your account, although you can still use your regular password.
- FreeBSD by default will force you to use a one time password if it is configured for your account.
- If you want both OPIE and password authentication, FreeBSD allows you to list trusted networks or hosts in `/etc/opieaccess`.
- Instead of carrying a list of passwords around, you can use s/key generators on a portable device that you trust, such as a palm pilot.
- For more info, check the man pages.

<http://bsdtalk.blogspot.com/2007/06/bsdtalk117-one-time-passwords.html> bsdtalk security one time passwords <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk117.mp3 4 Mb 6 minutes MP3 version mp3 bsdtalk117.ogg 6 minutes Ogg version ogg

Rick Macklem and NFSv4

Interview with Rick Macklem about his work with NFSv4. More information at <http://snowwhite.cis.uoguelph.ca/nfsv4/>. <http://bsdtalk.blogspot.com/2007/06/bsdtalk116-rick-macklem-and-nfsv4.html> bsdtalk interview nfs rick macklem <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk116.mp3> 6 Mb 13 minutes MP3 version mp3 bsdtalk116.ogg 13 minutes Ogg version ogg

Jun-ichiro “itojun” Itoh Hagino

Interview with KAME project core researcher Jun-ichiro “itojun” Itoh Hagino. <http://bsdtalk.blogspot.com/2007/06/bsdtalk115-few-freebsd-core-team.html> bsdtalk interview kame itojun jun-ichiro itoh hagino <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk115.mp3> 4 Mb 10 minutes MP3 version mp3 bsdtalk115.ogg 10 minutes Ogg version ogg

A Few FreeBSD Core Team Members

An interview with a few of the FreeBSD Core Team members: Brooks Davis, Warner Losh, George V. Neville-Neil, Hiroki Sato, and Robert Watson. The interview was recorded at BSDCan in Ottawa, Canada. <http://bsdtalk.blogspot.com/2007/05/bsdtalk114-few-freebsd-core-team.html> bsdtalk interview freebsd core brooks davis warner losh george neville-neil hiroki sato robert watson <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk114.mp3> 16 Mb 35 minutes MP3 version mp3 bsdtalk114.ogg 35 minutes Ogg version ogg

Designing BSD Rootkits Author Joseph Kong

Interview with Joseph Kong, Author of “Designing BSD Rootkits: An Introduction to Kernel Hacking” from No Starch Press. The interview was recorded at BSDCan in Ottawa. <http://bsdtalk.blogspot.com/2007/05/bsdtalk113-designing-bsd-rootkits.html> bsdtalk interview kernel rootkits books joseph kong <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk113.mp3> 8 Mb 15 minutes MP3 version mp3 bsdtalk113.ogg 15 minutes Ogg version ogg

Qing Li and Tatuya Jinmei

Interview at at BSDCan with Qing Li and Tatuya Jinmei. We talk about the books that they authored with Keiichi Shima: “IPv6 Core Protocols Implementation” and “IPv6 Advanced Protocols Implementation.” The books are available at Amazon.com or on the publisher’s web site, www.mkp.com. <http://bsdtalk.blogspot.com/2007/05/bsdtalk112-qing-li-and-tatuya-jinmei.html> bsdtalk interview ipv6 books qing li tatuya jimei <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk112.mp3> 10 Mb 20 minutes MP3 version mp3 bsdtalk112.ogg 20 minutes Ogg version ogg

FreeBSD Developer Diane Bruce

Interview with FreeBSD developer Diane Bruce. We talk about Ham Radio on BSD. Slides from one of her talks: http://www.oarc.net/presentations/hamradio_on_freebsd.pdf <http://bsdtalk.blogspot.com/2007/05/bsdtalk111-freebsd-developer-diane.html> bsdtalk interview freebsd diane bruce <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk111.mp3> 5 Mb 10 minutes MP3 version mp3 bsdtalk111.ogg 10 minutes Ogg version ogg

Josh Berkus, Postgresql Lead at Sun Microsystems

Interview with Josh Berkus, Postgresql Lead at Sun Microsystems. We talk about the upcoming PGCon on 23-24 May 2007. More info at <http://www.pgcon.org>. <http://bsdtalk.blogspot.com/2007/05/bsdtalk110-josh-berkus-postgresql-lead.html> bsdtalk interview postgresql josh berkus <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk110.mp3> 9 Mb 19 minutes MP3 version mp3 bsdtalk110.ogg 19 minutes Ogg version ogg

George Neville-Neil and Using VMs for Development

George Neville-Neil and Using VMs for Development. See <http://blogs.freebsdish.org/gnn> for more information. <http://bsdtalk.blogspot.com/2007/04/bsdtalk109-george-neville-neil-and.html> bsdtalk interview virtual machines george neville-neil <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk109.mp3> 6 Mb 12 minutes MP3 version mp3 bsdtalk109.ogg 12 minutes Ogg version ogg

Matt Juszczak from bsdfjobs.net

Interview with Matt Juszczak from bsdfjobs.net. <http://bsdtalk.blogspot.com/2007/04/bsdtalk108-matt-juszczak-from.html> bsdtalk interview bsdfjobs matt juszczak <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk108.mp3>

4 Mb 8 minutes MP3 version mp3 bsdtalk108.ogg 4 minutes Ogg version ogg

Contiki OS Developer Adam Dunkels

Interview with Contiki OS Developer Adam Dunkels. You can find more information at <http://www.sics.se/contiki/>. <http://bsdtalk.blogspot.com/2007/04/bsdtalk107-contiki-os-developer-adam.html> bsdtalk interview contikios adam dunkels <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk107.mp3 13 Mb 27 minutes MP3 version mp3 bsdtalk107.ogg 27 minutes Ogg version ogg

Interview with Matthieu Herrb about Xenocara

Interview with Matthieu Herrb about Xenocara. <http://bsdtalk.blogspot.com/2007/04/bsdtalk106-interview-with-matthieu.html> bsdtalk interview xenocara matthieu herrb <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk106.mp3 7 Mb 14 minutes MP3 version mp3 bsdtalk106.ogg 14 minutes Ogg version ogg

Intro to PF with Jason Dixon

Introduction to PF with Jason Dixon. <http://bsdtalk.blogspot.com/2007/03/bsdtalk105-intro-to-pf-with-jason-dixon.html> bsdtalk interview pf jason dixon <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk105.mp3 12 Mb 25 minutes MP3 version mp3 bsdtalk105.ogg 25 minutes Ogg version ogg

Getting to know X

Getting to know the X Window System.

Make sure you are in a text only mode. You might need to change how the system boots, or boot into single user mode.

- “startx” to make sure X is working right.
- “X” by itself gives the basic grey screen.
- “ctrl” and “alt” and “backspace” keys at the same time will zap X.
- “X & xterm -display :0”
- “xterm -geometry +300+300”
- “twm” or “metacity”

<http://bsdtalk.blogspot.com/2007/03/bsdtalk104-getting-to-know-x.html> bsdtalk X
<http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk104.mp3 5 Mb 10 minutes MP3 version mp3 bsdtalk104.ogg
10 minutes Ogg version ogg

Robert Ricci from Emulab

Interview with Robert Ricci from www.Emulab.net. <http://bsdtalk.blogspot.com/2007/03/bsdtalk103-robert-ricci-from-emulab.html> bsdtalk interview emulab robert ricci <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk103.mp3 8 Mb 16 minutes MP3 version mp3 bsdtalk103.ogg 16 minutes Ogg version ogg

Cisco Distinguished Engineer Randall Stewart

Interview with Cisco Distinguished Engineer Randall Stewart. We talk about the Stream Control Transmission Protocol and his work bringing it to FreeBSD. <http://bsdtalk.blogspot.com/2007/03/bsdtalk102-cisco-distinguished-engineer.html> bsdtalk interview cisco freebsd stream control transmission protocol randall stewart <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/> bsdtalk102.mp3 17 Mb 35 minutes MP3 version mp3 bsdtalk102.ogg 35 minutes Ogg version ogg

FreeBSD Developer George Neville-Neil

Interview with FreeBSD developer George Neville-Neil. We talk about the packet construction set and the packet debugger. <http://bsdtalk.blogspot.com/2007/02/bsdtalk101-freebsd-developer-george.html> bsdtalk interview freebsd

packet construction set george neville-neil <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk101.mp3> 10 Mb 19 minutes MP3 version mp3 bsdtalk101.ogg 19 minutes Ogg version ogg

NetBSD Developer Lubomir Sedlacik

Interview with NetBSD Developer Lubomir Sedlacik. We talk about pkgsrcCon 2007. <http://bsdtalk.blogspot.com/2007/02/bsdtalk100-netbsd-developer-lubomir.html> bsdtalk interview netbsd pkgsrc-con lubomir sedlacik <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk100.mp3> 7 Mb 13 minutes MP3 version mp3 bsdtalk100.ogg 13 minutes Ogg version ogg

AsiaBSDCon PC Chair George Neville-Neil

Interview with AsiaBSDCon 2007 Program Committee Chair George Neville-Neil. <http://bsdtalk.blogspot.com/2007/02/bsdtalk099-asiabsdcon-pc-chair-george.html> bsdtalk interview asiabsdcon asiabsdcon2007 george neville-neil <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk099.mp3> 7 Mb 14 minutes MP3 version mp3 bsdtalk099.ogg 14 minutes Ogg version ogg

DragonFlyBSD Developer Matthew Dillon

Interview with DragonFlyBSD developer Matthew Dillon. We talk about the 1.8 release. <http://bsdtalk.blogspot.com/2007/02/bsdtalk098-dragonflybsd-developer.html> bsdtalk interview dragonflybsd matthew dillon <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk098.mp3> 12 Mb 24 minutes MP3 version mp3 bsdtalk098.ogg 24 minutes Ogg version ogg

OpenBSD Developer Pierre-Yves Ritschard

Interview with OpenBSD Developer Pierre-Yves Ritschard. We talk about hoststated. <http://bsdtalk.blogspot.com/2007/02/bsdtalk097-openbsd-developer-pierre.html> bsdtalk interview openbsd hoststated pierre-yves ritschard <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk097.mp3> 8 Mb 16 minutes MP3 version mp3 bsdtalk097.ogg 16 minutes Ogg version ogg

Artist and Musician Ty Semaka

Interview with Artist and Musician Ty Semaka. You can find his work at <http://www.tysemaka.com/>, and also on the OpenBSD CDs, posters, and shirts. <http://bsdtalk.blogspot.com/2007/01/bsdtalk096-artist-and-musician-ty.html> bsdtalk interview openbsd artwork ty semaka <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk096.mp3> 6 Mb 12 minutes MP3 version mp3 bsdtalk096.ogg 12 minutes Ogg version ogg

OpenBSD Developer Claudio Jeker

Interview with OpenBSD Developer Claudio Jeker. <http://bsdtalk.blogspot.com/2007/01/bsdtalk095-openbsd-developer-claudio.html> bsdtalk interview openbsd claudio jeker <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk095.mp3> 7 Mb 15 minutes MP3 version mp3 bsdtalk095.ogg 15 minutes Ogg version ogg

BSD Consultant Jeremy C. Reed

Interview with BSD Consultant Jeremy C. Reed from <http://www.reedmedia.net/> <http://bsdtalk.blogspot.com/2007/01/bsdtalk094-bsd-consultant-jeremy-c.html> bsdtalk interview consultancy jeremy c reed <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk094.mp3> 8 Mb 16 minutes MP3 version mp3 bsdtalk094.ogg 16 minutes Ogg version ogg

EMC Lab Admin Glen R. J. Neff

Interview with EMC Lab Administrator Glen R. J. Neff. <http://bsdtalk.blogspot.com/2007/01/bsdtalk093-emc-lab-admin-glen-r-j-neff.html> bsdtalk interview emc lab glen r j neff <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk093.mp3> 15 Mb 30 minutes MP3 version mp3 bsdtalk093.ogg 30 minutes Ogg version ogg

Run Your Own Server Podcast Host Adam Glen

Interview with Adam Glen, one of the hosts of the Run Your Own Server Podcast. <http://bsdtalk.blogspot.com/2007/01/bsdtalk092-run-your-own-server-podcast.html> bsdtalk interview run your own server adam glen <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk092.mp3> 6 Mb 12 minutes MP3 version mp3 bsdtalk092.ogg 12 minutes Ogg version ogg

Phil Pereira from bsdnexus.com

Interview with Phil Pereira from bsdnexus.com. <http://bsdtalk.blogspot.com/2007/01/bsdtalk091-phil-pereira-from.html> bsdtalk interview bsdnexus phil pereira <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk091.mp3> 9 Mb 18 minutes MP3 version mp3 bsdtalk091.ogg 18 minutes Ogg version ogg

Sys Admin Mike Erdely

Interview with Sys Admin Mike Erdely. You can find more information on his use of binpatch at <http://erdelynet.com/binpatch>. <http://bsdtalk.blogspot.com/2006/12/bsdtalk090-sys-admin-mike-erdely.html> bsdtalk interview binpatch mike erdely <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk090.mp3> 8 Mb 17 minutes MP3 version mp3 bsdtalk090.ogg 17 minutes Ogg version ogg

NetBSD Release Engineer Jeff Rizzo

Interview with NetBSD Release Engineer Jeff Rizzo. We talk about the upcoming 4.0 release. <http://bsdtalk.blogspot.com/2006/12/bsdtalk089-netbsd-release-engineer.html> bsdtalk interview netbsd jeff rizzo <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk089.mp3> 7 Mb 15 minutes MP3 version mp3 bsdtalk089.ogg 15 minutes Ogg version ogg

A Year of BSDTalk

A short ramble about the first year of bsdtalk. <http://bsdtalk.blogspot.com/2006/12/bsdtalk088-year-of-bsdtalk.html> bsdtalk anniversary <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk088.mp3> 4 Mb 8 minutes MP3 version mp3 bsdtalk088.ogg 8 minutes Ogg version ogg

FreeBSD Developer Joseph Koshy

Interview with FreeBSD developer Joseph Koshy about libELF. You can find more information about libELF at <http://wiki.freebsd.org/LibElf>. <http://bsdtalk.blogspot.com/2006/12/bsdtalk087-freebsd-developer-joseph.html> bsdtalk interview freebsd libelf joseph koshy <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk087.mp3> 5 Mb 9 minutes MP3 version mp3 bsdtalk087.ogg 9 minutes Ogg version ogg

FreeBSD Developer Kip Macy

Interview with FreeBSD developer Kip Macy. We talk about the Ultrasparc T1 port. <http://bsdtalk.blogspot.com/2006/12/bsdtalk086-freebsd-developer-kip-macy.html> bsdtalk interview freebsd ultrasparc t1 kip macy <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk086.mp3> 10 Mb 22 minutes MP3 version mp3 bsdtalk086.ogg 22 minutes Ogg version ogg

FreeBSD Port Committer Thomas McLaughlin

Interview with FreeBSD Port Committer Thomas McLaughlin about the BSD# project. <http://bsdtalk.blogspot.com/2006/11/bsdtalk085-freebsd-port-committer.html> bsdtalk interview freebsd bsd# thomas mclaughlin <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk085.mp3> 9 Mb 18 minutes MP3 version mp3 bsdtalk085.ogg 18 minutes Ogg version ogg

FreeBSD Release Engineer Bruce Mah

Interview with FreeBSD Release Engineer Bruce Mah. <http://bsdtalk.blogspot.com/2006/11/bsdtalk084-freebsd-release-engineer.html> bsdtalk interview freebsd release engineer bruce mah <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk084.mp3> 7 Mb 15 minutes MP3 version mp3 bsdtalk084.ogg 15 minutes Ogg version ogg

Pkgsrc Developer Johnny Lam

Interview with pkgsrc developer Johnny Lam. <http://bsdtalk.blogspot.com/2006/11/bsdtalk083-pkgsrc-developer-johnny-lam.html> bsdtalk interview pkgsrc johnny lam <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk083.mp3> 6 Mb 13 minutes MP3 version mp3 bsdtalk083.ogg 13 minutes Ogg version ogg

OpenBSD Developer Jason Wright

Interview with OpenBSD developer Jason Wright. We talk about his work on sparc and also amateur radio. <http://bsdtalk.blogspot.com/2006/11/bsdtalk082-openbsd-developer-jason.html> bsdtalk interview openbsd sparc radio jason wright <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk082.mp3> 8 Mb 17 minutes MP3 version mp3 bsdtalk082.ogg 17 minutes Ogg version ogg

Thorsten Glaser from MirOS

Interview with Thorsten Glaser from MirOS, which can be found at www.mirbsd.org. <http://bsdtalk.blogspot.com/2006/11/bsdtalk081-thorsten-glaser-from-miros.html> bsdtalk interview miros thomas glaser <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk081.mp3> 9 Mb 19 minutes MP3 version mp3 bsdtalk081.ogg 19 minutes Ogg version ogg

EuroBSDCon Organizer Massimiliano Stucchi

Interview with EuroBSDCon organizer Massimiliano Stucchi. <http://bsdtalk.blogspot.com/2006/11/bsdtalk080-eurobsdcon-organizer.html> bsdtalk interview eurobsdcon eurobsdcon2006 massimiliano stucchi <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk080.mp3> 4 Mb 8 minutes MP3 version mp3 bsdtalk080.ogg 8 minutes Ogg version ogg

OpenBSD Developer David Gwynne

Interview with OpenBSD developer David Gwynne. We talk about the upcoming 4.0 release of OpenBSD and current projects that he is working on. <http://bsdtalk.blogspot.com/2006/10/bsdtalk079-openbsd-developer-david.html> bsdtalk interview openbsd david gwynne <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk079.mp3> 8 Mb 16 minutes MP3 version mp3 bsdtalk079.ogg 16 minutes Ogg version ogg

Kris Moore from PC-BSD

Interview with Kris Moore from PC-BSD. <http://bsdtalk.blogspot.com/2006/10/bsdtalk078-kris-moore-from-pc-bsd.html> bsdtalk interview pc-bsd kris moore <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk078.mp3> 10 Mb 21 minutes MP3 version mp3 bsdtalk078.ogg 21 minutes Ogg version ogg

Matt Olander from iXsystems

Interview with Matt Olander from www.ixsystems.com. <http://bsdtalk.blogspot.com/2006/10/bsdtalk077-matt-olander-from-ixsystems.html> bsdtalk interview ixsystems matt olander <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk077.mp3> 9 Mb 19 minutes MP3 version mp3 bsdtalk077.ogg 19 minutes Ogg version ogg

OpenBSD Developer Marc Balmer

Interview with OpenBSD Developer Marc Balmer. We talk about www.opencon.org and his work with OpenBSD. <http://bsdtalk.blogspot.com/2006/10/bsdtalk076-openbsd-developer-marc.html> bsdtalk interview opencon openbsd marc balmer <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk076.mp3> 7 Mb 15 minutes MP3 version mp3 bsdtalk076.ogg 15 minutes Ogg version ogg

Interview with Hiroki Sato and George Neville-Neil from AsiaBSDCon

Interview with Hiroki Sato and George Neville-Neil from AsiaBSDCon. More info at <http://2006.asiabsdcon.org/>. <http://bsdtalk.blogspot.com/2006/10/bsdtalk074-interview-with-hiroki-sato.html> bsdtalk interview asiabsdcon asiabsdcon2006 hiroki sato george neville-neil <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk074.mp3> 6 Mb 13 minutes MP3 version mp3 bsdtalk074.ogg 13 minutes Ogg version ogg

Interview with Sevan Janiyan

Interview with Sevan Janiyan. We talk about the Brighton Chilli WiFi hotspot project, which can be found at <http://brightonchilli.geeklan.co.uk/> <http://bsdtalk.blogspot.com/2006/10/bsdtalk073-interview-with-sevan.html> bsdtalk interview brighton chilli wifi sevan janiyan <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk073.mp3> 6 Mb 13 minutes MP3 version mp3 bsdtalk073.ogg 13 minutes Ogg version ogg

Interview with Poul-Henning Kamp about Varnish

Interview with Poul-Henning Kamp about Varnish. More information at <http://www.varnish-cache.org/>. <http://bsdtalk.blogspot.com/2006/10/bsdtalk072-interview-with-poul-henning.html> bsdtalk interview varnish poul-

henning kamp <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk072.mp3> 17 Mb 36 minutes MP3 version mp3
bsdtalk072.ogg 36 minutes Ogg version ogg

Interview with Einar Th. Einarsson from f-prot.com

Interview with Einar Th. Einarsson from f-prot.com. <http://bsdtalk.blogspot.com/2006/09/bsdtalk071-interview-with-einar-th.html> bsdtalk interview f-prot einar th einarsson <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk071.mp3> 8 Mb 17 minutes MP3 version mp3 bsdtalk071.ogg 17 minutes Ogg version ogg

Interview with NetBSD Developer Tim Rightnour

Interview with NetBSD Developer Tim Rightnour. We talk about NetBSD/prep. <http://bsdtalk.blogspot.com/2006/09/bsdtalk070-interview-with-netbsd.html> bsdtalk interview netbsd tim rightnour <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk070.mp3> 7 Mb 15 minutes MP3 version mp3 bsdtalk070.ogg 15 minutes Ogg version ogg

Interview with Christoph Egger about Xen on OpenBSD

Interview with Christoph Egger about Xen on OpenBSD. <http://bsdtalk.blogspot.com/2006/09/bsdtalk069-interview-with-christoph.html> bsdtalk interview openbsd xen christoph egger <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk069.mp3> 7 Mb 15 minutes MP3 version mp3 bsdtalk069.ogg 15 minutes Ogg version ogg

Interview with OpenBSD Developer Bob Beck

Interview with OpenBSD Developer Bob Beck. <http://bsdtalk.blogspot.com/2006/09/bsdtalk068-interview-with-openbsd.html> bsdtalk interview openbsd bob beck <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk068.mp3> 12 Mb 26 minutes MP3 version mp3 bsdtalk068.ogg 26 minutes Ogg version ogg

Interview with Dan Langille about backups

Interview with Dan Langille about backups. Check out <http://www.bacula.org/> <http://bsdtalk.blogspot.com/2006/09/bsdtalk067-interview-with-dan-langille.html> bsdtalk interview bacula dan langille <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk067.mp3> 10 Mb 22 minutes MP3 version mp3 bsdtalk067.ogg 22 minutes Ogg version ogg

Interview with Michael Dexter about sysjail

Interview with Michael Dexter about sysjail. <http://sysjail.bsd.lv/> <http://bsdtalk.blogspot.com/2006/09/bsdtalk066-interview-with-michael.html> bsdtalk interview sysjail michael dexter <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk066.mp3> 16 Mb 35 minutes MP3 version mp3 bsdtalk066.ogg 35 minutes Ogg version ogg

Interview with Eirik Øverby.

Interview with Eirik Øverby. We talk about his use of BSD and Jails. <http://bsdtalk.blogspot.com/2006/09/bsdtalk065-interview-with-eirik-verby.html> bsdtalk interview jails eirik Overby <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk065.mp3> 9 Mb 18 minutes MP3 version mp3 bsdtalk065.ogg 18 minutes Ogg version ogg

Interview with NetBSD Developer Jason Thorpe

Interview with NetBSD Developer Jason Thorpe <http://bsdtalk.blogspot.com/2006/09/bsdtalk064-interview-with-netbsd.html> bsdtalk interview netbsd jason thorpe <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk064.mp3> 18 Mb 38 minutes MP3 version mp3 bsdtalk064.ogg 38 minutes Ogg version ogg

Interview with Mitchell Smith about BSD and Accessibility

Interview with Mitchell Smith about BSD and Accessibility. <http://bsdtalk.blogspot.com/2006/08/bsdtalk063-interview-with-mitchell.html> bsdtalk interview accessibility mitchell smith <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk063.mp3> 8 Mb 17 minutes MP3 version mp3 bsdtalk063.ogg 17 minutes Ogg version ogg

Interview with YAWS developer Claes Klacke Wikstrom

Interview with YAWS developer Claes “Klacke” Wikstrom. <http://bsdtalk.blogspot.com/2006/08/bsdtalk062-interview-with-yaws.html> bsdtalk interview yaws claes wikstrom <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk062.mp3> 8 Mb 18 minutes MP3 version mp3 bsdtalk062.ogg 18 minutes Ogg version ogg

Interview with lighttpd developer Jan Kneschke

Interview with lighttpd developer Jan Kneschke. <http://bsdtalk.blogspot.com/2006/08/bsdtalk061-interview-with-lighttpd.html> bsdtalk interview lighttpd jan kneschke <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk061.mp3> 17 Mb 35 minutes MP3 version bsdtalk interview lighttpd jan kneschke bsdtalk061.ogg 35 minutes Ogg version bsdtalk interview lighttpd jan kneschke

My BSD History

My BSD History, by Will Backman of BSDTalk, and a bit on accessibility. <http://bsdtalk.blogspot.com/2006/08/bsdtalk060-my-bsd-history.html> bsdtalk accessibility <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk060.mp3> 5 Mb 10 minutes MP3 version mp3 bsdtalk060.ogg 10 minutes Ogg version ogg

Interview with Matt Morley

Interview with Matt Morley, BSD user. <http://bsdtalk.blogspot.com/2006/08/bsdtalk059-interview-with-matt-morley.html> bsdtalk interview matt morley <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk059.mp3> 11 Mb 25 minutes MP3 version mp3 bsdtalk059.ogg 25 minutes Ogg version ogg

Interview with Jason Thaxter from gomoos.org

Interview with Jason Thaxter from gomoos.org. <http://bsdtalk.blogspot.com/2006/07/bsdtalk058-interview-with-jason.html> bsdtalk interview gomoos jason thaxter <http://cisx1.uma.maine.edu/~wbackman/bsdtalk/bsdtalk058.mp3> 11 Mb 23 minutes MP3 version mp3 bsdtalk058.ogg 23 minutes Ogg version ogg

Using BSD in SchmooCon Labs

Using BSD in SchmooCon Labs

DCBSDCon 2009, Ken Caruso

clive URL: <http://www.youtube.com/watch?v=9ZhfuP4jghY> <http://www.youtube.com/watch?v=9ZhfuP4jghY> youtube presentation dcbsdcon dcbsdcon2009 bsd schmoocon ken caruso <http://www.youtube.com/watch?v=9ZhfuP4jghY> 35:08 Flash flash

Sleeping Beauty - NetBSD on Modern laptops

P9A: Sleeping Beauty - NetBSD on Modern Laptops

AsiaBSDCon 2008, Jorg Sonnenberger

clive URL: <http://www.youtube.com/watch?v=v9ygBFjGR50> <http://www.youtube.com/watch?v=v9ygBFjGR50> youtube presentation asiabsdcon2008 asiabsdcon netbsd laptops jorg sonnenberger <http://www.youtube.com/watch?v=v9ygBFjGR50> 1:20:56 Flash flash

OpenBSD Network Stack Internals

P8A: OpenBSD Network Stack Internals

AsiaBSDCon 2008, Claudio Jeker

clive URL: <http://www.youtube.com/watch?v=V85It0dGUF4> <http://www.youtube.com/watch?v=V85It0dGUF4> youtube presentation asiabsdcon2008 asiabsdcon openbsd claudio jeker <http://www.youtube.com/watch?v=V85It0dGUF4> 53:41 Flash flash

25 years with BSD

Thinking RealSpace: Life with BSD - ~25 years with BSD

AsiaBSDCon 2008, Hideki Sunahara

clive URL: <http://www.youtube.com/watch?v=brYdkQ120Do>

<http://www.youtube.com/watch?v=brYdkQ120Do> youtube keynote asiabsdcon2008 asiabsdcon bsd hideki sunahara <http://www.youtube.com/watch?v=brYdkQ120Do> 44:43 Flash flash

P6A: A Portable iSCSI Initiator

P3B: A Portable iSCSI Initiator

AsiaBSDCon 2008, Alistair Crooks

clive URL: <http://www.youtube.com/watch?v=MiZY7PMu7Ic>

<http://www.youtube.com/watch?v=MiZY7PMu7Ic> youtube presentation asiabsdcon2008 asiabsdcon iscsi alistair crooks <http://www.youtube.com/watch?v=MiZY7PMu7Ic> 40:57 Flash flash

P3B: BSD Implementations of XCAST6

P3B: BSD Implementations of XCAST6

AsiaBSDCon 2008, Yuji Imai

clive URL: <http://www.youtube.com/watch?v=g1Ga48smqyI> <http://www.youtube.com/watch?v=g1Ga48smqyI>

youtube presentation asiabsdcon2008 asiabsdcon xcast6 yuji imai

<http://www.youtube.com/watch?v=g1Ga48smqyI> 55:42 Flash flash

P5A: Logical Resource Isolation in the NetBSD Kernel

P5A: Logical Resource Isolation in the NetBSD Kernel

AsiaBSDCon 2008, Kristaps Dzonsons

clive URL: <http://www.youtube.com/watch?v=c63VneyQI-k> <http://www.youtube.com/watch?v=c63VneyQI-k>

youtube presentation asiabsdcon2008 asiabsdcon netbsd kristaps dzonsons

<http://www.youtube.com/watch?v=c63VneyQI-k> 56:29 Flash flash

P4B: Send and Receive of File System Protocols: Userspace Approach With puffs

P4B: Send and Receive of File System Protocols: Userspace Approach With puffs

AsiaBSDCon 2008, Antti Kantee

clive URL: <http://www.youtube.com/watch?v=ziGeB8iRA0c> <http://www.youtube.com/watch?v=ziGeB8iRA0c>

youtube presentation asiabsdcon2008 asiabsdcon puffs antti kantee

<http://www.youtube.com/watch?v=ziGeB8iRA0c> 47:29 Flash flash

P1B: Tracking FreeBSD in a Commercial Setting

P1B: Tracking FreeBSD in a Commercial Setting

AsiaBSDCon 2008, M. Warner Losh

clive URL: <http://www.youtube.com/watch?v=VaZ9Ef04bJg> <http://www.youtube.com/watch?v=VaZ9Ef04bJg>
youtube presentation asiabsdcon2008 asiabsdcon freebsd warner losh
<http://www.youtube.com/watch?v=VaZ9Ef04bJg> 33:40 Flash flash

A Brief History of the BSD Fast Filesystem, Kirk McKusick

A Brief History of the BSD Fast Filesystem, Kirk McKusick

AsiaBSDCon 2008, Dr. Kirk McKusick

clive URL: <http://www.youtube.com/watch?v=tzieR5MM06M>
<http://www.youtube.com/watch?v=tzieR5MM06M> youtube presentation asiabsdcon2008 asiabsdcon bsd fast
filesystem kirk mckusick <http://www.youtube.com/watch?v=tzieR5MM06M> 42:01 Flash flash

PC-BSD, Matt Olander, AsiaBSDCon 2008

PC-BSD, Matt Olander, AsiaBSDCon 2008

clive URL: <http://www.youtube.com/watch?v=N0q37X-MJzY>
<http://www.youtube.com/watch?v=N0q37X-MJzY> youtube presentation asiabsdcon2008 asiabsdcon pc-bsd
matt olander <http://www.youtube.com/watch?v=N0q37X-MJzY> 28:50 Flash flash

Using FreeBSD to Promote Open Source Development Methods, Brooks Davis, AsiaBSDCon 2008

Using FreeBSD to Promote Open Source Development Methods, Brooks Davis, AsiaBSDCon 2008

clive URL: <http://www.youtube.com/watch?v=4lcrinKBMas> <http://www.youtube.com/watch?v=4lcrinKBMas>
youtube presentation asiabsdcon2008 asiabsdcon freebsd promotion open source development models brooks
davis <http://www.youtube.com/watch?v=4lcrinKBMas> 30:07 Flash flash

Keynote, Peter Losher, Internet Systems Consortium, AsiaBSDCon 2008

Keynote, Peter Losher, Internet Systems Consortium, AsiaBSDCon 2008

clive URL: <http://www.youtube.com/watch?v=vQbdG7TwhKo>
<http://www.youtube.com/watch?v=vQbdG7TwhKo> youtube keynote asiabsdcon2008 asiabsdcon peter losher
<http://www.youtube.com/watch?v=vQbdG7TwhKo> 42:44 Flash flash

GEOM - in Infrastructure We Trust, Pawel Jakub Dawidek, AsiaBSDCon 2008

GEOM - in Infrastructure We Trust, Pawel Jakub Dawidek, AsiaBSDCon 2008

clive URL: <http://www.youtube.com/watch?v=xMpmOezBJZo>
<http://www.youtube.com/watch?v=xMpmOezBJZo> youtube presentation asiabsdcon2008 asiabsdcon geom
pawel jakub dawidek <http://www.youtube.com/watch?v=xMpmOezBJZo> 46:38 Flash flash

Reducing Lock Contention in a Multi-Core System, Randall Stewart, AsiaBSDCon 2008

Reducing Lock Contention in a Multi-Core System, Randall Stewart, AsiaBSDCon 2008

clive URL: <http://www.youtube.com/watch?v=OQOMva1SmbY>

<http://www.youtube.com/watch?v=OQOMva1SmbY> youtube presentation asiabsdcon2008 asiabsdcon multicore lock contention randall stewart <http://www.youtube.com/watch?v=OQOMva1SmbY> 28:12 Flash flash

FreeBSD Kernel Internals, Dr. Marshall Kirk McKusick

The first hour of Marshall Kirk McKusick's course on FreeBSD kernel internals based on his book, The Design and Implementation of the FreeBSD Operating System. This course has been given at BSD Conferences and technology companies around the world.

clive URL: <http://www.youtube.com/watch?v=nwbqBdghh6E>

<http://www.youtube.com/watch?v=nwbqBdghh6E> youtube course freebsd design and implementation of the freebsd operating system kirk mckusick <http://www.youtube.com/watch?v=nwbqBdghh6E> 59:57 Flash flash

May 2008 developer Vimage report

A sneak peak into the FreeBSD development process.

Warning 2 hours! filmed over 2 days. (The schedule worked out was optimistic to say the least but it's still looking ok...)

Marko Zec and Julian Elischer report back to the developers at BSDCan on the progress on virtualizing the network stack in FreeBSD. This has been a long term project but at the time of this recording was just reaching the point of feasibility. In this video you can see some of the dynamics of the group as developers become familiar with the project and discussions take place regarding such things as maintainability, ABI compatibility, and even what to call the feature. In this video you can see the decision being made by a "quorum" of developers to take this project mainstream.

The sound is less than perfect, but it's what we have.

This is a montage of 3 video sources, one of which is a lower resolution, but at times it was the only camera capturing the action. (the other ran out of tape for a while)

Thanks to Ed Maste for the added footage.

I will be doing more editing later and will be substituting in better footage in some places.

clive URL: <http://au.youtube.com/watch?v=Px-pSXm32dE> <http://www.youtube.com/watch?v=Px-pSXm32dE> youtube freebsd vimage marko zec julian elischer <http://www.youtube.com/watch?v=Px-pSXm32dE> 2:44:36 Flash flash

ZFS in FreeBSD, by Pawel Jakub Dawidek

Pawel goes over ZFS, and tells us the state of the FreeBSD port. Source: Julian

clive URL: <http://au.youtube.com/watch?v=5-CR3o-Q2CU> <http://www.youtube.com/watch?v=5-CR3o-Q2CU> youtube freebsd zfs pawel jakub <http://www.youtube.com/watch?v=5-CR3o-Q2CU> 54:34 Flash flash

Isilon and FreeBSD

Zach Loafman explains how Isilon uses FreeBSD and how the company adds to it and interacts with the FreeBSD

community.

clive URL: <http://au.youtube.com/watch?v=OIMocIwM5QU>
<http://www.youtube.com/watch?v=OIMocIwM5QU> youtube freebsd isilon zach loafman
<http://www.youtube.com/watch?v=OIMocIwM5QU> 28:58 Flash flash

FreeBSD networking work summary

Robert Watson reports on work currently under way to optimize the networking stack for new hardware. Source: Julian

clive URL: <http://www.youtube.com/watch?v=ohLVNmI3ICg>
<http://www.youtube.com/watch?v=ohLVNmI3ICg> youtube freebsd networking robert watson
<http://www.youtube.com/watch?v=ohLVNmI3ICg> 55:21 Flash flash

Kris Moore and PCBSD

PCBSD from a developer's perspective. Source: Julian

clive URL: <http://au.youtube.com/watch?v=aHRRa-OvwxM> <http://au.youtube.com/watch?v=aHRRa-OvwxM>
youtube pcbsd kris moore <http://au.youtube.com/watch?v=aHRRa-OvwxM> 25:14 Flash flash

FreeBSD, klaster pocztowy

“Projektowanie korporacyjnego klastra pocztowego”, Jan Srzednicki at MeetBSD 2007 in Warsaw, Poland.

clive URL: <http://www.youtube.com/watch?v=9B8MDy-37TI>
<http://www.youtube.com/watch?v=9B8MDy-37TI> youtube meetbsd meetbsd2007 polish jan srzednicki
<http://www.youtube.com/watch?v=9B8MDy-37TI> 1:07:56 Flash flash

Meet BSD projects from GSoC 2007

“Meet BSD projects from Google Summer of Code 2007”, Pawel Solyga at MeetBSD 2007 in Warsaw, Poland.

clive URL: <http://www.youtube.com/watch?v=snVtilaj-KI> <http://www.youtube.com/watch?v=snVtilaj-KI>
youtube meetbsd meetbsd2007 google soc pawel solyga <http://www.youtube.com/watch?v=snVtilaj-KI> 34:37
Flash flash

Google Summer of Code 2008. BSD summary

A panel discusses the GSOC project and how it and BSD get on. Source: Julian

clive URL: http://www.youtube.com/watch?v=3I3tuhSmp_E http://www.youtube.com/watch?v=3I3tuhSmp_E
youtube meetbsd meetbsd2008 google soc http://www.youtube.com/watch?v=3I3tuhSmp_E 35:15 Flash flash

Embedded FreeBSD

“FreeBSD do zabudowy czyli nie tylko pecety”, Rafal Jaworowski at MeetBSD 2007 in Warsaw, Poland.

clive URL: <http://www.youtube.com/watch?v=2HcIJvJX4y8> <http://www.youtube.com/watch?v=2HcIJvJX4y8>
youtube meetbsd meetbsd2007 embedded freebsd polish rafal jaworowski
<http://www.youtube.com/watch?v=2HcIJvJX4y8> 1:11:09 Flash flash

DTrace

“DTrace - Monitoring i strojenie systemu w XXI wieku”, Slawomir Zak at MeetBSD 2007 in Warsaw, Poland.

clive URL: <http://www.youtube.com/watch?v=5VK6tV4y3r0>
<http://www.youtube.com/watch?v=5VK6tV4y3r0> youtube meetbsd meetbsd2007 dtrace polish slawomir zak
<http://www.youtube.com/watch?v=5VK6tV4y3r0> 1:04:23 Flash flash

New features in FreeBSD 7

“New features and improvements in FreeBSD 7”, Kris Kennaway at MeetBSD 2007 in Warsaw, Poland

clive URL: <http://www.youtube.com/watch?v=XUjJWhlnujQ>
<http://www.youtube.com/watch?v=XUjJWhlnujQ> youtube meetbsd meetbsd2007 freebsd kris kennaway
<http://www.youtube.com/watch?v=XUjJWhlnujQ> 1:07:18 Flash flash

Detangling and debugging

“Detangling and debugging: friends in unexpected places”, Philip Paeps at MeetBSD 2007 in Warsaw, Poland.

clive URL: <http://www.youtube.com/watch?v=G8Fm8mgPyDc>
<http://www.youtube.com/watch?v=G8Fm8mgPyDc> youtube meetbsd meetbsd2007 debugging philip paeps
<http://www.youtube.com/watch?v=G8Fm8mgPyDc> 18:36 Flash flash

FreeBSD Profiling, Kris Kennaway, MeetBSD 2008

FreeBSD Profiling tools, tips and tricks, Kris Kennaway, MeetBSD 2008

clive URL: http://www.youtube.com/watch?v=Mfb5_uG7BCA
http://www.youtube.com/watch?v=Mfb5_uG7BCA youtube meetbsd meetbsd2008 freebsd profiling kris
kennaway http://www.youtube.com/watch?v=Mfb5_uG7BCA 1:06:23 Flash flash

BSD v. GPL, Jason Dixon, NYCBSDCon 2008

BSD vs GPL is a sweeping epic, focused on the dichotomy between good and evil. It peers inside the hearts and minds of the creators of these movements and dissects their battle for world domination. No common documentary will dare to follow the path that BSD vs GPL blazes. This presentation was given by Jason Dixon at the NYC BSD Conference at Columbia University on October 11, 2008

clive URL: <http://www.youtube.com/watch?v=mMmbjJI5su0>
<http://www.youtube.com/watch?v=mMmbjJI5su0> youtube nycbsdcon nycbsdcon2008 bsd versus gpl jason
dixon <http://www.youtube.com/watch?v=mMmbjJI5su0> 16:21 Flash flash

BSD is Dying, Jason Dixon, NYCBSDCon 2007

A tongue-in-cheek look at the history and future of the BSD movement. Modeled after the presentation styles of Lessig and Hardt, the talk provides a light-hearted introspection of the leaders, technologies, and community that forges ahead despite having been left for dead some 15 years past. This presentation was given by Jason Dixon at the NYC BSD Conference at Columbia University on October 28, 2006

clive URL: <http://www.youtube.com/watch?v=g7tvI6JCXD0> <http://www.youtube.com/watch?v=g7tvI6JCXD0>
youtube nycbsdcon nycbsdcon2007 bsd is dying jason dixon <http://www.youtube.com/watch?v=g7tvI6JCXD0>
17:41 Flash flash

PC-BSD: FreeBSD on the Desktop

“PC-BSD: FreeBSD on the Desktop”, Matt Olander at MeetBSD 2007 in Warsaw, Poland.

clive URL: <http://www.youtube.com/watch?v=oC4gsipGfQU>
<http://www.youtube.com/watch?v=oC4gsipGfQU> youtube meetbsd meetbsd2007 pc-bsd matt olander
<http://www.youtube.com/watch?v=oC4gsipGfQU> 31:30 Flash flash

FreeBSD, Protecting Privacy with Tor

“Protecting your Privacy with FreeBSD and Tor”, Christian Brüffer at MeetBSD 2007 in Warsaw, Poland.

clive URL: <http://www.youtube.com/watch?v=OwBh8ro7xHQ>
<http://www.youtube.com/watch?v=OwBh8ro7xHQ> youtube meetbsd meetbsd2007 freebsd tor privacy
christian bruffer <http://www.youtube.com/watch?v=OwBh8ro7xHQ> 46:24 Flash flash

FreeBSD, Building a Computing Cluster

“Reflections on Building a High-Performance Computing Cluster using FreeBSD”, Brooks Davis at MeetBSD 2007 in Warsaw, Poland.

clive URL: <http://www.youtube.com/watch?v=BpsRb9fJ4Ds> <http://www.youtube.com/watch?v=BpsRb9fJ4Ds>
youtube meetbsd meetbsd2007 freebsd cluster performance brooks davis
<http://www.youtube.com/watch?v=BpsRb9fJ4Ds> 47:51 Flash flash

Isolating Cluster Jobs for Performance and Predictability, Brooks Davis, MeetBSD 2008

Isolating Cluster Jobs for Performance and Predictability by Brooks Davis, The Aerospace Corporation, MeetBSD November 15, 2008

clive URL: <http://www.youtube.com/watch?v=0uBFLJm7IHc>
<http://www.youtube.com/watch?v=0uBFLJm7IHc> youtube meetbsd meetbsd2008 cluster performance brooks
davis <http://www.youtube.com/watch?v=0uBFLJm7IHc> 43:40 Flash flash

BSD Certification, MeetBSD 2008

BSD Certification by Dru Lavigne, Chair, BSD Certification Group, MeetBSD November 15, 2008

clive URL: <http://www.youtube.com/watch?v=rGQmLYplO9U>

<http://www.youtube.com/watch?v=rGQmLYplO9U> youtube meetbsd meetbsd2008 bsd certification dru lavigne <http://www.youtube.com/watch?v=rGQmLYplO9U> 44:14 Flash flash

Embedding FreeBSD, MeetBSD 2008

Embedding FreeBSD by Warner Losh and Philip Paeps, MeetBSD November 15, 2008

clive URL: <http://www.youtube.com/watch?v=Fc3xYrxvIU0> <http://www.youtube.com/watch?v=Fc3xYrxvIU0> youtube meetbsd meetbsd2008 embedded freebsd philip paeps warner losh <http://www.youtube.com/watch?v=Fc3xYrxvIU0> 38:56 Flash flash

FreeBSD Foundation Update & Recognition, MeetBSD 2008

Robert Watson provides a status update on the non-profit FreeBSD Foundation at MeetBSD November 16, 2008

clive URL: <http://www.youtube.com/watch?v=sNQ2d41Vn2A>

<http://www.youtube.com/watch?v=sNQ2d41Vn2A> youtube meetbsd meetbsd2008 freebsd foundation robert watson <http://www.youtube.com/watch?v=sNQ2d41Vn2A> 16:22 Flash flash

Lousy virtualization, Happy users: FreeBSD's jail(2) facility

Lousy virtualization, Happy users: FreeBSD's jail(2) facility by Poul-Henning Kamp (phk@FreeBSD.org) <http://www.ukuug.org/events/spring2007/programme/> ukuug presentation freebsd jails poul-henning kamp <http://www.ukuug.org/events/spring2007/programme/jails.pdf> 2.7 Mb Slides pdf

Poul-Henning Kamp - GBDE – Spook strength disk encryption

GBDE is a disk encryption facility designed with both usability and strength as requirements and it attempts to protect both the user and the data. The talk is about avoiding self-deceiving analysis, how to make real world usable cryptography and generally protect yourself and your data. Required skill level: Laptop user. <http://conferences.suug.ch/sucon/04/> suug presentation gbde poul-henning kamp <http://phk.freebsd.dk/pubs/bsdcon-03.gbde.paper.pdf> 104 Kb Paper pdf <http://www.suug.ch/sucon/04/slides/gbde.pdf> 113 Kb Slides pdf

Max Laier - PF - Extended Introduction

The talk will introduce packet filter (pf) - a *BSD firewall system - and summarize its history and projected future. After providing a short overview of pf's general functionality and some firewall basics, it will concentrate on packet filter's advanced feature-set from the administrator's point of view. The talk will also cover the integration of ALTQ, a mature framework for traffic shaping and prioritization. Finally it will provide a short overview of the "Common Address Redundancy Protocol" (CARP) and its integration in pf. <http://conferences.suug.ch/sucon/04/> suug presentation pf altq max laier <http://people.freebsd.org/~mlaier/sucon.pdf> 1 Mb Slides pdf http://mirror.switch.ch/sucon-04/max_laier-pf_extended_introduction.avi 94 Mb Video/MPEG avi http://mirror.switch.ch/sucon-04/max_laier-pf_extended_introduction.mp3 22 Mb Audio/MP3 mp3

Poul-Henning Kamp - Old mistakes repeated (but you do get the source code now)

UNIX is the best operating system ever designed so everybody is running UNIX on their computer, right ? This presentation takes a partisan look at why UNIX never became a big success in the eighties, failed to win the market in the nineties, and still struggles in the market in the new millennium. Poul-Henning will take a critical look at the mistakes of the past and the mistakes of the present and try to make it really clear what needs to happen for UNIX to become a real success. <http://conferences.suug.ch/sucon/04/> suug presentation unix mistakes poul-henning kamp <http://www.suug.ch/sucon/04/slides/oldmistakes.pdf> 65 Kb Slides pdf

DCBSDCon 2009 - Photos

Photos of the 2009 DCBSDCon <http://www.flickr.com/photos/34727619@N03/dcbsdcon> dcbsdcon dcbsdcon2009 photos

EuroBSDCon 2008 - Paeps Philip - How-to embed FreeBSD

This paper provides a how-to embed FreeBSD. A console server built from an AT91RM9200 based ARM system will be explored. This paper will talk about the selection of hardware. It will explore creating images for the target system, as well as concentrate on different alternatives for deploying the system. A number of different options exist today, and no comprehensive guide for navigating through the choices exists today. This paper will explore the different alternatives that exist today for producing images targeted at different size requirements. The differing choices for storage in an embedded environment are explored. The techniques used to access rich debugging environments are discussed. <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 embed freebsd philip paeps <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2828&type=ogg> OGG 1 byte 43 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2828&type=mp3> MP3 1 byte 43 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2828&type=pdf> PDF 1 byte 17 pages pdf

EuroBSDCon 2008 - George Neville-Neil - Multicast Performance in FreeBSD

In the past ten years most of the research in network protocols has gone into TCP, leaving UDP to languish as a local configuration protocol. While the majority of Internet traffic is TCP, UDP remains the only IP protocol that works over multicast and as such has some specific, and interesting uses in some areas of computing. In 2008 we undertook a study of the performance of UDP multicast on both 1Gbps and 10Gbps Ethernet networks in order to see if changing the physical layer of the network would give a linear decrease in packet latency. To measure the possible gains we developed a new network protocol test program, mctest, which is capable of recording packet round trip times from many hosts simultaneously and which we believe accurately represents how many environments use multicast. The mctest program has been integrated into FreeBSD and is now being used to verify the proper operation of multicast on various pieces of 10Gbps hardware. <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 multicast freebsd george neville-neil <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2827&type=ogg> OGG 1 byte 39 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2827&type=mp3> MP3 1 byte 39 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2827&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Pedro Giffuni - Working with Engineering Applications in FreeBSD

In recent years, traditional branches of engineering like Civil, Chemical, Mechanical, Electrical and Industrial Engineering are requiring extensive computing facilities for their needs. Several well known labs (Sandia, Lawrence Livermore) rely on huge clusters to do all types of complex analysis that were unthinkable a couple of decades ago. While the free BSD variants share the environment with traditional UNIX systems, frequently used for such computations, it was not common to find adequate free software packages to carry complex calculations. Eventually commercial versions of important math related packages started to appear for the Linux platform. Even when the big packages were distant, the BSDs learned and adapted in resourceful ways: Matlab and Mathematica, running under Linux emulation, demanded functionality from the BSDs and NetBSD implemented a signal trampoline to be able to run AutoCAD with IRIX binary compatibility. A notable project that was always available under a free license was Berkeley's Spice circuit analysis program, however it was an exception rather than the rule. Even when the scientific community pressed for a while to get other important tools like NASA's FEA package Nastran under a free license, the objective of being able to access and enhance open scientific tools was elusive. About a decade ago the situation started to improve: FreeBSD's ports system started growing exponentially, first with a high content in the math category, afterwards with a CAD section and after sustained growth in those categories a science section was created. This growth was mostly pushed by Universities and their research projects and in general are not well known with respect to the commercial counterparts. I started porting math/engineering code for FreeBSD around 1996. Back then it was absolutely unthinkable for a Mechanical Engineer to depend only on FreeBSD for it's daily work. The situation nowadays is different: there are some very high quality engineering analysis packages like EDF's Code Aster, with more than 12 years of professional development, that just can't be ignored. A Finite Element package, like Code Aster, can easily cost 5000 US\$, is priced according to the maximum problem size it can solve, can require yearly licenses, and is rarely available with source code. In NASTRAN's case the source code is only available for US citizens under a yearly fee. Free software does have serious limitations though; just like in office applications

there are proprietary CAD formats or sometimes the package simply doesn't have the required functionality. Having the sources, of course, always has the advantage of being able to implement (or pay for) some specific functionality you might need. Many commercial packages have been recently ported to Linux, but even when they gain some of the advantages of an open environment they still have yet another limitation: they have been very slow to make use of the multicore features of the new processors in the market, a huge limitation now that the speed war between processors has been limited by the overheating problem. The objective of the talk is to give an overview of several CAD/CAE packages that have been made available recently as part of FreeBSD's ports system and the decisions that were made to port them. BRLCAD and Varkon are two CAD utilities that made a transition from closed source to an open environment and in the process in the process of getting ported to BSD have gained greater portability and general "bug" fixes critical for their consolidation as usable and maintainable projects. There are also some tricks that have not been well documented: it is possible to enable threads and some extra optimizations on some packages, and it is also possible to replace the standard BLAS library with the faster GOTO BLAS without rebuilding the package. It is also possible to build the packages optimized for a clustered environment, but perhaps what is most interesting of all is how all the packages interrelate with each other and can turn FreeBSD into a complete engineering environment. No OS distribution so far is offering all the engineering specific utilities offered through FreeBSD's ports system: from design to visualization, passing through analysis FreeBSD is becoming an option that can't be ignored, and best of all, it is an effort that will benefit not only FreeBSD but the wider audience.

Pedro F. Giffuni M. Sc. Industrial Engineering - University of Pittsburgh Mechanical Engineer - Universidad Nacional de Colombia I was born in Bogota, Colombia but I am an Italian citizen. My experience with computers started when I was about 12 years old With the TRS-80 Color Computer first using Basic and the OS-9. I studied electronics for 3 years but became tired of worrying about "whatever happened to electrons in there" and moved to Mechanical Engineering. For a while I rested from the computer world until the Internet came stepping along. I started using FreeBSD around 1995 and soon fell in love with the idea of being able to install a complete version of UNIX from the net with just one floppy. After submitting a the 999th port to the FreeBSD project Walnut Creek was kind enough to give me a subscription for several years to FreeBSD's CD-ROM. Since then I've been on and off porting software packages or fixing the bugs I have caused while porting them. Of course there has always been great respect for the other BSDs and their wonderful license and while I've given up on the idea of one day seeing a "UnifiedBSD" I am glad to see different approaches sharing ideas in a healthful environment.

Keywords: BSD, engineering, CAE, CAD, math, mechanical, FreeBSD ports

<http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 freebsd engineering applications pedro giffuni <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2826&type=ogg> OGG 1 byte 51 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2826&type=mp3> MP3 1 byte 51 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2826&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Constantine Murenin - OpenBSD Hardware Sensors Framework

In this talk, we will discuss the past and present history and the design principles of the OpenBSD hardware sensors framework. Sensors framework provides a unified interface for storing, registering and accessing information about hardware monitoring sensors. Sensor types include, but are not limited to, temperature, voltage, fan RPM, time offset and logical drive status. The framework spans sensor_attach(9), sysctl(3), sysctl(8), sensorsd(8), ntpd(8), snmpd(8) and more than 67 drivers, ranging from I2C temperature sensors and Super I/O hardware monitors to IPMI, RAID and SCSI enclosures. Several third-party tools are also available, for example, a plug-in for Nagios and ports/sysutils/symon. Originally based on some ideas from NetBSD, the framework has sustained many improvements in OpenBSD, and was ported and committed to FreeBSD and DragonFly BSD.

Constantine A. Murenin is an MMath graduate student at the David R. Cheriton School of Computer Science at the University of Waterloo (CA). Prior to his graduate appointment, Constantine attended and subsequently graduated from East Carolina University (US) and De Montfort University (UK), receiving two bachelor degrees in computer science, with honors and honours respectively. A FreeBSD Google Summer of Code 2007 Student, OpenBSD Committer and Mozilla Contributor, Constantine's interests range from standards compliance and usability at all levels, to quiet computing and hardware monitoring.

<http://Constantine.SU/> <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 presentation
openbsd hardware sensors constantine murenin <http://www.openbsd.org/papers/eurobsdcon2008-sensors.pdf>
PDF 539395 bytes 38 pages pdf

EuroBSDCon 2008 - Ion-Mihai Tetcu - Improving FreeBSD ports/packages quality

This talk is focused on ways to improve the quality of FreeBSD's ports and packages and it's partially based on the 5 months experience of writing and running the consecutive versions of "QA Tindy".

Ion-Mihai "IONut" Tetcu is a 28 years old FreeBSD ports committer and maintains about 40 ports scattered in the Ports Tree. He lives in Bucharest, Romania where he runs and co-owns an IT company and he's a member of Romanian FreeBSD and FreeUnix User Group (RoFUG). His non-IT interests include history, philosophy and mountain climbing. <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 freebsd ports packages ion-mihai tetcu <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2824&type=ogg> OGG 1 byte 56 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2824&type=mp3> MP3 1 byte 56 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2824&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Yvan Vanhullebus - IPSec tools: past, present and future

The first part will explain what have been major changes since Manu's presentation at Bale's EuroBSDCon, including more detailed informations on changes which have a significant impact on administrator's bad habits (why the common way of doing it is bad, why it was sometimes needed in the past, how to do it the good way now, why this is far better), on both the UserLand (ipsec-tools project) and maybe in [Free|Net]BSD kernels/ IPSec stacks.

The second part will talk about the future of the project. News of the next major version (which may be out or about to be out when we'll be at EuroBSDCon), news works which are planned or which are done but not yet public, but also news about the team: it's new members, new tools, what we would like to do in the future, a Yvan VANHULLEBUS works as an R&D security engineer for NETASQ since 2000, where he works on FreeBSD OS. He started to work on KAME's IPSec stack in 2001, provided many patches for various parts of the stack, then became one of the maintainers of ipsec-tools project, a fork of KAME's userland daemon. He became a NetBSD developer when ipsec-tools was migrated to NetBSD's CVS.

<http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 ipsec yvan vanhullebus
<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2823&type=ogg> OGG 1 byte 46 minutes ogg
<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2823&type=mp3> MP3 1 byte 46 minutes mp3
<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2823&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 Keynote - George Neville-Neil - Thinking about thinking code

EuroBSDCon 2008 Keynote - George Neville-Neil - Thinking about thinking code
<http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 george neville-neil <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2822&type=ogg> OGG 1 byte 37 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2822&type=mp3> MP3 1 byte 37 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2822&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Robert Watson - FreeBSD Network Stack Performance Optimizations for Modern Hardware

The arrival of high CPU core density, with commodity quad-core notebooks and 32-core servers, combined with 10gbps networking have transformed network design principles for operating systems. This talk will describe changes in the FreeBSD 6.x, 7.x, and forthcoming 8.x network stacks required to exploit multiple cores and serve 10gbps networks. The goal of the session will be to introduce the audience to general strategies used to improve performance, their rationales, and their impact on applications and users:

- Introduction to the SMPng Project and the follow-on Netperf Project
- Workloads and performance measurement
- Efficient primitives to support modern network stacks
- Multi-core and cache-aware network memory allocator
- Fine-grained network stack locking
- Load-balancing and contention-avoidance across multiple CPUs
- CPU affinity for network stack data structures
- TCP performance enhancements including TSO, LRO, and TOE
- Zero-copy Berkely Packet Filter (BPF) buffers
- Direct network stack dispatch from interrupt handlers
- Multiple input and output queues

Robert Watson is a researcher at the University of Cambridge Computer Laboratory investigating operating system and network security. Prior to joining the Computer Laboratory to work on a PhD, he was Senior Principal Scientist at McAfee Research, now SPARTA ISSO, a leading security research and development organization, directing government and commercial research contracts for customers that include DARPA, the US Navy, and Apple Computer. His research interests include operating system security, network stack structure and performance, and windowing system structure. He is also a member of the FreeBSD Core Team and president of the FreeBSD Foundation. <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 freebsd network stack hardware robert watson
<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2821&type=ogg> OGG 1 byte 53 minutes ogg
<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2821&type=mp3> MP3 1 byte 53 minutes mp3
<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2821&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Martin Schuette - Improved NetBSD Syslogd

Martin Schuette has three main goals, defined by three internet drafts to implement:

- TLS transport is the most obvious improvement: it provides a reliable network transport with data encryption and peer authentication. To make full use of this a buffering mechanism to bridge temporary network errors is implemented as well.
- Syslog-protocol extends the message format to use a complete timestamp, include a fully qualified domain name, and allow UTF-8 messages. It also offers a structured data field to unambiguously encode application dependent information.
- Syslog-sign will allow any syslog sender to digitally sign its messages, so their integrity can be verified later. This enable the detection of loss, deletion or other manipulation syslog data after network transfer or archiving on storage media.

Martin Schuette is a student of computer science in Potsdam, Germany, and has been working as a part-time system administrator for BSD servers since 2004.

In 2007 Martin Schuette already gave a talk on Syslog at the Chemnitze Linux-Tage

(<http://chemnitzer.linux-tage.de/2007/vortraege/detail.html?idx=547> in german; for a newer english version see these slides for a seminar talk: <http://fara.cs.uni-potsdam.de/~mschuett/uni/syslog-protocols-080522.pdf>).

<http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 netbsd syslogd martin schuette
<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2820&type=ogg> OGG 1 byte 42 minutes ogg
<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2820&type=mp3> MP3 1 byte 42 minutes mp3
<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2820&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Aggelos Economopoulos - An MP-capable network stack for DragonFlyBSD with minimal use of locks

Given the modern trend towards multi-core shared memory multiprocessors, it is inconceivable for production OS kernels not to be reentrant. The typical approach for allowing multiple execution contexts to simultaneously execute in kernel mode has been to use fine-grained locking for synchronising access to shared resources. While this technique has been proven efficient, empirical evidence suggests that the resulting locking rules tend to be cumbersome even for the experienced kernel programmer, leading to bugs that are hard to diagnose. Moreover, scaling to more processors requires extensive use of locks, which may impose unnecessary locking overhead for small scale multiprocessor systems. This talk will describe the typical approach and then discuss the alternative approach taken in the DragonFlyBSD network stack. We will give an overview of the various protocol threads employed for network I/O processing and the common-case code paths for packet reception and transmission. Additionally, we'll need to make a passing reference to DragonFlyBSD's message passing model. This should establish a baseline, allowing us to focus on the recent work by the author to eliminate use of the Big Giant Lock in the performance-critical paths for the TCP and UDP protocols. The decision to constrain this work on the two by far most widely-used transport protocols was made in order to (a) limit the amount of work necessary and (b) explore the effectiveness of the approach on the cases that matter at this point in time. <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 dragonflybsd mp network stack aggelos economopoulos <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2817&type=ogg> OGG 1 byte 42 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2817&type=mp3> MP3 1 byte 42 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2817&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Edd Barret - Modern Typesetting on BSD

Edd Barrett will speak about using the BSD Platform as a means of typesetting from a practical standpoint at EuroBSDcon 2008. Edd Barrett does not wish to go into the technicalities of each typesetter, but rather state which are good for certain types of document, and which tools (ports and packages), integrate well with the available typesetters.

Edd Barrett is a student from the UK, currently on "placement year" as a systems administrator for Bournemouth University. Open Source *NIX has been his platform of choice for many years and he has been using OpenBSD for about 3 years now, simply because it is small, clean, correct and secure. Just recently he has started developing things I want or need for OpenBSD. <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 typesetting bsd edd barrett
<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2816&type=ogg> OGG 1 byte 33 minutes ogg
<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2816&type=mp3> MP3 1 byte 33 minutes mp3
<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2816&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Michael Dexter - Zen and the Art of Multiplicity Maintenance: An applied survey of BSD-licensed multiplicity strategies from chroot to mult

Many BSD-licensed strategies of various levels of maturity exist to implement multiplicity, herein defined as the introduction of plurality to traditionally singular computing environments via isolation, virtualization, or other method. For example, the chroot utility introduces an additional isolated root execution environment within that of the host; or an emulator provides highly-isolated virtual systems that can run complete native or foreign operating systems. Motivations for multiplicity vary, but a demonstrable desire exists for users to obtain root or run a foreign binary or operating system. We propose a hands-on survey of portable and integrated BSD-licensed multiplicity strategies applicable to the FreeBSD, OpenBSD, DragonFlyBSD and NetBSD operating systems on the i386

architecture. We will also address three oft-coupled disciplines: software storage devices, the installation of operating system and userlands in multiplicity environments plus the management of select multiplicity environments. Finally we will comment on each strategies potential limits of isolation, compatibility, independence and potential overhead in comparison to traditional systems. Keywords: multiplicity, virtualization, chroot, jail, hypervisor, xen, compat.

Michael Dexter has used Unix systems since 1991 and BSD-licensed multiplicity strategies for over five years. He is the Program Manager at the BSD Fund and Project Manager of the BSD.lv Project.

<http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 bsd michael dexter

<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2815&type=ogg> OGG 1 byte 38 minutes ogg

<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2815&type=mp3> MP3 1 byte 38 minutes mp3

<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2815&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Nick Barkas - Dynamic memory allocation for dirhash in UFS2

Hello My name is Nick Barkas. I'm a master's student studying scientific computing at Kungliga Tekniska hgskolan (KTH) in Stockholm, Sweden. I have just begun work on a Google Summer of Code project with FreeBSD: Dynamic memory allocation for dirhash in UFS2 . I would like to present my results from this project at EuroBSDCon this year. This project is very much a work in progress now so it is a bit difficult to summarize what I would ultimately present. I will try to describe an outline, though. First I will give background information on dirhash: an explanation of the directory data structure in UFS2, how directory lookups in this structure necessitate a linear search, and how dirhash speeds these lookups up without having to change anything about the directory data structure. Next I will explain the current limitation that dirhash's maximum memory use must be manually specified by administrators, or left at a small conservative default of 2MB. I will explain some different methods I will have explored to try and make this maximum memory limit dynamically increase and decrease as the system has more or less free memory, and which method I will have ultimately settled on and implemented. Then I'll present some test results of performance of operations on very large directories with and without dynamic memory allocation enabled for dirhash. Next I will talk about how speed gains from dirhash are limited by the fact that the hash tables exist only in memory and must be recreated after each system boot, as big directories are scanned for the first time, or even have to be recreated for a directory that has not been scanned in some time if its dirhash has been discarded to free memory. These problems can be eliminated by using an on-disk index for directory entries. I will talk about some of the challenges of implementing on-disk indexing, such as remaining backwards compatible with older versions of UFS2 and interoperating properly with softupdates. Then, if my SoC project has permitted me time to work on this aspect of it, I will explain some possible methods for adding directory indexing to UFS2 that meets these challenges, and which of those ideas I will have implemented. Finally I will present results of some benchmarks on this filesystem with indices, and compare to performance with dirhash, and with no indices or dirhashes.

Keywords: dirhash, ufs2, filesystems, performance tuning <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 ufs2 nick barkas <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2814&type=ogg> OGG 1 byte 32 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2814&type=mp3> MP3 1 byte 32 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2814&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Paul Richards - eXtreme Programming: FreeBSD a case study

Traditional project management methodologies are typically based on the waterfall model where there are distinct phases: requirements capture, design, implementation, testing, delivery. Once a project has moved on to the next phase there is no going back. The end result is often a late project that no-one wants anymore because the requirements have fundamentally changed by the time the project is delivered. <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 freebsd extreme programming paul richards <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2813&type=ogg> OGG 1 byte 54 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2813&type=mp3> MP3 1 byte 54 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2813&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Hauke Fath - Managing BSD desktop clients - Fencing in the herd

The members of the BSD family have traditionally prospered off the desktop, as operating systems on servers and embedded systems. The advent of MacOS X has marked a change, and moved the desktop more into focus. Modern desktop systems create a richer software landscape, with more diverse requirements, than their server counterparts. User demands, software package interdependencies and frequent security issues result in a change rate that can put a considerable load on the admin staff. Without central management tools, previously identical installations diverge quickly. This paper looks at concepts and strategies for managing tens to hundreds of modern, Unix-like desktop clients. The available management tools range from simple, image-based software distribution, mainly used for setting up uniform clients, to “intelligent” rule-based engines capable of search-and-replace operations on configuration files. We will briefly compare their properties and limitations, then take a closer look at Radmin, a suite for file level administration of Unix clients. Radmin has been in use in the Institute of Telecommunication at Technische Universität Darmstadt for over three years, managing NetBSD and Debian Linux clients in the labs as well as faculty members’ machines. We will explore the Radmin suite’s underlying concepts and functionality. In order to see how the concept holds up, we will discuss real-world scenarios from the system life-cycle of Installation, configuration changes, security updates, component updates, and system upgrades.

Hauke Fath works as a systems administrator for the Institut für Nachrichtentechnik (telecommunication) at Technische Universität Darmstadt. He has been using NetBSD since 1994, when he first booted a NetBSD 1.0A kernel on a Macintosh SE/30. NetBSD helped shaping his career by causing a slow drift from application programmer’s work towards systems and network administration. Hauke Fath holds a MS in Physics and became a NetBSD developer in late 2006.

Keywords: Managing Unix desktop clients, software distribution, tripwire

<http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 bsd desktop hauke fath

<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2812&type=ogg> OGG 1 byte 50 minutes ogg

<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2812&type=mp3> MP3 1 byte 50 minutes mp3

<http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2812&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Joerg Sonnenberger - Sleeping beauty - NetBSD on Modern Laptops

This paper discusses the NetBSD Power Management Framework (PMF) and related changes to the kernel. The outlined changes allow NetBSD to support essential functions like suspend-to-RAM on most post-Y2K X86 machines. They are also the foundation for intelligent handling of device activity by enabling devices on-demand. This work is still progressing. Many of the features will be available in the up-coming NetBSD 5.0 release. The NetBSD kernel is widely regarded to be one of the cleanest and most portable Operating System kernels available. For various reasons it is also assumed that NetBSD only runs well on older hardware. In the summer of 2006 Charles Hannum, one of the founders of NetBSD, left with a long mail mentioning as important issues the lack of proper power management and suspend-to-RAM support. One year later, Jared D. McNeill posted a plan for attacking this issue based on ideas derived from the Windows Driver Model. This plan would evolve into the new NetBSD Power Management Framework (PMF for short). <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 netbsd laptops joerg sonnenberger <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2811&type=ogg> OGG 1 byte 54 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2811&type=mp3> MP3 1 byte 54 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2811&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Brooks Davis - Isolating cluster jobs for performance and predictability

The Aerospace Corporation operates a federally funded research and development center in support of national-security, civil and commercial space programs. Many of our 2400+ engineers use a variety of computing technologies to support their work. Applications range from small models which are easily handled by desktops to parameter studies involving thousands of cpu hours and traditional, large scale parallel codes such as computational fluid dynamics and molecular modeling applications. Our primary resources used to support these large applications are computing clusters. Our current primary cluster, the Fellowship cluster consists of 352 dual-processor nodes with a total of 14xx cores. Two additional clusters, beginning at 150 dual-processor nodes each are being constructed to augment Fellowship. As in any multiuser computing environment with limited resources, user competition for resources is a significant burden. Users want everything they need to do their job, right now. Unfortunately, other users may need those resources at the same time. Thus, systems to arbitrate this resource contention are necessary. On Fellowship we have deployed the Sun Grid Engine scheduler which sched-

uled batch jobs across the nodes. In the next section we discuss the performance problems that can occur when sharing resources in a high performance computing cluster. We then discuss range of possibilities to address these problems. We then explain the solutions we are investigating and describe our experiments with them. We then conclude with a discussion of future work. <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 freebsd cluster brooks davis <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2810&type=ogg> OGG 1 byte 51 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2810&type=mp3> MP3 1 byte 51 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2810&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Russel Sutherland - UTOVPN: A BSD based VPN service for the masses

The University of Toronto is a large educational institutional with over 70,000 students and 10,000 staff and faculty. For the past three years, we have developed and implemented a ubiquitous VPN service, based up on OpenVPN and FreeBSD. The service has over 3000 active customers, with up to 35 simultaneous users. The system supports, Linux, Mac OS X and Windows XP/Vista/2000 clients. Tools have been developed to create a central CA which enables users to log in to a secure server and get their customized client, certificates and configuration. The NSIS installer is used to generate the customized windows installers. Similar packages are generated for the various Unix based clients. Additional WWW/PHP based tools, have been developed to monitor and log usage of the service, using standard graphs, alarms for excessive use and a certificate revocation mechanism. The system has been integrated into the local identity management system (Kerberos/LDAP) in order to authorize and authenticate users upon initiation and per session usage. All code is Open Source and freely available. <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 freebsd vpn russel sutherland <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2808&type=ogg> OGG 1 byte 52 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2808&type=mp3> MP3 1 byte 52 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2808&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - George Neville-Neil - Four years of summer of code

The Google Summer of Code is a program designed to provide students with real world experience contributing to open source projects during the summer break in university studies. Each year Google selects a number of open source projects to act as mentoring organizations. Students are invited to submit project proposals for the open source projects that are most interesting to them. FreeBSD was one of the projects selected to participate in the inaugural Summer of Code in 2005 and we have participated each year since then. Over the past 4 years a total of 79 students have participated in the program and it has become a very significant source of new committers to FreeBSD. This talk will examine in detail the selection criteria for projects, the impact that successful projects have had, and some suggestions for how we can better leverage this program in the future. <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 google soc george neville-neil <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2807&type=ogg> OGG 1 byte 27 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2807&type=mp3> MP3 1 byte 27 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2807&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Antti Kantee - Converting kernel file systems to services

ABSD/UNIX operating system is traditionally split into two pieces: the kernel and userspace. Historically the reasons for this were clear: the UNIX kernel was a simple entity. However, over time the kernel has grown more and more complex. Currently, most of the same functionality is available both in userspace and the kernel, but under different names. Examples include synchronization routines and threading support. For instance, to lock a mutex in the NetBSD kernel, the call is `mutex_enter()`, while in userspace the routine which does exactly the same thing is known as `pthread_mutex_enter()`. Taking another classic example, a BSD style OS has `malloc()/free()` available both in userspace and the kernel, but with different linkage (the kernel `malloc` interface is currently being widely deprecated, though). This imposes a completely arbitrary division between the kernel and userspace. Most functionality provided by an operating system should be treated as a service instead of explicitly pinning it down as a userspace daemon or a kernel driver. Currently, due to the arbitrarily difference in programming interface names, functionality must be explicitly ported between the kernel and userspace if it is to run in one or the other environment. By unifying the environments where possible, the arbitrary division is weakened and porting between these environments becomes simpler.

Antti Kantee has been a NetBSD developer for many many moons. He has managed to work on quite a few bits and pieces of a BSD system: userland utilities, the `pkgsrc` packaging system, networking, virtual memory,

device drivers, hardware support and file systems.

See also <http://www.netbsd.org/docs/puffs/rump.htm> <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 antti kantee <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2806&type=ogg> OGG 1 byte 55 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2806&type=mp3> MP3 1 byte 55 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2806&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2008 - Matthieu Herrb - Input handling in wscons and X.Org

This talk will present the different layers that handle input, from the key that gets pressed or the mouse motion to the applications, all the way through the kernel drivers, X drivers and libraries, in the case of the OpenBSD/NetBSD wscons driver and the current and future X.Org server. It will cover stuff like keyboard mappings, touch-screen calibration, multi-pointer X or input coordinates transformations. It will show some problems of current implementations and try to show how current evolutions can solve them.

Matthieu Herrb is maintaing X on OpenBSD. I've been using X on various systems (SunOS, NetBSD, OpenBSD, Mac OS X,...) since 1989. He has been a member of the XFree86 Core Team for a short period in 2003 and is now a member of the X.Org Foundation BoD. Matthieu Herrb works at LAAS a research laborarory of the French National Research Agency (CNRS) both on robotics and network security. <http://2008.eurobsdcon.org/talks.html> eurobsdcon eurobsdcon2008 wscons x.org matthieu herrb <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2805&type=ogg> OGG 1 byte 57 minutes ogg <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2805&type=mp3> MP3 1 byte 57 minutes mp3 <http://audiovideocours.u-strasbg.fr/avc/courseaccess?id=2805&type=pdf> PDF 1 byte n pages pdf

EuroBSDCon 2007 Videos

http://misc.allbsd.de/Vortrag/EuroBSDCon_2007/ EuroBSDCon 2007 Papers eurobsdcon eurobsdcon2007 videos <http://video.eurobsdcon.dk/2007/> AnttiKanteeAndAlistairCrooks.EuroBSDCon.2007.avi Antti Kantee - ReFUSE: Userspace FUSE Reimplementation Using puffs 197 Mb avi refuse antti kantee BrooksDavis.EuroBSDCon.2007.avi Brooks Davis - Using FreeBSD to Promote Open Source Development Methods 92 Mb avi promotion freebsd brooks davis ClaudioJeker.EuroBSDCon.2007.avi Claudio Jeker - Routing on OpenBSD 394 Mb avi routing openbsd claudio jeker GeorgeNeville-Neil.EuroBSDCon.2007.avi George Neville-Neil - Network Protocol Testing in FreeBSD and in General 271 Kb avi network testing freebsd george neville-neil JohnHartmann.EuroBSDCon.2007.avi John P Hartmann - Real Men's Pipes - When UNIX meets the mainframe mindset 315 Mb avi pipes unix mainframes john p hartmann MarshallKirkMcKusick.EuroBSDCon.2007.avi Kirk Mckusick - A Brief History of the BSD Fast Filesystem 251 Mb avi fast file system kirk mckusick PawelJakubDawidek.EuroBSDCon.2007.avi Pawel Jakub - FreeBSD/ZFS - last word in operating/file systems 203 Mb avi zfs freebsd pawel jakub Pierre-YvesRitschard.EuroBSDCon.2007.avi Pierre Yves Ritschard - Load Balancing 219 Mb avi load balancing pierre yves ritschard RyanBickhart.EuroBSDCon.2007.avi Ryan Bickhart - Transparent TCP-to-SCTP Translation Shim Layer 376 Mb avi tcp-to-sctp freebsd ryan bickhart SorenStraarup.EuroBSDCon.2007.avi Soren Straarup - An ARM from shoulder to hand 141 Mb avi arm soren straarup Sam-eurobsdcon-large.mov Sam Leffler - Long Distance Wireless (for Emerging Regions) 248 Mb mov sam leffler SamSmith.EuroBSDCon.2007.avi Sam Smith - Fighting "Technical fires" 147 Mb avi sam smith SimonLNielsen.EuroBSDCon.2007.avi Simon L Nielsen - The FreeBSD Security Officer function 195 Kb avi freebsd security officer simon l nielsen StephenBorrill.EuroBSDCon.2007.avi Stephen Borrill - Building products with NetBSD - thin-clients 364 Mb avi netbsd thin clients stephen borrill StevenMurdoch.EuroBSDCon.2007.avi Steven Murdoch - Hot or Not: Fingerprinting hosts through clock skew 235 Mb avi finger printing clocks Steven Murdoch YvanVanhullebus.EuroBSDCon.2007.avi Yvan VanHullebus - NETASQ and BSD: a success story 382 Mb avi netasq yvan vanhullebus GregersPetersen.EuroBSDCon.2007.avi Gregers Petersen - Open Source - is it something new? 285 Mb avi open source gregers petersen

EuroBSDCon 2007 Papers

<http://2007.eurobsdcon.org/presentations/> EuroBSDCon 2007 Papers eurobsdcon eurobsdcon2007 papers http://misc.allbsd.de/Vortrag/EuroBSDCon_2007/ Antti_Kantee/refuse.pdf Antti Kantee - ReFUSE: Userspace

FUSE Reimplementation Using puffs 102 Kb pdf refuse antti kantee Brooks_Davis/davis-eurobsdcon2007.pdf Brooks Davis - Using FreeBSD to Promote Open Source Development Methods 989 Kb pdf promotion freebsd brooks davis Brooks_Davis/eurobsdcon2007-cluster-tutorial.pdf Brooks Davis - Building clusters with FreeBSD 2.2 Mb pdf clusters freebsd brooks davis Claudio_Jeker/routing_on_openbsd.tar Claudio Jeker - Routing on OpenBSD 1.3 Mb tar routing openbsd claudio jeker George_Neville-Neil/EuroBSD2007.pdf George Neville-Neil - Network Protocol Testing in FreeBSD and in General 251 Kb pdf network testing freebsd george neville-neil Isaac_Levy/ike-jail-with_SRC.tbz Isaac Levy - FreeBSD jail(8) Overview, the Secure Virtual Server 120 Mb jail freebsd isaac levy John_P_Hartmann/fbsd2007.odp John P Hartmann - Real Men's Pipes - When UNIX meets the mainframe mindset 382 Kb odp pipes unix mainframes john p hartmann John_P_Hartmann/pipjarg.pdf John P Hartmann - CMS Pipelines Explained 118 Kb pdf cms pipes john p hartmann Kirk_Mckusick/talk.pdf Kirk Mckusick - A Brief History of the BSD Fast Filesystem 145 Kb pdf fast file system kirk mckusick Marc_Balmer/radio_clocks.pdf Marc Balmer - Supporting Radio Clocks in OpenBSD 304 Kb pdf radio clocks openbsd marc balmer Marko_Zec/TUTORIAL.PDF Marko Zec - Network stack virtualization for FreeBSD 7.0 401 Kb pdf network stack virtualization freebsd marko zec Pawel_Jakub_Dawidek/eurobsdcon07_zfs.pdf Pawel Jakub - FreeBSD/ZFS - last word in operating/file systems 337 Kb pdf zfs freebsd pawel jakub Peter_Hansteen/pf-firewall.pdf Peter Hansteen - Firewalling with OpenBSD's PF packet filter 531 Kb pdf pf openbsd peter hansteen Pierre_Yves_Ritschard/loadbalancin.tgz Pierre Yves Ritschard - Load Balancing 23 Kb html load balancing pierre yves ritschard Robert_Watson/20070914-security-features.pdf Robert Watson - FreeBSD Advanced Security Features 152 Kb pdf security freebsd robert watson Ryan_Bickhart/Ryan_Bickhart.pdf Ryan Bickhart - Transparent TCP-to-SCTP Translation Shim Layer 491 Kb pdf tcp-to-sctp freebsd ryan bickhart Ryan_Bickhart/Ryan_Bickhart.ppt Ryan Bickhart - Transparent TCP-to-SCTP Translation Shim Layer 692 Kb ppt tcp-to-sctp freebsd ryan bickhart Soren_Straarup/arm_from_hand_to_shoulder_eurobsdcon_2007.pdf Soren Straarup - An ARM from shoulder to hand 307 Kb pdf arm soren straarup Sam_Leffler/EuroBSDCon2007.pdf Sam Leffler - Long Distance Wireless (for Emerging Regions) 19 Mb pdf sam leffler Sam_Smith/eurobsdcon-talk.pdf Sam Smith - Fighting "Technical fires" 1.4 Mb pdf sam smith Simon_L_Nielsen/freebsd-so-function-eurobsdcon-2007.pdf Simon L Nielsen - The FreeBSD Security Officer function 251 Kb pdf freebsd security officer simon l nielsen Stephen_Borrill/eurobsdcon.pdf Stephen Borrill - Building products with NetBSD - thin-clients 407 Kb pdf netbsd thin clients stephen borrill Steven_Murdoch/eurobsdcon07hotornot.pdf Steven Murdoch - Hot or Not: Fingerprinting hosts through clock skew 6.1 Mb pdf finger printing clocks Steven Murdoch Yvan_VanHullebus/2007-09-15-NETASQ-BSD-pub.pdf Yvan VanHullebus - NETASQ and BSD: a success story 2.4 Mb pdf netasq yvan vanhullebus

EuroBSDCon 2007 Photos

<http://www.flickr.com/photos/tags/eurobsdcon2007/> EuroBSDCon 2007 Photos by various people eurobsdcon eurobsdcon2007 photos flickr <http://www.flickr.com/photos/edkikkert/sets/72157602007517635/> Ed Kikkert - EuroBSD-Con 2007 taken place in Copenhagen, Denmark 14-15 September 2007 at the Symbion Science Park ed kikkert http://www.flickr.com/photos/tom_snow/sets/72157602050540536/ Tom (Snow) - Foto's taken bij Tom and Robert of www.snow.nl tom snow <http://www.flickr.com/photos/rickvanderzwet/sets/72157602002839498/> Rick van der Zwet rick van der zwet <http://www.flickr.com/photos/13801854@N02/sets/72157602081330565/> Petermhansteen petermhansteen <http://www.flickr.com/photos/12884927@N07/sets/72157601996279923/> Eystein.aarseth - Photos from EuroBSDCon in Copenhagen, Denmark, september 2007 eystein aarseth

Andre Opperman - The papers I write for EuroBSDCon 05

The papers I write for EuroBSDCon 05 on New Networking Feature in FreeBSD 6.0 and Optimizing FreeBSD IP and TCP in 7-CURRENT <http://people.freebsd.org/~andre/> eurobsdcon eurobsdcon2005 paper freebsd networking andre opperman <http://people.freebsd.org/~andre/New%20Networking%20Features%20in%20FreeBSD%206.pdf> 92 Kb New Networking Features in FreeBSD 6 pdf Optimizing%20the%20FreeBSD%20IP%20and%20TCP%20Stack.pdf 1 Mb Optimizing the FreeBSD IP and TCP Stack pdf

The presentation I gave at SUCON 04

The presentation I gave at SUCON 04 on 2nd September 2004 about enhancements/changes in FreeBSD 5.3 Networking Stack. sucon presentation freebsd networking andre opperman <http://people.freebsd.org/~andre/FreeBSD-5.3-Networking.pdf> 115 Kb FreeBSD-5.3-Networking.pdf pdf

AsiaBSDCon 2009 Paper List

Papers of the AsiaBSDCon 2009 <http://2009.asiabsdcon.org/papers/> asiabsdcon asiabsdcon2009 <http://2009.asiabsdcon.org/papers/abc2009-P1A-paper.pdf> 351 Kb 9 pages PC-BSD - Making FreeBSD on the Desktop a reality by Kris Moore pdf paper freebsd pcbsd kris moore [abc2009-P1B-paper.pdf](http://2009.asiabsdcon.org/papers/abc2009-P1B-paper.pdf) 58 Kb 3 pages Crypto Acceleration on FreeBSD by Philip Paeps pdf paper crypto acceleration freebsd philip paeps [abc2009-P2A-paper.pdf](http://2009.asiabsdcon.org/papers/abc2009-P2A-paper.pdf) 401 Kb 6 pages OpenBGPD - Bringing full views to OpenBSD since 2004 by Claudio Jeker pdf paper openbgpd openbsd claudio jeker [abc2009-P2B-paper.pdf](http://2009.asiabsdcon.org/papers/abc2009-P2B-paper.pdf) 359 Kb 12 pages FreeBSD on high performance multi-core embedded PowerPC systems - Rafal Jaworowski pdf paper freebsd high performance rafal jaworowski [abc2009-P3A-paper.pdf](http://2009.asiabsdcon.org/papers/abc2009-P3A-paper.pdf) 662 Kb 7 pages Isolating Cluster Users (and Their Jobs) for Performance and Predictability by Brooks Davis pdf paper clusters brooks davis [abc2009-P3B-paper.pdf](http://2009.asiabsdcon.org/papers/abc2009-P3B-paper.pdf) 245 Kb 14 pages OpenBSD Hardware Sensors Framework by Constantine A. Murenin pdf paper openbsd hardware sensors constantine murenin [abc2009-P4A-paper.pdf](http://2009.asiabsdcon.org/papers/abc2009-P4A-paper.pdf) 753 Kb 4 pages FreeBSD and SOI-Asia Project Mohamad by Dikshie Fauzie pdf paper freebsd dikshie fauzie [abc2009-P4B-paper.pdf](http://2009.asiabsdcon.org/papers/abc2009-P4B-paper.pdf) 67 Kb 8 pages An Overview of FreeBSD/mips by M. Warner Losh pdf paper freebsd mips warner losh [abc2009-P5A-paper.pdf](http://2009.asiabsdcon.org/papers/abc2009-P5A-paper.pdf) 213 Kb 10 pages Environmental Independence: BSD Kernel TCP/IP in Userspace by Antti Kantee pdf paper tcpip antti kantee [abc2009-P5B-paper.pdf](http://2009.asiabsdcon.org/papers/abc2009-P5B-paper.pdf) 154 Kb 20 pages Active-Active Firewall Cluster Support in OpenBSD by David Gwynne pdf paper firewall cluster openbsd david gwynne [abc2009-P6A-paper.pdf](http://2009.asiabsdcon.org/papers/abc2009-P6A-paper.pdf) 55 Kb 7 pages The Locking Infrastructure in the FreeBSD kernel by Attilio Rao pdf paper locking freebsd attilio rao [abc2009-P6B-paper.pdf](http://2009.asiabsdcon.org/papers/abc2009-P6B-paper.pdf) 114 Kb 8 pages Deprecating groff for BSD manual display by Kristaps Dzonsons pdf paper groff kristaps dzonsons [abc2009-P7B-paper.pdf](http://2009.asiabsdcon.org/papers/abc2009-P7B-paper.pdf) 150 Kb 3 pages Mail system for distributed network by Andrey Zakharchenko pdf paper mail andrey zakharchenko

AsiaBSDCon 2008 Paper List

Papers of the AsiaBSDCon 2007 <http://2008.asiabsdcon.org/papers/> asiabsdcon asiabsdcon2008 <http://2008.asiabsdcon.org/papers/abc2008-proc-cover.pdf> 467 Kb Cover page pdf paper cover [abc2008-proc-all.pdf](http://2008.asiabsdcon.org/papers/abc2008-proc-all.pdf) 9.3 Mb Whole of the proceedings pdf paper P1A-paper.pdf 6.4 Mb PC-BSD: FreeBSD on the Desktop, Matt Olander (iXsystems) pdf paper pc-bsd freebsd desktop matt olander P1B-paper.pdf 94 Kb Tracking FreeBSD in a Commercial Setting, M. Warner Losh (Cisco Systems, Inc.) pdf paper freebsd warner losh P3A-paper.pdf 92 Kb Gaols: Implementing Jails Under the kauth Framework, Christoph Badura (The NetBSD Foundation) pdf paper netbsd jails kauth christoph badura P3B-paper.pdf 526 Kb BSD implementations of XCAST6, Yuji IMAI, Takahiro KUROSAWA, Koichi SUZUKI, Eiichi MURAMOTO, Katsuomi HAMAJIMA, Hajimu UMEMOTO, and Nobuo KAWAGUTI (XCAST fan club, Japan) pdf paper bsd xcast6 yuji imai takahiro kurosawa koichi suzuki eiichi muramoto katsuomi hamajima hajimu umemoto nobuo kawaguti P4A-paper.pdf 483 Kb Using FreeBSD to Promote Open Source Development Methods, Brooks Davis, Michael AuYeung, Mark Thomas (The Aerospace Corporation) pdf paper freebsd promotion brooks david michael auyeung mark thomas P4B-paper.pdf 126 Kb Send and Receive of File System Protocols: Userspace Approach With puffs, Antti Kantee (Helsinki University of Technology, Finland) pdf paper puffs antti kantee P5A-paper.pdf 97 Kb Logical Resource Isolation in the NetBSD Kernel, Kristaps Dzonsons (Centre for Parallel Computing, Swedish Royal Institute of Technology) pdf paper netbsd resources kristaps dzonsons P5B-paper.pdf 91 Kb GEOM — in Infrastructure We Trust, Pawel Jakub Dawidek (The FreeBSD Project) pdf paper freebsd geom pawel jakub dawidek P6A-paper.pdf 341 Kb A Portable iSCSI Initiator, Alistair Crooks (The NetBSD Foundation) pdf paper netbsd iscsi alistair crooks P8A-paper.pdf 410 Kb OpenBSD Network Stack Internals, Claudio Jeker (The OpenBSD Project) pdf paper openbsd network stack claudio jeker P8B-paper.pdf 72 Kb Reducing Lock Contention in a Multi-Core System, Randall Stewart (Cisco Systems, Inc.) pdf paper freebsd lock contention smp randall stewart P9A-paper.pdf 87 Kb Sleeping Beauty — NetBSD on Modern Laptops, Jorg Sonnenberger, Jared D. McNeill (The NetBSD Foundation) pdf paper netbsd laptops jorg sonnenberger jared d mcneill

AsiaBSDCon 2007 Paper/Slides List

Slides and papers of the AsiaBSDCon 2007 <http://2007.asiabsdcon.org/papers/> asiabsdcon asiabsdcon2007 <http://2007.asiabsdcon.org/papers/abc2007-proc-cover.pdf> 588 Kb Cover page pdf paper cover [abc2007-proc-all.pdf](http://2007.asiabsdcon.org/papers/abc2007-proc-all.pdf) 6.5 Mb Whole of the Proceedings pdf paper P01-paper.pdf 412 Kb A NetBSD-based IPv6 NEMO Mobile Router, Jean Lorchat, Koshiro Mitsuya, Romain Kuntz (Keio University, Japan) [paper] pdf paper netbsd ipv6 nemo jean lorchat koshiro mitsuya romain kuntz P02-paper.pdf 1371 Kb Reflections on Building a High Performance Computing Cluster Using FreeBSD, Brooks Davis (The Aerospace Corporation/brooks at FreeBSD.org, USA) [paper] pdf paper freebsd high performance computing brooks davis P03-paper.pdf 86 Kb Support for Radio Clocks in OpenBSD, Marc Balmer (mbalmer at openbsd.org, Switzerland) [paper] pdf paper openbsd radio clocks marc balmer P04-paper.pdf 68 Kb puffs - Pass to Userspace Framework File System, Antti Kantee (Helsinki University of Technology, Finland) [paper] pdf pa-

per puffs antii kantee P04-slides.pdf 116 Kb puffs - Pass to Userspace Framework File System, Antti Kantee (Helsinki University of Technology, Finland) [slides] pdf slides puffs antii kantee P05-paper.pdf 140 Kb An ISP Perspective, jail(8) Virtual Private Servers, Isaac Levy (NYC*BUG/LESMUUG, USA) [paper] pdf paper freebsd jail isp isaac levy P05-slides.pdf 20 Mb An ISP Perspective, jail(8) Virtual Private Servers, Isaac Levy (NYC*BUG/LESMUUG, USA) [slides] pdf slides freebsd jail isp isaac levy P06-paper.pdf 32 Kb Nsswitch Development: Nss-modules and libc Separation and Caching, Michael A Bushkov (Southern Federal University/bushman at FreeBSD.org, Russia) [paper] pdf paper nsswitch michael bushkov P08-paper.pdf 328 Kb How the FreeBSD Project Works, Robert N M Watson (University of Cambridge/rwatson at FreeBSD.org, United Kingdom) [paper] pdf paper freebsd freebsd project robert watson P10-paper.pdf 311 Kb SHISA: The Mobile IPv6/NEMO BS Stack Implementation Current Status, Keiichi Shima (Internet Initiative Japan Inc., Japan), Koshiro Mitsuya, Ryuji Wakikawa (Keio University, Japan), Tsuyoshi Momose (NEC Corporation, Japan), Keisuke Uehara (Keio University, Japan) [paper] pdf paper ipv6 nemo keiichi shima koshiro mitsuya ryuji wakikawa tsuyoshi momose keisuke uehara P11-slides.pdf 601 Kb Bluffs: BSD Logging Updated Fast File System, Stephan Uphoff (Yahoo!, Inc./ups at FreeBSD.org, USA) [slides] pdf slides bluffs stephan uphoff P12-paper.pdf 1071 Kb Implementation and Evaluation of the Dual Stack Mobile IPv6, Koshiro Mitsuya, Ryuji Wakikawa, Jun Murai (Keio University, Japan) [paper] pdf paper ipv6 koshiro mitsuya ryuji wakikawa jun murai P15-paper.pdf 97 Kb Security Measures in OpenSSH, Damien Miller (djm at openbsd.org, Australia) [paper] pdf paper openssh damien miller P16-paper.pdf 96 Kb Porting the ZFS File System to the FreeBSD Operating System, Pawel Jakub Dawidek (pjd at FreeBSD.org, Poland) [paper] pdf paper freebsd zfs pawel jakub dawidek P16-slides.pdf 278 Kb Porting the ZFS File System to the FreeBSD Operating System, Pawel Jakub Dawidek (pjd at FreeBSD.org, Poland) [slides] pdf slides freebsd zfs pawel jakub dawidek

Robert Watson's Slides from EuroBSDCon 2004

Robert Watson will describe the design and application of the TrustedBSD MAC Framework, a flexible kernel security framework developed on FreeBSD, and recently experimentally ported to Apple's Darwin operating system. The MAC Framework permits loadable access control kernel modules to be loaded, modifying the security behavior of the operating system, including SEBSD, a port of the SELinux FLASK/TE security model to FreeBSD. <http://www.watson.org/~robert/freebsd/2004eurobsdcon/> eurobsdcon eurobsdcon2004 slides trustedbsd freebsd mac robert watson <http://www.watson.org/~robert/freebsd/2004eurobsdcon/20041031-eurobsdcon-macframework.pdf> 270 Kb TrustedBSD MAC Framework on FreeBSD and Darwin pdf

Robert Watson's Slides from UKUUG LISA 2006

UKUUG LISA 2006 took place in Durham, UK in March, 2006. On this page, you can find my slides from this conference.

OpenBSM is a BSD-licensed implementation of Sun's Basic Security Module (BSM) API and file format, and is the foundation of the TrustedBSD audit implementation for FreeBSD. This talk will cover the requirements, design, and implementation of audit support for FreeBSD. Security audit support provides detailed logging of security-relevant events, and meets the requirements of the CAPP Common Criteria protection profile. <http://www.watson.org/~robert/freebsd/2006ukuuglisa/> ukuug slides openbsm trustedbsd freebsd robert watson <http://www.watson.org/~robert/freebsd/2006ukuuglisa/20060323-ukuug2006lisa-audit.pdf> 199 Kb CAPP-Compliant Security Event Audit System for Mac OS X and FreeBSD (UKUUG LISA 2006). pdf

Robert Watson's Slides from EuroBSDCon 2006 and FreeBSD Developer Summit

EuroBSDCon 2006 took place in Milan, Italy, and not only offered excellent food on a flexible schedule, but also an interesting array of talks on work spanning the BSD's. On this page, you can find my slides from the FreeBSD developer summit and full conference.

Status report on the TrustedBSD Project: introduction and status regarding Audit, plus a TODO list; introduction to the priv(9) work recently merged to 7.x.

The FreeBSD Project is one of the oldest and most successful open source operating system projects, seeing wide deployment across the IT industry. From the root name servers, to top tier ISPs, to core router operating

systems, to firewalls, to embedded appliances, you can't use a networked computer for ten minutes without using FreeBSD dozens of times. Part of FreeBSD's reputation for quality and reliability comes from the nature of its development organization—driven by a hundreds of highly skilled volunteers, from high school students to university professors. And unlike most open source projects, the FreeBSD Project has developers who have been working on the same source base for over twenty years. But how does this organization work? Who pays the bandwidth bills, runs the web servers, writes the documentation, writes the code, and calls the shots? And how can developers in a dozen time zones reach agreement on the time of day, let alone a kernel architecture? This presentation will attempt to provide, in 45 minutes, a brief if entertaining snapshot into what makes FreeBSD run. <http://www.watson.org/~robert/freebsd/2006eurobsdcon/> eurobsdcon eurobsdcon2006 robert watson <http://www.watson.org/~robert/freebsd/2006eurobsdcon/20061110-devsummit-trustedbsd.pdf> 166 Kb TrustedBSD presentation on Audit and priv(9) (Developer Summit) pdf slides trustedbsd freebsd 20061111-eurobsdcon2006-how-freebsd-works.pdf 4.4 Mb How the FreeBSD Project Works (EuroBSDCon 2006 Full Conference) pdf slides freebsd freebsd project

Robert Watson's Slides from BSDCan 2006 and FreeBSD Developer Summit

As usual, Dan Langille ran an excellent [BSDCan conference](#). On this page, you can find my slides from the developer summit and full conference, excluding the contents of the WIPs, for which I don't have permission to redistribute the slides. <http://www.watson.org/~robert/freebsd/2006bsdcan/> bsdcan bsdcan2006 notes devsummit robert watson <http://www.watson.org/~robert/freebsd/2006bsdcan/20060511-devsummit-network-cabal-summary.pdf> 72 Kb Notes from the 10 May 2006 Meeting of the Network Stack Cabal (Developer Summit) pdf freebsd 20060511-devsummit-smpng-network-summary.pdf 91 Kb SMPng Network Stack Update (Developer Summit) pdf smp 20060511-devsummit-trustedbsd-mac-framework-retrofit.pdf 120 Kb TrustedBSD Project Update (Developer Summit) pdf trust-edbsd 20060512-bsdcan2006-how-freebsd-works.pdf 4.4 Mb Kb How the FreeBSD Project Works (BSDCan 2006 Full Conference) pdf freebsd freebsd project

Robert Watson's Slides from EuroBSDCon 2005

EuroBSDCon 2005 took place in Basel, Switzerland in November, 2005. Due to an injury, I was unable to attend the conference itself, and my talks were presented in absentia by Poul-Henning Kamp and Ed Maste, who have my greatest appreciation!

The FreeBSD SMPng Project has spent the past five years redesigning and reimplementing SMP support for the FreeBSD operating system, moving from a Giant-locked kernel to a fine-grained locking implementation with greater kernel threading and parallelism. This paper introduces the FreeBSD SMPng Project, its architectural goals and implementation approach. It then explores the impact of SMPng on the FreeBSD network stack, including strategies for integrating SMP support into the network stack, locking approaches, optimizations, and challenges. <http://www.watson.org/~robert/freebsd/2005eurobsdcon/> eurobsdcon eurobsdcon2005 slides freebsd smp robert watson poul-henning kamp ed maste <http://www.watson.org/~robert/freebsd/2005eurobsdcon/eurobsdcon2005-netperf.pdf> 370 Kb Introduction to Multithreading and Multiprocessing in the FreeBSD SMPng Network Stack pdf

Robert Watson's Slides from BSDCan 2004

BSDCan 2004 took place at the University of Ottawa in Ottawa, Canada. On this page, you can find my slides from the conference.

Robert Watson will describe a variety of pieces of work done as part of the TrustedBSD Project, including the TrustedBSD MAC Framework, Audit facilities for FreeBSD, as well as supporting infrastructure work such as GEOM/GBDE, UFS2, OpenPAM. He will also discuss how certification and evaluation play into feature selection, design, and documentation. <http://www.watson.org/~robert/freebsd/2004bsdcan/> bsdcan bsdcan2004 slides trustedbsd freebsd robert watson <http://www.watson.org/~robert/freebsd/2004bsdcan/20040515-2004bsdcan-trustedbsd.pdf> 277 Kb TrustedBSD: Trusted Operating System Features for BSD pdf

Robert Watson's Slides from AsiaBSDCon 2004

AsiaBSDCon 2004 took place in Taipei, Taiwan, in March 2004, and was hosted by Academia Sinica. <http://www.watson.org/~robert/freebsd/2004asiabsdcon/> asiabsdcon asiabsdcon2004 robert watson <http://www.watson.org/~robert/freebsd/2004asiabsdcon/200403-asiabsdcon2004-trustedbsd.pdf> 135 Kb Extensible Kernel Security through the TrustedBSD MAC Framework. pdf slides trustedbsd mac 20040313-asiabsdcon04-bsdbof.pdf 1.4 Mb AsiaBSDCon 2004 BSD (FreeBSD) BoF session pdf slides freebsd

A Tale of Four Kernels

The FreeBSD, GNU/Linux, Solaris, and Windows operating systems have kernels that provide comparable facilities. Interestingly, their code bases share almost no common parts, while their development processes vary dramatically. We analyze the source code of the four systems by collecting metrics in the areas of file organization, code structure, code style, the use of the C preprocessor, and data organization. The aggregate results indicate that across various areas and many different metrics, four systems developed using wildly different processes score comparably. This allows us to posit that the structure and internal quality attributes of a working, non-trivial software artifact will represent first and foremost the engineering requirements of its construction, with the influence of process being marginal, if any. <http://www.spinellis.gr/pubs/> freebsd linux solaris windows article kernel diomidis spinellis <http://www.spinellis.gr/pubs/conf/2008-ICSE-4kernel/html/Spi08b.html> Diomidis Spinellis. A tale of four kernels. In Wilhem Schfer, Matthew B. Dwyer, and Volker Gruhn, editors, ICSE '08: Proceedings of the 30th International Conference on Software Engineering, pages 381-390, New York, May 2008. Association for Computing Machinery. [html conf/2008-ICSE-4kernel/html/Spi08b.pdf](http://www.spinellis.gr/pubs/conf/2008-ICSE-4kernel/html/Spi08b.pdf) Diomidis Spinellis. A tale of four kernels. In Wilhem Schfer, Matthew B. Dwyer, and Volker Gruhn, editors, ICSE '08: Proceedings of the 30th International Conference on Software Engineering, pages 381-390, New York, May 2008. Association for Computing Machinery. pdf

Global software development in the FreeBSD project

FreeBSD is a sophisticated operating system developed and maintained as open-source software by a team of more than 350 individuals located throughout the world. This study uses developer location data, the configuration management repository, and records from the issue database to examine the extent of global development and its effect on productivity, quality, and developer cooperation. The key findings are that global development allows round-the-clock work, but there are some marked differences between the type of work performed at different regions. The effects of multiple dispersed developers on the quality of code and productivity are negligible. Mentoring appears to be sometimes associated with developers living closer together, but ad-hoc cooperation seems to work fine across continents. <http://www.spinellis.gr/pubs/> freebsd article global software development diomidis spinellis <http://www.spinellis.gr/pubs/conf/2006-GSD-FreeBSD/html/GSD-FreeBSD.html> International Workshop on Global Software Development for the Practitioner, pages 73-79. ACM Press, May 2006 [html conf/2006-GSD-FreeBSD/html/GSD-FreeBSD-presentation.pdf](http://www.spinellis.gr/pubs/conf/2006-GSD-FreeBSD/html/GSD-FreeBSD-presentation.pdf) In NASSCOM Quality Summit 2006: Setting benchmarks in global outsourcing, Bangalore, India, September 2006. National Association of Software and Services Companies (NASSCOM). [html trade/2006-LinuxFormat-GSD/html/GSDEV.htm](http://www.spinellis.gr/pubs/conf/2006-GSD-FreeBSD/html/GSD-FreeBSD-presentation.pdf) Linux Format, (11):60-63, September/October 2006. In Greek. [html](http://www.spinellis.gr/pubs/conf/2006-GSD-FreeBSD/html/GSD-FreeBSD.html)

BSDCan-2006 Photos - Friday

http://www.db.net/gallery/BSDCan/BSDCan_2006_Friday/ Photos taken during the Conference on Friday at BSDCan 2006 in Ottawa by Diane Bruce. 2006 bsdcn bsdcn2006 photos diane bruce

BSDCan-2006 Photos - Saturday

http://www.db.net/gallery/BSDCan/BSDCan_2006_Saturday/ Photos taken during the Conference on Saturday at BSDCan 2006 in Ottawa by Diane Bruce. 2006 bsdcn bsdcn2006 photos diane bruce

What's your biggest Time Management problem?

What's your biggest Time Management problem?

Tom Limoncelli is a FreeBSD user and the author of the O'Reilly book, "Time Management for System Administrators". He'll be giving a brief presentation with highlights from his book then will take questions from the audience. Whether you are a system administrator, a developer (or even a Linux user) this presentation will help you with something more precious a quad-processor AMD box.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10172> nycbug presentation time management tom limoncelli
<http://www.fetissof.org/public/nycbug/nycbug-03-04-09.mp3> MP3 version mp3 11 Mb

Postfix Performance Tuning

Money can buy you bandwidth, but latency is forever!

John Mashey, MIPS

Victor will cover an array of issues connected to Postfix performance tuning, including:

- Latency, concurrency and throughput
- Postfix input processing
- Queue file format rationale
- Input processing bottlenecks
- Pre-queue filters, milters, content filters
- Tuning for fast (enough) input
- Postfix on-disk queues, requirements and architecture
- What is a “transport”?
- Postfix “nqmgr” scheduler algorithm
- Per-destination in memory queues
- Per-destination scheduler controls
- SMTP delivery
- Understanding delay logging
- Transport process limits, concurrency limits
- Scaling to thousands of output processes
- Connection caching, TLS session caching, feedback controls

Speaker Bio

Victor Duvovni trained in mathematics, switched tracks to CS in 1980s leaving Princeton with a master’s degree in mathematics and newly acquired skills in Unix system administration and system programming. In 1990 moved to Lehman Brothers, worked on system management tooling, and network engineering. Ported “Moirá” from MIT to Lehman, built efficient build systems that predated (and partly inspired) Jumpstart. In 1994 joined ESM to market “CMDB” tools to enterprise users, but this did not pan out, in the mean time learned Tcl, and contributed bunch of patches to the 7.x early 8.x TCL releases. In 1997 returned to New York, working in IT Security at Morgan Stanley since late 1999. At Morgan Stanley, developed a hobby in perimeter email security, becoming an active Postfix user and very soon contributor in May of 2001. In addition to many smaller feature improvements, contributed initial implementation of SMTP connection caching, overhauled and currently maintain LDAP and TLS support. Made significant design contributions to queue manager in collaboration with Wietse and Patrik Raq. In 2.6 contributing support for TLS EC ciphers and multi-instance management tooling, ideally also TLS SNI if time permits.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10168> nycbug presentation postfix john mashey
<http://www.fetissof.org/public/nycbug/nycbug-02-04-09.mp3> MP3 version mp3 11 Mb

Introduction to Puppet

What it is and how can it make system administration less painful

About the speaker:

Larry Ludwig - Principal Consultant/Founder of Empowering Media. Empowering Media is a consulting firm and managed hosting provider. Larry Ludwig has been in the industry for over 15 years as a system administration and system programmer. He's had previous experience working for many Fortune 500 corporations and holds a BS in CS from Clemson University. Larry, along with Eric E. Moore and Brian Gupta are founding members of the NYC Puppet usergroup.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10171> nycbug presentation puppet larry ludwig
<http://www.fetissof.org/public/nycbug/nycbug-01-07-09.mp3> MP3 version mp3 11 Mb

Hardware Performance Monitoring Counters

Many modern CPUs provide on chip counters for performance events such as retiring instructions and cache misses. The hwpmc driver and libraries in FreeBSD give systems administrators and programmers access to APIs which make it possible to measure performance without modifying source code and with minimal intrusion into application execution. This talk will be a brief introduction to HWPMC, and how to use it.

Bio: George Neville-Neil is the co-author with Kirk McKusick of The Design and Implementation of the FreeBSD Operating System. He works on networking an operating systems for fun and profit.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10166> nycbug presentation george neville-neil counters
<http://www.fetissof.org/public/nycbug/nycbug-11-05-08.mp3> MP3 version mp3 4 Mb

New York City BSD Con 2008: BSD v. GPL - a.k.a. not the sequel to “BSD is Dying”

BSD vs GPL is a sweeping epic, focused on the dichotomy between good and evil. It peers inside the hearts and minds of the creators of these movements and dissects their battle for world domination. No common documentary will dare to follow the path that BSD vs GPL blazes.

<http://talks.dixongroup.net/nycbsdcon2008/> nycbsdcon nycbsdcon2008 presentation humor bsd versus gpl jason dixon
<http://talks.dixongroup.net/nycbsdcon2008/BSDvGPL.mp4> 15 Mb MP4 mp4

New York City BSD Con 2008

Slides of presentations given at New York City BSD Conference 2008. <http://www.nycbsdcon.org> nycbsdcon2008 nycbsdcon presentation <http://www.squid-cache.org/~adrian/talks/20081007%20-%20NYCBSDCON%20-%20Disk%20IO.pdf> 197 Kb 92 pages Adrian Chadd: High-throughput concurrent disk IO in FreeBSD. pdf freebsd high performance adrian chadd http://www.nycbsdcon.org/2008/files/dillon_hammer.tgz 820 Kb 16 pages Matthew Dillon: The HAMMER File System. html hammer metthew dillon http://www.nycbsdcon.org/2008/files/magnusson_pcc.pdf 123 Kb 29 pages Anders Magnusson: Design and Implementation of the Portable C Compiler. pdf pcc anders magnusson <http://www.openbsd.org/papers/nycbsdcon08-pie/> 21 pages Kurt Miller: OpenBSD's Position Independent Executables (PIE) Implementation. html openbsd pie kurt miller <http://www.silby.com/nycbsdcon08/NYCBSDCon-tcpdiff.pdf> 88 Kb 28 pages Mike Silbersack: Detecting TCP regressions with tcpdiff. pdf tcp regression tcpdiff mike silbersack http://www.nycbsdcon.org/2008/files/wright_hardware-wrong.pdf 1.7 Mb 22 pages Jason L Wright: When Hardware Is Wrong, or “They can Fix It In Software”. pdf hardware jason l wright http://www.nycbsdcon.org/2008/files/vidal_atf.pdf 570 Kb 18 pages Julio M. Merino Vidal: An introduction to the Automated Testing Framework (ATF) for NetBSD. pdf netbsd atf julio m merino vidal

New York City BSD Con 2008

Audio recordings of presentations given at New York City BSD Conference 2008. Courtesy of nikolai at fetissof.org. The main page also has links to the slides. <http://www.fetissof.org/public/nycbsdcon08/> nycbsdcon2008 nycbsdcon presentation <http://www.fetissof.org/public/nycbsdcon08/1.1.mp3> 14 Mb Adrian Chadd: High-throughput concurrent disk IO in FreeBSD. mp3 freebsd high performance adrian chadd 1.2.mp3 9 Mb Jason L Wright: When Hardware Is Wrong, or “They can Fix It In Software”. mp3 hardware jason l wright 1.3.mp3 14 Mb Matthew Dillon: The HAMMER File System. mp3 hammer metthew dillon 1.4.mp3 15 Mb Anders Magnusson: Design and Implementation of the Portable C Compiler. mp3 pcc anders magnusson 1.5.mp3 11 Mb Michael Shalayeff: Porting PCC. mp3 pcc

michael shalayeff 1.6.mp3 10 Mb Julio M. Merino Vidal: An introduction to the Automated Testing Framework (ATF) for NetBSD. mp3 netbsd atf julio m merino vidal 1.7.mp3 15 Mb Jeremy C. Reed: Introduction to DNSSEC. mp3 dnssec jeremy c reed 1.8.mp3 4 Mb Jason Dixon: BSD versus GPL. mp3 bsd versus gpl jason dixon 2.2.mp3 16 Mb Pawel Jakub Dawidek: A closer look at the ZFS file system. mp3 freebsd zfs pawel jakub dawidek 2.3.mp3 10 Mb Kurt Miller: OpenBSD's Position Independent Executables (PIE) Implementation. mp3 openbsd pie kurt miller 2.4.mp3 11 Mb Mike Silbersack: Detecting TCP regressions with tcpdiff. mp3 tcp regression tcpdiff mike silbersack 2.5.mp3 10 Mb Michael Lucas: Network Refactoring, or doing an oil change at 80 MPH. mp3 network refactoring michael lucas

Public Key sudo

Two tools which have become the norm in Linux- and Unix-based environments are SSH for secure communications, and sudo for performing administrative tasks. These are independent programs with substantially different purposes, but they are often used in conjunction. In this talk, I describe a flaw in their interaction, and then present our solution called public-key sudo.

Public-key sudo is an extension to the sudo authentication mechanism which allows for public key authentication using the SSH public key framework. I describe our implementation of a generic SSH authentication module and the sudo modifications required to use this module.

Bio:

Matthew Burnside is a Ph.D. student in the Computer Science department at Columbia University, in New York. He works for Professor Angelos Keromytis in the Network Security Lab. He received his B.A and M.Eng from MIT in 2000, and 2002, respectively. His research interests are in network anonymity, trust management, and enterprise-scale policy enforcement.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10160> nycbug presentation sudo public key matthew burnside
<http://www.fetissof.org/public/nycbug/nycbug-08-06-08.mp3> MP3 version mp3 2 Mb

Configuration Management with Cfengine

Configuration Management with Cfengine

Cfengine is a policy-based configuration management system. Its primary function is to provide automated configuration and maintenance of computers, from a policy specification.

The cfengine project was started in 1993 as a reaction to the complexity and non-portability of shell scripting for Unix configuration management, and continues today. The aim was to absorb frequently used coding paradigms into a declarative, domain-specific language that would offer self-documenting configuration.

about the speaker:

Steven Kreuzer has been working with Open Source technologies since as long as he can remember, starting out with a 486 salvaged from a dumpster behind his neighborhood computer store. In his spare time he enjoys doing things with technology that have absolutely no redeeming social value.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10157> nycbug presentation configuration management cfengine
<http://www.fetissof.org/public/nycbug/nycbug-07-02-08.mp3> MP3 version mp3 58 minutes 6 Mb

Managing OpenBSD Environments

This talk is the result of an after-meeting discussion with a few folks, when it became apparent that there is some confusion as to how to deal with OpenBSD in small and large environments. The topic of installation and upgrading came up again. This talk is aimed to hopefully dispel many of the rumors, provide a thorough description and walk

through of the various stages of running OpenBSD in any size environment, and some of the features and tools at the administrator's disposal.

Okan Demirmen has been working with UNIX-like systems for as long as he can remember and has found OpenBSD to match some of the same philosophies in which he believes, namely simplicity and correctness, and reap the benefits of such.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10154> nycbug presentation openbsd system management
<http://www.fetissof.org/public/nycbug/nycbug-05-07-08.mp3> MP3 version mp3 103 minutes 11 Mb

Building a High-Performance Computing Cluster Using FreeBSD

Special NYC*BUG meeting with FreeBSD developer Brooks Davis

Since late 2000 we have developed and maintained a general purpose technical and scientific computing cluster running the FreeBSD operating system. In that time we have grown from a cluster of 8 dual Intel Pentium III systems to our current mix of 64 dual, quad-core Intel Xeon and 289 dual AMD Opteron systems.

In this talk we reflect on the system architecture as documented in our BSDCon 2003 paper “Building a High-performance Computing Cluster Using FreeBSD” and our changes since that time. After a brief overview of the current cluster we revisit the architectural decisions in that paper and reflect on their long term success. We then discuss lessons learned in the process. Finally, we conclude with thoughts on future cluster expansion and designs.

Bio

Brooks Davis is an Engineering Specialist in the High Performance Computing Section of the Computer Systems Research Department at The Aerospace Corporation. He has been a FreeBSD user since 1994, a FreeBSD committer since 2001, and a core team member since 2006. He earned a Bachelors Degree in Computer Science from Harvey Mudd College in 1998.

His computing interests include high performance computing, networking, security, mobility, and, of course, finding ways to use FreeBSD in all these areas. When not computing, he enjoys reading, cooking, brewing and pounding on red-hot iron in his garage blacksmith shop.

<http://www.nycbug.org> nycbug presentation high performance computing freebsd brooks davis
<http://www.fetissof.org/public/nycbug/nycbug-03-20-08.mp3> MP3 version mp3 80 minutes 9 Mb

User Interfaces and How People Think

“User Interfaces and How People Think” will introduce concepts of designing software for different users by observing how they think about and do what they do. While much of design today focuses on the front-end of computer systems, there is opportunity to innovate in every area where a human interacts with software.

Bio: Jeffery Mau is a user experience designer with the leading business and technology consulting firm Sapient. He has helped clients create great customer experiences in the financial services, education, entertainment and telecommunications industries. With a passion for connecting people with technology, Jeff specializes in Information Architecture and Business Strategy. Jeff holds a Masters in Design from the IIT Institute of Design in Chicago, Illinois.

<http://www.nycbug.org> nycbug presentation user interfaces <http://www.fetissof.org/public/nycbug/nycbug-03-05-08.mp3> MP3 version mp3 78 minutes 9 Mb

Open Meeting on OpenSSH

Open Meeting on OpenSSH

February's NYCBUG meeting is a broad look at OpenSSH, the de facto method for remote administration and more. OpenSSH celebrated its 8th anniversary this past September, and we thought this would be a great opportunity to discuss OpenSSH, and for others to contribute their hacks and interesting applications.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10150> nycbug presentation openssh
<http://www.fetissof.org/public/nycbug/nycbug-02-06-08.mp3> MP3 version mp3 63 minutes 7 Mb

SSARES

SSARES: Secure Searchable Automated Remote Email Storage - A usable, secure email system on a remote untrusted server

The increasing centralization of networked services places user data at considerable risk. For example, many users store email on remote servers rather than on their local disk. Doing so allows users to gain the benefit of regular backups and remote access, but it also places a great deal of unwarranted trust in the server. Since most email is stored in plaintext, a compromise of the server implies the loss of confidentiality and integrity of the email stored therein. Although users could employ an end-to-end encryption scheme (e.g., PGP), such measures are not widely adopted, require action on the sender side, only provide partial protection (the email headers remain in the clear), and prevent the users from performing some common operations, such as server-side search.

To address this problem, we present Secure Searchable Automated Remote Email Storage (SSARES), a novel system that offers a practical approach to both securing remotely stored email and allowing privacy-preserving search of that email collection. Our solution encrypts email (the headers, body, and attachments) as it arrives on the server using public-key encryption. SSARES uses a combination of Identity Based Encryption and Bloom Filters to create a searchable index. This index reveals little information about search keywords and queries, even against adversaries that compromise the server. SSARES remains largely transparent to both the sender and recipient. However, the system also incurs significant costs, primarily in terms of expanded storage requirements. We view our work as a starting point toward creating privacy-friendly hosted services.

Angelos Keromytis is an Associate Professor with the Department of Computer Science at Columbia University, and director of the Network Security Laboratory. He received his B.Sc. in Computer Science from the University of Crete, Greece, and his M.Sc. and Ph.D. from the Computer and Information Science (CIS) Department, University of Pennsylvania. He is the author and co-author of more than 100 papers on refereed conferences and journals, and has served on over 40 conference program committees. He is an associate editor of the ACM Transactions on Information and Systems Security (TISSEC). He recently co-authored a book on using graphics cards for security, and is a co-founder of StackSafe Inc. His current research interests revolve around systems and network security, and cryptography.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10133> nycbug presentation ipv6 gene cronk
<http://www.fetissof.org/public/nycbug/nycbug-10-03-07.mp3> MP3 version mp3 67 minutes 7 Mb
http://www1.cs.columbia.edu/~angelos/Papers/2007/SSARES_ACSAC.pdf Paper pdf 10 pages 443 Kb

Gene Cronk on Implementing IPv6

This talk will be on some of the basics of IPv6 including addressing, subnetting, and tools to test connectivity. There will be a lab (network permitting), and setups for an as of yet undisclosed flavor of BSD as well as some of the well known daemons (Apache 2, SSHD) will be demonstrated. Setting up a BSD OS as an IPv6 router and tunneling system will also be covered.

Bio

Gene Cronk, CISSP-ISSAP, NSA-IAM is a freelance network security consultant, specializing in *NIX solutions. He has been working with computers for well over 20 years, electronics for over 15, and IPv6 specifically for 4 years. He has given talks on IPv6 and a multitude of other topics at DefCon, ShmooCon and other "underground" venues.

Gene is from Jacksonville, FL. When not involved in matters concerning IPv6, he can be found gaming (Anarchy Online), helping out with the [Jacksonville Linux User's Group](#), being one of the benevolent dictators of the [Hacker Pimps Security Think Tank](#), or fixing up his house.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10133> nycbug presentation ipv6 gene cronk
<http://www.fetissof.org/public/nycbug/> nycbug-10-03-07.mp3 MP3 version mp3 60 minutes 14Mb

Using Cryptography to Improve Web Application Performance and Security

Cryptography has a reputation of slowing down applications. However if done correctly, it can actually be used to improve performance by storing high-value/high-cost results “in public.” In addition the same techniques can solve common security problems such as authorization, parameter scanning, and parameter rewriting.

All are welcome - no previous experience with cryptography is required, and the techniques will be presented in a programming-language neutral format.

Nick Galbreath have been working on high performance servers and web security at various high profile startups since 1994 (most recently Right Media). He holds a Master degree of Mathematics from Boston University, and published a book on cryptography. He currently lives in the Lower East Side.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10129> nycbug presentation cryptography nick galbreath
<http://www.fetissof.org/public/nycbug/> nycbug-09-05-07.mp3 MP3 version mp3 18Mb

Marc Spitzer on Nagios

Nagios is a platform for monitoring services and the hosts they reside on. It provides a reasonable tool for monitoring your network and you can not beat the price.

We plan on covering the following topics:

- what it is
- how it works
- where to get it
- how to install it
- how to configure it
- how to customize it for your environment
- where the data is stored
- how to write a basic plug-in

About the Speaker

Marc Spitzer started as a VAX/VMS operator who taught himself some basic scripting in DCL to help me remember how to do procedures that did not come up enough to actually remember all the steps, this was in 1990. Since then he has worked with HP-UX, Solaris, Windows, Linux, and the BSDs, FreeBSD being his favorite. He has held a variety of positions, admin and engineering, where he has been able to introduce BSD into his work place. He currently works for Columbia University as a Systems Administrator.

He is a founding member of NYCBUG and LispNYC and on the board of UNIGroup.

Most of his career has been building tools to solve operational problems, with extra effort going to the ones that irritated him personally. He takes a great deal of pride in not needing a budget to solve most problems.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10122> nycbug presentation nagios marc spitzer
<http://www.fetissof.org/public/nycbug/> nycbug-08-01-07.mp3 MP3 version mp3 19Mb

Isaac ‘Ike’ Levy on the Real Unix Tradition

“The Real Unix Tradition”

UNIX hackers, all standing on the shoulders of giants.

”...the number of UNIX installations has grown to 10, with more expected...” - Dennis Ritchie and Ken Thompson, June 1972

“Well, it was all Open Source, before anybody really called it that”. - Brian Redman, 2003

UNIX is the oldest active and growing computing culture alive today. From it’s humble roots in the back room at Bell Laboratories, to today’s global internet infrastructure- UNIX has consistently been at the core of major advances in computing. Today, the BSD legacy is the most direct continuation of the most successful principles in UNIX, and continues to lead major advances in computing.

Why? What’s so great about UNIX?

This lecture aims to prove that UNIX history is surprisingly useful (and fun)- for developers, sysadmins, and anyone working with BSD systems.

About the speaker

Isaac Levy, (ike) is a freelance BSD hacker based in NYC. He runs Diversaform Inc. as an engine to make his hacking feed itself, (and ike). Diversaform specializes in *BSD based solutions, providing ‘IT special weapons and tatics’ for various sized business clients, as well as running a small high-availability datacenter operation from lower Manhattan. With regard to FreeBSD jail(8), ike was a partner in the first jail (8)-based web hosting ISP in America, iMeme, and has been developing internet applications in and out of jails since 1999. Isaac is a proud member of NYC*BUG (the New York City *BSD Users Group), and a long time member of LESMUUG, (the Lower East Side Mac Unix Users Group).

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10107> nycbug presentation unix tradition isaac levy
<http://www.fetissof.org/public/nycbug/nycbug-07-05-07.mp3> MP3 version mp3 10Mb

Steven Kreuzer on Denial of Service Mitigation Techniques

Protecting your servers, workstations and networks can only go so far. Attacks which consume your available Internet-facing bandwidth, or overpower your CPU, can still take you offline. His presentation will discuss techniques for mitigating the effects of such attacks on servers designed to provide network intensive services such as HTTP or routing.

About the speaker

Steven Kreuzer is currently employed by Right Media as a Systems Administrator focusing on building and managing high transaction infrastructures around the globe. He has been working with Open Source technologies since as long as he can remember, starting out with a 486 salvaged from a dumpster behind his neighborhood computer store. In his spare time he enjoys doing things with technology that have absolutely no redeeming social value.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10108> nycbug presentation denialofservice steven kreuzer
<http://www.fetissof.org/public/nycbug/nycbug-06-06-07.mp3> MP3 version mp3 10Mb

Amitai Schlair on pkgsrcCon.

The fourth annual [pkgsrcCon](#) is April 27-29 in Barcelona. As might be expected when brains congregate, pkgsrcCon traditionally results in a flurry of activity toward new directions and initiatives. Mere hours after returning to New York, Amitai will give us a recap of [the proceedings](#), including his presentation, “Packaging djbware.”

Amitai Schlair is a pkgsrc developer who has worked in such diverse areas as Mac OS X platform support and packages of software by Dan Bernstein. His full-time undergraduate studies at Columbia are another contributing factor to his impending insanity. He consults in software and IT.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10102> nycbug presentation pkgsrccon netbsd amitai schlair
<http://www.fetissof.org/public/nycbug/> nycbug-05-02-07.mp3 MP3 version mp3 21Mb

Ray Lai: on OpenCVS

This presentation was inspired by the recent Subversion presentation. It will talk about the origins of OpenRCS and OpenCVS, its real-world usage in the OpenBSD project, and why OpenBSD will continue to use CVS.

Ray is an OpenBSD developer who uses Subversion by day, CVS by night. Taking the phrase “complexity is the enemy of security” to heart, he believes that the beauty of UNIX’s security is in its simplicity.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10104> nycbug presentation cvs openbsd ray lai
<http://www.fetissof.org/public/nycbug/> nycbug-04-04-07.mp3 MP3 version mp3

Matthew Burnside: Integrated Enterprise Security Mgmt

Integrated Enterprise Security Management

Security policies are a key component in protecting enterprise networks. But, while there are many diverse defensive options available, current models and mechanisms for mechanically-enforced security policies are limited to traditional admission-based access control. Defensive capabilities include among others logging, firewalls, honeypots, rollback/recovery, and intrusion detection systems, while policy enforcement is essentially limited to one-off access control. Furthermore, access-control mechanisms operate independently on each service, which can (and often does) lead to inconsistent or incorrect application of the intended system-wide policy. We propose a new scheme for global security policies. Every policy decision is made with near-global knowledge, and re-evaluated as global knowledge changes. Using a variety of actuators, we make the full array of defensive capabilities available to the global policy. Our goal is a coherent, enterprise-wide response to any network threat.

Biography

Matthew Burnside is a Ph.D. student in the Computer Science department at Columbia University, in New York. He works for Professor Angelos Keromytis in the Network Security Lab. He received his B.A and M.Eng from MIT in 2000, and 2002, respectively. His main research interests are in computer security, trust management, and network anonymity.

<http://www.nycbug.org/index.php?NAV=Home;SUBM=10089> mp3 presentation enterprise security matthew burnside
<http://www.fetissof.org/public/nycbug/> nycbug-03-07-07.mp3 MP3 version mp3

Ivan Ivanov on The Version Control System Subversion

The presentation will discuss Subversion from both client and server points of view. It will show how to create repositories and how to make them accessible over the network using different access schemes like <http://>, <file://> or <svn://>. Pointers are given on securing the repositories and on authenticating and authorizing the clients. Next, the presentation shows how an user interacts with the repository and describes some of the important Subversion client commands. Finally, it deals with administrating the repository using “hook scripts”.

Ivan Ivanov is generally interested in Version Control Systems since his student years in Sofia University, Bulgaria, where he set up and maintained a CVS server for an academic project. When Subversion became a fact and proved to be “a better CVS” he researched it and last year deployed it for his NYC-based employer Ariel Partners (<http://www.arielpartners.com/>). He integrated the Subversion repositories with Apache Web Server over https to enable a reliable and secure way to access them from any point.

<http://www.fetissof.org/public/nycbug/> nycbug presentation subversion ivan ivanov
<http://www.fetissof.org/public/nycbug/> nycbug-02-07-07.mp3 MP3 version nycbug presentation subversion ivan ivanov

Okan Demirmen on PF

We have had lots of meetings that have peripherally discussed OpenBSD's wildly popular PF firewall... but finally we will have a meeting focused on it. <http://www.fetissof.org/public/nycbug/> nycbug presentation opensbsd pf okan demirmen <http://www.fetissof.org/public/nycbug/nycbug-01-03-07.mp3> MP3 version mp3

New York City BSD Con 2006: BSD is Dying - A Cautionary Tale of Sex and Greed

BSD is Dying

A Cautionary Tale of Sex and Greed

Jason Dixon

October 28, 2006

First and foremost, I would like to thank the unique presentation styles of Dick Hardt and Lawrence Lessig for inspiring me to create this presentation.

The following videos were created by exporting the original Keynote presentation slides into QuickTime video, then manually synchronizing them using iMovie HD with the audio recordings captured by Nikolai Fetissov. They were then exported into QuickTime, mpeg4 (H.264/AAC), and iPod movie formats. If you are having difficulties with the MP4 copy, and are unable to view QuickTime movies, please contact me and I'll try to assist.

<http://talks.dixongroup.net/nycbsdcon2006/> nycbug presentation humor bsd is dying jason dixon
http://talks.dixongroup.net/nycbsdcon2006/BSD_is_Dying_640x480.mov 19Mb QuickTime mov
[BSD_is_Dying_640x480.mp4](http://talks.dixongroup.net/nycbsdcon2006/BSD_is_Dying_640x480.mp4) 31Mb MP4 mp4 [BSD_is_Dying_640x480.m4v](http://talks.dixongroup.net/nycbsdcon2006/BSD_is_Dying_640x480.m4v) 36Mb iPod m4v

New York City BSD Con 2006

Audio recordings of presentations given at New York City BSD Conference 2006. Courtesy of nikolai at fetissof.org. The main page also has links to the slides. <http://www.fetissof.org/public/nycbsdcon06/> nycbug nycbsdcon nycbsdcon2006 presentation <http://www.fetissof.org/public/nycbsdcon06/1.1.mp3> 14 Mb Corey Benninger: Security with Ruby on Rails in BSD mp3 ruby ruby on rails security corey benninger 1.2.mp3 10 Mb Brian A. Seklecki: A Framework for NetBSD Network Appliances. mp3 netbsd brian a seklecki 1.3.mp3 15 Mb Bob Beck: PF, it is not just for firewalls anymore. mp3 pf bob beck 1.4.mp3 9 Mb Bjorn Nelson: A Build System for FreeBSD mp3 freebsd bjorn nelson 1.5.mp3 13 Mb Johnny C. Lam: The "hidden dependency" problem. mp3 johnny c lam 1.6.mp3 11 Mb Marco Peereboom: Bio & Sensors in OpenBSD. mp3 opensbsd sensors marco peerenboom 1.7.mp3 12 Mb Russell Sutherland: BSD on the Edge of the Enterprise. mp3 russel sutherland 1.8.mp3 5 Mb Jason Dixon: BSD Is Dying. mp3 humor bsd is dying jason dixon 2.1.mp3 9 Mb Jason Wright: OpenBSD on sparc64. mp3 opensbsd sparc64 jason wright 2.2.mp3 15 Mb Kristaps Johnson: BSD Virtualisation with sysjail. mp3 sysjail kristaps johnson 2.3.mp3 16 Mb Wietse Venema: Postfix as a Secure Programming Example. mp3 postfix wietse venema 2.4.mp3 16 Mb Bob Beck: spamd - spam deferral daemon. mp3 spamd bob beck

Isaac 'Ike' Levy on m0n0wall and PFSense

UNIX professionals are busy these days. Setting up routers and firewalls are fundamental to any network, but in environments where the focus is on various applications, (servers, workstations, and the software that runs on them), it's difficult for a business not to choose off-the-shelf SOHO routers and networking gear. The web management GUIs are understandable by everyone, (even techs without UNIX knowledge), and the gear is cheap - this saves time and money.

In the meantime, the features of your average Linksys or Netgear router often leave MUCH to be desired, (https auth management, for one simple example).

Enter m0n0wall and PFSense, 2 BSD based packaged router/firewall solutions that are as solid and full featured as you'd expect from any BSD system- PLUS THEY HAVE HTML WEB INTERFACES FOR MANAGEMENT!

m0n0wall and PFSense become an easy sell in any small professional environment, any competent tech can manage the network within minutes... At home, in every hackers home network, they free the hacker to have trusted tools available, but are as time-saving as using any Linksys router.

m0n0wall and PFSense are both light and clean, designed to run on embedded systems- (Soekris, WRAP), but are monsters when unleashed on even legacy PCs around the office. If you manage UNIX networks and systems all day, do you really want to manage the router for your DSL when you get home? But then doesn't it bug you to use a chincey Linksys box?

Ike has been a member of NYC*BUG since we first launched in January 2004. He is a long-time member of the Lower East Side Mac Unix User Group. He has spoken frequently on a number of topics at various venues, particularly on the issue of FreeBSD's jail (8). nycbug presentation monowall pfsense isaac levy
<http://www.fetissof.org/public/nycbug/nycbug-09-06-06.mp3> 9 Mb mp3

Alfred Perlstein on Sendmail Hacks

Alfred will discuss the hacks used to turn Sendmail into a high performance solution for delivering millions of messages to OKCupid's subscribers. Topics covered will be system tuning and sendmail hacks used in house to achieve massive throughput.

Alfred Perlstein is the CTO of OKCupid.com, the largest free online dating site. He has been a FreeBSD hacker for five years, he's worked on NFS, VFS, pthreads, networking and general system maintenance during his tenure on both FreeBSD and OS X kernels. nycbug presentation sendmail alfred perlstein
<http://www.fetissof.org/public/nycbug/nycbug-07-05-06.mp3> 11 Mb mp3

Episode 07 of "FreeBSD for all" uploaded

This week we talk about podcast clients, ipfw firewall etc. <http://freebsdforall.blogspot.com/2006/07/episode-07.html>
 freebsd for all talk podcast clients ipfw http://www.archive.org/download/FreeBSD_for_all_podcast_Episode_07/FreeBSD_for_all_podcast_Episode_07.mp3 11 Mb 23 minutes 128 kbps MP3 version mp3
[FreeBSD_for_all_podcast_Episode_07_64kb.mp3](http://www.archive.org/download/FreeBSD_for_all_podcast_Episode_07/FreeBSD_for_all_podcast_Episode_07_64kb.mp3) 23 minutes 64 kbps MP3 version mp3
[FreeBSD_for_all_podcast_Episode_07.ogg](http://www.archive.org/download/FreeBSD_for_all_podcast_Episode_07/FreeBSD_for_all_podcast_Episode_07.ogg) 23 minutes Ogg version ogg

Episode 06 of "FreeBSD for all" uploaded

This week we talk about

- Macromedia plugin
- FreeBSD-Linux differences part 2
- John Baldwin Introduction
- Podcast announcement - call for co-hosts!

<http://freebsdforall.blogspot.com/2006/06/episode-06.html> freebsd for all talk john baldwin
 freebsd vs linux http://www.archive.org/download/FreeBSD_for_all_podcast_Episode_06/FreeBSD_for_all_podcast_Episode_06.mp3 MP3 version mp3 [FreeBSD_for_all_podcast_Episode_06.ogg](http://www.archive.org/download/FreeBSD_for_all_podcast_Episode_06/FreeBSD_for_all_podcast_Episode_06.ogg) Ogg version ogg [FreeBSD_for_all_podcast_Episode_06_64kb.mp3](http://www.archive.org/download/FreeBSD_for_all_podcast_Episode_06/FreeBSD_for_all_podcast_Episode_06_64kb.mp3) 64 kbps MP3 version mp3

Nate Lawson on ACPI

Our Topic: FreeBSD's ACPI implementation: The details.

Our Speaker: Nate Lawson, FreeBSD Committer.

Our Topic: FreeBSD's ACPI implementation is based on code for ACPI released by Intel. Nate and others wrote the glue code to make this code work on FreeBSD. He explains how this was done, and why. bafug

presentation freebsd acpi nate lawson <http://people.freebsd.org/~julian/BAFUG/talks/ACPI/bafug7-nate2.mov>
245 Mb mov

Network Protocol Development Tools and Techniques for FreeBSD

Our Topic: Network Protocol Development Tools and Techniques for FreeBSD

Our Speaker: George Neville-Neil, co-author of the “Design and Implementation of the FreeBSD Operating System” “daemon” book.

Our Topic: While computers have gotten faster and more powerful the tools we use to develop network protocols, such as TCP, UDP, IPv4 and IPv6 have not. Most network protocols are developed, in C, in the kernel, and require a lot of work to test. Over the past year or so I have been working with virtual machines, a couple of pieces of open source software, and begun developing a library for use in protocol testing. This talk will cover three topics:

1. Developing and testing kernel code with Virtual Machines
2. Finding good tests for networking code
3. Packet Construction Set (PCS) a new library for writing protocol tests

bafug presentation freebsd packet construction set george neville-neil <http://people.freebsd.org/~julian/BAFUG/talks/bafug6-gnn.mov> 211 Mb mov

Tim Kientzler on developing libarchive and tar

libarchive.....Tim Kientzler on developing libarchive and tar. <http://people.freebsd.org/~julian/BAFUG/talks/libarchive/bafug-presentation-libarchive-tim-kientzler> <http://people.freebsd.org/~julian/BAFUG/talks/libarchive/bafug5-tim-1.mov> 50 Mb Part 1 bafug presentation libarchive tim kientzler [bafug5-tim-2.mov](http://people.freebsd.org/~julian/BAFUG/talks/libarchive/bafug5-tim-2.mov) 125 Mb Part 2 bafug presentation libarchive tim kientzler [bafug5-tim-3.mov](http://people.freebsd.org/~julian/BAFUG/talks/libarchive/bafug5-tim-3.mov) 30 Mb Part 3 bafug presentation libarchive tim kientzler

Fosdem 2006: BSD

We talk with Daniel Seuffert about BSD. Several flavours of BSD were represented in a joint BSD booth: OpenBSD, FreeBSD, NetBSD and MirOS. Daniel is representative of the FreeBSD project and among other things talks about the different operating systems that are build on top of FreeBSD. For instance, there are two distributions called PC-BSD and DesktopBSD that are targeted towards desktop users. There also is a version that specializes on security entitled TrustedBSD. <http://www.source21.nl/2006/06/05/fosdem-2006-bsd/> source21 interview daniel seuffert http://www.source21.nl/media/20060605/bsd_-_daniel_seuffert.mp4 mp4

COMPLETE Hard Disk Encryption with FreeBSD

COMPLETE Hard Disk Encryption with FreeBSD, by Marc Schiesser

Learn how to effectively protect not only your data but also your applications.

Most technologies and techniques intended for securing digital data focus on protection while the machine is turned on mostly by defending against remote attacks. An attacker with physical access to the machine, however, can easily circumvent these defenses by reading out the contents of the storage medium on a different, fully accessible system or even compromise program code on it in order to leak encrypted information. Especially for mobile users, that threat is real. And for those carrying around sensitive data, the risk is most likely high. This talk will introduce a method of mitigating that particular risk by protecting not only the data through encryption, but also the applications and the operating system from being compromised while the machine is turned off.

<http://events.ccc.de/congress/2005/fahrplan/events/1139.en.html> ccc ccc2005 ccc22 presentation freebsd hddisk encryption marc schiesser http://events.ccc.de/congress/2005/fahrplan/attachments/687-slides_Complete_Hard_Disk_Encryption.pdf 679Kb Slides slides http://events.ccc.de/congress/2005/fahrplan/attachments/905-22C3-1139-en-complete_hddisk_encryption_with_freebsd.mp4.torrent 37Kb Bittorrent link mp4

FreeBSD Security Officer funktionen

“FreeBSD Security Officer funktionen” at the AAUUG, AAUUG, 22 August 2006 by Simon L. Nielsen (FreeBSD Deputy Security Officer) [http://www.aauug.dk/aauug presentation danish freebsd security officer simon l nielsen](http://www.aauug.dk/aauug%20presentation%20danish%20freebsd%20security%20officer%20simon%20l%20nielsen) <http://people.freebsd.org/~simon/presentations/freebsd-so-function-aauug-2006-08-22.pdf> 211 Kb PDF (danish) pdf

FreeBSD Security Officer funktionen

“FreeBSD Security Officer funktionen” at the BSD-DK, 26 August 2006 by Simon L. Nielsen (FreeBSD Deputy Security Officer) <http://people.freebsd.org/~simon/presentations/freebsd-so-function-bsd-dk-2006-08.pdf> 210 Kb PDF (danish) pdf

Releaseparty, the Varnish HTTP accelerator

VG sponsored the creation of a web-accellerator called “Varnish” because Squid was too slow for them. Varnish is being developed by Poul-Henning Kamp and the Norwegian Linux consultancy Linpro. This is the releaseparty for version 1.0.

The first half of the talk will introduce Varnish and present some of the novel features it brings to the business of web-serving.

The second half of the talk, using Varnish as the example, will show ways to get the most performance out of modern hardware and operating systems.

(The English text starts at about 5 minutes in the stream)

<http://www.nuug.no/aktiviteter/20060919-varnish/> nuug presentation varnish poul-henning kamp
<http://www.nuug.no/pub/video/published/20060919-varnish.mpeg> 230 Mb Video version mpeg 20060919-varnish.mp3 47.8 Mb MP3 version mp3

OpenBSD 4.5 Release Songs - Games

[Commentary still being written]

For RSS readers: Please note that the download URL is an FTP site.

<http://www.openbsd.org/lyrics.html#45> openbsd artwork <ftp://ftp.openbsd.org/pub/OpenBSD/songs/song45.mp3> 6.4 Mb 3:29 minutes MP3 version openbsd artwork [song45.ogg](ftp://ftp.openbsd.org/pub/OpenBSD/songs/song45.ogg) 4.5 Mb 3:29 minutes Ogg version openbsd artwork

OpenBSD 4.0 Release Songs - OpenVOX

This is an extra track by the artist Ty Semaka (who really has “had Puffy on his mind”) which we included on the audio CD.

This song details the process that Ty has to go through to make the art and music for each OpenBSD release. Ty and Theo really do go to a (very specific) bar and discuss what is going on in the project, and then try to find a theme that will work...

For RSS readers: Please note that the download URL is an FTP site.

http://www.openbsd.org/lyrics.html#audio_extra openbsd artwork <ftp://ftp.openbsd.org/pub/OpenBSD/songs/songty.mp3> 3.9 Mb 4 minutes MP3 version openbsd artwork [songty.ogg](ftp://ftp.openbsd.org/pub/OpenBSD/songs/songty.ogg) 6.0 Mb 4 minutes Ogg version openbsd artwork

OpenBSD 4.4 Release Song - “Source Wars - Episode IV - Trial of the BSD Knights”

Nearly 10 years ago Kirk McKusick wrote a history of the Berkeley Unix distributions for the O'Reilly book “Open Sources: Voices from the Open Source Revolution”. We recommend you read his story, entitled “Twenty Years of Berkeley Unix From AT&T-Owned to Freely Redistributable” first, to see how Kirk remembers how we got here. Sadly, since it showed up in book form originally, this text has probably not been read by enough people.

The USL(AT&T) vs BSDI/UCB court case settlement documents were not public until recently; their disclosure has made the facts more clear. But the story of how three people decided to free the BSD codebase of corporate pollution

– and release it freely – is more interesting than the lawsuit which followed. Sure, a stupid lawsuit happened which hindered the acceptance of the BSD code during a critical period. But how did a bunch of guys go through the effort of replacing so much AT&T code in the first place? After all, companies had lots of really evil lawyers back then too – were they not afraid?

After a decade of development, most of the AT&T code had already been replaced by university researchers and their associates. So Keith Bostic, Mike Karels and Kirk McKusick (the main UCB CSRG group) started going through the 4.3BSD codebase to cleanse the rest. Keith, in particular, built a ragtag team (in those days, USENIX conferences were a gold mine for such team building) and led these rebels to rewrite and replace all the Imperial AT&T code, piece by piece, starting with the libraries and userland programs. Anyone who helped only got credit as a Contributor – people like Chris Torek and a cast of .. hundreds more.

Then Mike and Kirk purified the kernel. After a bit more careful checking, this led to the release of a clean tree called Net/2 which was given to the world in June 1991 – the largest dump of free source code the world had ever received (for those days – not modern monsters like OpenOffice).

Some of these ragtags formed a company (BSDi) to sell a production system based on this free code base, and a year later Unix System Laboratories (basically AT&T) sued BSDi and UCB. Eventually AT&T lost and after a few trifling fixes (described in the lawsuit documents) the codebase was free. A few newer developments (and more free code) were added, and released in June 1994 as 4.4BSD-Lite. Just over 14 years later OpenBSD is releasing its own 4.4 release (and for a lot less than \$1000 per copy).

The OpenBSD 4.4 release is dedicated to Keith Bostic, Mike Karels, Kirk McKusick, and all of those who contributed to making Net/2 and 4.4BSD-Lite free.

<http://www.openbsd.org/lyrics.html#44> openbsd artwork <ftp://ftp.openbsd.org/pub/OpenBSD/songs/song44.mp3> 5.6 Mb 3 minutes 5 seconds MP3 version [mp3 song44.ogg](#) 4.4 Mb 3 minutes 5 seconds Ogg version [ogg](#)

OpenBSD 4.3 Release Song - “Home to Hypocrisy”

We are just plain tired of being lectured to by a man who is a lot like Naomi Campbell.

In 1998 when a United Airlines plane was waiting in the queue at Washington Dulles International Airport for take-off to New Orleans (where a Usenix conference was taking place), one man stood up from his seat, demanded that they stop waiting in the queue and be permitted to deplane. Even after orders from the crew and a pilot from the cockpit he refused to sit down. The plane exited the queue and returned to the airport gangway. Security personnel ran onto the plane and removed this man, Richard Stallman, from the plane. After Richard was removed from the plane, everyone else stayed onboard and continued their journey to New Orleans. A few OpenBSD developers were on that same plane, seated very closeby, so we have an accurate story of the events.

This is the man who presumes that he should preach to us about morality, freedom, and what is best for us. He believes it is his God-given role to tell us what is best for us, when he has shown that he takes actions which are not best for everyone. He prefers actions which he thinks are best for him – and him alone – and then lies to the public. Richard Stallman is no Spock.

We release our software in ways that are maximally free. We remove all restrictions on use and distribution, but leave a requirement to be known as the authors. We follow a pattern of free source code distribution that started in the mid-1980’s in Berkeley, from before Richard Stallman had any powerful influence which he could use so falsely.

We have a development sub-tree called “ports”. Our “ports” tree builds software that is ‘found on the net’ into packages that OpenBSD users can use more easily. A scaffold of Makefiles and scripts automatically fetch these pieces of software, apply patches as required by OpenBSD, and then build them into nice neat little tarballs. This is provided as a convenience for users. The ports tree is maintained by OpenBSD entirely separately from our main source tree. Some of the software which is fetched and compiled is not as free as we would like, but what can we do. All the other operating system projects make exactly the same decision, and provide these same conveniences to their users.

Richard felt that this “ports tree” of ours made OpenBSD non-free. He came to our mailing lists and lectured to us specifically, yet he said nothing to the many other vendors who do the same; many of them donate to the FSF and

perhaps that has something to do with it. Meanwhile, Richard has personally made sure that all the official GNU software – including Emacs – compiles and runs on Windows.

That man is a false leader. He is a hypocrite. There may be some people who listen to him. But we don't listen to people who do not follow their own stupid rules.

<http://www.openbsd.org/lyrics.html#43> openbsd artwork <ftp://ftp.openbsd.org/pub/OpenBSD/songs/> song43.mp3 8.2 Mb 4 minutes 48 seconds MP3 version mp3 song43.ogg 6.5 Mb 4 minutes 48 seconds Ogg version ogg

OpenBSD 4.2 Release Song - “100001 1010101”

Those of us who work on OpenBSD are often asked why we do what we do. This song's lyrics express the core motivations and goals which have remained unchanged over the years - secure, free, reliable software, that can be shared with anyone. Many other projects purport to share these same goals, and love to wrap themselves in a banner of “Open Source” and “Free Software”. Given how many projects there are one would think it might be easy to stick to those goals, but it doesn't seem to work out that way. A variety of desires drag many projects away from the ideals very quickly.

Much of any operating system's usability depends on device support, and there are some very tempting alternative ways to support devices available to those who will surrender their moral code. A project could compromise by entering into NDA agreements with vendors, or including binary objects in the operating system for which no source code exists, or tying their users down with contract terms hidden inside copyright notices. All of these choices surrender some subset of the ideals, and we simply will not do this. Sure, we care about getting devices working, but not at the expense of our original goals.

Of course since “free to share with anyone” is part of our goals, we've been at the forefront of many licensing and NDA issues, resulting in a good number of successes. This success had led to much recognition for the advancement of Free Software causes, but has also led to other issues.

We fully admit that some BSD licensed software has been taken and used by many commercial entities, but contributions come back more often than people seem to know, and when they do, they're always still properly attributed to the original authors, and given back in the same spirit that they were given in the first place.

That's the best we can expect from companies. After all, we make our stuff so free so that everyone can benefit – it remains a core goal; we really have not strayed at all in 10 years. But we can expect more from projects who talk about sharing – such as the various Linux projects.

Now rather than seeing us as friends who can cooperatively improve all codebases, we are seen as foes who oppose the GPL. The participants of “the race” are being manipulated by the FSF and their legal arm, the SFLC, for the FSF's aims, rather than the goal of getting good source into Linux (and all other code bases). We don't want this to come off as some conspiracy theory, but we simply urge those developers caution – they should ensure that the path they are being shown by those who have positioned themselves as leaders is still true. Run for yourself, not for their agenda.

The Race is there to be run, for ourselves, not for others. We do what we do to run our own race, and finish it the best we can. We don't rush off at every distraction, or worry how this will affect our image. We are here to have fun doing right.

<http://www.openbsd.org/lyrics.html#42> openbsd artwork <ftp://ftp.openbsd.org/pub/OpenBSD/songs/> song42.mp3 4.0 Mb 4 minutes 40 seconds MP3 version mp3 song42.ogg 6.4 Mb 4 minutes 4- seconds Ogg version ogg

OpenBSD 4.1 Release Song - Puffy Baba and the 40 Vendors

As developers of a free operating system, one of our prime responsibilities is device support. No matter how nice an operating system is, it remains useless and unusable without solid support for a wide percentage of the hardware that is available on the market. It is therefore rather unsurprising that more than half of our efforts focus on various aspects relating to device support.

Most parts of the operating system (from low kernel, through to libraries, all the way up to X, and then even to applications) use fairly obvious interface layers, where the “communication protocols” or “argument passing” mechanisms

(ie. APIs) can be understood by any developer who takes the time to read the free code. Device drivers pose an additional and significant challenge though: because many vendors refuse to document the exact behavior of their devices. The devices are black boxes. And often they are surprisingly weird, or even buggy.

When vendor documentation does not exist, the development process can become extremely hairy. Groups of developers have found themselves focused for months at a time, figuring out the most simple steps, simply because the hardware is a complete mystery. Access to documentation can ease these difficulties rapidly. However, getting access to the chip documentation from vendors is ... almost always a negotiation. If we had open access to documentation, anyone would be able to see how simple all these devices actually are, and device driver development would flourish (and not just in OpenBSD, either).

When we proceed into negotiations with vendors, asking for documentation, our position is often weak. One would assume that the modern market is fair, and that selling chips would be the primary focus of these vendors. But unfortunately a number of behemoth software vendors have spent the last 10 or 20 years building [political hurdles against the smaller players](#).

A particularly nasty player in this regard has been the Linux vendors and some Linux developers, who have played along with an American corporate model of requiring NDAs for chip documentation. This has effectively put Linux into the club with Microsoft, but has left all the other operating system communities – and their developers – with much less available clout for requesting documentation. In a more fair world, the Linux vendors would work with us, and the device driver support in all free operating systems would be fantastic by now.

We only ask that [users help](#) us in changing the political landscape.

<http://www.openbsd.org/lyrics.html#41> openbsd artwork <ftp://ftp.openbsd.org/pub/OpenBSD/songs/song41.mp3> 4.1 Mb 4 minutes 19 seconds MP3 version mp3 song41.ogg 8.3 Mb 4 minutes 19 seconds Ogg version ogg

OpenBSD 4.0 Release Song - Humppa negala

The last 10 years, every 6 month period has (without fail) resulted in an official OpenBSD release making it to the FTP servers. But CDs are also manufactured, which the project sells to continue our development goals.

While tests of the release binaries are done by developers around the world, Theo and some developers from Calgary or Edmonton (such as Peter Valchev or Bob Beck) test that the discs are full of (only) correct code. Ty Semaka works for approximately two months to design and draw artwork that will fit the designated theme, and coordinates with his music buddies to write and record a song that also matches the theme.

Then the discs and all the artwork gets delivered to the plant, so that they can be pressed in time for an official release date.

This release, instead of bemoaning vendors or organizations that try to make our task of writing free software more difficult, we instead celebrate the 10 years that we have been given (so far) to write free software, express our themes in art, and the 5 years that we have made music with a group of talented musicians.

OpenBSD developers have been torturing each other for years now with Humppa-style music, so this release our users get a taste of this too. Sometimes at hackathons you will hear the same songs being played on multiple laptops, out of sync. It is under such duress that much of our code gets written.

We feel like Pufferix and Bobilix delivering The Three Discs of Freedom to those who want them whenever the need arises, then returning to celebrate the (unlocked) source tree with all the other developers.

For RSS readers: Please note that the download URL is an FTP site.

<http://www.openbsd.org/lyrics.html#40> openbsd artwork <ftp://ftp.openbsd.org/pub/OpenBSD/songs/song40.mp3> 2.3 Mb 2 minutes 40 seconds MP3 version mp3 song40.ogg 3.6 Mb 2 minutes 40 seconds Ogg version ogg

EuroBSDCon 2006 pictures

EuroBSDCon 2006 pictures by Christian Laursen <http://photos.borderworlds.dk/eurobsdcon-2006/> eurobsdcon eurobsdcon2006 photos christian laursen

EuroBSDCon 2006 pictures

EuroBSDCon 2006 pictures by Erwin Lansing (erwin@) <http://foto.droso.org/2006/20061108-13/> eurobsdcon eurobsdcon2006 photos erwin lanning

Discussion - What's cooking for FreeBSD 7.0?

Discussion - What's cooking for FreeBSD 7.0? (Bulgarian) <http://openfest.org/archive/openfest-2007/> openfest openfest2007 discussion freebsd freebsd7 <http://ludost.net/of2007/d2h217.avi> 105 Mb AVI avi

Dimitri Vasileva - Visualizing Security Threats with Social Networking Software

Dimitri Vasileva - Visualizing Security Threats with Social Networking Software (Bulgarian) <http://openfest.org/archive/openfest-2007/> openfest openfest2007 presentation freebsd security social networking dimitri vasileva <http://ludost.net/of2007/d2h216.avi> 331 Mb AVI avi

Shcheryana Shopova - SNMP monitoring

Shcheryana Shopova - SNMP monitoring (Bulgarian) <http://openfest.org/archive/openfest-2007/> openfest openfest2007 presentation freebsd snmp monitoring shcheryana shopova <http://ludost.net/of2007/d2h215.avi> 271 Mb AVI avi

Willow Vachkov - FreeBSD and the new network and transport protocols (IPv6 and SCTP)

Willow Vachkov - FreeBSD and the new network and transport protocols (IPv6 and SCTP) (Bulgarian) <http://openfest.org/archive/openfest-2007/> openfest openfest2007 presentation freebsd ipv6 sctp willow vachkov <http://ludost.net/of2007/d2h214.avi> 251 Mb AVI avi

Atanas Bchvarov - Packet Filtering in FreeBSD

Atanas Bchvarov - Packet Filtering in FreeBSD (Bulgarian) <http://openfest.org/archive/openfest-2007/> openfest openfest2007 presentation freebsd atanas bchvarov <http://ludost.net/of2007/d2h213.avi> 186 Mb AVI avi

Nikolai Denev - FreeBSD goes Zettabyte

Nikolai Denev - FreeBSD goes Zettabyte (Bulgarian) <http://openfest.org/archive/openfest-2007/> openfest openfest2007 presentation freebsd zettabyte nikolai denev <http://ludost.net/of2007/d2h212.avi> 358 Mb AVI avi

Vasil Dimov - The FreeBSD ports collection - tips and tricks

Vasil Dimov - The FreeBSD ports collection - tips and tricks (Bulgarian) <http://openfest.org/archive/openfest-2007/> openfest openfest2007 presentation freebsd ports collection vasil dimov <http://ludost.net/of2007/d2h211.avi> 341 Mb AVI avi

FreeBSD ports Erwin Lansing

Case study : managing a worldwide open source project: FreeBSD port manager <http://openfest.org/program/> openfest openfest2006 presentation freebsd port manager erwin lanning <http://people.freebsd.org/~erwin/presentations/FreeBSD-portmgr-20061105-OpenFest.pdf> 128 Kb PDF pdf

The Linux Link Tech Show Episode 179

Special Guests Will Backman and Scott Ruecker. Will's talks about his podcast bsdtalk and about Linux and BSD in general. We are joined by Troels also. Dann on Devede and hopes for MythTV. Scott Ruecker talks about Scale and general linux and lxxer stuff. linux link tech show talk will backman http://www.tlts.org/audio/tlts_179-02-14-07.mp3 31 Mb 120 minutes MP3 version mp3

Ham Radio on FreeBSD

Last month I attended a meeting of the Ottawa Amateur Radio Club (OARC) as a member of my local BUG was giving a presentation on Ham Radio on FreeBSD. Diane Bruce, call sign VA3DB, has had her operator license since 1969 and is well known in the BSD community and for the development of ircd-hybrid. In the past year she has assisted in the creation of the [Hamradio category in the FreeBSD ports tree](#) and has become the maintainer of over 20 of the hamradio ports. She also contributed to the [FreeBSD entry at Hampedia](#), the Wikipedia for ham operators.

Her presentation slides are a great introduction to the various ham utilities which are available, including both descriptions and screenshots of the utilities in action.

oarc presentation radio diane bruce http://www.oarc.net/presentations/hamradio_on_freebsd.pdf 23 pages PDF file mp3

Installing OpenBSD in 5 minutes

<http://unix-tutorial.blogspot.com/2007/04/installing-openbsd-in-5-minutes.html> Installing OpenBSD. In real time :)
unix-tutorial flash openbsd

FreeBSD: Hard disk encryption

<http://unix-tutorial.blogspot.com/2007/02/freebsd-hard-disk-encryption.html> How to protect your data on FreeBSD machine even when your computer is turned off? This hard disk encryption guide will help. unix-tutorial flash freebsd encryption

FreeBSD: First time install and configure

<http://unix-tutorial.blogspot.com/2007/01/freebsd-first-time-install-and.html> Tutorial how to install and configure FreeBSD. It seems that comments in video are in Japanese :) unix-tutorial flash freebsd

FreeBSD: using ports system

<http://unix-tutorial.blogspot.com/2007/01/freebsd-using-ports-system.html> Using ports system in FreeBSD to install etherape. unix-tutorial flash freebsd ports

FreeBSD installation

<http://unix-tutorial.blogspot.com/2007/01/freebsd-installation.html> Step-by-step installation of FreeBSD operating system. unix-tutorial flash freebsd

NetBSD and sshfs

<http://unix-tutorial.blogspot.com/2007/04/netbsd-and-sshfs.html> Usage of sshfs on NetBSD with PUFFS. unix-tutorial flash netbsd puffs

Install Debian and NetBSD on Xen Domu

<http://unix-tutorial.blogspot.com/2007/04/install-debian-and-netbsd-on-xen-domu.html> Video tutorial on installation of Debian and NetBsd on Domu with Xen. unix-tutorial flash netbsd xen debian

NetBSD. More CPUs than Linux. + BSD ports/packages.

http://www.berklx.com/free/talk/faraday/presentations/source/3_netbsd_marc/ From the talks with subject “Free Alternatives To Microsoft” comes “NetBSD. More CPUs than Linux. + BSD ports/packages”. berklx netbsd packages

Chris Buechler and Scott Ullrich - pfSense: 2.0 and beyond

<http://www.bsdcn.org/2009/schedule/events/130.en.html>

pfSense: 2.0 and beyond

From firewall distribution to appliance building platform

pfSense is a BSD licensed customized distribution of FreeBSD tailored for use as a firewall and router. In addition to being a powerful, flexible firewalling and routing platform, it includes a long list of related features and a package system allowing further expandability without adding bloat and potential security vulnerabilities to the base distribution.

This session will start with an introduction to the project and its common uses, which have expanded considerably beyond firewalling. We will cover much of the new functionality coming in the 2.0 release, which contains significant enhancements to nearly every portion of the system as well as numerous new features.

While the primary function of the project is a firewalling and routing platform, with changes coming in pfSense 2.0, it has also become an appliance building framework enabling the creation of customized special purpose appliances. The m0n0wall code where pfSense originated has proved popular for this purpose, with AskoziaPBX and FreeNAS also based upon it, in addition to a number of commercial solutions. The goal of this appliance building framework is to enable creation of projects such as these without having to fork and maintain another code base. The existing appliances, including a DNS server using TinyDNS, VoIP with FreeSWITCH, and others will be discussed. For those interested in creating appliances, an overview of the process will be provided along with references for additional information.

bsdcan bsdcan2009 presentation pfsense chris buechler scott ullrich http://www.bsdcan.org/2009/schedule/attachments/94_pfSense_2_0.pdf
36 pages 3.2 Mb Slides pdf

Luigi Rizzo - GEOM based disk schedulers for FreeBSD

<http://www.bsdcan.org/2009/schedule/events/122.en.html>

GEOM based disk schedulers for FreeBSD

The high cost of seek operations makes the throughput of disk devices very sensitive to the offered workload. A disk scheduler can then help reorder requests to improve the overall throughput of the device, or improve the service guarantees for individual users, or both.

Research results in recent years have introduced, and proven the effectiveness of, a technique called “anticipatory scheduling”. The basic idea behind this technique is that, in some cases, requests that cause a seek should not be served immediately; instead, the scheduler should wait for a short period of time in case other requests arrive that do not require a seek to be served. With many common workloads, dominated by sequential synchronous requests, the potential loss of throughput caused by the disk idling times is more than balanced by the overall reduction of seeks.

While a fair amount of research on disk scheduling has been conducted on FreeBSD, the results were never integrated in the OS, perhaps because the various prototype implementations were very device-specific and operated within the device drivers. Ironically, anticipatory schedulers are instead a standard part of Linux kernels.

This talk has two major contributions:

First, we will show how, thanks to the flexibility of the GEOM architecture, an anticipatory disk scheduling framework has been implemented in FreeBSD with little or no modification to a GENERIC kernel. While these schedulers operate slightly above the layer where one would naturally put a scheduler, they can still achieve substantial performance improvements over the standard disk scheduler; in particular, even the simplest anticipatory schedulers can prevent the complete trashing of the disk performance that often occurs in presence of multiple processes accessing the disk.

Secondly, we will discuss how the basic anticipatory scheduling technique can be used not only to improve the overall throughput of the disk, but also to give service guarantees to individual disk clients, a feature that is extremely important in practice e.g., when serving applications with pseudo-real-time constraints such as audio or video streaming ones.

A prototype implementation of the scheduler that will be covered in the presentation is available at <http://info.iet.unipi.it/~luigi/FreeBSD/>

bsdcan bsdcan2009 presentation freebsd geom disk schedulers luigi rzzo
http://www.bsdcan.org/2009/schedule/attachments/100_gsched.pdf 40 pages 430 Kb Slides pdf

Constantine A. Murenin - Quiet Computing with BSD

<http://www.bsdcan.org/2009/schedule/events/119.en.html>

Quiet Computing with BSD

Programming system hardware monitors for quiet computing

In this talk, we will present a detailed overview of the features and common problems of microprocessor system hardware monitors as they relate to the topic of silent computing. In a nutshell, the topic of programmable fan control will be explored.

Silent computing is an important subject as its practice reduces the amount of unnecessary stress and improves the motivation of the workforce, at home and in the office.

Attendees will gain knowledge on how to effectively programme the chips to minimise fan noise and avoid system failure or shutdown during temperature fluctuations, as well as some basic principles regarding quiet computing.

Shortly before the talk, a patch for programming the most popular chips (like those from Winbond) will be released for the OpenBSD operating system, although the talk itself will be more specific to the microprocessor system hardware monitors themselves, as opposed to the interfacing with thereof in modern operating systems like OpenBSD, NetBSD, DragonFly BSD and FreeBSD.

bsdcan bsdcan2009 presentation openbsd hardware sensors constantine murenin
http://www.bsdcan.org/2009/schedule/attachments/95_BSDCan2009.cnst-fanctl.slides.pdf 16 pages 264 Kb Slides pdf

Fernando Gont - Results of a Security Assessment of the TCP and IP protocols and Common implementation Strategies
<http://www.bsdcan.org/2009/schedule/events/129.en.html>

Results of a Security Assessment of the TCP and IP protocols and Common implementation Strategies

Fernando Gont will present the results of security assessment of the TCP and IP protocols carried out on behalf of the United Kingdom's Centre for the Protection of National Infrastructure (Centre for the Protection of National Infrastructure). His presentation will provide an overview of the aforementioned project, and will describe some of the new insights that were gained as a result of this project. Additionally, it will provide an overview of the state of affairs of the different TCP/IP implementations found in BSD operating systems with respect to the aforementioned issues.

During the last twenty years, many vulnerabilities have been identified in the TCP/IP stacks of a number of systems. The discovery of these vulnerabilities led in most cases to reports being published by a number of CSIRTs and vendors, which helped to raise awareness about the threats and the best possible mitigations known at the time the reports were published. For some reason, much of the effort of the security community on the Internet protocols did not result in official documents (RFCs) being issued by the organization in charge of the standardization of the communication protocols in use by the Internet: the Internet Engineering Task Force (IETF). This basically led to a situation in which "known" security problems have not always been addressed by all vendors. In addition, in many cases vendors have implemented quick "fixes" to the identified vulnerabilities without a careful analysis of their effectiveness and their impact on interoperability. As a result, producing a secure TCP/IP implementation nowadays is a very difficult task, in large part because of the hard task of identifying relevant documentation and differentiating between that which provides correct advisory, and that which provides misleading advisory based on inaccurate or wrong assumptions. During 2006, the United Kingdom's Centre for the Protection of National Infrastructure embarked itself in an ambitious and arduous project: performing a security assessment of the TCP and IP protocols. The project did not limit itself to an analysis of the relevant IETF specifications, but also included an analysis of common implementation strategies found in the most popular TCP and IP implementations. The result of the project was a set of documents which identifies possible threats for the TCP and IP protocols and, where possible, proposes counter-measures to mitigate the identified threats. This presentation will describe some of the new insights that were gained as a result of this project. Additionally, it will provide an overview of the state of affairs of the different TCP/IP implementations found in BSD operating systems.

bsdcan bsdcan2009 presentation bsd security assessment fernado gont http://www.bsdcan.org/2009/schedule/attachments/72_fgont-bsdcan2009-proposal.pdf 3 pages 93 Kb Proposal pdf http://www.bsdcan.org/2009/schedule/attachments/73_InternetProtocol.pdf 63 pages 660 Kb Security Assessment of the Internet Protocol pdf http://www.bsdcan.org/2009/schedule/attachments/75_tn-03-09-security-assessment-TCP.pdf 130 pages 1.4 Mb Security Assessment of the Transmission Control Protocol (TCP) pdf http://www.bsdcan.org/2009/schedule/attachments/78_fgont-bsdcan2009-tcp-ip-security-assessment.pdf 64 pages 473 Kb Slides pdf

Randi Harper - Automating FreeBSD Installations

<http://www.bsdcn.org/2009/schedule/events/126.en.html>

Automating FreeBSD Installations

PXE Booting and install.cfg Demystified

This paper will provide an explanation of the tools involved in performing an automated FreeBSD install and a live demonstration of the process.

FreeBSD's sysinstall provides a powerful and flexible mechanism for automated installs but doesn't get used very often because of a lack of documentation.

bsdcn bsdcn2009 presentation freebsd pxe sysinstall randi harper http://www.bsdcn.org/2009/schedule/attachments/79_automating_fr
14 pages 33 Kb Slides odp

Brooks Davis - Isolating Cluster Jobs for Performance and Predictability

<http://www.bsdcn.org/2009/schedule/events/125.en.html>

Isolating Cluster Jobs for Performance and Predictability

At The Aerospace Corporation, we run a large FreeBSD based computing cluster to support engineering applications. These applications come in all shapes, sizes, and qualities of implementation. To support them and our diverse userbase we have been searching for ways to isolate jobs from one another in ways that are more effective than Unix time sharing and more fine grained than allocating whole nodes to jobs.

In this talk we discuss the problem space and our efforts so far. These efforts include implementation of partial file systems virtualization and CPU isolation using CPU sets.

bsdcn bsdcn2009 presentation freebsd cluster brooks davis http://www.bsdcn.org/2009/schedule/attachments/91_job-isolation-performance-talk.pdf 27 pages 1.4 Mb Slides pdf

John Baldwin - Multiple Passes of the FreeBSD Device Tree

<http://www.bsdcn.org/2009/schedule/events/118.en.html>

Multiple Passes of the FreeBSD Device Tree

The existing device driver framework in FreeBSD works fairly well for many tasks. However, there are a few problems that are not easily solved with the current design. These problems include having "real" device drivers for low-level hardware such as clocks and interrupt controllers, proper resource discovery and management, and allowing most drivers to always probe and attach in an environment where interrupts are enabled. I propose extending the device driver framework to support multiple passes over the device tree during boot. This would allow certain classes of drivers to be attached earlier and perform boot-time setup before other drivers are probed and attached. This in turn can be used to develop solutions to the earlier list of problems.

bsdcn bsdcn2009 presentation freebsd device tree john baldwin http://www.bsdcn.org/2009/schedule/attachments/83_article.pdf
8 pages 103 Kb Paper pdf http://www.bsdcn.org/2009/schedule/attachments/85_slides.pdf 15 pages 60 Kb Slides pdf

Colin Percival - scrypt: A new key derivation function

<http://www.bsdcn.org/2009/schedule/events/147.en.html>

scrypt: A new key derivation function

Doing our best to thwart TLAs armed with ASICs

Password-based key derivation functions are used for two primary purposes: First, to hash passwords so that an attacker who gains access to a password file does not immediately possess the passwords contained therewithin; and second, to generate cryptographic keys to be used for encrypting or authenticating data.

In both cases, if passwords do not have sufficient entropy, an attacker with the relevant data can perform a brute force attack, hashing potential passwords repeatedly until the correct key is found. While commonly used key derivation functions, such as Kamp's iterated MD5, Provos and Mazieres' bcrypt, and RSA Laboratories' PBKDF1 and PBKDF2 make an attempt to increase the difficulty of brute-force attacks, they all require very little memory, making them ideally suited to attack by custom hardware.

In this talk, I will introduce the concepts of memory-hard and sequential memory-hard functions, and argue that key derivation functions should be sequential memory-hard. I will present a key derivation function which, subject to common assumptions about cryptographic hash functions, is provably sequential memory-hard, and a variation which appears to be stronger (but not provably so). Finally, I will provide some estimates of the cost of performing brute force attacks on a variety of password strengths and key derivation functions.

bsdcan bsdcan2009 presentation scrypt colin percival http://www.bsdcan.org/2009/schedule/attachments/87_scrypt.pdf
16 pages 201 Kb Paper pdf http://www.bsdcan.org/2009/schedule/attachments/86_scrypt_slides.pdf 21 pages 556 Kb Slides pdf

George Neville-Neil - Thinking about thinking in code

<http://www.bsdcan.org/2009/schedule/events/145.en.html>

Thinking about thinking in code

Proposed keynote talk

This is not a talk that's specific to any BSD but is a more general talk about how we think about coding and how our thinking changes the way we code.

I compare how we built systems to how other industries build their products and talk about what we can learn from how we work and from how others work as well.

bsdcan bsdcan2009 keynote bsd george neville-neil http://www.bsdcan.org/2009/schedule/attachments/103_BSDCan2009Keynote.pdf
137 pages 4.0 Mb Slides pdf

Stephen Borrill - Building products with NetBSD - thin-clients

<http://www.bsdcan.org/2009/schedule/events/140.en.html>

Building products with NetBSD - thin-clients

NetBSD: delivering the goods

This talk will discuss what thin-clients are, why they are useful and why NetBSD is good choice to build such a device.

This talk will provide information on some alternatives and the strengths and weaknesses of NetBSD when used in such a device.

It will discuss problems that needed to be addressed such as how to get a device with rich functionality running from a small amount of flash storage, as well as recent developments in NetBSD that have helped improve the product.

bsdcan bsdcan2009 presentation netbsd thin client stephen borrill http://www.bsdcan.org/2009/schedule/attachments/77_BuildingProducts-with-NetBSD-Stephen-Borrill.pdf 60 pages 499 Kb Slides pdf

Cat Allman and Leslie Hawthorn - Getting Started in Free and Open Source

<http://www.bsdcan.org/2009/schedule/events/149.en.html>

Getting Started in Free and Open Source

Interested in getting involved? But don't really know where or how to start?

The talk is called "Getting Started in Free and Open Source". It's a talk for beginners who are interested to getting involved but don't really know where or how to start.

We cover the basics of: -why you might want to get involved -what you can get out of participating -more than coding is needed -how to chose a project -how to get started -etiquette of lists and other communication -dos and don't of joining a community

bsdcan bsdcan2009 presentation getting started cat allman leslie hawthorn
http://www.bsdcan.org/2009/schedule/attachments/99_BSDCan_allman_lhawthorn.odp 25 pages 893 Kb Slides
odf

Warner Losh - Tracking FreeBSD in a commercial Environment

<http://www.bsdcan.org/2009/schedule/events/143.en.html>

Tracking FreeBSD in a commercial Environment

How to stay current while staying sane

The FreeBSD project publishes two lines of source code: current and stable. All changes must first be committed to current and then are merged into stable. Commercial organizations wishing to use FreeBSD in their products must be aware of this policy. Four different strategies have developed for tracking FreeBSD over time. A company can choose to run only unmodified release versions of FreeBSD. A company may choose to import FreeBSD's sources once and then never merge newer versions. A company can choose to import each new stable branch as it is created, adding its own changes to that branch, as well as integrating new versions from FreeBSD from time to time. A company can track FreeBSD's current branch, adding to it their changes as well as newer FreeBSD changes. Which method a company chooses depends on the needs of the company. These methods are explored in detail, and their advantages and disadvantages are discussed. Tracking FreeBSD's ports and packages is not discussed.

Companies building products based upon FreeBSD have many choices in how to use the projects sources and binaries. The choices range from using unmodified binaries from FreeBSD's releases, to tracking modify FreeBSD heavily and tracking FreeBSD's evolution in a merged tree. Some companies may only need to maintain a stable version of FreeBSD with more bug fixes or customizations than the FreeBSD project wishes to place in that branch. Some companies also wish to contribute some subset of their changes back to the FreeBSD project.

FreeBSD provides an excellent base technology with which to base products. It is a proven leader in performance, reliability and scalability. The technology also offers a very business friendly license that allows companies to pick and choose which changes they wish to contribute to the community rather than forcing all changes to be contributed back, or attaching other undesirable license conditions to the code.

However, the FreeBSD project does not focus on integration of its technology into customized commercial products. Instead, the project focuses on producing a good, reliable, fast and scalable operating system and associated packages. The project maintains two lines of development. A current branch, where the main development of the project takes place, and a stable branch which is managed for stability and reliability. While the project maintains documentation on the system, including its development model, relatively little guidance has been given to companies in how to integrate FreeBSD into their products with a minimum of trouble.

Developing a sensible strategy to deal with both these portions of FreeBSD requires careful planning and analysis. FreeBSD's lack of guidelines to companies leaves it up to them to develop a strategy. FreeBSD's development model differs from some of the other Free and Open Source projects. People familiar with those systems often discover that methods that were well suited to them may not work as well with FreeBSD's development model. These two issues cause many companies to make poor decisions without understanding the problems that lie in their future.

Very little formal guidance exists for companies wishing to integrate FreeBSD into their products. Some email threads can be located via a Google search that could help companies, but many of them are full of contradictory information, and it is very disorganized. While the information about the FreeBSD development process is in the FreeBSD handbook, the implications of that process for companies integrating FreeBSD into their products are not discussed.

bsdcan bsdcan2009 presentation freebsd commercial environment waner losh
http://www.bsdcan.org/2009/schedule/attachments/82_bsdcan2009-paper.pdf 10 pages 104 Kb Slides pdf
http://www.bsdcan.org/2009/schedule/attachments/81_bsdcan2009-slides.pdf 45 pages 624 Kb Paper pdf

Kris Moore - PC-BSD - Making FreeBSD on the desktop a reality

<http://www.bsdcan.org/2009/schedule/events/133.en.html>

PC-BSD - Making FreeBSD on the desktop a reality

FreeBSD on the Desktop

While FreeBSD is a all-around great operating system, it is greatly lagging behind in desktop appeal. Why is this? In this talk, we will take a look at some of the desktop drawbacks of FreeBSD, and how are attempting to fix them through PC-BSD.

FreeBSD has a reputation for its rock-solid reliability, and top-notch performance in the server world, but is noticeably absent when it comes to the vast market of desktop computing. Why is this? FreeBSD offers many, if not almost all of the same open-source packages and software that can be found in the more popular Linux desktop distributions, yet even with the speed and reliability FreeBSD offers, a relative few number of users are deploying it on their desktops.

In this presentation we will take a look at some of the reasons why FreeBSD has not been as widely adopted in the desktop market as it has on the server side. Several of the desktop weaknesses of FreeBSD will be shown, along with how we are trying to fix these short-comings through a desktop-centric version of FreeBSD, known as PC-BSD. We will also take a look at the package management system employed by all open-source operating systems alike, and some of the pitfalls it brings, which may hinder widespread desktop adoption.

bsdcan bsdcan2009 presentation pc-bsd freebsd kris moore http://www.bsdcan.org/2009/schedule/attachments/76_pcbsd-bsdcan09.pdf 35 pages 512 Kb Slides pdf http://www.bsdcan.org/2009/schedule/attachments/74_bsdcan09-PCBSD.pdf 9 pages 351 Kb Paper pdf

Sean Bruno - Implementation of TARGET_MODE applications

<http://www.bsdcan.org/2009/schedule/events/127.en.html>

Implementation of TARGET_MODE applications

How we used TARGET_MODE in the kernel to create an interesting product

This presentation will cover a real world implementation of the TARGET_MODE infrastructure in the kernel (stable/6). Topics to include: drivers used (isp, aic7xxx, firewire). scsi_target userland code vs kernel drivers missing drivers (4/8G isp support, iSCSI target)

Target Mode describes a feature within certain drivers that allows a FreeBSD system to emulate a Target in the SCSI sense of the word. By recompiling your kernel with this feature enabled, it permits one to turn a FreeBSD system into an external hard disk. This feature of the FreeBSD kernel provides many interesting implementations and is highly desirable to many organizations whom run FreeBSD as their platform.

I have been tasked with the maintenance of a proprietary target driver that interfaces with the FreeBSD kernel to do offsite data mirroring at the block level. This talk will discuss the implementation of that kernel mode driver and the process my employer went through to implement a robust and flexible appliance.

Since I took over the implementation, we have implemented U160 SCSI(via aic7xxx), 2G Fibre Channel(via isp) and Firewire 400 (via sbp_targ). Each driver has it's own subtleties and requirements. I personally enhanced the existing Firewire target driver and was able to get some interesting results.

I hope to demonstrate a functional Firewire 400/800 target and show how useful this application can be for the embedded space. Also, I wish to demonstrate the need for iSCSI. USB and 4/8G Fibre Channel target implementations that use the TARGET_MODE infrastructure that is currently in place to allow others to expand their various interface types.

The presentation should consist of a high level overview, followed by detailed implementation instructions with regards to the Firewire implementation and finish up with a hands-on demonstration with a FreeBSD PC flipped into TARGET_MODE and a Mac.

bsdcan bsdcan2009 presentation freebsd firewire sean bruno http://www.bsdcan.org/2009/schedule/attachments/92_BSDCan_TMODE_I
22 pages 72 Kb Slides pdf

George Neville-Neil - Understanding and Tuning SCHED_ULE

<http://www.bsdcan.org/2009/schedule/events/117.en.html>

Understanding and Tuning SCHED_ULE

With the advent of widespread SMP and multicore CPU architectures it was necessary to implement a new scheduler in the FreeBSD operating system. The SCHEDULE scheduler was added for the 5 series of FreeBSD releases and has now matured to the point where it is the default scheduler in the 7.1 release. While scheduling processes was a difficult enough task in the uniprocessor world, moving to multiple processors, and multiple cores, has significantly increased the number of problems that await engineers who wish to squeeze every last ounce of performance out of their system. This talk will cover the basic design of SCHEDULE and focus a great deal of attention on how to tune the scheduler for different workloads, using the sysctl interfaces that have been provided for that purpose.

Understanding and tuning a scheduler used to be done only by operating systems designers and perhaps a small minority of engineers focusing on esoteric high performance systems. With the advent of widespread multi-processor and multi-core architectures it has become necessary for more users and administrators to decide how to tune their systems for the best performance. The SCHEDULE scheduler in FreeBSD provides a set of sysctl interfaces for tuning the scheduler at run time, but in order to use these interfaces effectively the scheduling process must first be understood. This presentation will give an overview of how SCHEDULE works and then will show several examples of tuning the system with the interfaces provided.

The goal of modifying the scheduler's parameters is to change the overall performance of programs on the system. One of the first problems presented to the person who wants to tune the scheduler is how to measure the effects of their changes. Simply tweaking the parameters and hoping that that will help is not going to lead to good results. In our recent experiments we have used the top(1) program to measure our results.

bsdcan bsdcan2009 presentation freebsd sched_ule george neville-neil http://www.bsdcan.org/2009/schedule/attachments/101_sched_tun
29 pages 228 Kb Slides pdf

Lawrence Stewart - Improving the FreeBSD TCP Implementation

<http://www.bsdcan.org/2009/schedule/events/121.en.html>

Improving the FreeBSD TCP Implementation.

An update on all things TCP in FreeBSD and how they affect you.

My involvement in improving the FreeBSD TCP stack has continued this past year, with much of the work targeted at FreeBSD 8. This talk will cover what these changes entail, why they are of interest to the FreeBSD community and how they help to improve our TCP implementation.

It has been a busy year since attending my inaugural BSDCan in 2008, where I talked about some of my work with TCP in FreeBSD.

I have continued the work on TCP analysis/debugging tools and integrating modular congestion control into FreeBSD as part of the NewTCP research project. I will provide a progress update on this work.

Additionally, a grant win from the FreeBSD Foundation to undertake a project titled “Improving the FreeBSD TCP Implementation” at Swinburne University’s Centre for Advanced Internet Architectures has been progressing well. The project focuses on bringing TCP Appropriate Byte Counting (RFC 3465), reassembly queue auto-tuning and integration of low-level analysis/debugging tools to the base system, all of which I will also discuss.

bsdcan bsdcan2009 presentation freebsd tcp lawrence stewart http://www.bsdcan.org/2009/schedule/attachments/89_bsdcan200905.pdf
38 pages 2.1 Mb Slides pdf

Joerg Sonnenberger - Journaling FFS with WAPBL

<http://www.bsdcan.org/2009/schedule/events/138.en.html>

Journaling FFS with WAPBL

NetBSD 5 is the first NetBSD release with a journaling filesystem. This lecture introduces the structure of the Fast File System, the modifications for WAPBL and specific constraints of the implementation.

The Fast File System (FFS) has been used in the BSD land for more than two decades. The original implementation offered two operational modes:

- safe and slow (sync)
- unsafe and fast (async) One decade ago, Kirk McKusick introduced the soft dependency mechanism to offset the performance impact without risk of mortal peril on the first crash. With the advent of Terabyte hard disks, the need for a file system check (fsck) after a crash becomes finally unacceptable. Even a background fsck like supported on FreeBSD consumes lots of CPU time and IO bandwidth.

Based on a donation from Wasabi Systems, Write Ahead Physical Block Logging (WAPBL) provides journaling for FFS with similar or better performance than soft dependencies during normal operation. Recovery time after crashes depends on the amount of outstanding IO operations and normally takes a few seconds.

This lecture gives a short overview of FFS and the consistency constraints for meta data updates. It introduces the WAPBL changes, both in terms of the on-disk format and the implementation in NetBSD. Finally the implementation is compared to the design of comparable file systems and specific issues of and plans for the current implementation are discussed.

bsdcan bsdcan2009 presentation netbsd wapbl ffs joerg sonnenberger <http://www.netbsd.org/gallery/presentations/joerg/bsdcan2009/wapbl>
24 pages 10 Kb Slides html

Ivan Voras - Remote and mass management of systems with finstall

<http://www.bsdcan.org/2009/schedule/events/115.en.html>

Remote and mass management of systems with finstall

Automated management on a largish scale

An important part of the “finstall” project, created as a graphical installer for FreeBSD, is a configuration server that can be used to remotely administer and configure arbitrary systems. It allows for remote scripting of administration tasks and is flexible enough to support complete reconfiguration of running systems.

The finstall project has two major parts - the front-end and the back-end. The front-end is just a GUI allowing the users to install the system in a convenient way. The back-end is a network-enabled XML-RPC server that is used by the front-end to perform its tasks. It can be used as a stand-alone configuration daemon. This talk will describe a way to make use of this property of finstall to remotely manage large groups of systems.

bsdcan bsdcan2009 presentation finstall management freebsd ivan voras http://www.bsdcan.org/2009/schedule/attachments/88_IvanVoras.pdf 24 pages 377 Kb Slides pdf

Mike Silbersack - Detecting TCP regressions with tcpdiff

<http://www.bsdcan.org/2009/schedule/events/120.en.html>

Detecting TCP regressions with tcpdiff

Determining if a TCP stack is working correctly is hard. The tcpdiff project aims for a simpler goal: To automatically detect differences in TCP behavior between different versions of an operating system and display those differences in an easy to understand format. The value judgement of whether a certain change between version X and Y of a TCP stack is good or bad will be left to human eyes.

Determining if a TCP stack is working correctly is hard. The tcpdiff project aims for a simpler goal: To automatically detect differences in TCP behavior between different versions of an operating system and display those differences in an easy to understand format. The value judgement of whether a certain change between version X and Y of a TCP stack is good or bad will be left to human eyes.

The initial version of tcpdiff presented at NYCBSDCon 2008 demonstrated that it could be used to detect at least two major TCP bugs that were introduced into FreeBSD in the past few years. The work from that presentation can be viewed at <http://www.silby.com/nycbsdcon08/>.

For BSDCan 2009, I hope to fix a number of bugs in tcpdiff, make it easier to use, set up nightly tests of FreeBSD, and improve it so that additional known bugs can be detected. Additionally, I plan to run it on OSes other than FreeBSD.

bsdcan bsdcan2009 presentation tcpdiff freebsd mike silbersack http://www.bsdcan.org/2009/schedule/attachments/90_BSDCan-tcpdiff.pdf 33 pages 89 Kb Slides pdf

Philip Paeps - Crypto Acceleration on FreeBSD

<http://www.bsdcan.org/2009/schedule/events/135.en.html>

Crypto Acceleration on FreeBSD

As more and more services on the internet become cryptographically secured, the load of cryptography on systems becomes heavier and heavier. Crypto acceleration hardware is available in different forms for different workloads. Embedded communications processors from VIA and AMD have limited acceleration facilities in silicon and various manufacturers build hardware for accelerating secure web traffic and IPSEC VPN tunnels.

This talk gives an overview of FreeBSD's crypto framework in the kernel and how it can be used together with OpenSSL to leverage acceleration hardware. Some numbers will be presented to demonstrate how acceleration can improve performance - and how it can curiously bring a system to a grinding halt.

Philip originally started playing with crypto acceleration when he saw the "crypto block" in one of his Soekris boards. As usual, addiction was instant and by the grace of the "you touch it, you own it" principle, he has been fiddling the crypto framework more than is good for him.

bsdcan bsdcan2009 presentation crypto acceleration freebsd philip paeps http://www.bsdcan.org/2009/schedule/attachments/97_crypto_acceleration.pdf 28 pages 361 Kb Slides pdf

Sean Bruno - Firewire BoF Plugfest

<http://www.bsdcan.org/2009/schedule/events/144.en.html>

Firewire BoF Plugfest

Debugging and testing of Firewire products with FreeBSD

Come one come all to a Firewire plugfest. Let's debug and test together and see if we can't knock out some features and bugs.

A hands-on testing and debugging session of the Firewire stack in FreeBSD.

Everyone who wishes to attend should bring their Firewire devices, ext Drives and Cameras, and their Laptops. I will be debugging and capturing data points to enhance and improve features in the Firewire stack.

We should be able to knock out quite a bunch of bugs if folks can bring their various Firewire devices along with their various PCs.

Even if your Firewire device works perfectly, bring it by so it can be documented as supported by the Firewire team!

bsdcan bsdcan2009 presentation firewire plugfest sean bruno http://www.bsdcan.org/2009/schedule/attachments/93_FireWireBoF.odp
1 page 37 Kb Slides odp

Peter Hansteen - Building the Network You Need with PF, the OpenBSD packet filter

<http://www.bsdcan.org/2009/schedule/track/Tutorial/114.en.html>

Building the Network You Need with PF, the OpenBSD packet filter.

Building the network you need is the central theme for any network admin. This tutorial is for aspiring or seasoned network professionals with at least a basic knowledge of networking in general and TCP/IP particular. The session aims at teaching tools and techniques to make sure you build your network to work the way it's supposed to, keeping you in charge. Central to the toolbox is the OpenBSD PF packet filter, supplemented with tools that interact with it. Whether you are a greybeard looking for ways to optimize your setups or a greenhorn just starting out, this session will give you valuable insight into the inner life of your network and provide pointers to how to use that knowledge to build the network you need. The session will also offer some fresh information on changes introduced in OpenBSD 4.5, the most recent version of PF and OpenBSD. The tutorial is loosely based on Hansteen's recent book, /The Book of PF/ (No Starch Press), with updates and adaptations based on developments since the book's publication date.

bsdcan bsdcan2009 tutorial pf openbsd peter hansteen http://www.bsdcan.org/2009/schedule/attachments/98_BSDCan2009_hansteen_pf
68 pages 2.5 Mb Slides html

George Neville-Neil - Networking from the Bottom Up: Device Drivers

<http://www.bsdcan.org/2009/schedule/track/Tutorial/146.en.html>

Networking from the Bottom Up: Device Drivers.

In this tutorial I will describe how to write and maintain network drivers in FreeBSD and use the example of the Intel Gigabit Ethernet driver (igb) throughout the course.

Students will learn the basic data structures and APIs necessary to implement a network driver in FreeBSD. The tutorial is general enough that it can be applied to other BSDs, and likely to other embedded and UNIX like systems while being specific enough that given a device and a manual the student should be able to develop a working driver on their own. This is the first of a series of lectures on network that I am developing over the next year or so.

bsdcan bsdcan2009 tutorial device drivers george neville-neil http://www.bsdcan.org/2009/schedule/attachments/102_devices.pdf
68 pages 480 Kb PDF file pdf

Daniel Braniss

<http://www.bsdcan.org/2008/schedule/events/102.en.html>

iSCSI

not an Apple appliance.

iSCSI is not an Apple appliance.

The i in iSCSI stands for internet, some say for insecure, personally I like to think interesting. I'll try to share the road followed from RFC-3720 to the actual working driver, the challenges, the frustrations.

bsdcan bsdcan2008 presentation iscsi daniel braniss http://www.bsdcan.org/2008/schedule/attachments/65_bsdcan.pdf
30 pages 1.4 Mb PDF file pdf

Scott Ullrich, Chris Buechler - pfSense Tutorial

<http://www.bsdcan.org/2008/schedule/events/80.en.html>

pfSense Tutorial

From Zero to Hero with pfSense

pfSense is a free, open source customized distribution of FreeBSD tailored for use as a firewall and router. In addition to being a powerful, flexible firewalling and routing platform, it includes a long list of related features and a package system allowing further expandability without adding bloat and potential security vulnerabilities to the base distribution. pfSense is a popular project with more than 1 million downloads since its inception, and proven in countless installations ranging from small home networks protecting a PC and an Xbox to large corporations, universities and other organizations protecting thousands of network devices.

This tutorial is being presented by the founders of the pfSense project, Chris Buechler and Scott Ullrich.

The session will start with an introduction to the project, hardware sizing and selection, installation, firewalling concepts and basic configuration, and continue to cover all the most popular features of the system. Common usage scenarios, deployment considerations, step by step configuration guidance, and best practices will be covered for each feature. Most configurations will be demonstrated in a live lab environment.

Attendees are assumed to have basic knowledge of TCP/IP and firewalling concepts, however no in-depth knowledge in these areas or prior knowledge of pfSense or FreeBSD is necessary.

bsdcan bsdcan2008 tutorial freebsd pfsense scott ullrich chris buechler http://www.bsdcan.org/2008/schedule/attachments/66_pfSenseTutorial.pdf 91 pages 4.1 Kb PDF file pdf

Bjoern A. Zeeb - BSDCan08 devsummit summary

<http://people.freebsd.org/~bz/200805DevSummit/> 200805DevSummit - BSDCan 2008 FreeBSD Developer summit summary bsdcan bsdcan2008 devsummit devsummit2008 freebsd writeup bjoern a zeeb

Rafal Jaworowski - FreeBSD Embedded Report

<http://wiki.freebsd.org/200805DevSummit> FreeBSD Embedded Report bsdcan
bsdcan2008 devsummit devsummit2008 freebsd embedded rafal jaworowski
http://wiki.freebsd.org/200805DevSummit?action=AttachFile&do=get&target=devsummit-200805-embedded_summary.pdf 6 pages 58 Kb PDF file pdf

Robert Watson - TCP SMP Scalability

<http://wiki.freebsd.org/200805DevSummit> TCP SMP Scalability bsdcan bsdcan2008 devsummit devsummit2008
freebsd smp robert watson <http://wiki.freebsd.org/200805DevSummit?action=AttachFile&do=get&target=20080515-stack-parallelism.pdf> 8 pages 70 Kb PDF file pdf

Erwin Lansing - What's happening in the world of ports and portmgr

<http://wiki.freebsd.org/200805DevSummit> What's happening in the world of ports and
portmgr bsdcan bsdcan2008 devsummit devsummit2008 freebsd portmgr erwin lansing
<http://wiki.freebsd.org/200805DevSummit?action=AttachFile&do=get&target=portmgr-BSDCan2008.pdf> 14 pages
146 Kb PDF file pdf

Kern Sibbald - Bacula

<http://www.bsdcn.org/2008/schedule/events/96.en.html>

Bacula

The Open Source Enterprise Backup Solution

The Bacula project started in January 2000 with several goals, one of which was the ability to backup any client from a Palm to a mainframe computer. Bacula is available under a GPL license.

Bacula uses several distinct components, each communicating via TCP/IP, to achieve a very scalable and robust solution to backups.

Kern is one of the original project founders and still one of the most productive Bacula developers.

bsdcan bsdcan2008 slides bacula kern sibbald http://www.bsdcan.org/2008/schedule/attachments/55_Bacula-BSDCan-talk-17May08.pdf 30 pages 505 Kb PDF file pdf

Warner Losh - FreeBSD/mips

<http://www.bsdcan.org/2008/schedule/events/86.en.html>

FreeBSD/mips

Embedding FreeBSD

FreeBSD now runs on the MIPS platform. FreeBSD/mips supports MIPS-32 and MIPS-64 targets, including SMP for multicore support.

FreeBSD/mips is targeted at the embedded MIPS marketplace. FreeBSD has run on the MIPS platform for many years. Juniper ported FreeBSD to the Mips platform in the late 1990's. However, concern about intellectual property issues kept Juniper from contributing the port back to FreeBSD until recently. The contributed port was a 64-bit mips port.

In the mean time, many efforts were made to bring FreeBSD to the mips platform. The first substantial effort to bring FreeBSD to the Mips platform was done by Juli Mallet. This effort made it to single user, but never further than that. This effort was abandoned due to a change in Juli's life. The port languished.

Two years ago at BSDcan, as my involvement with FreeBSD/arm was growing, I tried to rally the troops into doing a FreeBSD/mips port. My efforts resulted in what has been commonly called the "mips2" effort. The name comes from the choice of `//depot/projects/mips2` to host the work in perforce. A number of people worked on the earliest versions of the port, but it too languished and seemed destined to suffer the same fate as earlier efforts. Then, two individuals stood up and started working on the port. Wojciech A. Koszek and Oleksandr Tymoshenko pulled in code from the prior efforts. Through their efforts of stabilizing this code, the port to the single user stage and ported it to three different platforms. Others ported it to a few more. Snapshots of this work were released from time to time.

Cavium Networks picked up one of these snapshots and ported it to their multicore mips64 network processor. Cavium has kindly donated much of their work to the community.

In December, I started at Cisco systems. My first job was to merge all the divergent variants of FreeBSD/mips and get it into shape to push into the tree. With luck, this should be in the tree before I give my talk.

In parallel to this, other advances in the embedded support for FreeBSD have been happening as well. I'll talk about new device drivers, new subsystems, and new build tools that help to support the embedded developer.

bsdcan bsdcan2008 slides freebsd mips embedded warner losh http://www.bsdcan.org/2008/schedule/attachments/63_freebsd-mips-bsdcan-2008.pdf 19 pages 1.3 Mb PDF file pdf

Kris Moore - Building self-contained PBIs from Ports (Automagically)

<http://www.bsdcan.org/2008/schedule/events/81.en.html>

Building self-contained PBIs from Ports (Automagically)

Creating a self-contained application from the ports tree

PC-BSD provides a user-friendly desktop experience, for experts and casual users alike. PC-BSD is 100% FreeBSD under the hood, while providing desktop essentials, such as a graphical installation system, point-n-click package-management using the PBI system, and easy to use system management tools; All integrated into an easy to use K Desktop Environment (KDE).

The PBI (Push Button Installer) format is the cornerstone of the PC-BSD desktop, which allows users to install applications in a self-contained format, free from dependency problems, and compile issues that stop most casual users from desktop adoption. The PBI format also provides power and flexibility in user interaction, and scripting support, which allows applications to be fine-tuned to the best possible user experience.

This talk would go over in some detail our new PBI building system, which converts a FreeBSD port, such as FireFox, into a standalone self-contained PBI installer for PC-BSD desktops.

The presentation will be divided into two main sections:

The Push Button Installer (PBI) Format

- The basics of the PBI format
- The PBI format construction
- Add & Remove scripting support within PBI

Building PBIs from Ports “Auto-magically”

- The PBI build server & standalone software
- Module creation & configuration
- Converting messy ports into PBIs

bsdcan bsdcan2008 slides pc-bsd ports pbi kris moore http://www.bsdcan.org/2008/schedule/attachments/57_PBIPresentation
26 pages 120 Kb PDF file pdf

John Pertalion - An Open Source Enterprise VPN Solution with OpenVPN and OpenBSD

<http://www.bsdcan.org/2008/schedule/events/71.en.html>

An Open Source Enterprise VPN Solution with OpenVPN and OpenBSD

Solving the problem

At Appalachian State University, we utilize an open source VPN to allow faculty, staff and vendors secure access to Appalachian State University's internal network from any location that has an Internet connection. To implement our virtual private network project, we needed a secure VPN that is flexible enough to work with our existing network registration and LDAP authentication systems, has simple client installation, is redundant, allows multiple VPN server instances for special site-to-site tunnels and unique configurations, and can run on multiple platforms. Using OpenVPN running on OpenBSD, we met those requirements and added a distributed administration system that allows select users to allow VPN access to specific computers for external users and vendors without requiring intervention from our network or security personnel. Our presentation will start with a quick overview of OpenVPN and OpenBSD and then detail the specifics of our VPN implementation.

Dissatisfied with IPSec for road warrior VPN usage we went looking for a better solution. We had hoped that we could find a solution that would run on multiple platforms, was flexible and worked well. We found OpenVPN and have been pleased. Initially we ran it on RHEL. We migrated to OpenBSD for pf functionality and general security concerns. ...and because we like OpenBSD.

Our presentation will focus on the specifics of our VPN implementation. We will quickly cover the basics of OpenVPN and the most used features of OpenBSD. Moving along we will cover multiple authentication methods, redundancy, running multiple instances, integration with our netreg system, how pf has extended functionality, embedding in appliances, and client configuration. The system has proven helpful with providing vendor access where needed and we'll cover this aspect as well. Time permitting we will cover current enhancement efforts and future plans.

OpenVPN has been called the "Swiss army knife" of VPN solutions. We hope our presentation leaves participants with that feeling.

bsdcan bsdcan2008 slides openbsd openvpn john pertalion http://www.bsdcan.org/2008/schedule/attachments/59_OVPN-BSDCan2008.pdf 26 pages 127 Kb PDF file pdf

Ivan Voras - "finstall" - the new FreeBSD installer

<http://www.bsdcan.org/2008/schedule/events/69.en.html>

“fininstall” - the new FreeBSD installer

A graphical installer for FreeBSD

The “fininstall” project, sponsored by Google as a Summer of Code 2007 project, is an attempt to create a user-friendly graphical installer for FreeBSD, with enough strong technical features to appeal to the more professional users. A long term goal for it is to be a replacement for sysinstall, and as such should support almost all of the features present in sysinstall, as well as add support for new FreeBSD features such as GEOM, ZFS, etc. This talk will describe the architecture of “fininstall” and focus on its lesser known features such as remote installation.

“fininstall” is funded by Google SoC as a possible long-term replacement for sysinstall, as a “LiveCD” with the whole FreeBSD base system on the CD, with X11 and XFCE4 GUI. In the talk I intend to describe what I did so far, and what are the future plans for it. This includes the installer GUI, the backend (which has the potential to become a generic FreeBSD configuration backend) and the assorted tools developed for fininstall (“LiveCD” creation scripts). More information on fininstall can be found here: <http://wiki.freebsd.org/fininstall>.

bsdcan bsdcan2008 slides freebsd installer ivan voras http://www.bsdcan.org/2008/schedule/attachments/56_bsdcantalk.pdf
39 pages 1.1 Mb PDF file pdf

Poul-Henning Kamp - Measured (almost) does Air Traffic Control

<http://www.bsdcan.org/2008/schedule/events/68.en.html>

Measured (almost) does Air Traffic Control

Monitoring weird hardware reliably

The new Danish Air Traffic Control system, CASIMO, prompted the development on a modular and general software platform for data collection, control and monitoring of “weird hardware” of all sorts.

The talk will present the “measured” daemon, and detail some of the uses it has been put to, as an, admittedly peripheral, component of the ATC system.

Many “SCADA” systems suffer from lack of usable interfaces for external access to the data. Measured takes the opposite point of view and makes real-time situation available, and accepts control instructions as ASCII text stream over TCP connections. Several examples of how this can be used will be demonstrated.

Measured will run on any FreeBSD system, but has not been ported to other UNIX variants yet, and it is perfect for that “intelligent house” project of yours.

I believe I gave a WIP presentation of this about two years ago.

bsdcan bsdcan2008 slides air traffic control scada poulsen-henning kamp http://www.bsdcan.org/2008/schedule/attachments/64_BSDCan2008_AirTrafficControl.pdf 46 pages 7.7 Mb PDF file pdf

Chris Lattner - BSD licensed C++ compiler

<http://www.bsdcan.org/2008/schedule/events/99.en.html>

BSD licensed C++ compiler

LLVM is a suite of carefully designed open source libraries that implement compiler components (like language front-ends, code generators, aggressive optimizers, Just-In-Time compiler support, debug support, link-time optimization, etc.). The goal of the LLVM project is to build these components in a way that allows them to be combined together to create familiar tools (like a C compiler), interesting new tools (like an OpenGL JIT compiler), and many other things we haven't thought of yet. Because LLVM is under continuous development, clients of these components naturally benefit from improvements in the libraries.

This talk gives an overview of LLVM's design and approach to compiler construction, and gives several example applications. It describes applications of LLVM technology to llvm-gcc (a C/C++/Objective C compiler based on the GNU GCC front-end), the OpenGL stack in Mac OS/X Leopard, and Clang. Among other things, the Clang+LLVM Compiler provides a fully BSD-Licensed C and Objective-C compiler (with C++ in development) which compiles code several times faster than GCC, produces code that is faster than GCC in many cases, produces better warnings and error messages, and supports many other applications (e.g. static analysis and refactoring).

bsdcan bsdcan2008 slides bsd llvm chris lattner http://www.bsdcan.org/2008/schedule/attachments/53_BSDCan2008ChrisLattnerBSDC
33 pages 5.8 Mb PDF file pdf

Robert Watson - BSDCan 2008 - Closing

<http://www.bsdcan.org/2008/schedule/events/97.en.html>

Closing

Beer, prizes, secrets, Works In Progress

The traditional closing...

with some new and interesting twists. Sleep in if you must, but don't miss this session.

bsdcan bsdcan2008 slides robert watson http://www.bsdcan.org/2008/schedule/attachments/47_BSDCann2008Closing.pdf
55 pages 428 Kb PDF file pdf

Leslie Hawthorn - Google SoC

<http://www.bsdcan.org/2008/schedule/events/95.en.html>

Google SoC

Summer of Code

In this talk, I will briefly discuss some general ways Google's Open Source Team contributes to the wider community. The rest of the talk will explore some highlights of the Google Summer of Code program, our initiative to get university students involved in Open Source development.

I will cover the program's inception, lessons learned over time and tips for success in the program for both mentors and students. In particular, the talk will detail some experiences of the *BSD mentoring organizations involved in the program as a case study in successfully managing the program from the Open Source project's perspective. Any Google Summer of Code participants in the audience are welcome and encouraged to chime in with their own insights.

bsdcan bsdcan2008 slides google summer of code leslie hawthorn http://www.bsdcan.org/2008/schedule/attachments/52_LeslieHawthorn.pdf
44 pages 2.2 Mb PDF file pdf

Pawel Jakub Dawidek - A closer look at the ZFS file system

<http://www.bsdcan.org/2008/schedule/events/93.en.html>

A closer look at the ZFS file system

simple administration, transactional semantics, end-to-end data integrity

SUN's ZFS file system became part of FreeBSD on 6th April 2007. ZFS is a new kind of file system that provides simple administration, transactional semantics, end-to-end data integrity, and immense scalability. ZFS is not an incremental improvement to existing technology; it is a fundamentally new approach to data management. We've blown away 20 years of obsolete assumptions, eliminated complexity at the source, and created a storage system that's actually a pleasure to use.

ZFS presents a pooled storage model that completely eliminates the concept of volumes and the associated problems of partitions, provisioning, wasted bandwidth and stranded storage. Thousands of file systems can draw from a common storage pool, each one consuming only as much space as it actually needs. The combined I/O bandwidth of all devices in the pool is available to all filesystems at all times.

All operations are copy-on-write transactions, so the on-disk state is always valid. There is no need to fsck(1M) a ZFS file system, ever. Every block is checksummed to prevent silent data corruption, and the data is self-healing in replicated (mirrored or RAID) configurations. If one copy is damaged, ZFS detects it and uses another copy to repair it.

bsdcan bsdcan2008 slides zfs freebsd pawel jakub dawidek http://www.bsdcan.org/2008/schedule/attachments/58_BSDCan2008-ZFSInternals.pdf 33 pages 150 Kb PDF file pdf

Rafal Jaworowski - Interfacing embedded FreeBSD with U-Boot

<http://www.bsdcan.org/2008/schedule/events/74.en.html>

Interfacing embedded FreeBSD with U-Boot

Working with the de facto standard for an initial level boot loader

In the embedded world U-Boot is a de facto standard for an initial level boot loader (firmware). It runs on a great number of platforms and architectures, and is open source.

This talk covers the development work on integrating FreeBSD with U-Boot-based systems. Starting with an overview of differences between booting an all-purpose desktop computer vs. embedded system, FreeBSD booting concepts are explained along with requirements for the underlying firmware.

Historical attempts to interface FreeBSD with this firmware are mentioned and explanation given on why they failed or proved incomplete. Finally, the recently developed approach to integrate FreeBSD and U-Boot is presented, with implementation details and particular attention on how it's been made architecture and platform independent, and how loader(8) has been bound to it.

bsdcan bsdcan2008 slides embedded freebsd u-boot rafal jaworowski http://www.bsdcan.org/2008/schedule/attachments/49_2008_uboot
26 pages 300 Kb PDF file pdf

John Baldwin - Introduction to Debugging the FreeBSD Kernel

<http://www.bsdcan.org/2008/schedule/events/70.en.html>

Introduction to Debugging the FreeBSD Kernel

Just like every other piece of software, the FreeBSD kernel has bugs. Debugging a kernel is a bit different from debugging a userland program as there is nothing underneath the kernel to provide debugging facilities such as `ptrace()` or `procs`. This paper will give a brief overview of some of the tools available for investigating bugs in the FreeBSD kernel. It will cover the in-kernel debugger DDB and the external debugger kgdb which is used to perform post-mortem analysis on kernel crash dumps.

25.1 Introduction to Debugging the FreeBSD Kernel

- Basic crash messages, what a crash looks like

- typical `panic()` invocation
- page fault example

- “live” debugging with DDB

- stack traces
- `ps`
- deadlock examples
- `show lockchain`
- `show sleepchain`
- Adding new DDB commands

- KGDB

- inspecting processes and threads
- working with kernel modules
- using scripts to extend

- examining crashdumps using utilities

- `ps`, `netstat`, etc.

- debugging strategies

- kernel crashes
- system hangs

bsdcan bsdcan2008 slides paper debugging freebsd john baldwin http://www.bsdcan.org/2008/schedule/attachments/46_slides.pdf
26 pages 113 Kb slides, PDF file pdf http://www.bsdcan.org/2008/schedule/attachments/45_article.pdf 15 pages 121
Kb paper, PDF file pdf

John Birrell - DTrace for FreeBSD

<http://www.bsdcan.org/2008/schedule/events/66.en.html>

DTrace for FreeBSD

What on earth is that system doing?!

DTrace is a comprehensive dynamic tracing facility originally developed for Solaris that can be used by administrators and developers on live production systems to examine the behavior of both user programs and of the operating system itself. DTrace enables users to explore their system to understand how it works, track down performance problems across many layers of software, or locate the cause of aberrant behavior. DTrace lets users create their own custom programs to dynamically instrument the system and provide immediate, concise answers to arbitrary questions you can formulate using the DTrace D programming language.

This talk discusses the port of the DTrace facility to FreeBSD and demonstrates examples on a live FreeBSD system.

- Introduction to the D language - probes, predicates and actions.
- dtrace(8) and libdtrace - the userland side of the DTrace story.
- The DTrace kernel module, it's ioctl interface to userland and the provider infrastructure in the kernel.
- DTrace kernel hooks and the problem of code licensed under Sun's CDDL.
- What does a DTrace probe actually do?
- DTrace safety and how it is implemented.
- Build system changes to add CTF (Compact C Type Format) data to objects, shared libraries and executables.
- The DTrace test suite.
- A brief list of things to do to port the DTrace facility to other BSD-derived operating systems.

bsdcan bsdcan2008 slides dtrace freebsd john birrell http://www.bsdcan.org/2008/schedule/attachments/60_dtrace_bsdcan.pdf
49 pages 148 Kb PDF file pdf

Matthieu Herrb - X.org

<http://www.bsdcan.org/2008/schedule/events/94.en.html>

X.org

upcoming plans

The X.Org project provides an open source implementation of the X Window System. The development work is being done in conjunction with the freedesktop.org community. The X.Org Foundation is the educational non-profit corporation whose Board serves this effort, and whose Members lead this work.

The X window system has been changing a lot in the recent years, and still changing. This talk will present this evolution, summarizing what has already been done and showing the current roadmap for future evolutions, with some focus on how *BSD kernels can be affected by the developments done with Linux as the primary target.

bsdcan bsdcan2008 slides x.org matthieu herrb http://www.bsdcan.org/2008/schedule/attachments/51_bsdcan08-xorg.pdf 30 pages 1.6 Mb PDF file pdf

Adrian Chad - What Not To Do When Writing Network Applications

<http://www.bsdcan.org/2008/schedule/events/72.en.html>

What Not To Do When Writing Network Applications

The lessons learnt working with not-so-high-performance network applications

This talk will look at issues which face the modern network application developer, from the point of view of poorly-designed examples. This will cover internal code structure and dataflow, interaction with the TCP stack, IO scheduling in high and low latency environments and high-availability considerations. In essence, this presentation should be seen as a checklist of what not to do when writing network applications.

Plenty of examples of well designed network applications exist in the open and closed source world today. Unfortunately there are just as many examples of fast network applications as there are “fast but workload specific”; sometimes failing miserably in handling the general case. This may be due to explicit design (eg Varnish) but many are simply due to the designer not fully appreciating the wide variance in “networks” - and their network application degrades ungracefully when under duress. My aim in this presentation is to touch on a wide number of issues which face network application programmers - most of which seem not “application related” to the newcomer - such as including pipelining into network communication, managing a balance between accepting new requests and servicing existing requests, or providing back-pressure to a L4 loadbalancer in case of traffic bursts. Various schemes for working with these issues will be presented, and hopefully participants will walk away with more of an understanding about how the network, application and operating systems interact.

bsdcan bsdcan2008 slides network applications adrian chad http://www.bsdcan.org/2008/schedule/attachments/61_BSDCan2008-Network-Applications.pdf 73 pages 190 Kb PDF file pdf

Brooks Davis - Using FreeBSD to Promote Open Source Development Methods

<http://www.bsdcan.org/2008/schedule/events/64.en.html>

Using FreeBSD to Promote Open Source Development Methods

In this talk we present Aerosource, an initiative to bring Open Source Software development methods to internal software developers at The Aerospace Corporation.

Within Aerosource, FreeBSD is used in several key roles. First, we run most of our tools on top of FreeBSD. Second, the ports collection (both official ports and custom internal ones) eases our administrative burden. Third, the FreeBSD project serves as an example and role model for the results that can be achieved by an Open Source Software projects. We discuss the development infrastructure we have built for Aerosource based largely on BSD licensed software including FreeBSD, PostgreSQL, Apache, and Trac. We will also discuss our custom management tools including our system for managing our custom internal ports. Finally, we will cover our development successes and how we use projects like FreeBSD as exemplars of OSS development.

bsdcan bsdcan2008 abstract software development brooks davis http://www.bsdcan.org/2008/schedule/attachments/43_extended-abstract.pdf 2 pages 72 Kb PDF file pdf http://www.bsdcan.org/2008/schedule/attachments/62_freebsd-oss-methods.pdf 33 pages 1 Mb PDF file pdf

Randall Stewart - SCTP what it is and how to use it

<http://www.bsdcan.org/2008/schedule/events/91.en.html>

SCTP - SCTP what it is and how to use it

This talk will introduce the attendee into the interesting world of SCTP.

We will first discuss the new and different features that SCTP (a new transport in FreeBSD 7.0) provide to the user. Then we will shift gears and discuss the extended socket API that is available to SCTP users and will cover such items as:

- The two socket programming models
- Extended system calls that support the SCTP feature set.
- What model may fit you best

bsdcan bsdcan2008 abstract freebsd sctp randall stewart http://www.bsdcan.org/2008/schedule/attachments/44_bsdcan_sctp.pdf
10 pages 130 Kb PDF file pdf

Rafal Jaworowski - Porting FreeBSD/ARM to Marvell Orion System-On-Chip

<http://www.bsdcan.org/2008/schedule/events/73.en.html>

Porting FreeBSD/ARM to Marvell Orion System-On-Chip

This talk covers the development work on porting the FreeBSD/ARM to Marvell Orion family of highly integrated chips.

ARM architecture is widely adopted in the embedded devices, and since the architecture can be licensed, many implementation variations exist: Orion is a derivative compliant with the ARMv5TE definition, it provides a rich set of on-chip peripherals.

Present state of the FreeBSD support for ARM is explained, areas for improvement highlighted and its overall shape and condition presented.

The main discussion covers scope of the Orion port (what integrated peripherals required new development, what was adapted from existing code base); design decisions are explained for the most critical items, and implementation details revealed.

Summary notes are given on general porting methodology, debugging techniques and difficulties encountered during such undertaking.

bsdcan bsdcan2008 slides freebsd arm marvell orion rafal jaworowski http://www.bsdcan.org/2008/schedule/attachments/50_2008_marvell_orion.pdf
25 pages 193 Kb PDF file pdf

Dan Langille - BSDCan 2008 - Opening session

<http://www.bsdcan.org/2008/schedule/events/59.en.html>

Opening session

Welcome to BSDCan 2008

Traditional greetings bsdcan bsdcan2008 slides dan langille

http://www.bsdcan.org/2008/schedule/attachments/48_BSDCan2008Opening.pdf 17 pages 500 Kb PDF file pdf

BSDCan-2007 - Videos

<http://people.freebsd.org/~julian/BSDCan-2007/> The 2007 BSDCan conference

Kirk McKusick - Code Reading of Locally-Connected Sockets

bsdcan bsdcan2007 talks kirk mckusick

http://people.freebsd.org/~julian/BSDCan-2007/Kirk_UnixDomain.mov 35 minutes 77 Mb MOV file quicktime

BSDCan-2007 - Videos

<http://people.freebsd.org/~julian/BSDCan-2007/> The 2007 BSDCan conference

Erwin Lansing - The state of the FreeBSD Ports Tree

bsdcan bsdcan2007 talks erwin lansing ports

<http://people.freebsd.org/~julian/BSDCan-2007/Lansing-Portmanager.mov> 20 minutes 39 Mb MOV file quicktime

BSDCan-2007 Videos

<http://people.freebsd.org/~julian/BSDCan-2007/> The 2007 BSDCan conference - Introduction of people. bsdcan bsdcan2007 talks <http://people.freebsd.org/~julian/BSDCan-2007/Intro.mov> 9 minutes 16 Mb MOV file quicktime

BSDCan-2007 - Videos

<http://people.freebsd.org/~julian/BSDCan-2007/> The 2007 BSDCan conference

Kris Kennaway - Scalability Update 2007

Progress on FreeBSD SMP performance and scalability since BSDCan Dev Summit 2006 bsdcan bsdcan2007

talks kris kennaway scalability http://people.freebsd.org/~julian/BSDCan-2007/kris_kennaway-scalability.mov 73 minutes 148 Mb MOV file quicktime

BSDCan-2007 - Videos

<http://people.freebsd.org/~julian/BSDCan-2007/> The 2007 BSDCan conference

Qing Li - Routing, ARP and ND6 bsdcan bsdcan2007 talks qing li routing arp and nd6

http://people.freebsd.org/~julian/BSDCan-2007/QingLi_Arp.mov 30 minutes 63 Mb MOV file quicktime

BSDCan-2007 - Videos

<http://people.freebsd.org/~julian/BSDCan-2007/> The 2007 BSDCan conference

Marko Zec explains the vimage architecture bsdcan bsdcan2007 talks marko zec vimage

<http://people.freebsd.org/~julian/BSDCan-2007/marko-vimage.mov> 20 minutes 44 Mb MOV file quicktime

BSDCan-2007 - Videos

<http://people.freebsd.org/~julian/BSDCan-2007/> The 2007 BSDCan conference

Max Laier - PFIL, firewalls and locking bsdcan bsdcan2007 talks max laier ipf

http://people.freebsd.org/~julian/BSDCan-2007/max_ipf_pfil.mov 30 minutes 52 Mb MOV file quicktime

BSDCan-2007 Photos

<http://gallery.keltia.net/v/voyages/conferences/bsdcan-2007/> Photos taken during both DevSummit and Conference at BSDCan 2007 in Ottawa by Ollivier Robert. bsdcan bsdcan2007 photos ollivier robert

BSDCan-2007 Photos - Friday

http://www.db.net/gallery/BSDCan/BSDCan_2007_Friday/ Photos taken during both DevSummit and Conference on Friday at BSDCan 2007 in Ottawa by Diane Bruce. bsdcan bsdcan2007 photos diane bruce

BSDCan-2007 Photos - Saturday

http://www.db.net/gallery/BSDCan/BSDCan_2007_Saturday/ Photos taken during both DevSummit and Conference on Saturday at BSDCan 2007 in Ottawa by Diane Bruce. bsdcan bsdcan2007 photos diane bruce

BSDCan-2007 Photos - Scott Murphy

<http://scott5.vox.com/library/post/bsdcan-2007-photos.html> Photos taken at BSDCan 2007 by Scott Murphy bsdcan bsdcan2007 photos scott murphy

BSDCan-2007 Photos - Bjoern A. Zeeb

<http://www.zabbadoz.net/users/bz/BSDCan2007/BSDCan2007-public/> Photos taken at BSDCan 2007 by Bjoern A. Zeeb bsdcan bsdcan2007 photos bjoern a zeeb

BSDCan-2007 Photos - Randi Harper

<http://www.flickr.com/photos/freebsdgirl/sets/72157600230001160/> Photos taken at BSDCan 2007 by Randi Harper bsdcan bsdcan2007 photos randi harper freebsdgirl

BSDCan-2007 Photos - Dru Lavigne

<http://picasaweb.google.com/dru.lavigne/BSDCan2007> Photos taken at BSDCan 2007 by Dru Lavigne bsdcan bsdcan2007 photos dru lavigne

The FreeBSD Security Officer function

<http://people.freebsd.org/~simon/presentations/> “FreeBSD Security Officer function” at BSDCAN 2007 by Simon L. Nielsen (FreeBSD Deputy Security Officer) bsdcan bsdcan2007 pdf freebsd security officer simon l nielsen <http://people.freebsd.org/~simon/presentations/freebsd-so-function-bsdcan-2007.pdf> 252 Kb 29 pages PDF version pdf

FreeBSD Portsnap

<http://www.daemonology.net/papers/> “FreeBSD Portsnap - What (it is), Why (it was written), and How (it works)” by Colin Percival (cperciva@FreeBSD.org)

(Note: use ^L to get back in non-fullscreen mode) bsdcan bsdcan2007 pdf portsnap freebsd colin percival <http://www.daemonology.net/papers/bsdcan07.pdf> 1.3 Mb 88 pages PDF version pdf

BSDConTR 2007 - Photos

<http://www.bsdcontr.org/gallery/bsdcontr07/> Photos of the BSDConTR 2007 bsdcontr bsdcontr2007 photos

BSDConTR 2007 - Presentations

<http://www.bsdcontr.org/> Introducing FreeBSD 7.0 bsdcontr bsdcontr2007 pdf freebsd 7.0 freebsd kris kennaway <http://people.freebsd.org/~kris/scaling/7.0%20Preview.pdf> 336 Kb 37 pages PDF version pdf

Server deployment in mass-hosting environment using FreeBSD Ports system by Stanislav Sedov (in russian)

<http://blog.springdaemons.com/freebsd/>

Recently I have been attending Hostobzor 12th, the Russian conference of hosting providers, beeing held at Raivola hotel near St. Petersburg. The event was great as always thanks to organizers. There was a number of intersting talks given, a lot of interesting discussions held, and, what I appreciate better, a lot of new people with great ideas met.

I gave a talk on using the FreeBSD Ports system to mänge a large-scale virtual hosting installations based on Hosting Telesystems experience. I tried to describe in detail how we use the ports collection to deploy a large number of servers diverced by architecture and OS versions, how we build packages and distribute them among servers, talked about how we use Mercurial VCS to incrementally merge upstream changes into our modified ports collection and FreeBSD src trees. Hopefully, I’ve not screwed it much... At least, some people was interested a lot and asked interesting questions.

hostobzor hostobzor12 freebsd ports stanislav sedov russian <http://blog.springdaemons.com/assets/2008/11/23/text.pdf> 61 Kb 5 pages PDF version paper pdf <http://blog.springdaemons.com/assets/2008/11/23/slides.pdf> 470 Kb 30 pages PDF version slides pdf

Cambridge FreeBSD DevSummit2012 - Photos - Ollivier Robert

<http://gallery.keltia.net/v/voyages/conferences/devsummit-cam-2012/> Photos of the 2012 FreeBSD DevSummit at the University of Cambridge by Ollivier Robert devsummit2012 devsummit photos ollivier robert

Welcome - Cambridge University FreeBSD DevSummit - Robert Watson

<http://wiki.freebsd.org/200808DevSummit> Welcome by Robert Watson devsummit2008 devsummit pdf freebsd robert watson <http://wiki.freebsd.org/200808DevSummit?action=AttachFile&do=get&target=20080815-welcome.pdf> 264 Kb 12 pages PDF version pdf

variant Symlinks - Brooks Davis

<http://wiki.freebsd.org/200808DevSummit>

Variant Symlinks by Brooks Davis

devsummit2008 devsummit pdf freebsd variant symlinks brooks davis <http://wiki.freebsd.org/200808DevSummit?action=AttachFile&do=get&target=20080815-variant-symlinks-for-freebsd.pdf> 213 Kb 15 pages PDF version pdf

Cambridge FreeBSD DevSummit2008 - Photos - Kris Kennaway

<http://people.freebsd.org/~kris/Cambridge/> Photos of the 2008 FreeBSD DevSummit at the Cambridge University by Kris Kennaway. devsummit2008 devsummit photos kris kennaway

Cambridge FreeBSD DevSummit2008 - Photos - Ollivier Robert

<http://gallery.keltia.net/v/voyages/conferences/devsummit-cam/> Photos of the 2008 FreeBSD DevSummit at the Cambridge University by Ollivier Robert devsummit2008 devsummit photos ollivier robert

Cambridge FreeBSD DevSummit2008 - Photos - Simon Nielsen

<http://people.freebsd.org/~simon/gallery/cambridge-2008/> Photos of the 2008 FreeBSD DevSummit at the Cambridge University by Simon Nielsen. devsummit2008 devsummit photos simon nielsen

Van FreeBSD Documentatie projectleider tot FreeBSD Developer - Remko Lodder

http://www.nllgg.nl/communitydag_20081213#freebsd-doc-2-dev

In 2004 ben ik begonnen met het FreeBSD Dutch Documentation Project, een project dat inmiddels bijna het complete handboek vertaald heeft. Sinds die tijd zijn er vele wegen geweest die ik behandeld heb, van documentatie projectleider naar Security Team-lid tot aan FreeBSD Developer.

Remko Lodder is momenteel 25 jaar en werkt als Unix Engineer voor het bedrijf Snow B.V. waar hij zich momenteel met name bezig houdt met security (firewalls etc). Hij is sinds 2004 lid van het FreeBSD Development team en is momenteel 1 van de meest actieve developers binnen het team.

nllgg freebsd documentation nederlands remko lodder <http://www.evilmcoder.org/download/9/> 594 Kb 24 pages PDF version pdf

Een historisch overzicht van BSD - Hans van de Looy

http://www.nllgg.nl/communitydag_20081213#bsd-history

Hans zal een historisch overzicht geven van het ontstaan van *BSD vanaf de oorsprong van UNIX tot aan de nu bekende *BSD varianten. Hij zal daarbij met name ingaan wat de oorsprong en het ontstaan van een aantal *BSD-projecten zijn. Hierbij zal hij zeer kort ingaan op de verschillende licentieproblemen die we in het verleden gezien hebben en worden een aantal bekende personen en data weer eens even op de kaart geplaatst.

Hans van de Looy is oprichter van Madison Gurkha. Een bedrijf dat gespecialiseerd is op het gebied van het uitvoeren van technische ICT-beveiligingsonderzoeken, in de media ook wel aangeduid met Etisch Hacken. Tijdens dergelijke onderzoeken maakt hij ook regelmatig gebruik van op BSD* gebaseerde systemen.

nllgg bsd history hans van de looy <http://www.nllgg.nl/uploads/2078/HansvandeLooy.pdf> 5767 Kb 38 pages PDF version pdf

FreeBSD Google Summer of Code posters

<http://www.freebsd.org/projects/summerofcode.html> Two posters usable for the announcement of the participation of the FreeBSD Project in the Google Summer of Code. freebsd google summer of code <http://people.freebsd.org/~manolis/2009-freebsd-gsoc-alternate.png> 1.1 Mb 2480 x 3507 pixels PNG version png <http://people.freebsd.org/~manolis/2009-freebsd-gsoc-alternate.pdf> 815 Kb 1 page PDF version pdf

PmcTools talk at the Bangalore chapter of the ACM

<http://edoofus.blogspot.com/2009/04/pmctools-talk-at-bangalore-chapter-of.html>

In April 2009 I was invited to speak on FreeBSD/PmcTools by the Bangalore chapter of the ACM.

This was an overview talk. The talk briefly touched upon: the motivations and goals of the project, the programming APIs, some aspects of the implementation and on possible future work.

freebsd presentation freebsd pmctools joseph koshy <http://people.freebsd.org/~jkoshy/download/acm-apr-09.pdf> 550 Kb 48 pages PDF version pdf

FreeBSD Google Summer of Code <http://www.freebsd.org/projects/summerofcode.html>

bsdtalk <http://bsdtalk.blogspot.com/>
New York City *BSD User Group <http://www.nycbug.org/>
Bay Area FreeBSD User Group <http://www.bafug.org/>
FreeBSD for All <http://freebsdforall.blogspot.com/>
Daemon News <http://www.daemonnews.org/>
Source21.nl <http://www.source21.nl/>
22nd Chaos Communication Congress <http://events.ccc.de/congress/2005/>
Norwegian Unix Users Group <http://www.nuug.no/>
OpenBSD <http://www.openbsd.org/>
DCBSDCon <http://www.dcbsdcon.org/>
EuroBSDCon <http://www.eurobsdcon.org/>
OpenFest <http://openfest.org/>
Robert Watson <http://www.watson.org/~robert/>
Joseph Koshy <http://edoofus.blogspot.com/>
Swiss Unix Users Group Conference 2004 <http://conferences.suug.ch/sucon/04/>
Andre Opperman <http://people.freebsd.org/~andre/>
Poul-Henning Kamp <http://people.freebsd.org/~phk/>
Diomidis Spinellis <http://www.spinellis.gr/>
AAarhus Unix Users Group <http://www.aauug.dk/>
BSD UNIX bruger gruppe i Danmark <http://www.bsd-dk.dk/>
The Linux Tink Tech Show <http://www.tlts.org/>
UKUUG <http://www.ukuug.org/>
Ottawa Amateur Radio Club <http://www.oarc.net/>
AsiaBSDCon <http://www.asiabsdcon.org/>
Linux and FreeBSD video tutorials. For everyone. <http://www.asiabsdcon.org/>
Berklix.com Computer Services <http://www.berklix.com/>
BSDCan - The Technical BSD Conference <http://www.bsdcan.org/>
BSDConTR - Turkish Conference on BSD Systems <http://www.bsdcontr.org/>
Free and Open Source Software Developers' European Meeting <http://fosdem.org/>
MeetBSD <http://www.meetbsd.org/>
MeetBSD <http://www.meetbsd.com/>
BSDCon Spain <http://www.bsdcon.net/>
Google Tech Talks <http://video.google.com/videosearch?q=type%3Agoogle+engEDU&so=1>
FreeBSD Developer Summit - Cambridge <http://wiki.freebsd.org/DevSummit>
Hostobzor, the Russian conference of hosting provider <http://www.hostobzor.ru/>
YouTube bsdconferences channel <http://www.youtube.com/bsdconferences>

Nederlandse Linux Gebruikers Group <http://www.nlgg.nl/>

&title;

|>

33.1 Privacy Policy

The &os; Project recognizes that your privacy and the protection of your personal information is important to you and is committed to protecting your privacy. This Privacy Policy describes the measures taken by us to protect your privacy in connection with your use of www.FreeBSD.org (our “Site”). This policy does not cover mirrors of this content on other hosts (the “Mirrors”) or other &os;-related services in the &os;.org domain (“Affiliated Sites”).

This policy was last modified on October 19, 2012.

PLEASE READ THIS PRIVACY POLICY CAREFULLY. BY USING THIS SITE, YOU AGREE TO THE TERMS OF THIS POLICY.

33.2 What information is collected on our Site:

When you visit our Site we gather information regarding your use of our site, such as the pages you view and the time you view each page. We also collect information such as your Internet Protocol (“IP”) address, browser type and operating system information. The foregoing information is collectively referred to below as “Your Usage Data.”

We store information such as your email address, name, and organization or company (“Personally Identifiable Information”) only if you decide to send us such information by submitting a &os; bug or problem report (“PR”), subscribing to one of the &os; mailing lists or posting content to our Site forums. The &os; Project allows unlimited distribution of that content. Information submitted in those reports and lists, including your Personally Identifiable Information, is considered public and will be accessible to anyone on the web. Such information is not only stored on servers belonging to the &os; Foundation, but it is also stored on other servers that mirror the content of our Site. The &os; Foundation has no control over the use of that information, including your Personally Identifiable Information.

This website uses a cookie to record layout style preferences for individual users, as well track anonymous traffic data. A cookie is a small text file provided by a web server that may be placed on your computer by your web browser. Cookies cannot be used to run programs or deliver viruses to your computer. Cookies are uniquely assigned to you, and can only be read by a Web server in the domain that issued the cookie to you. You may refuse the use of cookies by selecting the appropriate settings on your browser, however please note that if you do this you may not be able to use the full functionality of this or other websites.

33.3 How the information collected is used:

The &os; Project takes the privacy of our users and members very seriously. We will never sell, rent, or otherwise provide your personally identifiable information to any third parties except as stated in this document. We will not associate any data gathered from your use of our Site with any personally identifying information.

The &os; Project will not use or share your personal information in a manner that differs from what is described in this Privacy Policy without your prior consent.

33.4 How your information is protected:

The &os; Project strictly protects the security of the personal information you provide. Personal information we collect is stored in password-controlled servers with limited access, and we carefully protect this information from loss or misuse, and from unauthorized access, disclosure, alteration, or destruction.

33.5 When your information may be disclosed:

The &os; Project may disclose personal information if required to do so by law or in the good-faith belief that such action is necessary to:

1. conform to the requirements of the law or comply with legal process served on The &os; Project or the Site;
2. protect and defend the rights or property of The &os; Project; or
3. act in urgent circumstances to protect personal safety of users of The &os; Project, its websites, or the public.

Anonymized aggregate traffic data about website visits may be shared for marketing or research purposes.

The &os; Project uses Google Analytics, a web analytics service provided by Google, Inc. Google Analytics is a tool that enables the &os; Project to see how visitors to our Site use our Site and how they arrive at our Site. Google Analytics uses “cookies”, which are text files placed on your computer, to help our Site to analyze visitor activity. We send Google Analytics data that reports on your use of the Site. That data includes Your Usage Data and is not anonymised before it is transmitted to Google Analytics. This un-anonymised data is stored by Google on servers, once it is transmitted to Google. Google will use this information for the purpose of evaluating your use of the Site, compiling reports on website activity for website operators and providing other services relating to website activity and internet usage. Google may also transfer Your Usage Data to third parties where required to do so by law, or where such third parties process the information on Google’s behalf. Google will not associate Your Usage Data with any other data held by Google. You may refuse the use of cookies by selecting the appropriate settings on your browser, however please note that if you do this you may not be able to use the full functionality of our Site. By using our Site, you consent to the processing of Your Usage Data by Google in the manner and for the purposes set out above.

With the exception of the transmission of Your Usage Data to Google for the sole purpose of using Google Analytics and the public access we permit to the bug reports you submit, the mailing lists you subscribe to, and posting content to our Site forums, the &os; Project does not share un-anonymised data with anyone.

33.6 Updates to this privacy policy:

We may occasionally update this Privacy Policy. When we do, we will also revise the “last modified” date at the top of the Privacy Policy. We encourage you to periodically review this Privacy Policy to stay informed about how we are protecting the personal information we collect. Your continued use of the service constitutes your agreement to this Privacy Policy and any updates.

33.7 Sale of assets:

In the unlikely event that The &os; Project or substantially all of its assets are acquired, your information may be one of the transferred assets.

33.8 Third party links:

In an attempt to provide you with increased value, we may include third party links on our site. These linked sites have separate and independent privacy policies. This Privacy Policy only covers our Site and does not cover any other website. We therefore have no responsibility or liability for the content and activities of these linked sites. Nonetheless, we seek to protect the integrity of our site and welcome any feedback about these linked sites (including if a specific link does not work).

33.9 Contacting Us:

We would like to receive your comments and questions about this Privacy Policy and any other matter you have regarding our Site. Please address comments or questions to us via e-mail at: privacy@FreeBSD.org.

&title;

]>

34.1 Go fix/close a PR!

It's easy, all you have to do is to follow [this link to the Bugzilla database](#), find a PR, figure out what needs done and do it.

If you are a committer, you can use the web interface to close PRs.

If you are not a committer, you should submit a followup containing the text :

This PR can be closed

on a line of its own, that will make it easier for a committer to deal with it later.

For various summaries of the PR database, see the following [charts and reports](#).

&title;

|>



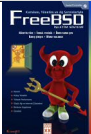

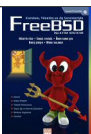

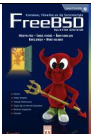


- | | |
|--|--|
| | <ul style="list-style-type: none">• <i>BOOKS</i>• <i>CDROMS</i>• <i>MAGAZINES</i>• PRESS |
|--|--|

Here you will find the covers of many FreeBSD related publications. If you know of any additional FreeBSD publications/CDROMs let us know, at www@FreeBSD.org, so that they may be added to this site.

The FreeBSD Handbook contains a considerably longer bibliography.

Click on any of the graphics to see a larger version.

35.1 Books

	A publication from Tatsumi Hosokawa and others. Among computer books, it is a top-seller in Japan and exceeded the sales of Bill Gates' "The Road Ahead" when published (it was #2, this book was #1).
	(Japanese FreeBSD book with 2.0.5, titled "FreeBSD: Fun and easy Installation")
	(Japanese FreeBSD book with 2.0.5, titled "FreeBSD Introductory Kit")
	This is BSDi's "The Complete FreeBSD" with installation guide, manual pages and installation CDs inside.
	This book was published (early 1997) in Taiwan. Its title is "FreeBSD: introduction and applications" and the author is Jian-Da Li (aka. jdli).
	This is the "Getting Started with FreeBSD" from Fuki-Shuppan. Other than the standard installation guide and Japanese environment, it emphasizes system administration and low-level information (such as the boot process, etc.) FreeBSD-2.2.2R and XFree86-3.2 on CDROM. 264 pages, 3,400 yen.
	The "Personal Unix Starter Kit - FreeBSD" from ASCII. Includes history of &unix;, a guide to build a Japanese documentation processing system and how to create ports. 2.1.7.1R and XFree86-3.2 in CDROM. 384 pages, 3,000 yen.
	BSD mit Methode, M. Schulze, B. Roehrig, M. Hoelzer und andere, C&L Computer und Literatur Verlag, 1998, 850 pages. 2 CDROMs, FreeBSD 2.2.6, NetBSD 1.2.1 and 1.3.2, OpenBSD 2.2 and 2.3. DM 98,-.
	

35.2 CDRoms

For more about recent releases go to [FreeBSD release information page](#).



This is InfoMagic's BSDisc, containing FreeBSD 2.0 and NetBSD 1.0.



This is the original 4.4 BSD Lite2 release from UC Berkeley, the core



The first of Laser5's "BSD" series. Contains FreeBSD-2.0.5R, NetBSD



The second of Laser5's "BSD" series. From this version, the CDs com



This is the Laser5 Japanese edition of the FreeBSD CDROM. It is a 4



This is the only FreeBSD CD Pacific Hitech produced before merging



This is the cover disc from the Korean *magazine*. Note the creative cover



This is it - the very first FreeBSD CD published! Both the FreeBSD Pr



This was the second FreeBSD CD published by Walnut Creek CDRom



This unusual CD is something of a collector's item now given that alm



This is the fixed-up version of the FreeBSD 2.0 CD. Note that the color



The FreeBSD 2.0.5 release CD. This was the first CD to feature Tatsuru



The FreeBSD 2.1 release CD. This was the first CD release on the 2.1



The FreeBSD 2.1.5 release CD.



The FreeBSD 2.1.6 release CD.



The Japanese version of 2.1.6. This was the first and last Japanese local



The FreeBSD 2.1.7 release CD. Also the last CD released on the 2.1.x



An early release SNAPshot of 2.2 (done before 2.2.1 was released).



The FreeBSD 2.2.1 release CD. This was the first CD on the 2.2 branch.



The FreeBSD 2.2.2 release CD.



The FreeBSD 3.0 snapshot CD.



The FreeBSD mailing list and newsgroup archives, turned into HTML



FreeBSD Toolkit: Six disc set of resources to make your FreeBSD exp



FreeBSD Alpha 4.2 - The full version of the DEC Alpha 64-bit UNIX



FreeBSD 4.2: The full version of the PC 32-bit UNIX operating system



FreeBSD 4.2 CD-ROM. Lehmanns CD-ROM Edition. January 2001, 4



FreeBSD 4.3 RELEASE CDROM. April 2001, Wind River Systems. I



FreeBSD Toolkit: Six disc set of resources to make your FreeBSD exp



FreeBSD 4.4 CD-ROM. Lehmanns CD-ROM Edition. November 2001








FreeBSD 4.4 RELEASE CDROM. Wind River Systems. September 2001



FreeBSD 4.5 RELEASE CDROM. February 2002, FreeBSD Mall Inc.

35.3 Magazines

	<p>Cover of Korean UNIX magazine, May 1997 issue. Also included <i>FreeBSD 2.2.1</i> with cover CDs.</p>
	<p>UNIX User Magazine November 1996 issue. Also included FreeBSD 2.1.5 on cover CD.</p>
	<p>This is the “FreeBSD Full Course” special in April 1997’s Software Design (published by Gijutsu Hyoron Sha). There are 80 pages of FreeBSD articles covering everything from installation to tracking -current.</p>
	<p>Quality Unix for FREE, by Brett Glass in Sm@rt Reseller Online September 1998</p>
	<p>This is the “BSD magazine” published by ASCII corporation, the world’s first publication specialized in BSD. BSD magazine covers FreeBSD, NetBSD, OpenBSD and BSD/OS. The premiere issue features articles on the history of BSD, installation, and Ports/Packages; it also includes 4 CD-ROMs containing FreeBSD 3.2-RELEASE, NetBSD 1.4.1 and OpenBSD 2.5.</p>
<p>35.3. Magazines</p>	<p>177</p>

&title;

]>



FreeBSD releases are classified into “Production Releases” and “Legacy Releases”. The former are best suited to users looking for the latest new features; the latter are for users wishing to stay with a more conservative upgrade strategy.

Releases are further classified by the length of time they will be supported by the Security Officer into “Normal” and “Extended” releases.

Documentation files for each release are available for viewing in HTML format on the Release Documentation page.

36.1 Currently Supported Releases

Complete information about the release date, the classification type, and the estimated End-Of-Life (EOL) for currently supported releases can be found on the Supported Releases section of the FreeBSD Security Information page.

36.2 Most Recent Release(s)

36.2.1 Production Releases

Release &rel.current; (&rel.current.date;) ‘*Announcement*’ <&u.rel.announce;>‘__’ : ‘*Release Notes*’ <&u.rel.notes;>‘__’ : ‘*Installation Instructions*’ <&u.rel.installation;>‘__’ : ‘*Hardware Notes*’ <&u.rel.hardware;>‘__’ : ‘*Readme*’ <&u.rel.readme;>‘__’ : ‘*Errata*’ <&u.rel.errata;>‘__’

36.2.2 Legacy Releases

Release &rel2.current; (&rel2.current.date;) ‘*Announcement*’ <&u.rel2.announce;>‘__’ : ‘*Release Notes*’ <&u.rel2.notes;>‘__’ : ‘*Installation Instructions*’ <&u.rel2.installation;>‘__’ : ‘*Hardware Notes*’ <&u.rel2.hardware;>‘__’ : ‘*Readme*’ <&u.rel2.readme;>‘__’ : ‘*Errata*’ <&u.rel2.errata;>‘__’

Release &rel3.current; (&rel3.current.date;) 'Announcement' <&u.rel3.announce;>'__': 'Release Notes' <&u.rel3.notes;>'__': 'Installation Instructions' <&u.rel3.installation;>'__': 'Hardware Notes' <&u.rel3.hardware;>'__': 'Readme' <&u.rel3.readme;>'__': 'Errata' <&u.rel3.errata;>'__'

36.3 Future Releases

For the schedule of upcoming releases, or for more information about the release engineering process, please visit the Release Engineering page.

The latest snapshots from our FreeBSD-STABLE and FreeBSD-CURRENT branches are also available. Please see Getting FreeBSD for details.

36.4 Prior Releases Which Have Reached End-Of-Life

Complete historical information about the release date, the classification type, and the effective End-Of-Life (EOL) for these releases may be found on the Unsupported Releases section of the FreeBSD Security Information page.

- **10.0** (January 2014) 'Announcement' <10.0R/announce.html>'__': 'Release Notes' <10.0R/relnotes.html>'__': 'Installation Instructions' <10.0R/installation.html>'__': 'Hardware Notes' <10.0R/hardware.html>'__': 'Readme' <10.0R/readme.html>'__': 'Errata' <10.0R/errata.html>'__'
- **9.2** (September 2013) 'Announcement' <9.2R/announce.html>'__': 'Release Notes' <9.2R/relnotes.html>'__': 'Installation Instructions' <9.2R/installation.html>'__': 'Hardware Notes' <9.2R/hardware.html>'__': 'Readme' <9.2R/readme.html>'__': 'Errata' <9.2R/errata.html>'__'
- **9.1** (December 2012) 'Announcement' <9.1R/announce.html>'__': 'Release Notes' <9.1R/relnotes.html>'__': 'Installation Instructions' <9.1R/installation.html>'__': 'Hardware Notes' <9.1R/hardware.html>'__': 'Readme' <9.1R/readme.html>'__': 'Errata' <9.1R/errata.html>'__'
- **9.0** (January 2012) 'Announcement' <9.0R/announce.html>'__': 'Release Notes' <9.0R/relnotes.html>'__': 'Installation Instructions' <9.0R/installation.html>'__': 'Hardware Notes' <9.0R/hardware.html>'__': 'Readme' <9.0R/readme.html>'__': 'Errata' <9.0R/errata.html>'__'
- **8.3** (April 2012) 'Announcement' <8.3R/announce.html>'__': 'Release Notes' <8.3R/relnotes.html>'__': 'Installation Instructions' <8.3R/installation.html>'__': 'Hardware Notes' <8.3R/hardware.html>'__': 'Readme' <8.3R/readme.html>'__': 'Errata' <8.3R/errata.html>'__'
- **8.2** (February 2011) 'Announcement' <8.2R/announce.html>'__': 'Release Notes' <8.2R/relnotes.html>'__': 'Hardware Notes' <8.2R/hardware.html>'__': 'Readme' <8.2R/readme.html>'__': 'Errata' <8.2R/errata.html>'__'
- **8.1** (July 2010) 'Announcement' <8.1R/announce.html>'__': 'Release Notes' <8.1R/relnotes.html>'__': 'Hardware Notes' <8.1R/hardware.html>'__': 'Readme' <8.1R/readme.html>'__': 'Errata' <8.1R/errata.html>'__'
- **8.0** (November 2009) 'Announcement' <8.0R/announce.html>'__': 'Release Notes' <8.0R/relnotes.html>'__': 'Hardware Notes' <8.0R/hardware.html>'__': 'Readme' <8.0R/readme.html>'__': 'Errata' <8.0R/errata.html>'__'
- **7.4** (February 2011) 'Announcement' <7.4R/announce.html>'__': 'Release Notes' <7.4R/relnotes.html>'__': 'Hardware Notes' <7.4R/hardware.html>'__': 'Readme' <7.4R/readme.html>'__': 'Errata' <7.4R/errata.html>'__'
- **7.3** (March 2010) 'Announcement' <7.3R/announce.html>'__': 'Release Notes' <7.3R/relnotes.html>'__': 'Hardware Notes' <7.3R/hardware.html>'__': 'Readme' <7.3R/readme.html>'__': 'Errata' <7.3R/errata.html>'__'
- **7.2** (May 2009) 'Announcement' <7.2R/announce.html>'__': 'Release Notes' <7.2R/relnotes.html>'__': 'Hardware Notes' <7.2R/hardware.html>'__': 'Readme' <7.2R/readme.html>'__': 'Errata' <7.2R/errata.html>'__'

- **7.1** (January 2009) ‘*Announcement* <7.1R/announce.html>’ __: ‘*Release Notes* <7.1R/relnotes.html>’ __: ‘*Hardware Notes* <7.1R/hardware.html>’ __: ‘*Readme* <7.1R/readme.html>’ __: ‘*Errata* <7.1R/errata.html>’ __
- **7.0** (February 2008) ‘*Announcement* <7.0R/announce.html>’ __: ‘*Release Notes* <7.0R/relnotes.html>’ __: ‘*Hardware Notes* <7.0R/hardware.html>’ __: ‘*Readme* <7.0R/readme.html>’ __: ‘*Errata* <7.0R/errata.html>’ __
- **6.4** (November 2008) ‘*Announcement* <6.4R/announce.html>’ __: ‘*Release Notes* <6.4R/relnotes.html>’ __: ‘*Hardware Notes* <6.4R/hardware.html>’ __: ‘*Installation Notes* <6.4R/installation.html>’ __: ‘*Readme* <6.4R/readme.html>’ __: ‘*Errata* <6.4R/errata.html>’ __
- **6.3** (January 2008) ‘*Announcement* <6.3R/announce.html>’ __: ‘*Release Notes* <6.3R/relnotes.html>’ __: ‘*Hardware Notes* <6.3R/hardware.html>’ __: ‘*Installation Notes* <6.3R/installation.html>’ __: ‘*Readme* <6.3R/readme.html>’ __: ‘*Errata* <6.3R/errata.html>’ __
- **6.2** (January 2007) ‘*Announcement* <6.2R/announce.html>’ __: ‘*Release Notes* <6.2R/relnotes.html>’ __: ‘*Hardware Notes* <6.2R/hardware.html>’ __: ‘*Installation Notes* <6.2R/installation.html>’ __: ‘*Readme* <6.2R/readme.html>’ __: ‘*Errata* <6.2R/errata.html>’ __
- **6.1** (May 2006) ‘*Announcement* <6.1R/announce.html>’ __: ‘*Release Notes* <6.1R/relnotes.html>’ __: ‘*Hardware Notes* <6.1R/hardware.html>’ __: ‘*Installation Notes* <6.1R/installation.html>’ __: ‘*Readme* <6.1R/readme.html>’ __: ‘*Errata* <6.1R/errata.html>’ __
- **6.0** (November 2005) ‘*Announcement* <6.0R/announce.html>’ __: ‘*Release Notes* <6.0R/relnotes.html>’ __: ‘*Hardware Notes* <6.0R/hardware.html>’ __: ‘*Installation Notes* <6.0R/installation.html>’ __: ‘*Readme* <6.0R/readme.html>’ __: ‘*Errata* <6.0R/errata.html>’ __
- **5.5** (May 2006) ‘*Announcement* <./5.5R/announce.html>’ __: ‘*Release Notes* <./5.5R/relnotes.html>’ __: ‘*Hardware Notes* <./5.5R/hardware.html>’ __: ‘*Installation Notes* <./5.5R/installation.html>’ __: ‘*Readme* <./5.5R/readme.html>’ __: ‘*Errata* <./5.5R/errata.html>’ __
- **5.4** (May 2005) ‘*Announcement* <./5.4R/announce.html>’ __: ‘*Release Notes* <./5.4R/relnotes.html>’ __: ‘*Hardware Notes* <./5.4R/hardware.html>’ __: ‘*Installation Notes* <./5.4R/installation.html>’ __: ‘*Readme* <./5.4R/readme.html>’ __: ‘*Errata* <./5.4R/errata.html>’ __: ‘*Migration Guide* <./5.4R/migration-guide.html>’ __
- **5.3** (November 2004) ‘*Announcement* <./5.3R/announce.html>’ __: ‘*Release Notes* <./5.3R/relnotes.html>’ __: ‘*Hardware Notes* <./5.3R/hardware.html>’ __: ‘*Installation Notes* <./5.3R/installation.html>’ __: ‘*Readme* <./5.3R/readme.html>’ __: ‘*Errata* <./5.3R/errata.html>’ __: ‘*Migration Guide* <./5.3R/migration-guide.html>’ __
- **5.2.1** (February 2004) ‘*Announcement* <./5.2.1R/announce.html>’ __: ‘*Release Notes* <./5.2.1R/relnotes.html>’ __: ‘*Hardware Notes* <./5.2.1R/hardware.html>’ __: ‘*Installation Notes* <./5.2.1R/installation.html>’ __: ‘*Readme* <./5.2.1R/readme.html>’ __: ‘*Errata* <./5.2.1R/errata.html>’ __: ‘*Early Adopter’s Guide* <./5.2.1R/early-adopter.html>’ __
- **5.2** (January 2004) ‘*Announcement* <./5.2R/announce.html>’ __: ‘*Release Notes* <./5.2R/relnotes.html>’ __: ‘*Hardware Notes* <./5.2R/hardware.html>’ __: ‘*Installation Notes* <./5.2R/installation.html>’ __: ‘*Readme* <./5.2R/readme.html>’ __: ‘*Errata* <./5.2R/errata.html>’ __: ‘*Early Adopter’s Guide* <./5.2R/early-adopter.html>’ __
- **5.1** (June, 2003) ‘*Announcement* <./5.1R/announce.html>’ __: ‘*Release Notes* <./5.1R/relnotes.html>’ __: ‘*Hardware Notes* <./5.1R/hardware.html>’ __: ‘*Installation Notes* <./5.1R/installation.html>’ __: ‘*Readme* <./5.1R/readme.html>’ __: ‘*Errata* <./5.1R/errata.html>’ __: ‘*Early Adopter’s Guide* <./5.1R/early-adopter.html>’ __
- **5.0** (January, 2003) ‘*Announcement* <./5.0R/announce.html>’ __: ‘*Release Notes* <./5.0R/relnotes.html>’ __: ‘*Hardware Notes* <./5.0R/hardware.html>’ __: ‘*Installation Notes* <./5.0R/installation.html>’ __: ‘*Readme*

<./5.0R/readme.html>‘__’: ‘Errata <./5.0R/errata.html>‘__’: ‘Early Adopter’s Guide <./5.0R/early-adopter.html>‘__

- **4.11** (January, 2005) ‘Announcement <./4.11R/announce.html>‘__’: ‘Release Notes <./4.11R/relnotes.html>‘__’: ‘Hardware Notes <./4.11R/hardware.html>‘__’: ‘Installation Notes <./4.11R/installation.html>‘__’: ‘Readme <./4.11R/readme.html>‘__’: ‘Errata <./4.11R/errata.html>‘__
- **4.10** (May, 2004) ‘Announcement <./4.10R/announce.html>‘__’: ‘Release Notes <./4.10R/relnotes.html>‘__’: ‘Hardware Notes <./4.10R/hardware.html>‘__’: ‘Installation Notes <./4.10R/installation.html>‘__’: ‘Readme <./4.10R/readme.html>‘__’: ‘Errata <./4.10R/errata.html>‘__
- **4.9** (October, 2003) ‘Announcement <./4.9R/announce.html>‘__’: ‘Release Notes <./4.9R/relnotes.html>‘__’: ‘Hardware Notes <./4.9R/hardware.html>‘__’: ‘Installation Notes <./4.9R/installation.html>‘__’: ‘Readme <./4.9R/readme.html>‘__’: ‘Errata <./4.9R/errata.html>‘__
- **4.8** (April, 2003) ‘Announcement <./4.8R/announce.html>‘__’: ‘Release Notes <./4.8R/relnotes.html>‘__’: ‘Hardware Notes <./4.8R/hardware.html>‘__’: ‘Installation Notes <./4.8R/installation.html>‘__’: ‘Readme <./4.8R/readme.html>‘__’: ‘Errata <./4.8R/errata.html>‘__
- **4.7** (October, 2002) ‘Announcement <./4.7R/announce.html>‘__’: ‘Release Notes <./4.7R/relnotes.html>‘__’: ‘Hardware Notes <./4.7R/hardware.html>‘__’: ‘Installation Notes <./4.7R/installation.html>‘__’: ‘Readme <./4.7R/readme.html>‘__’: ‘Errata <./4.7R/errata.html>‘__
- **4.6.2** (August, 2002) ‘Announcement <./4.6.2R/announce.html>‘__’: ‘Release Notes <./4.6.2R/relnotes.html>‘__’: ‘Hardware Notes <./4.6.2R/hardware.html>‘__’: ‘Readme <./4.6.2R/readme.html>‘__’: ‘Errata <./4.6.2R/errata.html>‘__
- **4.6** (June, 2002) ‘Announcement <./4.6R/announce.html>‘__’: ‘Release Notes <./4.6R/relnotes.html>‘__’: ‘Hardware Notes <./4.6R/hardware.html>‘__’: ‘Installation Notes <./4.6R/installation.html>‘__’: ‘Errata <./4.6R/errata.html>‘__
- **4.5** (January, 2002) ‘Announcement <./4.5R/announce.html>‘__’: ‘Release Notes <./4.5R/notes.html>‘__’: ‘Hardware Notes <./4.5R/hardware.html>‘__’: ‘Errata <./4.5R/errata.html>‘__
- **4.4** (September, 2001) ‘Announcement <./4.4R/announce.html>‘__’: ‘Release Notes <./4.4R/notes.html>‘__’: ‘Hardware Notes <./4.4R/hardware.html>‘__’: ‘Errata <./4.4R/errata.html>‘__
- **4.3** (April, 2001) ‘Announcement <./4.3R/announce.html>‘__’: ‘Release Notes <./4.3R/notes.html>‘__’: ‘Errata <./4.3R/errata.html>‘__
- **4.2** (November, 2000) ‘Announcement <./4.2R/announce.html>‘__’: ‘Release Notes <./4.2R/notes.html>‘__’: ‘Errata <./4.2R/errata.html>‘__
- **4.1.1** (September, 2000) ‘Announcement <./4.1.1R/announce.html>‘__’: ‘Release Notes <./4.1.1R/notes.html>‘__’: ‘Errata <./4.1.1R/errata.html>‘__
- **4.1** (July, 2000) ‘Announcement <./4.1R/announce.html>‘__’: ‘Release Notes <./4.1R/notes.html>‘__’: ‘Errata <./4.1R/errata.html>‘__
- **4.0** (March, 2000) ‘Announcement <./4.0R/announce.html>‘__’: ‘Release Notes <./4.0R/notes.html>‘__’: ‘Errata <./4.0R/errata.html>‘__
- **3.5** (June, 2000) ‘Announcement <./3.5R/announce.html>‘__’: ‘Release Notes <./3.5R/notes.html>‘__’: ‘Errata <./3.5R/errata.html>‘__
- **3.4** (December, 1999) ‘Announcement <./3.4R/announce.html>‘__’: ‘Release Notes <./3.4R/notes.html>‘__’: ‘Errata <./3.4R/errata.html>‘__
- **3.3** (September, 1999) ‘Announcement <./3.3R/announce.html>‘__’: ‘Release Notes <./3.3R/notes.html>‘__’: ‘Errata <./3.3R/errata.html>‘__
- **3.2** (May, 1999) ‘Announcement <./3.2R/announce.html>‘__’: ‘Release Notes <./3.2R/notes.html>‘__’: ‘Errata <./3.2R/errata.html>‘__

- **3.1** (February, 1999) ‘*Announcement* <./3.1R/announce.html>’ __ : ‘*Release Notes* <./3.1R/notes.html>’ __ : ‘*Errata* <./3.1R/errata.html>’ __
- **3.0** (October, 1998) ‘*Announcement* <./3.0R/announce.html>’ __ : ‘*Release Notes* <./3.0R/notes.html>’ __ : ‘*Errata* <./3.0R/errata.html>’ __
- **2.2.8** (December, 1998) ‘*Announcement* <./2.2.8R/announce.html>’ __ : ‘*Release Notes* <./2.2.8R/notes.html>’ __ : ‘*Errata* <./2.2.8R/errata.html>’ __
- **2.2.7** (July, 1998) ‘*Announcement* <./2.2.7R/announce.html>’ __ : ‘*Release Notes* <./2.2.7R/notes.html>’ __ : ‘*Errata* <./2.2.7R/errata.html>’ __
- **2.2.6** (March, 1998) ‘*Announcement* <./2.2.6R/announce.html>’ __ : ‘*Release Notes* <./2.2.6R/notes.html>’ __ : ‘*Errata* <./2.2.6R/errata.html>’ __
- **2.2.5** (October, 1997) ‘*Announcement* <./2.2.5R/announce.html>’ __ : ‘*Release Notes* <./2.2.5R/notes.html>’ __ : ‘*Errata* <./2.2.5R/errata.html>’ __
- **2.2.2** (May, 1997) ‘*Release Notes* <./2.2.2R/notes.html>’ __ : ‘*Errata* <./2.2.2R/errata.html>’ __
- **2.2.1** (April, 1997) ‘*Release Notes* <./2.2.1R/notes.html>’ __
- **2.2** (March, 1997) ‘*Announcement* <./2.2R/announce.html>’ __ : ‘*Release Notes* <./2.2R/notes.html>’ __
- **2.1.7** (February, 1997) ‘*Announcement* <./2.1.7R/announce.html>’ __ : ‘*Release Notes* <./2.1.7R/notes.html>’ __
- **2.1.6** (December, 1996) ‘*Announcement* <./2.1.6R/announce.html>’ __ : ‘*Release Notes* <./2.1.6R/notes.html>’ __
- **2.1.5** (July, 1996) ‘*Announcement* <./2.1.5R/announce.html>’ __ : ‘*Release Notes* <./2.1.5R/notes.html>’ __
- **2.1** (November, 1995) ‘*Announcement* <./2.1R/announce.html>’ __ : ‘*Release Notes* <./2.1R/notes.html>’ __
- **2.0.5** (June, 1995) ‘*Announcement* <./2.0.5R/announce.html>’ __ : ‘*Release Notes* <./2.0.5R/notes.html>’ __
- **2.0** (November, 1994) ‘*Announcement* <./2.0/announce.html>’ __ : ‘*Release Notes* <./2.0/notes.html>’ __
- **1.1.5.1** (July, 1994)
- **1.1.5** ‘*Release Notes* <./1.1.5/RELNOTES.FreeBSD>’ __
- **1.1** (May, 1994) ‘*Release Notes* <./1.1/RELNOTES.FreeBSD>’ __
- **1.0** (November, 1993) ‘*Announcement* <./1.0/announce.html>’ __

&title;

>



Each distribution of FreeBSD includes several documentation files describing the particular distribution (RELEASE, SNAPSHOTS, etc.). These files typically include:

- README: General introduction.
- Release Notes: Information about changes from the previous release of FreeBSD.
- Hardware Notes: A list of hardware devices known to work with FreeBSD.
- Installation Instructions: A brief guide to installing FreeBSD.
- Errata: Late-breaking news, including corrections, security advisories, and potential problems found after each release.

Of the files listed above, the release notes, hardware notes, and installation instructions are customized for each architecture supported by FreeBSD.

37.1 RELEASE versions of FreeBSD

The release documentation for each -RELEASE version of FreeBSD (for example, &rel.current;-RELEASE) can be found on the releases page of the FreeBSD Web site, as well as its mirrors.

These files (usually in both HTML and text forms) can be found in the top-level directory of each distribution (whether on CD-ROM, an FTP site, or the install floppy disks).

37.2 Snapshot versions of FreeBSD

The release documentation files for snapshots can generally be found in the top-level directory of each snapshot.

37.3 Documentation for -CURRENT and -STABLE

Automatically-generated HTML versions of the release documentation for FreeBSD -CURRENT and FreeBSD -STABLE are available on the FreeBSD Web site. These documents are continually changing; the versions on the Web site are rebuilt at the same time that the rest of the Web site is updated.

37.3.1 FreeBSD -CURRENT Release Documentation

- README
- Release Notes
- Hardware Notes
- Errata

37.3.2 FreeBSD 10-STABLE Release Documentation

- README
- Release Notes
- Hardware Notes
- Errata

37.3.3 FreeBSD 9-STABLE Release Documentation

- README
- Release Notes
- Hardware Notes
- Errata

37.4 Other Sites

Single-file HTML, PDF, and text renderings of the release documentation for FreeBSD -CURRENT, -STABLE, and recent -RELEASE versions can be found at the [Release Documentation Snapshot Site](#). The renderings on this page are updated at irregular, but frequent intervals.

&title;

]>

You are being redirected to [The Bugzilla bug submission form](#)

&title;

|>

39.1 What Are Snapshots?

As part of an ongoing effort to improve the overall release process *before* a release actually slips out the door with problems, we are now periodically producing interim test releases called *snapshots*. These snapshots will be very similar to full releases, except that they might not include all the bits included in a full release (such as packages and updated documentation).

39.2 Getting Snapshots

The latest snapshots made available can be found on the FreeBSD FTP mirrors at the address `&url.snapshots;`. They can also be found in the same directory on other FTP mirror sites.

Please note that sometimes the snapshots available to download may be somewhat outdated.

Currently the snapshots of `&rel.head.major;-CURRENT`, `&rel.current.major;-STABLE`, and `&rel2.current.major;-STABLE`, are available in directories whose URLs have the format `&url.snapshots;/<target>/<target_arch>/` where `<target>` and `<target_arch>` represent the architecture for which the snapshot was built. For each supported platform, the snapshot includes ISO images of the `bootonly`, `release`, and `memstick` images.

39.3 Things You Might Want to Know

In particular, before getting and installing a snapshot release, be aware of following:

- The snapshots are primarily for testing purposes and not fully tested compared to the releases. They may include experimental or degraded features that can corrupt your existing system.
- The major release number will not be changed in the main distribution for each snapshot. It will *only* be changed on the installation medium so that you know when the snapshot was made. These are *not* releases, these are *snapshots*, and it is important that this distinction be preserved. Although people can and will, of course, refer to snapshots by date in email, do not confuse them.
- Snapshots might not include package sets, but will generally include a ports tree.

- Finally, we will not necessarily update the documentation. For example, `README` may still refer to a previous release. This is because that is much less important than getting the real bug fixes and new features out for testing. Please do not send a bug report about version numbers.

Your feedback on these snapshots is greatly welcome. They are not just for our benefit - those who are coming to rely on &os; for mission critical applications should welcome a chance to get at more updated bits in a structured fashion. You can also use these snapshots as tangible evidence that your feedback is getting incorporated and that you (hopefully) will not have any unpleasant surprises in the next release. On the other hand, if you do send us hate mail next release and it turns out that you never even tried the snapshots, well, it cuts both ways!

&title;

]>

FreeBSD has a wide variety of community and commercial support options available for users. The Community section of this website details the support options available to users from the FreeBSD community, including a number of mailing lists.

Commercial support is also available from one of the many vendors offering commercial products, services, and/or consulting for FreeBSD.

40.1 Problem Reporting

Found a bug in FreeBSD? Noticed a mistake in the documentation? If it has not already been reported, please let us know.

To find out what has already been reported, it is possible to [search the problem report database](#) for specific category, assignee, originator, text, and so forth or [browse open reports by category](#).

If the bug has not yet been reported, please read over our problem reporting guidelines and then report the problem using our [problem report form](#).

&title;

|>

41.1 Choosing an Architecture

Modern PCs use the amd64 architecture, including those with Intel® branded processors. Computers with more than 3 GB of memory should use amd64. If the computer is an older, 32-bit only model, use i386. For embedded devices and single-board computers (SBC) such as the Raspberry Pi, Beagle Bone Black, Panda Board, and Zed Board, use the armv6 SD card image which supports ARMv6 and ARMv7 processors.

41.2 Choosing an Image

The &os; installer can be downloaded in a number of different formats including CD (disc1), DVD (dvd1), and Network Install (bootonly) sized ISO Disc Images, as well as regular and mini USB memory stick images. Later versions of &os; are also offered as prebuilt expandable Virtual Machine images, and as SD Cards for embedded platforms.

41.3 &os; Deployment Statistics

While &os; does not gather deployment statistics, having statistical information available is essential. Please consider installing the [sysutils/bsdstats](#) package, which collects hardware and software statistics, helping developers understand how to best focus their efforts. The information collected is available at the bsdstats.org website.

41.4 &os; &rel.current;-RELEASE

Installer Images	Virtual Machine Images	Documentation
<ul style="list-style-type: none"> • ‘amd64 <&url.rel;/amd64/amd64/ISO-IMAGES/&rel .current;/>‘__ - i386 - armv6 - ia64 - powerpc - powerpc64 - sparc64 	<ul style="list-style-type: none"> • ‘amd64 <&url.rel;/VM-IMAGES/&rel.current;-RELEASE/amd64/Latest/>‘__ - i386 	<ul style="list-style-type: none"> • ‘Released <&base;/releases/index.html#current>‘__: &rel.current.date; • ‘Release Notes <&u.rel.notes;>‘__ - ‘Hardware Compatibility List <&u.rel.hardware>;>‘__ - ‘Installation Instructions <&u.rel.installation;>‘__ - Errata

41.5 &os; &rel2.current;-RELEASE

Installer Images	Virtual Machine Images	Documentation
<ul style="list-style-type: none"> • ‘amd64 <&url.rel;/amd64/amd64/ISO-IMAGES/&rel 2.current;/>‘__ - i386 - ia64 - powerpc - powerpc64 _ - sparc64 	<ul style="list-style-type: none"> • ‘amd64 <&url.rel;/VM-IMAGES/&rel2.current;-RELEASE/amd64/Latest/>‘__ - i386 	<ul style="list-style-type: none"> • ‘Released <&base;/releases/index.html#current>‘__: &rel2.current.date; • ‘Release Notes <&u.rel2.notes;>‘__ - ‘Hardware Compatibility List <&u.rel2.hardwar e;>‘__ - ‘Installation Instructions <&u.rel2.installation;>‘__ - Errata

41.6 &os; &rel3.current;-RELEASE

Installer Images	Documentation
<ul style="list-style-type: none"> • ‘amd64 <&url.rel;/amd64/ISO-IMAGE S/&rel3.current;/>‘__ - i386 - pc98 	<ul style="list-style-type: none"> • ‘Released <&base;/releases/index.html#current>‘__: &rel3.current.date; • Release Notes • Hardware Compatibility List • ‘Installation Instructions <&u.rel3.installation;>‘__ - Errata

β.desc;

41.7 Development Snapshots

If you are interested in a purely experimental **snapshot** release of &os;-CURRENT (AKA &rel.head;-CURRENT), aimed at developers and bleeding-edge testers only, then please see the &os; Snapshot Releases page. For more information about past, present and future releases in general, please visit the release information page.

41.8 &os; &rel.head;-CURRENT

Installer Images	Virtual Machine Images	Documentation
<ul style="list-style-type: none"> • <code><url.snapshot>/amd64/amd64/ISO-IMAGES/<rel.head>/'__ - i386 - armv6 - powerpc - powerpc64 <url.snapshot>/powerpc/powerpc64/ISO-IMAGES/<rel.head>/'__ - sparc64</code> 	<ul style="list-style-type: none"> • <code><url.snapshot>/VM-IMAGES/<rel.head>-CURRENT/amd64/Latest/>'__ - i386</code> 	<ul style="list-style-type: none"> • <code>Release <base>/relnotes/CURRENT/relnotes/article.html>'__</code>

41.9 &os; &rel.current;-STABLE

Installer Images	Virtual Machine Images	Documentation
<ul style="list-style-type: none"> • <code><url.snapshot>/amd64/amd64/ISO-IMAGES/<rel.current>/'__ - i386 - ia64 - armv6 - powerpc - powerpc64 - sparc64</code> 	<ul style="list-style-type: none"> • <code><url.snapshot>/VM-IMAGES/<rel.current>-STABLE/amd64/Latest/>'__ - i386</code> 	<ul style="list-style-type: none"> • <code>Release <base>/relnotes/<rel.current>-STABLE/relnotes/article.html>'__</code>

If you plan on getting &os; via HTTP or FTP, please check the listing of **mirror sites** in the handbook to see if there is a site closer to you.

41.9.1 Install &os;

There are many options for installing &os;, including installation from CD-ROM, DVD, USB Memory Stick or even directly using anonymous FTP, HTTP, or NFS. Depending on the &os; version you want to install, please read through the &os; 9.X/10.X installation guide or the &os; 8.X installation guide before downloading the entire &os; distribution.

41.9.2 Purchase &os; Media

&os; can be acquired on CD-ROM or DVD from [FreeBSD Mall](#), or one of the other CD-ROM and DVD Publishers.

41.9.3 Past Releases

For downloading past releases, please visit the [FTP archive](#).

41.9.4 &os;-derived Operating System Distributions

&os; is widely used as a building block for other commercial and open-source operating systems. The projects below are widely used and of particular interest to &os; users.

- [FreeNAS](#) is an open source storage platform based on &os; and supports sharing across Windows, Apple, and UNIX-like systems.
- [PC-BSD](#) is a &os; derivative with a graphical installer and impressive desktop tools aimed at ease of use for the casual computer user.
- [pfSense](#) is a free, open source customized distribution of &os; tailored for use as a firewall and router.

41.9.5 Applications and Utility Software

The Ports Collection

The &os; Ports Collection is a diverse collection of utility and application software that has been ported to &os;.

- [&os; Ports Collection](#)
- Web interface to the Ports Collection
- [FreshPorts](#) - a more advanced web interface to the Ports Collection

For information about how *you* can contribute *your* favorite piece of software to the Ports Collection, have a look at ‘*The Porter’s Handbook*’ <[base;/doc/en_US.ISO8859-1/books/porters-handbook/index.html](#)> ‘___’ and the article ‘*Contributing to &os;*’ <[base;/doc/en_US.ISO8859-1/articles/contributing/index.html](#)> ‘___’.

Why you should use a BSD style license for your Open Source Project

Author BruceMontague

42.1 Introduction

This document makes a case for using a BSD style license for software and data; specifically it recommends using a BSD style license in place of the GPL. It can also be read as a BSD versus GPL Open Source License introduction and summary.

42.2 Very Brief Open Source History

Long before the term “Open Source” was used, software was developed by loose associations of programmers and freely exchanged. Starting in the early 1950’s, organizations such as [SHARE](#) and [DECUS](#) developed much of the software that computer hardware companies bundled with their hardware offerings. At that time computer companies were in the hardware business; anything that reduced software cost and made more programs available made the hardware companies more competitive.

This model changed in the 1960’s. In 1965 ADR developed the first licensed software product independent of a hardware company. ADR was competing against a free IBM package originally developed by IBM customers. ADR patented their software in 1968. To stop sharing of their program, they provided it under an equipment lease in which payment was spread over the lifetime of the product. ADR thus retained ownership and could control resale and reuse.

In 1969 the US Department of Justice charged IBM with destroying businesses by bundling free software with IBM hardware. As a result of this suit, IBM unbundled its software; that is, software became independent products separate from hardware.

In 1968 Informatics introduced the first commercial killer-app and rapidly established the concept of the software product, the software company, and very high rates of return. Informatics developed the perpetual license which is now standard throughout the computer industry, wherein ownership is never transferred to the customer.

42.3 Unix from a BSD Licensing Perspective

AT&T, who owned the original Unix implementation, was a publicly regulated monopoly tied up in anti-trust court; it was legally unable to sell a product into the software market. It was, however, able to provide it to academic institutions for the price of media.

Universities rapidly adopted Unix after an OS conference publicized its availability. It was extremely helpful that Unix ran on the PDP-11, a very affordable 16-bit computer, and was coded in a high-level language that was demonstrably good for systems programming. The DEC PDP-11 had, in effect, an open hardware interface designed to make it easy for customers to write their own OS, which was common. As DEC founder Ken Olsen famously proclaimed, “software comes from heaven when you have good hardware”.

Unix author Ken Thompson returned to his alma mater, University of California Berkeley (UCB), in 1975 and taught the kernel line-by-line. This ultimately resulted in an evolving system known as BSD (Berkeley Standard Distribution). UCB converted Unix to 32-bits, added virtual memory, and implemented the version of the TCP/IP stack upon which the Internet was essentially built. UCB made BSD available for the cost of media, under what became known as “the BSD license”. A customer purchased Unix from AT&T and then ordered a BSD tape from UCB.

In the mid-1980s a government anti-trust case against ATT ended with the break-up of ATT. ATT still owned Unix and was now able to sell it. ATT embarked on an aggressive licensing effort and most commercial Unices of the day became ATT-derived.

In the early 1990’s ATT sued UCB over license violations related to BSD. UCB discovered that ATT had incorporated, without acknowledgment or payment, many improvements due to BSD into ATT’s products, and a lengthy court case, primarily between ATT and UCB, ensued. During this period some UCB programmers embarked on a project to rewrite any ATT code associated with BSD. This project resulted in a system called BSD 4.4-lite (lite because it was not a complete system; it lacked 6 key ATT files).

A lengthy series of articles published slightly later in Dr. Dobbs magazine described a BSD-derived 386 PC version of Unix, with BSD-licensed replacement files for the 6 missing 4.4 lite files. This system, named 386BSD, was due to ex-UCB programmer William Jolitz. It became the original basis of all the PC BSDs in use today.

In the mid 1990s, Novell purchased ATT’s Unix rights and a (then secret) agreement was reached to terminate the lawsuit. UCB soon terminated its support for BSD.

42.4 The Current State of FreeBSD and BSD Licenses

The so-called [new BSD license](#) applied to FreeBSD within the last few years is effectively a statement that you can do anything with the program or its source, but you do not have any warranty and none of the authors has any liability (basically, you cannot sue anybody). This new BSD license is intended to encourage product commercialization. Any BSD code can be sold or included in proprietary products without any restrictions on the availability of your code or your future behavior.

Do not confuse the new BSD license with “public domain”. While an item in the public domain is also free for all to use, it has no owner.

42.5 The origins of the GPL

While the future of Unix had been so muddled in the late 1980s and early 1990s, the GPL, another development with important licensing considerations, reached fruition.

Richard Stallman, the developer of Emacs, was a member of the staff at MIT when his lab switched from home-grown to proprietary systems. Stallman became upset when he found that he could not legally add minor improvements to the system. (Many of Stallman’s co-workers had left to form two companies based on software developed at MIT and licensed by MIT; there appears to have been disagreement over access to the source code for this software). Stallman devised an alternative to the commercial software license and called it the GPL, or “GNU Public License”. He also started a non-profit foundation, the [Free Software Foundation](#) (FSF), which intended to develop an entire operating system, including all associated software, that would not be subject to proprietary licensing. This system was called GNU, for “GNU is Not Unix”.

The GPL was designed to be the antithesis of the standard proprietary license. To this end, any modifications that were made to a GPL program were required to be given back to the GPL community (by requiring that the source of the program be available to the user) and any program that used or linked to GPL code was required to be under the GPL. The GPL was intended to keep software from becoming proprietary. As the last paragraph of the GPL states:

“This General Public License does not permit incorporating your program into proprietary programs.”[1]

The [GPL](#) is a complex license so here are some rules of thumb when using the GPL:

- you can charge as much as you want for distributing, supporting, or documenting the software, but you cannot sell the software itself.
- the rule-of-thumb states that if GPL source is required for a program to compile, the program must be under the GPL. Linking statically to a GPL library requires a program to be under the GPL.
- the GPL requires that any patents associated with GPLed software must be licensed for everyone’s free use.
- simply aggregating software together, as when multiple programs are put on one disk, does not count as including GPLed programs in non-GPLed programs.
- output of a program does not count as a derivative work. This enables the gcc compiler to be used in commercial environments without legal problems.
- since the Linux kernel is under the GPL, any code statically linked with the Linux kernel must be GPLed. This requirement can be circumvented by dynamically linking loadable kernel modules. This permits companies to distribute binary drivers, but often has the disadvantage that they will only work for particular versions of the Linux kernel.

Due in part to its complexity, in many parts of the world today the legalities of the GPL are being ignored in regard to Linux and related software. The long-term ramifications of this are unclear.

42.6 The origins of Linux and the LGPL

While the commercial Unix wars raged, the Linux kernel was developed as a PC Unix clone. Linus Torvalds credits the existence of the GNU C compiler and the associated GNU tools for the existence of Linux. He put the Linux kernel under the GPL.

Remember that the GPL requires anything that statically links to any code under the GPL also be placed under the GPL. The source for this code must thus be made available to the user of the program. Dynamic linking, however, is not considered a violation of the GPL. Pressure to put proprietary applications on Linux became overwhelming. Such applications often must link with system libraries. This resulted in a modified version of the GPL called the [LGPL](#) (“Library”, since renamed to “Lesser”, GPL). The LGPL allows proprietary code to be linked to the GNU C library, glibc. You do not have to release the source to code which has been dynamically linked to an LGPLed library.

If you statically link an application with glibc, such as is often required in embedded systems, you cannot keep your application proprietary, that is, the source must be released. Both the GPL and LGPL require any modifications to the code directly under the license to be released.

42.7 Open Source licenses and the Orphaning Problem

One of the serious problems associated with proprietary software is known as “orphaning”. This occurs when a single business failure or change in a product strategy causes a huge pyramid of dependent systems and companies to fail for reasons beyond their control. Decades of experience have shown that the momentary size or success of a software supplier is no guarantee that their software will remain available, as current market conditions and strategies can change rapidly.

The GPL attempts to prevent orphaning by severing the link to proprietary intellectual property.

A BSD license gives a small company the equivalent of software-in-escrow without any legal complications or costs. If a BSD-licensed program becomes orphaned, a company can simply take over, in a proprietary manner, the program on which they are dependent. An even better situation occurs when a BSD code-base is maintained by a small informal consortium, since the development process is not dependent on the survival of a single company or product line. The survivability of the development team when they are mentally in the zone is much more important than simple physical availability of the source code.

42.8 What a license cannot do

No license can guarantee future software availability. Although a copyright holder can traditionally change the terms of a copyright at anytime, the presumption in the BSD community is that such an attempt simply causes the source to fork.

The GPL explicitly disallows revoking the license. It has occurred, however, that a company (Mattel) purchased a GPL copyright (cphack), revoked the entire copyright, went to court, and prevailed [2]. That is, they legally revoked the entire distribution and all derivative works based on the copyright. Whether this could happen with a larger and more dispersed distribution is an open question; there is also some confusion regarding whether the software was really under the GPL.

In another example, Red Hat purchased Cygnus, an engineering company that had taken over development of the FSF compiler tools. Cygnus was able to do so because they had developed a business model in which they sold support for GNU software. This enabled them to employ some 50 engineers and drive the direction of the programs by contributing the preponderance of modifications. As Donald Rosenberg states “projects using licenses like the GPL...live under constant threat of having someone take over the project by producing a better version of the code and doing it faster than the original owners.” [3]

42.9 GPL Advantages and Disadvantages

A common reason to use the GPL is when modifying or extending the gcc compiler. This is particularly apt when working with one-off specialty CPUs in environments where all software costs are likely to be considered overhead, with minimal expectations that others will use the resulting compiler.

The GPL is also attractive to small companies selling CDs in an environment where “buy-low, sell-high” may still give the end-user a very inexpensive product. It is also attractive to companies that expect to survive by providing various forms of technical support, including documentation, for the GPLed intellectual property world.

A less publicized and unintended use of the GPL is that it is very favorable to large companies that want to undercut software companies. In other words, the GPL is well suited for use as a marketing weapon, potentially reducing overall economic benefit and contributing to monopolistic behavior.

The GPL can present a real problem for those wishing to commercialize and profit from software. For example, the GPL adds to the difficulty a graduate student will have in directly forming a company to commercialize his research results, or the difficulty a student will have in joining a company on the assumption that a promising research project will be commercialized.

For those who must work with statically-linked implementations of multiple software standards, the GPL is often a poor license, because it precludes using proprietary implementations of the standards. The GPL thus minimizes the number of programs that can be built using a GPLed standard. The GPL was intended to not provide a mechanism to develop a standard on which one engineers proprietary products. (This does not apply to Linux applications because they do not statically link, rather they use a trap-based API.)

The GPL attempts to make programmers contribute to an evolving suite of programs, then to compete in the distribution and support of this suite. This situation is unrealistic for many required core system standards, which may be applied in widely varying environments which require commercial customization or integration with legacy standards under

existing (non-GPL) licenses. Real-time systems are often statically linked, so the GPL and LGPL are definitely considered potential problems by many embedded systems companies.

The GPL is an attempt to keep efforts, regardless of demand, at the research and development stages. This maximizes the benefits to researchers and developers, at an unknown cost to those who would benefit from wider distribution.

The GPL was designed to keep research results from transitioning to proprietary products. This step is often assumed to be the last step in the traditional technology transfer pipeline and it is usually difficult enough under the best of circumstances; the GPL was intended to make it impossible.

42.10 BSD Advantages

A BSD style license is a good choice for long duration research or other projects that need a development environment that:

- has near zero cost
- will evolve over a long period of time
- permits anyone to retain the option of commercializing final results with minimal legal issues.

This final consideration may often be the dominant one, as it was when the Apache project decided upon its license:

“This type of license is ideal for promoting the use of a reference body of code that implements a protocol for common service. This is another reason why we choose it for the Apache group - many of us wanted to see HTTP survive and become a true multiparty standard, and would not have minded in the slightest if Microsoft or Netscape choose to incorporate our HTTP engine or any other component of our code into their products, if it helped further the goal of keeping HTTP common... All this means that, strategically speaking, the project needs to maintain sufficient momentum, and that participants realize greater value by contributing their code to the project, even code that would have had value if kept proprietary.”

Developers tend to find the BSD license attractive as it keeps legal issues out of the way and lets them do whatever they want with the code. In contrast, those who expect primarily to use a system rather than program it, or expect others to evolve the code, or who do not expect to make a living from their work associated with the system (such as government employees), find the GPL attractive, because it forces code developed by others to be given to them and keeps their employer from retaining copyright and thus potentially “burying” or orphaning the software. If you want to force your competitors to help you, the GPL is attractive.

A BSD license is not simply a gift. The question “why should we help our competitors or let them steal our work?” comes up often in relation to a BSD license. Under a BSD license, if one company came to dominate a product niche that others considered strategic, the other companies can, with minimal effort, form a mini-consortium aimed at reestablishing parity by contributing to a competitive BSD variant that increases market competition and fairness. This permits each company to believe that it will be able to profit from some advantage it can provide, while also contributing to economic flexibility and efficiency. The more rapidly and easily the cooperating members can do this, the more successful they will be. A BSD license is essentially a minimally complicated license that enables such behavior.

A key effect of the GPL, making a complete and competitive Open Source system widely available at cost of media, is a reasonable goal. A BSD style license, in conjunction with ad-hoc-consortiums of individuals, can achieve this goal without destroying the economic assumptions built around the deployment-end of the technology transfer pipeline.

42.11 Specific Recommendations for using a BSD license

- The BSD license is preferable for transferring research results in a way that will widely be deployed and most benefit an economy. As such, research funding agencies, such as the NSF, ONR and DARPA, should encourage in the earliest phases of funded research projects, the adoption of BSD style licenses for software, data, results,

and open hardware. They should also encourage formation of standards based around implemented Open Source systems and ongoing Open Source projects.

- Government policy should minimize the costs and difficulties in moving from research to deployment. When possible, grants should require results to be available under a commercialization friendly BSD style license.
- In many cases, the long-term results of a BSD style license more accurately reflect the goals proclaimed in the research charter of universities than what occurs when results are copyrighted or patented and subject to proprietary university licensing. Anecdotal evidence exists that universities are financially better rewarded in the long run by releasing research results and then appealing to donations from commercially successful alumni.
- Companies have long recognized that the creation of de facto standards is a key marketing technique. The BSD license serves this role well, if a company really has a unique advantage in evolving the system. The license is legally attractive to the widest audience while the company's expertise ensures their control. There are times when the GPL may be the appropriate vehicle for an attempt to create such a standard, especially when attempting to undermine or co-opt others. The GPL, however, penalizes the evolution of that standard, because it promotes a suite rather than a commercially applicable standard. Use of such a suite constantly raises commercialization and legal issues. It may not be possible to mix standards when some are under the GPL and others are not. A true technical standard should not mandate exclusion of other standards for non-technical reasons.
- Companies interested in promoting an evolving standard, which can become the core of other companies' commercial products, should be wary of the GPL. Regardless of the license used, the resulting software will usually devolve to whoever actually makes the majority of the engineering changes and most understands the state of the system. The GPL simply adds additional legal friction to the result.
- Large companies, in which Open Source code is developed, should be aware that programmers appreciate Open Source because it leaves the software available to the employee when they change employers. Some companies encourage this behavior as an employment perk, especially when the software involved is not directly strategic. It is, in effect, a front-loaded retirement benefit with potential lost opportunity costs but no direct costs. Encouraging employees to work for peer acclaim outside the company is a cheap portable benefit a company can sometimes provide with near zero downside.
- Small companies with software projects vulnerable to orphaning should attempt to use the BSD license when possible. Companies of all sizes should consider forming such Open Source projects when it is to their mutual advantage to maintain the minimal legal and organization overheads associated with a true BSD-style Open Source project.
- Non-profits should participate in Open Source projects when possible. To minimize software engineering problems, such as mixing code under different licenses, BSD-style licenses should be encouraged. Being leery of the GPL should particularly be the case with non-profits that interact with the developing world. In some locales where application of law becomes a costly exercise, the simplicity of the new BSD license, as compared to the GPL, may be of considerable advantage.

42.12 Conclusion

In contrast to the GPL, which is designed to prevent the proprietary commercialization of Open Source code, the BSD license places minimal restrictions on future behavior. This allows BSD code to remain Open Source or become integrated into commercial solutions, as a project's or company's needs change. In other words, the BSD license does not become a legal time-bomb at any point in the development process.

In addition, since the BSD license does not come with the legal complexity of the GPL or LGPL licenses, it allows developers and companies to spend their time creating and promoting good code rather than worrying if that code violates licensing.

42.13 Addenda

[1] <http://www.gnu.org/licenses/gpl.html>

[2] <http://archives.cnn.com/2000/TECH/computing/03/28/cyberpatrol.mirrors/>

[3] Open Source: the Unauthorized White Papers, Donald K. Rosenberg, IDG Books, 2000. Quotes are from page 114, ``Effects of the GNU GPL''.

[4] In the "What License to Use?" section of
<http://www.oreilly.com/catalog/opensources/book/brian.html>

This whitepaper is a condensation of an original work available at
http://alumni.cse.ucsc.edu/~brucem/open_source_license.htm

Abstract

Author JosephKoshy

43.1 Introduction

FreeBSD today is well-known as a high-performance server operating system. It is deployed on millions of web servers and internet-facing hosts worldwide. FreeBSD code also forms an integral part of many products, ranging from appliances such as network routers, firewalls, and storage devices, to personal computers. Portions of FreeBSD have also been used in commercial shrink-wrapped software (see ?).

In this article we look at the FreeBSD project as a software engineering resource—as a collection of building blocks and processes which you can use to build products.

While FreeBSD’s source is distributed freely to the public, to fully enjoy the benefits of the project’s work, organizations need to *collaborate* with the project. In subsequent sections of this article we discuss effective means of collaboration with the project and the pitfalls that need to be avoided while doing so.

Caveat Reader.

The author believes that the characteristics of the FreeBSD Project listed in this article were substantially true at the time the article was conceived and written (2005). However, the reader should keep in mind that the practices and processes used by open-source communities can change over time, and that the information in this article should therefore be taken as indicative rather than normative.

43.1.1 Target Audience

This document would be of interest to the following broad groups of people:

- Decision makers in product companies looking at ways to improve their product quality, reduce their time to market and lower engineering costs in the long term.
- Technology consultants looking for best-practices in leveraging “open-source”.
- Industry observers interested in understanding the dynamics of open-source projects.
- Software developers seeking to use FreeBSD and looking for ways to contribute back.

43.1.2 Article Goals

After reading this article you should have:

- An understanding of the goals of the FreeBSD Project and its organizational structure.
- An overview of the available technology in the project.
- An understanding of its development model and release engineering processes.
- An understanding of how conventional corporate software development processes differ from that used in the FreeBSD project.
- Awareness of the communication channels used by the project and the level of transparency you can expect.
- Awareness of optimal ways of working with the project—how best to reduce engineering costs, improve time to market, manage security vulnerabilities, and preserve future compatibility with your product as the FreeBSD project evolves.

43.1.3 Article Structure

The rest of the article is structured as follows:

- ? introduces the FreeBSD project, explores its organizational structure, key technologies and release engineering processes.
- ? describes ways to collaborate with the FreeBSD project. It examines common pitfalls encountered by corporates working with voluntary projects like FreeBSD.
- ? concludes.

43.2 FreeBSD as a set of building blocks

FreeBSD makes an excellent foundation on which to build products:

- FreeBSD source code is distributed under a liberal BSD license facilitating its adoption in commercial products Mon2005 with minimum hassle.
- The FreeBSD project has excellent engineering practices that can be leveraged.
- The project offers exceptional transparency into its workings, allowing organizations using its code to plan effectively for the future.
- The culture of the FreeBSD project, carried over from the Computer Science Research Group at The University of California, Berkeley McKu1999-1, fosters high-quality work. Some features in FreeBSD define the state of the art.

GoldGab2005 examines the business reasons for using open-source in greater detail. For organizations, the benefits of using FreeBSD components in their products include a shorter time to market, lower development costs and lower development risks.

43.2.1 Building with FreeBSD

Here are a few ways organizations have used FreeBSD:

- As an upstream source for tested code for libraries and utilities.
By being “downstream” of the project, organizations leverage the new features, bug fixes and testing that the upstream code receives.

- As an embedded OS (for example, for an OEM router and firewall device). In this model, organizations use a customized FreeBSD kernel and application program set along with a proprietary management layer for their device. OEMs benefit from new hardware support being added by the FreeBSD project upstream, and from the testing that the base system receives.

FreeBSD ships with a self-hosting development environment that allows easy creation of such configurations.

- As a Unix compatible environment for the management functions of high-end storage and networking devices, running on a separate processor “blade”.

FreeBSD provides the tools for creating dedicated OS and application program images. Its implementation of a BSD unix API is mature and tested. FreeBSD can also provide a stable cross-development environment for the other components of the high-end device.

- As a vehicle to get widespread testing and support from a worldwide team of developers for non-critical “intellectual property”.

In this model, organizations contribute useful infrastructural frameworks to the FreeBSD project (for example, see MAN.NETGRAPH.3). The widespread exposure that the code gets helps to quickly identify performance issues and bugs. The involvement of top-notch developers also leads to useful extensions to the infrastructure that the contributing organization also benefits from.

- As a development environment supporting cross-development for embedded OSes like RTEMS and eCOS.

There are many full-fledged development environments in the OS.NUMPORTS-strong collection of applications ported and packaged with FreeBSD.

- As a way to support a Unix-like API in an otherwise proprietary OS, increasing its palatability for application developers.

Here parts of FreeBSD’s kernel and application programs are “ported” to run alongside other tasks in the proprietary OS. The availability of a stable and well tested Unix API implementation can reduce the effort needed to port popular applications to the proprietary OS. As FreeBSD ships with high-quality documentation for its internals and has effective vulnerability management and release engineering processes, the costs of keeping upto-date are kept low.

43.2.2 Technologies

There are a large number of technologies supported by the FreeBSD project. A selection of these are listed below:

- A complete system that can cross-host itself for many architectures:
- Support for the following technologies, protocols and standards: ATA, ATAPI, ATM, Bluetooth, CAM, CardBus, DHCP, DNS, EISA, Ethernet, FDDI, Fibre Channel, GPIB, IEEE 1394, IPv4, IPv6, IPSEC, IPX, ISDN, MAC, NIS, NFS, OpenSSH, OPIE, PAM, PCI, PCMCIA, POSIX, PnP, RAID, RPC, SATA, SCSI, SMB, TCP, USB, VESA, VLAN, VLB, WebNFS.
- A modular symmetric multiprocessing capable kernel, with loadable kernel modules and a flexible and easy to use configuration system.
- Support for emulation of Linux and SVR4 binaries at near machine speeds. Support for binary Windows (NDIS) network drivers.
- Libraries for many programming tasks: archivers, FTP and HTTP support, thread support, in addition to a full POSIX like programming environment.
- Advanced security features: Mandatory Access Control (MAN.MAC.9), jails (MAN.JAIL.2), ACLs, and in-kernel cryptographic device support.
- Advanced networking features: firewall-ing, QoS management, high-performance TCP/IP networking with support for many advanced features.

FreeBSD’s in-kernel Netgraph (MAN.NETGRAPH.4) framework allows kernel networking modules to be connected together in flexible ways.

- Support for advanced storage technologies: Fibre Channel, SCSI, software and hardware RAID, ATA and SATA.

FreeBSD supports a number of filesystems, and its native UFS2 filesystem supports soft updates, snapshots and very large filesystem sizes (16TB per filesystem) McKu1999.

FreeBSD’s in-kernel GEOM (MAN.GEOM.4) framework allows kernel storage modules to be composed in flexible ways.

- Over OS.NUMPORTS ported applications, both commercial and open-source, managed via the FreeBSD ports collection.

43.2.3 Organizational Structure

FreeBSD’s organizational structure is non-hierarchical.

There are essentially two kinds of contributors to FreeBSD, general users of FreeBSD, and developers with write access (known as committers in the jargon) to the source base.

There are many thousands of contributors in the first group; the vast majority of contributions to FreeBSD come from individuals in this group. Commit rights (write access) to the repository are granted to individuals who contribute consistently to the project. Commit rights come with additional responsibilities, and new committers are assigned mentors to help them learn the ropes.

Fig. 43.1: FreeBSD Organization

Conflict resolution is performed by a nine member “Core Team” that is elected from the group of committers.

FreeBSD does not have “corporate” committers. Individual committers are required to take responsibility for the changes they introduce to the code. The FreeBSD Committer’s guide ComGuide documents the rules and responsibilities for committers.

FreeBSD’s project model is examined in detail in Nik2005.

43.2.4 FreeBSD Release Engineering Processes

FreeBSD’s release engineering processes play a major role in ensuring that its released versions are of a high quality. At any point of time, FreeBSD’s volunteers support multiple code lines (?):

- New features and disruptive code enters on the development branch, also known as the -CURRENT branch.
- -STABLE branches are code lines that are branched from HEAD at regular intervals. Only tested code is allowed onto a -STABLE branch. New features are allowed once they have been tested and stabilized in the -CURRENT branch.
- -RELEASE branches are maintained by the FreeBSD security team. Only bug fixes for critical issues are permitted onto -RELEASE branches.

Fig. 43.2: FreeBSD Release Branches

Code lines are kept alive for as long as there is user and developer interest in them.

Machine architectures are grouped into “tiers”; Tier 1 architectures are fully supported by the project’s release engineering and security teams, Tier 2 architectures are supported on a best effort basis, and experimental architectures comprise Tier 3. The list of supported architectures is part of the FreeBSD documentation collection.

The release engineering team publishes a road map for future releases of FreeBSD on the project's web site. The dates laid down in the road map are not deadlines; FreeBSD is released when its code and documentation are ready.

FreeBSD's release engineering processes are described in RelEngDoc.

43.3 Collaborating with FreeBSD

Open-source projects like FreeBSD offer finished code of a very high quality Cov2005. Previous studies have examined the effect of source code availability on software development Com2004.

While access to quality source code can reduce the cost of initial development, in the long-term the costs of managing change begin to dominate. As computing environments change over the years and new security vulnerabilities are discovered, your product too needs to change and adapt. Using open-source code is best viewed not as a one-off activity, but as an *ongoing process*. The best projects to collaborate with are the ones that are *live*; i.e., with an active community, clear goals and a transparent working style.

- FreeBSD has an active developer community around it. At the time of writing there are many thousands of contributors from every populated continent in the world and over 300 individuals with write access to the project's source repositories.
- The goals of the FreeBSD project are Hub1994:
 - To develop a high-quality operating system for popular computer hardware, and,
 - To make our work available to all under a liberal license.
- FreeBSD enjoys an open and transparent working culture. Nearly all discussion in the project happens by email, on public mailing lists that are also archived for posterity. The project's policies are documented and maintained under revision control. Participation in the project is open to all.

43.3.1 Understanding FreeBSD culture

To be able to work effectively with the FreeBSD project, you need to understand the project's culture.

Volunteer driven projects operate under different rules than for-profit corporates. A common mistake that companies make when venturing into the open-source world is that of underplaying these differences.

Motivation.

Most contributions to FreeBSD are done voluntarily without monetary rewards entering the picture. The factors that motivate individuals are complex, ranging from altruism, to an interest in solving the kinds of problems that FreeBSD attempts to solve. In this environment, "elegance is never optional" Nor1993.

The Long Term View.

FreeBSD traces its roots back nearly twenty years to the work of the Computer Science Research Group at the University of California Berkeley.¹ A number of the original CSRG developers remain associated with the project.

The project values long-term perspectives Nor2001. A frequent acronym encountered in the project is DTTR, which stands for "Do The Right Thing".

Development Processes.

Computer programs are tools for communication: at one level programmers communicate their intentions using a precise notation to a tool (a compiler) that translates their instructions to executable code. At another level, the same notation is used for communication of intent between two programmers.

¹ FreeBSD's [source repository](#) contains a history of the project since its inception, and there are [CDROMs available](#) that contain earlier code from the CSRG.

Formal specifications and design documents are seldom used in the project. Clear and well-written code and well-written change logs (?) are used in their place. FreeBSD development happens by “rough consensus and running code” Carp1996.

```
r151864 | bde | 2005-10-29 09:34:50 -0700 (Sat, 29 Oct 2005) | 13 lines
```

Changed paths:

```
    M /head/lib/msun/src/e_rem_pio2f.c
```

Use double precision to simplify and optimize arg reduction for small and medium size args too: instead of conditionally subtracting a float 17+24, 17+17+24 or 17+17+17+24 bit approximation to $\pi/2$, always subtract a double 33+53 bit one. The float version is now closer to the double version than to old versions of itself -- it uses the same 33+53 bit approximation as the simplest cases in the double version, and where the float version had to switch to the slow general case at $|x| == 2^7\pi/2$, it now switches at $|x| == 2^{19}\pi/2$ the same as the double version.

This speeds up arg reduction by a factor of 2 for $|x|$ between $3\pi/4$ and $2^7\pi/4$, and by a factor of 7 for $|x|$ between $2^7\pi/4$ and $2^{19}\pi/4$.

Communication between programmers is enhanced by the use of a common coding standard MAN.STYLE.9.

Communication Channels.

FreeBSD’s contributors are spread across the world. Email (and to a lesser extent, IRC) is the preferred means of communication in the project.

43.3.2 Best Practices for collaborating with the FreeBSD project

We now look at a few best practices for making the best use of FreeBSD in product development.

Plan for the long term Setup processes that help in tracking the development of FreeBSD. For example:

Track FreeBSD source code.

The project makes it easy to mirror its SVN repository using svnsync. Having the complete history of the source is useful when debugging complex problems and offers valuable insight into the intentions of the original developers. Use a capable source control system that allows you to easily merge changes between the upstream FreeBSD code base and your own in-house code.

? shows a portion of an annotated listing of the file referenced by the change log in ?. The ancestry of each line of the source is clearly visible. Annotated listings showing the history of every file that is part of FreeBSD are [available on the web](#).

#REV	#WHO	#DATE	#TEXT
176410	bde	2008-02-19 07:42:46 -0800 (Tue, 19 Feb 2008)	#include <sys/cdefs.h>
176410	bde	2008-02-19 07:42:46 -0800 (Tue, 19 Feb 2008)	__FBSDID("\$FreeBSD\$");
2116	jkh	1994-08-19 02:40:01 -0700 (Fri, 19 Aug 1994)	
2116	jkh	1994-08-19 02:40:01 -0700 (Fri, 19 Aug 1994)	/* __ieee754_rem_pio2f(x,y)
8870	rgrimes	1995-05-29 22:51:47 -0700 (Mon, 29 May 1995)	*
176552	bde	2008-02-25 05:33:20 -0800 (Mon, 25 Feb 2008)	* return the remainder of x rem
176552	bde	2008-02-25 05:33:20 -0800 (Mon, 25 Feb 2008)	* use double precision for every
152535	bde	2005-11-16 18:20:04 -0800 (Wed, 16 Nov 2005)	* use __kernel_rem_pio2() for la
2116	jkh	1994-08-19 02:40:01 -0700 (Fri, 19 Aug 1994)	*/
2116	jkh	1994-08-19 02:40:01 -0700 (Fri, 19 Aug 1994)	
176465	bde	2008-02-22 07:55:14 -0800 (Fri, 22 Feb 2008)	#include <float.h>

176465	bde 2008-02-22 07:55:14 -0800 (Fri, 22 Feb 2008)
2116	jkh 1994-08-19 02:40:01 -0700 (Fri, 19 Aug 1994) #include "math.h"

Use a gatekeeper.

Appoint a gatekeeper to monitor FreeBSD development, to keep an eye out for changes that could potentially impact your products.

Report bugs upstream.

If you notice bug in the FreeBSD code that you are using, file a [bug report](#). This step helps ensure that you do not have to fix the bug the next time you take a code drop from upstream.

Leverage FreeBSD's release engineering efforts Use code from a -STABLE development branch of FreeBSD. These development branches are formally supported by FreeBSD's release engineering and security teams and comprise of tested code.

Donate code to reduce costs A major proportion of the costs associated with developing products is that of doing maintenance. By donating non-critical code to the project, you benefit by having your code see much wider exposure than it would otherwise get. This in turn leads to more bugs and security vulnerabilities being flushed out and performance anomalies being identified and fixed.

Get support effectively For products with tight deadlines, it is recommended that you hire or enter into a consulting agreement with a developer or firm with FreeBSD experience. The A.JOBS is a useful communication channel to find talent. The FreeBSD project maintains a gallery of consultants and consulting firms undertaking FreeBSD work. The [BSD Certification Group](#) offers certification for all the major BSD derived OSes.

For less critical needs, you can ask for help on the [project mailing lists](#). A useful guide to follow when asking for help is given in Ray2004.

Publicize your involvement You are not required to publicize your use of FreeBSD, but doing so helps both your effort as well as that of the project.

Letting the FreeBSD community know that your company uses FreeBSD helps improve your chances of attracting high quality talent. A large roster of support for FreeBSD also means more mind share for it among developers. This in turn yields a healthier foundation for your future.

Support FreeBSD developers Sometimes the most direct way to get a desired feature into FreeBSD is to support a developer who is already looking at a related problem. Help can range from hardware donations to direct financial assistance. In some countries, donations to the FreeBSD project enjoy tax benefits. The project has a dedicated donations liaison to assist donors. The project also maintains a web page where developers list their needs.

As a policy the FreeBSD project acknowledges all contributions received on its web site.

43.4 Conclusion

The FreeBSD project's goals are to create and give away the source code for a high-quality operating system. By working with the FreeBSD project you can reduce development costs and improve your time to market in a number of product development scenarios.

We examined the characteristics of the FreeBSD project that make it an excellent choice for being part of an organization's product strategy. We then looked at the prevailing culture of the project and examined effective ways of interacting with its developers. The article concluded with a list of best-practices that could help organizations collaborating with the project.

Carp1996 [The Architectural Principles of the Internet](#) B.Carpenter 1996

Com2004 [How is Open-Source Affecting Software Development?](#) DiomidisSpinellis ClemensSzyperski IEEE Computer Jan/Feb 2004 IEEE Computer Society

ComGuide Committer's Guide The FreeBSD Project 2005

Cov2005 [Coverity study on kernel security holes in Linux and FreeBSD](#) Coverity Inc. 2005

GoldGab2005 [Innovation Happens Elsewhere: Open Source as Business Strategy](#) RonGoldman RichardGabriel 2005 ISBN 1558608893 Morgan-Kaufmann

Hub1994 [Contributing to the FreeBSD Project](#) JordanHubbard 1994—2005 The FreeBSD Project

McKu1999 [Soft Updates: A Technique for Eliminating Most Synchronous Writes in the Fast Filesystem](#) KirkMcKusick GregoryGanger USENIX Annual Technical Conference 1999

McKu1999-1 [Twenty Years of Berkeley Unix: From AT&T-Owned to Freely Redistributable](#) Marshall KirkMcKusick [Open Sources: Voices from the Open Source Revolution](#) ISBN 1-56592-582-3 O'Reilly Inc. 1993

Mon2005 [Why you should use a BSD style license for your Open Source Project](#) BruceMontague The FreeBSD Project 2005

Nik2005 [A project model for the FreeBSD Project](#) NiklasSaers 2005 The FreeBSD Project

Nor1993 [Tutorial on Good Lisp Programming Style](#) PeterNorvig KentPitman 1993

Nor2001 [Teach Yourself Programming in Ten Years](#) PeterNorvig 2001

Ray2004 [How to ask questions the smart way](#) Eric StevenRaymond 2004

RelEngDoc FreeBSD Release Engineering MurrayStokely 2001 The FreeBSD Project

Committer's Guide

Author The OS Documentation Project

44.1 Administrative Details

<i>Login Methods</i>	MAN.SSH.1, protocol 2 only
<i>Main Shell Host</i>	freefall.FreeBSD.org
<i>“src/“ Subversion Root</i>	svn+ssh://svn.FreeBSD.org/base (see also ?).
<i>“doc/“ Subversion Root</i>	svn+ssh://svn.FreeBSD.org/doc (see also ?).
<i>“ports/“ Subversion Root</i>	svn+ssh://svn.FreeBSD.org/ports (see also ?).
<i>Internal Mailing Lists</i>	developers (technically called all-developers), doc-developers, doc-committers, ports-developers, ports-committers, src-developers, src-committers. (Each project repository has its own -developers and -committers mailing lists. Archives for these lists may be found in files /home/mail/repository-name-developers-archive and /home/mail/repository-name-committers-archive on the FreeBSD.org cluster.)
<i>Core Team monthly reports</i>	/home/core/public/monthly-reports on the FreeBSD.org cluster.
<i>Ports Management Team monthly reports</i>	/home/portmgr/public/monthly-reports on the FreeBSD.org cluster.
<i>Noteworthy “src/“ SVN Branches</i>	stable/8 (8.X-STABLE), stable/9 (9.X-STABLE), stable/10 (10.X-STABLE), head (-CURRENT)

MAN.SSH.1 is required to connect to the project hosts. For more information, see ?.

Useful links:

- OS Project Internal Pages
- OS Project Hosts
- OS Project Administrative Groups

44.2 OpenPGP Keys for OS

Cryptographic keys conforming to the OpenPGP (*Pretty Good Privacy*) standard are used by the OS project to authenticate committers. Messages carrying important information like public SSH keys can be signed with the OpenPGP key to prove that they are really from the committer. See [PGP & GPG: Email for the Practical Paranoid by Michael Lucas](#) and http://en.wikipedia.org/wiki/Pretty_Good_Privacy for more information.

44.2.1 Creating a Key

Existing keys can be used, but should be checked with `doc/head/share/pgpkeys/checkkey.sh` first.

For those who do not yet have an OpenPGP key, or need a new key to meet OS security requirements, here we show how to generate one.

Install `security/gnupg`. Enter these lines in `~/.gnupg/gpg.conf` to set minimum acceptable defaults:

```
fixed-list-mode
keyid-format 0xlong
personal-digest-preferences SHA512 SHA384 SHA256 SHA224
default-preference-list SHA512 SHA384 SHA256 SHA224 AES256 AES192 AES CAST5 BZIP2 ZLIB ZIP Uncompress
use-agent
verify-options show-uid-validity
list-options show-uid-validity
sig-notation issuer-fpr@notations.openpgp.fifthhorseman.net=%g
cert-digest-algo SHA512
```

Generate a key:

```
PROMPT.USER gpg --gen-key
gpg (GnuPG) 2.0.22; Copyright (C) 2013 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Warning: using insecure memory!
Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048
Requested keysize is 2048 bits
Please specify how long the key should be valid.
    0 = key does not expire
    <n>  = key expires in n days
    <n>w  = key expires in n weeks
    <n>m  = key expires in n months
    <n>y  = key expires in n years
Key is valid for? (0) 3y
Key expires at Wed Nov  4 17:20:20 2015 MST
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: Chucky Daemon
Email address: notreal@example.com
```

```

Comment:
You selected this USER-ID:
    "Chucky Daemon <notreal@example.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.

```

- 2048-bit keys with a three-year expiration provide adequate protection at present (2013-12). <http://danielpocock.com/rsa-key-sizes-2048-or-4096-bits> describes the situation in more detail.
- A three year key lifespan is short enough to obsolete keys weakened by advancing computer power, but long enough to reduce key management problems.
- Use your real name here, preferably matching that shown on government-issued ID to make it easier for others to verify your identity. Text that may help others identify you can be entered in the `Comment` section.

After the email address is entered, a passphrase is requested. Methods of creating a secure passphrase are contentious. Rather than suggest a single way, here are some links to sites that describe various methods: <http://world.std.com/~reinhold/diceware.html>, <http://www.iusmentis.com/security/passphrasefaq/>, <http://xkcd.com/936/>, <http://en.wikipedia.org/wiki/Passphrase>.

Protect your private key and passphrase. If either the private key or passphrase may have been compromised or disclosed, immediately notify accounts@FreeBSD.org and revoke the key.

Committing the new key is shown in ?.

44.3 Kerberos and LDAP web Password for OS Cluster

The OS cluster requires a Kerberos password to access certain services. The Kerberos password also serves as the LDAP web password, since LDAP is proxying to Kerberos in the cluster. Some of the services which require this include:

- [Bugzilla](#)
- [Jenkins](#)

To reset a Kerberos password in the OS cluster using a random password generator:

```

PROMPT.USER ssh kpasswd.freebsd.org

**Note**

This must be done from a machine outside of the OS.org cluster.

```

A Kerberos password can also be set manually by logging into freefall.FreeBSD.org and running:

```
PROMPT.USER kpasswd
```

44.4 Commit Bit Types

The OS repository has a number of components which, when combined, support the basic operating system source, documentation, third party application ports infrastructure, and various maintained utilities. When OS commit bits are allocated, the areas of the tree where the bit may be used are specified. Generally, the areas associated with a bit reflect who authorized the allocation of the commit bit. Additional areas of authority may be added at a later date: when this occurs, the committer should follow normal commit bit allocation procedures for that area of the tree, seeking approval from the appropriate entity and possibly getting a mentor for that area for some period of time.

<i>Committer Type</i>	<i>Responsible</i>	<i>Tree Components</i>
src	core@	src/, doc/ subject to appropriate review
doc	doceng@	doc/, ports/, src/ documentation
ports	portmgr@	ports/

Commit bits allocated prior to the development of the notion of areas of authority may be appropriate for use in many parts of the tree. However, common sense dictates that a committer who has not previously worked in an area of the tree seek review prior to committing, seek approval from the appropriate responsible party, and/or work with a mentor. Since the rules regarding code maintenance differ by area of the tree, this is as much for the benefit of the committer working in an area of less familiarity as it is for others working on the tree.

Committers are encouraged to seek review for their work as part of the normal development process, regardless of the area of the tree where the work is occurring.

44.4.1 Policy for Committer Activity in Other Trees

- All committers may modify `base/head/share/misc/committers-*.dot`, `base/head/usr.bin/calendar/calendars/calendar.freebsd`, and `ports/head/astro/xearth/files`.
- doc committers may commit documentation changes to `src` files, such as man pages, READMEs, fortune databases, calendar files, and comment fixes without approval from a src committer, subject to the normal care and tending of commits.
- Any committer may make changes to any other tree with an “Approved by” from a non-mentored committer with the appropriate bit.
- Committers can acquire an additional bit by the usual process of finding a mentor who will propose them to core, doceng, or portmgr, as appropriate. When approved, they will be added to ‘access’ and the normal mentoring period will ensue, which will involve a continuing of “Approved by” for some period.
- “Approved by” is only acceptable from non-mentored src committers – mentored committers can provide a “Reviewed by” but not an “Approved by”.

44.5 Subversion Primer

It is assumed that you are already familiar with the basic operation of Subversion. If not, start by reading the [Subversion Book](#).

44.5.1 Introduction

The OS source repository switched from CVS to Subversion on May 31st, 2008. The first real SVN commit is `r179447`.

The OS `doc/www` repository switched from CVS to Subversion on May 19th, 2012. The first real SVN commit is `r38821`.

The OS `ports` repository switched from CVS to Subversion on July 14th, 2012. The first real SVN commit is `r300894`.

Subversion can be installed from the OS Ports Collection by issuing these commands:

```
PROMPT.ROOT pkg install subversion
```

44.5.2 Getting Started

There are a few ways to obtain a working copy of the tree from Subversion. This section will explain them.

Direct Checkout

The first is to check out directly from the main repository. For the `src` tree, use:

```
PROMPT.USER svn checkout svn+ssh://svn.freebsd.org/base/head /usr/src
```

For the `doc` tree, use:

```
PROMPT.USER svn checkout svn+ssh://svn.freebsd.org/doc/head /usr/doc
```

For the `ports` tree, use:

```
PROMPT.USER svn checkout svn+ssh://svn.freebsd.org/ports/head /usr/ports
```

****Note****

Though the remaining examples in this document are written with the workflow of working with the `src` tree in mind, the underlying concepts are the same for working with the `doc` and the `ports` tree. Ports related Subversion operations are listed in ?.

The above command will check out a `CURRENT` source tree as `/usr/src/`, which can be any target directory on the local filesystem. Omitting the final argument of that command causes the working copy, in this case, to be named “head”, but that can be renamed safely.

`svn+ssh` means the SVN protocol tunnelled over SSH. The name of the server is `svn.freebsd.org`, `base` is the path to the repository, and `head` is the subdirectory within the repository.

If your OS login name is different from your login name on your local machine, you must either include it in the URL (for example `svn+ssh://jarjar@svn.freebsd.org/base/head`), or add an entry to your `~/.ssh/config` in the form:

```
Host svn.freebsd.org
  User jarjar
```

This is the simplest method, but it is hard to tell just yet how much load it will place on the repository.

Note

The `svn diff` does not require access to the server as SVN stores a reference copy of every file in the working copy. This, however, means that Subversion working copies are very large in size.

Checkout from a Mirror

Check out a working copy from a mirror by substituting the mirror’s URL for `svn+ssh://svn.freebsd.org/base`. This can be an official mirror or a mirror maintained by using `svnsync`.

There is a serious disadvantage to this method: every time something is to be committed, a `svn relocate` to the master repository has to be done, remembering to `svn relocate` back to the mirror after the commit. Also, since `svn relocate` only works between repositories that have the same UUID, some hacking of the local repository’s UUID has to occur before it is possible to start using it.

The hassle of a local `svnsync` mirror probably is not worth it unless the network connectivity situation or other factors demand it. If it is needed, see the end of this chapter for information on how to set one up.

RELENG_* Branches and General Layout

In `svn+ssh://svn.freebsd.org/base`, *base* refers to the source tree. Similarly, *ports* refers to the ports tree, and so on. These are separate repositories with their own change number sequences, access controls and commit mail.

For the base repository, HEAD refers to the -CURRENT tree. For example, `head/bin/ls` is what would go into `/usr/src/bin/ls` in a release. Some key locations are:

- `/head/` which corresponds to HEAD, also known as -CURRENT.
- `/stable/n` which corresponds to RELENG_n.
- `/releng/n.n` which corresponds to RELENG_n.n.
- `/release/n.n.n` which corresponds to RELENG_n.n.n_RELEASE.
- `/vendor*` is the vendor branch import work area. This directory itself does not contain branches, however its subdirectories do. This contrasts with the *stable*, *releng* and *release* directories.
- `/projects` and `/user` feature a branch work area, like in Perforce. As above, the `/user` directory does not contain branches itself.

OS Documentation Project Branches and Layout

In `svn+ssh://svn.freebsd.org/doc`, *doc* refers to the repository root of the source tree.

In general, most OS Documentation Project work will be done within the `head/` branch of the documentation source tree.

OS documentation is written and/or translated to various languages, each in a separate directory in the `head/` branch.

Each translation set contains several subdirectories for the various parts of the OS Documentation Project. A few noteworthy directories are:

- `/articles/` contains the source code for articles written by various OS contributors.
- `/books/` contains the source code for the different books, such as the OS Handbook.
- `/htdocs/` contains the source code for the OS website.

OS Ports Tree Branches and Layout

In `svn+ssh://svn.freebsd.org/ports`, *ports* refers to the repository root of the ports tree.

In general, most OS port work will be done within the `head/` branch of the ports tree which is the actual ports tree used to install software. Some other key locations are:

- `/branches/RELENG_n.n.n` which corresponds to RELENG_n.n.n is used to merge back security updates in preparation for a release.
- `/tags/RELEASE_n.n.n` which corresponds to RELEASE_n.n.n represents a release tag of the ports tree.
- `/tags/RELEASE_n_EOL` represents the end of life tag of a specific OS branch.

44.5.3 Daily Use

This section will explain how to perform common day-to-day operations with Subversion.

Help

SVN has built in help documentation. It can be accessed by typing the following command:

```
PROMPT.USER svn help
```

Additional information can be found in the [Subversion Book](#).

Checkout

As seen earlier, to check out the OS head branch:

```
PROMPT.USER svn checkout svn+ssh://svn.freebsd.org/base/head /usr/src
```

At some point, more than just HEAD will probably be useful, for instance when merging changes to stable/7. Therefore, it may be useful to have a partial checkout of the complete tree (a full checkout would be very painful).

To do this, first check out the root of the repository:

```
PROMPT.USER svn checkout --depth=immediates svn+ssh://svn.freebsd.org/base
```

This will give base with all the files it contains (at the time of writing, just ROADMAP.txt) and empty subdirectories for head, stable, vendor and so on.

Expanding the working copy is possible. Just change the depth of the various subdirectories:

```
PROMPT.USER svn up --set-depth=infinity base/head
PROMPT.USER svn up --set-depth=immediates base/release base/releng base/stable
```

The above command will pull down a full copy of head, plus empty copies of every release tag, every releng branch, and every stable branch.

If at a later date merging to 7-STABLE is required, expand the working copy:

```
PROMPT.USER svn up --set-depth=infinity base/stable/7
```

Subtrees do not have to be expanded completely. For instance, expanding only stable/7/sys and then later expand the rest of stable/7:

```
PROMPT.USER svn up --set-depth=infinity base/stable/7/sys
PROMPT.USER svn up --set-depth=infinity base/stable/7
```

Updating the tree with svn update will only update what was previously asked for (in this case, head and stable/7; it will not pull down the whole tree).

Note

Decreasing the depth of a working copy is not possible.

Anonymous Checkout

It is possible to anonymously check out the OS repository with Subversion. This will give access to a read-only tree that can be updated, but not committed back to the main repository. To do this, use the following command:

```
PROMPT.USER svn co https://svn0.us-west.FreeBSD.org/base/head /usr/src
```

Select the closest mirror and verify the mirror server certificate from the list of Subversion mirror sites.

Updating the Tree

To update a working copy to either the latest revision, or a specific revision:

```
PROMPT.USER svn update
PROMPT.USER svn update -r12345
```

Status

To view the local changes that have been made to the working copy:

```
PROMPT.USER svn status
```

To show local changes and files that are out-of-date do:

```
PROMPT.USER svn status --show-updates
```

Editing and Committing

Unlike Perforce, SVN does not need to be told in advance about file editing.

To commit all changes in the current directory and all subdirectories:

```
PROMPT.USER svn commit
```

To commit all changes in, for example, `lib/libfetch/` and `usr/bin/fetch/` in a single operation:

```
PROMPT.USER svn commit lib/libfetch usr/bin/fetch
```

There is also a commit wrapper for the ports tree to handle the properties and sanity checking your changes:

```
PROMPT.USER /usr/ports/Tools/scripts/psvn commit
```

Adding and Removing Files

Note

Before adding files, get a copy of `auto-props.txt` (there is also a [ports tree specific version](#)) and add it to `~/.subversion/config` according to the instructions in the file. If you added something before reading this, use `svn rm --keep-local` for just added files, fix your config file and re-add them again. The initial config file is created when you first run a svn command, even something as simple as `svn help`.

Files are added to a SVN repository with “`svn add`”. To add a file named *foo*, edit it, then:

```
PROMPT.USER svn add foo
```

****Note****

Most new source files should include a ```OS``` string near the start of the file. On commit, ```svn``` will expand the ```OS``` string, adding the file path, revision number, date and time of commit, and the username of the committer. Files which cannot be modified may be committed without the ```OS``` string.

Files can be removed with “`svn remove`”:


```
PROMPT.USER svn remove foo
```

Subversion does not require deleting the file before using `svn rm`, and indeed complains if that happens.

It is possible to add directories with `svn add`:

```
PROMPT.USER mkdir bar
PROMPT.USER svn add bar
```

Although `svn mkdir` makes this easier by combining the creation of the directory and the adding of it:

```
PROMPT.USER svn mkdir bar
```

Like files, directories are removed with `svn rm`. There is no separate command specifically for removing directories.

```
PROMPT.USER svn rm bar
```

Copying and Moving Files

This command creates a copy of `foo.c` named `bar.c`, with the new file also under version control:

```
PROMPT.USER svn copy foo.c bar.c
```

The example above is equivalent to:

```
PROMPT.USER cp foo.c bar.c
PROMPT.USER svn add bar.c
```

To move and rename a file:

```
PROMPT.USER svn move foo.c bar.c
```

Log and Annotate

`svn log` shows revisions and commit messages, most recent first, for files or directories. When used on a directory, all revisions that affected the directory and files within that directory are shown.

svn annotate, or equally “**svn praise**” or `svn blame`, shows the most recent revision number and who committed that revision for each line of a file.

Diffs

`svn diff` displays changes to the working copy. Diffs generated by SVN are unified and include new files by default in the diff output.

`svn diff` can show the changes between two revisions of the same file:

```
PROMPT.USER svn diff -r179453:179454 ROADMAP.txt
```

It can also show all changes for a specific changeset. The following will show what changes were made to the current directory and all subdirectories in changeset 179454:

```
PROMPT.USER svn diff -c179454 .
```

Reverting

Local changes (including additions and deletions) can be reverted using `svn revert`. It does not update out-of-date files, but just replaces them with pristine copies of the original version.

Conflicts

If an `svn update` resulted in a merge conflict, Subversion will remember which files have conflicts and refuse to commit any changes to those files until explicitly told that the conflicts have been resolved. The simple, not yet deprecated procedure is the following:

```
PROMPT.USER svn resolved foo
```

However, the preferred procedure is:

```
PROMPT.USER svn resolve --accept=working foo
```

The two examples are equivalent. Possible values for `--accept` are:

- `working`: use the version in your working directory (which one presumes has been edited to resolve the conflicts).
- `base`: use a pristine copy of the version you had before `svn update`, discarding your own changes, the conflicting changes, and possibly other intervening changes as well.
- `mine-full`: use what you had before `svn update`, including your own changes, but discarding the conflicting changes, and possibly other intervening changes as well.
- `theirs-full`: use the version that was retrieved when you did `svn update`, discarding your own changes.

44.5.4 Advanced Use

Sparse Checkouts

SVN allows *sparse*, or partial checkouts of a directory by adding `--depth` to a `svn checkout`.

Valid arguments to `--depth` are:

- `empty`: the directory itself without any of its contents.
- `files`: the directory and any files it contains.
- `immediates`: the directory and any files and directories it contains, but none of the subdirectories' contents.
- `infinity`: anything.

The `--depth` option applies to many other commands, including `svn commit`, `svn revert`, and “`svn diff`”.

Since `--depth` is sticky, there is a `--set-depth` option for “`svn update`” that will change the selected depth. Thus, given the

working copy produced by the previous example:

```
PROMPT.USER cd ~/freebsd
PROMPT.USER svn update --set-depth=immediates .
```

The above command will populate the working copy in `~/freebsd` with `ROADMAP.txt` and empty subdirectories, and nothing will happen when `svn update` is executed on the subdirectories. However, the following command will set the depth for head (in this case) to infinity, and fully populate it:

```
PROMPT.USER svn update --set-depth=infinity head
```

Direct Operation

Certain operations can be performed directly on the repository without touching the working copy. Specifically, this applies to any operation that does not require editing a file, including:

- `log`, `diff`
- `mkdir`
- `remove`, `copy`, `rename`
- `propset`, `propedit`, `propdel`
- `merge`

Branching is very fast. The following command would be used to branch `RELENG_8`:

```
PROMPT.USER svn copy svn+ssh://svn.freebsd.org/base/head svn+ssh://svn.freebsd.org/base/stable/8
```

This is equivalent to the following set of commands which take minutes and hours as opposed to seconds, depending on your network connection:

```
PROMPT.USER svn checkout --depth=immediates svn+ssh://svn.freebsd.org/base
PROMPT.USER cd base
PROMPT.USER svn update --set-depth=infinity head
PROMPT.USER svn copy head stable/8
PROMPT.USER svn commit stable/8
```

Merging with SVN

This section deals with merging code from one branch to another (typically, from head to a stable branch).

Note

In all examples below, `$FSVN` refers to the location of the OS Subversion repository, `svn+ssh://svn.freebsd.org/base/`.

About Merge Tracking

From the user's perspective, merge tracking information (or mergeinfo) is stored in a property called `svn:mergeinfo`, which is a comma-separated list of revisions and ranges of revisions that have been merged. When set on a file, it applies only to that file. When set on a directory, it applies to that directory and its descendants (files and directories) except for those that have their own `svn:mergeinfo`.

It is *not* inherited. For instance, `stable/6/contrib/openpam/` does not implicitly inherit mergeinfo from `stable/6/`, or `stable/6/contrib/`. Doing so would make partial checkouts very hard to manage. Instead, mergeinfo is explicitly propagated down the tree. For merging something into `branch/foo/bar/`, the following rules apply:

1. If `branch/foo/bar/` does not already have a mergeinfo record, but a direct ancestor (for instance, `branch/foo/`) does, then that record will be propagated down to `branch/foo/bar/` before information about the current merge is recorded.

2. Information about the current merge will *not* be propagated back up that ancestor.
3. If a direct descendant of `branch/foo/bar/` (for instance, `branch/foo/bar/baz/`) already has a mergeinfo record, information about the current merge will be propagated down to it.

If you consider the case where a revision changes several separate parts of the tree (for example, `branch/foo/bar/` and `branch/foo/quux/`), but you only want to merge some of it (for example, `branch/foo/bar/`), you will see that these rules make sense. If mergeinfo was propagated up, it would seem like that revision had also been merged to `branch/foo/quux/`, when in fact it had not been.

Selecting the Source and Target for `stable/10` and Newer

Starting with the `stable/10` branch, all merges should be merged to and committed from the root of the branch. All merges should look like:

```
PROMPT.USER svn merge -c r123456 ^/head/ checkout
PROMPT.USER svn commit checkout
```

Note that checkout should be a complete checkout of the branch to which the merge occurs.

Merges to `releng/` branches should always originate from the corresponding `stable/` branch. For example:

```
PROMPT.USER svn merge -c r123456 ^/stable/10 releng/10.0
```

Selecting the Source and Target for `stable/9` and Older

For `stable/9` and earlier, a different strategy was used, distributing mergeinfo around the tree so that merges could be performed without a complete checkout. This procedure proved extremely error-prone, with the convenience of partial checkouts for merges significantly outweighed by the complexity of picking mergeinfo targets. The below describes this now-obsolete procedure, which should be used *only for merges prior to “stable/10”*.

Because of mergeinfo propagation, it is important to choose the source and target for the merge carefully to minimise property changes on unrelated directories.

The rules for selecting the merge target (the directory that you will merge the changes to) can be summarized as follows:

1. Never merge directly to a file.
2. Never, ever merge directly to a file.
3. *Never; ever; ever* merge directly to a file.
4. Changes to kernel code should be merged to `sys/`. For instance, a change to the `MAN.ICHWD.4` driver should be merged to `sys/`, not `sys/dev/ichwd/`. Likewise, a change to the TCP/IP stack should be merged to `sys/`, not `sys/netinet/`.
5. Changes to code under `etc/` should be merged at `etc/`, not below it.
6. Changes to vendor code (code in `contrib/`, `crypto/` and so on) should be merged to the directory where vendor imports happen. For instance, a change to `crypto/openssl/util/` should be merged to `crypto/openssl/`. This is rarely an issue, however, since changes to vendor code are usually merged wholesale.
7. Changes to userland programs should as a general rule be merged to the directory that contains the Makefile for that program. For instance, a change to `usr.bin/xlint/arch/i386/` should be merged to `usr.bin/xlint/`.
8. Changes to userland libraries should as a general rule be merged to the directory that contains the Makefile for that library. For instance, a change to `lib/libc/gen/` should be merged to `lib/libc/`.

9. There may be cases where it makes sense to deviate from the rules for userland programs and libraries. For instance, everything under `lib/libpam/` is merged to `lib/libpam/`, even though the library itself and all of the modules each have their own Makefile.
10. Changes to manual pages should be merged to `share/man/manN/`, for the appropriate value of `N`.
11. Other changes to `share/` should be merged to the appropriate subdirectory and not to `share/` directly.
12. Changes to a top-level file in the source tree such as `UPDATING` or `Makefile.incl` should be merged directly to that file rather than to the root of the whole tree. Yes, this is an exception to the first three rules.
13. When in doubt, ask.

If you need to merge changes to several places at once (for instance, changing a kernel interface and every userland program that uses it), merge each target separately, then commit them together. For instance, if you merge a revision that changed a kernel API and updated all the userland bits that used that API, you would merge the kernel change to `sys`, and the userland bits to the appropriate userland directories, then commit all of these in one go.

The source will almost invariably be the same as the target. For instance, you will always merge `stable/7/lib/libc/` from `head/lib/libc/`. The only exception would be when merging changes to code that has moved in the source branch but not in the parent branch. For instance, a change to `MAN.PKILL.1` would be merged from `bin/pkill/` in `head` to `usr.bin/pkill/` in `stable/7`.

Preparing the Merge Target

Because of the mergeinfo propagation issues described earlier, it is very important that you never merge changes into a sparse working copy. You must always have a full checkout of the branch you will merge into. For instance, when merging from `HEAD` to `7`, you must have a full checkout of `stable/7`:

```
PROMPT.USER cd stable/7
PROMPT.USER svn up --set-depth=infinity
```

The target directory must also be up-to-date and must not contain any uncommitted changes or stray files.

Identifying Revisions

Identifying revisions to be merged is a must. If the target already has complete mergeinfo, ask SVN for a list:

```
PROMPT.USER cd stable/6/contrib/openpam
PROMPT.USER svn mergeinfo --show-revs=eligible $FSVN/head/contrib/openpam
```

If the target does not have complete mergeinfo, check the log for the merge source.

Merging

Now, let us start merging!

The Principles Say you would like to merge:

- revision `$R`
- in directory `$target` in stable branch `$B`
- from directory `$source` in `head`
- `$FSVN` is `svn+ssh://svn.freebsd.org/base`

Assuming that revisions \$P and \$Q have already been merged, and that the current directory is an up-to-date working copy of stable/\$B, the existing mergeinfo looks like this:

```
PROMPT.USER svn propget svn:mergeinfo -R $target
$target - /head/$source:$P,$Q
```

Merging is done like so:

```
PROMPT.USER svn merge -c$r $FSVN/head/$source $target
```

Checking the results of this is possible with `svn diff`.

The `svn:mergeinfo` now looks like:

```
PROMPT.USER svn propget svn:mergeinfo -R $target
$target - head/$source:$P,$Q,$R
```

If the results are not exactly as shown, assistance may be required before committing as mistakes may have been made, or there may be something wrong with the existing mergeinfo, or there may be a bug in Subversion.

Practical Example As a practical example, consider the following scenario. The changes to `netmap.4` in `r238987` are to be merged from `CURRENT` to `9-STABLE`. The file resides in `head/share/man/man4`. According to `?`, this is also where to do the merge. Note that in this example all paths are relative to the top of the svn repository. For more information on the directory layout, see `?`.

The first step is to inspect the existing mergeinfo.

```
PROMPT.USER svn propget svn:mergeinfo -R stable/9/share/man/man4
```

Take a quick note of how it looks before moving on to the next step; doing the actual merge:

```
PROMPT.USER svn merge -c r238987 svn+ssh://svn.freebsd.org/base/head/share/man/man4 stable/9/share/man/man4
--- Merging r238987 into 'stable/9/share/man/man4':
U    stable/9/share/man/man4/netmap.4
--- Recording mergeinfo for merge of r238987 into
'stable/9/share/man/man4':
U    stable/9/share/man/man4
```

Check that the revision number of the merged revision has been added. Once this is verified, the only thing left is the actual commit.

```
PROMPT.USER svn commit stable/9/share/man/man4
```

Merging into the Kernel (`sys/`) As stated above, merging into the kernel is different from merging in the rest of the tree. In many ways merging to the kernel is simpler because there is always the same merge target (`sys/`).

Once `svn merge` has been executed, `svn diff` has to be run on the directory to check the changes. This may show some unrelated property changes, but these can be ignored. Next, build and test the kernel, and, once the tests are complete, commit the code as normal, making sure that the commit message starts with “Merge r226222 from head”, or similar.

Precautions Before Committing

As always, build world (or appropriate parts of it).

Check the changes with `svn diff` and `svn stat`. Make sure all the files that should have been added or deleted were in fact added or deleted.

Take a closer look at any property change (marked by a M in the second column of “svn stat”). Normally, no svn:mergeinfo properties should be anywhere except the target directory (or directories). If something looks fishy, ask for help.

Committing

Make sure to commit a top level directory to have the mergeinfo included as well. Do not specify individual files on the command line. For more information about committing files in general, see the relevant section of this primer.

Vendor Imports with SVN

Important

Please read this entire section before starting a vendor import.

Note

Patches to vendor code fall into two categories:

- Vendor patches: these are patches that have been issued by the vendor, or that have been extracted from the vendor’s version control system, which address issues which in your opinion cannot wait until the next vendor release.
- OS patches: these are patches that modify the vendor code to address OS-specific issues.

The nature of a patch dictates where it should be committed:

- Vendor patches should be committed to the vendor branch, and merged from there to head. If the patch addresses an issue in a new release that is currently being imported, it *must not* be committed along with the new release: the release must be imported and tagged first, then the patch can be applied and committed. There is no need to re-tag the vendor sources after committing the patch.
- OS patches should be committed directly to head.

Preparing the Tree

If importing for the first time after the switch to Subversion, flattening and cleaning up the vendor tree is necessary, as well as bootstrapping the merge history in the main tree.

Flattening During the conversion from CVS to Subversion, vendor branches were imported with the same layout as the main tree. This means that the pf vendor sources ended up in vendor/pf/dist/contrib/pf. The vendor source is best directly in vendor/pf/dist.

To flatten the pf tree:

```
PROMPT.USER cd vendor/pf/dist/contrib/pf
PROMPT.USER svn mv $(svn list) ../..
PROMPT.USER cd ../..
PROMPT.USER svn rm contrib
PROMPT.USER svn propdel -R svn:mergeinfo .
PROMPT.USER svn commit
```

The `propdel` bit is necessary because starting with 1.5, Subversion will automatically add `svn:mergeinfo` to any directory that is copied or moved. In this case, as nothing is being merged from the deleted tree, they just get in the way.

Tags may be flattened as well (3, 4, 3.5 etc.); the procedure is exactly the same, only changing `dist` to `3.5` or similar, and putting the `svn commit` off until the end of the process.

Cleaning Up The `dist` tree can be cleaned up as necessary. Disabling keyword expansion is recommended, as it makes no sense on unmodified vendor code and in some cases it can even be harmful. OpenSSH, for example, includes two files that originated with OS and still contain the original version tags. To do this:

```
PROMPT.USER svn propdel svn:keywords -R .
PROMPT.USER svn commit
```

Bootstrapping Merge History If importing for the first time after the switch to Subversion, bootstrap `svn:mergeinfo` on the target directory in the main tree to the revision that corresponds to the last related change to the vendor tree, prior to importing new sources:

```
PROMPT.USER cd head/contrib/pf
PROMPT.USER svn merge --record-only svn+ssh://svn.freebsd.org/base/vendor/pf/dist@180876 .
PROMPT.USER svn commit
```

Importing New Sources

With two commits—one for the import itself and one for the tag—this step can optionally be repeated for every upstream release between the last import and the current import.

Preparing the Vendor Sources Unlike in CVS where only the needed parts were imported into the vendor tree to avoid bloating the main tree, Subversion is able to store a full distribution in the vendor tree. So, import everything, but merge only what is required.

A `svn add` is required to add any files that were added since the last vendor import, and `svn rm` is required to remove any that were removed since. Preparing sorted lists of the contents of the vendor tree and of the sources that are about to be imported is recommended, to facilitate the process.

```
PROMPT.USER cd vendor/pf/dist
PROMPT.USER svn list -R | grep -v '/$' | sort >../old
PROMPT.USER cd ../pf-4.3
PROMPT.USER find . -type f | cut -c 3- | sort >../new
```

With these two files, `comm -23 ../old ../new` will list removed files (files only in `old`), while `comm -13 ../old ../new` will list added files only in `new`.

Importing into the Vendor Tree Now, the sources must be copied into `dist` and the `svn add` and `svn rm` commands should be used as needed:

```
PROMPT.USER cd vendor/pf/pf-4.3
PROMPT.USER tar cf - . | tar xf - -C ../dist
PROMPT.USER cd ../dist
PROMPT.USER comm -23 ../old ../new | xargs svn rm
PROMPT.USER comm -13 ../old ../new | xargs svn --parents add
```


If any directories were removed, they will have to be `svn rm`d manually. Nothing will break if they are not, but they will remain in the tree.

Check properties on any new files. All text files should have `svn:eol-style` set to `native`. All binary files should have `svn:mime-type` set to `application/octet-stream` unless there is a more appropriate media type. Executable files should have `svn:executable` set to `*`. No other properties should exist on any file in the tree.

Committing is now possible, however it is good practice to make sure that everything is OK by using the `svn stat` and `svn diff` commands.

Tagging Once committed, vendor releases should be tagged for future reference. The best and quickest way to do this is directly in the repository:

```
PROMPT.USER svn cp svn+ssh://svn.freebsd.org/base/vendor/pf/dist svn+ssh://svn.freebsd.org/base/vendor/pf/dist
```

Once that is complete, `svn up` the working copy of `vendor/pf` to get the new tag, although this is rarely needed.

If creating the tag in the working copy of the tree, `svn:mergeinfo` results must be removed:

```
PROMPT.USER cd vendor/pf
PROMPT.USER svn cp dist 4.3
PROMPT.USER svn propdel svn:mergeinfo -R 4.3
```

Merging to Head

```
PROMPT.USER cd head/contrib/pf
PROMPT.USER svn up
PROMPT.USER svn merge --accept=postpone svn+ssh://svn.freebsd.org/base/vendor/pf/dist .
```

The `--accept=postpone` tells Subversion that it should not complain because merge conflicts will be taken care of manually.

Tip

The `cvs2svn` changeover occurred on June 3, 2008. When performing vendor merges for packages which were already present and converted by the `cvs2svn` process, the command used to merge `/vendor/package_name/dist` to `/head/package_location` (for example, `head/contrib/sendmail`) must use `-c REV` to indicate the revision to merge from the `/vendor` tree. For example:

```
PROMPT.USER svn checkout svn+ssh://svn.freebsd.org/base/head/contrib/sendmail
PROMPT.USER cd sendmail
PROMPT.USER svn merge -c r261190 ^/vendor/sendmail/dist .
```

`^` is an alias for the repository path.

Note

If using the Zsh shell, the `^` must be escaped with `\`. This means `^/head` should be `\^/head`.

It is necessary to resolve any merge conflicts.

Make sure that any files that were added or removed in the vendor tree have been properly added or removed in the main tree. To check diffs against the vendor branch:

```
PROMPT.USER svn diff --no-diff-deleted --old=svn+ssh://svn.freebsd.org/base/vendor/pf/dist --new=.
```

The `--no-diff-deleted` tells Subversion not to complain about files that are in the vendor tree but not in the main tree, i.e., things that would have previously been removed before the vendor import, like for example the vendor's makefiles and configure scripts.

Using CVS, once a file was off the vendor branch, it was not able to be put back. With Subversion, there is no concept of on or off the vendor branch. If a file that previously had local modifications, to make it not show up in diffs in the vendor tree, all that has to be done is remove any left-over cruft like OS version tags, which is much easier.

If any changes are required for the world to build with the new sources, make them now, and keep testing until everything builds and runs perfectly.

Committing the Vendor Import

Committing is now possible! Everything must be committed in one go. If done properly, the tree will move from a consistent state with old code, to a consistent state with new code.

From Scratch

Importing into the Vendor Tree This section is an example of importing and tagging byacc into head.

First, prepare the directory in vendor:

```
PROMPT.USER svn co --depth immediates $FSVN/vendor
PROMPT.USER cd vendor
PROMPT.USER svn mkdir byacc
PROMPT.USER svn mkdir byacc/dist
```

Now, import the sources into the `dist` directory. Once the files are in place, `svn add` the new ones, then `svn commit` and tag the imported version. To save time and bandwidth, direct remote committing and tagging is possible:

```
PROMPT.USER svn cp -m "Tag byacc 20120115" $FSVN/vendor/byacc/dist $FSVN/vendor/byacc/20120115
```

Merging to head Due to this being a new file, copy it for the merge:

```
PROMPT.USER svn cp -m "Import byacc to contrib" $FSVN/vendor/byacc/dist $FSVN/head/contrib/byacc
```

Working normally on newly imported sources is still possible.

Reverting a Commit

Reverting a commit to a previous version is fairly easy:

```
PROMPT.USER svn merge -r179454:179453 ROADMAP.txt
PROMPT.USER svn commit
```

Change number syntax, with negative meaning a reverse change, can also be used:

```
PROMPT.USER svn merge -c -179454 ROADMAP.txt
PROMPT.USER svn commit
```

This can also be done directly in the repository:

```
PROMPT.USER svn merge -r179454:179453 svn+ssh://svn.freebsd.org/base/ROADMAP.txt
```

****Note****

```
It is important to ensure that the mergeinfo is correct when
reverting a file in order to permit ``svn mergeinfo --eligible`` to
work as expected.
```

Reverting the deletion of a file is slightly different. Copying the version of the file that predates the deletion is required. For example, to restore a file that was deleted in revision N, restore version N-1:

```
PROMPT.USER svn copy svn+ssh://svn.freebsd.org/base/ROADMAP.txt@179454
PROMPT.USER svn commit
```

or, equally:

```
PROMPT.USER svn copy svn+ssh://svn.freebsd.org/base/ROADMAP.txt@179454 svn+ssh://svn.freebsd.org/base
```

Do *not* simply recreate the file manually and `svn add` it—this will cause history to be lost.

Fixing Mistakes

While we can do surgery in an emergency, do not plan on having mistakes fixed behind the scenes. Plan on mistakes remaining in the logs forever. Be sure to check the output of `svn status` and “`svn`

`diff`” before committing.

Mistakes will happen but, they can generally be fixed without disruption.

Take a case of adding a file in the wrong location. The right thing to do is to `svn move` the file to the correct location and commit. This causes just a couple of lines of metadata in the repository journal, and the logs are all linked up correctly.

The wrong thing to do is to delete the file and then `svn add` an independent copy in the correct location. Instead of a couple of lines of text, the repository journal grows an entire new copy of the file. This is a waste.

Setting up a svnsync Mirror

You probably do not want to do this unless there is a good reason for it. Such reasons might be to support many multiple local read-only client machines, or if your network bandwidth is limited. Starting a fresh mirror from empty would take a very long time. Expect a minimum of 10 hours for high speed connectivity. If you have international links, expect this to take 4 to 10 times longer.

A far better option is to grab a seed file. It is large (~1GB) but will consume less network traffic and take less time to fetch than a `svnsync` will. This is possible in one of the following three ways:

```
PROMPT.USER rsync -va --partial --progress freefall:/home/peter/svnmirror-base-r179637.tbz2 .
```

```
PROMPT.USER rsync -va --partial --progress rsync://repoman.freebsd.org:50873/svnseed/svnmirror-base-r
```

```
PROMPT.USER fetch ftp://ftp.freebsd.org/pub/FreeBSD/development/subversion/svnmirror-base-r221445.tar
```

Once you have the file, extract it to somewhere like `home/svnmirror/base/`. Then, update it, so that it fetches changes since the last revision in the archive:

```
PROMPT.USER svnsync sync file:///home/svnmirror/base
```

You can then set that up to run from MAN.CRON.8, do checkouts locally, set up a `svnsync` server for your local machines to talk to, etc.

The seed mirror is set to fetch from `svn://svn.freebsd.org/base`. The configuration for the mirror is stored in `revprop 0` on the local mirror. To see the configuration, try:

```
PROMPT.USER svn proplist -v --revprop -r 0 file:///home/svnmirror/base
```

Use `propset` to change things.

Committing High-ASCII Data

Files that have high-ASCII bits are considered binary files in SVN, so the pre-commit checks fail and indicate that the `mime-type` property should be set to `application/octet-stream`. However, the use of this is discouraged, so please do not set it. The best way is always avoiding high-ASCII data, so that it can be read everywhere with any text editor but if it is not avoidable, instead of changing the mime-type, set the `fbbsd:notbinary` property with `propset`:

```
PROMPT.USER svn propset fbbsd:notbinary yes foo.data
```

Maintaining a Project Branch

A project branch is one that is synced to head (or another branch) is used to develop a project then commit it back to head. In SVN, “dolphin” branching is used for this. A “dolphin” branch is one that diverges for a while and is finally committed back to the original branch. During development code migration in one direction (from head to the branch only). No code is committed back to head until the end. Once you commit back at the end, the branch is dead (although you can have a new branch with the same name after you delete the branch if you want).

As per http://people.freebsd.org/~peter/svn_notes.txt, work that is intended to be merged back into HEAD should be in `base/projects/`. If you are doing work that is beneficial to the OS community in some way but not intended to be merged directly back into HEAD then the proper location is `base/user/your-name/`. [This page](#) contains further details.

To create a project branch:

```
PROMPT.USER svn copy svn+ssh://svn.freebsd.org/base/head svn+ssh://svn.freebsd.org/base/projects/spi
```

To merge changes from HEAD back into the project branch:

```
PROMPT.USER cd copy_of_spif
PROMPT.USER svn merge svn+ssh://svn.freebsd.org/base/head
PROMPT.USER svn commit
```

It is important to resolve any merge conflicts before committing.

44.5.5 Some Tips

In commit logs etc., “rev 179872” should be spelled “r179872” as per convention.

Speeding up svn is possible by adding the following to `~/.ssh/config`:

```
Host *
ControlPath ~/.ssh/sockets/master-%l-%r@%h:%p
ControlMaster auto
ControlPersist yes
```

and then typing

```
mkdir ~/.ssh/sockets
```

Checking out a working copy with a stock Subversion client without OS-specific patches (OPTIONS_SET=FREEBSD_TEMPLATE) will mean that `$FreeBSD$` tags will not be expanded. Once the correct version has been installed, trick Subversion into expanding them like so:

```
PROMPT.USER svn propdel -R svn:keywords .
PROMPT.USER svn revert -R .
```

This will wipe out uncommitted patches.

44.6 Setup, Conventions, and Traditions

There are a number of things to do as a new developer. The first set of steps is specific to committers only. These steps must be done by a mentor for those who are not committers.

44.6.1 For New Committers

Those who have been given commit rights to the OS repositories must follow these steps.

- Get mentor approval before committing each of these changes!
- The `.ent` and `.xml` files mentioned below exist in the OS Documentation Project SVN repository at `'svn.FreeBSD.org/doc/ <svn.FreeBSD.org/doc/>'__`.
- New files that do not have the `FreeBSD=%H svn:keywords` property will be rejected when attempting to commit them to the repository. Be sure to read ? regarding adding and removing files. Verify that `~/subversion/config` contains the necessary “auto-props” entries from `auto-props.txt` mentioned there.
- All `src` commits should go to `OS.CURRENT` first before being merged to `OS.STABLE`. The `OS.STABLE` branch must maintain ABI and API compatibility with earlier versions of that branch. Do not merge changes that break this compatibility.

`doc/head/share/xml/authors.ent` — Add an author entity. Later steps depend on this entity, and missing this step will cause the `doc/` build to fail. This is a relatively easy task, but remains a good first test of version control skills.

`doc/head/en_US.ISO8859-1/articles/contributors/contrib.committers.xml` — Add an entry to the “Developers” section of the Contributors List. Entries are sorted by last name.

`doc/head/en_US.ISO8859-1/articles/contributors/contrib.additional.xml` — Remove the entry from the “Additional Contributors” section. Entries are sorted by last name.

`doc/head/share/xml/news.xml` — Add an entry. Look for the other entries that announce new committers and follow the format. Use the date from the commit bit approval email from core@FreeBSD.org.

`doc/head/share/pgpkeys/pgpkeys.ent` and `doc/head/share/pgpkeys/pgpkeys-developers.xml` - Add your PGP or GnuPG key. Those who do not yet have a key should see ?.

A.DES.EMAIL has written a shell script (`doc/head/share/pgpkeys/addkey.sh`) to make this easier. See the [README](#) file for more information.

Use `doc/head/share/pgpkeys/checkkey.sh` to verify that keys meet minimal best-practices standards.

After adding and checking a key, add both updated files to source control and then commit them. Entries in this file are sorted by last name.

Note

It is very important to have a current PGP/GnuPG key in the repository. The key may be required for positive identification of a committer. For example, the A.ADMINS might need it for account recovery. A complete keyring of FreeBSD.org users is available for download from <http://www.FreeBSD.org/doc/pgpkeyring.txt>.

`base/head/share/misc/committers-repository.dot` — Add an entry to the current committers section, where repository is `doc`, `ports`, or `src`, depending on the commit privileges granted.

Add an entry for each additional mentor/mentee relationship in the bottom section.

See ? to generate or set a Kerberos for use with other OS services like the bug tracking database.

OS Wiki Account — A wiki account allows sharing projects and ideas. Those who do not yet have an account can contact clusteradm@FreeBSD.org to obtain one.

Wiki Information - After gaining access to the wiki, some people add entries to the [How We Got Here](#), [Irc Nicks](#), and [Dogs of FreeBSD](#) pages.

`ports/astro/xearth/files/freebsd.committers.markers` and `src/usr.bin/calendar/calendars/calendars` - Some people add entries for themselves to these files to show where they are located or the date of their birthday.

Subscribers to A.SVN-SRC-ALL.NAME, A.SVN-PORTS-ALL.NAME or A.SVN-DOC-ALL.NAME might wish to unsubscribe to avoid receiving duplicate copies of commit messages and followups.

44.6.2 For Everyone

Introduce yourself to the other developers, otherwise no one will have any idea who you are or what you are working on. The introduction need not be a comprehensive biography, just write a paragraph or two about who you are, what you plan to be working on as a developer in OS, and who will be your mentor. Email this to the A.DEVELOPERS and you will be on your way!

Log into freefall.FreeBSD.org and create a `/var/forward/user` (where user is your username) file containing the e-mail address where you want mail addressed to yourusername@FreeBSD.org to be forwarded. This includes all of the commit messages as well as any other mail addressed to the A.COMMITTERS and the A.DEVELOPERS. Really large mailboxes which have taken up permanent residence on hub often get “accidentally” truncated without warning, so forward it or read it and you will not lose it.

Due to the severe load dealing with SPAM places on the central mail servers that do the mailing list processing the front-end server does do some basic checks and will drop some messages based on these checks. At the moment proper DNS information for the connecting host is the only check in place but that may change. Some people blame these checks for bouncing valid email. If you want these checks turned off for your email you can place a file named `.spam_lover` in your home directory on freefall.FreeBSD.org to disable the checks for your email.

Note

Those who are developers but not committers will not be subscribed to the committers or developers mailing lists. The subscriptions are derived from the access rights.

44.6.3 Mentors

All new developers have a mentor assigned to them for the first few months. A mentor is responsible for teaching the mentee the rules and conventions of the project and guiding their first steps in the developer community. The mentor is also personally responsible for the mentee’s actions during this initial period.

For committers: do not commit anything without first getting mentor approval. Document that approval with an `Approved by:` line in the commit message.

When the mentor decides that a mentee has learned the ropes and is ready to commit on their own, the mentor announces it with a commit to `mentors`.

44.7 Commit Log Messages

This section contains some suggestions and traditions for how commit logs are formatted.

As well as including an informative message with each commit you may need to include some additional information.

This information consists of one or more lines containing the key word or phrase, a colon, tabs for formatting, and then the additional information.

The key words or phrases are:

PR:	The problem report (if any) which is affected (typically, by being closed) by this commit. Only include one PR per line as the automated scripts which parse this line cannot understand more than one.
Differential Revision:	The full URL of the Phabricator review. For example: https://reviews.freebsd.org/D1708 .
Submitted by:	The name and e-mail address of the person that submitted the fix; for developers, just the username on the OS cluster. If the submitter is the maintainer of the port to which you are committing, include “(maintainer)” after the email address. Avoid obfuscating the email address of the submitter as this adds additional work when searching logs.
Reviewed by:	The name and e-mail address of the person or people that reviewed the change; for developers, just the username on the OS cluster. If a patch was submitted to a mailing list for review, and the review was favorable, then just include the list name.
Approved by:	The name and e-mail address of the person or people that approved the change; for developers, just the username on the OS cluster. It is customary to get prior approval for a commit if it is to an area of the tree to which you do not usually commit. In addition, during the run up to a new release all commits <i>must</i> be approved by the release engineering team. If these are your first commits then you should have passed them past your mentor first, and you should list your mentor, as in “username-of-mentor (mentor)”. If a team approved these commits then include the team name followed by the username of the approver in parentheses. For example: “re@ (username)”
Obtained from:	The name of the project (if any) from which the code was obtained. Do not use this line for the name of an individual person.
MFC after:	If you wish to receive an e-mail reminder to MFC at a later date, specify the number of days, weeks, or months after which an MFC is planned.
Relnotes:	If the change is a candidate for inclusion in the release notes for the next release from the branch, set to <code>yes</code> .
Security:	If the change is related to a security vulnerability or security exposure, include one or more references or a description of the issue. If possible, include a VuXML URL or a CVE ID.

You want to commit a change based on a PR submitted by John Smith containing a patch. The end of the commit message should look something like this.

```
...

PR:                12345
Submitted by:      John Smith <John.Smith@example.com>
```

You want to change the virtual memory system. You have posted patches to the appropriate mailing list (in this case, `freebsd-arch`) and the changes have been approved.

```
...

Reviewed by:        -arch
```

You want to commit a port. You have collaborated with the listed MAINTAINER, who has told you to go ahead and commit.

```
...
Approved by:      abc (maintainer)
```

Where abc is the account name of the person who approved.

You want to commit some code based on work done in the OpenBSD project.

```
...
Obtained from:    OpenBSD
```

You want to commit some code which will be merged from OS.CURRENT into the OS.STABLE branch after two weeks.

```
...
MFC after:        2 weeks
```

Where 2 is the number of days, weeks, or months after which an MFC is planned. The weeks option may be day, days, week, weeks, month, months.

In many cases you may need to combine some of these.

Consider the situation where a user has submitted a PR containing code from the NetBSD project. You are looking at the PR, but it is not an area of the tree you normally work in, so you have decided to get the change reviewed by the arch mailing list. Since the change is complex, you opt to MFC after one month to allow adequate testing.

The extra information to include in the commit would look something like

```
PR:                54321
Submitted by:      John Smith <John.Smith@example.com>
Reviewed by:       -arch
Obtained from:     NetBSD
MFC after:         1 month
Relnotes:         yes
```

44.8 Preferred License for New Files

Currently the OS Project suggests and uses the following text as the preferred license scheme:

```
/*-
 * Copyright (c) [year] [your name]
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
```



```

* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* [id for your version control system, if any]
*/

```

The OS project strongly discourages the so-called “advertising clause” in new code. Due to the large number of contributors to the OS project, complying with this clause for many commercial vendors has become difficult. If you have code in the tree with the advertising clause, please consider removing it. In fact, please consider using the above license for your code.

The OS project discourages completely new licenses and variations on the standard licenses. New licenses require the approval of the A.CORE to reside in the main repository. The more different licenses that are used in the tree, the more problems that this causes to those wishing to utilize this code, typically from unintended consequences from a poorly worded license.

Project policy dictates that code under some non-BSD licenses must be placed only in specific sections of the repository, and in some cases, compilation must be conditional or even disabled by default. For example, the GENERIC kernel must be compiled under only licenses identical to or substantially similar to the BSD license. GPL, APSL, CDDL, etc, licensed software must not be compiled into GENERIC.

Developers are reminded that in open source, getting “open” right is just as important as getting “source” right, as improper handling of intellectual property has serious consequences. Any questions or concerns should immediately be brought to the attention of the core team.

44.9 Keeping Track of Licenses Granted to the OS Project

Various software or data exist in the repositories where the OS project has been granted a special licence to be able to use them. A case in point are the Terminus fonts for use with MAN.VT.4. Here the author Dimitar Zhekov has allowed us to use the “Terminus BSD Console” font under a 2-clause BSD license rather than the regular Open Font License he normally uses.

It is clearly sensible to keep a record of any such license grants. To that end, the A.CORE has decided to keep an archive of them. Whenever the OS project is granted a special license we require the A.CORE to be notified. Any developers involved in arranging such a license grant, please send details to the A.CORE including:

- Contact details for people or organizations granting the special license.
- What files, directories etc. in the repositories are covered by the license grant including the revision numbers where any specially licensed material was committed.
- The date the license comes into effect from. Unless otherwise agreed, this will be the date the license was issued by the authors of the software in question.
- The license text.
- A note of any restrictions, limitations or exceptions that apply specifically to OS’s usage of the licensed material.
- Any other relevant information.

Once the A.CORE is satisfied that all the necessary details have been gathered and are correct, the secretary will send a PGP-signed acknowledgement of receipt including the license details. This receipt will be persistently archived and serve as our permanent record of the license grant.

The license archive should contain only details of license grants; this is not the place for any discussions around licensing or other subjects. Access to data within the license archive will be available on request to the A.CORE.

44.10 Developer Relations

If you are working directly on your own code or on code which is already well established as your responsibility, then there is probably little need to check with other committers before jumping in with a commit. If you see a bug in an area of the system which is clearly orphaned (and there are a few such areas, to our shame), the same applies. If, however, you are about to modify something which is clearly being actively maintained by someone else (and it is only by watching the `repository-committers` mailing list that you can really get a feel for just what is and is not) then consider sending the change to them instead, just as you would have before becoming a committer. For ports, you should contact the listed MAINTAINER in the Makefile. For other parts of the repository, if you are unsure who the active maintainer might be, it may help to scan the revision history to see who has committed changes in the past. An example script that lists each person who has committed to a given file along with the number of commits each person has made can be found at on freefall at `~eadler/bin/whodid`. If your queries go unanswered or the committer otherwise indicates a lack of interest in the area affected, go ahead and commit it.

Note

Avoid sending private emails to maintainers. Other people might be interested in the conversation, not just the final output.

If you are unsure about a commit for any reason at all, have it reviewed by `-hackers` before committing. Better to have it flamed then and there rather than when it is part of the repository. If you do happen to commit something which results in controversy erupting, you may also wish to consider backing the change out again until the matter is settled. Remember – with a version control system we can always change it back.

Do not impugn the intentions of someone you disagree with. If they see a different solution to a problem than you, or even a different problem, it is not because they are stupid, because they have questionable parentage, or because they are trying to destroy your hard work, personal image, or OS, but simply because they have a different outlook on the world. Different is good.

Disagree honestly. Argue your position from its merits, be honest about any shortcomings it may have, and be open to seeing their solution, or even their vision of the problem, with an open mind.

Accept correction. We are all fallible. When you have made a mistake, apologize and get on with life. Do not beat up yourself, and certainly do not beat up others for your mistake. Do not waste time on embarrassment or recrimination, just fix the problem and move on.

Ask for help. Seek out (and give) peer reviews. One of the ways open source software is supposed to excel is in the number of eyeballs applied to it; this does not apply if nobody will review code.

44.11 If in Doubt...

When you are not sure about something, whether it be a technical issue or a project convention be sure to ask. If you stay silent you will never make progress.

If it relates to a technical issue ask on the public mailing lists. Avoid the temptation to email the individual person that knows the answer. This way everyone will be able to learn from the question and the answer.

For project specific or administrative questions you should ask, in order:

- Your mentor or former mentor.
- An experienced committer on IRC, email, etc.
- Any team with a “hat”, as they should give you a definitive answer.

- If still not sure, ask on A.DEVELOPERS.

Once your question is answered, if no one pointed you to documentation that spelled out the answer to your question, document it, as others will have the same question.

44.12 Bugzilla

The OS Project utilizes Bugzilla for tracking bugs and change requests. Be sure that if you commit a fix or suggestion found in the PR database to close it. It is also considered nice if you take time to close any PRs associated with your commits, if appropriate.

Committers with non-OS.org Bugzilla accounts can have the old account merged with the OS.org account by entering a new bug. Choose `Supporting Services` as the Product, and `Bug Tracker` as the Component.

You can find out more about Bugzilla at:

- OS Problem Report Handling Guidelines
- <http://www.FreeBSD.org/support.html>

44.13 Phabricator

The OS Project utilizes [Phabricator](#) for code review requests. See the [CodeReview](#) wiki page for details.

44.14 Who's Who

Besides the repository meisters, there are other OS project members and teams whom you will probably get to know in your role as a committer. Briefly, and by no means all-inclusively, these are:

A.DOCENG doceng is the group responsible for the documentation build infrastructure, approving new documentation committers, and ensuring that the OS website and documentation on the FTP site is up to date with respect to the subversion tree. It is not a conflict resolution body. The vast majority of documentation related discussion takes place on the A.DOC. More details regarding the doceng team can be found in its [charter](#). Committers interested in contributing to the documentation should familiarize themselves with the Documentation Project Primer.

A.BDE.EMAIL Bruce is the Style Police-Meister. When you do a commit that could have been done better, Bruce will be there to tell you. Be thankful that someone is. Bruce is also very knowledgeable on the various standards applicable to OS.

A.RE.MEMBERS.EMAIL These are the members of the A.RE. This team is responsible for setting release deadlines and controlling the release process. During code freezes, the release engineers have final authority on all changes to the system for whichever branch is pending release status. If there is something you want merged from OS.CURRENT to OS.STABLE (whatever values those may have at any given time), these are the people to talk to about it.

Hiroki is also the keeper of the release documentation (`src/release/doc/*`). If you commit a change that you think is worthy of mention in the release notes, please make sure he knows about it. Better still, send him a patch with your suggested commentary.

A.DES.EMAIL Dag-Erling is the OS Security Officer and oversees the A.SECURITY-OFFICER.

A.WOLLMAN.EMAIL If you need advice on obscure network internals or are not sure of some potential change to the networking subsystem you have in mind, Garrett is someone to talk to. Garrett is also very knowledgeable on the various standards applicable to OS.

A.COMMITTERS A.SVN-SRC-ALL.NAME, A.SVN-PORTS-ALL.NAME and A.SVN-DOC-ALL.NAME are the mailing lists that the version control system uses to send commit messages to. You should *never* send email directly to these lists. You should only send replies to this list when they are short and are directly related to a commit.

A.DEVELOPERS All committers are subscribed to -developers. This list was created to be a forum for the committers “community” issues. Examples are Core voting, announcements, etc.

The A.DEVELOPERS is for the exclusive use of OS committers. In order to develop OS, committers must have the ability to openly discuss matters that will be resolved before they are publicly announced. Frank discussions of work in progress are not suitable for open publication and may harm OS.

All OS committers are expected not to not publish or forward messages from the A.DEVELOPERS outside the list membership without permission of all of the authors. Violators will be removed from the A.DEVELOPERS, resulting in a suspension of commit privileges. Repeated or flagrant violations may result in permanent revocation of commit privileges.

This list is *not* intended as a place for code reviews or for any technical discussion. In fact using it as such hurts the OS Project as it gives a sense of a closed list where general decisions affecting all of the OS using community are made without being “open”. Last, but not least *never, never ever, email the A.DEVELOPERS and CC:/BCC: another OS list*. Never, ever email another OS email list and CC:/BCC: the A.DEVELOPERS. Doing so can greatly diminish the benefits of this list.

44.15 SSH Quick-Start Guide

If you do not wish to type your password in every time you use MAN.SSH.1, and you use RSA or DSA keys to authenticate, MAN.SSH-AGENT.1 is there for your convenience. If you want to use MAN.SSH-AGENT.1, make sure that you run it before running other applications. X users, for example, usually do this from their `.xsession` or `.xinitrc`. See MAN.SSH-AGENT.1 for details.

Generate a key pair using MAN.SSH-KEYGEN.1. The key pair will wind up in your `$HOME/.ssh/` directory.

Send your public key (`$HOME/.ssh/id_dsa.pub` or `$HOME/.ssh/id_rsa.pub`) to the person setting you up as a committer so it can be put into your login in `/etc/ssh-keys/` on freefall.

Now you should be able to use MAN.SSH-ADD.1 for authentication once per session. This will prompt you for your private key’s pass phrase, and then store it in your authentication agent (MAN.SSH-AGENT.1). If you no longer wish to have your key stored in the agent, issuing `ssh-add -d` will remove it.

Test by doing something such as “ssh freefall.FreeBSD.org ls /usr”.

For more information, see security/openssh, MAN.SSH.1, MAN.SSH-ADD.1, MAN.SSH-AGENT.1, MAN.SSH-KEYGEN.1, and MAN.SCP.1.

For information on adding, changing, or removing MAN.SSH.1 keys, see “this article <this article>’__.

44.16 COVERITY Availability for OS Committers

All OS developers can obtain access to Coverity analysis results of all OS Project software. All who are interested in obtaining access to the analysis results of the automated Coverity runs, can sign up at “Coverity

Scan <Coverity Scan>’__.

The OS wiki includes a mini-guide for developers who are interested in working with the COVERITY analysis reports: <http://wiki.freebsd.org/CoverityPrevent>. Please note that this mini-guide is only readable by OS developers, so if you cannot access this page, you will have to ask someone to add you to the appropriate Wiki access list.

Finally, all OS developers who are going to use COVERITY are always encouraged to ask for more details and usage information, by posting any questions to the mailing list of the OS developers.

44.17 The OS Committers' Big List of Rules

1. Respect other committers.
2. Respect other contributors.
3. Discuss any significant change *before* committing.
4. Respect existing maintainers (if listed in the `MAINTAINER` field in `Makefile` or in `MAINTAINER` in the top-level directory).
5. Any disputed change must be backed out pending resolution of the dispute if requested by a maintainer. Security related changes may override a maintainer's wishes at the Security Officer's discretion.
6. Changes go to `OS.CURRENT` before `OS.STABLE` unless specifically permitted by the release engineer or unless they are not applicable to `OS.CURRENT`. Any non-trivial or non-urgent change which is applicable should also be allowed to sit in `OS.CURRENT` for at least 3 days before merging so that it can be given sufficient testing. The release engineer has the same authority over the `OS.STABLE` branch as outlined for the maintainer in rule #5.
7. Do not fight in public with other committers; it looks bad. If you must "strongly disagree" about something, do so only in private.
8. Respect all code freezes and read the `committers` and `developers` mailing lists in a timely manner so you know when a code freeze is in effect.
9. When in doubt on any procedure, ask first!
10. Test your changes before committing them.
11. Do not commit to anything under the `src/contrib`, `src/crypto`, or `src/sys/contrib` trees without *explicit* approval from the respective maintainer(s).

As noted, breaking some of these rules can be grounds for suspension or, upon repeated offense, permanent removal of commit privileges. Individual members of core have the power to temporarily suspend commit privileges until core as a whole has the chance to review the issue. In case of an "emergency" (a committer doing damage to the repository), a temporary suspension may also be done by the repository meisters. Only a 2/3 majority of core has the authority to suspend commit privileges for longer than a week or to remove them permanently. This rule does not exist to set core up as a bunch of cruel dictators who can dispose of committers as casually as empty soda cans, but to give the project a kind of safety fuse. If someone is out of control, it is important to be able to deal with this immediately rather than be paralyzed by debate. In all cases, a committer whose privileges are suspended or revoked is entitled to a "hearing" by core, the total duration of the suspension being determined at that time. A committer whose privileges are suspended may also request a review of the decision after 30 days and every 30 days thereafter (unless the total suspension period is less than 30 days). A committer whose privileges have been revoked entirely may request a review after a period of 6 months has elapsed. This review policy is *strictly informal* and, in all cases, core reserves the right to either act on or disregard requests for review if they feel their original decision to be the right one.

In all other aspects of project operation, core is a subset of committers and is bound by the *same rules*. Just because someone is in core this does not mean that they have special dispensation to step outside any of the lines painted here; core's "special powers" only kick in when it acts as a group, not on an individual basis. As individuals, the core team members are all committers first and core second.

44.17.1 Details

1. Respect other committers.

This means that you need to treat other committers as the peer-group developers that they are. Despite our occasional attempts to prove the contrary, one does not get to be a committer by being stupid and nothing rankles more than being treated that way by one of your peers. Whether we always feel respect for one another or not (and everyone has off days), we still have to *treat* other committers with respect at all times, on public forums and in private email.

Being able to work together long term is this project's greatest asset, one far more important than any set of changes to the code, and turning arguments about code into issues that affect our long-term ability to work harmoniously together is just not worth the trade-off by any conceivable stretch of the imagination.

To comply with this rule, do not send email when you are angry or otherwise behave in a manner which is likely to strike others as needlessly confrontational. First calm down, then think about how to communicate in the most effective fashion for convincing the other person(s) that your side of the argument is correct, do not just blow off some steam so you can feel better in the short term at the cost of a long-term flame war. Not only is this very bad “energy economics”, but repeated displays of public aggression which impair our ability to work well together will be dealt with severely by the project leadership and may result in suspension or termination of your commit privileges. The project leadership will take into account both public and private communications brought before it. It will not seek the disclosure of private communications, but it will take it into account if it is volunteered by the committers involved in the complaint.

All of this is never an option which the project's leadership enjoys in the slightest, but unity comes first. No amount of code or good advice is worth trading that away.

2. Respect other contributors.

You were not always a committer. At one time you were a contributor. Remember that at all times. Remember what it was like trying to get help and attention. Do not forget that your work as a contributor was very important to you. Remember what it was like. Do not discourage, belittle, or demean contributors. Treat them with respect. They are our committers in waiting. They are every bit as important to the project as committers. Their contributions are as valid and as important as your own. After all, you made many contributions before you became a committer. Always remember that.

Consider the points raised under ? and apply them also to contributors.

3. Discuss any significant change *before* committing.

The repository is not where changes should be initially submitted for correctness or argued over, that should happen first in the mailing lists and the commit should only happen once something resembling consensus has been reached. This does not mean that you have to ask permission before correcting every obvious syntax error or manual page misspelling, simply that you should try to develop a feel for when a proposed change is not quite such a no-brainer and requires some feedback first. People really do not mind sweeping changes if the result is something clearly better than what they had before, they just do not like being *surprized* by those changes. The very best way of making sure that you are on the right track is to have your code reviewed by one or more other committers.

When in doubt, ask for review!

4. Respect existing maintainers if listed.

Many parts of OS are not “owned” in the sense that any specific individual will jump up and yell if you commit a change to “their” area, but it still pays to check first. One convention we use is to put a maintainer line in the `Makefile` for any package or subtree which is being actively maintained by one or more people; see http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/developers-handbook/policies.html for documentation on this. Where sections of code have several maintainers, commits to affected areas by one maintainer need to be reviewed by at least one other maintainer. In cases where the “maintainer-ship” of something is not clear,

you can also look at the repository logs for the file(s) in question and see if someone has been working recently or predominantly in that area.

Other areas of OS fall under the control of someone who manages an overall category of OS evolution, such as internationalization or networking. See <http://www.FreeBSD.org/administration.html> for more information on this.

5. Any disputed change must be backed out pending resolution of the dispute if requested by a maintainer. Security related changes may override a maintainer's wishes at the Security Officer's discretion.

This may be hard to swallow in times of conflict (when each side is convinced that they are in the right, of course) but a version control system makes it unnecessary to have an ongoing dispute raging when it is far easier to simply reverse the disputed change, get everyone calmed down again and then try to figure out what is the best way to proceed. If the change turns out to be the best thing after all, it can be easily brought back. If it turns out not to be, then the users did not have to live with the bogus change in the tree while everyone was busily debating its merits. People *very* rarely call for back-outs in the repository since discussion generally exposes bad or controversial changes before the commit even happens, but on such rare occasions the back-out should be done without argument so that we can get immediately on to the topic of figuring out whether it was bogus or not.

6. Changes go to OS.CURRENT before OS.STABLE unless specifically permitted by the release engineer or unless they are not applicable to OS.CURRENT. Any non-trivial or non-urgent change which is applicable should also be allowed to sit in OS.CURRENT for at least 3 days before merging so that it can be given sufficient testing. The release engineer has the same authority over the OS.STABLE branch as outlined in rule #5.

This is another “do not argue about it” issue since it is the release engineer who is ultimately responsible (and gets beaten up) if a change turns out to be bad. Please respect this and give the release engineer your full cooperation when it comes to the OS.STABLE branch. The management of OS.STABLE may frequently seem to be overly conservative to the casual observer, but also bear in mind the fact that conservatism is supposed to be the hallmark of OS.STABLE and different rules apply there than in OS.CURRENT. There is also really no point in having OS.CURRENT be a testing ground if changes are merged over to OS.STABLE immediately. Changes need a chance to be tested by the OS.CURRENT developers, so allow some time to elapse before merging unless the OS.STABLE fix is critical, time sensitive or so obvious as to make further testing unnecessary (spelling fixes to manual pages, obvious bug/typo fixes, etc.) In other words, apply common sense.

Changes to the security branches (for example, `releng/9.3`) must be approved by a member of the A.SECURITY-OFFICER, or in some cases, by a member of the A.RE.

7. Do not fight in public with other committers; it looks bad. If you must “strongly disagree” about something, do so only in private.

This project has a public image to uphold and that image is very important to all of us, especially if we are to continue to attract new members. There will be occasions when, despite everyone's very best attempts at self-control, tempers are lost and angry words are exchanged. The best thing that can be done in such cases is to minimize the effects of this until everyone has cooled back down. That means that you should not air your angry words in public and you should not forward private correspondence to public mailing lists or aliases. What people say one-to-one is often much less sugar-coated than what they would say in public, and such communications therefore have no place there - they only serve to inflame an already bad situation. If the person sending you a flame-o-gram at least had the grace to send it privately, then have the grace to keep it private yourself. If you feel you are being unfairly treated by another developer, and it is causing you anguish, bring the matter up with core rather than taking it public. Core will do its best to play peace makers and get things back to sanity. In cases where the dispute involves a change to the codebase and the participants do not appear to be reaching an amicable agreement, core may appoint a mutually-agreeable third party to resolve the dispute. All parties involved must then agree to be bound by the decision reached by this third party.

8. Respect all code freezes and read the `committers` and `developers` mailing list on a timely basis so you know when a code freeze is in effect.

Committing unapproved changes during a code freeze is a really big mistake and committers are expected to

keep up-to-date on what is going on before jumping in after a long absence and committing 10 megabytes worth of accumulated stuff. People who abuse this on a regular basis will have their commit privileges suspended until they get back from the OS Happy Reeducation Camp we run in Greenland.

9. When in doubt on any procedure, ask first!

Many mistakes are made because someone is in a hurry and just assumes they know the right way of doing something. If you have not done it before, chances are good that you do not actually know the way we do things and really need to ask first or you are going to completely embarrass yourself in public. There is no shame in asking “how in the heck do I do this?” We already know you are an intelligent person; otherwise, you would not be a committer.

10. Test your changes before committing them.

This may sound obvious, but if it really were so obvious then we probably would not see so many cases of people clearly not doing this. If your changes are to the kernel, make sure you can still compile both GENERIC and LINT. If your changes are anywhere else, make sure you can still make world. If your changes are to a branch, make sure your testing occurs with a machine which is running that code. If you have a change which also may break another architecture, be sure and test on all supported architectures. Please refer to the [OS Internal Page](#) for a list of available resources. As other architectures are added to the OS supported platforms list, the appropriate shared testing resources will be made available.

11. Do not commit to anything under the `src/contrib`, `src/crypto`, and `src/sys/contrib` trees without *explicit* approval from the respective maintainer(s).

The trees mentioned above are for contributed software usually imported onto a vendor branch. Committing something there, even if it does not take the file off the vendor branch, may cause unnecessary headaches for those responsible for maintaining that particular piece of software. Thus, unless you have *explicit* approval from the maintainer (or you are the maintainer), do *not* commit there!

Please note that this does not mean you should not try to improve the software in question; you are still more than welcome to do so. Ideally, you should submit your patches to the vendor. If your changes are OS-specific, talk to the maintainer; they may be willing to apply them locally. But whatever you do, do *not* commit there by yourself!

Contact the A.CORE if you wish to take up maintainership of an unmaintained part of the tree.

44.17.2 Policy on Multiple Architectures

OS has added several new architecture ports during recent release cycles and is truly no longer an I386 centric operating system. In an effort to make it easier to keep OS portable across the platforms we support, core has developed the following mandate:

Our 32-bit reference platform is ARCH.I386, and our 64-bit reference platform is ARCH.SPARC64. Major design work (including major API and ABI changes) must prove itself on at least one 32-bit and at least one 64-bit platform, preferably the primary reference platforms, before it may be committed to the source tree.

The ARCH.I386 and ARCH.SPARC64 platforms were chosen due to being more readily available to developers and as representatives of more diverse processor and system designs - big versus little endian, register file versus register stack, different DMA and cache implementations, hardware page tables versus software TLB management etc.

The ARCH.IA64 platform has many of the same complications that ARCH.SPARC64 has, but is still limited in availability to developers.

We will continue to re-evaluate this policy as cost and availability of the 64-bit platforms change.

Developers should also be aware of our Tier Policy for the long term support of hardware architectures. The rules here are intended to provide guidance during the development process, and are distinct from the requirements for features

and architectures listed in that section. The Tier rules for feature support on architectures at release-time are more strict than the rules for changes during the development process.

44.17.3 Other Suggestions

When committing documentation changes, use a spell checker before committing. For all XML docs, verify that the formatting directives are correct by running `make lint` and `textproc/igor`.

For manual pages, run `sysutils/manck` and `textproc/igor` over the manual page to verify all of the cross references and file references are correct and that the man page has all of the appropriate `MLINKs` installed.

Do not mix style fixes with new functionality. A style fix is any change which does not modify the functionality of the code. Mixing the changes obfuscates the functionality change when asking for differences between revisions, which can hide any new bugs. Do not include whitespace changes with content changes in commits to `doc/`. The extra clutter in the diffs makes the translators' job much more difficult. Instead, make any style or whitespace changes in separate commits that are clearly labeled as such in the commit message.

44.17.4 Deprecating Features

When it is necessary to remove functionality from software in the base system the following guidelines should be followed whenever possible:

1. Mention is made in the manual page and possibly the release notes that the option, utility, or interface is deprecated. Use of the deprecated feature generates a warning.
2. The option, utility, or interface is preserved until the next major (point zero) release.
3. The option, utility, or interface is removed and no longer documented. It is now obsolete. It is also generally a good idea to note its removal in the release notes.

44.18 Support for Multiple Architectures

OS is a highly portable operating system intended to function on many different types of hardware architectures. Maintaining clean separation of Machine Dependent (MD) and Machine Independent (MI) code, as well as minimizing MD code, is an important part of our strategy to remain agile with regards to current hardware trends. Each new hardware architecture supported by OS adds substantially to the cost of code maintenance, toolchain support, and release engineering. It also dramatically increases the cost of effective testing of kernel changes. As such, there is strong motivation to differentiate between classes of support for various architectures while remaining strong in a few key architectures that are seen as the OS “target audience”.

44.18.1 Statement of General Intent

The OS Project targets “production quality commercial off-the-shelf (COTS) workstation, server, and high-end embedded systems”. By retaining a focus on a narrow set of architectures of interest in these environments, the OS Project is able to maintain high levels of quality, stability, and performance, as well as minimize the load on various support teams on the project, such as the ports team, documentation team, security officer, and release engineering teams. Diversity in hardware support broadens the options for OS consumers by offering new features and usage opportunities (such as support for 64-bit CPUs, use in embedded environments, etc.), but these benefits must always be carefully considered in terms of the real-world maintenance cost associated with additional platform support.

The OS Project differentiates platform targets into four tiers. Each tier includes a specification of the requirements for an architecture to be in that tier, as well as specifying the obligations of developers with regards to the platform. In addition, a policy is defined regarding the circumstances required to change the tier of an architecture.

44.18.2 Tier 1: Fully Supported Architectures

Tier 1 platforms are fully supported by the security officer, release engineering, and toolchain maintenance staff. New features added to the operating system must be fully functional across all Tier 1 architectures for every release (features which are inherently architecture-specific, such as support for hardware device drivers, may be exempt from this requirement). In general, all Tier 1 platforms must have build and Tinderbox support either in the FreeBSD.org cluster, or be easily available for all developers. Embedded platforms may substitute an emulator available in the OS cluster for actual hardware.

Tier 1 architectures are expected to be Production Quality with respects to all aspects of the OS operating system, including installation and development environments.

Tier 1 architectures are expected to be completely integrated into the source tree and have all features necessary to produce an entire system relevant for that target architecture. Tier 1 architectures generally have at least 6 active developers.

Tier 1 architectures are expected to be fully supported by the ports system. All the ports should build on a Tier 1 platform, or have the appropriate filters to prevent the inappropriate ones from building there. The packaging system must support all Tier 1 architectures. To ensure an architecture's Tier 1 status, proponents of that architecture must show that all relevant packages can be built on that platform.

Tier 1 embedded architectures must be able to cross-build packages on at least one other Tier 1 architecture. The packages must be the most relevant for the platform, but may be a non-empty subset of those that build natively.

Tier 1 architectures must be fully documented. All basic operations need to be covered by the handbook or other documents. All relevant integration documentation must also be integrated into the tree, or readily available.

Current Tier 1 platforms are ARCH.I386 and ARCH.AMD64.

44.18.3 Tier 2: Developmental Architectures

Tier 2 platforms are not supported by the security officer and release engineering teams. Platform maintainers are responsible for toolchain support in the tree. The toolchain maintainers are expected to work with the platform maintainers to refine these changes. Major new toolchain components are allowed to break support for Tier 2 architectures if the OS-local changes have not been incorporated upstream. The toolchain maintainers are expected to provide prompt review of any proposed changes and cannot block, through their inaction, changes going into the tree. New features added to OS should be feasible to implement on these platforms, but an implementation is not required before the feature may be added to the OS source tree. New features that may be difficult to implement on Tier 2 architectures should provide a means of disabling them on those architectures. The implementation of a Tier 2 architecture may be committed to the main OS tree as long as it does not interfere with production work on Tier 1 platforms, or substantially with other Tier 2 platforms. Before a Tier 2 platform can be added to the OS base source tree, the platform must be able to boot multi-user on actual hardware. Generally, there must be at least three active developers working on the platform.

Tier 2 architectures are usually systems targeted at Tier 1 support, but that are still under development. Architectures reaching end of life may also be moved from Tier 1 status to Tier 2 status as the availability of resources to continue to maintain the system in a Production Quality state diminishes. Well supported niche architectures may also be Tier 2.

Tier 2 architectures have basic support for them integrated into the ports infrastructure. They may have cross compilation support added, at the discretion of portmgr. Some ports must built natively into packages if the package system supports that architecture. If not integrated into the base system, some external patches for the architecture for ports must be available.

Tier 2 architectures can be integrated into the OS handbook. The basics for how to get a system running must be documented, although not necessarily for every single board or system a Tier 2 architecture supports. The supported hardware list must exist and should be relatively recent. It should be integrated into the OS documentation.

Current Tier 2 platforms are ARCH.ARM, ARCH.IA64 (through OS 10), ARCH.PC98, ARCH.POWERPC, and ARCH.SPARC64.

44.18.4 Tier 3: Experimental Architectures

Tier 3 platforms are not supported by the security officer and release engineering teams. At the discretion of the toolchain maintainers, they may be supported in the toolchain. Tier 3 platforms are architectures in the early stages of development, for non-mainstream hardware platforms, or which are considered legacy systems unlikely to see broad future use. Initial support for Tier 3 platforms should be worked on in external SCM repositories. The transition to OS's subversion should take place after the platform boots multi-user on hardware; sharing via subversion is needed for wider exposure; and multiple developers are actively working on the platform. Platforms that transition to Tier 3 status may be removed from the tree if they are no longer actively supported by the OS developer community at the discretion of the release engineer.

Tier 3 platforms may have ports support, either integrated or external, but do not require it.

Tier 3 platforms must have the basics documented for how to build a kernel and how to boot it on at least one target hardware or emulation environment. This documentation need not be integrated into the OS tree.

Current Tier 3 platforms are ARCH.MIPS.

44.18.5 Tier 4: Unsupported Architectures

Tier 4 systems are not supported in any form by the project.

All systems not otherwise classified into a support tier are Tier 4 systems. The ARCH.IA64 platform is transitioning to Tier 4 status in OS 11.

44.18.6 Policy on Changing the Tier of an Architecture

Systems may only be moved from one tier to another by approval of the OS Core Team, which shall make that decision in collaboration with the Security Officer, Release Engineering, and toolchain maintenance teams.

44.19 Ports Specific FAQ

44.19.1 Adding a New Port

Q: How do I add a new port?

A: First, please read the section about repository copies.

The easiest way to add a new port is the `addport` script located in the `ports/Tools/scripts` directory. It adds a port from the directory specified, determining the category automatically from the port `Makefile`. It also adds an entry to the port's category `Makefile`. It was written by A.MHARO.EMAIL, A.WILL.EMAIL, and A.GARGA.EMAIL. When sending questions about this script to the A.PORTS, please also CC A.CREES.EMAIL, the current maintainer.

Q: Any other things I need to know when I add a new port?

A: Check the port, preferably to make sure it compiles and packages correctly. This is the recommended sequence:

```
PROMPT.ROOT make install
PROMPT.ROOT make package
PROMPT.ROOT make deinstall
PROMPT.ROOT pkg add package you built above
PROMPT.ROOT make deinstall
PROMPT.ROOT make reinstall
PROMPT.ROOT make package
```

The Porters Handbook contains more detailed instructions.

Use MAN.PORTLINT.1 to check the syntax of the port. You do not necessarily have to eliminate all warnings but make sure you have fixed the simple ones.

If the port came from a submitter who has not contributed to the Project before, add that person's name to the Additional Contributors section of the OS Contributors List.

Close the PR if the port came in as a PR. To close a PR, change the state to “Issue

Resolved” and the resolution as Fixed.

44.19.2 Removing an Existing Port

Q: How do I remove an existing port?

A: First, please read the section about repository copies. Before you remove the port, you have to verify there are no other ports depending on it.

- Make sure there is no dependency on the port in the ports collection:
 - The port's PKGNAME should appear in exactly one line in a recent INDEX file.
 - No other ports should contain any reference to the port's directory or PKGNAME in their Makefiles
- Then, remove the port:

Remove the port's files and directory with `svn remove`.

Remove the SUBDIR listing of the port in the parent directory Makefile.

Add an entry to `ports/MOVED`.

Remove the port from `ports/LEGAL` if it is there.

Alternatively, you can use the `rmport` script, from `ports/Tools/scripts`. This script was written by A.VD.EMAIL. When sending questions about this script to the A.PORTS, please also CC A.CREES.EMAIL, the current maintainer.

44.19.3 Re-adding a Deleted Port

Q: How do I re-add a deleted port?

A: This is essentially the reverse of deleting a port.

Important

Do not use `svn add` to add the port. Follow these steps. If they are unclear, or are not working, ask for help, do not just `svn add` the port.

Figure out when the port was removed. Use this [list](#), or look for the port on [freshports](#), and then copy the last living revision of the port:

```
PROMPT.USER cd /usr/ports/category
PROMPT.USER svn cp 'svn+ssh://svn.freebsd.org/ports/head/category/portname/@XXXXXX' portname
```

Pick the revision that is just before the removal. For example, if the revision where it was removed is 269874, use 269873.

It is also possible to specify a date. In that case, pick a date that is before the removal but after the last commit to the port.

```
PROMPT.USER cd /usr/ports/category
PROMPT.USER svn cp 'svn+ssh://svn.freebsd.org/ports/head/category/portname/@{YYYY-MM-DD}' portname
```

Make the changes necessary to get the port working again. If it was deleted because the distfiles are no longer available, either volunteer to host the distfiles, or find someone else to do so.

If some files have been added, or were removed during the resurrection process, use “`svn`

`add`” or `svn remove` to make sure all the files in the

port will be committed.

Restore the `SUBDIR` listing of the port in the parent directory `Makefile`, keeping the entries sorted.

Delete the port entry from `ports/MOVED`.

If the port had an entry in `ports/LEGAL`, restore it.

`svn commit` these changes, preferably in one step.

Tip

The `addport` script mentioned in ? now detects when the port to add has previously existed, and attempts to handle all except the `ports/LEGAL` step automatically.

44.19.4 Repository Copies

Q: When do we need a repository copy?

A: When you want to add a port that is related to any port that is already in the tree in a separate directory, you have to do a repository copy. Here *related* means it is a different version or a slightly modified version. Examples are `print/ghostscript*` (different versions) and `x11-wm/windowmaker*` (English-only and internationalized version).

Another example is when a port is moved from one subdirectory to another, or when you want to change the name of a directory because the author(s) renamed their software even though it is a descendant of a port already in a tree.

Q: What do I need to do?

A: With Subversion, a repo copy can be done by any committer:

- Doing a repo copy:

Verify that the target directory does not exist.

Use `svn up` to make certain the original files, directories, and checkout information is current.

Use `svn move` or `svn copy` to do the repo copy.

Upgrade the copied port to the new version. Remember to add or change the `PKGNAMEPREFIX` or `PKGNAME_SUFFIX` so there are no duplicate ports with the same name. In some rare cases it may be necessary to change the `PORTNAME` instead of adding `PKGNAMEPREFIX` or `PKGNAME_SUFFIX`, but this should only be done when it is really needed — e.g., using an existing port as the base for a very similar program with a different name, or upgrading a port to a new upstream version which actually changes the distribution name,

like the transition from `textproc/libxml` to `textproc/libxml2`. In most cases, adding or changing `PKGNAMEPREFIX` or `PKGNAME_SUFFIX` should suffice.

Add the new subdirectory to the `SUBDIR` listing in the parent directory `Makefile`. You can run `make checksubdirs` in the parent directory to check this.

If the port changed categories, modify the `CATEGORIES` line of the port's `Makefile` accordingly

Add an entry to `ports/MOVED`, if you remove the original port.

Commit all changes on one commit.

- When removing a port:

Perform a thorough check of the ports collection for any dependencies on the old port location/name, and update them. Running `grep` on `INDEX` is not enough because some ports have dependencies enabled by compile-time options. A full `grep -r` of the ports collection is recommended.

Remove the old port and the old `SUBDIR` entry.

Add an entry to `ports/MOVED`.

- After repo moves (“rename” operations where a port is copied and the old location is removed):

Follow the same steps that are outlined in the previous two entries, to activate the new location of the port and remove the old one.

44.19.5 Ports Freeze

Q: What is a “ports freeze”?

A: A “ports freeze” was a restricted state the ports tree was put in before a release. It was used to ensure a higher quality for the packages shipped with a release. It usually lasted a couple of weeks. During that time, build problems were fixed, and the release packages were built. This practice is no longer used, as the packages for the releases are built from the current stable, quarterly branch. For more information on how to merge commits to the quarterly branch, see ?.

44.19.6 Creating a New Category

Q: What is the procedure for creating a new category?

A: Please see *Proposing a New Category* in the *Porter's Handbook*. Once that procedure has been followed and the PR has been assigned to A.PORTMGR, it is their decision whether or not to approve it. If they do, it is their responsibility to do the following:

Perform any needed moves. (This only applies to physical categories.)

Update the `VALID_CATEGORIES` definition in `ports/Mk/bsd.port.mk`.

Assign the PR back to you.

Q: What do I need to do to implement a new physical category?

A: Upgrade each moved port's `Makefile`. Do not connect the new category to the build yet.

To do this, you will need to:

Change the port's `CATEGORIES` (this was the point of the exercise, remember?) The new category should be listed *first*. This will help to ensure that the `PKG_ORIGIN` is correct.

Run a `make describe`. Since the top-level `make index` that you will be running in a few steps is an iteration of `make describe` over the entire ports hierarchy, catching any errors here will save you having to re-run that step later on.

If you want to be really thorough, now might be a good time to run `MAN.PORTLINT.1`.

Check that the `PKGORIGIN`s are correct. The ports system uses each port's `CATEGORIES` entry to create its `PKGORIGIN`, which is used to connect installed packages to the port directory they were built from. If this entry is wrong, common port tools like `MAN.PKG.VERSION.1` and `MAN.PORTUPGRADE.1` fail.

To do this, use the `chkorigin.sh` tool, as follows: `“env PORTSDIR=/path/to/ports sh -e /path/to/ports/Tools/scripts/chkorigin.sh“`. This will check

every port in the ports tree, even those not connected to the build, so you can run it directly after the move operation. Hint: do not forget to look at the `PKGORIGIN`s of any slave ports of the ports you just moved!

On your own local system, test the proposed changes: first, comment out the `SUBDIR` entries in the old ports' categories' Makefiles; then enable building the new category in `ports/Makefile`. Run `make checksubdirs` in the affected category directories to check the `SUBDIR` entries. Next, in the `ports/` directory, run `make index`. This can take over 40 minutes on even modern systems; however, it is a necessary step to prevent problems for other people.

Once this is done, you can commit the updated `ports/Makefile` to connect the new category to the build and also commit the Makefile changes for the old category or categories.

Add appropriate entries to `ports/MOVED`.

Update the documentation by modifying the following:

- the list of categories in the Porter's Handbook
- `doc/en_US.ISO8859-1/htdocs/ports`. Note that these are now displayed by sub-groups, as specified in `doc/en_US.ISO8859-1/htdocs/ports/categories.descriptions`.

(Note: these are in the docs, not the ports, repository). If you are not a docs committer, you will need to submit a PR for this.

Only once all the above have been done, and no one is any longer reporting problems with the new ports, should the old ports be deleted from their previous locations in the repository.

It is not necessary to manually update the ports web pages to reflect the new category. This is done automatically via the change to `en_US.ISO8859-1/htdocs/ports/categories` and the automated rebuild of `INDEX`.

Q: What do I need to do to implement a new virtual category?

A: This is much simpler than a physical category. You only need to modify the following:

- the list of categories in the Porter's Handbook
- `en_US.ISO8859-1/htdocs/ports/categories`

44.19.7 Miscellaneous Questions

Q: How do I know if my port is building correctly or not?

A: The packages are built multiple times each week. If a port fails, the maintainer will receive an email from `pkg-fallout@FreeBSD.org`.

Reports for all the package builds (official, experimental, and non-regression) are aggregated at pkg-status.FreeBSD.org.

Q: I added a new port. Do I need to add it to the `INDEX`?

A: No. The file can either be generated by running `make index`, or a pre-generated version can be downloaded with `make fetchindex`.

Q: Are there any other files I am not allowed to touch?

A: Any file directly under `ports/`, or any file under a subdirectory that starts with an uppercase letter (`Mk/`, `Tools/`, etc.). In particular, the A.PORTMGR is very protective of `ports/Mk/bsd.port*.mk` so do not commit changes to those files unless you want to face their wrath.

Q: What is the proper procedure for updating the checksum for a port's distfile when the file changes without a version change?

A: When the checksum for a port's distfile is updated due to the author updating the file without changing the port's revision, the commit message should include a summary of the relevant diffs between the original and new distfile to ensure that the distfile has not been corrupted or maliciously altered. If the current version of the port has been in the ports tree for a while, a copy of the old distfile will usually be available on the ftp servers; otherwise the author or maintainer should be contacted to find out why the distfile has changed.

Q: What is the procedure to request authorization for merging a commit to the quarterly branch?

A: When doing the commit, add the branch name to the MFH: line, for example:

```
MFH:      2014Q1
```

It will automatically notify A.PORTS-SECTEAM and A.PORTMGR. They will then decide if the commit can be merged and answer with the procedure.

If the commit has already been made, send an email to A.PORTS-SECTEAM and A.PORTMGR with the revision number and a small description of why the commit needs to be merged.

A script is provided to automate merging a specific commit: `ports/Tools/scripts/mfh`. It is used as follows:

```
PROMPT.USER /usr/ports/Tools/scripts/mfh 2015Q1 380362
U   2015Q1
Checked out revision 380443.
A   2015Q1/security
Updating '2015Q1/security/rubygem-sshkit':
A   2015Q1/security/rubygem-sshkit
A   2015Q1/security/rubygem-sshkit/Makefile
A   2015Q1/security/rubygem-sshkit/distinfo
A   2015Q1/security/rubygem-sshkit/pkg-descr
Updated to revision 380443.
--- Merging r380362 into '2015Q1':
U   2015Q1/security/rubygem-sshkit/Makefile
U   2015Q1/security/rubygem-sshkit/distinfo
--- Recording mergeinfo for merge of r380362 into '2015Q1':
U   2015Q1
--- Recording mergeinfo for merge of r380362 into '2015Q1/security':
G   2015Q1/security
--- Eliding mergeinfo from '2015Q1/security':
U   2015Q1/security
--- Recording mergeinfo for merge of r380362 into '2015Q1/security/rubygem-sshkit':
G   2015Q1/security/rubygem-sshkit
--- Eliding mergeinfo from '2015Q1/security/rubygem-sshkit':
U   2015Q1/security/rubygem-sshkit
M   2015Q1
M   2015Q1/security/rubygem-sshkit/Makefile
M   2015Q1/security/rubygem-sshkit/distinfo
Index: 2015Q1/security/rubygem-sshkit/Makefile
=====
--- 2015Q1/security/rubygem-sshkit/Makefile      (revision 380443)
```



```

+++ 2015Q1/security/rubygem-sshkit/Makefile      (working copy)
@@ -2,7 +2,7 @@
# $FreeBSD$

PORTNAME=      sshkit
-PORTVERSION=   1.6.1
+PORTVERSION=   1.7.0
CATEGORIES=     security rubygems
MASTER_SITES=  RG

Index: 2015Q1/security/rubygem-sshkit/distinfo
=====
--- 2015Q1/security/rubygem-sshkit/distinfo      (revision 380443)
+++ 2015Q1/security/rubygem-sshkit/distinfo      (working copy)
@@ -1,2 +1,2 @@
-SHA256 (rubygem/sshkit-1.6.1.gem) = 8ca67e46bb4ea50fdb0553cda77552f3e41b17a5aa919877d93875dfa22c03a
-SIZE (rubygem/sshkit-1.6.1.gem) = 135680
+SHA256 (rubygem/sshkit-1.7.0.gem) = 90effd1813363bae7355f4a45ebc8335a8ca74acc8d0933ba6ee6d40f281a2c
+SIZE (rubygem/sshkit-1.7.0.gem) = 136192
Index: 2015Q1
=====
--- 2015Q1      (revision 380443)
+++ 2015Q1      (working copy)

Property changes on: 2015Q1
_____
Modified: svn:mergeinfo
Merged /head:r380362
Do you want to commit? (no = start a shell) [y/n]

```

At that point, the script will either open a shell for you to fix things, or open your text editor with the commit message all prepared and then commit the merge.

The script assumes that you can connect to `svn.FreeBSD.org` with SSH directly, so if your local login name is different than your OS cluster account, you need a few lines in your `~/.ssh/config`:

```

Host svn.freebsd.org # Can be *.freebsd.org
  User freebsd-login

```

44.20 Issues Specific to Developers Who Are Not Committers

A few people who have access to the OS machines do not have commit bits. Almost all of this document will apply to these developers as well (except things specific to commits and the mailing list memberships that go with them). In particular, we recommend that you read:

- *Administrative Details*
- *Conventions*

Note

You should get your mentor to add you to the “Additional Contributors” (`doc/en_US.ISO8859-1/articles/contributors/contrib.additional.xml`), if you are not already listed there.

- *Developer Relations*
- *SSH Quick-Start Guide*

- *The OS Committers' Big List of Rules*

44.21 Information About GA

As of December 12, 2012, GA was enabled on the OS Project website to collect anonymized usage statistics regarding usage of the site. The information collected is valuable to the OS Documentation Project, in order to identify various problems on the OS website.

44.21.1 GA General Policy

The OS Project takes visitor privacy very seriously. As such, the OS Project website honors the “Do Not Track” header *before* fetching the tracking code from Google. For more information, please see the [OS Privacy Policy](#).

GA access is *not* arbitrarily allowed — access must be requested, voted on by the A.DOCENG, and explicitly granted.

Requests for GA data must include a specific purpose. For example, a valid reason for requesting access would be “to see the most frequently used web browsers when viewing OS web pages to ensure page rendering speeds are acceptable.”

Conversely, “to see what web browsers are most frequently used” (without stating *why*) would be rejected.

All requests must include the timeframe for which the data would be required. For example, it must be explicitly stated if the requested data would be needed for a timeframe covering a span of 3 weeks, or if the request would be one-time only.

Any request for GA data without a clear, reasonable reason beneficial to the OS Project will be rejected.

44.21.2 Data Available Through GA

A few examples of the types of GA data available include:

- Commonly used web browsers
- Page load times
- Site access by language

44.22 Miscellaneous Questions

Q: Why are trivial or cosmetic changes to files on a vendor branch a bad idea?

- From now on, every new vendor release of that file will need to have patches merged in by hand.
- From now on, every new vendor release of that file will need to have patches *verified* by hand.

Q: How do I add a new file to a branch?

A: To add a file onto a branch, simply checkout or update to the branch you want to add to and then add the file using the add operation as you normally would. This works fine for the `doc` and `ports` trees. The `src` tree uses SVN and requires more care because of the `mergeinfo` properties. See the [Subversion Primer](#) for details on how to perform an MFC.

Q: How do I access people.FreeBSD.org to put up personal or project information?

A: people.FreeBSD.org is the same as freefall.FreeBSD.org. Just create a `public_html` directory. Anything you place in that directory will automatically be visible under <http://people.FreeBSD.org/>.

Q: Where are the mailing list archives stored?

A: The mailing lists are archived under `/g/mail` which will show up as `/hub/g/mail` with MAN.PWD.1. This location is accessible from any machine on the OS cluster.

Q: I would like to mentor a new committer. What process do I need to follow?

A: See the [New Account Creation Procedure](#) document on the internal pages.

Q: Are there any perks of being an OS committer?

A: Recognition as a competent software engineer is the longest lasting value. In addition, getting a chance to work with some of the best people that every engineer would dream of meeting is a great perk!

OS committers can get a free 4-CD or DVD set at conferences from [OS Mall, Inc.](#).

In addition, developers may request a cloaked hostmask for their account on the Freenode IRC network in the form of `freebsd/developer/freefall name` or `freebsd/developer/NickServ name`. To request a cloak, send an email to `A.IRC.EMAIL` with your requested hostmask and NickServ account name.

Contributing to FreeBSD

Author JordanHubbard

Author SamLawrance

Author MarkLinimon

contributing So you want to contribute to OS? That is great! OS *relies* on the contributions of its user base to survive. Your contributions are not only appreciated, they are vital to OS's continued growth.

A large and growing number of international contributors, of greatly varying ages and areas of technical expertise, develop OS. There is always more work to be done than there are people available to do it, and more help is always appreciated.

As a volunteer, what you do is limited only by what you want to do. However, we do ask that you are aware of what other members of the OS community will expect of you. You may want to take this into account before deciding to volunteer.

The OS project is responsible for an entire operating system environment, rather than just a kernel or a few scattered utilities. As such, our `TODO` lists span a very wide range of tasks: from documentation, beta testing and presentation, to the system installer and highly specialized types of kernel development. People of any skill level, in almost any area, can almost certainly help the project.

Commercial entities engaged in FreeBSD-related enterprises are also encouraged to contact us. Do you need a special extension to make your product work? You will find us receptive to your requests, given that they are not too outlandish. Are you working on a value-added product? Please let us know! We may be able to work cooperatively on some aspect of it. The free software world is challenging many existing assumptions about how software is developed, sold, and maintained, and we urge you to at least give it a second look.

45.1 What Is Needed

The following list of tasks and sub-projects represents something of an amalgam of various `TODO` lists and user requests.

45.1.1 Ongoing Non-Programmer Tasks

Many people who are involved in FreeBSD are not programmers. The Project includes documentation writers, Web designers, and support people. All that these people need to contribute is an investment of time and a willingness to learn.

1. Read through the FAQ and Handbook periodically. If anything is badly explained, out of date or even just completely wrong, let us know. Even better, send us a fix (Docbook is not difficult to learn, but there is no objection to ASCII submissions).
2. Help translate FreeBSD documentation into your native language. If documentation already exists for your language, you can help translate additional documents or verify that the translations are up-to-date. First take a look at the Translations FAQ in the FreeBSD Documentation Project Primer. You are not committing yourself to translating every single FreeBSD document by doing this — as a volunteer, you can do as much or as little translation as you desire. Once someone begins translating, others almost always join the effort. If you only have the time or energy to translate one part of the documentation, please translate the installation instructions.
3. Read the A.QUESTIONS occasionally (or even regularly). It can be very satisfying to share your expertise and help people solve their problems; sometimes you may even learn something new yourself! These forums can also be a source of ideas for things to work on.

45.1.2 Ongoing Programmer Tasks

Most of the tasks listed here require either a considerable investment of time, or an in-depth knowledge of the FreeBSD kernel, or both. However, there are also many useful tasks which are suitable for “weekend hackers”.

1. If you run FreeBSD-CURRENT and have a good Internet connection, there is a machine `current.FreeBSD.org` which builds a full release once a day—every now and again, try to install the latest release from it and report any failures in the process.
2. Read the A.BUGS. There might be a problem you can comment constructively on or with patches you can test. Or you could even try to fix one of the problems yourself.
3. If you know of any bug fixes which have been successfully applied to -CURRENT but have not been merged into -STABLE after a decent interval (normally a couple of weeks), send the committer a polite reminder.
4. Move contributed software to `src/contrib` in the source tree.
5. Make sure code in `src/contrib` is up to date.
6. Build the source tree (or just part of it) with extra warnings enabled and clean up the warnings.
7. Fix warnings for ports which do deprecated things like using `gets()` or including `malloc.h`.
8. If you have contributed any ports and you had to make OS-specific changes, send your patches back to the original authors (this will make your life easier when they bring out the next version).
9. Get copies of formal standards like POSIX. Compare FreeBSD’s behavior to that required by the standard. If the behavior differs, particularly in subtle or obscure corners of the specification, send in a PR about it. If you are able, figure out how to fix it and include a patch in the PR. If you think the standard is wrong, ask the standards body to consider the question.
10. Suggest further tasks for this list!

45.1.3 Work through the PR Database

problem reports database The [FreeBSD PR list](#) shows all the current active problem reports and requests for enhancement that have been submitted by FreeBSD users. The PR database includes both programmer and non-programmer tasks. Look through the open PRs, and see if anything there takes your interest. Some of these might be very simple tasks that just need an extra pair of eyes to look over them and confirm that the fix in the PR is a good one. Others might be much more complex, or might not even have a fix included at all.

Start with the PRs that have not been assigned to anyone else. If a PR is assigned to someone else, but it looks like something you can handle, email the person it is assigned to and ask if you can work on it—they might already have a patch ready to be tested, or further ideas that you can discuss with them.

45.1.4 Ongoing Ports Tasks

The Ports Collection is a perpetual work in progress. We want to provide our users with an easy to use, up to date, high quality repository of third party software. We need people to donate some of their time and effort to help us achieve this goal.

Anyone can get involved, and there are lots of different ways to do so. Contributing to ports is an excellent way to help “give back” something to the project. Whether you are looking for an ongoing role, or a fun challenge for a rainy day, we would love to have your help!

There are a number of easy ways you can contribute to keeping the ports tree up to date and in good working order:

- Find some cool or useful software and create a port for it.
- There are a large number of ports that have no maintainer. Become a maintainer and *adopt a port*.
- If you have created or adopted a port, be aware of *what you need to do as a maintainer*.
- When you are looking for a quick challenge you could *fix a bug or a broken port*.

45.1.5 Pick one of the items from the Ideas page

The [OS list of projects and ideas for volunteers](#) is also available for people willing to contribute to the OS project. The list is being regularly updated and contains items for both programmers and non-programmers with information about each project.

45.2 How to Contribute

Contributions to the system generally fall into one or more of the following 5 categories:

45.2.1 Bug Reports and General Commentary

An idea or suggestion of *general* technical interest should be mailed to the A.HACKERS. Likewise, people with an interest in such things (and a tolerance for a *high* volume of mail!) may subscribe to the A.HACKERS. See The FreeBSD Handbook for more information about this and other mailing lists.

If you find a bug or are submitting a specific change, please report it using the [bug submission form](#). Try to fill-in each field of the bug report. Unless they exceed 65KB, include any patches directly in the report. If the patch is suitable to be applied to the source tree put [PATCH] in the synopsis of the report. When including patches, *do not* use cut-and-paste because cut-and-paste turns tabs into spaces and makes them unusable. When patches are a lot larger than 20KB, consider compressing them (eg. with MAN.GZIP.1 or MAN.BZIP2.1) prior to uploading them.

After filing a report, you should receive confirmation along with a tracking number. Keep this tracking number so that you can update us with details about the problem.

See also this article on how to write good problem reports.

45.2.2 Changes to the Documentation

documentation submissions Changes to the documentation are overseen by the A.DOC. Please look at the FreeBSD Documentation Project Primer for complete instructions. Send submissions and changes (even small ones are welcome!) using the same method any other bug report.

45.2.3 Changes to Existing Source Code

FreeBSD-CURRENT An addition or change to the existing source code is a somewhat trickier affair and depends a lot on how far out of date you are with the current state of FreeBSD development. There is a special on-going release of FreeBSD known as “FreeBSD-CURRENT” which is made available in a variety of ways for the convenience of developers working actively on the system. See The FreeBSD Handbook for more information about getting and using FreeBSD-CURRENT.

Working from older sources unfortunately means that your changes may sometimes be too obsolete or too divergent for easy re-integration into FreeBSD. Chances of this can be minimized somewhat by subscribing to the A.ANNOUNCE and the A.CURRENT lists, where discussions on the current state of the system take place.

Assuming that you can manage to secure fairly up-to-date sources to base your changes on, the next step is to produce a set of diffs to send to the FreeBSD maintainers. This is done with the MAN.DIFF.1 command.

The preferred MAN.DIFF.1 format for submitting patches is the unified output format generated by `diff -u`.

diff

```
PROMPT.USER diff -u oldfile newfile
```

or

```
PROMPT.USER diff -u -r -N olldir newdir
```

would generate a set of unified diffs for the given source file or directory hierarchy.

See MAN.DIFF.1 for more information.

Once you have a set of diffs (which you may test with the MAN.PATCH.1 command), you should submit them for inclusion with OS as a bug report. *Do not* just send the diffs to the A.HACKERS or they will get lost! We greatly appreciate your submission (this is a volunteer project!); because we are busy, we may not be able to address it immediately, but it will remain in the PR database until we do. Indicate your submission by including [PATCH] in the synopsis of the report.

If you feel it appropriate (e.g. you have added, deleted, or renamed files), bundle your changes into a tar file. Archives created with MAN.SHAR.1 are also welcome.

If your change is of a potentially sensitive nature, such as if you are unsure of copyright issues governing its further distribution then you should send it to A.CORE directly rather than submitting as a bug report. The A.CORE reaches a much smaller group of people who do much of the day-to-day work on FreeBSD. Note that this group is also *very busy* and so you should only send mail to them where it is truly necessary.

Please refer to MAN.INTRO.9 and MAN.STYLE.9 for some information on coding style. We would appreciate it if you were at least aware of this information before submitting code.

45.2.4 New Code or Major Value-Added Packages

In the case of a significant contribution of a large body work, or the addition of an important new feature to FreeBSD, it becomes almost always necessary to either send changes as tar files or upload them to a web or FTP site for other people to access. If you do not have access to a web or FTP site, ask on an appropriate FreeBSD mailing list for someone to host the changes for you.

When working with large amounts of code, the touchy subject of copyrights also invariably comes up. OS prefers free software licenses such as BSD or ISC. Copyleft licenses such as GPLv2 are sometimes permitted. The complete listing can be found on the core team licensing policy page.

45.2.5 Money or Hardware

We are always very happy to accept donations to further the cause of the FreeBSD Project and, in a volunteer effort like ours, a little can go a long way! Donations of hardware are also very important to expanding our list of supported peripherals since we generally lack the funds to buy such items ourselves.

Donating Funds

The FreeBSD Foundation is a non-profit, tax-exempt foundation established to further the goals of the FreeBSD Project. As a 501(c)3 entity, the Foundation is generally exempt from US federal income tax as well as Colorado State income tax. Donations to a tax-exempt entity are often deductible from taxable federal income.

Donations may be sent in check form to: The FreeBSD Foundation P.O. Box 20247, Boulder, CO 80308 USA

The FreeBSD Foundation is now able to accept donations through the web with PayPal. To place a donation, please visit the Foundation [web site](#).

More information about the FreeBSD Foundation can be found in [The FreeBSD Foundation – an Introduction](#). To contact the Foundation by email, write to bod@FreeBSDFoundation.org.

Donating Hardware

donations The FreeBSD Project happily accepts donations of hardware that it can find good use for. If you are interested in donating hardware, please contact the Donations Liaison Office.

45.3 Contributing to ports

45.3.1 Adopting an unmaintained port

Choosing an unmaintained port

Taking over maintainership of ports that are unmaintained is a great way to get involved. Unmaintained ports are only updated and fixed when somebody volunteers to work on them. There are a large number of unmaintained ports. It is a good idea to start with adopting a port that you use regularly.

Unmaintained ports have their MAINTAINER set to `ports@FreeBSD.org`. A list of unmaintained ports and their current errors and problem reports can be seen at the [OS Ports Monitoring System](#).

Some ports affect a large number of others due to dependencies and slave port relationships. Generally, we want people to have some experience before they maintain such ports.

You can find out whether or not a port has dependencies or slave ports by looking at a master index of ports called INDEX. (The name of the file varies by release of OS; for instance, INDEX-8.) Some ports have conditional dependencies that are not included in a default INDEX build. We expect you to be able to recognize such ports by looking through other ports' Makefiles.

How to adopt the port

First make sure you understand your *responsibilities as a maintainer*. Also read the Porter's Handbook. *Please do not commit yourself to more than you feel you can comfortably handle.*

You may request maintainership of any unmaintained port as soon as you wish. Simply set MAINTAINER to your own email address and send a PR (Problem Report) with the change. If the port has build errors or needs updating,

you may wish to include any other changes in the same PR. This will help because many committers are less willing to assign maintainership to someone who does not have a known track record with OS. Submitting PRs that fix build errors or update ports are the best ways to establish one.

File your PR with category `ports` and class `change-request`. A committer will examine your PR, commit the changes, and finally close the PR. Sometimes this process can take a little while (committers are volunteers, too :).

45.3.2 The challenge for port maintainers

This section will give you an idea of why ports need to be maintained and outline the responsibilities of a port maintainer.

Why ports require maintenance

Creating a port is a once-off task. Ensuring that a port is up to date and continues to build and run requires an ongoing maintenance effort. Maintainers are the people who dedicate some of their time to meeting these goals.

The foremost reason ports need maintenance is to bring the latest and greatest in third party software to the OS community. An additional challenge is to keep individual ports working within the Ports Collection framework as it evolves.

As a maintainer, you will need to manage the following challenges:

- **New software versions and updates..**

New versions and updates of existing ported software become available all the time, and these need to be incorporated into the Ports Collection in order to provide up-to-date software.

- **Changes to dependencies..**

If significant changes are made to the dependencies of your port, it may need to be updated so that it will continue to work correctly.

- **Changes affecting dependent ports..**

If other ports depend on a port that you maintain, changes to your port may require coordination with other maintainers.

- **Interaction with other users, maintainers and developers..**

Part of being a maintainer is taking on a support role. You are not expected to provide general support (but we welcome it if you choose to do so). What you should provide is a point of coordination for OS-specific issues regarding your ports.

- **Bug hunting..**

A port may be affected by bugs which are specific to OS. You will need to investigate, find, and fix these bugs when they are reported. Thoroughly testing a port to identify problems before they make their way into the Ports Collection is even better.

- **Changes to ports infrastructure and policy..**

Occasionally the systems that are used to build ports and packages are updated or a new recommendation affecting the infrastructure is made. You should be aware of these changes in case your ports are affected and require updating.

- **Changes to the base system..**

OS is under constant development. Changes to software, libraries, the kernel or even policy changes can cause flow-on change requirements to ports.

Maintainer responsibilities

Keep your ports up to date

This section outlines the process to follow to keep your ports up to date.

This is an overview. More information about upgrading a port is available in the Porter's Handbook.

Monitor the upstream vendor for new versions, updates and security fixes for the software. Announcement mailing lists or news web pages are useful for doing this. Sometimes users will contact you and ask when your port will be updated. If you are busy with other things or for any reason just cannot update it at the moment, ask if they will help you by submitting an update.

You may also receive automated email from the `OS Ports Version Check` informing you that a newer version of your port's distfile is available. More information about that system (including how to stop future emails) will be provided in the message.

When they become available, incorporate the changes into the port. You need to be able to generate a patch between the original port and your updated port.

Thoroughly review and test your changes:

- Build, install and test your port on as many platforms and architectures as you can. It is common for a port to work on one branch or platform and fail on another.
- Make sure your port's dependencies are complete. The recommended way of doing this is by installing your own ports tinderbox. See *resources* for more information.
- Check that the packing list is up to date. This involves adding in any new files and directories and removing unused entries.
- Verify your port using `MAN.PORTLINT.1` as a guide. See *resources* for important information about using portlint.
- Consider whether changes to your port might cause any other ports to break. If this is the case, coordinate the changes with the maintainers of those ports. This is especially important if your update changes the shared library version; in this case, at the very least, the dependent ports will need to get a `PORTREVISION` bump so that they will automatically be upgraded by automated tools such as portmaster or `MAN.PORTUPGRADE.1`.

Send your update by submitting a PR with an explanation of the changes and a patch containing the differences between the original port and the updated one. Please refer to Writing FreeBSD Problem Reports for information on how to write a really good PR.

Note

Please do not submit a `MAN.SHAR.1` archive of the entire port; instead, use `MAN.DIFF.1 -ruN`. In this way, committers can much more easily see exactly what changes are being made. The Porter's Handbook section on Upgrading has more information.

At some stage a committer will deal with your PR. It may take minutes, or it may take weeks — so please be patient.

If a committer finds a problem with your changes, they will most likely refer it back to you. A prompt response will help get your PR committed faster, and is better for maintaining a thread of conversation when trying to resolve any problems.

Your changes will be committed and your port will have been updated. The PR will then be closed by the committer. That's it!

Ensure your ports continue to build correctly

This section is about discovering and fixing problems that stop your ports from building correctly.

OS only guarantees that the Ports Collection works on the `-STABLE` branches. In theory, you should be able to get by with running the latest release of each stable branch (since the ABIs are not supposed to change) but if you can run the branch, that is even better.

Since the majority of OS installations run on PC-compatible machines (what is termed the `i386` architecture), we expect you to keep the port working on that architecture. We prefer that ports also work on the `amd64` architecture running native. It is completely fair to ask for help if you do not have one of these machines.

Note

The usual failure modes for non-`x86` machines are that the original programmers assumed that, for instance, pointers are `ints`, or that a relatively lax older gcc compiler was being used. More and more, application authors are reworking their code to remove these assumptions — but if the author is not actively maintaining their code, you may need to do this yourself.

These are the tasks you need to perform to ensure your port is able to be built:

Check your mail for mail from `pkg-fallout@FreeBSD.org` and the [distfiles scanner](#) to see if any of the port which are failing to build are out of date.

Once you are aware of a problem, collect information to help you fix it. Build errors reported by `pkg-fallout` are accompanied by logs which will show you where the build failed. If the failure was reported to you by a user, ask them to send you information which may help in diagnosing the problem, such as:

- Build logs
- The commands and options used to build the port (including options set in `/etc/make.conf`)
- A list of packages installed on their system as shown by `MAN.PKG.INFO.1`
- The version of OS they are running as shown by `MAN.UNAME.1` “-a”
- When their ports collection was last updated
- When their ports tree and `INDEX` was last updated

Unfortunately there is no straightforward process to follow to do this. Remember, though: if you are stuck, ask for help! The A.PORTS is a good place to start, and the upstream developers are often very helpful.

Just as with updating a port, you should now incorporate changes, review and test, submit your changes in a PR, and provide feedback if required.

In some cases, you will have to make patches to the port to make it run on FreeBSD. Some (but not all) upstream authors will accept such patches back into their code for the next release. If so, this may even help their users on other BSD-based systems as well and perhaps save duplicated effort. Please consider sending any applicable patches to the authors as a courtesy.

Investigate bug reports and PRs related to your port

This section is about discovering and fixing bugs.

OS-specific bugs are generally caused by assumptions about the build and runtime environments that do not apply to OS. You are less likely to encounter a problem of this type, but it can be more subtle and difficult to diagnose.

These are the tasks you need to perform to ensure your port continues to work as intended:

Bugs may be reported to you through email via the [Problem Report database](#). Bugs may also be reported directly to you by users.

You should respond to PRs and other reports within 14 days, but please try not to take that long. Try to respond as soon as possible, even if it is just to say you need some more time before you can work on the PR.

If you have not responded after 14 days, any committer may commit from a PR that you have not responded to via a `maintainer-timeout`.

If the person reporting the bug has not also provided a fix, you need to collect the information that will allow you to generate one.

If the bug is reproducible, you can collect most of the required information yourself. If not, ask the person who reported the bug to collect the information for you, such as:

- A detailed description of their actions, expected program behavior and actual behavior
- Copies of input data used to trigger the bug
- Information about their build and execution environment — for example, a list of installed packages and the output of `MAN.ENV.1`
- Core dumps
- Stack traces

Some bug reports may be incorrect. For example, the user may have simply misused the program; or their installed packages may be out of date and require updating. Sometimes a reported bug is not specific to OS. In this case report the bug to the upstream developers. If the bug is within your capabilities to fix, you can also patch the port so that the fix is applied before the next upstream release.

As with build errors, you will need to sort out a fix to the problem. Again, remember to ask if you are stuck!

Just as with updating a port, you should now incorporate changes, review and test, and submit your changes in a PR (or send a follow-up if a PR already exists for the problem). If another user has submitted changes in the PR, you can also send a follow-up saying whether or not you approve the changes.

Providing support

Part of being a maintainer is providing support — not for the software in general — but for the port and any OS-specific quirks and problems. Users may contact you with questions, suggestions, problems and patches. Most of the time their correspondence will be specific to OS.

Occasionally you may have to invoke your skills in diplomacy, and kindly point users seeking general support to the appropriate resources. Less frequently you will encounter a person asking why the `RPMs` are not up to date or how can they get the software to run under Foo Linux. Take the opportunity to tell them that your port is up to date (if it is, of course!), and suggest that they try OS.

Sometimes users and developers will decide that you are a busy person whose time is valuable and do some of the work for you. For example, they might:

- submit a PR or send you patches to update your port,
- investigate and perhaps provide a fix to a PR, or
- otherwise submit changes to your port.

In these cases your main obligation is to respond in a timely manner. Again, the timeout for non-responsive maintainers is 14 days. After this period changes may be committed unapproved. They have taken the trouble to do this for you; so please try to at least respond promptly. Then review, approve, modify or discuss their changes with them as soon as possible.

If you can make them feel that their contribution is appreciated (and it should be) you will have a better chance persuading them to do more things for you in the future :-).

45.3.3 Finding and fixing a broken port

There are two really good places to find a port that needs some attention.

You can use the [web interface](#) to the Problem Report database to search through and view unresolved PRs. The majority of ports PRs are updates, but with a little searching and skimming over synopses you should be able to find something interesting to work on (the `sw-bug` class is a good place to start).

The other place is the [OS Ports Monitoring System](#). In particular look for unmaintained ports with build errors and ports that are marked `BROKEN`. It is OK to send changes for a maintained port as well, but remember to ask the maintainer in case they are already working on the problem.

Once you have found a bug or problem, collect information, investigate and fix! If there is an existing PR, follow up to that. Otherwise create a new PR. Your changes will be reviewed and, if everything checks out, committed.

45.3.4 When to call it quits

As your interests and commitments change, you may find that you no longer have time to continue some (or all) of your ports contributions. That is fine! Please let us know if you are no longer using a port or have otherwise lost time or interest in being a maintainer. In this way we can go ahead and allow other people to try to work on existing problems with the port without waiting for your response. Remember, OS is a volunteer project, so if maintaining a port is no fun anymore, it is probably time to let someone else do it!

In any case, the Ports Management Team (`portmgr`) reserves the right to reset your maintainership if you have not actively maintained your port in some time. (Currently, this is set to 3 months.) By this, we mean that there are unresolved problems or pending updates that have not been worked on during that time.

45.3.5 Resources for ports maintainers and contributors

The Porter's Handbook is your hitchhiker's guide to the ports system. Keep it handy!

Writing FreeBSD Problem Reports describes how to best formulate and submit a PR. In 2005 more than eleven thousand ports PRs were submitted! Following this article will greatly assist us in reducing the time needed to handle your PRs.

The [Problem Report database](#).

The [FreeBSD Ports Monitoring System](#) can show you cross-referenced information about ports such as build errors and problem reports. If you are a maintainer you can use it to check on the build status of your ports. As a contributor you can use it to find broken and unmaintained ports that need to be fixed.

The [FreeBSD Ports distfile scanner](#) can show you ports for which the distfiles are not fetchable. You can check on your own ports or use it to find ports that need their `MASTER_SITES` updated.

`ports-mgmt/poudriere` is the most thorough way to test a port through the entire cycle of installation, packaging, and deinstallation. documentation is located at the [poudriere home page](#)

`MAN.PORTLINT.1` is an application which can be used to verify that your port conforms to many important stylistic and functional guidelines. `portlint` is a simple heuristic application, so you should use it *only as a guide*. If `portlint` suggests changes which seem unreasonable, consult the Porter's Handbook or ask for advice.

The A.PORTS is for general ports-related discussion. It is a good place to ask for help. You can [subscribe](#), or [read and search the list archives](#). Reading the archives of the A.PORTS-BUGS and the A.CVS-PORTS may also be of interest.

CUPS on FreeBSD

Author ChessGriffin

46.1 An Introduction to the Common Unix Printing System (CUPS)

printing CUPS CUPS, the Common UNIX Printing System, provides a portable printing layer for UNIX-based operating systems. It has been developed by Easy Software Products to promote a standard printing solution for all UNIX vendors and users.

CUPS uses the Internet Printing Protocol (IPP) as the basis for managing print jobs and queues. The Line Printer Daemon (LPD), Server Message Block (SMB), and AppSocket (aka JetDirect) protocols are also supported with reduced functionality. CUPS adds network printer browsing and PostScript Printer Description (PPD) based printing options to support real-world printing under UNIX. As a result, CUPS is ideally-suited for sharing and accessing printers in mixed environments of OS, LINUX, MACOS X, or WINDOWS.

The main site for CUPS is <http://www.cups.org/>.

46.2 Installing the CUPS Print Server

To install CUPS using a precompiled binary, issue the following command from a root terminal:

```
PROMPT.ROOT pkg install cups
```

Other optional, but recommended, packages are `print/gutenprint-cups` and `print/hplip`, both of which add drivers and utilities for a variety of printers. Once installed, the CUPS configuration files can be found in the directory `/usr/local/etc/cups`.

46.3 Configuring the CUPS Print Server

After installation, a few files must be edited in order to configure the CUPS server. First, create or modify, as the case may be, the file `/etc/devfs.rules` and add the following information to set the proper permissions on all potential printer devices and to associate printers with the cups user group:

```
[system=10]
add path 'unlpt*' mode 0660 group cups
add path 'ulpt*' mode 0660 group cups
add path 'lpt*' mode 0660 group cups
```

```
add path 'usb/X.Y.Z' mode 0660 group cups
```

****Note****

Note that X, Y, and Z should be replaced with the target USB device listed in the ```/dev/usb``` directory that corresponds to the printer. To find the correct device, examine the output of `MAN.DMESG.8`, where ```ugenX.Y``` lists the printer device, which is a symbolic link to a USB device in ```/dev/usb```.

Next, add two lines to `/etc/rc.conf` as follows:

```
cupsd_enable="YES"
devfs_system_ruleset="system"
```

These two entries will start the CUPS print server on boot and invoke the local devfs rule created above, respectively.

In order to enable CUPS printing under certain MICROSOFT.WINDOWS clients, the line below should be uncommented in `/usr/local/etc/cups/mime.types` and `/usr/local/etc/cups/mime.convs`:

```
application/octet-stream
```

Once these changes have been made, the `MAN.DEVFS.8` and CUPS systems must both be restarted, either by rebooting the computer or issuing the following two commands in a root terminal:

```
PROMPT.ROOT /etc/rc.d/devfs restart
PROMPT.ROOT /usr/local/etc/rc.d/cupsd restart
```

46.4 Configuring Printers on the CUPS Print Server

After the CUPS system has been installed and configured, the administrator can begin configuring the local printers attached to the CUPS print server. This part of the process is very similar, if not identical, to configuring CUPS printers on other UNIX-based operating systems, such as a LINUX distribution.

The primary means for managing and administering the CUPS server is through the web-based interface, which can be found by launching a web browser and entering `http://localhost:631` in the browser's URL bar. If the CUPS server is on another machine on the network, substitute the server's local IP address for `localhost`. The CUPS web interface is fairly self-explanatory, as there are sections for managing printers and print jobs, authorizing users, and more. Additionally, on the right-hand side of the Administration screen are several check-boxes allowing easy access to commonly-changed settings, such as whether to share published printers connected to the system, whether to allow remote administration of the CUPS server, and whether to allow users additional access and privileges to the printers and print jobs.

Adding a printer is generally as easy as clicking "Add Printer" at the Administration screen of the CUPS web interface, or clicking one of the "New Printers Found" buttons also at the Administration screen. When presented with the "Device" drop-down box, simply select the desired locally-attached printer, and then continue through the process. If one has added the `print/gutenprint-cups` or `print/hplip` ports or packages as referenced above, then additional print drivers will be available in the subsequent screens that might provide more stability or features.

46.5 Configuring CUPS Clients

Once the CUPS server has been configured and printers have been added and published to the network, the next step is to configure the clients, or the machines that are going to access the CUPS server. If one has a single desktop machine that is acting as both server and client, then much of this information may not be needed.

46.5.1 UNIX Clients

CUPS will also need to be installed on your UNIX clients. Once CUPS is installed on the clients, then CUPS printers that are shared across the network are often automatically discovered by the printer managers for various desktop environments such as GNOME or KDE. Alternatively, one can access the local CUPS interface on the client machine at <http://localhost:631> and click on “Add Printer” in the Administration section. When presented with the “Device” drop-down box, simply select the networked CUPS printer, if it was automatically discovered, or select `ipp` or `http` and enter the IPP or HTTP URI of the networked CUPS printer, usually in one of the two following syntaxes:

```
ipp://server-name-or-ip/printers/printername
```

```
http://server-name-or-ip:631/printers/printername
```

If the CUPS clients have difficulty finding other CUPS printers shared across the network, sometimes it is helpful to add or create a file `/usr/local/etc/cups/client.conf` with a single entry as follows:

```
ServerName server-ip
```

In this case, `server-ip` would be replaced by the local IP address of the CUPS server on the network.

46.5.2 WINDOWS Clients

Versions of WINDOWS prior to XP did not have the capability to natively network with IPP-based printers. However, WINDOWSXP and later versions do have this capability. Therefore, to add a CUPS printer in these versions of WINDOWS is quite easy. Generally, the WINDOWS administrator will run the WINDOWS Add Printer wizard, select “Network

Printer” and then enter the URI in the following syntax:

```
http://server-name-or-ip:631/printers/printername
```

If one has an older version of WINDOWS without native IPP printing support, then the general means of connecting to a CUPS printer is to use `net/samba3` and CUPS together, which is a topic outside the scope of this chapter.

46.6 CUPS Troubleshooting

Difficulties with CUPS often lies in permissions. First, double check the `MAN.DEVFS.8` permissions as outlined above. Next, check the actual permissions of the devices created in the file system. It is also helpful to make sure your user is a member of the `cups` group. If the permissions check boxes in the Administration section of the CUPS web interface do not seem to be working, another fix might be to manually backup the main CUPS configuration file located at `/usr/local/etc/cups/cupsd.conf` and edit the various configuration options and try different combinations of configuration options. One sample `/usr/local/etc/cups/cupsd.conf` to test is listed below. Please note that this sample `cupsd.conf` file sacrifices security for easier configuration; once the administrator successfully connects to the CUPS server and configures the clients, it is advisable to revisit this configuration file and begin locking down access.

```
# Log general information in error_log - change "info" to "debug" for
# troubleshooting...
LogLevel info

# Administrator user group...
SystemGroup wheel

# Listen for connections on Port 631.
Port 631
```

```
#Listen localhost:631
Listen /var/run/cups.sock

# Show shared printers on the local network.
Browsing On
BrowseOrder allow,deny
#BrowseAllow @LOCAL
BrowseAllow 192.168.1.* # change to local LAN settings
BrowseAddress 192.168.1.* # change to local LAN settings

# Default authentication type, when authentication is required...
DefaultAuthType Basic
DefaultEncryption Never # comment this line to allow encryption

# Allow access to the server from any machine on the LAN
<Location />
    Order allow,deny
    #Allow localhost
    Allow 192.168.1.* # change to local LAN settings
</Location>

# Allow access to the admin pages from any machine on the LAN
<Location /admin>
    #Encryption Required
    Order allow,deny
    #Allow localhost
    Allow 192.168.1.* # change to local LAN settings
</Location>

# Allow access to configuration files from any machine on the LAN
<Location /admin/conf>
    AuthType Basic
    Require user @SYSTEM
    Order allow,deny
    #Allow localhost
    Allow 192.168.1.* # change to local LAN settings
</Location>

# Set the default printer/job policies...
<Policy default>
    # Job-related operations must be done by the owner or an administrator...
    <Limit Send-Document Send-URI Hold-Job Release-Job Restart-Job Purge-Jobs \
Set-Job-Attributes Create-Job-Subscription Renew-Subscription Cancel-Subscription \
Get-Notifications Reprocess-Job Cancel-Current-Job Suspend-Current-Job Resume-Job \
CUPS-Move-Job>
        Require user @OWNER @SYSTEM
        Order deny,allow
    </Limit>

    # All administration operations require an administrator to authenticate...
    <Limit Pause-Printer Resume-Printer Set-Printer-Attributes Enable-Printer \
Disable-Printer Pause-Printer-After-Current-Job Hold-New-Jobs Release-Held-New-Jobs \
Deactivate-Printer Activate-Printer Restart-Printer Shutdown-Printer Startup-Printer \
Promote-Job Schedule-Job-After CUPS-Add-Printer CUPS-Delete-Printer CUPS-Add-Class \
CUPS-Delete-Class CUPS-Accept-Jobs CUPS-Reject-Jobs CUPS-Set-Default>
        AuthType Basic
        Require user @SYSTEM
        Order deny,allow
```

```
</Limit>

# Only the owner or an administrator can cancel or authenticate a job...
<Limit Cancel-Job CUPS-Authenticate-Job>
    Require user @OWNER @SYSTEM
    Order deny,allow
</Limit>

<Limit All>
    Order deny,allow
</Limit>
</Policy>
```

Explaining BSD

Author GregLehey

47.1 What is BSD?

BSD stands for “Berkeley Software Distribution”. It is the name of distributions of source code from the University of California, Berkeley, which were originally extensions to AT&T’s Research UNIX operating system. Several open source operating system projects are based on a release of this source code known as 4.4BSD-Lite. In addition, they comprise a number of packages from other Open Source projects, including notably the GNU project. The overall operating system comprises:

- The BSD kernel, which handles process scheduling, memory management, symmetric multi-processing (SMP), device drivers, etc.

Unlike the Linux kernel, there are several different BSD kernels with differing capabilities.

- The C library, the base API for the system.

The BSD C library is based on code from Berkeley, not the GNU project.

- Utilities such as shells, file utilities, compilers and linkers.

Some of the utilities are derived from the GNU project, others are not.

- The X Window system, which handles graphical display.

The X Window system used in most versions of BSD is maintained by the [X.Org project](#). OS allows the user to choose from a variety of desktop environments, such as Gnome, KDE, or Xfce; and lightweight window managers like Openbox, Fluxbox, or Awesome.

- Many other programs and utilities.

47.2 What, a real UNIX?

The BSD operating systems are not clones, but open source derivatives of AT&T’s Research UNIX operating system, which is also the ancestor of the modern UNIX System V. This may surprise you. How could that happen when AT&T has never released its code as open source?

It is true that AT&T UNIX is not open source, and in a copyright sense BSD is very definitely *not* UNIX, but on the other hand, AT&T has imported sources from other projects, noticeably the Computer Sciences Research Group (CSRG) of the University of California in Berkeley, CA. Starting in 1976, the CSRG started releasing tapes of their software, calling them *Berkeley Software Distribution* or *BSD*.

Initial BSD releases consisted mainly of user programs, but that changed dramatically when the CSRG landed a contract with the Defense Advanced Research Projects Agency (DARPA) to upgrade the communications protocols on their network, ARPANET. The new protocols were known as the *Internet Protocols*, later *TCP/IP* after the most important protocols. The first widely distributed implementation was part of 4.2BSD, in 1982.

In the course of the 1980s, a number of new workstation companies sprang up. Many preferred to license UNIX rather than developing operating systems for themselves. In particular, Sun Microsystems licensed UNIX and implemented a version of 4.2BSD, which they called SUNOS. When AT&T themselves were allowed to sell UNIX commercially, they started with a somewhat bare-bones implementation called System III, to be quickly followed by System V. The System V code base did not include networking, so all implementations included additional software from the BSD, including the TCP/IP software, but also utilities such as the *cs*h shell and the *vi* editor. Collectively, these enhancements were known as the *Berkeley Extensions*.

The BSD tapes contained AT&T source code and thus required a UNIX source license. By 1990, the CSRG's funding was running out, and it faced closure. Some members of the group decided to release the BSD code, which was Open Source, without the AT&T proprietary code. This finally happened with the *Networking Tape 2*, usually known as *Net/2*. *Net/2* was not a complete operating system: about 20% of the kernel code was missing. One of the CSRG members, William F. Jolitz, wrote the remaining code and released it in early 1992 as *386BSD*. At the same time, another group of ex-CSRG members formed a commercial company called [Berkeley Software Design Inc.](#) and released a beta version of an operating system called *BSD/386*, which was based on the same sources. The name of the operating system was later changed to *BSD/OS*.

386BSD never became a stable operating system. Instead, two other projects split off from it in 1993: [NetBSD](#) and [FreeBSD](#). The two projects originally diverged due to differences in patience waiting for improvements to 386BSD: the NetBSD people started early in the year, and the first version of FreeBSD was not ready until the end of the year. In the meantime, the code base had diverged sufficiently to make it difficult to merge. In addition, the projects had different aims, as we will see below. In 1996, [OpenBSD](#) split off from NetBSD, and in 2003, [DragonFlyBSD](#) split off from FreeBSD.

47.3 Why is BSD not better known?

For a number of reasons, BSD is relatively unknown:

1. The BSD developers are often more interested in polishing their code than marketing it.
2. Much of Linux's popularity is due to factors external to the Linux projects, such as the press, and to companies formed to provide Linux services. Until recently, the open source BSDs had no such proponents.
3. BSD developers tend to be more experienced than Linux developers, and have less interest in making the system easy to use. Newcomers tend to feel more comfortable with Linux.
4. In 1992, AT&T sued [BSDI](#), the vendor of BSD/386, alleging that the product contained AT&T-copyrighted code. The case was settled out of court in 1994, but the spectre of the litigation continues to haunt people. As recently as March 2000 an article published on the web claimed that the court case had been "recently settled".

One detail that the lawsuit did clarify is the naming: in the 1980s, BSD was known as "BSD UNIX". With the elimination of the last vestige of AT&T code from BSD, it also lost the right to the name UNIX. Thus you will see references in book titles to "the 4.3BSD UNIX operating system" and "the 4.4BSD operating system".

5. There is a perception that the BSD projects are fragmented and belligerent. The [Wall Street Journal](#) spoke of "balkanization" of the BSD projects. Like the law suit, this perception bases mainly on ancient history.

47.4 Comparing BSD and Linux

So what is really the difference between, say, Debian Linux and FreeBSD? For the average user, the difference is surprisingly small: Both are UNIX like operating systems. Both are developed by non-commercial projects (this does not apply to many other Linux distributions, of course). In the following section, we will look at BSD and compare it to Linux. The description applies most closely to FreeBSD, which accounts for an estimated 80% of the BSD installations, but the differences from NetBSD, OpenBSD and DragonFlyBSD are small.

47.4.1 Who owns BSD?

No one person or corporation owns BSD. It is created and distributed by a community of highly technical and committed contributors all over the world. Some of the components of BSD are Open Source projects in their own right and managed by different project maintainers.

47.4.2 How is BSD developed and updated?

The BSD kernels are developed and updated following the Open Source development model. Each project maintains a publicly accessible *source tree* under the [Concurrent Versions System](#) (CVS), which contains all source files for the project, including documentation and other incidental files. CVS allows users to “check out” (in other words, to extract a copy of) any desired version of the system.

A large number of developers worldwide contribute to improvements to BSD. They are divided into three kinds:

- Contributors write code or documentation. They are not permitted to commit (add code) directly to the source tree. In order for their code to be included in the system, it must be reviewed and checked in by a registered developer, known as a *committer*.
- Committers are developers with write access to the source tree. In order to become a committer, an individual must show ability in the area in which they are active.

It is at the individual committer’s discretion whether they should obtain authority before committing changes to the source tree. In general, an experienced committer may make changes which are obviously correct without obtaining consensus. For example, a documentation project committer may correct typographical or grammatical errors without review. On the other hand, developers making far-reaching or complicated changes are expected to submit their changes for review before committing them. In extreme cases, a core team member with a function such as Principal Architect may order that changes be removed from the tree, a process known as backing out. All committers receive mail describing each individual commit, so it is not possible to commit secretly.

- The Core team. FreeBSD and NetBSD each have a core team which manages the project. The core teams developed in the course of the projects, and their role is not always well-defined. It is not necessary to be a developer in order to be a core team member, though it is normal. The rules for the core team vary from one project to the other, but in general they have more say in the direction of the project than non-core team members have.

This arrangement differs from Linux in a number of ways:

1. No one person controls the content of the system. In practice, this difference is overrated, since the Principal Architect can require that code be backed out, and even in the Linux project several people are permitted to make changes.
2. On the other hand, there *is* a central repository, a single place where you can find the entire operating system sources, including all older versions.

3. BSD projects maintain the entire “Operating System”, not only the kernel. This distinction is only marginally useful: neither BSD nor Linux is useful without applications. The applications used under BSD are frequently the same as the applications used under Linux.
4. As a result of the formalized maintenance of a single CVS source tree, BSD development is clear, and it is possible to access any version of the system by release number or by date. CVS also allows incremental updates to the system: for example, the FreeBSD repository is updated about 100 times a day. Most of these changes are small.

47.4.3 BSD releases

FreeBSD, NetBSD and OpenBSD provide the system in three different “releases”. As with Linux, releases are assigned a number such as 1.4.1 or 3.5. In addition, the version number has a suffix indicating its purpose:

1. The development version of the system is called **CURRENT**. FreeBSD assigns a number to **CURRENT**, for example FreeBSD 5.0-CURRENT. NetBSD uses a slightly different naming scheme and appends a single-letter suffix which indicates changes in the internal interfaces, for example NetBSD 1.4.3G. OpenBSD does not assign a number (“OpenBSD-current”). All new development on the system goes into this branch.
2. At regular intervals, between two and four times a year, the projects bring out a **RELEASE** version of the system, which is available on CD-ROM and for free download from FTP sites, for example OpenBSD 2.6-RELEASE or NetBSD 1.4-RELEASE. The **RELEASE** version is intended for end users and is the normal version of the system. NetBSD also provides *patch releases* with a third digit, for example NetBSD 1.4.2.
3. As bugs are found in a **RELEASE** version, they are fixed, and the fixes are added to the CVS tree. In FreeBSD, the resultant version is called the **STABLE** version, while in NetBSD and OpenBSD it continues to be called the **RELEASE** version. Smaller new features can also be added to this branch after a period of test in the **CURRENT** branch.

By contrast, Linux maintains two separate code trees: the stable version and the development version. Stable versions have an even minor version number, such as 2.0, 2.2 or 2.4. Development versions have an odd minor version number, such as 2.1, 2.3 or 2.5. In each case, the number is followed by a further number designating the exact release. In addition, each vendor adds their own userland programs and utilities, so the name of the distribution is also important. Each distribution vendor also assigns version numbers to the distribution, so a complete description might be something like “TurboLinux 6.0 with kernel 2.2.14”

47.4.4 What versions of BSD are available?

In contrast to the numerous Linux distributions, there are only four major open source BSDs. Each BSD project maintains its own source tree and its own kernel. In practice, though, there appear to be fewer divergences between the userland code of the projects than there is in Linux.

It is difficult to categorize the goals of each project: the differences are very subjective. Basically,

- OS aims for high performance and ease of use by end users, and is a favourite of web content providers. It runs on a number of platforms and has significantly more users than the other projects.
- NetBSD aims for maximum portability: “of course it runs NetBSD”. It runs on machines from palmtops to large servers, and has even been used on NASA space missions. It is a particularly good choice for running on old non-INTEL hardware.
- OpenBSD aims for security and code purity: it uses a combination of the open source concept and rigorous code reviews to create a system which is demonstrably correct, making it the choice of security-conscious organizations such as banks, stock exchanges and US Government departments. Like NetBSD, it runs on a number of platforms.

- DragonFlyBSD aims for high performance and scalability under everything from a single-node UP system to a massively clustered system. DragonFlyBSD has several long-range technical goals, but focus lies on providing a SMP-capable infrastructure that is easy to understand, maintain and develop for.

There are also two additional BSD UNIX operating systems which are not open source, BSD/OS and Apple's MACOS X:

- BSD/OS was the oldest of the 4.4BSD derivatives. It was not open source, though source code licenses were available at relatively low cost. It resembled FreeBSD in many ways. Two years after the acquisition of BSDi by Wind River Systems, BSD/OS failed to survive as an independent product. Support and source code may still be available from Wind River, but all new development is focused on the VxWorks embedded operating system.
- **MACOS X** is the latest version of the operating system for APPLE's MAC line. The BSD core of this operating system, **Darwin**, is available as a fully functional open source operating system for x86 and PPC computers. The Aqua/Quartz graphics system and many other proprietary aspects of MACOS X remain closed-source, however. Several Darwin developers are also FreeBSD committers, and vice-versa.

47.4.5 How does the BSD license differ from the GNU Public license?

Linux is available under the **GNU General Public License** (GPL), which is designed to eliminate closed source software. In particular, any derivative work of a product released under the GPL must also be supplied with source code if requested. By contrast, the **BSD license** is less restrictive: binary-only distributions are allowed. This is particularly attractive for embedded applications.

47.4.6 What else should I know?

Since fewer applications are available for BSD than Linux, the BSD developers created a Linux compatibility package, which allows Linux programs to run under BSD. The package includes both kernel modifications, in order to correctly perform Linux system calls, and Linux compatibility files such as the C library. There is no noticeable difference in execution speed between a Linux application running on a Linux machine and a Linux application running on a BSD machine of the same speed.

The "all from one supplier" nature of BSD means that upgrades are much easier to handle than is frequently the case with Linux. BSD handles library version upgrades by providing compatibility modules for earlier library versions, so it is possible to run binaries which are several years old with no problems.

47.4.7 Which should I use, BSD or Linux?

What does this all mean in practice? Who should use BSD, who should use Linux?

This is a very difficult question to answer. Here are some guidelines:

- "If it ain't broke, don't fix it": If you already use an open source operating system, and you are happy with it, there is probably no good reason to change.
- BSD systems, in particular FreeBSD, can have notably higher performance than Linux. But this is not across the board. In many cases, there is little or no difference in performance. In some cases, Linux may perform better than FreeBSD.
- In general, BSD systems have a better reputation for reliability, mainly as a result of the more mature code base.
- BSD projects have a better reputation for the quality and completeness of their documentation. The various documentation projects aim to provide actively updated documentation, in many languages, and covering all aspects of the system.
- The BSD license may be more attractive than the GPL.

- BSD can execute most Linux binaries, while Linux can not execute BSD binaries. Many BSD implementations can also execute binaries from other UNIX like systems. As a result, BSD may present an easier migration route from other systems than Linux would.

47.4.8 Who provides support, service, and training for BSD?

BSDi / [FreeBSD Mall, Inc.](#) have been providing support contracts for FreeBSD for nearly a decade.

In addition, each of the projects has a list of consultants for hire: FreeBSD, [NetBSD](#), and [OpenBSD](#).

Filtering Bridges

Author AlexDupre

48.1 Why use a filtering bridge?

More and more frequently, thanks to the lowering costs of broad band Internet connections (xDSL) and also because of the reduction of available IPv4 addresses, many companies are connected to the Internet 24 hours on 24 and with few (sometimes not even a power of 2) IP addresses. In these situations it is often desirable to have a firewall that filters incoming and outgoing traffic from and towards Internet, but a packet filtering solution based on router may not be applicable, either due to subnetting issues, the router is owned by the connectivity supplier (ISP), or because it does not support such functionalities. In these scenarios the use of a filtering bridge is highly advised.

A bridge-based firewall can be configured and inserted between the xDSL router and your Ethernet hub/switch without any IP numbering issues.

48.2 How to Install

Adding bridge functionalities to a FreeBSD system is not difficult. Since 4.5 release it is possible to load such functionalities as modules instead of having to rebuild the kernel, simplifying the procedure a great deal. In the following subsections I will explain both installation ways.

Important

Do not follow both instructions: a procedure *excludes* the other one. Select the best choice according to your needs and abilities.

Before going on, be sure to have at least two Ethernet cards that support the promiscuous mode for both reception and transmission, since they must be able to send Ethernet packets with any address, not just their own. Moreover, to have a good throughput, the cards should be PCI bus mastering cards. The best choices are still the Intel ETHEREXPRESS Pro, followed by the TM.3COM 3c9xx series. To simplify the firewall configuration it may be useful to have two cards of different manufacturers (using different drivers) in order to distinguish clearly which interface is connected to the router and which to the inner network.

48.2.1 Kernel Configuration

So you have decided to use the older but well tested installation method. To begin, you have to add the following rows to your kernel configuration file:

```
options BRIDGE
options IPFWALL
options IPFWALL_VERBOSE
```

The first line is to compile the bridge support, the second one is the firewall and the third one is the logging functions of the firewall.

Now it is necessary to build and install the new kernel. You may find detailed instructions in the Building and Installing a Custom Kernel section of the FreeBSD Handbook.

48.2.2 Modules Loading

If you have chosen to use the new and simpler installation method, the only thing to do now is add the following row to `/boot/loader.conf`:

```
bridge_load="YES"
```

In this way, during the system startup, the `bridge.ko` module will be loaded together with the kernel. It is not required to add a similar row for the `ipfw.ko` module, since it will be loaded automatically after the execution of the steps in the following section.

48.3 Final Preparation

Before rebooting in order to load the new kernel or the required modules (according to the previously chosen installation method), you have to make some changes to the `/etc/rc.conf` configuration file. The default rule of the firewall is to reject all IP packets. Initially we will set up an `open` firewall, in order to verify its operation without any issue related to packet filtering (in case you are going to execute this procedure remotely, such configuration will avoid you to remain isolated from the network). Put these lines in `/etc/rc.conf`:

```
firewall_enable="YES"
firewall_type="open"
firewall_quiet="YES"
firewall_logging="YES"
```

The first row will enable the firewall (and will load the module `ipfw.ko` if it is not compiled in the kernel), the second one to set up it in `open` mode (as explained in `/etc/rc.firewall`), the third one to not show rules loading and the fourth one to enable logging support.

About the configuration of the network interfaces, the most used way is to assign an IP to only one of the network cards, but the bridge will work equally even if both interfaces or none has a configured IP. In the last case (IP-less) the bridge machine will be still more hidden, as inaccessible from the network: to configure it, you have to login from console or through a third network interface separated from the bridge. Sometimes, during the system startup, some programs require network access, say for domain resolution: in this case it is necessary to assign an IP to the external interface (the one connected to Internet, where DNS server resides), since the bridge will be activated at the end of the startup procedure. It means that the `fxp0` interface (in our case) must be mentioned in the `ifconfig` section of the `/etc/rc.conf` file, while the `xl0` is not. Assigning an IP to both the network cards does not make much sense, unless, during the start procedure, applications should access to services on both Ethernet segments.

There is another important thing to know. When running IP over Ethernet, there are actually two Ethernet protocols in use: one is IP, the other is ARP. ARP does the conversion of the IP address of a host into its Ethernet address (MAC layer). In order to allow the communication between two hosts separated by the bridge, it is necessary that the bridge will forward ARP packets. Such protocol is not included in the IP layer, since it exists only with IP over Ethernet. The FreeBSD firewall filters exclusively on the IP layer and therefore all non-IP packets (ARP included) will be forwarded without being filtered, even if the firewall is configured to not permit anything.

Now it is time to reboot the system and use it as before: there will be some new messages about the bridge and the firewall, but the bridge will not be activated and the firewall, being in `open` mode, will not avoid any operations.

If there are any problems, you should sort them out now before proceeding.

48.4 Enabling the Bridge

At this point, to enable the bridge, you have to execute the following commands (having the shrewdness to replace the names of the two network interfaces `fxp0` and `x10` with your own ones):

```
PROMPT.ROOT sysctl net.link.ether.bridge.config=fxp0:0,x10:0
PROMPT.ROOT sysctl net.link.ether.bridge.ipfw=1
PROMPT.ROOT sysctl net.link.ether.bridge.enable=1
```

The first row specifies which interfaces should be activated by the bridge, the second one will enable the firewall on the bridge and finally the third one will enable the bridge.

Note

If you have OS 5.1-RELEASE or previous the `sysctl` variables are spelled differently. See `MAN.BRIDGE.4` for details.

At this point you should be able to insert the machine between two sets of hosts without compromising any communication abilities between them. If so, the next step is to add the `net.link.ether.bridge.[blah]=[blah]` portions of these rows to the `/etc/sysctl.conf` file, in order to have them execute at startup.

48.5 Configuring The Firewall

Now it is time to create your own file with custom firewall rules, in order to secure the inside network. There will be some complication in doing this because not all of the firewall functionalities are available on bridged packets. Furthermore, there is a difference between the packets that are in the process of being forwarded and packets that are being received by the local machine. In general, incoming packets are run through the firewall only once, not twice as is normally the case; in fact they are filtered only upon receipt, so rules that use `out` or `xmit` will never match. Personally, I use `in via` which is an older syntax, but one that has a sense when you read it. Another limitation is that you are restricted to use only `pass` or `drop` commands for packets filtered by a bridge. Sophisticated things like `divert`, `forward` or `reject` are not available. Such options can still be used, but only on traffic to or from the bridge machine itself (if it has an IP address).

New in FreeBSD 4.0, is the concept of stateful filtering. This is a big improvement for UDP traffic, which typically is a request going out, followed shortly thereafter by a response with the exact same set of IP addresses and port numbers (but with source and destination reversed, of course). For firewalls that have no statekeeping, there is almost no way to deal with this sort of traffic as a single session. But with a firewall that can “remember” an outgoing UDP packet and, for the next few minutes, allow a response, handling UDP services is trivial. The following example shows how to do it. It is possible to do the same thing with TCP packets. This allows you to avoid some denial of service attacks and other nasty tricks, but it also typically makes your state table grow quickly in size.

Let’s look at an example setup. Note first that at the top of `/etc/rc.firewall` there are already standard rules for the loopback interface `lo0`, so we should not have to care for them anymore. Custom rules should be put in a separate file (say `/etc/rc.firewall.local`) and loaded at system startup, by modifying the row of `/etc/rc.conf` where we defined the open firewall:

```
firewall_type="/etc/rc.firewall.local"

**Important**
```

You have to specify the **full** path, otherwise it will not be loaded with the risk to remain isolated from the network.

For our example imagine to have the `fxp0` interface connected towards the outside (Internet) and the `xl0` towards the inside (LAN). The bridge machine has the IP 1.2.3.4 (it is not possible that your ISP can give you an address quite like this, but for our example it is good).

```
# Things that we have kept state on before get to go through in a hurry
add check-state

# Throw away RFC 1918 networks
add drop all from 10.0.0.0/8 to any in via fxp0
add drop all from 172.16.0.0/12 to any in via fxp0
add drop all from 192.168.0.0/16 to any in via fxp0

# Allow the bridge machine to say anything it wants
# (if the machine is IP-less do not include these rows)
add pass tcp from 1.2.3.4 to any setup keep-state
add pass udp from 1.2.3.4 to any keep-state
add pass ip from 1.2.3.4 to any

# Allow the inside hosts to say anything they want
add pass tcp from any to any in via xl0 setup keep-state
add pass udp from any to any in via xl0 keep-state
add pass ip from any to any in via xl0

# TCP section
# Allow SSH
add pass tcp from any to any 22 in via fxp0 setup keep-state
# Allow SMTP only towards the mail server
add pass tcp from any to relay 25 in via fxp0 setup keep-state
# Allow zone transfers only by the slave name server [dns2.nic.it]
add pass tcp from 193.205.245.8 to ns 53 in via fxp0 setup keep-state
# Pass ident probes. It is better than waiting for them to timeout
add pass tcp from any to any 113 in via fxp0 setup keep-state
# Pass the "quarantine" range
add pass tcp from any to any 49152-65535 in via fxp0 setup keep-state

# UDP section
# Allow DNS only towards the name server
add pass udp from any to ns 53 in via fxp0 keep-state
# Pass the "quarantine" range
add pass udp from any to any 49152-65535 in via fxp0 keep-state

# ICMP section
# Pass 'ping'
add pass icmp from any to any icmp types 8 keep-state
# Pass error messages generated by 'traceroute'
add pass icmp from any to any icmp types 3
add pass icmp from any to any icmp types 11

# Everything else is suspect
add drop log all from any to any
```

Those of you who have set up firewalls before may notice some things missing. In particular, there are no anti-spoofing rules, in fact we did *not* add:

```
add deny all from 1.2.3.4/8 to any in via fxp0
```

That is, drop packets that are coming in from the outside claiming to be from our network. This is something that you would commonly do to be sure that someone does not try to evade the packet filter, by generating nefarious packets that look like they are from the inside. The problem with that is that there is *at least* one host on the outside interface that you do not want to ignore: the router. But usually, the ISP anti-spoofs at their router, so we do not need to bother that much.

The last rule seems to be an exact duplicate of the default rule, that is, do not let anything pass that is not specifically allowed. But there is a difference: all suspected traffic will be logged.

There are two rules for passing SMTP and DNS traffic towards the mail server and the name server, if you have them. Obviously the whole rule set should be flavored to personal taste, this is only a specific example (rule format is described accurately in the MAN.IPFW.8 man page). Note that in order for “relay” and “ns” to work, name service lookups must work *before* the bridge is enabled. This is an example of making sure that you set the IP on the correct network card. Alternatively it is possible to specify the IP address instead of the host name (required if the machine is IP-less).

People that are used to setting up firewalls are probably also used to either having a `reset` or a `forward` rule for ident packets (TCP port 113). Unfortunately, this is not an applicable option with the bridge, so the best thing is to simply pass them to their destination. As long as that destination machine is not running an ident daemon, this is relatively harmless. The alternative is dropping connections on port 113, which creates some problems with services like IRC (the ident probe must timeout).

The only other thing that is a little weird that you may have noticed is that there is a rule to let the bridge machine speak, and another for internal hosts. Remember that this is because the two sets of traffic will take different paths through the kernel and into the packet filter. The inside net will go through the bridge, while the local machine will use the normal IP stack to speak. Thus the two rules to handle the different cases. The “in via

fxp0” rules work for both paths. In general, if you use

in via rules throughout the filter, you will need to make an exception for locally generated packets, because they did not come in via any of our interfaces.

48.6 Contributors

Many parts of this article have been taken, updated and adapted from an old text about bridging, edited by Nick Sayer. A pair of inspirations are due to an introduction on bridging by Steve Peterson.

A big thanks to Luigi Rizzo for the implementation of the bridge code in FreeBSD and for the time he has dedicated to me answering all of my related questions.

A thanks goes out also to Tom Rhodes who looked over my job of translation from Italian (the original language of this article) into English.

Fonts and FreeBSD

Author DaveBodenstab

49.1 Introduction

There are many sources of fonts available, and one might ask how they might be used with FreeBSD. The answer can be found by carefully searching the documentation for the component that one would like to use. This is very time consuming, so this tutorial is an attempt to provide a shortcut for others who might be interested.

49.2 Basic terminology

There are many different font formats and associated font file suffixes. A few that will be addressed here are:

.pfa, **.pfb** POSTSCRIPT type 1 fonts. The **.pfa** is the Ascii form and **.pfb** the *Binary* form.

.afm The font metrics associated with a type 1 font.

.pfm The printer font metrics associated with a type 1 font.

.ttf A TRUETYPE font

.fot An indirect reference to a TrueType font (not an actual font)

.fon, **.fnt** Bitmapped screen fonts

The **.fot** file is used by WINDOWS as sort of a symbolic link to the actual TRUETYPE font (**.ttf**) file. The **.fon** font files are also used by Windows. I know of no way to use this font format with FreeBSD.

49.3 What font formats can I use?

Which font file format is useful depends on the application being used. FreeBSD by itself uses no fonts. Application programs and/or drivers may make use of the font files. Here is a small cross reference of application/driver to the font type suffixes:

Driver

vt **.hex**

syscons **.fnt**

Application

Ghostscript .pfa, .pfb, .ttf

X11 .pfa, .pfb

Groff .pfa, .afm

Povray .ttf

The .fnt suffix is used quite frequently. I suspect that whenever someone wanted to create a specialized font file for their application, more often than not they chose this suffix. Therefore, it is likely that files with this suffix are not all the same format; specifically, the .fnt files used by syscons under FreeBSD may not be the same format as a .fnt file one encounters in the MS-DOS/WINDOWS environment. I have not made any attempt at using other .fnt files other than those provided with FreeBSD.

49.4 Setting a virtual console to 80x60 line mode

First, an 8x8 font must be loaded. To do this, /etc/rc.conf should contain the line (change the font name to an appropriate one for your locale):

```
font8x8="iso-8x8"      # font 8x8 from /usr/share/syscons/fonts/* (or NO).
```

The command to actually switch the mode is MAN.VIDCONTROL.1:

```
PROMPT.USER vidcontrol VGA_80x60
```

Various screen-oriented programs, such as MAN.VI.1, must be able to determine the current screen dimensions. As this is achieved this through ioctl calls to the console driver (such as MAN.SYSCONS.4) they will correctly determine the new screen dimensions.

To make this more seamless, one can embed these commands in the startup scripts so it takes place when the system boots. To do this is add this line to /etc/rc.conf

```
allscreens_flags="VGA_80x60" # Set this vidcontrol mode for all virtual screens
```

References: MAN.RC.CONF.5, MAN.VIDCONTROL.1.

49.5 Using type 1 fonts with X11

X11 can use either the .pfa or the .pfb format fonts. The X11 fonts are located in various subdirectories under /usr/X11R6/lib/X11/fonts. Each font file is cross referenced to its X11 name by the contents of the fonts.dir file in each directory.

There is already a directory named Type1. The most straight forward way to add a new font is to put it into this directory. A better way is to keep all new fonts in a separate directory and use a symbolic link to the additional font. This allows one to more easily keep track of ones fonts without confusing them with the fonts that were originally provided. For example:

```
Create a directory to contain the font files
PROMPT.USER mkdir -p /usr/local/share/fonts/type1
PROMPT.USER cd /usr/local/share/fonts/type1

Place the .pfa, .pfb and .afm files here
One might want to keep readme files, and other documentation
for the fonts here also
PROMPT.USER cp /cdrom/fonts/atm/showboat/showboat.pfb .
PROMPT.USER cp /cdrom/fonts/atm/showboat/showboat.afm .
```

Maintain an index to cross reference the fonts

```
PROMPT.USER echo showboat - InfoMagic CICA, Dec 1994, /fonts/atm/showboat >>INDEX
```

Now, to use a new font with X11, one must make the font file available and update the font name files. The X11 font names look like:

```
-bitstream-character-medium-r-normal-xxx-0-0-0-0-p-0-iso8859-1
|           |           |           |           |           |           |           \           \
|           |           |           |           |           |           |           +-----+- character set
|           |           |           |           |           |           |           +- average width
|           |           |           |           |           |           |           +- spacing
|           |           |           |           |           |           |           +- vertical res.
|           |           |           |           |           |           |           +- horizontal res.
|           |           |           |           |           |           |           +- points
|           |           |           |           |           |           |           +- pixels
|           |           |           |           |           |           |           \
foundry    family   weight    slant    width    additional style
```

A new name needs to be created for each new font. If you have some information from the documentation that accompanied the font, then it could serve as the basis for creating the name. If there is no information, then you can get some idea by using `MAN.STRING` on the font file. For example:

```
PROMPT: strings showboat.pfb | more
%!FontType1-1.0: Showboat 001.001
%%CreationDate: 1/15/91 5:16:03 PM
%%VMUsage: 1024 45747
% Generated by Fontographer 3.1
% Showboat
  1991 by David Rakowski.  Alle Rechte Vorbehalten.
FontDirectory/Showboat known{/Showboat findfont dup/UniqueID known{dup
/UniqueID get 4962377 eq exch/FontType get 1 eq and}{pop false}ifelse
{save true}{false}ifelse}{false}ifelse
12 dict begin
/FontInfo 9 dict dup begin
/version (001.001) readonly def
/FullName (Showboat) readonly def
/FamilyName (Showboat) readonly def
/Weight (Medium) readonly def
/ItalicAngle 0 def
/isFixedPitch false def
/UnderlinePosition -106 def
/UnderlineThickness 16 def
/Notice (Showboat
  1991 by David Rakowski.  Alle Rechte Vorbehalten.) readonly def
end readonly def
/FontName /Showboat def
--stdin--
```

Using this information, a possible name might be:

```
-type1-Showboat-medium-r-normal-decorative-0-0-0-0-p-0-iso8859-1
```

The components of our name are:

Foundry Lets just name all the new fonts `type1`.

Family The name of the font.

Weight Normal, bold, medium, semibold, etc. From the MAN.STRINGS.1 output above, it appears that this font has a weight of *medium*.

Slant *roman, italic, oblique*, etc. Since the *ItalicAngle* is zero, *roman* will be used.

Width Normal, wide, condensed, extended, etc. Until it can be examined, the assumption will be *normal*.

Additional style Usually omitted, but this will indicate that the font contains decorative capital letters.

Spacing proportional or monospaced. *Proportional* is used since *isFixedPitch* is false.

All of these names are arbitrary, but one should strive to be compatible with the existing conventions. A font is referenced by name with possible wild cards by an X11 program, so the name chosen should make some sense. One might begin by simply using `...-normal-r-normal-...-p-...` as the name, and then use `MAN.XFONTSEL.1` to examine it and adjust the name based on the appearance of the font.

So, to complete our example:

```
Make the font accessible to X11
PROMPT.USER cd /usr/X11R6/lib/X11/fonts/Type1
PROMPT.USER ln -s /usr/local/share/fonts/type1/showboat.pfb .

Edit fonts.dir and fonts.scale, adding the line describing the font
and incrementing the number of fonts which is found on the first line.
PROMPT.USER ex fonts.dir
:lp
25
:lc
26
.
:$a
showboat.pfb -type1-showboat-medium-r-normal-decorative-0-0-0-0-p-0-iso8859-1
.
:wq

fonts.scale seems to be identical to fonts.dir...
PROMPT.USER cp fonts.dir fonts.scale

Tell X11 that things have changed
PROMPT.USER xset fp rehash

Examine the new font
PROMPT.USER xfontsel -pattern -type1-*
```

References: `MAN.XFONTSEL.1`, `MAN.XSET.1`, *The X Windows System in a Nutshell*, O'Reilly & Associates.

49.6 Using type 1 fonts with Ghostscript

Ghostscript references a font via its `Fontmap` file. This must be modified in a similar way to the X11 `fonts.dir` file. Ghostscript can use either the `.pfa` or the `.pfb` format fonts. Using the font from the previous example, here is how to use it with Ghostscript:

```
Put the font in Ghostscript's font directory
PROMPT.USER cd /usr/local/share/ghostscript/fonts
PROMPT.USER ln -s /usr/local/share/fonts/type1/showboat.pfb .

Edit Fontmap so Ghostscript knows about the font
PROMPT.USER cd /usr/local/share/ghostscript/4.01
PROMPT.USER ex Fontmap
:$a
/Showboat          (showboat.pfb) ; % From CICA /fonts/atm/showboat
```

```
.
:wq

Use Ghostscript to examine the font
PROMPT.USER gs prfont.ps
Aladdin Ghostscript 4.01 (1996-7-10)
Copyright (C) 1996 Aladdin Enterprises, Menlo Park, CA. All rights
reserved.
This software comes with NO WARRANTY: see the file PUBLIC for details.
Loading Times-Roman font from /usr/local/share/ghostscript/fonts/tir____.pfb...
/1899520 581354 1300084 13826 0 done.
GS>Showboat DoFont
Loading Showboat font from /usr/local/share/ghostscript/fonts/showboat.pfb...
1939688 565415 1300084 16901 0 done.
>>showpage, press <return> to continue<<
>>showpage, press <return> to continue<<
>>showpage, press <return> to continue<<
GS>quit
```

References: `fonts.txt` in the Ghostscript 4.01 distribution

49.7 Using type 1 fonts with Groff

Now that the new font can be used by both X11 and Ghostscript, how can one use the new font with groff? First of all, since we are dealing with type 1 POSTSCRIPT fonts, the groff device that is applicable is the *ps* device. A font file must be created for each font that groff can use. A groff font name is just a file in `/usr/share/groff_font/devps`. With our example, the font file could be `/usr/share/groff_font/devps/SHOWBOAT`. The file must be created using tools provided by groff.

The first tool is `afmtodit`. This is not normally installed, so it must be retrieved from the source distribution. I found I had to change the first line of the file, so I did:

```
PROMPT.USER cp /usr/src/gnu/usr.bin/groff/afmtodit/afmtodit.pl /tmp
PROMPT.USER ex /tmp/afmtodit.pl
:lc
#!/usr/bin/perl -P-
.
:wq
```

This tool will create the groff font file from the metrics file (`.afm` suffix.) Continuing with our example:

```
Many .afm files are in Mac format... ^M delimited lines
We need to convert them to UNIX style ^J delimited lines
PROMPT.USER cd /tmp
PROMPT.USER cat /usr/local/share/fonts/type1/showboat.afm |
    tr '\015' '\012' >showboat.afm

Now create the groff font file
PROMPT.USER cd /usr/share/groff_font/devps
PROMPT.USER /tmp/afmtodit.pl -d DESC -e text.enc /tmp/showboat.afm generate/textmap SHOWBOAT
```

The font can now be referenced with the name `SHOWBOAT`.

If Ghostscript is used to drive the printers on the system, then nothing more needs to be done. However, if true POSTSCRIPT printers are used, then the font must be down loaded to the printer in order for the font to be used (unless the printer happens to have the showboat font built in or on an accessible font disk.) The final step is to create a down loadable font. The `pfbtops` tool is used to create the `.pfa` format of the font, and the download file is

modified to reference the new font. The download file must reference the internal name of the font. This can easily be determined from the groff font file as illustrated:

```
Create the .pfa font file
PROMPT.USER pfbtops /usr/local/share/fonts/type1/showboat.pfb >showboat.pfa
```

Of course, if the .pfa file is already available, just use a symbolic link to reference it.

```
Get the internal font name
PROMPT.USER fgrep internalname SHOWBOAT
internalname Showboat

Tell groff that the font must be down loaded
PROMPT.USER ex download
:$a
Showboat      showboat.pfa
.
:wq
```

To test the font:

```
PROMPT.USER cd /tmp
PROMPT.USER cat >example.t <<EOF
.sp 5
.ps 16
This is an example of the Showboat font:
.br
.ps 48
.vs (\n(.s+2)p
.sp
.ft SHOWBOAT
ABCDEFGHFI
.br
JKLMNOPQR
.br
STUVWXYZ
.sp
.ps 16
.vs (\n(.s+2)p
.fp 5 SHOWBOAT
.ft R
To use it for the first letter of a paragraph, it will look like:
.sp 50p
\s(48\f5H\s0fRere is the first sentence of a paragraph that uses the
showboat font as its first letter.
Additional vertical space must be used to allow room for the larger
letter.
EOF
PROMPT.USER groff -Tps example.t >example.ps

To use ghostscript/ghostview
PROMPT.USER ghostview example.ps

To print it
PROMPT.USER lpr -Ppostscript example.ps
```

References: /usr/src/gnu/usr.bin/groff/afmtodit/afmtodit.man, MAN.GROFF.FONT.5,
MAN.GROFF.CHAR.7, MAN.PFBTOPS.1.

49.8 Converting TrueType fonts to a groff/PostScript format for groff

This potentially requires a bit of work, simply because it depends on some utilities that are not installed as part of the base system. They are:

ttf2pf TrueType to PostScript conversion utilities. This allows conversion of a TrueType font to an ascii font metric (.afm) file.

Currently available at <http://sunsite.icm.edu.pl/pub/GUST/contrib/BachoTeX98/ttf2pf/>. Note: These files are PostScript programs and must be downloaded to disk by holding down the Shift key when clicking on the link. Otherwise, your browser may try to launch ghostview to view them.

The files of interest are:

- GS_TTF.PS
- PF2AFM.PS
- ttf2pf.ps

The funny upper/lower case is due to their being intended also for DOS shells. `ttf2pf.ps` makes use of the others as upper case, so any renaming must be consistent with this. (Actually, `GS_TTF.PS` and `PFS2AFM.PS` are supposedly part of the Ghostscript distribution, but it is just as easy to use these as an isolated utility. FreeBSD does not seem to include the latter.) You also may want to have these installed to `/usr/local/share/groff_font/devps(?)`.

afmtodit Creates font files for use with groff from ascii font metrics file. This usually resides in the directory, `/usr/src/contrib/groff/afmtodit`, and requires some work to get going.

Note

If you are paranoid about working in the `/usr/src` tree, simply copy the contents of the above directory to a work location.

In the work area, you will need to make the utility. Just type:

```
# make -f Makefile.sub afmtodit
```

You may also need to copy `/usr/contrib/groff/devps/generate/textmap` to `/usr/share/groff_font/devps/generate` if it does not already exist.

Once all these utilities are in place, you are ready to commence:

1. Create the .afm file by typing:

```
% gs -dNODISPLAY -q -- ttf2pf.ps TTF_name PS_font_name AFM_name
```

Where, `TTF_name` is your TrueType font file, `PS_font_name` is the file name for the .pfa file, `AFM_name` is the name you wish for the .afm file. If you do not specify output file names for the .pfa or .afm files, then default names will be generated from the TrueType font file name.

This also produces a .pfa file, the ascii PostScript font metrics file (.pfb is for the binary form). This will not be needed, but could (I think) be useful for a fontserver.

For example, to convert the 30f9 Barcode font using the default file names, use the following command:

```
% gs -dNODISPLAY -- ttf2pf.ps 30f9.ttf
Aladdin Ghostscript 5.10 (1997-11-23)
Copyright (C) 1997 Aladdin Enterprises, Menlo Park, CA. All rights reserved.
This software comes with NO WARRANTY: see the file PUBLIC for details.
Converting 30f9.ttf to 30f9.pfa and 30f9.afm.
```

If you want the converted fonts to be stored in `A.pfa` and `B.afm`, then use this command:

```
% gs -dNODISPLAY -- ttf2pf.ps 3of9.ttf A B
Aladdin Ghostscript 5.10 (1997-11-23)
Copyright (C) 1997 Aladdin Enterprises, Menlo Park, CA. All rights reserved.
This software comes with NO WARRANTY; see the file PUBLIC for details.
Converting 3of9.ttf to A.pfa and B.afm.
```

2. Create the groff PostScript file:

Change directories to `/usr/share/groff_font/devps` so as to make the following command easier to execute. You will probably need root privileges for this. (Or, if you are paranoid about working there, make sure you reference the files `DESC`, `text.enc` and `generate/textmap` as being in this directory.)

```
% afmtodit -d DESC -e text.enc file.afm \
generate/textmap PS_font_name
```

Where, `file.afm` is the `AFM_name` created by `ttf2pf.ps` above, and `PS_font_name` is the font name used from that command, as well as the name that `MAN.GROFF.1` will use for references to this font. For example, assuming you used the first `ttf2pf.ps` command above, then the 3of9 Barcode font can be created using the command:

```
% afmtodit -d DESC -e text.enc 3of9.afm \
generate/textmap 3of9
```

Ensure that the resulting `PS_font_name` file (e.g., `3of9` in the example above) is located in the directory `/usr/share/groff_font/devps` by copying or moving it there.

Note that if `ttf2pf.ps` assigns a font name using the one it finds in the TrueType font file and you want to use a different name, you must edit the `.afm` file prior to running `afmtodit`. This name must also match the one used in the Fontmap file if you wish to pipe `MAN.GROFF.1` into `MAN.GS.1`.

49.9 Can TrueType fonts be used with other programs?

The TrueType font format is used by Windows, Windows 95, and Mac's. It is quite popular and there are a great number of fonts available in this format.

Unfortunately, there are few applications that I am aware of that can use this format: Ghostscript and Povray come to mind. Ghostscript's support, according to the documentation, is rudimentary and the results are likely to be inferior to type 1 fonts. Povray version 3 also has the ability to use TrueType fonts, but I rather doubt many people will be creating documents as a series of raytraced pages :-).

This rather dismal situation may soon change. The [FreeType Project](#) is currently developing a useful set of FreeType tools:

- The `xfstt` font server for X11 can serve TrueType fonts in addition to regular fonts. Though currently in beta, it is said to be quite usable. See [Juliusz Chroboczek's page](#) for further information. Porting instructions for FreeBSD can be found at [Stephen Montgomery's software page](#).
- `xfstt` is another font server for X11, available under '`ftp://sunsite.unc.edu/pub/Linux/X11/fonts/` < `ftp://sunsite.unc.edu/pub/Linux/X11/fonts/` > '___.
- A program called `ttf2bdf` can produce BDF files suitable for use in an X environment from TrueType files. Linux binaries are said to be available from [ftp://crl.nmsu.edu/CLR/multiling/General/](#).
- and others ...

49.10 Where can additional fonts be obtained?

Many fonts are available on the Internet. They are either entirely free, or are share-ware. In addition many fonts are available in the `x11-fonts/` category in the ports collection

49.11 Additional questions

- What use are the `.pfm` files?
- Can one generate the `.afm` file from a `.pfa` or `.pfb`?
- How to generate the groff character mapping files for PostScript fonts with non-standard character names?
- Can `xditview` and `devX??` devices be set up to access all the new fonts?
- It would be good to have examples of using TrueType fonts with `Povray` and `Ghostscript`.

How to get best results from the FreeBSD-questions mailing list

Author GregLehey

50.1 Introduction

FreeBSD-questions is a mailing list maintained by the FreeBSD project to help people who have questions about the normal use of FreeBSD. Another group, FreeBSD-hackers, discusses more advanced questions such as future development work.

Note

The term “hacker” has nothing to do with breaking into other people’s computers. The correct term for the latter activity is “cracker”, but the popular press has not found out yet. The FreeBSD hackers disapprove strongly of cracking security, and have nothing to do with it. For a longer description of hackers, see Eric Raymond’s [How To Become A Hacker](#)

This is a regular posting aimed to help both those seeking advice from FreeBSD-questions (the “newcomers”), and also those who answer the questions (the “hackers”).

Inevitably there is some friction, which stems from the different viewpoints of the two groups. The newcomers accuse the hackers of being arrogant, stuck-up, and unhelpful, while the hackers accuse the newcomers of being stupid, unable to read plain English, and expecting everything to be handed to them on a silver platter. Of course, there is an element of truth in both these claims, but for the most part these viewpoints come from a sense of frustration.

In this document, I would like to do something to relieve this frustration and help everybody get better results from FreeBSD-questions. In the following section, I recommend how to submit a question; after that, we will look at how to answer one.

50.2 How to subscribe to FreeBSD-questions

FreeBSD-questions is a mailing list, so you need mail access. Point your WWW browser to the information page of the FreeBSD-questions mailing list. In the section titled “Subscribing to freebsd-questions” fill in the “Your email address” field; the other fields are optional.

Note

The password fields in the subscription form provide only mild security, but should prevent others from messing with your subscription. *Do not use a valuable password* as it will occasionally be emailed back to you in cleartext.

You will receive a confirmation message from mailman; follow the included instructions to complete your subscription.

Finally, when you get the “Welcome” message from mailman telling you the details of the list and subscription area password, *please save it*. If you ever should want to leave the list, you will need the information there. See the next section for more details.

50.3 How to unsubscribe from FreeBSD-questions

When you subscribed to FreeBSD-questions, you got a welcome message from mailman. In this message, amongst other things, it told you how to unsubscribe. Here is a typical message:

```
Welcome to the freebsd-questions@freebsd.org mailing list!

To post to this list, send your email to:

    freebsd-questions@freebsd.org

General information about the mailing list is at:

    http://lists.freebsd.org/mailman/listinfo/freebsd-questions

If you ever want to unsubscribe or change your options (e.g., switch to
or from digest mode, change your password, etc.), visit your
subscription page at:

http://lists.freebsd.org/mailman/options/freebsd-questions/grog%40lemsi.de

You can also make such adjustments via email by sending a message to:

    freebsd-questions-request@freebsd.org

with the word 'help' in the subject or body (don't include the
quotes), and you will get back a message with instructions.

You must know your password to change your options (including changing
the password, itself) or to unsubscribe. It is:

    12345

Normally, Mailman will remind you of your freebsd.org mailing list
passwords once every month, although you can disable this if you
prefer. This reminder will also include instructions on how to
unsubscribe or change your account options. There is also a button on
your options page that will email your current password to you.
```

From the URL specified in your “Welcome” message you may visit the “Account management page” and enter a request to “Unsubscribe” you from FreeBSD-questions mailing list.

A confirmation message will be sent to you from mailman; follow the included instructions to finish unsubscribing.

If you have done this, and you still can not figure out what is going on, send a message to freebsd-questions-request@FreeBSD.org, and they will sort things out for you. *Do not* send a message to FreeBSD-questions: they can not help you.

50.4 Should I ask `-questions` or `-hackers`?

Two mailing lists handle general questions about FreeBSD, `FreeBSD-questions` and `FreeBSD-hackers`. In some cases, it is not really clear which group you should ask. The following criteria should help for 99% of all questions, however:

1. If the question is of a general nature, ask `FreeBSD-questions`. Examples might be questions about installing FreeBSD or the use of a particular UNIX utility.
2. If you think the question relates to a bug, but you are not sure, or you do not know how to look for it, send the message to `FreeBSD-questions`.
3. If the question relates to a bug, and you are *sure* that it is a bug (for example, you can pinpoint the place in the code where it happens, and you maybe have a fix), then send the message to `FreeBSD-hackers`.
4. If the question relates to enhancements to FreeBSD, and you can make suggestions about how to implement them, then send the message to `FreeBSD-hackers`.

There are also a number of other specialized mailing lists, which caters to more specific interests. The criteria above still apply, and it is in your interest to stick to them, since you are more likely to get good results that way.

50.5 Before submitting a question

You can (and should) do some things yourself before asking a question on one of the mailing lists:

- Try solving the problem on your own. If you post a question which shows that you have tried to solve the problem, your question will generally attract more positive attention from people reading it. Trying to solve the problem yourself will also enhance your understanding of FreeBSD, and will eventually let you use your knowledge to help others by answering questions posted to the mailing lists.
- Read the manual pages, and the FreeBSD documentation (either installed in `/usr/doc` or accessible via WWW at <http://www.FreeBSD.org>), especially the handbook and the FAQ.
- Browse and/or search the archives for the mailing list, to see if your question or a similar one has been asked (and possibly answered) on the list. You can browse and/or search the mailing list archives at <http://www.FreeBSD.org/mail> and <http://www.FreeBSD.org/search/search.html#mailinglists> respectively. This can be done at other WWW sites as well, for example at <http://marc.theaimsgroup.com>.
- Use a search engine such as [Google](#) or [Yahoo](#) to find answers to your question. Google even has a [BSD-specific search interface](#).

50.6 How to submit a question

When submitting a question to `FreeBSD-questions`, consider the following points:

- Remember that nobody gets paid for answering a FreeBSD question. They do it of their own free will. You can influence this free will positively by submitting a well-formulated question supplying as much relevant information as possible. You can influence this free will negatively by submitting an incomplete, illegible, or rude question. It is perfectly possible to send a message to `FreeBSD-questions` and not get an answer even if you follow these rules. It is much more possible to not get an answer if you do not. In the rest of this document, we will look at how to get the most out of your question to `FreeBSD-questions`.
- Not everybody who answers FreeBSD questions reads every message: they look at the subject line and decide whether it interests them. Clearly, it is in your interest to specify a subject. “FreeBSD problem” or “Help”

are not enough. If you provide no subject at all, many people will not bother reading it. If your subject is not specific enough, the people who can answer it may not read it.

- Format your message so that it is legible, and PLEASE DO NOT SHOUT!!!!. We appreciate that a lot of people do not speak English as their first language, and we try to make allowances for that, but it is really painful to try to read a message written full of typos or without any line breaks.

Do not underestimate the effect that a poorly formatted mail message has, not just on the FreeBSD-questions mailing list. Your mail message is all people see of you, and if it is poorly formatted, one line per paragraph, badly spelt, or full of errors, it will give people a poor impression of you.

A lot of badly formatted messages come from [bad mailers](#) or [badly configured mailers](#). The following mailers are known to send out badly formatted messages without you finding out about them:

- EUDORA
- exmh
- MICROSOFT Exchange
- MICROSOFT OUTLOOK

Try not to use MIME: a lot of people use mailers which do not get on very well with MIME.

- Make sure your time and time zone are set correctly. This may seem a little silly, since your message still gets there, but many of the people you are trying to reach get several hundred messages a day. They frequently sort the incoming messages by subject and by date, and if your message does not come before the first answer, they may assume they missed it and not bother to look.
- Do not include unrelated questions in the same message. Firstly, a long message tends to scare people off, and secondly, it is more difficult to get all the people who can answer all the questions to read the message.
- Specify as much information as possible. This is a difficult area, and we need to expand on what information you need to submit, but here is a start:
 - In nearly every case, it is important to know the version of FreeBSD you are running. This is particularly the case for FreeBSD-CURRENT, where you should also specify the date of the sources, though of course you should not be sending questions about -CURRENT to FreeBSD-questions.
 - With any problem which *could* be hardware related, tell us about your hardware. In case of doubt, assume it is possible that it is hardware. What kind of CPU are you using? How fast? What motherboard? How much memory? What peripherals?

There is a judgement call here, of course, but the output of the MAN.DMESG.8 command can frequently be very useful, since it tells not just what hardware you are running, but what version of FreeBSD as well.

- If you get error messages, do not say “I get error messages”, say (for example) “I get the error message ‘No route to host’”.
 - If your system panics, do not say “My system panicked”, say (for example) “my system panicked with the message ‘free vnode isn’t’”.
 - If you have difficulty installing FreeBSD, please tell us what hardware you have. In particular, it is important to know the IRQs and I/O addresses of the boards installed in your machine.
 - If you have difficulty getting PPP to run, describe the configuration. Which version of PPP do you use? What kind of authentication do you have? Do you have a static or dynamic IP address? What kind of messages do you get in the log file?
- A lot of the information you need to supply is the output of programs, such as MAN.DMESG.8, or console messages, which usually appear in `/var/log/messages`. Do not try to copy this information by typing it in again; it is a real pain, and you are bound to make a mistake. To send log file contents, either make a copy of

the file and use an editor to trim the information to what is relevant, or cut and paste into your message. For the output of programs like MAN.DMESG.8, redirect the output to a file and include that. For example,

```
PROMPT.USER dmesg > /tmp/dmesg.out
```

This redirects the information to the file `/tmp/dmesg.out`.

- If you do all this, and you still do not get an answer, there could be other reasons. For example, the problem is so complicated that nobody knows the answer, or the person who does know the answer was offline. If you do not get an answer after, say, a week, it might help to re-send the message. If you do not get an answer to your second message, though, you are probably not going to get one from this forum. Resending the same message again and again will only make you unpopular.

To summarize, let's assume you know the answer to the following question (yes, it is the same one in each case). You choose which of these two questions you would be more prepared to answer:

```
Subject: HELP!!?!??
I just can't get hits damn silly FereBSD system to
workd, and Im really good at this tsuff, but I have never seen
anythign sho difficult to install, it jst wont work whatever I try
so why don't you guys tell me what I doing wrong.
```

```
Subject: Problems installing FreeBSD

I've just got the FreeBSD 2.1.5 CDRom from Walnut Creek, and I'm having a lot
of difficulty installing it. I have a 66 MHz 486 with 16 MB of
memory and an Adaptec 1540A SCSI board, a 1.2GB Quantum Fireball
disk and a Toshiba 3501XA CDRom drive. The installation works just
fine, but when I try to reboot the system, I get the message
Missing Operating System.
```

50.7 How to follow up to a question

Often you will want to send in additional information to a question you have already sent. The best way to do this is to reply to your original message. This has three advantages:

1. You include the original message text, so people will know what you are talking about. Do not forget to trim unnecessary text out, though.
2. The text in the subject line stays the same (you did remember to put one in, did you not?). Many mailers will sort messages by subject. This helps group messages together.
3. The message reference numbers in the header will refer to the previous message. Some mailers, such as [mutt](#), can *thread* messages, showing the exact relationships between the messages.

50.8 How to answer a question

Before you answer a question to FreeBSD-questions, consider:

1. A lot of the points on submitting questions also apply to answering questions. Read them.
2. Has somebody already answered the question? The easiest way to check this is to sort your incoming mail by subject: then (hopefully) you will see the question followed by any answers, all together.

If somebody has already answered it, it does not automatically mean that you should not send another answer. But it makes sense to read all the other answers first.

3. Do you have something to contribute beyond what has already been said? In general, “Yeah, me too” answers do not help much, although there are exceptions, like when somebody is describing a problem they are having, and they do not know whether it is their fault or whether there is something wrong with the hardware or software. If you do send a “me too” answer, you should also include any further relevant information.
4. Are you sure you understand the question? Very frequently, the person who asks the question is confused or does not express themselves very well. Even with the best understanding of the system, it is easy to send a reply which does not answer the question. This does not help: you will leave the person who submitted the question more frustrated or confused than ever. If nobody else answers, and you are not too sure either, you can always ask for more information.
5. Are you sure your answer is correct? If not, wait a day or so. If nobody else comes up with a better answer, you can still reply and say, for example, “I do not know if this is correct, but since nobody else has replied, why don’t you try replacing your ATAPI CDROM with a frog?”.
6. Unless there is a good reason to do otherwise, reply to the sender and to FreeBSD-questions. Many people on the FreeBSD-questions are “lurkers”: they learn by reading messages sent and replied to by others. If you take a message which is of general interest off the list, you are depriving these people of their information. Be careful with group replies; lots of people send messages with hundreds of CCs. If this is the case, be sure to trim the Cc: lines appropriately.
7. Include relevant text from the original message. Trim it to the minimum, but do not overdo it. It should still be possible for somebody who did not read the original message to understand what you are talking about.
8. Use some technique to identify which text came from the original message, and which text you add. I personally find that prepending “> `” to the original message works best. Leaving white space after the “> `” and leave empty lines between your text and the original text both make the result more readable.
9. Put your response in the correct place (after the text to which it replies). It is very difficult to read a thread of responses where each reply comes before the text to which it replies.
10. Most mailers change the subject line on a reply by prepending a text such as “Re:”. If your mailer does not do it automatically, you should do it manually.
11. If the submitter did not abide by format conventions (lines too long, inappropriate subject line), *please* fix it. In the case of an incorrect subject line (such as “HELP!?!?”), change the subject line to (say) “Re: Difficulties with sync PPP (was: HELP!?!?)”. That way other people trying to follow the thread will have less difficulty following it.

In such cases, it is appropriate to say what you did and why you did it, but try not to be rude. If you find you can not answer without being rude, do not answer.

If you just want to reply to a message because of its bad format, just reply to the submitter, not to the list. You can just send him this message in reply, if you like.

Build Your Own OS Update Server

Author Jason Helfman

51.1 Acknowledgments

This article was subsequently printed at [BSD Magazine](#).

51.2 Introduction

Experienced users or administrators are often responsible for several machines or environments. They understand the difficult demands and challenges of maintaining such an infrastructure. Running a FBUS.AP makes it easier to deploy security and software patches to selected test machines before rolling them out to production. It also means a number of systems can be updated from the local network rather than a potentially slower Internet connection. This article outlines the steps involved in creating an internal FBUS.AP.

51.3 Prerequisites

To build an internal FBUS.AP some requirements should be met.

- A running OS system.

Note

At a minimum, updates require building on a OS release greater than or equal to the target release version for distribution.

- A user account with at least 4 GB of available space. This will allow the creation of updates for 7.1 and 7.2, but the exact space requirements may change from version to version.
- An MAN.SSH.1 account on a remote machine to upload distributed updates.
- A web server, like Apache, with over half of the space required for the build. For instance, test builds for 7.1 and 7.2 consume a total amount of 4 GB, and the webserver space needed to distribute these updates is 2.6 GB.
- Basic knowledge of shell scripting with Bourne shell, MAN.SH.1.

51.4 Configuration: Installation & Setup

Download the `freebsd-update-server` software by installing `devel/subversion`, and execute:

```
PROMPT.USER svn co
    http://svn.freebsd.org/base/user/cperciva/freebsd-update-build
    freebsd-update-server
```

Update `scripts/build.conf` appropriately. It is sourced during all build operations.

Here is the default `build.conf`, which should be modified to suit your environment.

```
# Main configuration file for FreeBSD Update builds.  The
# release-specific configuration data is lower down in
# the scripts tree.

# Location from which to fetch releases
export FTP=ftp://ftp2.freebsd.org/pub/FreeBSD/releases

# Host platform
export HOSTPLATFORM=`uname -m`

# Host name to use inside jails
export BUILDHOSTNAME=${HOSTPLATFORM}-builder.daemonology.net

# Location of SSH key
export SSHKEY=/root/.ssh/id_dsa

# SSH account into which files are uploaded
MASTERACCT=builder@wadham.daemonology.net

# Directory into which files are uploaded
MASTERDIR=update-master.freebsd.org
```

Parameters for consideration would be:

- This is the location where ISO images are downloaded from (by the `fetchiso()` subroutine of `scripts/build.subr`). The location configured is not limited to FTP URIs. Any URI scheme supported by standard `MAN.FETCH.1` utility should work fine.

Customizations to the `fetchiso()` code can be installed by copying the default `build.subr` script to the release and architecture-specific area at `scripts/RELEASE/ARCHITECTURE/build.subr` and applying local changes.

- The name of the build host. This information will be displayed on updated systems when issuing:

```
PROMPT.USER uname -v
```

- The SSH key for uploading files to the update server. A key pair can be created by typing `ssh-keygen -t dsa`. This parameter is optional; standard password authentication will be used as a fallback authentication method when `SSHKEY` is not defined.

The `MAN.SSH-KEYGEN.1` manual page has more detailed information about SSH and the appropriate steps for creating and using one.

- Account for uploading files to the update server.
- Directory on the update server where files are uploaded to.

The default `build.conf` shipped with the `freebsd-update-server` sources is suitable for building `ARCH.I386` releases of OS. As an example of building an update server for other architectures, the following steps outline the configuration

changes needed for ARCH.AMD64:

Create a build environment for ARCH.AMD64:

```
PROMPT.USER mkdir -p /usr/local/freebsd-update-server/scripts/7.2-RELEASE/amd64
```

Install a build.conf in the newly created build directory. The build configuration options for OS 7.2-RELEASE on ARCH.AMD64 should be similar to:

```
# SHA256 hash of RELEASE disc1.iso image.
export RELH=1ealf6f652d7c5f5eab7ef9f8edbed50cb664b08ed761850f95f48e86cc71ef5

# Components of the world, source, and kernels
export WORLDPARTS="base catpages dict doc games info manpages proflibs lib32"
export SOURCEPARTS="base bin contrib crypto etc games gnu include krb5 \
                    lib libexec release rescue sbin secure share sys tools \
                    ubin usbin cddl"
export KERNELPARTS="generic"

# EOL date
export EOL=1275289200
```

- The MAN.SHA256.1 hash key for the desired release, is published within the respective release announcement.
- To generate the “End of Life” number for build.conf, refer to the “Estimated EOL” posted on the OS Security Website. The value of EOL can be derived from the date listed on the web site, using the MAN.DATE.1 utility, for example:

```
PROMPT.USER date -j -f '%Y%m%d-%H%M%S' '20090401-000000' '+%s'
```

51.5 Building Update Code

The first step is to run scripts/make.sh. This will build some binaries, create directories, and generate an RSA signing key used for approving builds. In this step, a passphrase will have to be supplied for the final creation of the signing key.

```
PROMPT.ROOT sh scripts/make.sh
cc -O2 -fno-strict-aliasing -pipe findstamps.c -o findstamps
findstamps.c: In function 'usage':
findstamps.c:45: warning: incompatible implicit declaration of built-in function 'exit'
cc -O2 -fno-strict-aliasing -pipe unstamp.c -o unstamp
install findstamps ../bin
install unstamp ../bin
rm -f findstamps unstamp
Generating RSA private key, 4096 bit long modulus
.....++
.....++
e is 65537 (0x10001)

Public key fingerprint:
27ef53e48dc869eea6c3136091cc6ab8589f967559824779e855d58a2294de9e

Encrypting signing key for root
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:

**Note**
```

Keep a note of the generated key fingerprint. This value is required in ``/etc/freebsd-update.conf`` for binary updates.

At this point, we are ready to stage a build.

```
PROMPT.ROOT cd /usr/local/freebsd-update-server
PROMPT.ROOT sh scripts/init.sh amd64 7.2-RELEASE
```

What follows is a sample of an *initial* build run.

```
PROMPT.ROOT sh scripts/init.sh amd64 7.2-RELEASE
Mon Aug 24 16:04:36 PDT 2009 Starting fetch for FreeBSD/amd64 7.2-RELEASE
/usr/local/freebsd-update-server/work/7.2-RELEASE100% of 588 MB 359 kBps 00m00s
Mon Aug 24 16:32:38 PDT 2009 Verifying discl hash for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 16:32:44 PDT 2009 Extracting components for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 16:34:05 PDT 2009 Constructing world+src image for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 16:35:57 PDT 2009 Extracting world+src for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 23:36:24 UTC 2009 Building world for FreeBSD/amd64 7.2-RELEASE
Tue Aug 25 00:31:29 UTC 2009 Distributing world for FreeBSD/amd64 7.2-RELEASE
Tue Aug 25 00:32:36 UTC 2009 Building and distributing kernels for FreeBSD/amd64 7.2-RELEASE
Tue Aug 25 00:44:44 UTC 2009 Constructing world components for FreeBSD/amd64 7.2-RELEASE
Tue Aug 25 00:44:56 UTC 2009 Distributing source for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 17:46:18 PDT 2009 Moving components into staging area for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 17:46:33 PDT 2009 Identifying extra documentation for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 17:47:13 PDT 2009 Extracting extra docs for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 17:47:18 PDT 2009 Indexing release for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 17:50:44 PDT 2009 Indexing world0 for FreeBSD/amd64 7.2-RELEASE

Files built but not released:
Files released but not built:
Files which differ by more than contents:
Files which differ between release and build:
kernel|generic|/GENERIC/hptrr.ko
kernel|generic|/GENERIC/kernel
src|sys|/sys/conf/newvers.sh
world|base|/boot/loader
world|base|/boot/pxeboot
world|base|/etc/mail/freebsd.cf
world|base|/etc/mail/freebsd.submit.cf
world|base|/etc/mail/sendmail.cf
world|base|/etc/mail/submit.cf
world|base|/lib/libcrypto.so.5
world|base|/usr/bin/ntpq
world|base|/usr/lib/libalias.a
world|base|/usr/lib/libalias_cuseeme.a
world|base|/usr/lib/libalias_dummy.a
world|base|/usr/lib/libalias_ftp.a
...
```

Then the build of the world is performed again, with world patches. A more detailed explanation may be found in `scripts/build.subr`.

Warning

During this second build cycle, the network time protocol daemon, `MAN.NTPD.8`, is turned off. Per A.CPERCIVA.EMAIL, Security Officer Emeritus of OS, “the `freebsd-update-server` build code needs to identify timestamps which are stored in files so that they can be ignored when comparing builds to determine which files need to be updated. This timestamp-finding works by doing two builds 400 days apart and comparing the results.”

```

Mon Aug 24 17:54:07 PDT 2009 Extracting world+src for FreeBSD/amd64 7.2-RELEASE
Wed Sep 29 00:54:34 UTC 2010 Building world for FreeBSD/amd64 7.2-RELEASE
Wed Sep 29 01:49:42 UTC 2010 Distributing world for FreeBSD/amd64 7.2-RELEASE
Wed Sep 29 01:50:50 UTC 2010 Building and distributing kernels for FreeBSD/amd64 7.2-RELEASE
Wed Sep 29 02:02:56 UTC 2010 Constructing world components for FreeBSD/amd64 7.2-RELEASE
Wed Sep 29 02:03:08 UTC 2010 Distributing source for FreeBSD/amd64 7.2-RELEASE
Tue Sep 28 19:04:31 PDT 2010 Moving components into staging area for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 19:04:46 PDT 2009 Extracting extra docs for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 19:04:51 PDT 2009 Indexing world1 for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 19:08:04 PDT 2009 Locating build stamps for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 19:10:19 PDT 2009 Cleaning staging area for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 19:10:19 PDT 2009 Preparing to copy files into staging area for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 19:10:20 PDT 2009 Copying data files into staging area for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 12:16:57 PDT 2009 Copying metadata files into staging area for FreeBSD/amd64 7.2-RELEASE
Mon Aug 24 12:16:59 PDT 2009 Constructing metadata index and tag for FreeBSD/amd64 7.2-RELEASE

Files found which include build stamps:
kernel|generic|/GENERIC/hptrr.ko
kernel|generic|/GENERIC/kernel
world|base|/boot/loader
world|base|/boot/pxeboot
world|base|/etc/mail/freebsd.cf
world|base|/etc/mail/freebsd.submit.cf
world|base|/etc/mail/sendmail.cf
world|base|/etc/mail/submit.cf
world|base|/lib/libcrypto.so.5
world|base|/usr/bin/ntpq
world|base|/usr/include/osreldate.h
world|base|/usr/lib/libalias.a
world|base|/usr/lib/libalias_cuseeme.a
world|base|/usr/lib/libalias_dummy.a
world|base|/usr/lib/libalias_ftp.a
...

```

Finally, the build completes.

```

Values of build stamps, excluding library archive headers:
v1.2 (Aug 25 2009 00:40:36)
v1.2 (Aug 25 2009 00:38:22)
@(#)FreeBSD 7.2-RELEASE #0: Tue Aug 25 00:38:29 UTC 2009
FreeBSD 7.2-RELEASE #0: Tue Aug 25 00:38:29 UTC 2009
  root@server.myhost.com:/usr/obj/usr/src/sys/GENERIC
7.2-RELEASE
Mon Aug 24 23:55:25 UTC 2009
Mon Aug 24 23:55:25 UTC 2009
##### built by root@server.myhost.com on Tue Aug 25 00:16:15 UTC 2009
##### built by root@server.myhost.com on Tue Aug 25 00:16:15 UTC 2009
##### built by root@server.myhost.com on Tue Aug 25 00:16:15 UTC 2009
##### built by root@server.myhost.com on Tue Aug 25 00:16:15 UTC 2009
Mon Aug 24 23:46:47 UTC 2009
ntpq 4.2.4p5-a Mon Aug 24 23:55:53 UTC 2009 (1)
  * Copyright (c) 1992-2009 The FreeBSD Project.
Mon Aug 24 23:46:47 UTC 2009
Mon Aug 24 23:55:40 UTC 2009
Aug 25 2009
ntpd 4.2.4p5-a Mon Aug 24 23:55:52 UTC 2009 (1)
ntpdate 4.2.4p5-a Mon Aug 24 23:55:53 UTC 2009 (1)
ntpd 4.2.4p5-a Mon Aug 24 23:55:53 UTC 2009 (1)
Tue Aug 25 00:21:21 UTC 2009

```

```
Tue Aug 25 00:21:21 UTC 2009
Tue Aug 25 00:21:21 UTC 2009
Mon Aug 24 23:46:47 UTC 2009
```

```
FreeBSD/amd64 7.2-RELEASE initialization build complete. Please
review the list of build stamps printed above to confirm that
they look sensible, then run
# sh -e approve.sh amd64 7.2-RELEASE
to sign the release.
```

Approve the build if everything is correct. More information on determining this can be found in the distributed source file named `USAGE`. Execute `scripts/approve.sh`, as directed. This will sign the release, and move components into a staging area suitable for uploading.

```
PROMPT.ROOT cd /usr/local/freebsd-update-server
PROMPT.ROOT sh scripts/mountkey.sh
```

```
PROMPT.ROOT sh -e scripts/approve.sh amd64 7.2-RELEASE
Wed Aug 26 12:50:06 PDT 2009 Signing build for FreeBSD/amd64 7.2-RELEASE
Wed Aug 26 12:50:06 PDT 2009 Copying files to patch source directories for FreeBSD/amd64 7.2-RELEASE
Wed Aug 26 12:50:06 PDT 2009 Copying files to upload staging area for FreeBSD/amd64 7.2-RELEASE
Wed Aug 26 12:50:07 PDT 2009 Updating databases for FreeBSD/amd64 7.2-RELEASE
Wed Aug 26 12:50:07 PDT 2009 Cleaning staging area for FreeBSD/amd64 7.2-RELEASE
```

After the approval process is complete, the upload procedure may be started.

```
PROMPT.ROOT cd /usr/local/freebsd-update-server
PROMPT.ROOT sh scripts/upload.sh amd64 7.2-RELEASE

**Note**

In the event update code needs to be re-uploaded, this may be done
by changing to the public distributions directory for the target
release and updating attributes of the *uploaded* file.

::

PROMPT.ROOT cd /usr/local/freebsd-update-server/pub/7.2-RELEASE/amd64
PROMPT.ROOT touch -t 200801010101.01 uploaded
```

The uploaded files will need to be in the document root of the webserver in order for updates to be distributed. The exact configuration will vary depending on the web server used. For the Apache web server, please refer to the Configuration of Apache servers section in the Handbook.

Update client's `KeyPrint` and `ServerName` in `/etc/freebsd-update.conf`, and perform updates as instructed in the OS Update section of the Handbook.

Important

In order for `FBUS.AP` to work properly, updates for both the *current* release and the release *one wants to upgrade to* need to be built. This is necessary for determining the differences of files between releases. For example, when upgrading a OS system from 7.1-RELEASE to 7.2-RELEASE, updates will need to be built and uploaded to your distribution server for both versions.

For reference, the entire run of ``init.sh <init.txt>`__` is attached.

51.6 Building a Patch

Every time a security advisory or security notice is announced, a patch update can be built.

For this example, 7.1-RELEASE will be used.

A couple of assumptions are made for a different release build:

- Setup the correct directory structure for the initial build.
- Perform an initial build for 7.1-RELEASE.

Create the patch directory of the respective release under `/usr/local/freebsd-update-server/patches/`.

```
PROMPT.USER mkdir -p /usr/local/freebsd-update-server/patches/7.1-RELEASE/
PROMPT.USER cd /usr/local/freebsd-update-server/patches/7.1-RELEASE
```

As an example, take the patch for MAN.NAMED.8. Read the advisory, and grab the necessary file from OS Security Advisories. More information on interpreting the advisory, can be found in the OS Handbook.

In the [security brief](#), this advisory is called SA-09:12.bind. After downloading the file, it is required to rename the file to an appropriate patch level. It is suggested to keep this consistent with official OS patch levels, but its name may be freely chosen. For this build, let us follow the currently established practice of OS and call this p7. Rename the file:

```
PROMPT.USER cd /usr/local/freebsd-update-server/patches/7.1-RELEASE/; mv bind.patch 7-SA-09:12.bind

**Note**

When running a patch level build, it is assumed that previous
patches are in place. When a patch build is run, it will run all
patches contained in the patch directory.

There can be custom patches added to any build. Use the number zero,
or any other number.

**Warning**

It is up to the administrator of the FBUS.AP to take appropriate
measures to verify the authenticity of every patch.
```

At this point, a *diff* is ready to be built. The software checks first to see if a `scripts/init.sh` has been run on the respective release prior to running the diff build.

```
PROMPT.ROOT cd /usr/local/freebsd-update-server
PROMPT.ROOT sh scripts/diff.sh amd64 7.1-RELEASE 7
```

What follows is a sample of a *differential* build run.

```
PROMPT.ROOT sh -e scripts/diff.sh amd64 7.1-RELEASE 7
Wed Aug 26 10:09:59 PDT 2009 Extracting world+src for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 17:10:25 UTC 2009 Building world for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 18:05:11 UTC 2009 Distributing world for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 18:06:16 UTC 2009 Building and distributing kernels for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 18:17:50 UTC 2009 Constructing world components for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 18:18:02 UTC 2009 Distributing source for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 11:19:23 PDT 2009 Moving components into staging area for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 11:19:37 PDT 2009 Extracting extra docs for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 11:19:42 PDT 2009 Indexing world0 for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 11:23:02 PDT 2009 Extracting world+src for FreeBSD/amd64 7.1-RELEASE-p7
Thu Sep 30 18:23:29 UTC 2010 Building world for FreeBSD/amd64 7.1-RELEASE-p7
```

```

Thu Sep 30 19:18:15 UTC 2010 Distributing world for FreeBSD/amd64 7.1-RELEASE-p7
Thu Sep 30 19:19:18 UTC 2010 Building and distributing kernels for FreeBSD/amd64 7.1-RELEASE-p7
Thu Sep 30 19:30:52 UTC 2010 Constructing world components for FreeBSD/amd64 7.1-RELEASE-p7
Thu Sep 30 19:31:03 UTC 2010 Distributing source for FreeBSD/amd64 7.1-RELEASE-p7
Thu Sep 30 12:32:25 PDT 2010 Moving components into staging area for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 12:32:39 PDT 2009 Extracting extra docs for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 12:32:43 PDT 2009 Indexing world1 for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 12:35:54 PDT 2009 Locating build stamps for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 12:36:58 PDT 2009 Reverting changes due to build stamps for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 12:37:14 PDT 2009 Cleaning staging area for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 12:37:14 PDT 2009 Preparing to copy files into staging area for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 12:37:15 PDT 2009 Copying data files into staging area for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 12:43:23 PDT 2009 Copying metadata files into staging area for FreeBSD/amd64 7.1-RELEASE-p7
Wed Aug 26 12:43:25 PDT 2009 Constructing metadata index and tag for FreeBSD/amd64 7.1-RELEASE-p7
...
Files found which include build stamps:
kernel|generic|/GENERIC/hptrr.ko
kernel|generic|/GENERIC/kernel
world|base|/boot/loader
world|base|/boot/pxeboot
world|base|/etc/mail/freebsd.cf
world|base|/etc/mail/freebsd.submit.cf
world|base|/etc/mail/sendmail.cf
world|base|/etc/mail/submit.cf
world|base|/lib/libcrypto.so.5
world|base|/usr/bin/ntpq
world|base|/usr/include/osreldate.h
world|base|/usr/lib/libalias.a
world|base|/usr/lib/libalias_cuseeme.a
world|base|/usr/lib/libalias_dummy.a
world|base|/usr/lib/libalias_ftp.a
...
Values of build stamps, excluding library archive headers:
v1.2 (Aug 26 2009 18:13:46)
v1.2 (Aug 26 2009 18:11:44)
@(#)FreeBSD 7.1-RELEASE-p7 #0: Wed Aug 26 18:11:50 UTC 2009
FreeBSD 7.1-RELEASE-p7 #0: Wed Aug 26 18:11:50 UTC 2009
  root@server.myhost.com:/usr/obj/usr/src/sys/GENERIC
7.1-RELEASE-p7
Wed Aug 26 17:29:15 UTC 2009
Wed Aug 26 17:29:15 UTC 2009
##### built by root@server.myhost.com on Wed Aug 26 17:49:58 UTC 2009
##### built by root@server.myhost.com on Wed Aug 26 17:49:58 UTC 2009
##### built by root@server.myhost.com on Wed Aug 26 17:49:58 UTC 2009
##### built by root@server.myhost.com on Wed Aug 26 17:49:58 UTC 2009
Wed Aug 26 17:20:39 UTC 2009
ntpq 4.2.4p5-a Wed Aug 26 17:29:42 UTC 2009 (1)
  * Copyright (c) 1992-2009 The FreeBSD Project.
Wed Aug 26 17:20:39 UTC 2009
Wed Aug 26 17:29:30 UTC 2009
Aug 26 2009
ntpd 4.2.4p5-a Wed Aug 26 17:29:41 UTC 2009 (1)
ntpddate 4.2.4p5-a Wed Aug 26 17:29:42 UTC 2009 (1)
ntpdcc 4.2.4p5-a Wed Aug 26 17:29:42 UTC 2009 (1)
Wed Aug 26 17:55:02 UTC 2009
Wed Aug 26 17:55:02 UTC 2009
Wed Aug 26 17:55:02 UTC 2009
Wed Aug 26 17:20:39 UTC 2009

```



```
...
```

Updates are printed, and approval is requested.

```
New updates:
kernel|generic|/GENERIC/kernel.symbols|f|0|0|0555|0|7c8dc176763f96ced0a57fc04e7c1b8d793f27e006dd13e0b
kernel|generic|/GENERIC/kernel|f|0|0|0555|0|33197e8cf15bbbac263d17f39c153c9d489348c2c534f7ca1120a118
kernel|generic|/|d|0|0|0755|0||
src|base|/|d|0|0|0755|0||
src|bin|/|d|0|0|0755|0||
src|cddl|/|d|0|0|0755|0||
src|contrib|/contrib/bind9/bin/named/update.c|f|0|10000|0644|0|4d434abf0983df9bc47435670d307fa882ef4
src|contrib|/contrib/bind9/lib/dns/openssldsa_link.c|f|0|10000|0644|0|c6805c39f3da2a06dd3f163f26c314
src|contrib|/contrib/bind9/lib/dns/opensslrsa_link.c|f|0|10000|0644|0|fa0f7417ee9da42cc8d0fd96ad24e7
...
FreeBSD/amd64 7.1-RELEASE update build complete. Please review
the list of build stamps printed above and the list of updated
files to confirm that they look sensible, then run
# sh -e approve.sh amd64 7.1-RELEASE
to sign the build.
```

Follow the same process as noted before for approving a build:

```
PROMPT.ROOT sh -e scripts/approve.sh amd64 7.1-RELEASE
Wed Aug 26 12:50:06 PDT 2009 Signing build for FreeBSD/amd64 7.1-RELEASE
Wed Aug 26 12:50:06 PDT 2009 Copying files to patch source directories for FreeBSD/amd64 7.1-RELEASE
Wed Aug 26 12:50:06 PDT 2009 Copying files to upload staging area for FreeBSD/amd64 7.1-RELEASE
Wed Aug 26 12:50:07 PDT 2009 Updating databases for FreeBSD/amd64 7.1-RELEASE
Wed Aug 26 12:50:07 PDT 2009 Cleaning staging area for FreeBSD/amd64 7.1-RELEASE

The FreeBSD/amd64 7.1-RELEASE update build has been signed and is
ready to be uploaded. Remember to run
# sh -e umountkey.sh
to unmount the decrypted key once you have finished signing all
the new builds.
```

After approving the build, upload the software:

```
PROMPT.ROOT cd /usr/local/freebsd-update-server
PROMPT.ROOT sh scripts/upload.sh amd64 7.1-RELEASE
```

For reference, the entire run of ``diff.sh <diff.txt>`__` is attached.

51.7 Tips

- If a custom release is built using the native `make release` procedure, `freebsd-update-server` code will work from your release. As an example, a release without ports or documentation can be built by clearing functionality pertaining to documentation subroutines `findextradocs()`, `addextradocs()` and altering the download location in `fetchiso()`, respectively, in `scripts/build.subr`. As a last step, change the `MAN.SHA256.1` hash in `build.conf` under your respective release and architecture and you are ready to build off your custom release.

```
# Compare ${WORKDIR}/release and ${WORKDIR}/${1}, identify which parts
# of the world|doc subcomponent are missing from the latter, and
# build a tarball out of them.
findextradocs () {
}
```

```
# Add extra docs to ${WORKDIR}/${1}
addextradocs () {
}
```

- **Adding “-j NUMBER” flags to buildworld and obj targets in the**

scripts/build.subr script may speed up processing depending on the hardware used, however it is not necessary. Using these flags in other targets is not recommended, as it may cause the build to become unreliable.

```
# Build the world
log "Building world"
cd /usr/src &&
make -j 2 ${COMPATFLAGS} buildworld 2>&1

# Distribute the world
log "Distributing world"
cd /usr/src/release &&
make -j 2 obj &&
make ${COMPATFLAGS} release.1 release.2 2>&1
```

- **Create an appropriate DNS SRV record for the update server, and put others behind it with variable weights.** Using this facility will provide update mirrors, however this tip is not necessary unless you wish to provide a redundant service.

```
_http._tcp.update.myserver.com.      IN SRV    0 2 80  host1.myserver.com.
                                SRV    0 1 80  host2.myserver.com.
                                SRV    0 0 80  host3.myserver.com.
```

Writing a GEOM Class

Author Ivan Voras

52.1 Introduction

52.1.1 Documentation

Documentation on kernel programming is scarce — it is one of few areas where there is nearly nothing in the way of friendly tutorials, and the phrase “use the source!” really holds true. However, there are some bits and pieces (some of them seriously outdated) floating around that should be studied before beginning to code:

- The FreeBSD Developer’s Handbook — part of the documentation project, it does not contain anything specific to kernel programming, but rather some general useful information.
- The FreeBSD Architecture Handbook — also from the documentation project, contains descriptions of several low-level facilities and procedures. The most important chapter is 13, Writing FreeBSD device drivers.
- The Blueprints section of [FreeBSD Diary](#) web site — contains several interesting articles on kernel facilities.
- The man pages in section 9 — for important documentation on kernel functions.
- The MAN.GEOM.4 man page and [PHK’s GEOM slides](#) — for general introduction of the GEOM subsystem.
- Man pages MAN.G.BIO.9, MAN.G.EVENT.9, MAN.G.DATA.9, MAN.G.GEOM.9, MAN.G.PROVIDER.9, MAN.G.CONSUMER.9, MAN.G.ACCESS.9 & others linked from those, for documentation on specific functionalities.
- The MAN.STYLE.9 man page — for documentation on the coding-style conventions which must be followed for any code which is to be committed to the FreeBSD Subversion tree.

52.2 Preliminaries

The best way to do kernel development is to have (at least) two separate computers. One of these would contain the development environment and sources, and the other would be used to test the newly written code by network-booting and network-mounting filesystems from the first one. This way if the new code contains bugs and crashes the machine, it will not mess up the sources (and other “live” data). The second system does not even require a proper display. Instead, it could be connected with a serial cable or KVM to the first one.

But, since not everybody has two or more computers handy, there are a few things that can be done to prepare an otherwise “live” system for developing kernel code. This setup is also applicable for developing in a [VMWare](#) or [QEmu](#) virtual machine (the next best thing after a dedicated development machine).

52.2.1 Modifying a System for Development

For any kernel programming a kernel with `INVARIANTS` enabled is a must-have. So enter these in your kernel configuration file:

```
options INVARIANT_SUPPORT
options INVARIANTS
```

For more debugging you should also include `WITNESS` support, which will alert you of mistakes in locking:

```
options WITNESS_SUPPORT
options WITNESS
```

For debugging crash dumps, a kernel with debug symbols is needed:

```
makeoptions    DEBUG=-g
```

With the usual way of installing the kernel (“`make installkernel`”) the debug kernel will not be automatically installed. It is called `kernel.debug` and located in `/usr/obj/usr/src/sys/KERNELNAME/`. For convenience it should be copied to `/boot/kernel/`.

Another convenience is enabling the kernel debugger so you can examine a kernel panic when it happens. For this, enter the following lines in your kernel configuration file:

```
options KDB
options DDB
options KDB_TRACE
```

For this to work you might need to set a `sysctl` (if it is not on by default):

```
debug.debugger_on_panic=1
```

Kernel panics will happen, so care should be taken with the filesystem cache. In particular, having `softupdates` might mean the latest file version could be lost if a panic occurs before it is committed to storage. Disabling `softupdates` yields a great performance hit, and still does not guarantee data consistency. Mounting filesystem with the “`sync`” option is needed for that. For a compromise, the `softupdates` cache delays can be shortened. There are three `sysctl`’s that are useful for this (best to be set in `/etc/sysctl.conf`):

```
kern.filedelay=5
kern.dirdelay=4
kern.metadelays=3
```

The numbers represent seconds.

For debugging kernel panics, kernel core dumps are required. Since a kernel panic might make filesystems unusable, this crash dump is first written to a raw partition. Usually, this is the swap partition. This partition must be at least as large as the physical RAM in the machine. On the next boot, the dump is copied to a regular file. This happens after filesystems are checked and mounted, and before swap is enabled. This is controlled with two `/etc/rc.conf` variables:

```
dumpdev="/dev/ad0s4b"
dumpdir="/usr/core"
```

The `dumpdev` variable specifies the swap partition and `dumpdir` tells the system where in the filesystem to relocate the core dump on reboot.

Writing kernel core dumps is slow and takes a long time so if you have lots of memory (>256M) and lots of panics it could be frustrating to sit and wait while it is done (twice — first to write it to swap, then to relocate it to filesystem). It is convenient then to limit the amount of RAM the system will use via a `/boot/loader.conf` tunable:

```
hw.physmem="256M"
```

If the panics are frequent and filesystems large (or you simply do not trust `softupdates+background fsck`) it is advisable to turn background fsck off via `/etc/rc.conf` variable:

```
background_fsck="NO"
```

This way, the filesystems will always get checked when needed. Note that with background fsck, a new panic could happen while it is checking the disks. Again, the safest way is not to have many local filesystems by using another computer as an NFS server.

52.2.2 Starting the Project

For the purpose of creating a new GEOM class, an empty subdirectory has to be created under an arbitrary user-accessible directory. You do not have to create the module directory under `/usr/src`.

52.2.3 The Makefile

It is good practice to create `Makefiles` for every nontrivial coding project, which of course includes kernel modules.

Creating the `Makefile` is simple thanks to an extensive set of helper routines provided by the system. In short, here is how a minimal `Makefile` looks for a kernel module:

```
SRCS=g_journal.c
KMOD=geom_journal

.include <bsd.kmod.mk>
```

This `Makefile` (with changed filenames) will do for any kernel module, and a GEOM class can reside in just one kernel module. If more than one file is required, list it in the `SRCS` variable, separated with whitespace from other filenames.

52.3 On FreeBSD Kernel Programming

52.3.1 Memory Allocation

See `MAN.MALLOC.9`. Basic memory allocation is only slightly different than its userland equivalent. Most notably, `malloc()` and `free()` accept additional parameters as is described in the man page.

A “malloc type” must be declared in the declaration section of a source file, like this:

```
static MALLOC_DEFINE(M_GJOURNAL, "gjournal data", "GEOM_JOURNAL Data");
```

To use this macro, `sys/param.h`, `sys/kernel.h` and `sys/malloc.h` headers must be included.

There is another mechanism for allocating memory, the UMA (Universal Memory Allocator). See `MAN.UMA.9` for details, but it is a special type of allocator mainly used for speedy allocation of lists comprised of same-sized items (for example, dynamic arrays of structs).

52.3.2 Lists and Queues

See `MAN.QUEUE.3`. There are a LOT of cases when a list of things needs to be maintained. Fortunately, this data structure is implemented (in several ways) by C macros included in the system. The most used list type is `TAILQ`

because it is the most flexible. It is also the one with largest memory requirements (its elements are doubly-linked) and also the slowest (although the speed variation is on the order of several CPU instructions more, so it should not be taken seriously).

If data retrieval speed is very important, see `MAN.TREE.3` and `MAN.HASHINIT.9`.

52.3.3 BIOs

Structure `bio` is used for any and all Input/Output operations concerning GEOM. It basically contains information about what device (‘provider’) should satisfy the request, request type, offset, length, pointer to a buffer, and a bunch of “user-specific” flags and fields that can help implement various hacks.

The important thing here is that `bios` are handled asynchronously. That means that, in most parts of the code, there is no analogue to userland’s `MAN.READ.2` and `MAN.WRITE.2` calls that do not return until a request is done. Rather, a developer-supplied function is called as a notification when the request gets completed (or results in error).

The asynchronous programming model (also called “event-driven”) is somewhat harder than the much more used imperative one used in userland (at least it takes a while to get used to it). In some cases the helper routines `g_write_data()` and `g_read_data()` can be used, but *not always*. In particular, they cannot be used when a mutex is held; for example, the GEOM topology mutex or the internal mutex held during the `.start()` and `.stop()` functions.

52.4 On GEOM Programming

52.4.1 Ggate

If maximum performance is not needed, a much simpler way of making a data transformation is to implement it in userland via the `ggate` (GEOM gate) facility. Unfortunately, there is no easy way to convert between, or even share code between the two approaches.

52.4.2 GEOM Class

GEOM classes are transformations on the data. These transformations can be combined in a tree-like fashion. Instances of GEOM classes are called *geoms*.

Each GEOM class has several “class methods” that get called when there is no geom instance available (or they are simply not bound to a single instance):

- `.init` is called when GEOM becomes aware of a GEOM class (when the kernel module gets loaded.)
- `.fini` gets called when GEOM abandons the class (when the module gets unloaded)
- `.taste` is called next, once for each provider the system has available. If applicable, this function will usually create and start a geom instance.
- `.destroy_geom` is called when the geom should be disbanded
- `.ctlconf` is called when user requests reconfiguration of existing geom

Also defined are the GEOM event functions, which will get copied to the geom instance.

Field `.geoms` in the `g_class` structure is a LIST of geoms instantiated from the class.

These functions are called from the `g_event` kernel thread.

52.4.3 Softc

The name “softc” is a legacy term for “driver private data”. The name most probably comes from the archaic term “software control block”. In GEOM, it is a structure (more precise: pointer to a structure) that can be attached to a geom instance to hold whatever data is private to the geom instance. Most GEOM classes have the following members:

- `struct g_provider *provider`: The “provider” this geom instantiates
- `uint16_t n_disks`: Number of consumer this geom consumes
- `struct g_consumer **disks`: Array of `struct g_consumer*`. (It is not possible to use just single indirection because `struct g_consumer*` are created on our behalf by GEOM).

The `softc` structure contains all the state of geom instance. Every geom instance has its own `softc`.

52.4.4 Metadata

Format of metadata is more-or-less class-dependent, but MUST start with:

- 16 byte buffer for null-terminated signature (usually the class name)
- `uint32` version ID

It is assumed that geom classes know how to handle metadata with version ID’s lower than theirs.

Metadata is located in the last sector of the provider (and thus must fit in it).

(All this is implementation-dependent but all existing code works like that, and it is supported by libraries.)

52.4.5 Labeling/creating a GEOM

The sequence of events is:

- user calls `MAN.GEOM.8` utility (or one of its hardlinked friends)
- the utility figures out which geom class it is supposed to handle and searches for `geom_CLASSNAME.so` library (usually in `/lib/geom`).
- it `MAN.DLOPEN.3-s` the library, extracts the definitions of command-line parameters and helper functions.

In the case of creating/labeling a new geom, this is what happens:

- `MAN.GEOM.8` looks in the command-line argument for the command (usually `label`), and calls a helper function.
- The helper function checks parameters and gathers metadata, which it proceeds to write to all concerned providers.
- This “spoils” existing geoms (if any) and initializes a new round of “tasting” of the providers. The intended geom class recognizes the metadata and brings the geom up.

(The above sequence of events is implementation-dependent but all existing code works like that, and it is supported by libraries.)

52.4.6 GEOM Command Structure

The helper `geom_CLASSNAME.so` library exports `class_commands` structure, which is an array of `struct g_command` elements. Commands are of uniform format and look like:

`verb [-options] geomname [other]`

Common verbs are:

- `label` — to write metadata to devices so they can be recognized at tasting and brought up in geoms
- `destroy` — to destroy metadata, so the geoms get destroyed

Common options are:

- `-v` : be verbose
- `-f` : force

Many actions, such as labeling and destroying metadata can be performed in userland. For this, `struct g_command` provides field `gc_func` that can be set to a function (in the same `.so`) that will be called to process a verb. If `gc_func` is `NULL`, the command will be passed to kernel module, to `.ctlreq` function of the geom class.

52.4.7 Geoms

Geoms are instances of GEOM classes. They have internal data (a softc structure) and some functions with which they respond to external events.

The event functions are:

- `.access` : calculates permissions (read/write/exclusive)
- `.dumpconf` : returns XML-formatted information about the geom
- `.orphan` : called when some underlying provider gets disconnected
- `.spoiled` : called when some underlying provider gets written to
- `.start` : handles I/O

These functions are called from the `g_down` kernel thread and there can be no sleeping in this context, (see definition of sleeping elsewhere) which limits what can be done quite a bit, but forces the handling to be fast.

Of these, the most important function for doing actual useful work is the `.start()` function, which is called when a BIO request arrives for a provider managed by a instance of geom class.

52.4.8 GEOM Threads

There are three kernel threads created and run by the GEOM framework:

- `g_down` : Handles requests coming from high-level entities (such as a userland request) on the way to physical devices
- `g_up` : Handles responses from device drivers to requests made by higher-level entities
- `g_event` : Handles all other cases: creation of geom instances, access counting, “spoil” events, etc.

When a user process issues “read data X at offset Y of a file” request, this is what happens:

- The filesystem converts the request into a struct bio instance and passes it to the GEOM subsystem. It knows what geom instance should handle it because filesystems are hosted directly on a geom instance.
- The request ends up as a call to the `.start()` function made on the `g_down` thread and reaches the top-level geom instance.

- This top-level geom instance (for example the partition slicer) determines that the request should be routed to a lower-level instance (for example the disk driver). It makes a copy of the bio request (bio requests *ALWAYS* need to be copied between instances, with `g_clone_bio()`!), modifies the data offset and target provider fields and executes the copy with `g_io_request()`
- The disk driver gets the bio request also as a call to `.start()` on the `g_down` thread. It talks to hardware, gets the data back, and calls `g_io_deliver()` on the bio.
- Now, the notification of bio completion “bubbles up” in the `g_up` thread. First the partition slicer gets `.done()` called in the `g_up` thread, it uses information stored in the bio to free the cloned bio structure (with `g_destroy_bio()`) and calls `g_io_deliver()` on the original request.
- The filesystem gets the data and transfers it to userland.

See MAN.G.BIO.9 man page for information how the data is passed back and forth in the `bio` structure (note in particular the `bio_parent` and `bio_children` fields and how they are handled).

One important feature is: *THERE CAN BE NO SLEEPING IN G_UP AND G_DOWN THREADS*. This means that none of the following things can be done in those threads (the list is of course not complete, but only informative):

- Calls to `msleep()` and `tsleep()`, obviously.
- Calls to `g_write_data()` and `g_read_data()`, because these sleep between passing the data to consumers and returning.
- Waiting for I/O.
- Calls to MAN.MALLOC.9 and `uma_zalloc()` with `M_WAITOK` flag set
- `sx` and other sleepable locks

This restriction is here to stop GEOM code clogging the I/O request path, since sleeping is usually not time-bound and there can be no guarantees on how long will it take (there are some other, more technical reasons also). It also means that there is not much that can be done in those threads; for example, almost any complex thing requires memory allocation. Fortunately, there is a way out: creating additional kernel threads.

52.4.9 Kernel Threads for Use in GEOM Code

Kernel threads are created with MAN.KTHREAD.CREATE.9 function, and they are sort of similar to userland threads in behaviour, only they cannot return to caller to signify termination, but must call MAN.KTHREAD.EXIT.9.

In GEOM code, the usual use of threads is to offload processing of requests from `g_down` thread (the `.start()` function). These threads look like “event handlers”: they have a linked list of event associated with them (on which events can be posted by various functions in various threads so it must be protected by a mutex), take the events from the list one by one and process them in a big `switch()` statement.

The main benefit of using a thread to handle I/O requests is that it can sleep when needed. Now, this sounds good, but should be carefully thought out. Sleeping is well and very convenient but can very effectively destroy performance of the geom transformation. Extremely performance-sensitive classes probably should do all the work in `.start()` function call, taking great care to handle out-of-memory and similar errors.

The other benefit of having a event-handler thread like that is to serialize all the requests and responses coming from different geom threads into one thread. This is also very convenient but can be slow. In most cases, handling of `.done()` requests can be left to the `g_up` thread.

Mutexes in FreeBSD kernel (see MAN.MUTEX.9) have one distinction from their more common userland cousins — the code cannot sleep while holding a mutex). If the code needs to sleep a lot, MAN.SX.9 locks may be more appropriate. On the other hand, if you do almost everything in a single thread, you may get away with no mutexes at all.

Implementing UFS Journaling on a Desktop PC

Author ManolisKiagias

53.1 Introduction

While professional servers are usually well protected from unforeseen shutdowns, the typical desktop is at the mercy of power failures, accidental resets, and other user related incidents that can lead to unclean shutdowns. Soft Updates usually protect the file system efficiently in such cases, although most of the times a lengthy background check is required. On rare occasions, file system corruption reaches a point where user intervention is required and data may be lost.

The new journaling capability provided by GEOM can greatly assist in such scenarios, by virtually eliminating the time required for file system checking, and ensuring that the file system is quickly restored to a consistent state.

This article describes a procedure for implementing UFS journaling on a typical desktop PC scenario (one hard disk used for both operating system and data). It should be followed during a fresh installation of OS. The steps are simple enough and do not require overly complex interaction with the command line.

After reading this article, you will know:

- How to reserve space for journaling during a new installation of OS.
- How to load and enable the `geom_journal` module (or build support for it in your custom kernel).
- How to convert your existing file systems to utilize journaling, and what options to use in `/etc/fstab` to mount them.
- How to implement journaling in new (empty) partitions.
- How to troubleshoot common problems associated with journaling.

Before reading this article, you should be able to:

- Understand basic UNIX and OS concepts.
- Be familiar with the installation procedure of OS and the `sysinstall` utility.

Warning

The procedure described here is intended for preparing a new installation where no actual user data is stored on the disk yet. While it is possible to modify and extend this procedure for systems already in production, you should *backup* all important data before doing so. Messing around with disks and partitions at a low level can lead to fatal mistakes and data loss.

53.2 Understanding Journaling in OS

The journaling provided by GEOM in OS 7.X is not file system specific (unlike for example the ext3 file system in LINUX) but is functioning at the block level. Though this means it can be applied to different file systems, for OS 7.0-RELEASE, it can only be used on UFS2.

This functionality is provided by loading the `geom_journal.ko` module into the kernel (or building it into a custom kernel) and using the `gjournal` command to configure the file systems. In general, you would like to journal large file systems, like `/usr`. You will need however (see the following section) to reserve some free disk space.

When a file system is journaled, some disk space is needed to keep the journal itself. The disk space that holds the actual data is referred to as the *data provider*, while the one that holds the journal is referred to as the *journal provider*. The data and journal providers need to be on different partitions when journaling an existing (non-empty) partition. When journaling a new partition, you have the option to use a single provider for both data and journal. In any case, the `gjournal` command combines both providers to create the final journaled file system. For example:

- You wish to journal your `/usr` file system, stored in `/dev/ad0s1f` (which already contains data).
- You reserved some free disk space in a partition in `/dev/ad0s1g`.
- Using `gjournal`, a new `/dev/ad0s1f.journal` device is created where `/dev/ad0s1f` is the data provider, and `/dev/ad0s1g` is the journal provider. This new device is then used for all subsequent file operations.

The amount of disk space you need to reserve for the journal provider depends on the usage load of the file system and not on the size of the data provider. For example on a typical office desktop, a 1 GB journal provider for the `/usr` file system will suffice, while a machine that deals with heavy disk I/O (i.e. video editing) may need more. A kernel panic will occur if the journal space is exhausted before it has a chance to be committed.

Note

The journal sizes suggested here, are highly unlikely to cause problems in typical desktop use (such as web browsing, word processing and playback of media files). If your workload includes intense disk activity, use the following rule for maximum reliability: Your RAM size should fit in 30% of the journal provider's space. For example, if your system has 1 GB RAM, create an approximately 3.3 GB journal provider. (Multiply your RAM size with 3.3 to obtain the size of the journal).

For more information about journaling, please read the manual page of `MAN.GJOURNAL.8`.

53.3 Steps During the Installation of OS

53.3.1 Reserving Space for Journaling

A typical desktop machine usually has one hard disk that stores both the OS and user data. Arguably, the default partitioning scheme selected by `sysinstall` is more or less suitable: A desktop machine does not need a large `/var` partition, while `/usr` is allocated the bulk of the disk space, since user data and a lot of packages are installed into its subdirectories.

The default partitioning (the one obtained by pressing A at the OS partition editor, called Disklabel) does not leave any unallocated space. Each partition that will be journaled, requires another partition for the journal. Since the `/usr` partition is the largest, it makes sense to shrink this partition slightly, to obtain the space required for journaling.

In our example, an 80 GB disk is used. The following screenshot shows the default partitions created by Disklabel during installation:

If this is more or less what you need, it is very easy to adjust for journaling. Simply use the arrow keys to move the highlight to the `/usr` partition and press D to delete it.

Now, move the highlight to the disk name at the top of the screen and press **C** to create a new partition for `/usr`. This new partition should be smaller by 1 GB (if you intend to journal `/usr` only), or 2 GB (if you intend to journal both `/usr` and `/var`). From the pop-up that appears, opt to create a file system, and type `/usr` as the mount point.

Note

Should you journal the `/var` partition? Normally, journaling makes sense on quite large partitions. You may decide not to journal `/var`, although doing so on a typical desktop will cause no harm. If the file system is lightly used (quite probable for a desktop) you may wish to allocate less disk space for its journal.

In our example, we journal both `/usr` and `/var`. You may of course adjust the procedure to your own needs.

To keep things as easy going as possible, we are going to use `sysinstall` to create the partitions required for journaling. However, during installation, `sysinstall` insists on asking a mount point for each partition you create. At this point, you do not have any mount points for the partitions that will hold the journals, and in reality you *do not even need them*. These are not partitions that we are ever going to mount somewhere.

To avoid these problems with `sysinstall`, we are going to create the journal partitions as swap space. Swap is never mounted, and `sysinstall` has no problem creating as many swap partitions as needed. After the first reboot, `/etc/fstab` will have to be edited, and the extra swap space entries removed.

To create the swap, again use the arrow keys to move the highlight to the top of Disklabel screen, so that the disk name itself is highlighted. Then press **N**, enter the desired size (1024M), and select “swap space” from the pop-up menu that appears. Repeat for every journal you wish to create. In our example, we create two partitions to provide for the journals of `/usr` and `/var`. The final result is shown in the following screenshot:

When you have completed creating the partitions, we suggest you write down the partition names, and mount points, so you can easily refer to this information during the configuration phase. This will help alleviate mistakes that may damage your installation. The following table shows our notes for the sample configuration:

Partition	Mount Point	Journal
ad0s1d	/var	ad0s1h
ad0s1f	/usr	ad0s1g

Table: Partitions and Journals

Continue the installation as you would normally do. We would however suggest you postpone installation of third party software (packages) until you have completely setup journaling.

53.3.2 Booting for the first time

Your system will come up normally, but you will need to edit `/etc/fstab` and remove the extra swap partitions you created for the journals. Normally, the swap partition you will actually use is the one with the “b” suffix (i.e. `ad0s1b` in our example). Remove all other swap space entries and reboot so that OS will stop using them.

When the system comes up again, we will be ready to configure journaling.

53.4 Setting Up Journaling

53.4.1 Executing `gjournal`

Having prepared all the required partitions, it is quite easy to configure journaling. We will need to switch to single user mode, so login as root and type:

```
PROMPT.ROOT shutdown now
```

Press Enter to get the default shell. We will need to unmount the partitions that will be journaled, in our example /usr and /var:

```
PROMPT.ROOT umount /usr /var
```

Load the module required for journaling:

```
PROMPT.ROOT gjournal load
```

Now, use your notes to determine which partition will be used for each journal. In our example, /usr is ad0s1f and its journal will be ad0s1g, while /var is ad0s1d and will be journaled to ad0s1h. The following commands are required:

```
PROMPT.ROOT gjournal label ad0s1f ad0s1g

GEOM_JOURNAL: Journal 2948326772: ad0s1f contains data.
GEOM_JOURNAL: Journal 2948326772: ad0s1g contains journal.

PROMPT.ROOT gjournal label ad0s1d ad0s1h

GEOM_JOURNAL: Journal 3193218002: ad0s1d contains data.
GEOM_JOURNAL: Journal 3193218002: ad0s1h contains journal.

**Note**

If the last sector of either partition is used, ``gjournal`` will
return an error. You will have to run the command using the ``-f``
flag to force an overwrite, i.e.:

::

    PROMPT.ROOT gjournal label -f ad0s1d ad0s1h
```

Since this is a new installation, it is highly unlikely that anything will be actually overwritten.

At this point, two new devices are created, namely ad0s1d.journal and ad0s1f.journal. These represent the /var and /usr partitions we have to mount. Before mounting, we must however set the journal flag on them and clear the Soft Updates flag:

```
PROMPT.ROOT tuneefs -J enable -n disable ad0s1d.journal

tuneefs: gjournal set
tuneefs: soft updates cleared

PROMPT.ROOT tuneefs -J enable -n disable ad0s1f.journal

tuneefs: gjournal set
tuneefs: soft updates cleared
```

Now, mount the new devices manually at their respective places (note that we can now use the async mount option):

```
PROMPT.ROOT mount -o async /dev/ad0s1d.journal /var
PROMPT.ROOT mount -o async /dev/ad0s1f.journal /usr
```

Edit /etc/fstab and update the entries for /usr and /var:

```
/dev/ad0s1f.journal  /usr      ufs      rw,async  2      2
/dev/ad0s1d.journal  /var      ufs      rw,async  2      2
```

****Warning****

Make sure the above entries are correct, or you will have trouble starting up normally after you reboot!

Finally, edit `/boot/loader.conf` and add the following line so the `MAN.GJOURNAL.8` module is loaded at every boot:

```
geom_journal_load="YES"
```

Congratulations! Your system is now set for journaling. You can either type `exit` to return to multi-user mode, or reboot to test your configuration (recommended). During the boot you will see messages like the following:

```
ad0: 76293MB XEC XE800JD-00HBC0 08.02D08 at ata0-master SATA150
GEOM_JOURNAL: Journal 2948326772: ad0s1g contains journal.
GEOM_JOURNAL: Journal 3193218002: ad0s1h contains journal.
GEOM_JOURNAL: Journal 3193218002: ad0s1d contains data.
GEOM_JOURNAL: Journal ad0s1d clean.
GEOM_JOURNAL: Journal 2948326772: ad0s1f contains data.
GEOM_JOURNAL: Journal ad0s1f clean.
```

After an unclean shutdown, the messages will vary slightly, i.e.:

```
GEOM_JOURNAL: Journal ad0s1d consistent.
```

This usually means that `MAN.GJOURNAL.8` used the information in the journal provider to return the file system to a consistent state.

53.4.2 Journaling Newly Created Partitions

While the above procedure is necessary for journaling partitions that already contain data, journaling an empty partition is somewhat easier, since both the data and the journal provider can be stored in the same partition. For example, assume a new disk was installed, and a new partition `/dev/ad1s1d` was created. Creating the journal would be as simple as:

```
PROMPT.ROOT gjournal label ad1s1d
```

The journal size will be 1 GB by default. You may adjust it by using the `-s` option. The value can be given in bytes, or appended by `K`, `M` or `G` to denote Kilobytes, Megabytes or Gigabytes respectively. Note that `gjournal` will not allow you to create unsuitably small journal sizes.

For example, to create a 2 GB journal, you could use the following command:

```
PROMPT.ROOT gjournal label -s 2G ad1s1d
```

You can then create a file system on your new partition, and enable journaling using the `-J` option:

```
PROMPT.ROOT newfs -J /dev/ad1s1d.journal
```

53.4.3 Building Journaling into Your Custom Kernel

If you do not wish to load `geom_journal` as a module, you can build its functions right into your kernel. Edit your custom kernel configuration file, and make sure it includes these two lines:

```
options UFS_GJOURNAL # Note: This is already in GENERIC
options GEOM_JOURNAL # You will have to add this one
```

Rebuild and reinstall your kernel following the relevant instructions in the OS Handbook.

Do not forget to remove the relevant “load” entry from `/boot/loader.conf` if you have previously used it.

53.5 Troubleshooting Journaling

The following section covers frequently asked questions regarding problems related to journaling.

Q: I am getting kernel panics during periods of high disk activity. How is this related to journaling?

A: The journal probably fills up before it has a chance to get committed (flushed) to disk. Keep in mind the size of the journal depends on the usage load, and not the size of the data provider. If your disk activity is high, you need a larger partition for the journal. See the note in the *Understanding Journaling* section.

Q: I made some mistake during configuration, and I cannot boot normally now. Can this be fixed some way?

A: You either forgot (or misspelled) the entry in `/boot/loader.conf`, or there are errors in your `/etc/fstab` file. These are usually easy to fix. Press Enter to get to the default single user shell. Then locate the root of the problem:

```
PROMPT.ROOT cat /boot/loader.conf
```

If the `geom_journal_load` entry is missing or misspelled, the journaled devices are never created. Load the module manually, mount all partitions, and continue with multi-user boot:

```
PROMPT.ROOT gjournal load

GEOM_JOURNAL: Journal 2948326772: ad0s1g contains journal.
GEOM_JOURNAL: Journal 3193218002: ad0s1h contains journal.
GEOM_JOURNAL: Journal 3193218002: ad0s1d contains data.
GEOM_JOURNAL: Journal ad0s1d clean.
GEOM_JOURNAL: Journal 2948326772: ad0s1f contains data.
GEOM_JOURNAL: Journal ad0s1f clean.

PROMPT.ROOT mount -a
PROMPT.ROOT exit
(boot continues)
```

If, on the other hand, this entry is correct, have a look at `/etc/fstab`. You will probably find a misspelled or missing entry. In this case, mount all remaining partitions by hand and continue with the multi-user boot.

Q: Can I remove journaling and return to my standard file system with Soft Updates?

A: Sure. Use the following procedure, which reverses the changes. The partitions you created for the journal providers can then be used for other purposes, if you so wish.

Login as root and switch to single user mode:

```
PROMPT.ROOT shutdown now
```

Unmount the journaled partitions:

```
PROMPT.ROOT umount /usr /var
```

Synchronize the journals:


```
PROMPT.ROOT gjournal sync
```

Stop the journaling providers:

```
PROMPT.ROOT gjournal stop ad0s1d.journal
PROMPT.ROOT gjournal stop ad0s1f.journal
```

Clear journaling metadata from all the devices used:

```
PROMPT.ROOT gjournal clear ad0s1d
PROMPT.ROOT gjournal clear ad0s1f
PROMPT.ROOT gjournal clear ad0s1g
PROMPT.ROOT gjournal clear ad0s1h
```

Clear the file system journaling flag, and restore the Soft Updates flag:

```
PROMPT.ROOT tuneefs -J disable -n enable ad0s1d

tuneefs: gjournal cleared
tuneefs: soft updates set

PROMPT.ROOT tuneefs -J disable -n enable ad0s1f

tuneefs: gjournal cleared
tuneefs: soft updates set
```

Remount the old devices by hand:

```
PROMPT.ROOT mount -o rw /dev/ad0s1d /var
PROMPT.ROOT mount -o rw /dev/ad0s1f /usr
```

Edit `/etc/fstab` and restore it to its original state:

/dev/ad0s1f	/usr	ufs	rw	2	2
/dev/ad0s1d	/var	ufs	rw	2	2

Finally, edit `/boot/loader.conf`, remove the entry that loads the `geom_journal` module and reboot.

53.6 Further Reading

Journaling is a fairly new feature of OS, and as such, it is not very well documented yet. You may however find the following additional references useful:

- A new section on journaling is now part of the OS Handbook.
- [This post](#) in A.CURRENT.NAME by MAN.GJOURNAL.8's developer, A.PJD.EMAIL.
- [This post](#) in A.QUESTIONS.NAME by A.IVORAS.EMAIL.
- The manual pages of MAN.GJOURNAL.8 and MAN.GEOM.8.

Mirroring FreeBSD

Author JunKuriyama

Author ValentinoVaschetto

Author DanielLang

Author KenSmith

Note

We are not accepting new mirrors at this time.

54.1 Contact Information

The Mirror System Coordinators can be reached through email at mirror-admin@FreeBSD.org. There is also a A.HUBS.

54.2 Requirements for FreeBSD mirrors

54.2.1 Disk Space

Disk space is one of the most important requirements. Depending on the set of releases, architectures, and degree of completeness you want to mirror, a huge amount of disk space may be consumed. Also keep in mind that *official* mirrors are probably required to be complete. The web pages should always be mirrored completely. Also note that the numbers stated here are reflecting the current state (at REL2.CURRENT-RELEASE/REL.CURRENT-RELEASE). Further development and releases will only increase the required amount. Also make sure to keep some (ca. 10-20%) extra space around just to be sure. Here are some approximate figures:

- Full FTP Distribution: 1.4 TB
- CTM deltas: 10 GB
- Web pages: 1GB

The current disk usage of FTP Distribution can be found at <ftp://ftp.FreeBSD.org/pub/FreeBSD/dir.sizes>.

54.2.2 Network Connection/Bandwidth

Of course, you need to be connected to the Internet. The required bandwidth depends on your intended use of the mirror. If you just want to mirror some parts of FreeBSD for local use at your site/intranet, the demand may be much smaller than if you want to make the files publicly available. If you intend to become an official mirror, the bandwidth required will be even higher. We can only give rough estimates here:

- Local site, no public access: basically no minimum, but < 2 Mbps could make syncing too slow.
- Unofficial public site: 34 Mbps is probably a good start.
- Official site: > 100 Mbps is recommended, and your host should be connected as close as possible to your border router.

54.2.3 System Requirements, CPU, RAM

One thing this depends on the expected number of clients, which is determined by the server's policy. It is also affected by the types of services you want to offer. Plain FTP or HTTP services may not require a huge amount of resources. Watch out if you provide rsync. This can have a huge impact on CPU and memory requirements as it is considered a memory hog. The following are just examples to give you a very rough hint.

For a moderately visited site that offers rsync, you might consider a current CPU with around 800MHz - 1 GHz, and at least 512MB RAM. This is probably the minimum you want for an *official* site.

For a frequently used site you definitely need more RAM (consider 2GB as a good start) and possibly more CPU, which could also mean that you need to go for a SMP system.

You also want to consider a fast disk subsystem. Operations on the SVN repository require a fast disk subsystem (RAID is highly advised). A SCSI controller that has a cache of its own can also speed up things since most of these services incur a large number of small modifications to the disk.

54.2.4 Services to offer

Every mirror site is required to have a set of core services available. In addition to these required services, there are a number of optional services that server administrators may choose to offer. This section explains which services you can provide and how to go about implementing them.

FTP (required for FTP fileset)

This is one of the most basic services, and it is required for each mirror offering public FTP distributions. FTP access must be anonymous, and no upload/download ratios are allowed (a ridiculous thing anyway). Upload capability is not required (and *must* never be allowed for the FreeBSD file space). Also the FreeBSD archive should be available under the path `/pub/FreeBSD`.

There is a lot of software available which can be set up to allow anonymous FTP (in alphabetical order).

- `/usr/libexec/ftpd`: FreeBSD's own ftpd can be used. Be sure to read `MAN.FTPD.8`.
- `ftp/ncftpd`: A commercial package, free for educational use.
- `ftp/oftpd`: An ftpd designed with security as a main focus.
- `ftp/proftpd`: A modular and very flexible ftpd.
- `ftp/pure-ftpd`: Another ftpd developed with security in mind.
- `ftp/twoftpd`: As above.

- ftp/vsftpd: The “very secure” ftpd.

FreeBSD’s ftpd, proftpd and maybe ncftpd are among the most commonly used FTPds. The others do not have a large userbase among mirror sites. One thing to consider is that you may need flexibility in limiting how many simultaneous connections are allowed, thus limiting how much network bandwidth and system resources are consumed.

Rsync (optional for FTP fileset)

Rsync is often offered for access to the contents of the FTP area of FreeBSD, so other mirror sites can use your system as their source. The protocol is different from FTP in many ways. It is much more bandwidth friendly, as only differences between files are transferred instead of whole files when they change. Rsync does require a significant amount of memory for each instance. The size depends on the size of the synced module in terms of the number of directories and files. Rsync can use `rsh` and `ssh` (now default) as a transport, or use its own protocol for stand-alone access (this is the preferred method for public rsync servers). Authentication, connection limits, and other restrictions may be applied. There is just one software package available:

- net/rsync

HTTP (required for web pages, optional for FTP fileset)

If you want to offer the FreeBSD web pages, you will need to install a web server. You may optionally offer the FTP fileset via HTTP. The choice of web server software is left up to the mirror administrator. Some of the most popular choices are:

- www/apache22: Apache is the most widely deployed web server on the Internet. It is used extensively by the FreeBSD Project.
- www/thttpd: If you are going to be serving a large amount of static content you may find that using an application such as thttpd is more efficient than Apache. It is optimized for excellent performance on FreeBSD.
- www/boa: Boa is another alternative to thttpd and Apache. It should provide considerably better performance than Apache for purely static content. It does not, at the time of this writing, contain the same set of optimizations for FreeBSD that are found in thttpd.
- www/nginx: Nginx is a high performance edge web server with a low memory footprint and key features to build a modern and efficient web infrastructure. Features include a HTTP server, HTTP and mail reverse proxy, caching, load balancing, compression, request throttling, connection multiplexing and reuse, SSL offload and HTTP media streaming.

54.3 How to Mirror FreeBSD

Ok, now you know the requirements and how to offer the services, but not how to get it. :-) This section explains how to actually mirror the various parts of FreeBSD, what tools to use, and where to mirror from.

54.3.1 Mirroring the FTP site

The FTP area is the largest amount of data that needs to be mirrored. It includes the *distribution sets* required for network installation, the *branches* which are actually snapshots of checked-out source trees, the *ISO Images* to write CD-ROMs with the installation distribution, a live file system, and a snapshot of the ports tree. All of course for various FreeBSD versions, and various architectures.

The best way to mirror the FTP area is rsync. You can install the port net/rsync and then use rsync to sync with your upstream host. rsync is already mentioned in ?. Since rsync access is not required, your preferred upstream site may not allow it. You may need to hunt around a little bit to find a site that allows rsync access.

Note

Since the number of rsync clients will have a significant impact on the server machine, most admins impose limitations on their server. For a mirror, you should ask the site maintainer you are syncing from about their policy, and maybe an exception for your host (since you are a mirror).

A command line to mirror FreeBSD might look like:

```
PROMPT.USER rsync -vaHz --delete rsync://ftp4.de.FreeBSD.org/FreeBSD/ /pub/FreeBSD/
```

Consult the documentation for rsync, which is also available at <http://rsync.samba.org/>, about the various options to be used with rsync. If you sync the whole module (unlike subdirectories), be aware that the module-directory (here “FreeBSD”) will not be created, so you cannot omit the target directory. Also you might want to set up a script framework that calls such a command via MAN.CRON.8.

54.3.2 Mirroring the WWW pages

The FreeBSD website should only be mirrored via rsync.

A command line to mirror the FreeBSD web site might look like:

```
PROMPT.USER rsync -vaHz --delete rsync://bit0.us-west.freebsd.org/FreeBSD-www-data/ /usr/local/www/
```

54.3.3 Mirroring Packages

Due to very high requirements of bandwidth, storage and administration the OS Project has decided not to allow public mirrors of packages. For sites with lots of machines, it might be advantageous to run a caching HTTP proxy for the MAN.PKG.8 process. Alternatively specific packages and their dependencies can be fetched by running something like the following:

```
PROMPT.USER pkg fetch -d -o /usr/local/mirror vim
```

Once those packages have been fetched, the repository metadata must be generated by running:

```
PROMPT.USER pkg repo /usr/local/mirror
```

Once the packages have been fetched and the metadata for the repository has been generated, serve the packages up to the client machines via HTTP. For additional information see the man pages for MAN.PKG.8, specifically the MAN.PKG-REPO.8 page.

54.3.4 How often should I mirror?

Every mirror should be updated at a minimum of once per day. Certainly a script with locking to prevent multiple runs happening at the same time will be needed to run from MAN.CRON.8. Since nearly every admin does this in their own way, specific instructions cannot be provided. It could work something like this:

Put the command to run your mirroring application in a script. Use of a plain `/bin/sh` script is recommended.

Add some output redirections so diagnostic messages are logged to a file.

Test if your script works. Check the logs.

Use MAN.CRONTAB.1 to add the script to the appropriate user’s MAN.CRONTAB.5. This should be a different user than what your FTP daemon runs as so that if file permissions inside your FTP area are not world-readable those files can not be accessed by anonymous FTP. This is used to “stage” releases — making sure all of the official mirror sites have all of the necessary release files on release day.

Here are some recommended schedules:

- FTP fileset: daily
- WWW pages: daily

54.4 Where to mirror from

This is an important issue. So this section will spend some effort to explain the backgrounds. We will say this several times: under no circumstances should you mirror from ftp.FreeBSD.org.

54.4.1 A few words about the organization

Mirrors are organized by country. All official mirrors have a DNS entry of the form ftpN.CC.FreeBSD.org. *CC* (i.e. country code) is the *top level domain* (TLD) of the country where this mirror is located. *N* is a number, telling that the host would be the *Nth* mirror in that country. (Same applies to wwwN.CC.FreeBSD.org, etc.) There are mirrors with no *CC* part. These are the mirror sites that are very well connected and allow a large number of concurrent users. ftp.FreeBSD.org is actually two machines, one currently located in Denmark and the other in the United States. It is *NOT* a master site and should never be used to mirror from. Lots of online documentation leads “interactive” users to ftp.FreeBSD.org so automated mirroring systems should find a different machine to mirror from.

Additionally there exists a hierarchy of mirrors, which is described in terms of *tiers*. The master sites are not referred to but can be described as *Tier-0*. Mirrors that mirror from these sites can be considered *Tier-1*, mirrors of *Tier-1*-mirrors, are *Tier-2*, etc. Official sites are encouraged to be of a low *tier*, but the lower the tier the higher the requirements in terms as described in ?. Also access to low-tier-mirrors may be restricted, and access to master sites is definitely restricted. The *tier*-hierarchy is not reflected by DNS and generally not documented anywhere except for the master sites. However, official mirrors with low numbers like 1-4, are usually *Tier-1* (this is just a rough hint, and there is no rule).

54.4.2 Ok, but where should I get the stuff now?

Under no circumstances should you mirror from ftp.FreeBSD.org. The short answer is: from the site that is closest to you in Internet terms, or gives you the fastest access.

I just want to mirror from somewhere!

If you have no special intentions or requirements, the statement in ? applies. This means:

Check for those which provide fastest access (number of hops, round-trip-times) and offer the services you intend to use (like rsync).

Contact the administrators of your chosen site stating your request, and asking about their terms and policies.

Set up your mirror as described above.

I am an official mirror, what is the right site for me?

In general the description in ? still applies. Of course you may want to put some weight on the fact that your upstream should be of a low tier. There are some other considerations about *official* mirrors that are described in ?.

I want to access the master sites!

If you have good reasons and good prerequisites, you may want and get access to one of the master sites. Access to these sites is generally restricted, and there are special policies for access. If you are already an *official* mirror, this certainly helps you getting access. In any other case make sure your country really needs another mirror. If it already has three or more, ask the “zone administrator” (hostmaster@CC.FreeBSD.org) or A.HUBS first.

Whoever helped you become, an *official* should have helped you gain access to an appropriate upstream host, either one of the master sites or a suitable Tier-1 site. If not, you can send email to mirror-admin@FreeBSD.org to request help with that.

There is one master site for the FTP fileset.

<ftp-master.FreeBSD.org>

This is the master site for the FTP fileset.

<ftp-master.FreeBSD.org> provides rsync access, in addition to FTP. Refer to ?.

Mirrors are also encouraged to allow rsync access for the FTP contents, since they are *Tier-1*-mirrors.

54.5 Official Mirrors

Official mirrors are mirrors that

- 1. have a FreeBSD.org DNS entry (usually a CNAME).
- b) are listed as an official mirror in the FreeBSD documentation (like handbook).

So far to distinguish official mirrors. Official mirrors are not necessarily *Tier-1*-mirrors. However you probably will not find a *Tier-1*-mirror, that is not also official.

54.5.1 Special Requirements for official (tier-1) mirrors

It is not so easy to state requirements for all official mirrors, since the project is sort of tolerant here. It is more easy to say, what *official tier-1 mirrors* are required to. All other official mirrors can consider this a big *should*.

Tier-1 mirrors are required to:

- carry the complete fileset
- allow access to other mirror sites
- provide FTP and rsync access

Furthermore, admins should be subscribed to the A.HUBS. See [this link](#) for details, how to subscribe.

Important

It is *very* important for a hub administrator, especially Tier-1 hub admins, to check the [release schedule](#) for the next FreeBSD release. This is important because it will tell you when the next release is scheduled to come out, and thus giving you time to prepare for the big spike of traffic which follows it.

It is also important that hub administrators try to keep their mirrors as up-to-date as possible (again, even more crucial for Tier-1 mirrors). If Mirror1 does not update for a while, lower tier mirrors will begin to mirror old data from Mirror1 and thus begins a downward spiral... Keep your mirrors up to date!

54.5.2 How to become official then?

We are not accepting any new mirrors at this time.

54.6 Some statistics from mirror sites

Here are links to the stat pages of your favorite mirrors (a.k.a. the only ones who feel like providing stats).

54.6.1 FTP site statistics

- <ftp.is.FreeBSD.org> - hostmaster@is.FreeBSD.org - (Bandwidth) (FTP processes) (HTTP processes)
- <ftp2.ru.FreeBSD.org> - mirror@macomnet.ru - (Bandwidth) (HTTP and FTP users)

Independent Verification of IPsec Functionality in FreeBSD

Author DavidHonig

55.1 The Problem

First, let's assume you have *installed *IPsec**. How do you know it is *working*? Sure, your connection will not work if it is misconfigured, and it will work when you finally get it right. MAN.NETSTAT.1 will list it. But can you independently confirm it?

55.2 The Solution

First, some crypto-relevant info theory:

1. Encrypted data is uniformly distributed, i.e., has maximal entropy per symbol;
2. Raw, uncompressed data is typically redundant, i.e., has sub-maximal entropy.

Suppose you could measure the entropy of the data to- and from- your network interface. Then you could see the difference between unencrypted data and encrypted data. This would be true even if some of the data in “encrypted mode” was not encrypted—as the outermost IP header must be if the packet is to be routable.

55.2.1 MUST

Ueli Maurer's “Universal Statistical Test for Random Bit Generators”(MUST) quickly measures the entropy of a sample. It uses a compression-like algorithm. *The code is given below* for a variant which measures successive (~quarter megabyte) chunks of a file.

55.2.2 Tcpdump

We also need a way to capture the raw network data. A program called MAN.TCPDUMP.1 lets you do this, if you have enabled the *Berkeley Packet Filter* interface in your *kernel's config file*.

The command:

```
tcpdump -c 4000 -s 10000 -w dumpfile.bin
```

will capture 4000 raw packets to dumpfile.bin. Up to 10,000 bytes per packet will be captured in this example.

55.3 The Experiment

Here is the experiment:

Open a window to an IPsec host and another window to an insecure host.

Now start *capturing packets*.

In the “secure” window, run the UNIX command `MAN.YES.1`, which will stream the `y` character. After a while, stop this. Switch to the insecure window, and repeat. After a while, stop.

Now run *MUST* on the captured packets. You should see something like the following. The important thing to note is that the secure connection has 93% (6.7) of the expected value (7.18), and the “normal” connection has 29% (2.1) of the expected value.

```
PROMPT.USER tcpdump -c 4000 -s 10000 -w ipsecdemo.bin
PROMPT.USER uliscan ipsecdemo.bin

Uliscan 21 Dec 98
L=8 256 258560
Measuring file ipsecdemo.bin
Init done
Expected value for L=8 is 7.1836656
6.9396 -----
6.6177 -----
6.4100 -----
2.1101 -----
2.0838 -----
2.0983 -----
```

55.4 Caveat

This experiment shows that IPsec *does* seem to be distributing the payload data *uniformly*, as encryption should. However, the experiment described here *cannot* detect many possible flaws in a system (none of which do I have any evidence for). These include poor key generation or exchange, data or keys being visible to others, use of weak algorithms, kernel subversion, etc. Study the source; know the code.

55.5 IPsec—Definition

Internet Protocol security extensions to IPv4; required for IPv6. A protocol for negotiating encryption and authentication at the IP (host-to-host) level. SSL secures only one application socket; SSH secures only a login; PGP secures only a specified file or message. IPsec encrypts everything between two hosts.

55.6 Installing IPsec

Most of the modern versions of FreeBSD have IPsec support in their base source. So you will need to include the `IPSEC` option in your kernel config and, after kernel rebuild and reinstall, configure IPsec connections using `MAN.SETKEY.8` command.

A comprehensive guide on running IPsec on FreeBSD is provided in FreeBSD Handbook.

55.7 src/sys/i386/conf/KERNELNAME

This needs to be present in the kernel config file in order to capture network data with MAN.TCPDUMP.1. Be sure to run MAN.CONFIG.8 after adding this, and rebuild and reinstall.

```
device bpf
```

55.8 Maurer's Universal Statistical Test (for block size=8 bits)

You can find the same code at [this link](#).

```
/*
  ULISCAN.c  ---blocksize of 8

  1 Oct 98
  1 Dec 98
  21 Dec 98      uliscan.c derived from ueli8.c

  This version has // comments removed for Sun cc

  This implements Ueli M Maurer's "Universal Statistical Test for Random
  Bit Generators" using L=8

  Accepts a filename on the command line; writes its results, with other
  info, to stdout.

  Handles input file exhaustion gracefully.

  Ref: J. Cryptology v 5 no 2, 1992 pp 89-105
  also on the web somewhere, which is where I found it.

  -David Honig
  honig@sprynet.com

  Usage:
  ULISCAN filename
  outputs to stdout
*/

#define L 8
#define V (1<<L)
#define Q (10*V)
#define K (100 *Q)
#define MAXSAMP (Q + K)

#include <stdio.h>
#include <math.h>

int main(argc, argv)
int argc;
char **argv;
{
  FILE *fptr;
  int i, j;
  int b, c;
  int table[V];
```

```
double sum = 0.0;
int iproduct = 1;
int run;

extern double    log(/* double x */);

printf("Ulisca 21 Dec 98 \nL=%d %d %d \n", L, V, MAXSAMP);

if (argc < 2) {
    printf("Usage: Ulisca filename\n");
    exit(-1);
} else {
    printf("Measuring file %s\n", argv[1]);
}

fptr = fopen(argv[1], "rb");

if (fptr == NULL) {
    printf("Can't find %s\n", argv[1]);
    exit(-1);
}

for (i = 0; i < V; i++) {
    table[i] = 0;
}

for (i = 0; i < Q; i++) {
    b = fgetc(fptr);
    table[b] = i;
}

printf("Init done\n");

printf("Expected value for L=8 is 7.1836656\n");

run = 1;

while (run) {
    sum = 0.0;
    iproduct = 1;

    if (run)
        for (i = Q; run && i < Q + K; i++) {
            j = i;
            b = fgetc(fptr);

            if (b < 0)
                run = 0;

            if (run) {
                if (table[b] > j)
                    j += K;

                sum += log((double)(j-table[b]));

                table[b] = i;
            }
        }
}
```

```
if (!run)
    printf("Premature end of file; read %d blocks.\n", i - Q);

sum = (sum/((double)(i - Q))) / log(2.0);
printf("%4.4f ", sum);

for (i = 0; i < (int)(sum*8.0 + 0.50); i++)
    printf("-");

printf("\n");

/* refill initial table */
if (0) {
    for (i = 0; i < Q; i++) {
        b = fgetc(fp);
        if (b < 0) {
            run = 0;
        } else {
            table[b] = i;
        }
    }
}
}
```

LDAP Authentication

Author Toby Burress

56.1 Preface

This document is intended to give the reader enough of an understanding of LDAP to configure an LDAP server. This document will attempt to provide an explanation of `net/nss_ldap` and `security/pam_ldap` for use with client machines services for use with the LDAP server.

When finished, the reader should be able to configure and deploy a OS server that can host an LDAP directory, and to configure and deploy a OS server which can authenticate against an LDAP directory.

This article is not intended to be an exhaustive account of the security, robustness, or best practice considerations for configuring LDAP or the other services discussed herein. While the author takes care to do everything correctly, they do not address security issues beyond a general scope. This article should be considered to lay the theoretical groundwork only, and any actual implementation should be accompanied by careful requirement analysis.

56.2 Configuring LDAP

LDAP stands for “Lightweight Directory Access Protocol” and is a subset of the X.500 Directory Access Protocol. Its most recent specifications are in [RFC4510](#) and friends. Essentially it is a database that expects to be read from more often than it is written to.

The LDAP server [OpenLDAP](#) will be used in the examples in this document; while the principles here should be generally applicable to many different servers, most of the concrete administration is OpenLDAP-specific. There are several server versions in ports, for example `net/openldap24-server`. Client servers will need the corresponding `net/openldap24-client` libraries.

There are (basically) two areas of the LDAP service which need configuration. The first is setting up a server to receive connections properly, and the second is adding entries to the server’s directory so that OS tools know how to interact with it.

56.2.1 Setting Up the Server for Connections

Note

This section is specific to OpenLDAP. If you are using another server, you will need to consult that server’s documentation.

Installing OpenLDAP

First, install OpenLDAP:

```
PROMPT.ROOT cd /usr/ports/net/openldap24-server
PROMPT.ROOT make install clean
```

This installs the `slapd` and `slurpd` binaries, along with the required OpenLDAP libraries.

Configuring OpenLDAP

Next we must configure OpenLDAP.

You will want to require encryption in your connections to the LDAP server; otherwise your users' passwords will be transferred in plain text, which is considered insecure. The tools we will be using support two very similar kinds of encryption, SSL and TLS.

TLS stands for "Transportation Layer Security". Services that employ TLS tend to connect on the *same* ports as the same services without TLS; thus an SMTP server which supports TLS will listen for connections on port 25, and an LDAP server will listen on 389.

SSL stands for "Secure Sockets Layer", and services that implement SSL do *not* listen on the same ports as their non-SSL counterparts. Thus SMTPS listens on port 465 (not 25), HTTPS listens on 443, and LDAPS on 636.

The reason SSL uses a different port than TLS is because a TLS connection begins as plain text, and switches to encrypted traffic after the `STARTTLS` directive. SSL connections are encrypted from the beginning. Other than that there are no substantial differences between the two.

Note

We will adjust OpenLDAP to use TLS, as SSL is considered deprecated.

Once OpenLDAP is installed via ports, the following configuration parameters in `/usr/local/etc/openldap/slapd.conf` will enable TLS:

```
security ssf=128

TLSCertificateFile /path/to/your/cert.crt
TLSCertificateKeyFile /path/to/your/cert.key
TLSCACertificateFile /path/to/your/cacert.crt
```

Here, `ssf=128` tells OpenLDAP to require 128-bit encryption for all connections, both search and update. This parameter may be configured based on the security needs of your site, but rarely you need to weaken it, as most LDAP client libraries support strong encryption.

The `cert.crt`, `cert.key`, and `cacert.crt` files are necessary for clients to authenticate *you* as the valid LDAP server. If you simply want a server that runs, you can create a self-signed certificate with OpenSSL:

```
PROMPT.USER openssl genrsa -out cert.key 1024
Generating RSA private key, 1024 bit long modulus
.....++++++
...++++++
e is 65537 (0x10001)
PROMPT.USER openssl req -new -key cert.key -out cert.csr
```

At this point you should be prompted for some values. You may enter whatever values you like; however, it is important the "Common Name" value be the fully qualified domain name of the OpenLDAP server. In our case, and the examples here, the server is `server.example.org`. Incorrectly setting this value will cause clients to fail when making connections. This can be the cause of great frustration, so ensure that you follow these steps closely.

Finally, the certificate signing request needs to be signed:

```
PROMPT.USER openssl x509 -req -in cert.csr -days 365 -signkey cert.key -out cert.crt
Signature ok
subject=/C=AU/ST=Some-State/O=Internet Widgits Pty Ltd
Getting Private key
```

This will create a self-signed certificate that can be used for the directives in `slapd.conf`, where `cert.crt` and `cacert.crt` are the same file. If you are going to use many OpenLDAP servers (for replication via `slurpd`) you will want to see ? to generate a CA key and use it to sign individual server certificates.

Once this is done, put the following in `/etc/rc.conf`:

```
slapd_enable="YES"
```

Then run `“/usr/local/etc/rc.d/slapd start”`. This should start OpenLDAP. Confirm that it is listening on 389 with

```
PROMPT.USER sockstat -4 -p 389
ldap      slapd      3261  7  tcp4      *:389      *:*
```

Configuring the Client

Install the `net/openldap24-client` port for the OpenLDAP libraries. The client machines will always have OpenLDAP libraries since that is all security/pam_ldap and net/nss_ldap support, at least for the moment.

The configuration file for the OpenLDAP libraries is `/usr/local/etc/openldap/ldap.conf`. Edit this file to contain the following values:

```
base dc=example,dc=org
uri ldap://server.example.org/
ssl start_tls
tls_cacert /path/to/your/cacert.crt

**Note**

It is important that your clients have access to ``cacert.crt``,
otherwise they will not be able to connect.

**Note**

There are two files called ``ldap.conf``. The first is this file,
which is for the OpenLDAP libraries and defines how to talk to the
server. The second is ``/usr/local/etc/ldap.conf``, and is for
pam\_ldap.
```

At this point you should be able to run `ldapsearch -Z` on the client machine; `-Z` means “use TLS”. If you encounter an error, then something is configured wrong; most likely it is your certificates. Use `MAN.OPENSSSL.1`’s `s_client` and `s_server` to ensure you have them configured and signed properly.

56.2.2 Entries in the Database

Authentication against an LDAP directory is generally accomplished by attempting to bind to the directory as the connecting user. This is done by establishing a “simple” bind on the directory with the user name supplied. If there is an entry with the `uid` equal to the user name and that entry’s `userPassword` attribute matches the password supplied, then the bind is successful.

The first thing we have to do is figure out is where in the directory our users will live.

The base entry for our database is `dc=example,dc=org`. The default location for users that most clients seem to expect is something like `ou=people,base`, so that is what will be used here. However keep in mind that this is configurable.

So the `ldif` entry for the `people` organizational unit will look like:

```
dn: ou=people,dc=example,dc=org
objectClass: top
objectClass: organizationalUnit
ou: people
```

All users will be created as subentries of this organizational unit.

Some thought might be given to the object class your users will belong to. Most tools by default will use `people`, which is fine if you simply want to provide entries against which to authenticate. However, if you are going to store user information in the LDAP database as well, you will probably want to use `inetOrgPerson`, which has many useful attributes. In either case, the relevant schemas need to be loaded in `slapd.conf`.

For this example we will use the `person` object class. If you are using `inetOrgPerson`, the steps are basically identical, except that the `sn` attribute is required.

To add a user `testuser`, the `ldif` would be:

```
dn: uid=tuser,ou=people,dc=example,dc=org
objectClass: person
objectClass: posixAccount
objectClass: shadowAccount
objectClass: top
uidNumber: 10000
gidNumber: 10000
homeDirectory: /home/tuser
loginShell: /bin/csh
uid: tuser
cn: tuser
```

I start my LDAP users' UIDs at 10000 to avoid collisions with system accounts; you can configure whatever number you wish here, as long as it is less than 65536.

We also need group entries. They are as configurable as user entries, but we will use the defaults below:

```
dn: ou=groups,dc=example,dc=org
objectClass: top
objectClass: organizationalUnit
ou: groups

dn: cn=tuser,ou=groups,dc=example,dc=org
objectClass: posixGroup
objectClass: top
gidNumber: 10000
cn: tuser
```

To enter these into your database, you can use `slapadd` or `ldapadd` on a file containing these entries. Alternatively, you can use `sysutils/ldapvi`.

The `ldapsearch` utility on the client machine should now return these entries. If it does, your database is properly configured to be used as an LDAP authentication server.

56.3 Client Configuration

The client should already have OpenLDAP libraries from `?`, but if you are installing several client machines you will need to install `net/openldap24-client` on each of them.

OS requires two ports to be installed to authenticate against an LDAP server, `security/pam_ldap` and `net/nss_ldap`.

56.3.1 Authentication

`security/pam_ldap` is configured via `/usr/local/etc/ldap.conf`.

Note

This is a *different file* than the OpenLDAP library functions' configuration file, `/usr/local/etc/openldap/ldap.conf`; however, it takes many of the same options; in fact it is a superset of that file. For the rest of this section, references to `ldap.conf` will mean `/usr/local/etc/ldap.conf`.

Thus, we will want to copy all of our original configuration parameters from `openldap/ldap.conf` to the new `ldap.conf`. Once this is done, we want to tell `security/pam_ldap` what to look for on the directory server.

We are identifying our users with the `uid` attribute. To configure this (though it is the default), set the `pam_login_attribute` directive in `ldap.conf`:

```
pam_login_attribute uid
```

With this set, `security/pam_ldap` will search the entire LDAP directory under `base` for the value `uid=username`. If it finds one and only one entry, it will attempt to bind as that user with the password it was given. If it binds correctly, then it will allow access. Otherwise it will fail.

PAM

PAM, which stands for “Pluggable Authentication Modules”, is the method by which OS authenticates most of its sessions. To tell OS we wish to use an LDAP server, we will have to add a line to the appropriate PAM file.

Most of the time the appropriate PAM file is `/etc/pam.d/sshd`, if you want to use SSH (remember to set the relevant options in `/etc/ssh/sshd_config`, otherwise SSH will not use PAM).

To use PAM for authentication, add the line

```
auth sufficient /usr/local/lib/pam_ldap.so no_warn
```

Exactly where this line shows up in the file and which options appear in the fourth column determine the exact behavior of the authentication mechanism; see `MAN.PAM.D.5`

With this configuration you should be able to authenticate a user against an LDAP directory. PAM will perform a bind with your credentials, and if successful will tell SSH to allow access.

However it is not a good idea to allow *every* user in the directory into *every* client machine. With the current configuration, all that a user needs to log into a machine is an LDAP entry. Fortunately there are a few ways to restrict user access.

`ldap.conf` supports a `pam_groupdn` directive; every account that connects to this machine needs to be a member of the group specified here. For example, if you have

```
pam_groupdn cn=servername,ou=accessgroups,dc=example,dc=org
```

in `ldap.conf`, then only members of that group will be able to log in. There are a few things to bear in mind, however.

Members of this group are specified in one or more `memberUid` attributes, and each attribute must have the full distinguished name of the member. So `memberUid: someuser` will not work; it must be:

```
memberUid: uid=someuser,ou=people,dc=example,dc=org
```

Additionally, this directive is not checked in PAM during authentication, it is checked during account management, so you will need a second line in your PAM files under `account`. This will require, in turn, *every* user to be listed in the group, which is not necessarily what we want. To avoid blocking users that are not in LDAP, you should enable the `ignore_unknown_user` attribute. Finally, you should set the `ignore_authinfo_unavail` option so that you are not locked out of every computer when the LDAP server is unavailable.

Your `pam.d/sshd` might then end up looking like this:

```
auth                required      pam_nologin.so      no_warn
auth                sufficient    pam_opie.so         no_warn no_fake_prompts
auth                requisite     pam_opieaccess.so   no_warn allow_local
auth                sufficient    /usr/local/lib/pam_ldap.so   no_warn
auth                required      pam_unix.so         no_warn try_first_pass

account             required      pam_login_access.so
account             required      /usr/local/lib/pam_ldap.so   no_warn ignore_authinfo_unavail ignore_

**Note**

Since we are adding these lines specifically to ``pam.d/sshd``, this
will only have an effect on SSH sessions. LDAP users will be unable
to log in at the console. To change this behavior, examine the other
files in ``/etc/pam.d`` and modify them accordingly.
```

56.3.2 Name Service Switch

NSS is the service that maps attributes to names. So, for example, if a file is owned by user 1001, an application will query NSS for the name of 1001, and it might get `bob` or `ted` or whatever the user's name is.

Now that our user information is kept in LDAP, we need to tell NSS to look there when queried.

The `net/nss_ldap` port does this. It uses the same configuration file as `security/pam_ldap`, and should not need any extra parameters once it is installed. Instead, what is left is simply to edit `/etc/nsswitch.conf` to take advantage of the directory. Simply replace the following lines:

```
group: compat
passwd: compat
```

with

```
group: files ldap
passwd: files ldap
```

This will allow you to map usernames to UIDs and UIDs to usernames.

Congratulations! You should now have working LDAP authentication.

56.3.3 Caveats

Unfortunately, as of the time this was written OS did not support changing user passwords with MAN.PASSWD.1. Because of this, most administrators are left to implement a solution themselves. I provide some examples here. Note that if you write your own password change script, there are some security issues you should be made aware of; see ?

```
#!/bin/sh

stty -echo
read -p "Old Password: " oldp; echo
read -p "New Password: " np1; echo
read -p "Retype New Password: " np2; echo
stty echo

if [ "$np1" != "$np2" ]; then
    echo "Passwords do not match."
    exit 1
fi

ldappasswd -D uid="$USER",ou=people,dc=example,dc=org \
    -w "$oldp" \
    -a "$oldp" \
    -s "$np1"

**Caution**

This script does hardly any error checking, but more important it is
very cavalier about how it stores your passwords. If you do anything
like this, at least adjust the ``security.bsd.see_other_uids``
sysctl value:

::

    PROMPT.ROOT sysctl security.bsd.see_other_uids=0.
```

A more flexible (and probably more secure) approach can be used by writing a custom program, or even a web interface. The following is part of a Ruby library that can change LDAP passwords. It sees use both on the command line, and on the web.

```
require 'ldap'
require 'base64'
require 'digest'
require 'password' # ruby-password

ldap_server = "ldap.example.org"
luser = "uid=#{ENV['USER']},ou=people,dc=example,dc=org"

# get the new password, check it, and create a salted hash from it
def get_password
    pwd1 = Password.get("New Password: ")
    pwd2 = Password.get("Retype New Password: ")

    raise if pwd1 != pwd2
    pwd1.check # check password strength

    salt = rand.to_s.gsub(/0\.\/, '')
    pass = pwd1.to_s
    hash = "{SSHA}" + Base64.encode64(Digest::SHA1.digest("#{pass}#{salt}")) + salt).chomp!
    return hash
```

```
end

oldp = Password.get("Old Password: ")
newp = get_password

# We'll just replace it. That we can bind proves that we either know
# the old password or are an admin.

replace = LDAP::Mod.new(LDAP::LDAP_MOD_REPLACE | LDAP::LDAP_MOD_BVALUES,
                        "userPassword",
                        [newp])

conn = LDAP::SSLConn.new(ldap_server, 389, true)
conn.set_option(LDAP::LDAP_OPT_PROTOCOL_VERSION, 3)
conn.bind(luser, oldp)
conn.modify(luser, [replace])
```

Although not guaranteed to be free of security holes (the password is kept in memory, for example) this is cleaner and more flexible than a simple `sh` script.

56.4 Security Considerations

Now that your machines (and possibly other services) are authenticating against your LDAP server, this server needs to be protected at least as well as `/etc/master.passwd` would be on a regular server, and possibly even more so since a broken or cracked LDAP server would break every client service.

Remember, this section is not exhaustive. You should continually review your configuration and procedures for improvements.

56.4.1 Setting Attributes Read-only

Several attributes in LDAP should be read-only. If left writable by the user, for example, a user could change his `uidNumber` attribute to 0 and get root access!

To begin with, the `userPassword` attribute should not be world-readable. By default, anyone who can connect to the LDAP server can read this attribute. To disable this, put the following in `slapd.conf`:

```
access to dn.subtree="ou=people,dc=example,dc=org"
  attrs=userPassword
  by self write
  by anonymous auth
  by * none

access to *
  by self write
  by * read
```

This will disallow reading of the `userPassword` attribute, while still allowing users to change their own passwords.

Additionally, you'll want to keep users from changing some of their own attributes. By default, users can change any attribute (except for those which the LDAP schemas themselves deny changes), such as `uidNumber`. To close this hole, modify the above to

```
access to dn.subtree="ou=people,dc=example,dc=org"
  attrs=userPassword
  by self write
```



```

    by anonymous auth
    by * none

access to attrs=homeDirectory,uidNumber,gidNumber
    by * read

access to *
    by self write
    by * read

```

This will stop users from being able to masquerade as other users.

56.4.2 root Account Definition

Often the root or manager account for the LDAP service will be defined in the configuration file. OpenLDAP supports this, for example, and it works, but it can lead to trouble if `slapd.conf` is compromised. It may be better to use this only to bootstrap yourself into LDAP, and then define a root account there.

Even better is to define accounts that have limited permissions, and omit a root account entirely. For example, users that can add or remove user accounts are added to one group, but they cannot themselves change the membership of this group. Such a security policy would help mitigate the effects of a leaked password.

Creating a Management Group

Say you want your IT department to be able to change home directories for users, but you do not want all of them to be able to add or remove users. The way to do this is to add a group for these admins:

```

dn: cn=homemanagement,dc=example,dc=org
objectClass: top
objectClass: posixGroup
cn: homemanagement
gidNumber: 121 # required for posixGroup
memberUid: uid=tuser,ou=people,dc=example,dc=org
memberUid: uid=user2,ou=people,dc=example,dc=org

```

And then change the permissions attributes in `slapd.conf`:

```

access to dn.subtree="ou=people,dc=example,dc=org"
    attr=homeDirectory
    by dn="cn=homemanagement,dc=example,dc=org"
    dnattr=memberUid write

```

Now `tuser` and `user2` can change other users' home directories.

In this example we have given a subset of administrative power to certain users without giving them power in other domains. The idea is that soon no single user account has the power of a root account, but every power root had is had by at least one user. The root account then becomes unnecessary and can be removed.

56.4.3 Password Storage

By default OpenLDAP will store the value of the `userPassword` attribute as it stores any other data: in the clear. Most of the time it is base 64 encoded, which provides enough protection to keep an honest administrator from knowing your password, but little else.

It is a good idea, then, to store passwords in a more secure format, such as SSHA (salted SHA). This is done by whatever program you use to change users' passwords.

56.5 Useful Aids

There are a few other programs that might be useful, particularly if you have many users and do not want to configure everything manually.

security/pam_mkhomedir is a PAM module that always succeeds; its purpose is to create home directories for users which do not have them. If you have dozens of client servers and hundreds of users, it is much easier to use this and set up skeleton directories than to prepare every home directory.

sysutils/cpu is a MAN.PW.8-like utility that can be used to manage users in the LDAP directory. You can call it directly, or wrap scripts around it. It can handle both TLS (with the `-x` flag) and SSL (directly).

sysutils/ldapvi is a great utility for editing LDAP values in an LDIF-like syntax. The directory (or subsection of the directory) is presented in the editor chosen by the EDITOR environment variable. This makes it easy to enable large-scale changes in the directory without having to write a custom tool.

security/openssh-portable has the ability to contact an LDAP server to verify SSH keys. This is extremely nice if you have many servers and do not want to copy your public keys across all of them.

56.6 OpenSSL Certificates for LDAP

If you are hosting two or more LDAP servers, you will probably not want to use self-signed certificates, since each client will have to be configured to work with each certificate. While this is possible, it is not nearly as simple as creating your own certificate authority, and signing your servers' certificates with that.

The steps here are presented as they are with very little attempt at explaining what is going on—further explanation can be found in MAN.OPENSLL.1 and its friends.

To create a certificate authority, we simply need a self-signed certificate and key. The steps for this again are

```
PROMPT.USER openssl genrsa -out root.key 1024
PROMPT.USER openssl req -new -key root.key -out root.csr
PROMPT.USER openssl x509 -req -days 1024 -in root.csr -signkey root.key -out root.crt
```

These will be your root CA key and certificate. You will probably want to encrypt the key and store it in a cool, dry place; anyone with access to it can masquerade as one of your LDAP servers.

Next, using the first two steps above create a key `ldap-server-one.key` and certificate signing request `ldap-server-one.csr`. Once you sign the signing request with `root.key`, you will be able to use `ldap-server-one.*` on your LDAP servers.

Note

Do not forget to use the fully qualified domain name for the “common name” attribute when generating the certificate signing request; otherwise clients will reject a connection with you, and it can be very tricky to diagnose.

To sign the key, use `-CA` and `-CAkey` instead of `-signkey`:

```
PROMPT.USER openssl x509 -req -days 1024 \
-in ldap-server-one.csr -CA root.crt -CAkey root.key \
-out ldap-server-one.crt
```

The resulting file will be the certificate that you can use on your LDAP servers.

Finally, for clients to trust all your servers, distribute `root.crt` (the *certificate*, not the key!) to each client, and specify it in the `TLSCACertificateFile` directive in `ldap.conf`.

OS Support for Leap Seconds

57.1 Introduction

A *leap second* is an ad-hoc one-second correction to synchronize atomic timescales with Earth rotation. This article describes how OS interacts with leap seconds.

As of this writing, the next leap second will occur at 2015-Jun-30 23:59:60 UTC. This leap second will occur during a business day for North and South America and the Asia/Pacific region.

Leap seconds are announced by [IERS](#) on [Bulletin C](#).

Standard leap second behavior is described in [RFC 7164](#). Also see `MAN.TIME2POSIX.3`.

57.2 Default Leap Second Handling on OS

The easiest way to handle leap seconds is with the POSIX time rules OS uses by default, combined with NTP. When `MAN.NTPD.8` is running and the time is synchronized with upstream NTP servers that handle leap seconds correctly, the leap second will cause the system time to automatically repeat the last second of the day. No other adjustments are necessary.

If the upstream NTP servers do not handle leap seconds correctly, `MAN.NTPD.8` will step the time by one second after the errant upstream server has noticed and stepped itself.

If NTP is not being used, manual adjustment of the system clock will be required after the leap second has passed.

57.3 Cautions

Leap seconds are inserted at the same instant all over the world: UTC midnight. In Japan that is mid-morning, in the Pacific mid-day, in the Americas late afternoon, and in Europe at night.

We believe and expect that OS, if provided correct and stable NTP service, will work as designed during this leap second, as it did during the previous ones.

However, we caution that practically no applications have ever asked the kernel about leap seconds. Our experience is that, as designed, leap seconds are essentially a replay of the second before the leap second, and this is a surprise to most application programmers.

Other operating systems and other computers may or may not handle the leap-second the same way as OS, and systems without correct and stable NTP service will not know anything about leap seconds at all.

It is not unheard of for computers to crash because of leap seconds, and experience has shown that a large fraction of all public NTP servers might handle and announce the leap second incorrectly.

Please try to make sure nothing horrible happens because of the leap second.

57.4 Testing

It is possible to test whether a leap second will be used. Due to the nature of NTP, the test might work up to 24 hours before the leap second. Some major reference clock sources only announce leap seconds one hour ahead of the event. Query the NTP daemon:

```
PROMPT.USER ntpq -c 'rv 0 leap'
```

Output that includes `leap_add_sec` indicates proper support of the leap second. Before the 24 hours leading up to the leap second, or after the leap second has passed, `leap_none` will be shown.

57.5 Conclusion

In practice, leap seconds are usually not a problem on OS. We hope that this overview helps clarify what to expect and how to make the leap second event proceed more smoothly.

LINUX emulation in OS

Author RomanDivacky

58.1 Introduction

In the last few years the open source UNIX based operating systems started to be widely deployed on server and client machines. Among these operating systems I would like to point out two: OS, for its BSD heritage, time proven code base and many interesting features and LINUX for its wide user base, enthusiastic open developer community and support from large companies. OS tends to be used on server class machines serving heavy duty networking tasks with less usage on desktop class machines for ordinary users. While LINUX has the same usage on servers, but it is used much more by home based users. This leads to a situation where there are many binary only programs available for LINUX that lack support for OS.

Naturally, a need for the ability to run LINUX binaries on a OS system arises and this is what this thesis deals with: the emulation of the LINUX kernel in the OS operating system.

During the Summer of 2006 Google Inc. sponsored a project which focused on extending the LINUX emulation layer (the so called Linuxulator) in OS to include LINUX 2.6 facilities. This thesis is written as a part of this project.

58.2 A look inside...

In this section we are going to describe every operating system in question. How they deal with syscalls, trapframes etc., all the low-level stuff. We also describe the way they understand common UNIX primitives like what a PID is, what a thread is, etc. In the third subsection we talk about how UNIX on UNIX emulation could be done in general.

58.2.1 What is UNIX

UNIX is an operating system with a long history that has influenced almost every other operating system currently in use. Starting in the 1960s, its development continues to this day (although in different projects). UNIX development soon forked into two main ways: the BSDs and System III/V families. They mutually influenced themselves by growing a common UNIX standard. Among the contributions originated in BSD we can name virtual memory, TCP/IP networking, FFS, and many others. The System V branch contributed to SysV interprocess communication primitives, copy-on-write, etc. UNIX itself does not exist any more but its ideas have been used by many other operating systems world wide thus forming the so called UNIX-like operating systems. These days the most influential ones are LINUX, Solaris, and possibly (to some extent) OS. There are in-company UNIX derivatives (AIX, HP-UX etc.), but these have been more and more migrated to the aforementioned systems. Let us summarize typical UNIX characteristics.

58.2.2 Technical details

Every running program constitutes a process that represents a state of the computation. Running process is divided between kernel-space and user-space. Some operations can be done only from kernel space (dealing with hardware etc.), but the process should spend most of its lifetime in the user space. The kernel is where the management of the processes, hardware, and low-level details take place. The kernel provides a standard unified UNIX API to the user space. The most important ones are covered below.

Communication between kernel and user space process

Common UNIX API defines a syscall as a way to issue commands from a user space process to the kernel. The most common implementation is either by using an interrupt or specialized instruction (think of `SYSENTER/SYSCALL` instructions for ia32). Syscalls are defined by a number. For example in OS, the syscall number 85 is the `MAN.SWAPON.2` syscall and the syscall number 132 is `MAN.MKFIFO.2`. Some syscalls need parameters, which are passed from the user-space to the kernel-space in various ways (implementation dependant). Syscalls are synchronous.

Another possible way to communicate is by using a trap. Traps occur asynchronously after some event occurs (division by zero, page fault etc.). A trap can be transparent for a process (page fault) or can result in a reaction like sending a signal (division by zero).

Communication between processes

There are other APIs (System V IPC, shared memory etc.) but the single most important API is signal. Signals are sent by processes or by the kernel and received by processes. Some signals can be ignored or handled by a user supplied routine, some result in a predefined action that cannot be altered or ignored.

Process management

Kernel instances are processed first in the system (so called init). Every running process can create its identical copy using the `MAN.FORK.2` syscall. Some slightly modified versions of this syscall were introduced but the basic semantic is the same. Every running process can morph into some other process using the `MAN.EXEC.3` syscall. Some modifications of this syscall were introduced but all serve the same basic purpose. Processes end their lives by calling the `MAN.EXIT.2` syscall. Every process is identified by a unique number called PID. Every process has a defined parent (identified by its PID).

Thread management

Traditional UNIX does not define any API nor implementation for threading, while POSIX defines its threading API but the implementation is undefined. Traditionally there were two ways of implementing threads. Handling them as separate processes (1:1 threading) or envelope the whole thread group in one process and managing the threading in userspace (1:N threading). Comparing main features of each approach:

1:1 threading

- – heavyweight threads
- – the scheduling cannot be altered by the user (slightly mitigated by the POSIX API)
- – no syscall wrapping necessary
- – can utilize multiple CPUs

1:N threading

- – lightweight threads
- – scheduling can be easily altered by the user
- – syscalls must be wrapped
- – cannot utilize more than one CPU

58.2.3 What is OS?

The OS project is one of the oldest open source operating systems currently available for daily use. It is a direct descendant of the genuine UNIX so it could be claimed that it is a true UNIX although licensing issues do not permit that. The start of the project dates back to the early 1990's when a crew of fellow BSD users patched the 386BSD operating system. Based on this patchkit a new operating system arose named OS for its liberal license. Another group created the NetBSD operating system with different goals in mind. We will focus on OS.

OS is a modern UNIX-based operating system with all the features of UNIX. Preemptive multitasking, multiuser facilities, TCP/IP networking, memory protection, symmetric multiprocessing support, virtual memory with merged VM and buffer cache, they are all there. One of the interesting and extremely useful features is the ability to emulate other UNIX-like operating systems. As of December 2006 and 7-CURRENT development, the following emulation functionalities are supported:

- OS/i386 emulation on OS/amd64
- OS/i386 emulation on OS/ia64
- LINUX-emulation of LINUX operating system on OS
- NDIS-emulation of Windows networking drivers interface
- NetBSD-emulation of NetBSD operating system
- PECoff-support for PECoff OS executables
- SVR4-emulation of System V revision 4 UNIX

Actively developed emulations are the LINUX layer and various OS-on-OS layers. Others are not supposed to work properly nor be usable these days.

OS development happens in a central CVS repository where only a selected team of so called committers can write. This repository possesses several branches; the most interesting are the HEAD branch, in OS nomenclature called -CURRENT, and RELENG_X branches, where X stands for a number indicating a major version of OS. As of December 2006, there are development branches for 6.X development (RELENG_6) and for the 5.X development (RELENG_5). Other branches are closed and not actively maintained or only fed with security patches by the Security Officer of the OS project.

Historically the active development was done in the HEAD branch so it was considered extremely unstable and supposed to happen to break at any time. This is not true any more as the Perforce (commercial version control system) repository was introduced so that active development happens there. There are many branches in Perforce where development of certain parts of the system happens and these branches are from time to time merged back to the main CVS repository thus effectively putting the given feature to the OS operating system. The same happened with the `rdivacky_linuxulator` branch where development of this thesis code was going on.

More info about the OS operating system can be found at [2].

Technical details

OS is traditional flavor of UNIX in the sense of dividing the run of processes into two halves: kernel space and user space run. There are two types of process entry to the kernel: a syscall and a trap. There is only one way to return. In the subsequent sections we will describe the three gates to/from the kernel. The whole description applies to the i386 architecture as the Linuxulator only exists there but the concept is similar on other architectures. The information was taken from [1] and the source code.

System entries

OS has an abstraction called an execution class loader, which is a wedge into the `MAN.EXECVE.2` syscall. This employs a structure `sysentvec`, which describes an executable ABI. It contains things like `errno` translation table, signal translation table, various functions to serve syscall needs (stack fixup, coredumping, etc.). Every ABI the OS kernel wants to support must define this structure, as it is used later in the syscall processing code and at some other places. System entries are handled by trap handlers, where we can access both the kernel-space and the user-space at once.

Syscalls

Syscalls on OS are issued by executing interrupt `0x80` with register `%eax` set to a desired syscall number with arguments passed on the stack.

When a process issues an interrupt `0x80`, the `int0x80` syscall trap handler is issued (defined in `sys/i386/i386/exception.s`), which prepares arguments (i.e. copies them on to the stack) for a call to a C function `MAN.SYSCALL.2` (defined in `sys/i386/i386/trap.c`), which processes the passed in `trapframe`. The processing consists of preparing the syscall (depending on the `sysvec` entry), determining if the syscall is 32-bit or 64-bit one (changes size of the parameters), then the parameters are copied, including the syscall. Next, the actual syscall function is executed with processing of the return code (special cases for `ERESTART` and `EJUSTRETURN` errors). Finally an `userret()` is scheduled, switching the process back to the userspace. The parameters to the actual syscall handler are passed in the form of `struct thread *td, struct syscall args *arguments` where the second parameter is a pointer to the copied in structure of parameters.

Traps

Handling of traps in OS is similar to the handling of syscalls. Whenever a trap occurs, an assembler handler is called. It is chosen between `alltraps`, `alltraps` with regs pushed or `calltrap` depending on the type of the trap. This handler prepares arguments for a call to a C function `trap()` (defined in `sys/i386/i386/trap.c`), which then processes the occurred trap. After the processing it might send a signal to the process and/or exit to userland using `userret()`.

Exits

Exits from kernel to userspace happen using the assembler routine `doreti` regardless of whether the kernel was entered via a trap or via a syscall. This restores the program status from the stack and returns to the userspace.

UNIX primitives

OS operating system adheres to the traditional UNIX scheme, where every process has a unique identification number, the so called PID (Process ID). PID numbers are allocated either linearly or randomly ranging from 0 to `PID_MAX`.

The allocation of PID numbers is done using linear searching of PID space. Every thread in a process receives the same PID number as result of the `MAN.GETPID.2` call.

There are currently two ways to implement threading in OS. The first way is M:N threading followed by the 1:1 threading model. The default library used is M:N threading (`libpthread`) and you can switch at runtime to 1:1 threading (`libthr`). The plan is to switch to 1:1 library by default soon. Although those two libraries use the same kernel primitives, they are accessed through different API(es). The M:N library uses the `kse_*` family of syscalls while the 1:1 library uses the `thr_*` family of syscalls. Because of this, there is no general concept of thread ID shared between kernel and userspace. Of course, both threading libraries implement the `pthread` thread ID API. Every kernel thread (as described by `struct thread`) has `td_tid` identifier but this is not directly accessible from userland and solely serves the kernel's needs. It is also used for 1:1 threading library as `pthread`'s thread ID but handling of this is internal to the library and cannot be relied on.

As stated previously there are two implementations of threading in OS. The M:N library divides the work between kernel space and userspace. Thread is an entity that gets scheduled in the kernel but it can represent various number of userspace threads. M userspace threads get mapped to N kernel threads thus saving resources while keeping the ability to exploit multiprocessor parallelism. Further information about the implementation can be obtained from the man page or [1]. The 1:1 library directly maps a userland thread to a kernel thread thus greatly simplifying the scheme. None of these designs implement a fairness mechanism (such a mechanism was implemented but it was removed recently because it caused serious slowdown and made the code more difficult to deal with).

58.2.4 What is LINUX

LINUX is a UNIX-like kernel originally developed by Linus Torvalds, and now being contributed to by a massive crowd of programmers all around the world. From its mere beginnings to today's, with wide support from companies such as IBM or Google, LINUX is being associated with its fast development pace, full hardware support and benevolent dictator model of organization.

LINUX development started in 1991 as a hobbyist project at University of Helsinki in Finland. Since then it has obtained all the features of a modern UNIX-like OS: multiprocessing, multiuser support, virtual memory, networking, basically everything is there. There are also highly advanced features like virtualization etc.

As of 2006 LINUX seems to be the most widely used open source operating system with support from independent software vendors like Oracle, RealNetworks, Adobe, etc. Most of the commercial software distributed for LINUX can only be obtained in a binary form so recompilation for other operating systems is impossible.

Most of the LINUX development happens in a Git version control system. Git is a distributed system so there is no central source of the LINUX code, but some branches are considered prominent and official. The version number scheme implemented by LINUX consists of four numbers A.B.C.D. Currently development happens in 2.6.C.D, where C represents major version, where new features are added or changed while D is a minor version for bugfixes only.

More information can be obtained from [4].

Technical details

LINUX follows the traditional UNIX scheme of dividing the run of a process in two halves: the kernel and user space. The kernel can be entered in two ways: via a trap or via a syscall. The return is handled only in one way. The further description applies to LINUX 2.6 on the I386 architecture. This information was taken from [3].

Syscalls

Syscalls in LINUX are performed (in userspace) using `syscallX` macros where X substitutes a number representing the number of parameters of the given syscall. This macro translates to a code that loads `%eax` register with a number of the syscall and executes interrupt `0x80`. After this syscall return is called, which translates negative return values

to positive `errno` values and sets `res` to `-1` in case of an error. Whenever the interrupt `0x80` is called the process enters the kernel in system call trap handler. This routine saves all registers on the stack and calls the selected syscall entry. Note that the LINUX calling convention expects parameters to the syscall to be passed via registers as shown here:

1. parameter -> `%ebx`
2. parameter -> `%ecx`
3. parameter -> `%edx`
4. parameter -> `%esi`
5. parameter -> `%edi`
6. parameter -> `%ebp`

There are some exceptions to this, where LINUX uses different calling convention (most notably the `clone` syscall).

Traps

The trap handlers are introduced in `arch/i386/kernel/traps.c` and most of these handlers live in `arch/i386/kernel/entry.S`, where handling of the traps happens.

Exits

Return from the syscall is managed by syscall `MAN.EXIT.3`, which checks for the process having unfinished work, then checks whether we used user-supplied selectors. If this happens stack fixing is applied and finally the registers are restored from the stack and the process returns to the userspace.

UNIX primitives

In the 2.6 version, the LINUX operating system redefined some of the traditional UNIX primitives, notably PID, TID and thread. PID is defined not to be unique for every process, so for some processes (threads) `MAN.GETPPID.2` returns the same value. Unique identification of process is provided by TID. This is because NPTL (New POSIX Thread Library) defines threads to be normal processes (so called 1:1 threading). Spawning a new process in LINUX 2.6 happens using the `clone` syscall (fork variants are reimplemented using it). This clone syscall defines a set of flags that affect behaviour of the cloning process regarding thread implementation. The semantic is a bit fuzzy as there is no single flag telling the syscall to create a thread.

Implemented clone flags are:

- `CLONE_VM` - processes share their memory space
- `CLONE_FS` - share umask, cwd and namespace
- `CLONE_FILES` - share open files
- `CLONE_SIGHAND` - share signal handlers and blocked signals
- `CLONE_PARENT` - share parent
- `CLONE_THREAD` - be thread (further explanation below)
- `CLONE_NEWNS` - new namespace
- `CLONE_SYSVSEM` - share SysV undo structures
- `CLONE_SETTLS` - setup TLS at supplied address

- `CLONE_PARENT_SETTID` - set TID in the parent
- `CLONE_CHILD_CLEARTID` - clear TID in the child
- `CLONE_CHILD_SETTID` - set TID in the child

`CLONE_PARENT` sets the real parent to the parent of the caller. This is useful for threads because if thread A creates thread B we want thread B to be parented to the parent of the whole thread group. `CLONE_THREAD` does exactly the same thing as `CLONE_PARENT`, `CLONE_VM` and `CLONE_SIGHAND`, rewrites PID to be the same as PID of the caller, sets exit signal to be none and enters the thread group. `CLONE_SETTLS` sets up GDT entries for TLS handling. The `CLONE_*_*TID` set of flags sets/clears user supplied address to TID or 0.

As you can see the `CLONE_THREAD` does most of the work and does not seem to fit the scheme very well. The original intention is unclear (even for authors, according to comments in the code) but I think originally there was one threading flag, which was then parcelled among many other flags but this separation was never fully finished. It is also unclear what this partition is good for as glibc does not use that so only hand-written use of the clone permits a programmer to access this features.

For non-threaded programs the PID and TID are the same. For threaded programs the first thread PID and TID are the same and every created thread shares the same PID and gets assigned a unique TID (because `CLONE_THREAD` is passed in) also parent is shared for all processes forming this threaded program.

The code that implements `MAN.PTHREAD.CREATE.3` in NPTL defines the clone flags like this:

```
int clone_flags = (CLONE_VM | CLONE_FS | CLONE_FILES | CLONE_SIGNAL
| CLONE_SETTLS | CLONE_PARENT_SETTID
| CLONE_CHILD_CLEARTID | CLONE_SYSVSEM
#ifdef __ASSUME_NO_CLONE_DETACHED == 0
| CLONE_DETACHED
#endif
| 0);
```

The `CLONE_SIGNAL` is defined like

```
#define CLONE_SIGNAL (CLONE_SIGHAND | CLONE_THREAD)
```

the last 0 means no signal is sent when any of the threads exits.

58.2.5 What is emulation

According to a dictionary definition, emulation is the ability of a program or device to imitate another program or device. This is achieved by providing the same reaction to a given stimulus as the emulated object. In practice, the software world mostly sees three types of emulation - a program used to emulate a machine (QEMU, various game console emulators etc.), software emulation of a hardware facility (OpenGL emulators, floating point units emulation etc.) and operating system emulation (either in kernel of the operating system or as a userspace program).

Emulation is usually used in a place, where using the original component is not feasible nor possible at all. For example someone might want to use a program developed for a different operating system than they use. Then emulation comes in handy. Sometimes there is no other way but to use emulation - e.g. when the hardware device you try to use does not exist (yet/anymore) then there is no other way but emulation. This happens often when porting an operating system to a new (non-existent) platform. Sometimes it is just cheaper to emulate.

Looking from an implementation point of view, there are two main approaches to the implementation of emulation. You can either emulate the whole thing - accepting possible inputs of the original object, maintaining inner state and emitting correct output based on the state and/or input. This kind of emulation does not require any special

conditions and basically can be implemented anywhere for any device/program. The drawback is that implementing such emulation is quite difficult, time-consuming and error-prone. In some cases we can use a simpler approach. Imagine you want to emulate a printer that prints from left to right on a printer that prints from right to left. It is obvious that there is no need for a complex emulation layer but simply reversing of the printed text is sufficient. Sometimes the emulating environment is very similar to the emulated one so just a thin layer of some translation is necessary to provide fully working emulation! As you can see this is much less demanding to implement, so less time-consuming and error-prone than the previous approach. But the necessary condition is that the two environments must be similar enough. The third approach combines the two previous. Most of the time the objects do not provide the same capabilities so in a case of emulating the more powerful one on the less powerful we have to emulate the missing features with full emulation described above.

This master thesis deals with emulation of UNIX on UNIX, which is exactly the case, where only a thin layer of translation is sufficient to provide full emulation. The UNIX API consists of a set of syscalls, which are usually self contained and do not affect some global kernel state.

There are a few syscalls that affect inner state but this can be dealt with by providing some structures that maintain the extra state.

No emulation is perfect and emulations tend to lack some parts but this usually does not cause any serious drawbacks. Imagine a game console emulator that emulates everything but music output. No doubt that the games are playable and one can use the emulator. It might not be that comfortable as the original game console but its an acceptable compromise between price and comfort.

The same goes with the UNIX API. Most programs can live with a very limited set of syscalls working. Those syscalls tend to be the oldest ones (MAN.READ.2/MAN.WRITE.2, MAN.FORK.2 family, MAN.SIGNAL.3 handling, MAN.EXIT.3, MAN.SOCKET.2 API) hence it is easy to emulate because their semantics is shared among all UNIXes, which exist today.

58.3 Emulation

58.3.1 How emulation works in OS

As stated earlier, OS supports running binaries from several other UNIXes. This works because OS has an abstraction called the execution class loader. This wedges into the MAN.EXECVE.2 syscall, so when MAN.EXECVE.2 is about to execute a binary it examines its type.

There are basically two types of binaries in OS. Shell-like text scripts which are identified by `#!` as their first two characters and normal (typically ELF) binaries, which are a representation of a compiled executable object. The vast majority (one could say all of them) of binaries in OS are from type ELF. ELF files contain a header, which specifies the OS ABI for this ELF file. By reading this information, the operating system can accurately determine what type of binary the given file is.

Every OS ABI must be registered in the OS kernel. This applies to the OS native OS ABI, as well. So when MAN.EXECVE.2 executes a binary it iterates through the list of registered APIs and when it finds the right one it starts to use the information contained in the OS ABI description (its syscall table, `errno` translation table, etc.). So every time the process calls a syscall, it uses its own set of syscalls instead of some global one. This effectively provides a very elegant and easy way of supporting execution of various binary formats.

The nature of emulation of different OSes (and also some other subsystems) led developers to invite a handler event mechanism. There are various places in the kernel, where a list of event handlers are called. Every subsystem can register an event handler and they are called accordingly. For example, when a process exits there is a handler called that possibly cleans up whatever the subsystem needs to be cleaned.

Those simple facilities provide basically everything that is needed for the emulation infrastructure and in fact these are basically the only things necessary to implement the LINUX emulation layer.

58.3.2 Common primitives in the OS kernel

Emulation layers need some support from the operating system. I am going to describe some of the supported primitives in the OS operating system.

Locking primitives

Contributed by: A.ATTILIO.EMAIL

The OS synchronization primitive set is based on the idea to supply a rather huge number of different primitives in a way that the better one can be used for every particular, appropriate situation.

To a high level point of view you can consider three kinds of synchronization primitives in the OS kernel:

- atomic operations and memory barriers
- locks
- scheduling barriers

Below there are descriptions for the 3 families. For every lock, you should really check the linked manpage (where possible) for more detailed explanations.

Atomic operations and memory barriers

Atomic operations are implemented through a set of functions performing simple arithmetics on memory operands in an atomic way with respect to external events (interrupts, preemption, etc.). Atomic operations can guarantee atomicity just on small data types (in the magnitude order of the `.long` architecture C data type), so should be rarely used directly in the end-level code, if not only for very simple operations (like flag setting in a bitmap, for example). In fact, it is rather simple and common to write down a wrong semantic based on just atomic operations (usually referred as lock-less). The OS kernel offers a way to perform atomic operations in conjunction with a memory barrier. The memory barriers will guarantee that an atomic operation will happen following some specified ordering with respect to other memory accesses. For example, if we need that an atomic operation happen just after all other pending writes (in terms of instructions reordering buffers activities) are completed, we need to explicitly use a memory barrier in conjunction to this atomic operation. So it is simple to understand why memory barriers play a key role for higher-level locks building (just as refcounts, mutexes, etc.). For a detailed explanatory on atomic operations, please refer to `MAN.ATOMIC.9`. It is far, however, noting that atomic operations (and memory barriers as well) should ideally only be used for building front-ending locks (as mutexes).

Refcounts

Refcounts are interfaces for handling reference counters. They are implemented through atomic operations and are intended to be used just for cases, where the reference counter is the only one thing to be protected, so even something like a spin-mutex is deprecated. Using the refcount interface for structures, where a mutex is already used is often wrong since we should probably close the reference counter in some already protected paths. A manpage discussing refcount does not exist currently, just check `sys/refcount.h` for an overview of the existing API.

Locks

OS kernel has huge classes of locks. Every lock is defined by some peculiar properties, but probably the most important is the event linked to contesting holders (or in other terms, the behaviour of threads unable to acquire the lock). OS's locking scheme presents three different behaviours for contenders:

1. spinning

2. blocking
3. sleeping

Note

numbers are not casual

Spinning locks

Spin locks let waiters to spin until they cannot acquire the lock. An important matter do deal with is when a thread contests on a spin lock if it is not descheduled. Since the OS kernel is preemptive, this exposes spin lock at the risk of deadlocks that can be solved just disabling interrupts while they are acquired. For this and other reasons (like lack of priority propagation support, poorness in load balancing schemes between CPUs, etc.), spin locks are intended to protect very small paths of code, or ideally not to be used at all if not explicitly requested (explained later).

Blocking

Block locks let waiters to be descheduled and blocked until the lock owner does not drop it and wakes up one or more contenders. In order to avoid starvation issues, blocking locks do priority propagation from the waiters to the owner. Block locks must be implemented through the turnstile interface and are intended to be the most used kind of locks in the kernel, if no particular conditions are met.

Sleeping

Sleep locks let waiters to be descheduled and fall asleep until the lock holder does not drop it and wakes up one or more waiters. Since sleep locks are intended to protect large paths of code and to cater asynchronous events, they do not do any form of priority propagation. They must be implemented through the MAN.SLEEPQUEUE.9 interface.

The order used to acquire locks is very important, not only for the possibility to deadlock due at lock order reversals, but even because lock acquisition should follow specific rules linked to locks natures. If you give a look at the table above, the practical rule is that if a thread holds a lock of level *n* (where the level is the number listed close to the kind of lock) it is not allowed to acquire a lock of superior levels, since this would break the specified semantic for a path. For example, if a thread holds a block lock (level 2), it is allowed to acquire a spin lock (level 1) but not a sleep lock (level 3), since block locks are intended to protect smaller paths than sleep lock (these rules are not about atomic operations or scheduling barriers, however).

This is a list of lock with their respective behaviours:

- spin mutex - spinning - MAN.MUTEX.9
- sleep mutex - blocking - MAN.MUTEX.9
- pool mutex - blocking - MAN.MTX.POOL.9
- sleep family - sleeping - MAN.SLEEP.9 pause tsleep msleep spin msleep rw msleep sx
- condvar - sleeping - MAN.CONDVAR.9
- rwlock - blocking - MAN.RWLOCK.9
- sxlock - sleeping - MAN.SX.9
- lockmgr - sleeping - MAN.LOCKMGR.9
- semaphores - sleeping - MAN.SEMA.9

Among these locks only mutexes, sxlocks, rwlocks and lockmgrs are intended to handle recursion, but currently recursion is only supported by mutexes and lockmgrs.

Scheduling barriers

Scheduling barriers are intended to be used in order to drive scheduling of threading. They consist mainly of three different stubs:

- critical sections (and preemption)
- sched_bind
- sched_pin

Generally, these should be used only in a particular context and even if they can often replace locks, they should be avoided because they do not let the diagnose of simple eventual problems with locking debugging tools (as MAN.WITNESS.4).

Critical sections

The OS kernel has been made preemptive basically to deal with interrupt threads. In fact, in order to avoid high interrupt latency, time-sharing priority threads can be preempted by interrupt threads (in this way, they do not need to wait to be scheduled as the normal path previews). Preemption, however, introduces new racing points that need to be handled, as well. Often, in order to deal with preemption, the simplest thing to do is to completely disable it. A critical section defines a piece of code (borderlined by the pair of functions MAN.CRITICAL.ENTER.9 and MAN.CRITICAL.EXIT.9, where preemption is guaranteed to not happen (until the protected code is fully executed). This can often replace a lock effectively but should be used carefully in order to not lose the whole advantage that preemption brings.

sched_pin/sched_unpin

Another way to deal with preemption is the sched_pin() interface. If a piece of code is closed in the sched_pin() and sched_unpin() pair of functions it is guaranteed that the respective thread, even if it can be preempted, it will always be executed on the same CPU. Pinning is very effective in the particular case when we have to access at per-cpu datas and we assume other threads will not change those data. The latter condition will determine a critical section as a too strong condition for our code.

sched_bind/sched_unbind

sched_bind is an API used in order to bind a thread to a particular CPU for all the time it executes the code, until a sched_unbind function call does not unbind it. This feature has a key role in situations where you cannot trust the current state of CPUs (for example, at very early stages of boot), as you want to avoid your thread to migrate on inactive CPUs. Since sched_bind and sched_unbind manipulate internal scheduler structures, they need to be enclosed in sched_lock acquisition/releasing when used.

Proc structure

Various emulation layers sometimes require some additional per-process data. It can manage separate structures (a list, a tree etc.) containing these data for every process but this tends to be slow and memory consuming. To solve this problem the OS proc structure contains p_emuldata, which is a void pointer to some emulation layer specific data. This proc entry is protected by the proc mutex.

The OS proc structure contains a p_sysent entry that identifies, which ABI this process is running. In fact, it is a pointer to the sysentvec described above. So by comparing this pointer to the address where the sysentvec structure for the given ABI is stored we can effectively determine whether the process belongs to our emulation layer. The code typically looks like:


```
if (__predict_true(p->p_sysent != &elf_LINUX_sysvec))
    return;
```

As you can see, we effectively use the `__predict_true` modifier to collapse the most common case (OS process) to a simple return operation thus preserving high performance. This code should be turned into a macro because currently it is not very flexible, i.e. we do not support LINUX64 emulation nor A.OUT LINUX processes on i386.

VFS

The OS VFS subsystem is very complex but the LINUX emulation layer uses just a small subset via a well defined API. It can either operate on vnodes or file handlers. Vnode represents a virtual vnode, i.e. representation of a node in VFS. Another representation is a file handler, which represents an opened file from the perspective of a process. A file handler can represent a socket or an ordinary file. A file handler contains a pointer to its vnode. More than one file handler can point to the same vnode.

namei

The MAN.NAMEI.9 routine is a central entry point to pathname lookup and translation. It traverses the path point by point from the starting point to the end point using lookup function, which is internal to VFS. The MAN.NAMEI.9 syscall can cope with symlinks, absolute and relative paths. When a path is looked up using MAN.NAMEI.9 it is inputted to the name cache. This behaviour can be suppressed. This routine is used all over the kernel and its performance is very critical.

vn_fullpath

The MAN.VN.FULLPATH.9 function takes the best effort to traverse VFS name cache and returns a path for a given (locked) vnode. This process is unreliable but works just fine for the most common cases. The unreliability is because it relies on VFS cache (it does not traverse the on medium structures), it does not work with hardlinks, etc. This routine is used in several places in the Linuxulator.

Vnode operations

- `fgetvp` - given a thread and a file descriptor number it returns the associated vnode
- MAN.VN.LOCK.9 - locks a vnode
- `vn_unlock` - unlocks a vnode
- MAN.VOP.READDIR.9 - reads a directory referenced by a vnode
- MAN.VOP.GETATTR.9 - gets attributes of a file or a directory referenced by a vnode
- MAN.VOP.LOOKUP.9 - looks up a path to a given directory
- MAN.VOP.OPEN.9 - opens a file referenced by a vnode
- MAN.VOP.CLOSE.9 - closes a file referenced by a vnode
- MAN.VPUT.9 - decrements the use count for a vnode and unlocks it
- MAN.VRELE.9 - decrements the use count for a vnode
- MAN.VREF.9 - increments the use count for a vnode

File handler operations

- `fget` - given a thread and a file descriptor number it returns associated file handler and references it
- `fdrop` - drops a reference to a file handler
- `fhold` - references a file handler

58.4 LINUX emulation layer -MD part

This section deals with implementation of LINUX emulation layer in OS operating system. It first describes the machine dependent part talking about how and where interaction between userland and kernel is implemented. It talks about syscalls, signals, ptrace, traps, stack fixup. This part discusses i386 but it is written generally so other architectures should not differ very much. The next part is the machine independent part of the Linuxulator. This section only covers i386 and ELF handling. A.OUT is obsolete and untested.

58.4.1 Syscall handling

Syscall handling is mostly written in `linux_sysvec.c`, which covers most of the routines pointed out in the `sysentvec` structure. When a LINUX process running on OS issues a syscall, the general syscall routine calls `linux_prepsyscall` routine for the LINUX ABI.

LINUX prepsyscall

LINUX passes arguments to syscalls via registers (that is why it is limited to 6 parameters on i386) while OS uses the stack. The LINUX `prepsyscall` routine must copy parameters from registers to the stack. The order of the registers is: `%ebx`, `%ecx`, `%edx`, `%esi`, `%edi`, `%ebp`. The catch is that this is true for only *most* of the syscalls. Some (most notably `clone`) uses a different order but it is luckily easy to fix by inserting a dummy parameter in the `linux_clone` prototype.

Syscall writing

Every syscall implemented in the Linuxulator must have its prototype with various flags in `syscalls.master`. The form of the file is:

```
...
AUE_FORK STD          { int linux_fork(void); }
...
AUE_CLOSE NOPROTO    { int close(int fd); }
...
```

The first column represents the syscall number. The second column is for auditing support. The third column represents the syscall type. It is either `STD`, `OBSOL`, `NOPROTO` and `UNIMPL`. `STD` is a standard syscall with full prototype and implementation. `OBSOL` is obsolete and defines just the prototype. `NOPROTO` means that the syscall is implemented elsewhere so do not prepend ABI prefix, etc. `UNIMPL` means that the syscall will be substituted with the `nosys` syscall (a syscall just printing out a message about the syscall not being implemented and returning `ENOSYS`).

From `syscalls.master` a script generates three files: `linux_syscall.h`, `linux_proto.h` and `linux_sysent.c`. The `linux_syscall.h` contains definitions of syscall names and their numerical value, e.g.:

```
...
#define LINUX_SYS_linux_fork 2
...
#define LINUX_SYS_close 6
...
```

The `linux_proto.h` contains structure definitions of arguments to every syscall, e.g.:

```
struct linux_fork_args {
    register_t dummy;
};
```

And finally, `linux_sysent.c` contains structure describing the system entry table, used to actually dispatch a syscall, e.g.:

```
{ 0, (sy_call_t *)linux_fork, AUE_FORK, NULL, 0, 0 }, /* 2 = linux_fork */
{ AS(close_args), (sy_call_t *)close, AUE_CLOSE, NULL, 0, 0 }, /* 6 = close */
```

As you can see `linux_fork` is implemented in `Linuxulator` itself so the definition is of `STD` type and has no argument, which is exhibited by the dummy argument structure. On the other hand `close` is just an alias for real OS `MAN.CLOSE.2` so it has no linux arguments structure associated and in the system entry table it is not prefixed with `linux` as it calls the real `MAN.CLOSE.2` in the kernel.

Dummy syscalls

The `LINUX` emulation layer is not complete, as some syscalls are not implemented properly and some are not implemented at all. The emulation layer employs a facility to mark unimplemented syscalls with the `DUMMY` macro. These dummy definitions reside in `linux_dummy.c` in a form of `DUMMY(syscall);`, which is then translated to various syscall auxiliary files and the implementation consists of printing a message saying that this syscall is not implemented. The `UNIMPL` prototype is not used because we want to be able to identify the name of the syscall that was called in order to know what syscalls are more important to implement.

58.4.2 Signal handling

Signal handling is done generally in the OS kernel for all binary compatibilities with a call to a compat-dependent layer. `LINUX` compatibility layer defines `linux_sendsig` routine for this purpose.

LINUX sendsig

This routine first checks whether the signal has been installed with a `SA_SIGINFO` in which case it calls `linux_rt_sendsig` routine instead. Furthermore, it allocates (or reuses an already existing) signal handle context, then it builds a list of arguments for the signal handler. It translates the signal number based on the signal translation table, assigns a handler, translates sigset. Then it saves context for the `sigreturn` routine (various registers, translated trap number and signal mask). Finally, it copies out the signal context to the userspace and prepares context for the actual signal handler to run.

linux_rt_sendsig

This routine is similar to `linux_sendsig` just the signal context preparation is different. It adds `siginfo`, `ucontext`, and some `POSIX` parts. It might be worth considering whether those two functions could not be merged with a benefit of less code duplication and possibly even faster execution.

linux_sigreturn

This syscall is used for return from the signal handler. It does some security checks and restores the original process context. It also unmask the signal in process signal mask.

58.4.3 Ptrace

Many UNIX derivates implement the MAN.PTRACE.2 syscall in order to allow various tracking and debugging features. This facility enables the tracing process to obtain various information about the traced process, like register dumps, any memory from the process address space, etc. and also to trace the process like in stepping an instruction or between system entries (syscalls and traps). MAN.PTRACE.2 also lets you set various information in the traced process (registers etc.). MAN.PTRACE.2 is a UNIX-wide standard implemented in most UNIXes around the world.

LINUX emulation in OS implements the MAN.PTRACE.2 facility in `linux_ptrace.c`. The routines for converting registers between LINUX and OS and the actual MAN.PTRACE.2 syscall emulation syscall. The syscall is a long switch block that implements its counterpart in OS for every MAN.PTRACE.2 command. The MAN.PTRACE.2 commands are mostly equal between LINUX and OS so usually just a small modification is needed. For example, `PT_GETREGS` in LINUX operates on direct data while OS uses a pointer to the data so after performing a (native) MAN.PTRACE.2 syscall, a copyout must be done to preserve LINUX semantics.

The MAN.PTRACE.2 implementation in Linuxulator has some known weaknesses. There have been panics seen when using `strace` (which is a MAN.PTRACE.2 consumer) in the Linuxulator environment. Also `PT_SYSCALL` is not implemented.

58.4.4 Traps

Whenever a LINUX process running in the emulation layer traps the trap itself is handled transparently with the only exception of the trap translation. LINUX and OS differs in opinion on what a trap is so this is dealt with here. The code is actually very short:

```
static int
translate_traps(int signal, int trap_code)
{
    if (signal != SIGBUS)
        return signal;

    switch (trap_code) {

        case T_PROTFLT:
        case T_TSSFLT:
        case T_DOUBLEFLT:
        case T_PAGEFLT:
            return SIGSEGV;

        default:
            return signal;
    }
}
```

58.4.5 Stack fixup

The RTLD run-time link-editor expects so called AUX tags on stack during an `execve` so a fixup must be done to ensure this. Of course, every RTLD system is different so the emulation layer must provide its own stack fixup routine

to do this. So does Linuxulator. The `elf_linux_fixup` simply copies out AUX tags to the stack and adjusts the stack of the user space process to point right after those tags. So RTLD works in a smart way.

58.4.6 A.OUT support

The LINUX emulation layer on i386 also supports LINUX A.OUT binaries. Pretty much everything described in the previous sections must be implemented for A.OUT support (beside traps translation and signals sending). The support for A.OUT binaries is no longer maintained, especially the 2.6 emulation does not work with it but this does not cause any problem, as the linux-base in ports probably do not support A.OUT binaries at all. This support will probably be removed in future. Most of the stuff necessary for loading LINUX A.OUT binaries is in `imgact_linux.c` file.

58.5 LINUX emulation layer -MI part

This section talks about machine independent part of the Linuxulator. It covers the emulation infrastructure needed for LINUX 2.6 emulation, the thread local storage (TLS) implementation (on i386) and futexes. Then we talk briefly about some syscalls.

58.5.1 Description of NPTL

One of the major areas of progress in development of LINUX 2.6 was threading. Prior to 2.6, the LINUX threading support was implemented in the `linuxthreads` library. The library was a partial implementation of POSIX threading. The threading was implemented using separate processes for each thread using the `clone` syscall to let them share the address space (and other things). The main weaknesses of this approach was that every thread had a different PID, signal handling was broken (from the `pthreads` perspective), etc. Also the performance was not very good (use of `SIGUSR` signals for threads synchronization, kernel resource consumption, etc.) so to overcome these problems a new threading system was developed and named NPTL.

The NPTL library focused on two things but a third thing came along so it is usually considered a part of NPTL. Those two things were embedding of threads into a process structure and futexes. The additional third thing was TLS, which is not directly required by NPTL but the whole NPTL userland library depends on it. Those improvements yielded in much improved performance and standards conformance. NPTL is a standard threading library in LINUX systems these days.

The OS Linuxulator implementation approaches the NPTL in three main areas. The TLS, futexes and PID mangling, which is meant to simulate the LINUX threads. Further sections describe each of these areas.

58.5.2 LINUX 2.6 emulation infrastructure

These sections deal with the way LINUX threads are managed and how we simulate that in OS.

Runtime determining of 2.6 emulation

The LINUX emulation layer in OS supports runtime setting of the emulated version. This is done via `MAN.SYSCTL.8`, namely `compat.linux.osrelease`, which is set to 2.4.2 by default (as of April 2007) and with all LINUX versions up to 2.6 it just determined what `MAN.UNAME.1` outputs. It is different with 2.6 emulation where setting this `MAN.SYSCTL.8` affects runtime behaviour of the emulation layer. When set to 2.6.x it sets the value of `linux_use_linux26` while setting to something else keeps it unset. This variable (plus per-prison variables of the very same kind) determines whether 2.6 infrastructure (mainly PID mangling) is used in the code or not. The version setting is done system-wide and this affects all LINUX processes. The `MAN.SYSCTL.8` should not be changed when running any LINUX binary as it might harm things.

LINUX processes and thread identifiers

The semantics of LINUX threading are a little confusing and uses entirely different nomenclature to OS. A process in LINUX consists of a `struct task` embedding two identifier fields - PID and TGID. PID is *not* a process ID but it is a thread ID. The TGID identifies a thread group in other words a process. For single-threaded process the PID equals the TGID.

The thread in NPTL is just an ordinary process that happens to have TGID not equal to PID and have a group leader not equal to itself (and shared VM etc. of course). Everything else happens in the same way as to an ordinary process. There is no separation of a shared status to some external structure like in OS. This creates some duplication of information and possible data inconsistency. The LINUX kernel seems to use task -> group information in some places and task information elsewhere and it is really not very consistent and looks error-prone.

Every NPTL thread is created by a call to the `clone` syscall with a specific set of flags (more in the next subsection). The NPTL implements strict 1:1 threading.

In OS we emulate NPTL threads with ordinary OS processes that share VM space, etc. and the PID gymnastic is just mimicked in the emulation specific structure attached to the process. The structure attached to the process looks like:

```
struct linux_emuldata {
    pid_t pid;

    int *child_set_tid; /* in clone(): Child.s TID to set on clone */
    int *child_clear_tid; /* in clone(): Child.s TID to clear on exit */

    struct linux_emuldata_shared *shared;

    int pdeath_signal; /* parent death signal */

    LIST_ENTRY(linux_emuldata) threads; /* list of linux threads */
};
```

The PID is used to identify the OS process that attaches this structure. The `child_set_tid` and `child_clear_tid` are used for TID address copyout when a process exits and is created. The `shared` pointer points to a structure shared among threads. The `pdeath_signal` variable identifies the parent death signal and the `threads` pointer is used to link this structure to the list of threads. The `linux_emuldata_shared` structure looks like:

```
struct linux_emuldata_shared {

    int refs;

    pid_t group_pid;

    LIST_HEAD(, linux_emuldata) threads; /* head of list of linux threads */
};
```

The `refs` is a reference counter being used to determine when we can free the structure to avoid memory leaks. The `group_pid` is to identify PID (= TGID) of the whole process (= thread group). The `threads` pointer is the head of the list of threads in the process.

The `linux_emuldata` structure can be obtained from the process using `em_find`. The prototype of the function is:

```
struct linux_emuldata *em_find(struct proc *, int locked);
```

Here, `proc` is the process we want the `emuldata` structure from and the `locked` parameter determines whether we want to lock or not. The accepted values are `EMUL_DOLOCK` and `EMUL_DOUNLOCK`. More about locking later.

PID mangling

Because of the described different view knowing what a process ID and thread ID is between OS and LINUX we have to translate the view somehow. We do it by PID mangling. This means that we fake what a PID (=TGID) and TID (=PID) is between kernel and userland. The rule of thumb is that in kernel (in Linuxulator) PID = PID and TGID = shared -> group pid and to userland we present PID = shared -> group_pid and TID = proc -> p_pid. The PID member of `linux_emuldata` structure is a OS PID.

The above affects mainly `getpid`, `getppid`, `gettid` syscalls. Where we use PID/TGID respectively. In copyout of TIDs in `child_clear_tid` and `child_set_tid` we copy out OS PID.

Clone syscall

The `clone` syscall is the way threads are created in LINUX. The syscall prototype looks like this:

```
int linux_clone(l_int flags, void *stack, void *parent_tidptr, int dummy,
void * child_tidptr);
```

The `flags` parameter tells the syscall how exactly the processes should be cloned. As described above, LINUX can create processes sharing various things independently, for example two processes can share file descriptors but not VM, etc. Last byte of the `flags` parameter is the exit signal of the newly created process. The `stack` parameter if non-NULL tells, where the thread stack is and if it is NULL we are supposed to copy-on-write the calling process stack (i.e. do what normal `MAN.FORK.2` routine does). The `parent_tidptr` parameter is used as an address for copying out process PID (i.e. thread id) once the process is sufficiently instantiated but is not runnable yet. The `dummy` parameter is here because of the very strange calling convention of this syscall on i386. It uses the registers directly and does not let the compiler do it what results in the need of a dummy syscall. The `child_tidptr` parameter is used as an address for copying out PID once the process has finished forking and when the process exits.

The syscall itself proceeds by setting corresponding flags depending on the flags passed in. For example, `CLONE_VM` maps to `RFMEM` (sharing of VM), etc. The only nit here is `CLONE_FS` and `CLONE_FILES` because OS does not allow setting this separately so we fake it by not setting `RFFDG` (copying of fd table and other fs information) if either of these is defined. This does not cause any problems, because those flags are always set together. After setting the flags the process is forked using the internal `fork1` routine, the process is instrumented not to be put on a run queue, i.e. not to be set runnable. After the forking is done we possibly reparent the newly created process to emulate `CLONE_PARENT` semantics. Next part is creating the emulation data. Threads in LINUX does not signal their parents so we set exit signal to be 0 to disable this. After that setting of `child_set_tid` and `child_clear_tid` is performed enabling the functionality later in the code. At this point we copy out the PID to the address specified by `parent_tidptr`. The setting of process stack is done by simply rewriting thread frame `%esp` register (`%rsp` on amd64). Next part is setting up TLS for the newly created process. After this `MAN.VFORK.2` semantics might be emulated and finally the newly created process is put on a run queue and copying out its PID to the parent process via `clone` return value is done.

The `clone` syscall is able and in fact is used for emulating classic `MAN.FORK.2` and `MAN.VFORK.2` syscalls. Newer glibc in a case of 2.6 kernel uses `clone` to implement `MAN.FORK.2` and `MAN.VFORK.2` syscalls.

Locking

The locking is implemented to be per-subsystem because we do not expect a lot of contention on these. There are two locks: `emul_lock` used to protect manipulating of `linux_emuldata` and `emul_shared_lock` used to manipulate `linux_emuldata_shared`. The `emul_lock` is a nonsleepable blocking mutex while `emul_shared_lock` is a sleepable blocking `sx_lock`. Because of the per-subsystem locking we can coalesce some locks and that is why the `em` find offers the non-locking access.

58.5.3 TLS

This section deals with TLS also known as thread local storage.

Introduction to threading

Threads in computer science are entities within a process that can be scheduled independently from each other. The threads in the process share process wide data (file descriptors, etc.) but also have their own stack for their own data. Sometimes there is a need for process-wide data specific to a given thread. Imagine a name of the thread in execution or something like that. The traditional UNIX threading API, pthreads provides a way to do it via `MAN.PTHREAD.KEY.CREATE.3`, `MAN.PTHREAD.SETSPECIFIC.3` and `MAN.PTHREAD.GETSPECIFIC.3` where a thread can create a key to the thread local data and using `MAN.PTHREAD.GETSPECIFIC.3` or `MAN.PTHREAD.GETSPECIFIC.3` to manipulate those data. You can easily see that this is not the most comfortable way this could be accomplished. So various producers of C/C++ compilers introduced a better way. They defined a new modifier keyword `thread` that specifies that a variable is thread specific. A new method of accessing such variables was developed as well (at least on i386). The pthreads method tends to be implemented in userspace as a trivial lookup table. The performance of such a solution is not very good. So the new method uses (on i386) segment registers to address a segment, where TLS area is stored so the actual accessing of a thread variable is just appending the segment register to the address thus addressing via it. The segment registers are usually `%gs` and `%fs` acting like segment selectors. Every thread has its own area where the thread local data are stored and the segment must be loaded on every context switch. This method is very fast and used almost exclusively in the whole i386 UNIX world. Both OS and LINUX implement this approach and it yields very good results. The only drawback is the need to reload the segment on every context switch which can slowdown context switches. OS tries to avoid this overhead by using only 1 segment descriptor for this while LINUX uses 3. Interesting thing is that almost nothing uses more than 1 descriptor (only Wine seems to use 2) so LINUX pays this unnecessary price for context switches.

Segments on i386

The i386 architecture implements the so called segments. A segment is a description of an area of memory. The base address (bottom) of the memory area, the end of it (ceiling), type, protection, etc. The memory described by a segment can be accessed using segment selector registers (`%cs`, `%ds`, `%ss`, `%es`, `%fs`, `%gs`). For example let us suppose we have a segment which base address is 0x1234 and length and this code:

```
mov %edx, %gs:0x10
```

This will load the content of the `%edx` register into memory location 0x1244. Some segment registers have a special use, for example `%cs` is used for code segment and `%ss` is used for stack segment but `%fs` and `%gs` are generally unused. Segments are either stored in a global GDT table or in a local LDT table. LDT is accessed via an entry in the GDT. The LDT can store more types of segments. LDT can be per process. Both tables define up to 8191 entries.

Implementation on LINUX i386

There are two main ways of setting up TLS in LINUX. It can be set when cloning a process using the `clone` syscall or it can call `set_thread_area`. When a process passes `CLONE_SETTLS` flag to `clone`, the kernel expects the memory pointed to by the `%esi` register a LINUX user space representation of a segment, which gets translated to the machine representation of a segment and loaded into a GDT slot. The GDT slot can be specified with a number or -1 can be used meaning that the system itself should choose the first free slot. In practice, the vast majority of programs use only one TLS entry and does not care about the number of the entry. We exploit this in the emulation and in fact depend on it.

Emulation of LINUX TLS

i386

Loading of TLS for the current thread happens by calling `set_thread_area` while loading TLS for a second process in `clone` is done in the separate block in `clone`. Those two functions are very similar. The only difference being the actual loading of the GDT segment, which happens on the next context switch for the newly created process while `set_thread_area` must load this directly. The code basically does this. It copies the LINUX form segment descriptor from the userland. The code checks for the number of the descriptor but because this differs between OS and LINUX we fake it a little. We only support indexes of 6, 3 and -1. The 6 is genuine LINUX number, 3 is genuine OS one and -1 means autoselection. Then we set the descriptor number to constant 3 and copy out this to the userspace. We rely on the userspace process using the number from the descriptor but this works most of the time (have never seen a case where this did not work) as the userspace process typically passes in 1. Then we convert the descriptor from the LINUX form to a machine dependant form (i.e. operating system independent form) and copy this to the OS defined segment descriptor. Finally we can load it. We assign the descriptor to threads PCB (process control block) and load the `%gs` segment using `load_gs`. This loading must be done in a critical section so that nothing can interrupt us. The `CLONE_SETTLS` case works exactly like this just the loading using `load_gs` is not performed. The segment used for this (segment number 3) is shared for this use between OS processes and LINUX processes so the LINUX emulation layer does not add any overhead over plain OS.

amd64

The amd64 implementation is similar to the i386 one but there was initially no 32bit segment descriptor used for this purpose (hence not even native 32bit TLS users worked) so we had to add such a segment and implement its loading on every context switch (when a flag signaling use of 32bit is set). Apart from this the TLS loading is exactly the same just the segment numbers are different and the descriptor format and the loading differs slightly.

58.5.4 Futexes

Introduction to synchronization

Threads need some kind of synchronization and POSIX provides some of them: mutexes for mutual exclusion, read-write locks for mutual exclusion with biased ratio of reads and writes and condition variables for signaling a status change. It is interesting to note that POSIX threading API lacks support for semaphores. Those synchronization routines implementations are heavily dependant on the type threading support we have. In pure 1:M (userspace) model the implementation can be solely done in userspace and thus be very fast (the condition variables will probably end up being implemented using signals, i.e. not fast) and simple. In 1:1 model, the situation is also quite clear - the threads must be synchronized using kernel facilities (which is very slow because a syscall must be performed). The mixed M:N scenario just combines the first and second approach or rely solely on kernel. Threads synchronization is a vital part of thread-enabled programming and its performance can affect resulting program a lot. Recent benchmarks on OS operating system showed that an improved `sx_lock` implementation yielded 40% speedup in ZFS (a heavy `sx` user), this is in-kernel stuff but it shows clearly how important the performance of synchronization primitives is.

Threaded programs should be written with as little contention on locks as possible. Otherwise, instead of doing useful work the thread just waits on a lock. Because of this, the most well written threaded programs show little locks contention.

Futexes introduction

LINUX implements 1:1 threading, i.e. it has to use in-kernel synchronization primitives. As stated earlier, well written threaded programs have little lock contention. So a typical sequence could be performed as two atomic increase/decrease mutex reference counter, which is very fast, as presented by the following example:


```
pthread_mutex_lock(&mutex);
....
pthread_mutex_unlock(&mutex);
```

1:1 threading forces us to perform two syscalls for those mutex calls, which is very slow.

The solution LINUX 2.6 implements is called futexes. Futexes implement the check for contention in userspace and call kernel primitives only in a case of contention. Thus the typical case takes place without any kernel intervention. This yields reasonably fast and flexible synchronization primitives implementation.

Futex API

The futex syscall looks like this:

```
int futex(void *uaddr, int op, int val, struct timespec *timeout, void *uaddr2, int val3);
```

In this example `uaddr` is an address of the mutex in userspace, `op` is an operation we are about to perform and the other parameters have per-operation meaning.

Futexes implement the following operations:

- FUTEX_WAIT
- FUTEX_WAKE
- FUTEX_FD
- FUTEX_REQUEUE
- FUTEX_CMP_REQUEUE
- FUTEX_WAKE_OP

FUTEX_WAIT

This operation verifies that on address `uaddr` the value `val` is written. If not, `EWOULDBLOCK` is returned, otherwise the thread is queued on the futex and gets suspended. If the argument `timeout` is non-zero it specifies the maximum time for the sleeping, otherwise the sleeping is infinite.

FUTEX_WAKE

This operation takes a futex at `uaddr` and wakes up `val` first futexes queued on this futex.

FUTEX_FD

This operations associates a file descriptor with a given futex.

FUTEX_REQUEUE

This operation takes `val` threads queued on futex at `uaddr`, wakes them up, and takes `val2` next threads and requeues them on futex at `uaddr2`.

FUTEX_CMP_REQUEUE

This operation does the same as `FUTEX_REQUEUE` but it checks that `val3` equals to `val1` first.

FUTEX_WAKE_OP

This operation performs an atomic operation on `val3` (which contains coded some other value) and `uaddr`. Then it wakes up `val1` threads on `futex` at `uaddr` and if the atomic operation returned a positive number it wakes up `val2` threads on `futex` at `uaddr2`.

The operations implemented in `FUTEX_WAKE_OP`:

- `FUTEX_OP_SET`
- `FUTEX_OP_ADD`
- `FUTEX_OP_OR`
- `FUTEX_OP_AND`
- `FUTEX_OP_XOR`

Note

There is no `val2` parameter in the `futex` prototype. The `val2` is taken from the `struct timespec *timeout` parameter for operations `FUTEX_REQUEUE`, `FUTEX_CMP_REQUEUE` and `FUTEX_WAKE_OP`.

Futex emulation in OS

The `futex` emulation in OS is taken from NetBSD and further extended by us. It is placed in `linux_futex.c` and `linux_futex.h` files. The `futex` structure looks like:

```
struct futex {
    void *f_uaddr;
    int f_refcount;

    LIST_ENTRY(futex) f_list;

    TAILQ_HEAD(lf_waiting_proc, waiting_proc) f_waiting_proc;
};
```

And the structure `waiting_proc` is:

```
struct waiting_proc {
    struct thread *wp_t;

    struct futex *wp_new_futex;

    TAILQ_ENTRY(waiting_proc) wp_list;
};
```

futex_get / futex_put

A `futex` is obtained using the `futex_get` function, which searches a linear list of `futexes` and returns the found one or creates a new `futex`. When releasing a `futex` from the use we call the `futex_put` function, which decreases a

reference counter of the futex and if the refcount reaches zero it is released.

futex_sleep

When a futex queues a thread for sleeping it creates a `working_proc` structure and puts this structure to the list inside the futex structure then it just performs a `MAN.TSLEEP.9` to suspend the thread. The sleep can be timed out. After `MAN.TSLEEP.9` returns (the thread was woken up or it timed out) the `working_proc` structure is removed from the list and is destroyed. All this is done in the `futex_sleep` function. If we got woken up from `futex_wake` we have `wp_new_futex` set so we sleep on it. This way the actual requeueing is done in this function.

futex_wake

Waking up a thread sleeping on a futex is performed in the `futex_wake` function. First in this function we mimic the strange LINUX behaviour, where it wakes up `N` threads for all operations, the only exception is that the `REQUEUE` operations are performed on `N+1` threads. But this usually does not make any difference as we are waking up all threads. Next in the function in the loop we wake up `n` threads, after this we check if there is a new futex for requeueing. If so, we requeue up to `n2` threads on the new futex. This cooperates with `futex_sleep`.

futex_wake_op

The `FUTEX_WAKE_OP` operation is quite complicated. First we obtain two futexes at addresses `uaddr` and `uaddr2` then we perform the atomic operation using `val3` and `uaddr2`. Then `val` waiters on the first futex is woken up and if the atomic operation condition holds we wake up `val2` (i.e. `timeout`) waiter on the second futex.

futex atomic operation

The atomic operation takes two parameters `encoded_op` and `uaddr`. The encoded operation encodes the operation itself, comparing value, operation argument, and comparing argument. The pseudocode for the operation is like this one:

```
oldval = *uaddr2
*uaddr2 = oldval OP oparg
```

And this is done atomically. First a copying in of the number at `uaddr` is performed and the operation is done. The code handles page faults and if no page fault occurs `oldval` is compared to `cmparg` argument with `cmp` comparator.

Futex locking

Futex implementation uses two lock lists protecting `sx_lock` and global locks (either Giant or another `sx_lock`). Every operation is performed locked from the start to the very end.

58.5.5 Various syscalls implementation

In this section I am going to describe some smaller syscalls that are worth mentioning because their implementation is not obvious or those syscalls are interesting from other point of view.

*at family of syscalls

During development of LINUX 2.6.16 kernel, the *at syscalls were added. Those syscalls (openat for example) work exactly like their at-less counterparts with the slight exception of the `dirfd` parameter. This parameter changes where the given file, on which the syscall is to be performed, is. When the `filename` parameter is absolute `dirfd` is ignored but when the path to the file is relative, it comes to the play. The `dirfd` parameter is a directory relative to which the relative pathname is checked. The `dirfd` parameter is a file descriptor of some directory or `AT_FDCWD`. So for example the `openat` syscall can be like this:

```
file descriptor 123 = /tmp/foo/, current working directory = /tmp/

openat(123, /tmp/bah\, flags, mode) /* opens /tmp/bah */
openat(123, bah\, flags, mode)      /* opens /tmp/foo/bah */
openat(AT_FDCWD, bah\, flags, mode)  /* opens /tmp/bah */
openat(stdio, bah\, flags, mode)     /* returns error because stdio is not a directory */
```

This infrastructure is necessary to avoid races when opening files outside the working directory. Imagine that a process consists of two threads, thread A and thread B. Thread A issues `open(/tmp/foo/bah\, flags, mode)` and before returning it gets preempted and thread B runs. Thread B does not care about the needs of thread A and renames or removes `/tmp/foo/`. We got a race. To avoid this we can open `/tmp/foo` and use it as `dirfd` for `openat` syscall. This also enables user to implement per-thread working directories.

LINUX family of *at syscalls contains: `linux_openat`, `linux_mkdirat`, `linux_mknodat`, `linux_fchownat`, `linux_futimesat`, `linux_fstatat64`, `linux_unlinkat`, `linux_renameat`, `linux_linkat`, `linux_symlinkat`, `linux_readlinkat`, `linux_fchmodat` and `linux_faccessat`. All these are implemented using the modified `MAN.NAMEI.9` routine and simple wrapping layer.

Implementation

The implementation is done by altering the `MAN.NAMEI.9` routine (described above) to take additional parameter `dirfd` in its `nameidata` structure, which specifies the starting point of the pathname lookup instead of using the current working directory every time. The resolution of `dirfd` from file descriptor number to a `vnode` is done in native *at syscalls. When `dirfd` is `AT_FDCWD` the `dvp` entry in `nameidata` structure is `NULL` but when `dirfd` is a different number we obtain a file for this file descriptor, check whether this file is valid and if there is `vnode` attached to it then we get a `vnode`. Then we check this `vnode` for being a directory. In the actual `MAN.NAMEI.9` routine we simply substitute the `dvp` `vnode` for `dp` variable in the `MAN.NAMEI.9` function, which determines the starting point. The `MAN.NAMEI.9` is not used directly but via a trace of different functions on various levels. For example the `openat` goes like this:

```
openat() --> kern_openat() --> vn_open() -> namei()
```

For this reason `kern_open` and `vn_open` must be altered to incorporate the additional `dirfd` parameter. No compat layer is created for those because there are not many users of this and the users can be easily converted. This general implementation enables OS to implement their own *at syscalls. This is being discussed right now.

ioctl

The `ioctl` interface is quite fragile due to its generality. We have to bear in mind that devices differ between LINUX and OS so some care must be applied to do `ioctl` emulation work right. The `ioctl` handling is implemented in `linux_ioctl.c`, where `linux_ioctl` function is defined. This function simply iterates over sets of `ioctl` handlers to find a handler that implements a given command. The `ioctl` syscall has three parameters, the file descriptor, command and an argument. The command is a 16-bit number, which in theory is divided into high 8 bits determining class of the `ioctl` command and low 8 bits, which are the actual command within the given set. The emulation takes advantage of this division. We implement handlers for each set, like `sound_handler` or `disk_handler`. Each

handler has a maximum command and a minimum command defined, which is used for determining what handler is used. There are slight problems with this approach because LINUX does not use the set division consistently so sometimes ioctls for a different set are inside a set they should not belong to (SCSI generic ioctls inside cdrom set, etc.). OS currently does not implement many LINUX ioctls (compared to NetBSD, for example) but the plan is to port those from NetBSD. The trend is to use LINUX ioctls even in the native OS drivers because of the easy porting of applications.

Debugging

Every syscall should be debuggable. For this purpose we introduce a small infrastructure. We have the `ldebug` facility, which tells whether a given syscall should be debugged (settable via a `sysctl`). For printing we have `LMSG` and `ARGS` macros. Those are used for altering a printable string for uniform debugging messages.

58.6 Conclusion

58.6.1 Results

As of April 2007 the LINUX emulation layer is capable of emulating the LINUX 2.6.16 kernel quite well. The remaining problems concern `futexes`, unfinished `*at` family of syscalls, problematic signals delivery, missing `epoll` and `inotify` and probably some bugs we have not discovered yet. Despite this we are capable of running basically all the LINUX programs included in OS Ports Collection with Fedora Core 4 at 2.6.16 and there are some rudimentary reports of success with Fedora Core 6 at 2.6.16. The Fedora Core 6 `linux_base` was recently committed enabling some further testing of the emulation layer and giving us some more hints where we should put our effort in implementing missing stuff.

We are able to run the most used applications like `www/linux-firefox`, `www/linux-opera`, `net-im/skype` and some games from the Ports Collection. Some of the programs exhibit bad behaviour under 2.6 emulation but this is currently under investigation and hopefully will be fixed soon. The only big application that is known not to work is the LINUX JAVA Development Kit and this is because of the requirement of `epoll` facility which is not directly related to the LINUX kernel 2.6.

We hope to enable 2.6.16 emulation by default some time after OS 7.0 is released at least to expose the 2.6 emulation parts for some wider testing. Once this is done we can switch to Fedora Core 6 `linux_base`, which is the ultimate plan.

58.6.2 Future work

Future work should focus on fixing the remaining issues with `futexes`, implement the rest of the `*at` family of syscalls, fix the signal delivery and possibly implement the `epoll` and `inotify` facilities.

We hope to be able to run the most important programs flawlessly soon, so we will be able to switch to the 2.6 emulation by default and make the Fedora Core 6 the default `linux_base` because our currently used Fedora Core 4 is not supported any more.

The other possible goal is to share our code with NetBSD and DragonflyBSD. NetBSD has some support for 2.6 emulation but its far from finished and not really tested. DragonflyBSD has expressed some interest in porting the 2.6 improvements.

Generally, as LINUX develops we would like to keep up with their development, implementing newly added syscalls. `Splice` comes to mind first. Some already implemented syscalls are also heavily crippled, for example `mremap` and others. Some performance improvements can also be made, finer grained locking and others.

58.6.3 Team

I cooperated on this project with (in alphabetical order):

- A.JHB.EMAIL
- A.KIB.EMAIL
- Emmanuel Dreyfus
- Scot Hetzel
- A.JKIM.EMAIL
- A.NETCHILD.EMAIL
- A.SSOUHLAL.EMAIL
- Li Xiao
- A.DAVIDXU.EMAIL

I would like to thank all those people for their advice, code reviews and general support.

58.7 Literatures

1. Marshall Kirk McKusick - George V. Neville-Neil. Design and Implementation of the OS operating system. Addison-Wesley, 2005.
2. <http://www.FreeBSD.org>
3. <http://tldp.org>
4. <http://www.linux.org>

FreeBSD Quickstart Guide for LINUX Users

Author JohnFerrell

59.1 Introduction

This document highlights some of the technical differences between OS and LINUX so that intermediate to advanced LINUX users can quickly familiarize themselves with the basics of OS.

This document assumes that OS is already installed. Refer to the Installing OS chapter of the OS Handbook for help with the installation process.

59.2 Default Shell

LINUX users are often surprised to find that Bash is not the default shell in OS. In fact, Bash is not included in the default installation. Instead, OS uses MAN.TCSH.1 as the default root shell, and the Bourne shell-compatible MAN.SH.1 as the default user shell. MAN.SH.1 is very similar to Bash but with a much smaller feature-set. Generally shell scripts written for MAN.SH.1 will run in Bash, but the reverse is not always true.

However, Bash and other shells are available for installation using the OS Packages and Ports Collection.

After installing another shell, use MAN.CHSH.1 to change a user's default shell. It is recommended that the root user's default shell remain unchanged since shells which are not included in the base distribution are installed to `/usr/local/bin`. In the event of a problem, the file system where `/usr/local/bin` is located may not be mounted. In this case, root would not have access to its default shell, preventing root from logging in and fixing the problem.

59.3 Packages and Ports: Adding Software in OS

OS provides two methods for installing applications: binary packages and compiled ports. Each method has its own benefits:

- Faster installation as compared to compiling large applications.
- Does not require an understanding of how to compile software.
- No need to install a compiler.
- Ability to customize installation options.

- Custom patches can be applied.

If an application installation does not require any customization, installing the package is sufficient. Compile the port instead whenever an application requires customization of the default options. If needed, a custom package can be compiled from ports using `make package`.

A complete list of all available ports and packages can be found [here](#).

59.3.1 Packages

Packages are pre-compiled applications, the OS equivalents of `.deb` files on Debian/Ubuntu based systems and `.rpm` files on Red Hat/Fedora based systems. Packages are installed using `pkg`. For example, the following command installs Apache 2.4:

```
PROMPT.ROOT pkg install apache24
```

For more information on packages refer to section 5.4 of the OS Handbook: Using `pkgng` for Binary Package Management.

59.3.2 Ports

The OS Ports Collection is a framework of `Makefiles` and patches specifically customized for installing applications from source on OS. When installing a port, the system will fetch the source code, apply any required patches, compile the code, and install the application and any required dependencies.

The Ports Collection, sometimes referred to as the ports tree, can be installed to `/usr/ports` using `MAN.PORTSNAP.8`. Detailed instructions for installing the Ports Collection can be found in section 5.5 of the OS Handbook.

To compile a port, change to the port's directory and start the build process. The following example installs Apache 2.4 from the Ports Collection:

```
PROMPT.ROOT cd /usr/ports/www/apache24
PROMPT.ROOT make install clean
```

A benefit of using ports to install software is the ability to customize the installation options. This example specifies that the `mod_lap` module should also be installed:

```
PROMPT.ROOT cd /usr/ports/www/apache24
PROMPT.ROOT make WITH_LDAP="YES" install clean
```

Refer to Using the Ports Collection for more information.

59.4 System Startup

Many LINUX distributions use the SysV init system, whereas OS uses the traditional BSD-style `MAN.INIT.8`. Under the BSD-style `MAN.INIT.8`, there are no run-levels and `/etc/inittab` does not exist. Instead, startup is controlled by `MAN.RC.8` scripts. At system boot, `/etc/rc` reads `/etc/rc.conf` and `/etc/defaults/rc.conf` to determine which services are to be started. The specified services are then started by running the corresponding service initialization scripts located in `/etc/rc.d/` and `/usr/local/etc/rc.d/`. These scripts are similar to the scripts located in `/etc/init.d/` on LINUX systems.

The scripts found in `/etc/rc.d/` are for applications that are part of the “base” system, such as `MAN.CRON.8`, `MAN.SSHD.8`, and `MAN.SYSLOG.3`. The scripts in `/usr/local/etc/rc.d/` are for user-installed applications such as Apache and Squid.

Since OS is developed as a complete operating system, user-installed applications are not considered to be part of the “base” system. User-installed applications are generally installed using Packages or Ports. In order to keep them separate from the base system, user-installed applications are installed under `/usr/local/`. Therefore, user-installed binaries reside in `/usr/local/bin/`, configuration files are in `/usr/local/etc/`, and so on.

Services are enabled by adding an entry for the service in `/etc/rc.conf`. The system defaults are found in `/etc/defaults/rc.conf` and these default settings are overridden by settings in `/etc/rc.conf`. Refer to [MAN.RC.CONF.5](#) for more information about the available entries. When installing additional applications, review the application’s install message to determine how to enable any associated services.

The following entries in `/etc/rc.conf` enable `MAN.SSHD.8`, enable Apache 2.4, and specify that Apache should be started with SSL.

```
# enable SSHD
sshd_enable="YES"
# enable Apache with SSL
apache24_enable="YES"
apache24_flags="-DSSL"
```

Once a service has been enabled in `/etc/rc.conf`, it can be started without rebooting the system:

```
PROMPT.ROOT service sshd start
PROMPT.ROOT service apache24 start
```

If a service has not been enabled, it can be started from the command line using `onestart`:

```
PROMPT.ROOT service sshd onestart
```

59.5 Network Configuration

Instead of a generic `ethX` identifier that LINUX uses to identify a network interface, OS uses the driver name followed by a number. The following output from [MAN.IFCONFIG.8](#) shows two INTEL Pro 1000 network interfaces (`em0` and `em1`):

```
PROMPT.USER ifconfig
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=b<RXCSUM, TXCSUM, VLAN_MTU>
    inet 10.10.10.100 netmask 0xffffffff broadcast 10.10.10.255
    ether 00:50:56:a7:70:b2
    media: Ethernet autoselect (1000baseTX <full-duplex>)
    status: active
em1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=b<RXCSUM, TXCSUM, VLAN_MTU>
    inet 192.168.10.222 netmask 0xffffffff broadcast 192.168.10.255
    ether 00:50:56:a7:03:2b
    media: Ethernet autoselect (1000baseTX <full-duplex>)
    status: active
```

An IP address can be assigned to an interface using [MAN.IFCONFIG.8](#). To remain persistent across reboots, the IP configuration must be included in `/etc/rc.conf`. The following `/etc/rc.conf` entries specify the hostname, IP address, and default gateway:

```
hostname="server1.example.com"
ifconfig_em0="inet 10.10.10.100 netmask 255.255.255.0"
defaultrouter="10.10.10.1"
```

Use the following entries to instead configure an interface for DHCP:

```
hostname="server1.example.com"
ifconfig_em0="DHCP"
```

59.6 Firewall

OS does not use LINUX IPTABLES for its firewall. Instead, OS offers a choice of three kernel level firewalls:

- PF
- IPFILTER
- IPFW

PF is developed by the OpenBSD project and ported to OS. PF was created as a replacement for IPFILTER and its syntax is similar to that of IPFILTER. PF can be paired with MAN.ALTQ.4 to provide QoS features.

This sample PF entry allows inbound SSH:

```
pass in on $ext_if inet proto tcp from any to ($ext_if) port 22
```

IPFILTER is the firewall application developed by Darren Reed. It is not specific to OS and has been ported to several operating systems including NetBSD, OpenBSD, SunOS, HP/UX, and Solaris.

The IPFILTER syntax to allow inbound SSH is:

```
pass in on $ext_if proto tcp from any to any port = 22
```

IPFW is the firewall developed and maintained by OS. It can be paired with MAN.DUMMYNET.4 to provide traffic shaping capabilities and simulate different types of network connections.

The IPFW syntax to allow inbound SSH would be:

```
ipfw add allow tcp from any to me 22 in via $ext_if
```

59.7 Updating OS

There are two methods for updating a OS system: from source or binary updates.

Updating from source is the most involved update method, but offers the greatest amount of flexibility. The process involves synchronizing a local copy of the OS source code with the OS Subversion servers. Once the local source code is up-to-date, a new version of the kernel and userland can be compiled.

Binary updates are similar to using `yum` or `apt-get` to update a LINUX system. In OS, MAN.FREEBSD-UPDATE.8 can be used fetch new binary updates and install them. These updates can be scheduled using MAN.CRON.8.

Note

When using MAN.CRON.8 to schedule updates, use `freebsd-update cron` in the MAN.CRONTAB.1 to reduce the possibility of a large number of machines all pulling updates at the same time:

```
0 3 * * * root /usr/sbin/freebsd-update cron
```

For more information on source and binary updates, refer to the chapter on updating in the OS Handbook.

59.8 procfs: Gone But Not Forgotten

In some LINUX distributions, one could look at `/proc/sys/net/ipv4/ip_forward` to determine if IP forwarding is enabled. In OS, `MAN.SYSCTL.8` is instead used to view this and other system settings.

For example, use the following to determine if IP forwarding is enabled on a OS system:

```
PROMPT.USER sysctl net.inet.ip.forwarding
net.inet.ip.forwarding: 0
```

Use `-a` to list all the system settings:

```
PROMPT.USER sysctl -a | more
```

If an application requires `procfs`, add the following entry to `/etc/fstab`:

```
proc                /proc              procfs    rw,noauto          0                0
```

Including `noauto` will prevent `/proc` from being automatically mounted at boot.

To mount the file system without rebooting:

```
PROMPT.ROOT mount /proc
```

59.9 Common Commands

Some common command equivalents are as follows:

LINUX command (Red Hat/Debian)	OS equivalent	Purpose
<code>yum install package / apt-get install package</code>	<code>pkg install package</code>	Install package from remote repository
<code>rpm -ivh package / dpkg -i package</code>	<code>pkg add package</code>	Install local package
<code>rpm -qa / dpkg -l</code>	<code>pkg info</code>	List installed packages
<code>lspci</code>	<code>pciconf</code>	List PCI devices
<code>lsmod</code>	<code>kldstat</code>	List loaded kernel modules
<code>modprobe</code>	<code>kldload / kldunload</code>	Load/Unload kernel modules
<code>strace</code>	<code>truss</code>	Trace system calls

59.10 Conclusion

This document has provided an overview of OS. Refer to the OS Handbook for more in-depth coverage of these topics as well as the many topics not covered by this document.

Frequently Asked Questions About The OS Mailing Lists

Author The OS Documentation Project

60.1 Introduction

As is usual with FAQs, this document aims to cover the most frequently asked questions concerning the OS mailing lists (and of course answer them!). Although originally intended to reduce bandwidth and avoid the same old questions being asked over and over again, FAQs have become recognized as valuable information resources.

This document attempts to represent a community consensus, and as such it can never really be *authoritative*. However, if you find technical errors within this document, or have suggestions about items that should be added, please either submit a PR, or email the A.DOC. Thanks.

Q: What is the purpose of the OS mailing lists?

A: The OS mailing lists serve as the primary communication channels for the OS community, covering many different topic areas and communities of interest.

Q: Who is the audience for the OS mailing lists?

A: This depends on charter of each individual list. Some lists are more oriented to developers; some are more oriented towards the OS community as a whole. Please see [this list](#) for the current summary.

Q: Are the OS mailing lists open for anyone to participate?

A: Again, this depends on charter of each individual list. Please read the charter of a mailing list before you post to it, and respect it when you post. This will help everyone to have a better experience with the lists.

If after reading the above lists, you still do not know which mailing list to post a question to, you will probably want to post to `freebsd-questions` (but see below, first).

Also note that the mailing lists have traditionally been open to postings from non-subscribers. This has been a deliberate choice, to help make joining the OS community an easier process, and to encourage open sharing of ideas. However, due to past abuse by some individuals, certain lists now have a policy where postings from non-subscribers must be manually screened to ensure that they are appropriate.

Q: How can I subscribe?

A: You can use [the Mailman web interface](#) to subscribe to any of the public lists.

Q: How can I unsubscribe?

A: You can use the same interface as above; or, you can follow the instructions that are at the bottom of every mailing list message that is sent.

Please do not send unsubscribe messages directly to the public lists themselves. First, this will not accomplish your goal, and second, it will irritate the existing subscribers, and you will probably get flamed. This is a classical mistake when using mailing lists; please try to avoid it.

Q: Are archives available?

A: Yes. Threaded archives are available [here](#).

Q: Are mailing lists available in a digest format?

A: Yes. See [the Mailman web interface](#).

60.2 Mailing List Etiquette

Participation in the mailing lists, like participation in any community, requires a common basis for communication. Please make only appropriate postings, and follow common rules of etiquette.

Q: What should I do before I post?

A: You have already taken the most important step by reading this document. However, if you are new to OS, you may first need to familiarize yourself with the software, and all the social history around it, by reading the numerous books and articles that are available. Items of particular interest include the OS Frequently Asked Questions (FAQ) document, the OS Handbook, and the articles How to get best results from the FreeBSD-questions mailing list, Explaining BSD, and OS First Steps.

It is always considered bad form to ask a question that is already answered in the above documents. This is not because the volunteers who work on this project are particularly mean people, but after a certain number of times answering the same questions over and over again, frustration begins to set in. This is particularly true if there is an existing answer to the question that is already available. Always keep in mind that almost all of the work done on OS is done by volunteers, and that we are only human.

Q: What constitutes an inappropriate posting?

- Postings must be in accordance with the charter of the mailing list.
- Personal attacks are discouraged. As good net-citizens, we should try to hold ourselves to high standards of behavior.
- Spam is not allowed, ever. The mailing lists are actively processed to ban offenders to this rule.

Q: What is considered proper etiquette when posting to the mailing lists?

- Please wrap lines at 75 characters, since not everyone uses fancy GUI mail reading programs.
- Please respect the fact that bandwidth is not infinite. Not everyone reads email through high-speed connections, so if your posting involves something like the content of `config.log` or an extensive stack trace, please consider putting that information up on a website somewhere and just provide a URL to it. Remember, too, that these postings will be archived indefinitely, so huge postings will simply inflate the size of the archives long after their purpose has expired.
- Format your message so that it is legible, and PLEASE DO NOT SHOUT!!!!. Do not underestimate the effect that a poorly formatted mail message has, and not just on the OS mailing lists. Your mail message is all that people see of you, and if it is poorly formatted, badly spelled, full of errors, and/or has lots of exclamation points, it will give people a poor impression of you.
- Please use an appropriate human language for a particular mailing list. Many non-English mailing lists are available.

For the ones that are not, we do appreciate that many people do not speak English as their first language, and we try to make allowances for that. It is considered particularly poor form to criticize non-native speakers for

spelling or grammatical errors. OS has an excellent track record in this regard; please, help us to uphold that tradition.

- Please use a standards-compliant Mail User Agent (MUA). A lot of badly formatted messages come from [bad mailers](#) or [badly configured mailers](#). The following mailers are known to send out badly formatted messages without you finding out about them:

- exmh
- MICROSOFT Exchange
- MICROSOFT OUTLOOK

Try not to use MIME: a lot of people use mailers which do not get on very well with MIME.

- Make sure your time and time zone are set correctly. This may seem a little silly, since your message still gets there, but many of the people on these mailing lists get several hundred messages a day. They frequently sort the incoming messages by subject and by date, and if your message does not come before the first answer, they may assume that they missed it and not bother to look.
- A lot of the information you need to supply is the output of programs, such as MAN.DMESG.8, or console messages, which usually appear in `/var/log/messages`. Do not try to copy this information by typing it in again; not only it is a real pain, but you are bound to make a mistake. To send log file contents, either make a copy of the file and use an editor to trim the information to what is relevant, or cut and paste into your message. For the output of programs like `dmesg`, redirect the output to a file and include that. For example,

```
PROMPT.USER dmesg > /tmp/dmesg.out
```

This redirects the information to the file `/tmp/dmesg.out`.

- When using cut-and-paste, please be aware that some such operations badly mangle their messages. This is of particular concern when posting contents of `Makefiles`, where `tab` is a significant character. This is a very common, and very annoying, problem with submissions to the Problem Reports database. `Makefiles` with tabs changed to either spaces, or the annoying `=3B` escape sequence, create a great deal of aggravation for committers.

Q: What are the special etiquette consideration when replying to an existing posting on the mailing lists?

- Please include relevant text from the original message. Trim it to the minimum, but do not overdo it. It should still be possible for somebody who did not read the original message to understand what you are talking about.

This is especially important for postings of the type “yes, I see this too”, where the initial posting was dozens or hundreds of lines.

- Use some technique to identify which text came from the original message, and which text you add. A common convention is to prepend “> ” to the original message. Leaving white space after the “> ” and leaving empty lines between your text and the original text both make the result more readable.
- Please ensure that the attributions of the text you are quoting is correct. People can become offended if you attribute words to them that they themselves did not write.
- Please do not `top post`. By this, we mean that if you are replying to a message, please put your replies after the text that you copy in your reply.
 - A: Because it reverses the logical flow of conversation.
 - Q: Why is top posting frowned upon?

(Thanks to Randy Bush for the joke.)

60.3 Recurring Topics On The Mailing Lists

Participation in the mailing lists, like participation in any community, requires a common basis for communication. Many of the mailing lists presuppose a knowledge of the Project's history. In particular, there are certain topics that seem to regularly occur to newcomers to the community. It is the responsibility of each poster to ensure that their postings do not fall into one of these categories. By doing so, you will help the mailing lists to stay on-topic, and probably save yourself being flamed in the process.

The best method to avoid this is to familiarize yourself with the [mailing list archives](#), to help yourself understand the background of what has gone before. In this, the [mailing list search interface](#) is invaluable. (If that method does not yield useful results, please supplement it with a search with your favorite major search engine).

By familiarizing yourself with the archives, not only will you learn what topics have been discussed before, but also how discussion tends to proceed on that list, who the participants are, and who the target audience is. These are always good things to know before you post to any mailing list, not just a OS mailing list.

There is no doubt that the archives are quite extensive, and some questions recur more often than others, sometimes as followups where the subject line no longer accurately reflects the new content. Nevertheless, the burden is on you, the poster, to do your homework to help avoid these recurring topics.

60.4 What Is A “Bikeshed”?

Literally, a *bikeshed* is a small outdoor shelter into which one may store one's two-wheeled form of transportation. However, in OS parlance, the term refers to topics that are simple enough that (nearly) anyone can offer an opinion about, and often (nearly) everyone does. The genesis of this term is explained in more detail in this document. You simply must have a working knowledge of this concept before posting to any OS mailing list.

More generally, a bikeshed is a topic that will tend to generate immediate meta-discussions and flames if you have not read up on their past history.

Please help us to keep the mailing lists as useful for as many people as possible by avoiding bikesheds whenever you can. Thanks.

60.5 Acknowledgments

A.GROG.EMAIL Original author of most of the material on mailing list etiquette, taken from the article on How to get best results from the FreeBSD-questions mailing list.

A.LINIMON.EMAIL Creation of the rough draft of this FAQ.

Introduction to NanoBSD

Author DanielGerzo

61.1 Introduction to NanoBSD

NanoBSD is a tool currently developed by A.PHK.EMAIL. It creates a OS system image for embedded applications, suitable for use on a Compact Flash card (or other mass storage medium).

It can be used to build specialized install images, designed for easy installation and maintenance of systems commonly called “computer appliances”. Computer appliances have their hardware and software bundled in the product, which means all applications are pre-installed. The appliance is plugged into an existing network and can begin working (almost) immediately.

The features of NanoBSD include:

- Ports and packages work as in OS — Every single application can be installed and used in a NanoBSD image, the same way as in OS.
- No missing functionality — If it is possible to do something with OS, it is possible to do the same thing with NanoBSD, unless the specific feature or features were explicitly removed from the NanoBSD image when it was created.
- Everything is read-only at run-time — It is safe to pull the power-plug. There is no necessity to run `MAN.FSCK.8` after a non-graceful shutdown of the system.
- Easy to build and customize — Making use of just one shell script and one configuration file it is possible to build reduced and customized images satisfying any arbitrary set of requirements.

61.2 NanoBSD Howto

61.2.1 The Design of NanoBSD

Once the image is present on the medium, it is possible to boot NanoBSD. The mass storage medium is divided into three parts by default:

- Two image partitions: `code#1` and `code#2`.
- The configuration file partition, which can be mounted under the `/cfg` directory at run time.

These partitions are normally mounted read-only.

The `/etc` and `/var` directories are `MAN.MD.4` (malloc) disks.

The configuration file partition persists under the `/cfg` directory. It contains files for `/etc` directory and is briefly mounted read-only right after the system boot, therefore it is required to copy modified files from `/etc` back to the `/cfg` directory if changes are expected to persist after the system restarts.

```
PROMPT.ROOT vi /etc/resolv.conf
[...]
PROMPT.ROOT mount /cfg
PROMPT.ROOT cp /etc/resolv.conf /cfg
PROMPT.ROOT umount /cfg

**Note**

The partition containing ``/cfg`` should be mounted only at boot
time and while overriding the configuration files.

Keeping ``/cfg`` mounted at all times is not a good idea, especially
if the NanoBSD system runs off a mass storage medium that may be
adversely affected by a large number of writes to the partition
(like when the filesystem syncer flushes data to the system disks).
```

61.2.2 Building a NanoBSD Image

A NanoBSD image is built using a simple `nanobsd.sh` shell script, which can be found in the `/usr/src/tools/tools/nanobsd` directory. This script creates an image, which can be copied on the storage medium using the `MAN.DD.1` utility.

The necessary commands to build a NanoBSD image are:

```
PROMPT.ROOT cd /usr/src/tools/tools/nanobsd
PROMPT.ROOT sh nanobsd.sh
PROMPT.ROOT cd /usr/obj/nanobsd.full
PROMPT.ROOT dd if=_.disk.full of=/dev/da0 bs=64k
```

- Change the current directory to the base directory of the NanoBSD build script.
- Start the build process.
- Change the current directory to the place where the built images are located.
- Install NanoBSD onto the storage medium.

61.2.3 Customizing a NanoBSD Image

This is probably the most important and most interesting feature of NanoBSD. This is also where you will be spending most of the time when developing with NanoBSD.

Invocation of the following command will force the `nanobsd.sh` to read its configuration from `myconf.nano` located in the current directory:

```
PROMPT.ROOT sh nanobsd.sh -c myconf.nano
```

Customization is done in two ways:

- Configuration options
- Custom functions

Configuration Options

With configuration settings, it is possible to configure options passed to both the buildworld and installworld stages of the NanoBSD build process, as well as internal options passed to the main build process of NanoBSD. Through these options it is possible to cut the system down, so it will fit on as little as 64MB. You can use the configuration options to trim down OS even more, until it will consists of just the kernel and two or three files in the userland.

The configuration file consists of configuration options, which override the default values. The most important directives are:

- `NANO_NAME` — Name of build (used to construct the workdir names).
- `NANO_SRC` — Path to the source tree used to build the image.
- `NANO_KERNEL` — Name of kernel configuration file used to build kernel.
- `CONF_BUILD` — Options passed to the buildworld stage of the build.
- `CONF_INSTALL` — Options passed to the installworld stage of the build.
- `CONF_WORLD` — Options passed to both the buildworld and the installworld stage of the build.
- `FlashDevice` — Defines what type of media to use. Check `FlashDevice.sub` for more details.

Custom Functions

It is possible to fine-tune NanoBSD using shell functions in the configuration file. The following example illustrates the basic model of custom functions:

```
cust_foo () (
    echo "bar=baz" > \
        ${NANO_WORLDDIR}/etc/foo
)
customize_cmd cust_foo
```

A more useful example of a customization function is the following, which changes the default size of the `/etc` directory from 5MB to 30MB:

```
cust_etc_size () (
    cd ${NANO_WORLDDIR}/conf
    echo 30000 > default/etc/md_size
)
customize_cmd cust_etc_size
```

There are a few default pre-defined customization functions ready for use:

- `cust_comconsole` — Disables `MAN.GETTY.8` on the VGA devices (the `/dev/ttyv*` device nodes) and enables the use of the `COM1` serial port as the system console.
- `cust_allow_ssh_root` — Allow root to login via `MAN.SSHD.8`.
- `cust_install_files` — Installs files from the `nanobsd/Files` directory, which contains some useful scripts for system administration.

Adding Packages

Packages can be added to a NanoBSD image using a custom function. The following function will install all the packages located in `/usr/src/tools/tools/nanobsd/packages`:

```
install_packages () (  
mkdir -p ${NANO_WORLDDIR}/packages  
cp /usr/src/tools/tools/nanobsd/packages/* ${NANO_WORLDDIR}/packages  
chroot ${NANO_WORLDDIR} sh -c 'cd packages; pkg_add -v *;cd ..;'  
rm -rf ${NANO_WORLDDIR}/packages  
)  
customize_cmd install_packages
```

Configuration File Example

A complete example of a configuration file for building a custom NanoBSD image can be:

```
NANO_NAME=custom  
NANO_SRC=/usr/src  
NANO_KERNEL=MYKERNEL  
NANO_IMAGES=2  
  
CONF_BUILD='  
WITHOUT_KLDLOAD=YES  
WITHOUT_NETGRAPH=YES  
WITHOUT_PAM=YES  
'  
  
CONF_INSTALL='  
WITHOUT_ACPI=YES  
WITHOUT_BLUETOOTH=YES  
WITHOUT_FORTRAN=YES  
WITHOUT_HTML=YES  
WITHOUT_LPR=YES  
WITHOUT_MAN=YES  
WITHOUT_SENDMAIL=YES  
WITHOUT_SHAREDOCS=YES  
WITHOUT_EXAMPLES=YES  
WITHOUT_INSTALLLIB=YES  
WITHOUT_CALENDAR=YES  
WITHOUT_MISC=YES  
WITHOUT_SHARE=YES  
'  
  
CONF_WORLD='  
WITHOUT_BIND=YES  
WITHOUT_MODULES=YES  
WITHOUT_KERBEROS=YES  
WITHOUT_GAMES=YES  
WITHOUT_RESCUE=YES  
WITHOUT_LOCALES=YES  
WITHOUT_SYSCONS=YES  
WITHOUT_INFO=YES  
'  
  
FlashDevice SanDisk 1G  
  
cust_nobeastie() (  
    touch ${NANO_WORLDDIR}/boot/loader.conf  
    echo "beastie_disable=\"YES\"" >> ${NANO_WORLDDIR}/boot/loader.conf  
)
```

```
customize_cmd cust_comconsole
customize_cmd cust_install_files
customize_cmd cust_allow_ssh_root
customize_cmd cust_nobeastie
```

61.2.4 Updating NanoBSD

The update process of NanoBSD is relatively simple:

Build a new NanoBSD image, as usual.

Upload the new image into an unused partition of a running NanoBSD appliance.

The most important difference of this step from the initial NanoBSD installation is that now instead of using `_.disk.full` (which contains an image of the entire disk), the `_.disk.image` image is installed (which contains an image of a single system partition).

Reboot, and start the system from the newly installed partition.

If all goes well, the upgrade is finished.

If anything goes wrong, reboot back into the previous partition (which contains the old, working image), to restore system functionality as fast as possible. Fix any problems of the new build, and repeat the process.

To install new image onto the running NanoBSD system, it is possible to use either the `update1` or `update2` script located in the `/root` directory, depending from which partition is running the current system.

According to which services are available on host serving new NanoBSD image and what type of transfer is preferred, it is possible to examine one of these three ways:

Using MAN.FTP.1

If the transfer speed is in first place, use this example:

```
PROMPT.ROOT ftp myhost
get _.disk.image "| sh update1"
```

Using MAN.SSH.1

If a secure transfer is preferred, consider using this example:

```
PROMPT.ROOT ssh myhost cat _.disk.image.gz | zcat | sh update1
```

Using MAN.NC.1

Try this example if the remote host is not running neither MAN.FTPD.8 or MAN.SSHD.8 service:

At first, open a TCP listener on host serving the image and make it send the image to client:

```
myhostPROMPT.ROOT nc -l 2222 < _.disk.image

**Note**

Make sure that the used port is not blocked to receive incoming
connections from NanoBSD host by firewall.
```

Connect to the host serving new image and execute `updatep1` script:

```
PROMPT.ROOT nc myhost 2222 | sh updatep1
```

For People New to Both FreeBSD and UNIX

Author AnneliseAnderson

62.1 Logging in and Getting Out

Log in (when you see `login:`) as a user you created during installation or as root. (Your FreeBSD installation will already have an account for root; who can go anywhere and do anything, including deleting essential files, so be careful!) The symbols `PROMPT.USER` and `PROMPT.ROOT` in the following stand for the prompt (yours may be different), with `PROMPT.USER` indicating an ordinary user and `PROMPT.ROOT` indicating root.

To log out (and get a new `login:` prompt) type

```
PROMPT.ROOT exit
```

as often as necessary. Yes, press enter after commands, and remember that UNIX is case-sensitive—`exit`, not `EXIT`.

To shut down the machine type

```
PROMPT.ROOT /sbin/shutdown -h now
```

Or to reboot type

```
PROMPT.ROOT /sbin/shutdown -r now
```

or

```
PROMPT.ROOT /sbin/reboot
```

You can also reboot with `Ctrl+Alt+Delete`. Give it a little time to do its work. This is equivalent to `/sbin/reboot` in recent releases of FreeBSD and is much, much better than hitting the reset button. You do not want to have to reinstall this thing, do you?

62.2 Adding A User with Root Privileges

If you did not create any users when you installed the system and are thus logged in as root, you should probably create a user now with

```
PROMPT.ROOT adduser
```

The first time you use `adduser`, it might ask for some defaults to save. You might want to make the default shell `MAN.CSH.1` instead of `MAN.SH.1`, if it suggests `sh` as the default. Otherwise just press enter to accept each default. These defaults are saved in `/etc/adduser.conf`, an editable file.

Suppose you create a user jack with full name *Jack Benimble*. Give jack a password if security (even kids around who might pound on the keyboard) is an issue. When it asks you if you want to invite jack into other groups, type wheel

```
Login group is ``jack''. Invite jack into other groups: wheel
```

This will make it possible to log in as jack and use the MAN.SU.1 command to become root. Then you will not get scolded any more for logging in as root.

You can quit `adduser` any time by typing `Ctrl+C`, and at the end you will have a chance to approve your new user or simply type `n` for no. You might want to create a second new user so that when you edit jack's login files, you will have a hot spare in case something goes wrong.

Once you have done this, use `exit` to get back to a login prompt and log in as jack. In general, it is a good idea to do as much work as possible as an ordinary user who does not have the power—and risk—of root.

If you already created a user and you want the user to be able to `su` to root, you can log in as root and edit the file `/etc/group`, adding jack to the first line (the group wheel). But first you need to practice MAN.VI.1, the text editor—or use the simpler text editor, MAN.EE.1, installed on recent versions of FreeBSD.

To delete a user, use the `rmuser` command.

62.3 Looking Around

Logged in as an ordinary user, look around and try out some commands that will access the sources of help and information within FreeBSD.

Here are some commands and what they do:

id Tells you who you are!

pwd Shows you where you are—the current working directory.

ls Lists the files in the current directory.

ls -F Lists the files in the current directory with a `*` after executables, a `/` after directories, and an `@` after symbolic links.

ls -l Lists the files in long format—size, date, permissions.

ls -a Lists hidden “dot” files with the others. If you are root, the “dot” files show up without the `-a` switch.

cd

Changes directories. “**cd ..**” backs up one level; note the space after `cd`. `cd /usr/local` goes there. `cd ~` goes to the home directory

of the person logged in—e.g., `/usr/home/jack`. Try “cd /cdrom**”, and then `ls`, to find out if your CDROM is**

mounted and working.

“**less**

filename”

Lets you look at a file (named filename) without changing it. Try `less /etc/fstab`. Type `q` to quit.

“**cat**

filename”

Displays filename on screen. If it is too long and you can see only the end of it, press ScrollLock and use the up-arrow to move backward; you can use ScrollLock with manual pages too. Press ScrollLock again to quit scrolling. You might want to try `cat` on some of the dot files in your home directory—“`cat`

```
.cshrc“, cat .login, cat .profile.
```

You will notice aliases in `.cshrc` for some of the `ls` commands (they are very convenient). You can create other aliases by editing `.cshrc`. You can make these aliases available to all users on the system by putting them in the system-wide `cs`h configuration file, `/etc/csh.cshrc`.

62.4 Getting Help and Information

Here are some useful sources of help. Text stands for something of your choice that you type in—usually a command or filename.

“**apropos**

```
text“
```

Everything containing string text in the `whatis` database.

“**man**

```
text“
```

The manual page for text. The major source of documentation for UNIX systems. `man ls` will tell you all the ways to use the `ls` command. Press Enter to move through text, Ctrl+B to go back a page, Ctrl+F to go forward, q or Ctrl+C to quit.

“**which**

```
text“
```

Tells you where in the user’s path the command text is found.

“**locate**

```
text“
```

All the paths where the string text is found.

“**whatis**

```
text“
```

Tells you what the command text does and its manual page. Typing `whatis *` will tell you about all the binaries in the current directory.

“**whereis**

```
text“
```

Finds the file text, giving its full path.

You might want to try using `whatis` on some common useful commands like `cat`, `more`, `grep`, `mv`, `find`, `tar`, `chmod`, `chown`, `date`, and `script`. `more` lets you read a page at a time as it does in DOS, e.g., “`ls -ll`

```
more“ or more filename. The * works as a wildcard—e.g., ls w* will show you files beginning with w.
```

Are some of these not working very well? Both `MAN.LOCATE.1` and `MAN.WHATIS.1` depend on a database that is rebuilt weekly. If your machine is not going to be left on over the weekend (and running FreeBSD), you might want to run the commands for daily, weekly, and monthly maintenance now and then. Run them as root and, for now, give each one time to finish before you start the next one.

```
PROMPT.ROOT periodic daily
output omitted
PROMPT.ROOT periodic weekly
output omitted
PROMPT.ROOT periodic monthly
output omitted
```

If you get tired of waiting, press **Alt+F2** to get another virtual console, and log in again. After all, it is a multi-user, multi-tasking system. Nevertheless these commands will probably flash messages on your screen while they are running; you can type `clear` at the prompt to clear the screen. Once they have run, you might want to look at `/var/mail/root` and `/var/log/messages`.

Running such commands is part of system administration—and as a single user of a UNIX system, you are your own system administrator. Virtually everything you need to be root to do is system administration. Such responsibilities are not covered very well even in those big fat books on UNIX, which seem to devote a lot of space to pulling down menus in windows managers. You might want to get one of the two leading books on systems administration, either Evi Nemeth et.al.'s *UNIX System Administration Handbook* (Prentice-Hall, 1995, ISBN 0-13-15051-7)—the second edition with the red cover; or Aileen Frisch's *Essential System Administration* (O'Reilly & Associates, 2002, ISBN 0-596-00343-9). I used Nemeth.

62.5 Editing Text

To configure your system, you need to edit text files. Most of them will be in the `/etc` directory; and you will need to `su` to root to be able to change them. You can use the easy `ee`, but in the long run the text editor `vi` is worth learning. There is an excellent tutorial on `vi` in `/usr/src/contrib/nvi/docs/tutorial`, if you have the system sources installed.

Before you edit a file, you should probably back it up. Suppose you want to edit `/etc/rc.conf`. You could just use `cd /etc` to get to the `/etc` directory and do:

```
PROMPT.ROOT cp rc.conf rc.conf.orig
```

This would copy `rc.conf` to `rc.conf.orig`, and you could later copy `rc.conf.orig` to `rc.conf` to recover the original. But even better would be moving (renaming) and then copying back:

```
PROMPT.ROOT mv rc.conf rc.conf.orig
PROMPT.ROOT cp rc.conf.orig rc.conf
```

because the `mv` command preserves the original date and owner of the file. You can now edit `rc.conf`. If you want the original back, you would then `mv rc.conf rc.conf.myedit` (assuming you want to preserve your edited version) and then

```
PROMPT.ROOT mv rc.conf.orig rc.conf
```

to put things back the way they were.

To edit a file, type

```
PROMPT.ROOT vi filename
```

Move through the text with the arrow keys. `Esc` (the escape key) puts `vi` in command mode. Here are some commands:

x delete letter the cursor is on

dd delete the entire line (even if it wraps on the screen)

i insert text at the cursor

a insert text after the cursor

Once you type `i` or `a`, you can enter text. `Esc` puts you back in command mode where you can type

`:w` to write your changes to disk and continue editing

`:wq` to write and quit

`:q!` to quit without saving changes

`/text` to move the cursor to text; `/Enter` (the enter key) to find the next instance of text.

`G` to go to the end of the file

`nG` to go to line `n` in the file, where `n` is a number

`Ctrl+L` to redraw the screen

`Ctrl+b` and `Ctrl+f` go back and forward a screen, as they do with `more` and `view`.

Practice with `vi` in your home directory by creating a new file with “`vi`

`filename`” and adding and deleting text, saving the file, and

calling it up again. `vi` delivers some surprises because it is really quite complex, and sometimes you will inadvertently issue a command that will do something you do not expect. (Some people actually like `vi`—it is more powerful than DOS EDIT—find out about the `:r` command.) Use `Esc` one or more times to be sure you are in command mode and proceed from there when it gives you trouble, save often with `:w`, and use `:q!` to get out and start over (from your last `:w`) when you need to.

Now you can `cd` to `/etc`, `su` to root, use `vi` to edit the file `/etc/group`, and add a user to wheel so the user has root privileges. Just add a comma and the user’s login name to the end of the first line in the file, press `Esc`, and use `:wq` to write the file to disk and quit. Instantly effective. (You did not put a space after the comma, did you?)

62.6 Other Useful Commands

`df` shows file space and mounted systems.

`ps aux` shows processes running. `ps ax` is a narrower form.

`rm filename` remove filename.

`rm -R dir` removes a directory `dir` and all subdirectories—careful!

`ls -R` lists files in the current directory and all subdirectories; I used a variant, “`ls -AFR >`

`where.txt`”, to get a list of all the files in `/` and

(separately) `/usr` before I found better ways to find files.

`passwd` to change user’s password (or root’s password)

`man hier` manual page on the UNIX filesystem

Use `find` to locate filename in `/usr` or any of its subdirectories with

```
PROMPT.USER find /usr -name "filename"
```

You can use `*` as a wildcard in “filename” (which should be in quotes). If you tell `find` to search in `/` instead of `/usr` it will look for the file(s) on all mounted filesystems, including the CDROM and the DOS partition.

An excellent book that explains UNIX commands and utilities is Abrahams & Larson, *Unix for the Impatient* (2nd ed., Addison-Wesley, 1996). There is also a lot of UNIX information on the Internet.

62.7 Next Steps

You should now have the tools you need to get around and edit files, so you can get everything up and running. There is a great deal of information in the FreeBSD handbook (which is probably on your hard drive) and FreeBSD's web site. A wide variety of packages and ports are on the CDROM as well as the web site. The handbook tells you more about how to use them (get the package if it exists, with `pkg_add`

`/cdrom/packages/All/packageName`, where `packageName` is the filename of the package). The CDROM has lists of the packages and ports with brief descriptions in `cdrom/packages/index`, `cdrom/packages/index.txt`, and `cdrom/ports/index`, with fuller descriptions in `/cdrom/ports/*/pkg/DESCR`, where the `*`s represent subdirectories of kinds of programs and program names respectively.

If you find the handbook too sophisticated (what with `ln` and all) on installing ports from the CDROM, here is what usually works:

Find the port you want, say `kermit`. There will be a directory for it on the CDROM. Copy the subdirectory to `/usr/local` (a good place for software you add that should be available to all users) with:

```
PROMPT.ROOT cp -R /cdrom/ports/comm/kermit /usr/local
```

This should result in a `/usr/local/kermit` subdirectory that has all the files that the `kermit` subdirectory on the CDROM has.

Next, create the directory `/usr/ports/distfiles` if it does not already exist using `mkdir`. Now check `/cdrom/ports/distfiles` for a file with a name that indicates it is the port you want. Copy that file to `/usr/ports/distfiles`; in recent versions you can skip this step, as FreeBSD will do it for you. In the case of `kermit`, there is no `distfile`.

Then `cd` to the subdirectory of `/usr/local/kermit` that has the file `Makefile`. Type

```
PROMPT.ROOT make all install
```

During this process the port will FTP to get any compressed files it needs that it did not find on the CDROM or in `/usr/ports/distfiles`. If you do not have your network running yet and there was no file for the port in `/cdrom/ports/distfiles`, you will have to get the `distfile` using another machine and copy it to `/usr/ports/distfiles`. Read `Makefile` (with `cat` or `more` or `view`) to find out where to go (the master distribution site) to get the file and what its name is. (Use binary file transfers!) Then go back to `/usr/local/kermit`, find the directory with `Makefile`, and type `make all install`.

62.8 Your Working Environment

Your shell is the most important part of your working environment. The shell is what interprets the commands you type on the command line, and thus communicates with the rest of the operating system. You can also write shell scripts a series of commands to be run without intervention.

Two shells come installed with FreeBSD: `csh` and `sh`. `csh` is good for command-line work, but scripts should be written with `sh` (or `bash`). You can find out what shell you have by typing `echo $SHELL`.

The `csh` shell is okay, but `tcsh` does everything `csh` does and more. It allows you to recall commands with the arrow keys and edit them. It has tab-key completion of filenames (`csh` uses the `Esc` key), and it lets you switch to the directory you were last in with `cd -`. It is also much easier to alter your prompt with `tcsh`. It makes life a lot easier.

Here are the three steps for installing a new shell:

Install the shell as a port or a package, just as you would any other port or package.

Use the `chsh` command to change your shell to `tcsh` permanently, or type `tcsh` at the prompt to change your shell without logging in again.

Note

It can be dangerous to change root's shell to something other than `sh` or `csh` on early versions of FreeBSD and many other versions of UNIX; you may not have a working shell when the system puts you into single user mode. The solution is to use “`su`

`-m`” to become root, which will give you the `tcsh` as root,

because the shell is part of the environment. You can make this permanent by adding it to your `.tcshrc` file as an alias with:

```
alias su su -m
```

When `tcsh` starts up, it will read the `/etc/csh.cshrc` and `/etc/csh.login` files, as does `csh`. It will also read the `.login` file in your home directory and the `.cshrc` file as well, unless you provide a `.tcshrc` file. This you can do by simply copying `.cshrc` to `.tcshrc`.

Now that you have installed `tcsh`, you can adjust your prompt. You can find the details in the manual page for `tcsh`, but here is a line to put in your `.tcshrc` that will tell you how many commands you have typed, what time it is, and what directory you are in. It also produces a `>` if you are an ordinary user and a `#` if you are root, but `tsch` will do that in any case:

```
set prompt = "%h %t %~ %#"
```

This should go in the same place as the existing `set prompt` line if there is one, or under “if(`$?prompt`) then” if not. Comment out the old line; you can always switch back to it if you prefer it. Do not forget the spaces and quotes. You can get the `.tcshrc` reread by typing `source .tcshrc`.

You can get a listing of other environmental variables that have been set by typing `env` at the prompt. The result will show you your default editor, pager, and terminal type, among possibly many others. A useful command if you log in from a remote location and can not run a program because the terminal is not capable is “`setenv TERM`

`vt100`”.

62.9 Other

As root, you can unmount the CDROM with `/sbin/umount /cdrom`, take it out of the drive, insert another one, and mount it with `/sbin/mount_cd9660 /dev/cd0a /cdrom` assuming `cd0a` is the device name for your CDROM drive. The most recent versions of FreeBSD let you mount the CDROM with just `/sbin/mount /cdrom`.

Using the live filesystem—the second of FreeBSD's CDROM disks—is useful if you have got limited space. What is on the live filesystem varies from release to release. You might try playing games from the CDROM. This involves using `lndir`, which gets installed with the X Window System, to tell the program(s) where to find the necessary files, because they are in the `/cdrom` file system instead of in `/usr` and its subdirectories, which is where they are expected to be. Read `man lndir`.

62.10 Comments Welcome

If you use this guide I would be interested in knowing where it was unclear and what was left out that you think should be included, and if it was helpful. My thanks to Eugene W. Stark, professor of computer science at SUNY-Stony Brook, and John Fieber for helpful comments.

Annelise Anderson, andrsn@andrsn.stanford.edu

Perforce in OS Development

Author Scott Long

63.1 Introduction

The OS project uses the Perforce version control system to manage experimental projects that are not ready for the main Subversion repository.

63.1.1 Availability, Documentation, and Resources

While Perforce is a commercial product, the client software for interacting with the server is freely available from Perforce. It can be easily installed on OS via the devel/p4 port or can be downloaded from the Perforce web site at <http://www.perforce.com/perforce/loadprog.html>, which also offers client applications for other OS's.

While there is a GUI client available, most people use the command line application called p4. This document is written from the point of view of using this command.

Detailed documentation is available online at <http://www.perforce.com/perforce/technical.html>.

Reading the “Perforce User’s Guide” and “Perforce Command Reference” is highly recommended. The p4 application also contains an extensive amount of online help accessible via `p4 help`.

The OS Perforce server is hosted on perforce.freebsd.org, port 1666. The repository is browsable online at <http://p4web.freebsd.org>.

63.2 Getting Started

The first step to using Perforce is to obtain an account on the server. If you already have a FreeBSD.org account, log into freefall, run the following command, and enter a password that is not the same as your OS login or any other SSH passphrase:

```
PROMPT.USER /usr/local/bin/p4newuser
```

Of course if you do not have a FreeBSD.org account, you will need to coordinate with your sponsor.

Warning

An email will be sent to your OS address that contains the password you specified above in cleartext. Be sure to change the password once your Perforce account has been created!

The next step is to set the environment variables that `p4` needs, and verify that it can connect to the server. The `P4PORT` variable is required to be set for all operations, and specifies the appropriate Perforce server to talk to. For the OS project, set it like so:

```
PROMPT.USER export P4PORT=perforce.freebsd.org:1666

**Note**

Users with shell access on the FreeBSD.org cluster may wish to
tunnel the Perforce client-server protocol via an SSH tunnel, in
which case the above string should be set to ``localhost``.
```

The OS server also requires that the `P4USER` and `P4PASSWD` variables be set. Use the username and password from above, like so:

```
PROMPT.USER export P4USER=username
PROMPT.USER export P4PASSWD=password
```

Test that this works by running the following command:

```
PROMPT.USER p4 info
```

This should return a list of information about the server. If it does not, check that you have the `P4PORT` variable set correctly.

63.3 Clients

Perforce provides access to the repository and tracks state on a per-client basis. In Perforce terms, a client is a specification that maps files and directories from the repository to the local machine. Each user can have multiple clients, and each client can access different or overlapping parts of the repository. The client also specifies the root directory of the file tree that it maps, and it specifies the machine that the tree lives on. Thus, working on multiple machines requires that multiple clients be used.

Clients may be accessed via `p4 client`. Running this command with no arguments will bring up a client template in an editor, allowing you to create a new client for your work. The important fields in this template are explained below:

Client: This is the name of the client spec. It can be anything you want, but it must be unique within the Perforce server. A naming convention that is commonly used is `username_machinename`, which makes it easy to identify clients when browsing them. A default name will be filled in that is just the machine name.

Description: This can contain a simple text description to help identify the client.

Root: This is the local directory that will serve as the root directory of all the files in the client mapping. This should be a unique location in your filesystem that does not overlap with other files or Perforce clients.

Options: Most of the default options are fine, though it is usually a good idea to make sure that the `compress` and `rmdir` options are present and do not have a `no` prefix on them. Details about each option are in the Perforce docs.

LineEnd: This handles CR-LF conversions and should be left to the default unless you have special needs for it.

View: This is where the server-to-local file mappings go. The default is

```
//depot/... //client/...
```

This will map the entire Perforce repository to the `Root` directory of your client. *DO NOT USE THIS DEFAULT!* The OS repo is huge, and trying to map and sync it all will take an enormous amount of resources.

Instead, only map the section of the repo that you intend to work on. For example, there is the `smpng` project tree at `//depot/projects/smpng`. A mapping for this might look like:

```
//depot/projects/smpng/... //client/...
```

The ... should be taken literally. It is a Perforce idiom for saying “this directory and all files and directories below it.”

A Perforce “view” can contain multiple mappings. Say you want to map in both the `SMPng` tree and the `NFS` tree. Your View might look like:

```
//depot/projects/smpng/... //client/smpng/...
//depot/projects/nfs/... //client/nfs/...
```

Remember that the client is the name of the client that was specified in the `Client` section, but in the `View` it also resolves to the directory that was specified in the `Root` section.

Also note that the same file or directory cannot be mapped multiple times in a single view. The following is illegal and will produce undefined results:

```
//depot/projects/smpng/... //client/smpng-foo/...
//depot/projects/smpng/... //client/smpng-bar/...
```

Views are a tricky part of the learning experience with Perforce, so do not be afraid to ask questions.

Existing clients can be listed via “`p4 clients`”. They can be viewed without being modified via

“`p4 client -o clientname`”.

Whenever you are interacting with files in Perforce, the `P4CLIENT` environment variable must be set to the name of the client that you are using, like so:

```
PROMPT.USER export P4CLIENT=myclientname
```

Note that client mappings in the repository are not exclusive; multiple clients can map in the same part of the repository. This allows multiple people to access and modify the same parts of the repository, allowing a team of people to work together on the same code.

63.4 Syncing

Once you have a client specification defined and the `P4CLIENT` variable set, the next step is to pull the files for that client down to your local machine. This is done with `p4 sync`, which instructs Perforce to synchronize the local files in your client with the repository. The first time it runs, it will download all of the files. Subsequent runs will only download files that have changed since the previous run. This allows you to stay in sync with others whom you might be working with.

Sync operations only work on files that the Perforce server knows has changed. If you change or delete a file locally without informing the server, doing a sync will not bring it back. However, doing a `p4 sync -f` will unconditionally sync all files, regardless of their state. This is useful for resolving problems where you think that your tree might be corrupt.

You can sync a subset of your tree or client by specifying a relative path to the sync command. For example, to only sync the `ufs` directory of the `smpng` project, you might do the following:

```
PROMPT.USER cd projectroot/smpng
PROMPT.USER p4 sync src/sys/ufs/...
```

Specifying a local relative path works for many other `p4` commands.

63.5 Branches

One of the strongest features of Perforce is branching. Branches are very cheap to create, and moving changes between related branches is very easy (as will be explained later). Branches also allow you to do very experimental work in a sandbox-like environment, without having to worry about colliding with others or destabilizing the main tree. They also provide insulation against mistakes while learning the Perforce system. With all of these benefits, it makes sense for each project to have its own branch, and we strongly encourage that with OS. Frequent submits of changes to the server are also encouraged.

Similar to Subversion, the Perforce repository (the “depot”) is a single flat tree. Every file, whether a unique creation or a derivative from a branch, is accessible via a simple path under the server `//depot` directory. When you create a branch, all you are doing is creating a new path under the `//depot`. This is in sharp contrast to systems like CVS, where each branch lives in the same path as its parent. With Perforce, the server tracks the relationship between the files in the parent and child, but the files themselves live under their own paths.

The first step to creating a branch is to create a branch specification. This is similar to a client specification, but is created via the command “`p4 branch`

```
branchname“.
```

The following important fields are explained:

Branch The name of the branch. It can be any name, but must be unique within the repository. The common convention in OS is to use `username_projectname`.

Description This can hold a simple text description to describe the branch.

View This is the branch mapping. Instead of mapping from the depot to the local machine like a client map, it maps between the branch parent and branch child in the depot. For example, you might want to create a branch of the `smpng` project. The mapping might look like:

```
//depot/projects/smpng/... //depot/projects/my-super-smpng/...
```

Or, you might want to create a brand new branch off of the stock OS sources:

```
//depot/vendor/freebsd/... //depot/projects/my-new-project/...
```

This will map the OS HEAD tree to your new branch.

Creating the branch spec only saves the spec itself in the server, it does not modify the depot or change any files. The directory that you specified in the branch is empty on the server until you populate it.

To populate your branch, first edit your client with `p4 client` and make sure that the branch directory is mapped in your client. You might need to add a `View` line like:

```
//depot/projects/my-new-project/... //myclient/my-new-project/...
```

The next step is to run `p4 integrate`, as described in the next section.

63.6 Integrations

“Integration” is the term used by Perforce to describe the action of moving changes from one part of the depot to another. It is most commonly done in conjunction with creating and maintaining branches. An integration is done when you want to initially populate a branch, and it is done when you want to move subsequent changes in the branch from the parent to the child, or from the child to the parent. A common example of this is periodically integrating changes from the vendor OS tree to your child branch tree, allowing you to keep up to date with changes in the OS tree. The Perforce server tracks the changes in each tree and knows when there are changes that can be integrated from one tree to another.

The common way to do an integration is with the following command:

```
PROMPT.USER p4 integrate -b branchname
```

branchname is the name given to a branch spec, as discussed in the previous section. This command will instruct Perforce to look for changes in the branch parent that are not yet in the child. From those changes it will prepare a list of diffs to move. If the integration is being done for the first time on a branch (for example doing an initial population operation), then the parent files will simply be copied to the child location on the local machine.

Once the integration operation is done, you must run `p4 resolve` to accept the changes and resolve possible conflicts. Conflicts can arise from overlapping changes that happened in both the parent and child copy of a file. Usually, however, there are no conflicts, and Perforce can quickly figure out how to merge the changes together. Use the following commands to do a resolve operation:

```
PROMPT.USER p4 resolve -as
PROMPT.USER p4 resolve
```

The first invocation will instruct Perforce to automatically merge the changes together and accept files that have no conflicts. The second invocation will allow you to inspect each file that has a possible conflict and resolve it by hand if needed.

Once all of the integrated files have been resolved, they need to be committed back to the repository. This is done via `p4 submit`, explained in the next section.

63.7 Submit

Changes that are made locally should be committed back to the Perforce server for safe keeping and so that others can access them. This is done via `p4 submit`. When you run this command, it will open up a submit template in an editor. OS has a custom template, and the important fields are described below:

```
Description:
    <enter description here>
PR:
Submitted by:
Reviewed by:
Approved by:
Obtained from:
MFP4 after:
```

It is good practice to provide at least 2-3 sentences that describe what the changes are that you are submitting. You should say what the change does, why it was done that way or what problem it solves, and what APIs it might change or other side effects it might have. This text should replace the `<enter description here>` line in the template. You should wrap your lines and start each line with a TAB. The tags below it are OS-specific and can be removed if not needed.

```
Files:
```

This is automatically populated with all of the files in your client that were marked in the add, delete, integrate, or edit states on the server. It is always a very good idea to review this list and remove files that might not be ready yet.

Once you save the editor session, the submit will happen to the server. This also means that the local copies of the submitted files will be copied back to the server. If anything goes wrong during this process, the submit will be aborted, and you will be notified that the submit has been turned into a changelist that must be corrected and re-submitted. Submits are atomic, so if one file fails, the entire submit is aborted.

Submits cannot be reverted, but they can be aborted while in the editor by exiting the editor without changing the `Description` text. Perforce will complain about this the first time you do it and will put you back in the editor.

Exiting the editor the second time will abort the operation. Reverting a submitted change is very difficult and is best handled on a case-by-case basis.

63.8 Editing

The state of each file in the client is tracked and saved on the server. In order to avoid collisions from multiple people working on the same file at once, Perforce tracks which files are opened for edit, and uses this to help with submit, sync, and integration operations later on.

To open a file for editing, use `p4 edit` like so:

```
PROMPT.USER p4 edit filename
```

This marks the file on the server as being in the *edit* state, which then allows it to be submitted after changes are made, or marks it for special handling when doing an integration or sync operation. Note that editing is not exclusive in Perforce. Multiple people can have the same file in the edit state (you will be informed of others when you run `edit`), and you can submit your changes even when others are still editing the file.

When someone else submits a change to a file that you are editing, you will need to resolve his changes with yours before your submit will succeed. The easiest way to do this is to either run a `p4 sync` or “`p4`

`submit`” and let it fail with the conflict, then run `p4 resolve`

to manually resolve and accept his changes into your copy, then run `p4 submit` to commit your changes to the repository.

If you have a file open for edit and you want to throw away your changes and revert it to its original state, run `p4 revert` like so:

```
PROMPT.USER p4 revert filename
```

This resyncs the file to the contents of the server, and removes the edit attribute from the server. Any local changes that you had will be lost. This is quite useful when you have made changes to a file but later decide that you do not want to keep them.

When a file is synced, it is marked read-only in the filesystem. When you tell the server to open it for editing, it is changed to read-write on the filesystem. While these permissions can easily be overridden by hand, they are meant to gently remind you that you should be using “`p4`

`edit`”. Files that have local changes but are not in the edit state

may get overwritten when doing a “`p4 sync`”.

63.9 Changes, Descriptions, and History

Changes to the Perforce depot can be listed via `p4 changes`. This will provide a brief description of each change, who made the change, and what its change number was. A change can be examined in detail via “`p4 describe`

`changenumber`”. This will provide the submit log and the diffs of the actual change.

Commonly, `p4 describe` is used in one of three ways:

“`p4 describe -s`
`CHANGE`”

List a short description of changeset *CHANGE*, including the commit log of the particular changeset and a list of the files it affected.

“p4 describe -du

CHANGE“

List a description of changeset *CHANGE*, including the commit log of the particular changeset, a list of the files it affected and a patch for each modified file, in a format similar to “unified diff” patches (but not exactly the same).

“p4 describe -dc

CHANGE“

List a description of changeset *CHANGE*, including the commit log of the particular changeset, a list of the files it affected and a patch for each modified file, in a format similar to “context diff” patches (but not exactly the same).

The history of a file, including all submits, integrations, and branches of it will be shown by “p4 filelog filename“.

63.10 Diffs

There are two methods of producing file diffs in Perforce, either against local changes that have not been submitted yet, or between two trees (or within a branch) in the depot. These are done with different commands, `diff` and `diff2`:

p4 diff This generates a diff of the local changes to files in the edit state. The `-du` and `-dc` flags can be used to create unified or context diffs, respectively, or the `P4DIFF` environment variable can be set to a local diff command to be used instead. It is a very good idea to use this command to review your changes before submitting them.

p4 diff2 This creates a diff between arbitrary files in the depot, or between files specified in a branch spec. The diff operation takes place on the server, so `P4DIFF` variable has no effect, though the `-du` and `-dc` flags do work. The two forms of this command are:

```
PROMPT.USER p4 diff2 -b branchname
```

and

```
PROMPT.USER p4 diff2 //depot/path1 //depot/path2
```

In all cases the diff will be written to the standard output. Unfortunately, Perforce produces a diff format that is slightly incompatible with the traditional Unix diff and patch tools. Using the `P4DIFF` variable to point to the real `MAN.DIFF.1` tool can help this, but only for `p4 diff`. The output of `diff2` command must be post-processed to be useful (the `-u` flag of `diff2` will produce unified diffs that are somewhat compatible, but it does not include files that have been added or deleted). There is a post-processing script at: <https://svnweb.freebsd.org/base/head/tools/tools/perforce/awkdifff?view=co>.

63.11 Adding and Removing Files

Integrating a branch will bring existing files into your tree, but you may still want to add new files or remove existing ones. Adding files is easily done by creating the file and then running `p4 add` like so:

```
PROMPT.USER p4 add filename
```

If you want to add a whole tree of files, run a command like:

```
PROMPT.USER find . -type f | xargs p4 add

**Note**

Perforce can track UNIX symlinks too, so you can probably use
``\! -type d`` as the matching expression in MAN.FIND.1 above. We
do not commit symlinks into the source tree of OS though, so this
should not be necessary.
```

Doing a `p4 submit` will then copy the file to the depot on the server. It is very important to only add files, not directories. Explicitly adding a directory will cause Perforce to treat it like a file, which is not what you want.

Removing a file is just as easy with the `p4 delete` command like so:

```
PROMPT.USER p4 delete filename
```

This will mark the file for deletion from the depot the next time that a submit is run. It will also remove the local copy of the file, so beware.

Of course, deleting a file does not actually remove it from the repository.

Deleted files can be resurrected by syncing them to a prior version. The only way to permanently remove a file is to use `p4 obliterate`. This command is irreversible and expensive, so it is only available to those with admin access.

63.12 Working with Diffs

Sometimes you might need to apply a diff from another source to a tree under Perforce control. If it is a large diff that affects lots of files, it might be inconvenient to manually run `p4 edit` on each file. There is a trick for making this easier. First, make sure that no files are open on your client and that your tree is synced and up to date. Then apply the diff using the normal tools, and coerce the permissions on the files if needed. Then run the following commands:

```
PROMPT.USER p4 diff -se ... | xargs p4 edit
PROMPT.USER p4 diff -sd ... | xargs p4 delete
PROMPT.USER find . -type f | xargs p4 add
```

The first command tells Perforce to look for files that have changed, even if they are not open. The second command tells Perforce to look for files that no longer exist on the local machine but do exist on the server. The third command then attempts to add all of the files that it can find locally. This is a very brute-force method, but it works because Perforce will only add the files that it does not already know about. The result of running these commands will be a set of files that are opened for edit, removal, or add, as appropriate.

Verify the active changelist with:

```
PROMPT.USER p4 changelist
PROMPT.USER p4 diff -du
```

and just do a `p4 submit` after that.

63.13 Renaming Files

Perforce does not have a built-in way of renaming files or moving them to a different part of the tree. Simply copying a file to the new location, doing a `p4 add` on it, and a “`p4`

delete“ on the old copy, works, but does not preserve change

history of the file. This can make future integrations with parents and children very bumpy, in fact. A better method of dealing with this is to do a one-time, in-tree integration, like so:

```
PROMPT.USER p4 integrate -i oldfile newfile
PROMPT.USER p4 resolve
PROMPT.USER p4 delete oldfile
PROMPT.USER p4 submit
```

The integration will force Perforce to keep a record of the relationship between the old and new names, which will assist it in future integrations. The `-i` flag tells it that it is a “baseless” integration, meaning that there is no branch history available for it to use in the integration. That is perfect for an integration like this, but should not be used for normal branch-based integrations.

63.14 Interactions Between OS Subversion and Perforce

The OS Perforce and Subversion repositories are completely separate. However, changes to Subversion are tracked at near-real-time in Perforce. Every 2 minutes, the Subversion server is polled for updates in the HEAD branch, and those updates are committed to Perforce in the `//depot/vendor/freebsd/...` tree. This tree is then available for branching and integrating to derivative projects. Any project that directly modifies that OS source code should have this tree as its branch parent (or grandparent, depending on the needs), and periodic integrations and syncs should be done so that your tree stays up to date and avoids conflicts with mainline development.

The bridge between Subversion and Perforce is one-way; changes to Subversion will be reflected in Perforce, but changes in Perforce will not be reflected in Subversion.

63.15 Offline Operation

One weakness of Perforce is that it assumes that network access to the server is always available. Most state, history, and metadata is saved on the server, and there is no provision for replicating the server like there is with SVN. It is possible to run a proxy server, but it only provides very limited utility for offline operation.

The best way to work offline is to make sure that your client has no open files and is fully synced before going offline. Then when editing a file, manually change the permissions to read-write. When you get back online, run the commands listed in the ? to automatically identify files that have been edited, added, and removed. It is quite common to be surprised by Perforce overwriting a locally changed file that was not opened for edit, so be extra vigilant with this.

63.16 Notes for Google Summer of Code

Most OS projects under the Google Summer of Code program are located on the OS Perforce server under one of the following locations:

- `//depot/projects/soc2005/project-name/...`
- `//depot/projects/soc2006/project-name/...`
- `//depot/projects/soc2007/project-name/...`
- `//depot/projects/soc2008/project-name/...`

The project mentor is responsible for choosing a suitable project name and getting the student going with Perforce.

Access to the OS Perforce server does not imply access to subversion, though we happily encourage all students to consider joining the project when the time is appropriate.

Further Reading

Author Dag-Erling Smørgrav **Contributed by**

64.1 Introduction

The Pluggable Authentication Modules (PAM) library is a generalized API for authentication-related services which allows a system administrator to add new authentication methods simply by installing new PAM modules, and to modify authentication policies by editing configuration files.

PAM was defined and developed in 1995 by Vipin Samar and Charlie Lai of Sun Microsystems, and has not changed much since. In 1997, the Open Group published the X/Open Single Sign-on (XSSO) preliminary specification, which standardized the PAM API and added extensions for single (or rather integrated) sign-on. At the time of this writing, this specification has not yet been adopted as a standard.

Although this article focuses primarily on FreeBSD 5.x, which uses OpenPAM, it should be equally applicable to FreeBSD 4.x, which uses Linux-PAM, and other operating systems such as Linux and SOLARIS.

64.2 Terms and conventions

64.2.1 Definitions

The terminology surrounding PAM is rather confused. Neither Samar and Lai's original paper nor the XSSO specification made any attempt at formally defining terms for the various actors and entities involved in PAM, and the terms that they do use (but do not define) are sometimes misleading and ambiguous. The first attempt at establishing a consistent and unambiguous terminology was a whitepaper written by Andrew G. Morgan (author of Linux-PAM) in 1999. While Morgan's choice of terminology was a huge leap forward, it is in this author's opinion by no means perfect. What follows is an attempt, heavily inspired by Morgan, to define precise and unambiguous terms for all actors and entities involved in PAM.

account The set of credentials the applicant is requesting from the arbitrator.

applicant The user or entity requesting authentication.

arbitrator The user or entity who has the privileges necessary to verify the applicant's credentials and the authority to grant or deny the request.

chain A sequence of modules that will be invoked in response to a PAM request. The chain includes information about the order in which to invoke the modules, what arguments to pass to them, and how to interpret the results.

client The application responsible for initiating an authentication request on behalf of the applicant and for obtaining the necessary authentication information from him.

facility One of the four basic groups of functionality provided by PAM: authentication, account management, session management and authentication token update.

module A collection of one or more related functions implementing a particular authentication facility, gathered into a single (normally dynamically loadable) binary file and identified by a single name.

policy The complete set of configuration statements describing how to handle PAM requests for a particular service. A policy normally consists of four chains, one for each facility, though some services do not use all four facilities.

server The application acting on behalf of the arbitrator to converse with the client, retrieve authentication information, verify the applicant's credentials and grant or deny requests.

service A class of servers providing similar or related functionality and requiring similar authentication. PAM policies are defined on a per-service basis, so all servers that claim the same service name will be subject to the same policy.

session The context within which service is rendered to the applicant by the server. One of PAM's four facilities, session management, is concerned exclusively with setting up and tearing down this context.

token A chunk of information associated with the account, such as a password or passphrase, which the applicant must provide to prove his identity.

transaction A sequence of requests from the same applicant to the same instance of the same server, beginning with authentication and session set-up and ending with session tear-down.

64.2.2 Usage examples

This section aims to illustrate the meanings of some of the terms defined above by way of a handful of simple examples.

Client and server are one

This simple example shows `alice` `MAN.SU.1`'ing to `root`.

```
PROMPT.USER whoami
alice
PROMPT.USER ls -l `which su`
-r-sr-xr-x 1 root  wheel  10744 Dec  6 19:06 /usr/bin/su
PROMPT.USER su -
Password: xi3kiune
PROMPT.ROOT whoami
root
```

- The applicant is `alice`.
- The account is `root`.
- The `MAN.SU.1` process is both client and server.
- The authentication token is `xi3kiune`.
- The arbitrator is `root`, which is why `MAN.SU.1` is `setuid root`.

Client and server are separate

The example below shows `eve` try to initiate an `MAN.SSH.1` connection to `login.example.com`, ask to log in as `bob`, and succeed. Bob should have chosen a better password!

```
PROMPT.USER whoami
eve
PROMPT.USER ssh bob@login.example.com
bob@login.example.com's password: god
Last login: Thu Oct 11 09:52:57 2001 from 192.168.0.1
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
    The Regents of the University of California. All rights reserved.
FreeBSD 4.4-STABLE (LOGIN) #4: Tue Nov 27 18:10:34 PST 2001

Welcome to FreeBSD!
PROMPT.USER
```

- The applicant is eve.
- The client is Eve's MAN.SSH.1 process.
- The server is the MAN.SSHD.8 process on login.example.com
- The account is bob.
- The authentication token is god.
- Although this is not shown in this example, the arbitrator is root.

Sample policy

The following is FreeBSD's default policy for sshd:

sshd	auth	required	pam_nologin.so no_warn
sshd	auth	required	pam_unix.so no_warn try_first_pass
sshd	account	required	pam_login_access.so
sshd	account	required	pam_unix.so
sshd	session	required	pam_lastlog.so no_fail
sshd	password	required	pam_permit.so

- This policy applies to the sshd service (which is not necessarily restricted to the MAN.SSHD.8 server.)
- auth, account, session and password are facilities.
- pam_nologin.so, pam_unix.so, pam_login_access.so, pam_lastlog.so and pam_permit.so are modules. It is clear from this example that pam_unix.so provides at least two facilities (authentication and account management.)

64.3 PAM Essentials

64.3.1 Facilities and primitives

The PAM API offers six different authentication primitives grouped in four facilities, which are described below.

auth *Authentication.* This facility concerns itself with authenticating the applicant and establishing the account credentials. It provides two primitives:

- MAN.PAM.AUTHENTICATE.3 authenticates the applicant, usually by requesting an authentication token and comparing it with a value stored in a database or obtained from an authentication server.
- MAN.PAM.SETCRED.3 establishes account credentials such as user ID, group membership and resource limits.

account *Account management.* This facility handles non-authentication-related issues of account availability, such as access restrictions based on the time of day or the server's work load. It provides a single primitive:

- MAN.PAM.ACCT.MGMT.3 verifies that the requested account is available.

session *Session management.* This facility handles tasks associated with session set-up and tear-down, such as login accounting. It provides two primitives:

- MAN.PAM.OPEN.SESSION.3 performs tasks associated with session set-up: add an entry in the `utmp` and `wtmp` databases, start an SSH agent, etc.
- MAN.PAM.CLOSE.SESSION.3 performs tasks associated with session tear-down: add an entry in the `utmp` and `wtmp` databases, stop the SSH agent, etc.

password *Password management.* This facility is used to change the authentication token associated with an account, either because it has expired or because the user wishes to change it. It provides a single primitive:

- MAN.PAM.CHAUTHOK.3 changes the authentication token, optionally verifying that it is sufficiently hard to guess, has not been used previously, etc.

64.3.2 Modules

Modules are a very central concept in PAM; after all, they are the “M” in “PAM”. A PAM module is a self-contained piece of program code that implements the primitives in one or more facilities for one particular mechanism; possible mechanisms for the authentication facility, for instance, include the UNIX password database, NIS, LDAP and Radius.

Module Naming

FreeBSD implements each mechanism in a single module, named `pam_mechanism.so` (for instance, `pam_unix.so` for the UNIX mechanism.) Other implementations sometimes have separate modules for separate facilities, and include the facility name as well as the mechanism name in the module name. To name one example, SOLARIS has a `pam_dial_auth.so.1` module which is commonly used to authenticate dialup users.

Module Versioning

FreeBSD's original PAM implementation, based on Linux-PAM, did not use version numbers for PAM modules. This would commonly cause problems with legacy applications, which might be linked against older versions of the system libraries, as there was no way to load a matching version of the required modules.

OpenPAM, on the other hand, looks for modules that have the same version number as the PAM library (currently 2), and only falls back to an unversioned module if no versioned module could be loaded. Thus legacy modules can be provided for legacy applications, while allowing new (or newly built) applications to take advantage of the most recent modules.

Although SOLARIS PAM modules commonly have a version number, they are not truly versioned, because the number is a part of the module name and must be included in the configuration.

64.3.3 Chains and policies

When a server initiates a PAM transaction, the PAM library tries to load a policy for the service specified in the MAN.PAM.START.3 call. The policy specifies how authentication requests should be processed, and is defined in a configuration file. This is the other central concept in PAM: the possibility for the admin to tune the system security policy (in the wider sense of the word) simply by editing a text file.

A policy consists of four chains, one for each of the four PAM facilities. Each chain is a sequence of configuration statements, each specifying a module to invoke, some (optional) parameters to pass to the module, and a control flag that describes how to interpret the return code from the module.

Understanding the control flags is essential to understanding PAM configuration files. There are four different control flags:

binding If the module succeeds and no earlier module in the chain has failed, the chain is immediately terminated and the request is granted. If the module fails, the rest of the chain is executed, but the request is ultimately denied.

This control flag was introduced by Sun in SOLARIS 9 (SUNOS 5.9), and is also supported by OpenPAM.

required If the module succeeds, the rest of the chain is executed, and the request is granted unless some other module fails. If the module fails, the rest of the chain is also executed, but the request is ultimately denied.

requisite If the module succeeds, the rest of the chain is executed, and the request is granted unless some other module fails. If the module fails, the chain is immediately terminated and the request is denied.

sufficient If the module succeeds and no earlier module in the chain has failed, the chain is immediately terminated and the request is granted. If the module fails, the module is ignored and the rest of the chain is executed.

As the semantics of this flag may be somewhat confusing, especially when it is used for the last module in a chain, it is recommended that the `binding` control flag be used instead if the implementation supports it.

optional The module is executed, but its result is ignored. If all modules in a chain are marked `optional`, all requests will always be granted.

When a server invokes one of the six PAM primitives, PAM retrieves the chain for the facility the primitive belongs to, and invokes each of the modules listed in the chain, in the order they are listed, until it reaches the end, or determines that no further processing is necessary (either because a `binding` or `sufficient` module succeeded, or because a `requisite` module failed.) The request is granted if and only if at least one module was invoked, and all non-optional modules succeeded.

Note that it is possible, though not very common, to have the same module listed several times in the same chain. For instance, a module that looks up user names and passwords in a directory server could be invoked multiple times with different parameters specifying different directory servers to contact. PAM treat different occurrences of the same module in the same chain as different, unrelated modules.

64.3.4 Transactions

The lifecycle of a typical PAM transaction is described below. Note that if any of these steps fails, the server should report a suitable error message to the client and abort the transaction.

1. If necessary, the server obtains arbitrator credentials through a mechanism independent of PAM—most commonly by virtue of having been started by `root`, or of being `setuid root`.
2. The server calls `MAN.PAM.START.3` to initialize the PAM library and specify its service name and the target account, and register a suitable conversation function.
3. The server obtains various information relating to the transaction (such as the applicant's user name and the name of the host the client runs on) and submits it to PAM using `MAN.PAM.SET.ITEM.3`.
4. The server calls `MAN.PAM.AUTHENTICATE.3` to authenticate the applicant.
5. The server calls `MAN.PAM.ACCT.MGMT.3` to verify that the requested account is available and valid. If the password is correct but has expired, `MAN.PAM.ACCT.MGMT.3` will return `PAM_NEW_AUTHTOK_REQD` instead of `PAM_SUCCESS`.
6. If the previous step returned `PAM_NEW_AUTHTOK_REQD`, the server now calls `MAN.PAM.CHAUTHOK.3` to force the client to change the authentication token for the requested account.

7. Now that the applicant has been properly authenticated, the server calls `MAN.PAM.SETCRED.3` to establish the credentials of the requested account. It is able to do this because it acts on behalf of the arbitrator, and holds the arbitrator's credentials.
8. Once the correct credentials have been established, the server calls `MAN.PAM.OPEN.SESSION.3` to set up the session.
9. The server now performs whatever service the client requested—for instance, provide the applicant with a shell.
10. Once the server is done serving the client, it calls `MAN.PAM.CLOSE.SESSION.3` to tear down the session.
11. Finally, the server calls `MAN.PAM.END.3` to notify the PAM library that it is done and that it can release whatever resources it has allocated in the course of the transaction.

64.4 PAM Configuration

64.4.1 PAM policy files

The `/etc/pam.conf` file

The traditional PAM policy file is `/etc/pam.conf`. This file contains all the PAM policies for your system. Each line of the file describes one step in a chain, as shown below:

<code>login</code>	<code>auth</code>	<code>required</code>	<code>pam_nologin.so</code>	<code>no_warn</code>
--------------------	-------------------	-----------------------	-----------------------------	----------------------

The fields are, in order: service name, facility name, control flag, module name, and module arguments. Any additional fields are interpreted as additional module arguments.

A separate chain is constructed for each service / facility pair, so while the order in which lines for the same service and facility appear is significant, the order in which the individual services and facilities are listed is not. The examples in the original PAM paper grouped configuration lines by facility, and the SOLARIS stock `pam.conf` still does that, but FreeBSD's stock configuration groups configuration lines by service. Either way is fine; either way makes equal sense.

The `/etc/pam.d` directory

OpenPAM and Linux-PAM support an alternate configuration mechanism, which is the preferred mechanism in FreeBSD. In this scheme, each policy is contained in a separate file bearing the name of the service it applies to. These files are stored in `/etc/pam.d/`.

These per-service policy files have only four fields instead of `pam.conf`'s five: the service name field is omitted. Thus, instead of the sample `pam.conf` line from the previous section, one would have the following line in `/etc/pam.d/login`:

<code>auth</code>	<code>required</code>	<code>pam_nologin.so</code>	<code>no_warn</code>
-------------------	-----------------------	-----------------------------	----------------------

As a consequence of this simplified syntax, it is possible to use the same policy for multiple services by linking each service name to a same policy file. For instance, to use the same policy for the `su` and `sudo` services, one could do as follows:

<code>PROMPT.ROOT cd /etc/pam.d</code>
<code>PROMPT.ROOT ln -s su sudo</code>

This works because the service name is determined from the file name rather than specified in the policy file, so the same file can be used for multiple differently-named services.

Since each service's policy is stored in a separate file, the `pam.d` mechanism also makes it very easy to install additional policies for third-party software packages.

The policy search order

As we have seen above, PAM policies can be found in a number of places. What happens if policies for the same service exist in multiple places?

It is essential to understand that PAM's configuration system is centered on chains.

64.4.2 Breakdown of a configuration line

As explained in [?](#), each line in `/etc/pam.conf` consists of four or more fields: the service name, the facility name, the control flag, the module name, and zero or more module arguments.

The service name is generally (though not always) the name of the application the statement applies to. If you are unsure, refer to the individual application's documentation to determine what service name it uses.

Note that if you use `/etc/pam.d/` instead of `/etc/pam.conf`, the service name is specified by the name of the policy file, and omitted from the actual configuration lines, which then start with the facility name.

The facility is one of the four facility keywords described in [?](#).

Likewise, the control flag is one of the four keywords described in [?](#), describing how to interpret the return code from the module. Linux-PAM supports an alternate syntax that lets you specify the action to associate with each possible return code, but this should be avoided as it is non-standard and closely tied in with the way Linux-PAM dispatches service calls (which differs greatly from the way SOLARIS and OpenPAM do it.) Unsurprisingly, OpenPAM does not support this syntax.

64.4.3 Policies

To configure PAM correctly, it is essential to understand how policies are interpreted.

When an application calls `MAN.PAM.START.3`, the PAM library loads the policy for the specified service and constructs four module chains (one for each facility.) If one or more of these chains are empty, the corresponding chains from the policy for the `other` service are substituted.

When the application later calls one of the six PAM primitives, the PAM library retrieves the chain for the corresponding facility and calls the appropriate service function in each module listed in the chain, in the order in which they were listed in the configuration. After each call to a service function, the module type and the error code returned by the service function are used to determine what happens next. With a few exceptions, which we discuss below, the following table applies:

	PAM_SUCCESS	PAM_IGNORE	other
binding	if (!fail) break;	•	fail = true;
required	•	•	fail = true;
requisite	•	•	fail = true; break;
sufficient	if (!fail) break;	•	•
optional	•	•	•

Table: PAM chain execution summary

If `fail` is true at the end of a chain, or when a “break” is reached, the dispatcher returns the error code returned by the first module that failed. Otherwise, it returns `PAM_SUCCESS`.

The first exception of note is that the error code `PAM_NEW_AUTHTOK_REQD` is treated like a success, except that if no module failed, and at least one module returned `PAM_NEW_AUTHTOK_REQD`, the dispatcher will return `PAM_NEW_AUTHTOK_REQD`.

The second exception is that `MAN.PAM.SETCRED.3` treats `binding` and `sufficient` modules as if they were `required`.

The third and final exception is that `MAN.PAM.CHAUTHOK.3` runs the entire chain twice (once for preliminary checks and once to actually set the password), and in the preliminary phase it treats `binding` and `sufficient` modules as if they were `required`.

64.5 FreeBSD PAM Modules

64.5.1 MAN.PAM.DENY.8

The `MAN.PAM.DENY.8` module is one of the simplest modules available; it responds to any request with `PAM_AUTH_ERR`. It is useful for quickly disabling a service (add it to the top of every chain), or for terminating chains of `sufficient` modules.

64.5.2 MAN.PAM.ECHO.8

The `MAN.PAM.ECHO.8` module simply passes its arguments to the conversation function as a `PAM_TEXT_INFO` message. It is mostly useful for debugging, but can also serve to display messages such as “Unauthorized access will be prosecuted” before starting the authentication procedure.

64.5.3 MAN.PAM.EXEC.8

The `MAN.PAM.EXEC.8` module takes its first argument to be the name of a program to execute, and the remaining arguments are passed to that program as command-line arguments. One possible application is to use it to run a program at login time which mounts the user’s home directory.

64.5.4 MAN.PAM.FTPUSERS.8

The `MAN.PAM.FTPUSERS.8` module

64.5.5 MAN.PAM.GROUP.8

The `MAN.PAM.GROUP.8` module accepts or rejects applicants on the basis of their membership in a particular file group (normally `wheel` for `MAN.SU.1`). It is primarily intended for maintaining the traditional behaviour of BSD `MAN.SU.1`, but has many other uses, such as excluding certain groups of users from a particular service.

64.5.6 MAN.PAM.GUEST.8

The MAN.PAM.GUEST.8 module allows guest logins using fixed login names. Various requirements can be placed on the password, but the default behaviour is to allow any password as long as the login name is that of a guest account. The MAN.PAM.GUEST.8 module can easily be used to implement anonymous FTP logins.

64.5.7 MAN.PAM.KRB5.8

The MAN.PAM.KRB5.8 module

64.5.8 MAN.PAM.KSU.8

The MAN.PAM.KSU.8 module

64.5.9 MAN.PAM.LASTLOG.8

The MAN.PAM.LASTLOG.8 module

64.5.10 MAN.PAM.LOGIN.ACCESS.8

The MAN.PAM.LOGIN.ACCESS.8 module provides an implementation of the account management primitive which enforces the login restrictions specified in the MAN.LOGIN.ACCESS.5 table.

64.5.11 MAN.PAM.NOLOGIN.8

The MAN.PAM.NOLOGIN.8 module refuses non-root logins when `/var/run/nologin` exists. This file is normally created by MAN.SHUTDOWN.8 when less than five minutes remain until the scheduled shutdown time.

64.5.12 MAN.PAM.OPIE.8

The MAN.PAM.OPIE.8 module implements the MAN.OPIE.4 authentication method. The MAN.OPIE.4 system is a challenge-response mechanism where the response to each challenge is a direct function of the challenge and a passphrase, so the response can be easily computed “just in time” by anyone possessing the passphrase, eliminating the need for password lists. Moreover, since MAN.OPIE.4 never reuses a challenge that has been correctly answered, it is not vulnerable to replay attacks.

64.5.13 MAN.PAM.OPIEACCESS.8

The MAN.PAM.OPIEACCESS.8 module is a companion module to MAN.PAM.OPIE.8. Its purpose is to enforce the restrictions codified in MAN.OPIEACCESS.5, which regulate the conditions under which a user who would normally authenticate herself using MAN.OPIE.4 is allowed to use alternate methods. This is most often used to prohibit the use of password authentication from untrusted hosts.

In order to be effective, the MAN.PAM.OPIEACCESS.8 module must be listed as `requisite` immediately after a `sufficient` entry for MAN.PAM.OPIE.8, and before any other modules, in the `auth` chain.

64.5.14 MAN.PAM.PASSWDQC.8

The MAN.PAM.PASSWDQC.8 module

64.5.15 MAN.PAM.PERMIT.8

The MAN.PAM.PERMIT.8 module is one of the simplest modules available; it responds to any request with `PAM_SUCCESS`. It is useful as a placeholder for services where one or more chains would otherwise be empty.

64.5.16 MAN.PAM.RADIUS.8

The MAN.PAM.RADIUS.8 module

64.5.17 MAN.PAM.RHOSTS.8

The MAN.PAM.RHOSTS.8 module

64.5.18 MAN.PAM.ROOTOK.8

The MAN.PAM.ROOTOK.8 module reports success if and only if the real user id of the process calling it (which is assumed to be run by the applicant) is 0. This is useful for non-networked services such as MAN.SU.1 or MAN.PASSWD.1, to which the `root` should have automatic access.

64.5.19 MAN.PAM.SECURETTY.8

The MAN.PAM.SECURETTY.8 module

64.5.20 MAN.PAM.SELF.8

The MAN.PAM.SELF.8 module reports success if and only if the names of the applicant matches that of the target account. It is most useful for non-networked services such as MAN.SU.1, where the identity of the applicant can be easily verified.

64.5.21 MAN.PAM.SSH.8

The MAN.PAM.SSH.8 module provides both authentication and session services. The authentication service allows users who have passphrase-protected SSH secret keys in their `~/ .ssh` directory to authenticate themselves by typing their passphrase. The session service starts MAN.SSH-AGENT.1 and preloads it with the keys that were decrypted in the authentication phase. This feature is particularly useful for local logins, whether in X (using MAN.XDM.1 or another PAM-aware X login manager) or at the console.

64.5.22 MAN.PAM.TACPLUS.8

The MAN.PAM.TACPLUS.8 module

64.5.23 MAN.PAM.UNIX.8

The MAN.PAM.UNIX.8 module implements traditional UNIX password authentication, using MAN.GETPWNAM.3 to obtain the target account's password and compare it with the one provided by the applicant. It also provides account management services (enforcing account and password expiration times) and password-changing services. This is probably the single most useful module, as the great majority of admins will want to maintain historical behaviour for at least some services.

64.6 PAM Application Programming

This section has not yet been written.

64.7 PAM Module Programming

This section has not yet been written.

64.8 Sample PAM Application

The following is a minimal implementation of MAN.SU.1 using PAM. Note that it uses the OpenPAM-specific MAN.OPENPAM.TTYCONV.3 conversation function, which is prototyped in `security/openpam.h`. If you wish build this application on a system with a different PAM library, you will have to provide your own conversation function. A robust conversation function is surprisingly difficult to implement; the one presented in ? is a good starting point, but should not be used in real-world applications.

64.9 Sample PAM Module

The following is a minimal implementation of MAN.PAM.UNIX.8, offering only authentication services. It should build and run with most PAM implementations, but takes advantage of OpenPAM extensions if available: note the use of MAN.PAM.GET.AUTHTOK.3, which enormously simplifies prompting the user for a password.

64.10 Sample PAM Conversation Function

The conversation function presented below is a greatly simplified version of OpenPAM's MAN.OPENPAM.TTYCONV.3. It is fully functional, and should give the reader a good idea of how a conversation function should behave, but it is far too simple for real-world use. Even if you are not using OpenPAM, feel free to download the source code and adapt MAN.OPENPAM.TTYCONV.3 to your uses; we believe it to be as robust as a tty-oriented conversation function can reasonably get.

64.11 Further Reading

64.12 Papers

Making Login Services Independent of Authentication Technologies SamarVipin LaiCharlie Sun Microsystems

[X/Open Single Sign-on Preliminary Specification](#) The Open Group 1-85912-144-6 June 1997

[Pluggable Authentication Modules](#) MorganAndrewG. 1999-10-06

64.13 User Manuals

[PAM Administration](#) Sun Microsystems

64.14 Related Web pages

[OpenPAM homepage](#) SmørgravDag-Erling ThinkSec AS

[Linux-PAM homepage](#) MorganAndrewG.

[Solaris PAM homepage](#) Sun Microsystems

OpenPGP Keys

These OpenPGP keys can be used to verify a signature or send encrypted email to FreeBSD.org officers or developers. The complete keyring can be downloaded at <http://www.FreeBSD.org/doc/pgpkeyring.txt>.

65.1 Officers

SECTION.PGPKEYS-OFFICERS Core Team Members =====

SECTION.PGPKEYS-CORE Developers =====

SECTION.PGPKEYS-DEVELOPERS Other Cluster Account Holders =====

SECTION.PGPKEYS-OTHER

Port Mentor Guidelines

Author The OS Ports Management Team

66.1 Guideline for Mentor/Mentee Relationships

This section is intended to help demystify the mentoring process, as well as a way to openly promote a constructive discussion to adapt and grow the guidelines. In our lives we have too many rules; we are not a government organization that inflicts regulation, but rather a collective of like minded individuals working toward a common goal, maintaining the quality assurance of the product we call the Ports Tree.

66.1.1 Why Mentor?

- For most of us, we were mentored into the Project, so return the favor by offering to mentor somebody else in.
- You have an irresistible urge to inflict knowledge on others.
- The usual punishment applies because you are sick and tired of committing somebody else's good work!

66.1.2 Mentor/Co-Mentor

Reasons for a co-mentorship:

- Significant timezone differential. Accessible, interactive mentor(s) available via IM is extremely helpful!
- Potential language barrier. Yes, OS is very English oriented, as is most software development, however, having a mentor who can speak a native language can be very useful.
- ENOTIME! Until there is a 30 hour day, and an 8 day week, some of us only have so much time to give. Sharing the load with somebody else will make it easier.
- A rookie mentor can benefit from the experience of a senior committer/mentor.
- Two heads are better than one.

Reasons for sole mentorship:

- You do not play nicely with others.
- You prefer to have a one-on-one relationship.
- The reasons for co-mentorship do not apply to you.

66.1.3 Expectations

We expect mentors to review and test-build all proposed patches, at least for an initial period lasting more than a week or two.

We expect that mentors should take responsibility for the actions of their mentee. A mentor should follow up with all commits the mentee makes, both approved and implicit.

We expect mentors to make sure their mentees read the Porter's Handbook, the PR handling guide, and the Committer's Guide. While it is not necessary to memorize all the details, every committer needs to have an overview of these things to be an effective part of the community (and avoid as many rookie mistakes as possible).

66.1.4 Selecting a Mentee

There is no defined rule for what makes a candidate ready; it can be a combination of number of PRs they have submitted, the number of ports maintained, frequency of ports updates and/or level of participation in a particular area of interest like GNOME, KDE, Gecko or others.

A candidate should have almost no timeouts, be responsive to requests, and generally helpful in supporting their ports.

There must be a history of commitment, as it is widely understood that training a committer requires time and effort. If somebody has been around longer, and spent the time observing how things are done, there is some anticipation of accumulated knowledge. All too often we have seen a maintainer submit a few PRs, show up in IRC and ask when they will be given a commit bit.

Being subscribed to, and following the mailing lists is very beneficial. There is no real expectation that submitting posts on the lists will make somebody a committer, but it demonstrates a commitment. Some mails offer insights into the knowledge of a candidate as well how they interact with others. Similarly participating in IRC can give somebody a higher profile.

Ask six different committers how many PRs a maintainer should submit prior to being nominated, and you will get six different answers. Ask those same individuals how long somebody should have been participating, same dilemma. How many ports should they have at a minimum? Now we have a bikeshed! Some things are just hard to quantify, a mentor will just have to use their best judgement, and hope that portmgr agrees.

66.1.5 Mentorship Duration

As the trust level develops and grows, the mentee may be granted "implicit" commit rights. This can include trivial changes to a `Makefile`, `pkg-descr` etc. Similarly, it may include `PORTVERSION` updates that do not include `plist` changes. Other circumstances may be formulated at the discretion of the Mentor. However, during the period of mentorship, a port version bump that affects dependent ports should be checked by a mentor.

Just as we are all varied individuals, each mentee has different learning curves, time commitments, and other influencing factors that will contribute to the time required before they can "fly solo". Empirically, a mentee should be observed for at least 3 months. 90-100 commits is another target that a mentor could use before releasing a mentee. Other factors to consider prior releasing a mentee are the number of mistakes they may have made, QATs received etc. If they are still making rookie mistakes, they still require mentor guidance.

66.1.6 Mentor/Co-Mentor Debate

When a request gets to portmgr, it usually reads as, "I propose 'foo' for a ports commit bit, I will co-mentor with 'bar'". Proposal received, voted, and carried.

The mentor is the primary point of contact or the "first among equals", the co-mentor is the backup.

Some reprobate, whose name shall be withheld, made the [first recorded co-mentor commit](#). Similar co-mentor commits have also been spotted in the src tree. Does this make it right? Does this make it wrong? It seems to be part of the evolution of how things are done.

66.1.7 Expectations

We expect mentees to be prepared for constructive criticism from the community. There's still a lot of "lore" that is not written down. Responding well to constructive criticism is what we hope we are selecting for by first reviewing their existing contributions on IRC and mailing lists.

We warn mentees that some of the criticism they receive may be less "constructive" than others, (whether through language communication problems, or excessive nit-picking), and that dealing with this gracefully is just part of being in a large community. In case of specific problems with specific people, or any questions, we hope that they will approach a portmgr member on IRC or by email.

Problem Report Handling Guidelines

Author Dag-Erling Smørgrav

Author Hiten Pandya

67.1 Introduction

Bugzilla is an issue management system used by the OS Project. As accurate tracking of outstanding software defects is important to FreeBSD's quality, the correct use of the software is essential to the forward progress of the Project.

Access to Bugzilla is available to the entire OS community. In order to maintain consistency within the database and provide a consistent user experience, guidelines have been established covering common aspects of bug management such as presenting followup, handling close requests, and so forth.

67.2 Problem Report Life-cycle

- The Reporter submits a bug report on the website. The bug is in the `Needs Triage` state.
- Jane Random BugBuster confirms that the bug report has sufficient information to be reproducible. If not, she goes back and forth with the reporter to obtain the needed information. At this point the bug is set to the `Open` state.
- Joe Random Committer takes interest in the PR and assigns it to himself, or Jane Random BugBuster decides that Joe is best suited to handle it and assigns it to him. The bug should be set to the “In Discussion” state.
- Joe has a brief exchange with the originator (making sure it all goes into the audit trail) and determines the cause of the problem.
- Joe pulls an all-nighter and whips up a patch that he thinks fixes the problem, and submits it in a follow-up, asking the originator to test it. He then sets the PRs state to `Patch Ready`.
- A couple of iterations later, both Joe and the originator are satisfied with the patch, and Joe commits it to `-CURRENT` (or directly to `-STABLE` if the problem does not exist in `-CURRENT`), making sure to reference the Problem Report in his commit log (and credit the originator if they submitted all or part of the patch) and, if appropriate, start an MFC countdown. The bug is set to the `Needs MFC` state.
- If the patch does not need MFCing, Joe then closes the PR as `Issue Resolved`.

Note

Many PRs are submitted with very little information about the problem, and some are either very complex to solve, or just scratch the surface of a larger problem; in these cases, it is very important to obtain all the necessary information needed to solve the problem. If the problem contained within cannot be solved, or has occurred again, it is necessary to re-open the PR.

67.3 Problem Report State

It is important to update the state of a PR when certain actions are taken. The state should accurately reflect the current state of work on the PR.

When a PR has been worked on and the developer(s) responsible feel comfortable about the fix, they will submit a followup to the PR and change its state to “feedback”. At this point, the originator should evaluate the fix in their context and respond indicating whether the defect has indeed been remedied.

A Problem Report may be in one of the following states:

open Initial state; the problem has been pointed out and it needs reviewing.

analyzed The problem has been reviewed and a solution is being sought.

feedback Further work requires additional information from the originator or the community; possibly information regarding the proposed solution.

patched A patch has been committed, but something (MFC, or maybe confirmation from originator) is still pending.

suspended The problem is not being worked on, due to lack of information or resources. This is a prime candidate for somebody who is looking for a project to take on. If the problem cannot be solved at all, it will be closed, rather than suspended. The documentation project uses “suspended” for “wish-list” items that entail a significant amount of work which no one currently has time for.

closed A problem report is closed when any changes have been integrated, documented, and tested, or when fixing the problem is abandoned.

Note

The “patched” state is directly related to feedback, so you may go directly to “closed” state if the originator cannot test the patch, and it works in your own testing.

67.4 Types of Problem Reports

While handling problem reports, either as a developer who has direct access to the Problem Reports database or as a contributor who browses the database and submits followups with patches, comments, suggestions or change requests, you will come across several different types of PRs.

- *PRs not yet assigned to anyone.*
- *PRs already assigned to someone.*
- *Duplicates of existing PRs.*
- *Stale PRs*
- *Non-Bug PRs*

The following sections describe what each different type of PRs is used for, when a PR belongs to one of these types, and what treatment each different type receives.

67.4.1 Unassigned PRs

When PRs arrive, they are initially assigned to a generic (placeholder) assignee. These are always prepended with `freebsd-`. The exact value for this default depends on the category; in most cases, it corresponds to a specific OS mailing list. Here is the current list, with the most common ones listed first:

Type	Categories	Default Assignee
base system	bin, conf, gnu, kern, misc	freebsd-bugs
architecture-specific	alpha, amd64, arm, i386, ia64, powerpc, sparc64	freebsd-arch
ports collection	ports	freebsd-ports-bugs
documentation shipped with the system	docs	freebsd-doc
OS web pages (not including docs)	Website	freebsd-www

Table: Default Assignees — most common

Type	Categories	Default Assignee
advocacy efforts	advocacy	freebsd-advocacy
JAVA.VIRTUAL.MACHINE problems	java	freebsd-java
standards compliance	standards	freebsd-standards
threading libraries	threads	freebsd-threads
MAN.USB.4 subsystem	usb	freebsd-usb

Table: Default Assignees — other

Do not be surprised to find that the submitter of the PR has assigned it to the wrong category. If you fix the category, do not forget to fix the assignment as well. (In particular, our submitters seem to have a hard time understanding that just because their problem manifested on an i386 system, that it might be generic to all of OS, and thus be more appropriate for `kern`. The converse is also true, of course.)

Certain PRs may be reassigned away from these generic assignees by anyone. There are several types of assignees: specialized mailing lists; mail aliases (used for certain limited-interest items); and individuals.

For assignees which are mailing lists, please use the long form when making the assignment (e.g., `freebsd-foo` instead of `f00`); this will avoid duplicate emails sent to the mailing list.

Note

Since the list of individuals who have volunteered to be the default assignee for certain types of PRs changes so often, it is much more suitable for the [FreeBSD wiki](#).

Here is a sample list of such entities; it is probably not complete.

Type	Suggested Category	Suggested Assignee	Assignee Type
problem specific to the ARM architecture	arm	freebsd-arm	mailing list
problem specific to the MIPS architecture	kern	freebsd-mips	mailing list
problem specific to the POWERPC architecture	kern	freebsd-ppc	mailing list
problem with Advanced Configuration and Power Management (MAN.ACPI.4)	kern	freebsd-acpi	mailing list
problem with Asynchronous Transfer Mode (ATM) drivers	kern	freebsd-atm	mailing list
problem with embedded or small-footprint OS systems (e.g., NanoBSD/PicoBSD/FreeBSD-arm)	kern	freebsd-embedded	mailing list
problem with FIREWIRE drivers	kern	freebsd-firewire	mailing list
problem with the filesystem code	kern	freebsd-fs	mailing list
problem with the MAN.GEOM.4 subsystem	kern	freebsd-geom	mailing list
problem with the MAN.IPFW.4 subsystem	kern	freebsd-ipfw	mailing list
problem with Integrated Services Digital Network (ISDN) drivers	kern	freebsd-isdn	mailing list
MAN.JAIL.8 subsystem	kern	freebsd-jail	mailing list
problem with LINUX or SVR4 emulation	kern	freebsd-emulation	mailing list
problem with the networking stack	kern	freebsd-net	mailing list
problem with the MAN.PF.4 subsystem	kern	freebsd-pf	mailing list
problem with the MAN.SCSI.4 subsystem	kern	freebsd-scsi	mailing list
problem with the MAN.SOUND.4 subsystem	kern	freebsd-multimedia	mailing list
problems with the MAN.WLAN.4 subsystem and wireless drivers	kern	freebsd-wireless	mailing list
problem with MAN.SYSINSTALL.8 or MAN.BSDINSTALL.8	bin	freebsd-sysinstall	mailing list
problem with the system startup scripts (MAN.RC.8)	kern	freebsd-rc	mailing list
problem with VIMAGE or VNET functionality and related code	kern	freebsd-virtualization	mailing list
problem with Xen emulation	kern	freebsd-xen	mailing list

Table: Common Assignees — base system

Type	Suggested Category	Suggested Assignee	Assignee Type
problem with the ports framework (<i>not</i> with an individual port!)	ports	portmgr	alias
port which is maintained by apache@FreeBSD.org	ports	apache	mailing list
port which is maintained by autotools@FreeBSD.org	ports	autotools	alias
port which is maintained by doceng@FreeBSD.org	ports	doceng	alias
port which is maintained by eclipse@FreeBSD.org	ports	freebsd-eclipse	mailing list
port which is maintained by gecko@FreeBSD.org	ports	gecko	mailing list
port which is maintained by gnome@FreeBSD.org	ports	gnome	mailing list
port which is maintained by hamradio@FreeBSD.org	ports	hamradio	alias
port which is maintained by haskell@FreeBSD.org	ports	haskell	alias
port which is maintained by java@FreeBSD.org	ports	freebsd-java	mailing list
port which is maintained by kde@FreeBSD.org	ports	kde	mailing list
port which is maintained by mono@FreeBSD.org	ports	mono	mailing list
port which is maintained by office@FreeBSD.org	ports	freebsd-office	mailing list
port which is maintained by perl@FreeBSD.org	ports	perl	mailing list
port which is maintained by python@FreeBSD.org	ports	freebsd-python	mailing list
port which is maintained by ruby@FreeBSD.org	ports	freebsd-ruby	mailing list
port which is maintained by secteam@FreeBSD.org	ports	secteam	alias
port which is maintained by vbox@FreeBSD.org	ports	vbox	alias
port which is maintained by x11@FreeBSD.org	ports	freebsd-x11	mailing list

Table: Common Assignees — Ports Collection

Ports PRs which have a maintainer who is a ports committer may be reassigned by anyone (but note that not every OS committer is necessarily a ports committer, so you cannot simply go by the email address alone.)

For other PRs, please do not reassign them to individuals (other than yourself) unless you are certain that the assignee really wants to track the PR. This will help to avoid the case where no one looks at fixing a particular problem because everyone assumes that the assignee is already working on it.

Type	Suggested Category	Suggested Assignee	Assignee Type
problem with PR database	bin	bugmeister	alias
problem with Bugzilla web form .	doc	bugmeister	alias

Table: Common Assignees — Other

67.4.2 Assigned PRs

If a PR has the `responsible` field set to the username of a FreeBSD developer, it means that the PR has been handed over to that particular person for further work.

Assigned PRs should not be touched by anyone but the assignee or bugmeister. If you have comments, submit a followup. If for some reason you think the PR should change state or be reassigned, send a message to the assignee. If the assignee does not respond within two weeks, unassign the PR and do as you please.

67.4.3 Duplicate PRs

If you find more than one PR that describe the same issue, choose the one that contains the largest amount of useful information and close the others, stating clearly the number of the superseding PR. If several PRs contain non-overlapping useful information, submit all the missing information to one in a followup, including references to the others; then close the other PRs (which are now completely superseded).

67.4.4 Stale PRs

A PR is considered stale if it has not been modified in more than six months. Apply the following procedure to deal with stale PRs:

- If the PR contains sufficient detail, try to reproduce the problem in `-CURRENT` and `-STABLE`. If you succeed, submit a followup detailing your findings and try to find someone to assign it to. Set the state to “analyzed” if appropriate.
- If the PR describes an issue which you know is the result of a usage error (incorrect configuration or otherwise), submit a followup explaining what the originator did wrong, then close the PR with the reason “User error” or “Configuration error”.
- If the PR describes an error which you know has been corrected in both `-CURRENT` and `-STABLE`, close it with a message stating when it was fixed in each branch.
- If the PR describes an error which you know has been corrected in `-CURRENT`, but not in `-STABLE`, try to find out when the person who corrected it is planning to MFC it, or try to find someone else (maybe yourself?) to do it. Set the state to “patched” and assign it to whomever will do the MFC.
- In other cases, ask the originator to confirm if the problem still exists in newer versions. If the originator does not reply within a month, close the PR with the notation “Feedback timeout”.

67.4.5 Non-Bug PRs

Developers that come across PRs that look like they should have been posted to `A.BUGS.NAME` or some other list should close the PR, informing the submitter in a comment why this is not really a PR and where the message should be posted.

The email addresses that Bugzilla listens to for incoming PRs have been published as part of the FreeBSD documentation, have been announced and listed on the web-site. This means that spammers found them.

Whenever you close one of these PRs, please do the following:

- Set the component to `junk` (under `Supporting Services`).
- Set `Responsible` to `nobody@FreeBSD.org`.
- Set `State` to `Issue Resolved`.

Setting the category to `junk` makes it obvious that there is no useful content within the PR, and helps to reduce the clutter within the main categories.

67.5 Further Reading

This is a list of resources relevant to the proper writing and processing of problem reports. It is by no means complete.

- [How to Write FreeBSD Problem Reports—guidelines for PR originators.](#)

Writing OS Problem Reports

Author Dag-Erling Smørgrav Contributed by

Author Mark Linimon

problem reports Introduction =====

One of the most frustrating experiences one can have as a software user is to submit a problem report only to have it summarily closed with a terse and unhelpful explanation like “not a bug” or “bogus PR”. Similarly, one of the most frustrating experiences as a software developer is to be flooded with problem reports that are not really problem reports but requests for support, or that contain little or no information about what the problem is and how to reproduce it.

This document attempts to describe how to write good problem reports. What, one asks, is a good problem report? Well, to go straight to the bottom line, a good problem report is one that can be analyzed and dealt with swiftly, to the mutual satisfaction of both user and developer.

Although the primary focus of this article is on OS problem reports, most of it should apply quite well to other software projects.

Note that this article is organized thematically, not chronologically. Read the entire document before submitting a problem report, rather than treating it as a step-by-step tutorial.

68.1 When to Submit a Problem Report

There are many types of problems, and not all of them should engender a problem report. Of course, nobody is perfect, and there will be times when what seems to be a bug in a program is, in fact, a misunderstanding of the syntax for a command or a typographical error in a configuration file (though that in itself may sometimes be indicative of poor documentation or poor error handling in the application). There are still many cases where submitting a problem report is clearly *not* the right course of action, and will only serve to frustrate both the submitter and the developers. Conversely, there are cases where it might be appropriate to submit a problem report about something else than a bug—an enhancement or a new feature, for instance.

So how does one determine what is a bug and what is not? As a simple rule of thumb, the problem is *not* a bug if it can be expressed as a question (usually of the form “How do I do X?” or “Where can I find Y?”). It is not always quite so black and white, but the question rule covers a large majority of cases. When looking for an answer, consider posing the question to the A.QUESTIONS.

Consider these factors when submitting PRs about ports or other software that is not part of OS itself:

- Please do not submit problem reports that simply state that a newer version of an application is available. Ports maintainers are automatically notified by portscout when a new version of an application becomes available. Actual patches to update a port to the latest version are welcome.

- For unmaintained ports (MAINTAINER is `ports@FreeBSD.org`), a PR without an included patch is unlikely to get picked up by a committer. To become the maintainer of an unmaintained port, submit a PR with the request (patch preferred but not required).
- In either case, following the process described in Porter's Handbook will yield the best results. (You might also wish to read *Contributing to the FreeBSD Ports Collection*.)

A bug that cannot be reproduced can rarely be fixed. If the bug only occurred once and you can not reproduce it, and it does not seem to happen to anybody else, chances are none of the developers will be able to reproduce it or figure out what is wrong. That does not mean it did not happen, but it does mean that the chances of your problem report ever leading to a bug fix are very slim. To make matters worse, often these kinds of bugs are actually caused by failing hard drives or overheating processors — you should always try to rule out these causes, whenever possible, before submitting a PR.

Next, to decide to whom you should file your problem report, you need to understand that the software that makes up OS is composed of several different elements:

- Code in the base system that is written and maintained by OS contributors, such as the kernel, the C library, and the device drivers (categorized as `kern`); the binary utilities (`bin`); the manual pages and documentation (`docs`); and the web pages (`www`). All bugs in these areas should be reported to the OS developers.
- Code in the base system that is written and maintained by others, and imported into OS and adapted. Examples include `MAN.CLANG.1`, and `MAN.SENDMAIL.8`. Most bugs in these areas should be reported to the OS developers; but in some cases they may need to be reported to the original authors instead if the problems are not OS-specific.
- Individual applications that are not in the base system but are instead part of the OS Ports Collection (category `ports`). Most of these applications are not written by OS developers; what OS provides is merely a framework for installing the application. Therefore, only report a problem to the OS developers when the problem is believed to be OS-specific; otherwise, report it to the authors of the software.

Then, ascertain whether the problem is timely. There are few things that will annoy a developer more than receiving a problem report about a bug she has already fixed.

If the problem is in the base system, first read the FAQ section on OS versions, if you are not already familiar with the topic. It is not possible for OS to fix problems in anything other than certain recent branches of the base system, so filing a bug report about an older version will probably only result in a developer advising you to upgrade to a supported version to see if the problem still recurs. The Security Officer team maintains the list of supported versions.

If the problem is in a port, note that you must first upgrade to the latest version of the Ports Collection and see if the problem still applies. Due to the rapid pace of changes in these applications, it is infeasible for OS to support anything other than the absolute latest versions, and problems with older version of applications simply cannot be fixed.

68.2 Preparations

A good rule to follow is to always do a background search before submitting a problem report. Maybe the problem has already been reported; maybe it is being discussed on the mailing lists, or recently was; it may even already be fixed in a newer version than what you are running. You should therefore check all the obvious places before submitting your problem report. For OS, this means:

- The OS Frequently Asked Questions (FAQ) list. The FAQ attempts to provide answers for a wide range of questions, such as those concerning hardware compatibility, user applications, and kernel configuration.
- The mailing lists—if you are not subscribed, use [the searchable archives](#) on the OS web site. If the problem has not been discussed on the lists, you might try posting a message about it and waiting a few days to see if someone can spot something that has been overlooked.

- Optionally, the entire web—use your favorite search engine to locate any references to the problem. You may even get hits from archived mailing lists or newsgroups you did not know of or had not thought to search through.
- Next, the searchable [OS PR database](#) (Bugzilla). Unless the problem is recent or obscure, there is a fair chance it has already been reported.
- Most importantly, attempt to see if existing documentation in the source base addresses your problem.

For the base OS code, you should carefully study the contents of `/usr/src/UPDATING` on your system or the latest version at <http://svnweb.freebsd.org/base/head/UPDATING?view=log>. (This is vital information if you are upgrading from one version to another—especially if you are upgrading to the OS.CURRENT branch).

However, if the problem is in something that was installed as a part of the OS Ports Collection, you should refer to `/usr/ports/UPDATING` (for individual ports) or `/usr/ports/CHANGES` (for changes that affect the entire Ports Collection). <http://svnweb.freebsd.org/ports/head/UPDATING?view=log> and <http://svnweb.freebsd.org/ports/head/CHANGES?view=log> are also available via svnweb.

68.3 Writing the Problem Report

Now that you have decided that your issue merits a problem report, and that it is a OS problem, it is time to write the actual problem report. Before we get into the mechanics of the program used to generate and submit PRs, here are some tips and tricks to help make sure that your PR will be most effective.

68.3.1 Tips and Tricks for Writing a Good Problem Report

- *Do not leave the “Synopsis” line empty.* The PRs go both onto a mailing list that goes all over the world (where the “Synopsis” is used for the Subject : line), but also into a database. Anyone who comes along later and browses the database by synopsis, and finds a PR with a blank subject line, tends just to skip over it. Remember that PRs stay in this database until they are closed by someone; an anonymous one will usually just disappear in the noise.
- *Avoid using a weak “Synopsis” line.* You should not assume that anyone reading your PR has any context for your submission, so the more you provide, the better. For instance, what part of the system does the problem apply to? Do you only see the problem while installing, or while running? To illustrate, instead of Synopsis: `portupgrade is broken`, see how much more informative this seems: “Synopsis: `port ports-mgmt/portupgrade coredumps` on `-current`“. (In the case of ports, it is especially helpful to have both the category and portname in the “Synopsis” line.)
- *If you have a patch, say so.* A PR with a patch included is much more likely to be looked at than one without. If you are including one, put the string `[patch]` (including the brackets) at the beginning of the “Synopsis”. (Although it is not mandatory to use that exact string, by convention, that is the one that is used.)
- *If you are a maintainer, say so.* If you are maintaining a part of the source code (for instance, a port), you might consider adding the string `[maintainer update]` (including the brackets) at the beginning of your synopsis line, and you definitely should set the “Class” of your PR to `maintainer-update`. This way any committer that handles your PR will not have to check.
- *Be specific.* The more information you supply about what problem you are having, the better your chance of getting a response.
 - Include the version of OS you are running (there is a place to put that, see below) and on which architecture. You should include whether you are running from a release (e.g., from a CD-ROM or download), or from a system maintained by Subversion (and, if so, what revision number you are at). If you are tracking the

OS.CURRENT branch, that is the very first thing someone will ask, because fixes (especially for high-profile problems) tend to get committed very quickly, and OS.CURRENT users are expected to keep up.

- Include which global options you have specified in your `make.conf`. Note: specifying `-O2` and above to `MAN.GCC.1` is known to be buggy in many situations. While the OS developers will accept patches, they are generally unwilling to investigate such issues due to simple lack of time and volunteers, and may instead respond that this just is not supported.
- If the problem can be reproduced easily, include information that will help a developer to reproduce it themselves. If a problem can be demonstrated with specific input then include an example of that input if possible, and include both the actual and the expected output. If this data is large or cannot be made public, then do try to create a minimal file that exhibits the same issue and that can be included within the PR.
- If this is a kernel problem, then be prepared to supply the following information. (You do not have to include these by default, which only tends to fill up the database, but you should include excerpts that you think might be relevant):
 - * your kernel configuration (including which hardware devices you have installed)
 - * whether or not you have debugging options enabled (such as `WITNESS`), and if so, whether the problem persists when you change the sense of that option
 - * the full text of any backtrace, panic or other console output, or entries in `/var/log/messages`, if any were generated
 - * the output of `pciconf -l` and relevant parts of your `dmesg` output if your problem relates to a specific piece of hardware
 - * the fact that you have read `src/UPDATING` and that your problem is not listed there (someone is guaranteed to ask)
 - * whether or not you can run any other kernel as a fallback (this is to rule out hardware-related issues such as failing disks and overheating CPUs, which can masquerade as kernel problems)
- If this is a ports problem, then be prepared to supply the following information. (You do not have to include these by default, which only tends to fill up the database, but you should include excerpts that you think might be relevant):
 - * which ports you have installed
 - * any environment variables that override the defaults in `bsd.port.mk`, such as `PORTSDIR`
 - * the fact that you have read `ports/UPDATING` and that your problem is not listed there (someone is guaranteed to ask)
- *Avoid vague requests for features.* PRs of the form “someone should really implement something that does so-and-so” are less likely to get results than very specific requests. Remember, the source is available to everyone, so if you want a feature, the best way to ensure it being included is to get to work! Also consider the fact that many things like this would make a better topic for discussion on `freebsd-questions` than an entry in the PR database, as discussed above.
- *Make sure no one else has already submitted a similar PR.* Although this has already been mentioned above, it bears repeating here. It only take a minute or two to use the web-based search engine at <https://bugs.freebsd.org/bugzilla/query.cgi>. (Of course, everyone is guilty of forgetting to do this now and then.)
- *Report only one issue per Problem Report.* Avoid including two or more problems within the same report unless they are related. When submitting patches, avoid adding multiple features or fixing multiple bugs in the same PR unless they are closely related—such PRs often take longer to resolve.
- *Avoid controversial requests.* If your PR addresses an area that has been controversial in the past, you should probably be prepared to not only offer patches, but also justification for why the patches are

“The Right Thing To Do”. As noted above, a careful search of the mailing lists using the archives at <http://www.FreeBSD.org/search/search.html#mailinglists> is always good preparation.

- *Be polite.* Almost anyone who would potentially work on your PR is a volunteer. No one likes to be told that they have to do something when they are already doing it for some motivation other than monetary gain. This is a good thing to keep in mind at all times on Open Source projects.

68.3.2 Before Beginning

Similar considerations apply to use of the [web-based PR submission form](#). Be careful of cut-and-paste operations that might change whitespace or other text formatting.

Finally, if the submission is lengthy, prepare the work offline so that nothing will be lost if there is a problem submitting it.

68.3.3 Attaching Patches or Files

When attaching a patch, be sure to use `-u` with `MAN.DIFF.1` to create or unified diff and make sure to specify the exact SVN revision numbers of the files you modified so the developers who read your report will be able to apply them easily. For problems with the kernel or the base utilities, a patch against `OS.CURRENT` (the `HEAD` Subversion branch) is preferred since all new code should be applied and tested there first. After appropriate or substantial testing has been done, the code will be merged/migrated to the `OS.STABLE` branch.

If you attach a patch inline, instead of as an attachment, note that the most common problem by far is the tendency of some email programs to render tabs as spaces, which will completely ruin anything intended to be part of a `Makefile`.

Do not send patches as attachments using “Content-Transfer-Encoding: quoted-printable”. These will perform character escaping and the

entire patch will be useless.

Also note that while including small patches in a PR is generally all right—particularly when they fix the problem described in the PR—large patches and especially new code which may require substantial review before committing should be placed on a web or ftp server, and the URL should be included in the PR instead of the patch. Patches in email tend to get mangled, and the larger the patch, the harder it will be for interested parties to unmangle it. Also, posting a patch on the web allows you to modify it without having to resubmit the entire patch in a followup to the original PR. Finally, large patches simply increase the size of the database, since closed PRs are not actually deleted but instead kept and simply marked as complete.

You should also take note that unless you explicitly specify otherwise in your PR or in the patch itself, any patches you submit will be assumed to be licensed under the same terms as the original file you modified.

68.3.4 Filling out the Template

In the email template only, you will find the following single-line fields:

- *Submitter-Id:* Do not change this. The default value of `current-users` is correct, even if you run `OS.STABLE`.
- *Confidential:* This is prefilled to `no`. Changing it makes no sense as there is no such thing as a confidential OS problem report—the PR database is distributed worldwide.
- *Severity:* One of `non-critical`, `serious` or `critical`. Do not overreact; refrain from labeling your problem `critical` unless it really is (e.g., data corruption issues, serious regression from previous functionality in `-CURRENT`) or `serious` unless it is something that will affect many users (kernel panics or freezes; problems with particular device drivers or system utilities). OS developers will not necessarily work on your

problem faster if you inflate its importance since there are so many other people who have done exactly that — in fact, some developers pay little attention to this field because of this.

- *Priority*: This field indicates how widespread the effects of this bug is likely to be.

The next section describes fields that are common to both the email interface and the [web interface](#):

- *Originator*: Please specify your real name, optionally followed by your email address in angle brackets. In the email interface, this is normally prefilled with the `gecos` field of the currently logged-in user.

Note

The email address you use will become public information and may become available to spammers. You should either have spam handling procedures in place, or use a temporary email account. However, please note that if you do not use a valid email account at all, we will not be able to ask you questions about your PR.

- *Organization*: Whatever you feel like. This field is not used for anything significant.
- *Synopsis*: Fill this out with a short and accurate description of the problem. The synopsis is used as the subject of the problem report email, and is used in problem report listings and summaries; problem reports with obscure synopses tend to get ignored.

As noted above, if your problem report includes a patch, please have the synopsis start with `[patch]` (including the brackets); if this is a ports PR and you are the maintainer, you may consider adding `[maintainer update]` (including the brackets) and set the “Class” of your PR to `maintainer-update`.

- *Category*: Choose an appropriate category.

The first thing you need to do is to decide what part of the system your problem lies in. Remember, OS is a complete operating system, which installs both a kernel, the standard libraries, many peripheral drivers, and a large number of utilities (the “base system”). However, there are thousands of additional applications in the Ports Collection. You’ll first need to decide if the problem is in the base system or something installed via the Ports Collection.

Here is a description of the major categories:

- If a problem is with the kernel, the libraries (such as standard C library `libc`), or a peripheral driver in the base system, in general you will use the `kern` category. (There are a few exceptions; see below). In general these are things that are described in section 2, 3, or 4 of the manual pages.
- If a problem is with a binary program such as `MAN.SH.1` or `MAN.MOUNT.8`, you will first need to determine whether these programs are in the base system or were added via the Ports Collection. If you are unsure, you can do “`whereis`

`programname`“. OS’s convention for the Ports Collection

is to install everything underneath `/usr/local`, although this can be overridden by a system administrator. For these, you will use the `ports` category (yes, even if the port’s category is `www`; see below). If the location is `/bin`, `/usr/bin`, `/sbin`, or `/usr/sbin`, it is part of the base system, and you should use the `bin` category. (A few programs, such as `MAN.GCC.1`, actually use the `gnu` category, but do not worry about that for now.) These are all things that are described in section 1 or 8 of the manual pages.

- If you believe that the error is in the startup (`rc`) scripts, or in some kind of other non-executable configuration file, then the right category is `conf` (configuration). These are things that are described in section 5 of the manual pages.
- If you have found a problem in the documentation set (articles, books, man pages), the correct choice is `docs`.
- If you are having a problem with the [FreeBSD web pages](#), the proper choice is `www`.

Note

if you are having a problem with something from a port named `www/someportname`, this nevertheless goes in the `ports` category.

There are a few more specialized categories.

- If the problem would otherwise be filed in `kern` but has to do with the USB subsystem, the correct choice is `usb`.
- If the problem would otherwise be filed in `kern` but has to do with the threading libraries, the correct choice is `threads`.
- If the problem would otherwise be in the base system, but has to do with our adherence to standards such as POSIX, the correct choice is `standards`.
- If the problem has to do with errors internal to a `JAVA.VIRTUAL.MACHINE` (JVM), even though `JAVA` was installed from the Ports Collection, you should select the `java` category. More general problems with `JAVA` ports still go under `ports`.

This leaves everything else.

- If you are convinced that the problem will only occur under the processor architecture you are using, select one of the architecture-specific categories: commonly `i386` for Intel-compatible machines in 32-bit mode; `amd64` for AMD machines running in 64-bit mode (this also includes Intel-compatible machines running in EMT64 mode); and less commonly `arm`, `ia64`, `powerpc`, and `sparc64`.

Note

These categories are quite often misused for “I do not know” problems. Rather than guessing, please just use `misc`.

You have a common PC-based machine, and think you have encountered a problem specific to a particular chipset or a particular motherboard: `i386` is the right category.

You are having a problem with an add-in peripheral card on a commonly seen bus, or a problem with a particular type of hard disk drive: in this case, it probably applies to more than one architecture, and `kern` is the right category.

- If you really do not know where the problem lies (or the explanation does not seem to fit into the ones above), use the `misc` category. Before you do so, you may wish to ask for help on the `A.QUESTIONS` first. You may be advised that one of the existing categories really is a better choice.

Here is the current list of categories (taken from <http://svnweb.freebsd.org/base/head/gnu/usr.bin/send-pr/categories>):

- `advocacy`: problems relating to OS’s public image. Obsolete.
- `amd64`: problems specific to the AMD64 platform.
- `arm`: problems specific to the ARM platform.
- `bin`: problems with userland programs in the base system.
- `conf`: problems with configuration files, default values, and so forth.
- `docs`: problems with manual pages or on-line documentation.
- `gnu`: problems with imported GNU software such as `MAN.GCC.1` or `MAN.GREP.1`.
- `i386`: problems specific to the I386 platform.
- `ia64`: problems specific to the ia64 platform.
- `java`: problems related to the `JAVA` Virtual Machine.

- `kern`: problems with the kernel, (non-platform-specific) device drivers, or the base libraries.
- `misc`: anything that does not fit in any of the other categories. (Note that there is almost nothing that truly belongs in this category, except for problems with the release and build infrastructure. Temporary build failures on `HEAD` do not belong here. Also note that it is easy for things to get lost in this category).
- `ports`: problems relating to the Ports Collection.
- `powerpc`: problems specific to the POWERPC platform.
- `sparc64`: problems specific to the SPARC64 platform.
- `standards`: Standards conformance issues.
- `threads`: problems related to the OS threads implementation (especially on `OS.CURRENT`).
- `usb`: problems related to the OS USB implementation.
- `www`: Changes or enhancements to the OS website.
- *Class*: Choose one of the following:
 - `sw-bug`: software bugs.
 - `doc-bug`: errors in documentation.
 - `change-request`: requests for additional features or changes in existing features.
 - `update`: updates to ports or other contributed software.
 - `maintainer-update`: updates to ports for which you are the maintainer.
- *Release*: The version of OS that you are running. This needs to be filled in.

Finally, there is a series of multi-line fields:

- *Environment*: This should describe, as accurately as possible, the environment in which the problem has been observed. This includes the operating system version, the version of the specific program or file that contains the problem, and any other relevant items such as system configuration, other installed software that influences the problem, etc.—quite simply everything a developer needs to know to reconstruct the environment in which the problem occurs.
- *Description*: A complete and accurate description of the problem you are experiencing. Try to avoid speculating about the causes of the problem unless you are certain that you are on the right track, as it may mislead a developer into making incorrect assumptions about the problem.
- *How-To-Repeat*: A summary of the actions you need to take to reproduce the problem.
- *Fix*: Preferably a patch, or at least a workaround (which not only helps other people with the same problem work around it, but may also help a developer understand the cause for the problem), but if you do not have any firm ideas for either, it is better to leave this field blank than to speculate.

68.3.5 Sending the Problem Report

If you are using the [web form](#):

Before you hit `submit`, you will need to fill in a field containing text that is represented in image form on the page. This unfortunate measure has had to be adopted due to misuse by automated systems and a few misguided individuals. It is a necessary evil that no one likes; please do not ask us to remove it.

Note that you are `strongly` advised to save your work somewhere before hitting `submit`. A common problem for users is to have their web browser displaying a stale image from its cache. If this happens to you, your submission will be rejected and you may lose your work.

If you are unable to view images for any reason, please accept our apologies for the inconvenience and email your problem report to the bugbuster team at freebsd-bugbusters@FreeBSD.org.

68.4 Follow-up

Once the problem report has been filed, you will receive a confirmation by email which will include the tracking number that was assigned to your problem report and a URL you can use to check its status. With a little luck, someone will take an interest in your problem and try to address it, or, as the case may be, explain why it is not a problem. You will be automatically notified of any change of status, and you will receive copies of any comments or patches someone may attach to your problem report's audit trail.

If someone requests additional information from you, or you remember or discover something you did not mention in the initial report, please submit a follow up. The number one reason for a bug not getting fixed is lack of communication with the originator.

- The easiest way is to use the comment option on the individual PR's web page, which you can reach from the [PR search page](#).

If the problem report remains open after the problem has gone away, just add a comment saying that the problem report can be closed, and, if possible, explaining how or when the problem was fixed.

Sometimes there is a delay of a week or two where the problem report remains untouched, not assigned or commented on by anyone. This can happen when there is an increased problem report backlog or during a holiday season. When a problem report has not received attention after several weeks, it is worth finding a committer particularly interested in working on it.

There are a few ways to do so, ideally in the following order, with a few days between attempting each communication channel:

- Find the relevant OS mailing list for the problem report from the list in the Handbook and send a message to that list asking about assistance or comments on the problem report.
- Join the relevant IRC channels. A partial listing is here: <https://wiki.freebsd.org/IrcChannels>. Inform the people in that channel about the problem report and ask for assistance. Be patient and stay in the channel after posting, so that the people from different time zones around the world have a chance to catch up.
- Find committers interested in the problem that was reported. If the problem was in a particular tool, binary, port, document, or source file, check the [SVN Repository](#). Locate the last few committers who made substantive changes to the file, and try to reach them via IRC or email. A list of committers and their emails can be found in the Contributors to OS article.

Remember that these people are volunteers, just like maintainers and users, so they might not be immediately available to assist with the problem report. Patience and consistency in the follow-ups is highly advised and appreciated. With enough care and effort dedicated to that follow-up process, finding a committer to take care of the problem report is just a matter of time.

68.5 If There Are Problems

If you found an issue with the bug system, file a bug! There is a category for exactly this purpose. If you are unable to do so, contact the bug wranglers at bugmeister@FreeBSD.org.

68.6 Further Reading

This is a list of resources relevant to the proper writing and processing of problem reports. It is by no means complete.

- [How to Report Bugs Effectively](#)—an excellent essay by Simon G. Tatham on composing useful (non-OS-specific) problem reports.
- Problem Report Handling Guidelines—valuable insight into how problem reports are handled by the OS developers.

Practical rc.d scripting in BSD

Author YarTikhiy

69.1 Introduction

The historical BSD had a monolithic startup script, `/etc/rc`. It was invoked by `MAN.INIT.8` at system boot time and performed all userland tasks required for multi-user operation: checking and mounting file systems, setting up the network, starting daemons, and so on. The precise list of tasks was not the same in every system; admins needed to customize it. With few exceptions, `/etc/rc` had to be modified, and true hackers liked it.

The real problem with the monolithic approach was that it provided no control over the individual components started from `/etc/rc`. For instance, `/etc/rc` could not restart a single daemon. The system admin had to find the daemon process by hand, kill it, wait until it actually exited, then browse through `/etc/rc` for the flags, and finally type the full command line to start the daemon again. The task would become even more difficult and prone to errors if the service to restart consisted of more than one daemon or demanded additional actions. In a few words, the single script failed to fulfil what scripts are for: to make the system admin's life easier.

Later there was an attempt to split out some parts of `/etc/rc` for the sake of starting the most important subsystems separately. The notorious example was `/etc/netstart` to bring up networking. It did allow for accessing the network from single-user mode, but it did not integrate well into the automatic startup process because parts of its code needed to interleave with actions essentially unrelated to networking. That was why `/etc/netstart` mutated into `/etc/rc.network`. The latter was no longer an ordinary script; it comprised of large, tangled `MAN.SH.1` functions called from `/etc/rc` at different stages of system startup. However, as the startup tasks grew diverse and sophisticated, the “quasi-modular” approach became even more of a drag than the monolithic `/etc/rc` had been.

Without a clean and well-designed framework, the startup scripts had to bend over backwards to satisfy the needs of rapidly developing BSD-based operating systems. It became obvious at last that more steps are necessary on the way to a fine-grained and extensible `rc` system. Thus BSD `rc.d` was born. Its acknowledged fathers were Luke Mewburn and the NetBSD community. Later it was imported into OS. Its name refers to the location of system scripts for individual services, which is in `/etc/rc.d`. Soon we will learn about more components of the `rc.d` system and see how the individual scripts are invoked.

The basic ideas behind BSD `rc.d` are *fine modularity* and *code reuse*. *Fine modularity* means that each basic “service” such as a system daemon or primitive startup task gets its own `MAN.SH.1` script able to start the service, stop it, reload it, check its status. A particular action is chosen by the command-line argument to the script. The `/etc/rc` script still drives system startup, but now it merely invokes the smaller scripts one by one with the `start` argument. It is easy to perform shutdown tasks as well by running the same set of scripts with the `stop` argument, which is done by `/etc/rc.shutdown`. Note how closely this follows the Unix way of having a set of small specialized tools, each fulfilling its task as well as possible. *Code reuse* means that common operations are implemented as `MAN.SH.1` functions and collected in `/etc/rc.subr`. Now a typical script can be just a few lines' worth of `MAN.SH.1` code. Finally, an important part of the `rc.d` framework is `MAN.RCORDER.8`, which helps `/etc/rc` to run the small

scripts orderly with respect to dependencies between them. It can help `/etc/rc.shutdown`, too, because the proper order for the shutdown sequence is opposite to that of startup.

The BSD `rc.d` design is described in *the original article by Luke Mewburn*, and the `rc.d` components are documented in great detail in *the respective manual pages*. However, it might not appear obvious to an `rc.d` newbie how to tie the numerous bits and pieces together in order to create a well-styled script for a particular task. Therefore this article will try a different approach to describe `rc.d`. It will show which features should be used in a number of typical cases, and why. Note that this is not a how-to document because our aim is not at giving ready-made recipes, but at showing a few easy entrances into the `rc.d` realm. Neither is this article a replacement for the relevant manual pages. Do not hesitate to refer to them for more formal and complete documentation while reading this article.

There are prerequisites to understanding this article. First of all, you should be familiar with the MAN.SH.1 scripting language in order to master `rc.d`. In addition, you should know how the system performs userland startup and shutdown tasks, which is described in MAN.RC.8.

This article focuses on the OS branch of `rc.d`. Nevertheless, it may be useful to NetBSD developers, too, because the two branches of BSD `rc.d` not only share the same design but also stay similar in their aspects visible to script authors.

69.2 Outlining the task

A little consideration before starting `$EDITOR` will not hurt. In order to write a well-tempered `rc.d` script for a system service, we should be able to answer the following questions first:

- Is the service mandatory or optional?
- Will the script serve a single program, e.g., a daemon, or perform more complex actions?
- Which other services will our service depend on, and vice versa?

From the examples that follow we will see why it is important to know the answers to these questions.

69.3 A dummy script

The following script just emits a message each time the system boots up:

```
#!/bin/sh

. /etc/rc.subr

name="dummy"
start_cmd="${name}_start"
stop_cmd=":"

dummy_start()
{
    echo "Nothing started."
}

load_rc_config $name
run_rc_command "$1"
```

Things to note are:

- An interpreted script should begin with the magic “shebang” line. That line specifies the interpreter program for the script. Due to the shebang line, the script can be invoked exactly like a binary program provided that it has

the execute bit set. (See MAN.CHMOD.1.) For example, a system admin can run our script manually, from the command line:

```
PROMPT.ROOT /etc/rc.d/dummy start
```

****Note****

In order to be properly managed by the ``rc.d`` framework, its scripts need to be written in the MAN.SH.1 language. If you have a service or port that uses a binary control utility or a startup routine written in another language, install that element in ``/usr/sbin`` (for the system) or ``/usr/local/sbin`` (for ports) and call it from a MAN.SH.1 script in the appropriate ``rc.d`` directory.

****Tip****

If you would like to learn the details of why ``rc.d`` scripts must be written in the MAN.SH.1 language, see how ``/etc/rc`` invokes them by means of ``run_rc_script``, then study the implementation of ``run_rc_script`` in ``/etc/rc.subr``.

- In `/etc/rc.subr`, a number of MAN.SH.1 functions are defined for an `rc.d` script to use. The functions are documented in MAN.RC.SUBR.8. While it is theoretically possible to write an `rc.d` script without ever using MAN.RC.SUBR.8, its functions prove extremely handy and make the job an order of magnitude easier. So it is no surprise that everybody resorts to MAN.RC.SUBR.8 in `rc.d` scripts. We are not going to be an exception.

An `rc.d` script must “source” `/etc/rc.subr` (include it using “.”) *before* it calls MAN.RC.SUBR.8 functions so that MAN.SH.1 has an opportunity to learn the functions. The preferred style is to source `/etc/rc.subr` first of all.

Note

Some useful functions related to networking are provided by another include file, `/etc/network.subr`.

- The mandatory variable name specifies the name of our script. It is required by MAN.RC.SUBR.8. That is, each `rc.d` script *must* set name before it calls MAN.RC.SUBR.8 functions.

Now it is the right time to choose a unique name for our script once and for all. We will use it in a number of places while developing the script. For a start, let us give the same name to the script file, too.

Note

The current style of `rc.d` scripting is to enclose values assigned to variables in double quotes. Keep in mind that it is just a style issue that may not always be applicable. You can safely omit quotes from around simple words without MAN.SH.1 metacharacters in them, while in certain cases you will need single quotes to prevent any interpretation of the value by MAN.SH.1. A programmer should be able to tell the language syntax from style conventions and use both of them wisely.

- The main idea behind MAN.RC.SUBR.8 is that an `rc.d` script provides handlers, or methods, for MAN.RC.SUBR.8 to invoke. In particular, `start`, `stop`, and other arguments to an `rc.d` script are handled this way. A method is a MAN.SH.1 expression stored in a variable named `argument_cmd`, where argument corresponds to what can be specified on the script’s command line. We will see later how MAN.RC.SUBR.8 provides default methods for the standard arguments.

Note

To make the code in `rc.d` more uniform, it is common to use `${name}` wherever appropriate. Thus a number of lines can be just copied from one script to another.

- We should keep in mind that MAN.RC.SUBR.8 provides default methods for the standard arguments. Consequently, we must override a standard method with a no-op MAN.SH.1 expression if we want it to do nothing.
- The body of a sophisticated method can be implemented as a function. It is a good idea to make the function name meaningful.

Important

It is strongly recommended to add the prefix `${name}` to the names of all functions defined in our script so they never clash with the functions from MAN.RC.SUBR.8 or another common include file.

- This call to MAN.RC.SUBR.8 loads MAN.RC.CONF.5 variables. Our script makes no use of them yet, but it still is recommended to load MAN.RC.CONF.5 because there can be MAN.RC.CONF.5 variables controlling MAN.RC.SUBR.8 itself.
- Usually this is the last command in an `rc.d` script. It invokes the MAN.RC.SUBR.8 machinery to perform the requested action using the variables and methods our script has provided.

69.4 A configurable dummy script

Now let us add some controls to our dummy script. As you may know, `rc.d` scripts are controlled with MAN.RC.CONF.5. Fortunately, MAN.RC.SUBR.8 hides all the complications from us. The following script uses MAN.RC.CONF.5 via MAN.RC.SUBR.8 to see whether it is enabled in the first place, and to fetch a message to show at boot time. These two tasks in fact are independent. On the one hand, an `rc.d` script can just support enabling and disabling its service. On the other hand, a mandatory `rc.d` script can have configuration variables. We will do both things in the same script though:

```
#!/bin/sh

. /etc/rc.subr

name=dummy
rcvar=dummy_enable

start_cmd="${name}_start"
stop_cmd=":"

load_rc_config $name
: ${dummy_enable:=no}
: ${dummy_msg="Nothing started."}

dummy_start()
{
    echo "$dummy_msg"
}

run_rc_command "$1"
```

What changed in this example?

- The variable `rcvar` specifies the name of the ON/OFF knob variable.
- Now `load_rc_config` is invoked earlier in the script, before any MAN.RC.CONF.5 variables are accessed.

Note

While examining `rc.d` scripts, keep in mind that MAN.SH.1 defers the evaluation of expressions in a function until the latter is called. Therefore it is not an error to invoke `load_rc_config` as late as just before `run_rc_command` and still access MAN.RC.CONF.5 variables from the method

functions exported to `run_rc_command`. This is because the method functions are to be called by `run_rc_command`, which is invoked *after* `load_rc_config`.

- A warning will be emitted by `run_rc_command` if `rcvar` itself is set, but the indicated knob variable is unset. If your `rc.d` script is for the base system, you should add a default setting for the knob to `/etc/defaults/rc.conf` and document it in `MAN.RC.CONF.5`. Otherwise it is your script that should provide a default setting for the knob. The canonical approach to the latter case is shown in the example.

Note

You can make `MAN.RC.SUBR.8` act as though the knob is set to `ON`, irrespective of its current setting, by prefixing the argument to the script with `one` or `force`, as in `onestart` or `forcestop`. Keep in mind though that `force` has other dangerous effects we will touch upon below, while `one` just overrides the `ON/OFF` knob. E.g., assume that `dummy_enable` is `OFF`. The following command will run the `start` method in spite of the setting:

```
PROMPT.ROOT /etc/rc.d/dummy onestart
```

- Now the message to be shown at boot time is no longer hard-coded in the script. It is specified by an `MAN.RC.CONF.5` variable named `dummy_msg`. This is a trivial example of how `MAN.RC.CONF.5` variables can control an `rc.d` script.

Important

The names of all `MAN.RC.CONF.5` variables used exclusively by our script *must* have the same prefix: `$(name)_`. For example: `dummy_mode`, `dummy_state_file`, and so on.

Note

While it is possible to use a shorter name internally, e.g., just `msg`, adding the unique prefix `$(name)_` to all global names introduced by our script will save us from possible collisions with the `MAN.RC.SUBR.8` namespace.

As a rule, `rc.d` scripts of the base system need not provide defaults for their `MAN.RC.CONF.5` variables because the defaults should be set in `/etc/defaults/rc.conf` instead. On the other hand, `rc.d` scripts for ports should provide the defaults as shown in the example.

- Here we use `dummy_msg` to actually control our script, i.e., to emit a variable message. Use of a shell function is overkill here, since it only runs a single command; an equally valid alternative is:

```
start_cmd="echo \"\${dummy_msg}\""
```

69.5 Startup and shutdown of a simple daemon

We said earlier that `MAN.RC.SUBR.8` could provide default methods. Obviously, such defaults cannot be too general. They are suited for the common case of starting and shutting down a simple daemon program. Let us assume now that we need to write an `rc.d` script for such a daemon called `mumbled`. Here it is:

```
#!/bin/sh

. /etc/rc.subr

name=mumbled
rcvar=mumbled_enable

command="/usr/sbin/${name}"

load_rc_config $name
run_rc_command "$1"
```

Pleasingly simple, isn't it? Let us examine our little script. The only new thing to note is as follows:

- The command variable is meaningful to MAN.RC.SUBR.8. If it is set, MAN.RC.SUBR.8 will act according to the scenario of serving a conventional daemon. In particular, the default methods will be provided for such arguments: `start`, `stop`, `restart`, `poll`, and `status`.

The daemon will be started by running `$command` with command-line flags specified by `$mumbled_flags`. Thus all the input data for the default `start` method are available in the variables set by our script. Unlike `start`, other methods may require additional information about the process started. For instance, `stop` must know the PID of the process to terminate it. In the present case, MAN.RC.SUBR.8 will scan through the list of all processes, looking for a process with its name equal to `$procname`. The latter is another variable of meaning to MAN.RC.SUBR.8, and its value defaults to that of `command`. In other words, when we set `command`, `procname` is effectively set to the same value. This enables our script to kill the daemon and to check if it is running in the first place.

Note

Some programs are in fact executable scripts. The system runs such a script by starting its interpreter and passing the name of the script to it as a command-line argument. This is reflected in the list of processes, which can confuse MAN.RC.SUBR.8. You should additionally set `command_interpreter` to let MAN.RC.SUBR.8 know the actual name of the process if `$command` is a script.

For each `rc.d` script, there is an optional MAN.RC.CONF.5 variable that takes precedence over `command`. Its name is constructed as follows: `${name}_program`, where `name` is the mandatory variable we discussed *earlier*. E.g., in this case it will be `mumbled_program`. It is MAN.RC.SUBR.8 that arranges `${name}_program` to override `command`.

Of course, MAN.SH.1 will permit you to set `${name}_program` from MAN.RC.CONF.5 or the script itself even if `command` is unset. In that case, the special properties of `${name}_program` are lost, and it becomes an ordinary variable your script can use for its own purposes. However, the sole use of `${name}_program` is discouraged because using it together with `command` became an idiom of `rc.d` scripting.

For more detailed information on default methods, refer to MAN.RC.SUBR.8.

69.6 Startup and shutdown of an advanced daemon

Let us add some meat onto the bones of the previous script and make it more complex and featureful. The default methods can do a good job for us, but we may need some of their aspects tweaked. Now we will learn how to tune the default methods to our needs.

```
#!/bin/sh

. /etc/rc.subr

name=mumbled
rcvar=mumbled_enable

command="/usr/sbin/${name}"
command_args="mock arguments > /dev/null 2>&1"

pidfile="/var/run/${name}.pid"

required_files="/etc/${name}.conf /usr/share/misc/${name}.rules"

sig_reload="USR1"
```



```

start_precmd="${name}_prestart"
stop_postcmd="echo Bye-bye"

extra_commands="reload plugh xyzzy"

plugh_cmd="mumbled_plugh"
xyzzy_cmd="echo 'Nothing happens.'"

mumbled_prestart()
{
    if checkyesno mumbled_smart; then
        rc_flags="-o smart ${rc_flags}"
    fi
    case "$mumbled_mode" in
        foo)
            rc_flags="-frotz ${rc_flags}"
            ;;
        bar)
            rc_flags="-baz ${rc_flags}"
            ;;
        *)
            warn "Invalid value for mumbled_mode"
            return 1
            ;;
    esac
    run_rc_command xyzzy
    return 0
}

mumbled_plugh()
{
    echo 'A hollow voice says "plugh".'
}

load_rc_config $name
run_rc_command "$1"

```

- Additional arguments to `$command` can be passed in `command_args`. They will be added to the command line after `$mumbled_flags`. Since the final command line is passed to `eval` for its actual execution, input and output redirections can be specified in `command_args`.

Note

Never include dashed options, like `-X` or `--foo`, in `command_args`. The contents of `command_args` will appear at the end of the final command line, hence they are likely to follow arguments present in `${name}_flags`; but most commands will not recognize dashed options after ordinary arguments. A better way of passing additional options to `$command` is to add them to the beginning of `${name}_flags`. Another way is to modify `rc_flags` *as shown later*.

- A good-mannered daemon should create a *pidfile* so that its process can be found more easily and reliably. The variable `pidfile`, if set, tells MAN.RC.SUBR.8 where it can find the pidfile for its default methods to use.

Note

In fact, MAN.RC.SUBR.8 will also use the pidfile to see if the daemon is already running before starting it. This check can be skipped by using the `faststart` argument.

- If the daemon cannot run unless certain files exist, just list them in `required_files`, and MAN.RC.SUBR.8 will check that those files do exist before starting the daemon. There also are `required_dirs` and `required_vars` for directories and environment variables, respectively. They all are described in detail in MAN.RC.SUBR.8.

Note

The default method from MAN.RC.SUBR.8 can be forced to skip the prerequisite checks by using `forrestart` as the argument to the script.

- We can customize signals to send to the daemon in case they differ from the well-known ones. In particular, `sig_reload` specifies the signal that makes the daemon reload its configuration; it is `SIGHUP` by default. Another signal is sent to stop the daemon process; the default is `SIGTERM`, but this can be changed by setting `sig_stop` appropriately.

Note

The signal names should be specified to MAN.RC.SUBR.8 without the `SIG` prefix, as it is shown in the example. The OS version of MAN.KILL.1 can recognize the `SIG` prefix, but the versions from other OS types may not.

- Performing additional tasks before or after the default methods is easy. For each command-argument supported by our script, we can define `argument_precmd` and `argument_postcmd`. These MAN.SH.1 commands are invoked before and after the respective method, as it is evident from their names.

Note

Overriding a default method with a custom `argument_cmd` still does not prevent us from making use of `argument_precmd` or `argument_postcmd` if we need to. In particular, the former is good for checking custom, sophisticated conditions that should be met before performing the command itself. Using `argument_precmd` along with `argument_cmd` lets us logically separate the checks from the action.

Do not forget that you can cram any valid MAN.SH.1 expressions into the methods, pre-, and post-commands you define. Just invoking a function that makes the real job is a good style in most cases, but never let style limit your understanding of what is going on behind the curtain.

- If we would like to implement custom arguments, which can also be thought of as *commands* to our script, we need to list them in `extra_commands` and provide methods to handle them.

Note

The `reload` command is special. On the one hand, it has a preset method in MAN.RC.SUBR.8. On the other hand, `reload` is not offered by default. The reason is that not all daemons use the same reload mechanism and some have nothing to reload at all. So we need to ask explicitly that the builtin functionality be provided. We can do so via `extra_commands`.

What do we get from the default method for `reload`? Quite often daemons reload their configuration upon reception of a signal — typically, `SIGHUP`. Therefore MAN.RC.SUBR.8 attempts to reload the daemon by sending a signal to it. The signal is preset to `SIGHUP` but can be customized via `sig_reload` if necessary.

- Our script supports two non-standard commands, `plugh` and `xyzzzy`. We saw them listed in `extra_commands`, and now it is time to provide methods for them. The method for `xyzzzy` is just inlined while that for `plugh` is implemented as the `mumbled_plugh` function.

Non-standard commands are not invoked during startup or shutdown. Usually they are for the system admin's convenience. They can also be used from other subsystems, e.g., MAN.DEVD.8 if specified in MAN.DEVD.CONF.5.

The full list of available commands can be found in the usage line printed by MAN.RC.SUBR.8 when the script is invoked without arguments. For example, here is the usage line from the script under study:

```
PROMPT.ROOT /etc/rc.d/mumbled
Usage: /etc/rc.d/mumbled [fast|force|one] (start|stop|restart|rcvar|reload|plugh|xyzzzy|status|pol
```

- A script can invoke its own standard or non-standard commands if needed. This may look similar to calling functions, but we know that commands and shell functions are not always the same thing. For instance, `xyzzzy` is not implemented as a function here. In addition, there can be a pre-command and post-command, which should be invoked orderly. So the proper way for a script to run its own command is by means of `MAN.RC.SUBR.8`, as shown in the example.
- A handy function named `checkyesno` is provided by `MAN.RC.SUBR.8`. It takes a variable name as its argument and returns a zero exit code if and only if the variable is set to `YES`, or `TRUE`, or `ON`, or `1`, case insensitive; a non-zero exit code is returned otherwise. In the latter case, the function tests the variable for being set to `NO`, `FALSE`, `OFF`, or `0`, case insensitive; it prints a warning message if the variable contains anything else, i.e., junk.

Keep in mind that for `MAN.SH.1` a zero exit code means true and a non-zero exit code means false.

Important

The `checkyesno` function takes a *variable name*. Do not pass the expanded *value* of a variable to it; it will not work as expected.

The following is the correct usage of `checkyesno`:

```
if checkyesno mumbled_enable; then
    foo
fi
```

On the contrary, calling `checkyesno` as shown below will not work — at least not as expected:

```
if checkyesno "${mumbled_enable}"; then
    foo
fi
```

- We can affect the flags to be passed to `$command` by modifying `rc_flags` in `$start_precmd`.
- In certain cases we may need to emit an important message that should go to `syslog` as well. This can be done easily with the following `MAN.RC.SUBR.8` functions: `debug`, `info`, `warn`, and `err`. The latter function then exits the script with the code specified.
- The exit codes from methods and their pre-commands are not just ignored by default. If `argument_precmd` returns a non-zero exit code, the main method will not be performed. In turn, `argument_postcmd` will not be invoked unless the main method returns a zero exit code.

Note

However, `MAN.RC.SUBR.8` can be instructed from the command line to ignore those exit codes and invoke all commands anyway by prefixing an argument with `force`, as in `forcestart`.

69.7 Connecting a script to the rc.d framework

After a script has been written, it needs to be integrated into `rc.d`. The crucial step is to install the script in `/etc/rc.d` (for the base system) or `/usr/local/etc/rc.d` (for ports). Both `<bsd.prog.mk>` and `<bsd.port.mk>` provide convenient hooks for that, and usually you do not have to worry about the proper ownership and mode. System scripts should be installed from `src/etc/rc.d` through the `Makefile` found there. Port scripts can be installed using `USE_RC_SUBR` as described in the Porter's Handbook.

However, we should consider beforehand the place of our script in the system startup sequence. The service handled by our script is likely to depend on other services. For instance, a network daemon cannot function without the network interfaces and routing up and running. Even if a service seems to demand nothing, it can hardly start before the basic filesystems have been checked and mounted.

We mentioned MAN.RCORDER.8 already. Now it is time to have a close look at it. In a nutshell, MAN.RCORDER.8 takes a set of files, examines their contents, and prints a dependency-ordered list of files from the set to `stdout`. The point is to keep dependency information *inside* the files so that each file can speak for itself only. A file can specify the following information:

- the names of the “conditions” (which means services to us) it *provides*;
- the names of the “conditions” it *requires*;
- the names of the “conditions” this file should run *before*;
- additional *keywords* that can be used to select a subset from the whole set of files (MAN.RCORDER.8 can be instructed via options to include or omit the files having particular keywords listed.)

It is no surprise that MAN.RCORDER.8 can handle only text files with a syntax close to that of MAN.SH.1. That is, special lines understood by MAN.RCORDER.8 look like MAN.SH.1 comments. The syntax of such special lines is rather rigid to simplify their processing. See MAN.RCORDER.8 for details.

Besides using MAN.RCORDER.8 special lines, a script can insist on its dependency upon another service by just starting it forcibly. This can be needed when the other service is optional and will not start by itself because the system admin has disabled it mistakenly in MAN.RC.CONF.5.

With this general knowledge in mind, let us consider the simple daemon script enhanced with dependency stuff:

```
#!/bin/sh

# PROVIDE: mumbled oldmumble
# REQUIRE: DAEMON cleanvar frotz
# BEFORE:  LOGIN
# KEYWORD: nojail shutdown

. /etc/rc.subr

name=mumbled
rcvar=mumbled_enable

command="/usr/sbin/${name}"
start_precmd="${name}_prestart"

mumbled_prestart()
{
    if ! checkyesno frotz_enable && \
        ! /etc/rc.d/frotz forcelstatus 1>/dev/null 2>&1; then
        force_depend frotz || return 1
    fi
    return 0
}

load_rc_config $name
run_rc_command "$1"
```

As before, detailed analysis follows:

- That line declares the names of “conditions” our script provides. Now other scripts can record a dependency on our script by those names.

Note

Usually a script specifies a single condition provided. However, nothing prevents us from listing several conditions there, e.g., for compatibility reasons.

In any case, the name of the main, or the only, `PROVIDE` : condition should be the same as `${name}`.

- So our script indicates which “conditions” provided by other scripts it depends on. According to the lines, our script asks MAN.RCORDER.8 to put it after the script(s) providing DAEMON and cleanvar, but before that providing LOGIN.

Note

The `BEFORE :` line should not be abused to work around an incomplete dependency list in the other script. The appropriate case for using `BEFORE :` is when the other script does not care about ours, but our script can do its task better if run before the other one. A typical real-life example is the network interfaces vs. the firewall: While the interfaces do not depend on the firewall in doing their job, the system security will benefit from the firewall being ready before there is any network traffic.

Besides conditions corresponding to a single service each, there are meta-conditions and their “placeholder” scripts used to ensure that certain groups of operations are performed before others. These are denoted by UPPERCASE names. Their list and purposes can be found in MAN.RC.8.

Keep in mind that putting a service name in the `REQUIRE :` line does not guarantee that the service will actually be running by the time our script starts. The required service may fail to start or just be disabled in MAN.RC.CONF.5. Obviously, MAN.RCORDER.8 cannot track such details, and MAN.RC.8 will not do that either. Consequently, the application started by our script should be able to cope with any required services being unavailable. In certain cases, we can help it as discussed *below*.

- As we remember from the above text, MAN.RCORDER.8 keywords can be used to select or leave out some scripts. Namely any MAN.RCORDER.8 consumer can specify through `-k` and `-s` options which keywords are on the “keep list” and “skip list”, respectively. From all the files to be dependency sorted, MAN.RCORDER.8 will pick only those having a keyword from the keep list (unless empty) and not having a keyword from the skip list.

In OS, MAN.RCORDER.8 is used by `/etc/rc` and `/etc/rc.shutdown`. These two scripts define the standard list of OS `rc.d` keywords and their meanings as follows:

nojail The service is not for MAN.JAIL.8 environment. The automatic startup and shutdown procedures will ignore the script if inside a jail.

nostart The service is to be started manually or not started at all. The automatic startup procedure will ignore the script. In conjunction with the `shutdown` keyword, this can be used to write scripts that do something only at system shutdown.

shutdown This keyword is to be listed *explicitly* if the service needs to be stopped before system shutdown.

Note

When the system is going to shut down, `/etc/rc.shutdown` runs. It assumes that most `rc.d` scripts have nothing to do at that time. Therefore `/etc/rc.shutdown` selectively invokes `rc.d` scripts with the `shutdown` keyword, effectively ignoring the rest of the scripts. For even faster shutdown, `/etc/rc.shutdown` passes the `faststop` command to the scripts it runs so that they skip preliminary checks, e.g., the pidfile check. As dependent services should be stopped before their prerequisites, `/etc/rc.shutdown` runs the scripts in reverse dependency order.

If writing a real `rc.d` script, you should consider whether it is relevant at system shutdown time. E.g., if your script does its work in response to the `start` command only, then you need not include this keyword. However, if your script manages a service, it is probably a good idea to stop it before the system proceeds to the final stage of its shutdown sequence described in MAN.HALT.8. In particular, a service should be stopped explicitly if it needs considerable time or special actions to shut down cleanly. A typical example of such a service is a database engine.

- To begin with, `force_depend` should be used with much care. It is generally better to revise the hierarchy of configuration variables for your `rc.d` scripts if they are interdependent.

If you still cannot do without `force_depend`, the example offers an idiom of how to invoke it conditionally. In the example, our mumbled daemon requires that another one, `frotz`, be started in advance. However, `frotz` is optional, too; and `MAN.RC.RECORDER.8` knows nothing about such details. Fortunately, our script has access to all `MAN.RC.CONF.5` variables. If `frotz_enable` is true, we hope for the best and rely on `rc.d` to have started `frotz`. Otherwise we forcibly check the status of `frotz`. Finally, we enforce our dependency on `frotz` if it is found to be not running. A warning message will be emitted by `force_depend` because it should be invoked only if a misconfiguration has been detected.

69.8 Giving more flexibility to an `rc.d` script

When invoked during startup or shutdown, an `rc.d` script is supposed to act on the entire subsystem it is responsible for. E.g., `/etc/rc.d/netif` should start or stop all network interfaces described by `MAN.RC.CONF.5`. Either task can be uniquely indicated by a single command argument such as `start` or `stop`. Between startup and shutdown, `rc.d` scripts help the admin to control the running system, and it is when the need for more flexibility and precision arises. For instance, the admin may want to add the settings of a new network interface to `MAN.RC.CONF.5` and then to start it without interfering with the operation of the existing interfaces. Next time the admin may need to shut down a single network interface. In the spirit of the command line, the respective `rc.d` script calls for an extra argument, the interface name.

Fortunately, `MAN.RC.SUBR.8` allows for passing any number of arguments to script's methods (within the system limits). Due to that, the changes in the script itself can be minimal.

How can `MAN.RC.SUBR.8` gain access to the extra command-line arguments. Should it just grab them directly? Not by any means. Firstly, an `MAN.SH.1` function has no access to the positional parameters of its caller, but `MAN.RC.SUBR.8` is just a sack of such functions. Secondly, the good manner of `rc.d` dictates that it is for the main script to decide which arguments are to be passed to its methods.

So the approach adopted by `MAN.RC.SUBR.8` is as follows: `run_rc_command` passes on all its arguments but the first one to the respective method verbatim. The first, omitted, argument is the name of the method itself: `start`, `stop`, etc. It will be shifted out by `run_rc_command`, so what is `$2` in the original command line will be presented as `$1` to the method, and so on.

To illustrate this opportunity, let us modify the primitive dummy script so that its messages depend on the additional arguments supplied. Here we go:

```
#!/bin/sh

. /etc/rc.subr

name="dummy"
start_cmd="${name}_start"
stop_cmd=":"
kiss_cmd="${name}_kiss"
extra_commands="kiss"

dummy_start()
{
    if [ $# -gt 0 ]; then
        echo "Greeting message: $*"
    else
        echo "Nothing started."
    fi
}

dummy_kiss()
{
```

```

    echo -n "A ghost gives you a kiss"
    if [ $# -gt 0 ]; then
        echo -n " and whispers: $*"
    fi
    case "$*" in
        *[\!?\])
            echo
            ;;
        *)
            echo .
            ;;
    esac
}

load_rc_config $name
run_rc_command "$@"

```

What essential changes can we notice in the script?

- All arguments you type after `start` can end up as positional parameters to the respective method. We can use them in any way according to our task, skills, and fancy. In the current example, we just pass all of them to `MAN.ECHO.1` as one string in the next line — note `$*` within the double quotes. Here is how the script can be invoked now:

```

PROMPT.ROOT /etc/rc.d/dummy start
Nothing started.
PROMPT.ROOT /etc/rc.d/dummy start Hello world!
Greeting message: Hello world!

```

- The same applies to any method our script provides, not only to a standard one. We have added a custom method named `kiss`, and it can take advantage of the extra arguments not less than `start` does. E.g.:

```

PROMPT.ROOT /etc/rc.d/dummy kiss
A ghost gives you a kiss.
PROMPT.ROOT /etc/rc.d/dummy kiss Once I was Etaoin Shrdlu...
A ghost gives you a kiss and whispers: Once I was Etaoin Shrdlu...

```

- If we want just to pass all extra arguments to any method, we can merely substitute `"$@"` for `"$1"` in the last line of our script, where we invoke `run_rc_command`.

Important

An `MAN.SH.1` programmer ought to understand the subtle difference between `$*` and `$@` as the ways to designate all positional parameters. For its in-depth discussion, refer to a good handbook on `MAN.SH.1` scripting. *Do not* use the expressions until you fully understand them because their misuse will result in buggy and insecure scripts.

Note

Currently `run_rc_command` may have a bug that prevents it from keeping the original boundaries between arguments. That is, arguments with embedded whitespace may not be processed correctly. The bug stems from `$*` misuse.

69.9 Further reading

The [original article by Luke Mewburn](#) offers a general overview of `rc.d` and detailed rationale for its design decisions. It provides insight on the whole `rc.d` framework and its place in a modern BSD operating system.

The manual pages MAN.RC.8, MAN.RC.SUBR.8, and MAN.RCORDER.8 document the `rc.d` components in great detail. You cannot fully use the `rc.d` power without studying the manual pages and referring to them while writing your own scripts.

The major source of working, real-life examples is `/etc/rc.d` in a live system. Its contents are easy and pleasant to read because most rough corners are hidden deep in MAN.RC.SUBR.8. Keep in mind though that the `/etc/rc.d` scripts were not written by angels, so they might suffer from bugs and suboptimal design decisions. Now you can improve them!

Using Greylist with OS

Author TomRhodes

70.1 Basic Configuration

Install perl using

```
PROMPT.ROOT pkg install lang/perl5.16
```

Now for the database server; MySQL is perfect for this sort of work. Install the databases/mysql40-server along with databases/p5-DBD-mysql40. The previous port should imply the installation of databases/p5-DBI-137 so that knocks off another step.

Install the perl based portable server plugin, net/p5-Net-Daemon port. Most of these port installations should have been straight forward. The next step will be more involved.

Now install the mail/p5-Sendmail-Milter port. As of this writing the Makefile contains a line beginning with BROKEN, just remove it or comment it out. It is only marked this way because OS neither has nor installs a threaded perl package by default. Once that line is removed it should build and install perfectly fine.

Create a directory to hold temporary configuration files:

```
PROMPT.ROOT mkdir /tmp/relaydelay
PROMPT.ROOT cd /tmp/relaydelay
```

Now that we have a temporary directory to work in, the following URLs should be sent to the fetch command:

```
PROMPT.ROOT fetch http://projects.puremagic.com/greylisting/releases/relaydelay-0.04.tgz
PROMPT.ROOT fetch http://lists.puremagic.com/pipermail/greylist-users/attachments/20030904/b8dafed9/1
```

The source code should now be unpacked:

```
PROMPT.ROOT gunzip -c relaydelay-0.04.tgz | tar xvf -
```

There should now be several files into the temporary directory by this point. The appropriate information can now be passed to the database server by importing it from the mysql.sql file:

```
PROMPT.ROOT mysql < relaydelay-0.04/mysql.sql
```

And patch the other files with the relaydelay.bin by running:

```
PROMPT.ROOT patch -d /tmp/relaydelay/relaydelay-0.04 < relaydelay.bin
```

Edit the `relaydelay.conf` and the `db_maintenance.pl` file to append the correct username and password for the MySQL database. If the database was built and installed like the above then no users or passwords exist. This should be altered before putting this into production, that is covered in the database documentation and is beyond the scope of this document.

Change the working directory to the `relaydelay-0.04` directory:

```
PROMPT.ROOT cd relaydelay-0.04
```

Copy or move the configuration files to their respective directories:

```
PROMPT.ROOT mv db_maintenance.pl relaydelay.pl /usr/local/sbin
PROMPT.ROOT mv relaydelay.conf /etc/mail
PROMPT.ROOT mv relaydelay.sh /usr/local/etc/rc.d/
```

Test the current configuration by running:

```
PROMPT.ROOT sh /usr/local/etc/rc.d/relaydelay.sh start

**Note**

This file will not exist if the previous MAN.MV.1 commands were
neglected.
```

If everything worked correctly a new file, `relaydelay.log`, should exist in `/var/log`. It should contain something similar to the following text:

```
Loaded Config File: /etc/mail/relaydelay.conf
Using connection 'local:/var/run/relaydelay.sock' for filter relaydelay
DBI Connecting to DBI:mysql:database=relaydelay:host=localhost:port=3306
Spawned relaydelay daemon process 38277.
Starting Sendmail::Milter 0.18 engine.
```

If this does not appear then something went wrong, review the screen output or look for anything new in the messages log file.

Glue everything together by adding the following line to `/etc/mail/sendmail.mc` or the customized site specific mc file:

```
INPUT_MAIL_FILTER(`relaydelay', `S=local:/var/run/relaydelay.sock, T=S:1m;R:2m;E:3m')dnl
```

Rebuild and reinstall the files in the `/etc/mail` directory and restart `sendmail`. A quick make restart should do the trick.

Obtain the perl script located at <http://lists.puremagic.com/pipermail/greylist-users/2003-November/000327.html> and save it in the `relaydelay-0.04` directory. In the following examples this script is referred to as `addlist.pl`.

Edit the `whitelist_ip.txt` file and modify it to include IP addresses of servers which should have the explicit abilities to bypass the relaydelay filters. i.e., domains from which email will not be issued a `TEMPFAIL` when received.

Some examples could include:

```
192.168.    # My internal network.
66.218.66  # Yahoo groups has unique senders.
```

The `blacklist_ip.txt` file should be treated similarly but with reversed rules. List within this file IPs which should be denied without being issued a `TEMPFAIL`. This list of domains will never have the opportunity to prove that they are legitimate email servers.

These files should now be imported into the database with the `addlist.pl` script obtained a few lines ago:

```
PROMPT.ROOT perl addlist.pl -whitelist 9999-12-31 23:59:59 < whitelist_ip.txt
PROMPT.ROOT perl addlist.pl -blacklist 9999-12-31 23:59:59 < blacklist_ip.txt
```

To have relaydelay start with every system boot, add the `relaydelay_enable="YES"` to the `/etc/rc.conf` file.

The `/var/log/relaydelay.log` log file should slowly fill up with success stories. Lines like the following should appear after a short time, depending on how busy the mail server is.

```
=== 2004-05-24 21:03:22 ===
Stored Sender: <someasshole@flawed-example.com>
Passed Recipient: <local_user@pittgoth.com>
  Relay: example.net [XXX.XX.XXX.XX] - If_Addr: MY_IP_ADDRESS
  RelayIP: XX.XX.XX.XX - RelayName: example.net - RelayIdent: - PossiblyForged: 0
  From: someasshole@flawed-example.com - To: local_user
  InMailer: esmtp - OutMailer: local - QueueID: i4P13Lo6000701111
  Email is known but block has not expired. Issuing a tempfail. rowid: 51
  IN ABORT CALLBACK - PrivData: 0<someasshole@flawed-example.com>
```

The following line may now be added to `/etc/newsyslog.conf` to cause for `relaydelay.log` rotation at every 100 Kb:

```
/var/log/relaydelay.log          644  3    100  *    Z

**Note**

At some point there was an error about improper ``perl`` variables
in the ``/etc/mail/relaydelay.conf``. If those two variables are
commented out then configuration may proceed as normal. Just
remember to uncomment them before starting the ``relaydelay``
process.
```

OS Release Engineering

Author MurrayStokelyI've been involved in the development of OS based products since 1997 at Walnut Creek CDRom, BSDi, and now Wind River Systems. OS 4.4 was the first official release of OS that I played a significant part in.

71.1 Introduction

The development of OS is a very open process. OS is comprised of contributions from thousands of people around the world. The OS Project provides Subversion¹ access to the general public so that others can have access to log messages, diffs (patches) between development branches, and other productivity enhancements that formal source code management provides. This has been a huge help in attracting more talented developers to OS. However, I think everyone would agree that chaos would soon manifest if write access to the main repository was opened up to everyone on the Internet. Therefore only a “select” group of nearly 300 people are given write access to the Subversion repository. These committers

² are usually the people who do the bulk of OS development. An elected Core Team³ of developers provide some level of direction over the project.

The rapid pace of OS development makes the main development branch unsuitable for the everyday use by the general public. In particular, stabilizing efforts are required for polishing the development system into a production quality release. To solve this conflict, development continues on several parallel tracks. The main development branch is the *HEAD* or *trunk* of our Subversion tree, known as “OS-CURRENT” or “-CURRENT” for short.

A set of more stable branches are maintained, known as “OS-STABLE” or “-STABLE” for short. All branches live in a master Subversion repository maintained by the OS Project. OS-CURRENT is the “bleeding-edge” of OS development where all new changes first enter the system. OS-STABLE is the development branch from which major releases are made. Changes go into this branch at a different pace, and with the general assumption that they have first gone into OS-CURRENT and have been thoroughly tested by our user community.

The term *stable* in the name of the branch refers to the presumed Application Binary Interface stability, which is promised by the project. This means that a user application compiled on an older version of the system from the same branch works on a newer system from the same branch. The ABI stability has improved greatly from the compared to previous releases. In most cases, binaries from the older *STABLE* systems run unmodified on newer systems, including *HEAD*, assuming that the system management interfaces are not used.

In the interim period between releases, weekly snapshots are built automatically by the OS Project build machines and made available for download from <ftp://ftp.FreeBSD.org/pub/FreeBSD/snapshots/>. The widespread availability

¹ Subversion, <http://subversion.apache.org>

² FreeBSD committers

³ OS Core Team

of binary release snapshots, and the tendency of our user community to keep up with -STABLE development with Subversion and “make buildworld”⁴ helps to keep OS-STABLE in a very reliable condition even before the quality assurance activities ramp up pending a major release.

In addition to installation ISO snapshots, weekly virtual machine images are also provided for use with VirtualBox, qemu, or other popular emulation software. The virtual machine images can be downloaded from <ftp://ftp.FreeBSD.org/pub/FreeBSD/snapshots/VM-IMAGES/>.

The virtual machine images are approximately 150MB MAN.XZ.1 compressed, and contain a 10GB sparse filesystem when attached to a virtual machine.

Bug reports and feature requests are continuously submitted by users throughout the release cycle. Problems reports are entered into our Bugzilla database through the web interface provided at <https://www.freebsd.org/support/bugreports.html>.

To service our most conservative users, individual release branches were introduced with OS 4.3. These release branches are created shortly before a final release is made. After the release goes out, only the most critical security fixes and additions are merged onto the release branch. In addition to source updates via Subversion, binary patchkits are available to keep systems on the *releng/X.Y* branches updated.

71.1.1 What this article describes

The following sections of this article describe:

- ? The different phases of the release engineering process leading up to the actual system build.
- ? The actual build process.
- ? How the base release may be extended by third parties.
- ? Some of the lessons learned through the release of OS 4.4.
- ? Future directions of development.

71.2 Release Process

New releases of OS are released from the -STABLE branch at approximately four month intervals. The OS release process begins to ramp up 70-80 days before the anticipated release date when the release engineer sends an email to the development mailing lists to remind developers that they only have 15 days to integrate new changes before the code freeze. During this time, many developers perform what have become known as “MFC sweeps”.

MFC stands for “Merge From CURRENT” and it describes the process of merging a tested change from our -CURRENT development branch to our -STABLE branch. Project policy requires any change to be first applied to trunk, and merged to the -STABLE branches after sufficient external testing was done by -CURRENT users (developers are expected to extensively test the change before committing to -CURRENT, but it is impossible for a person to exercise all usages of the general-purpose operating system). Minimal MFC period is 3 days, which is typically used only for trivial or critical bugfixes.

71.2.1 Code Review

Sixty days before the anticipated release, the source repository enters a “code freeze”. During this time, all commits to the -STABLE branch must be approved by A.RE. The approval process is technically enforced by a pre-commit hook. The kinds of changes that are allowed during this period include:

- Bug fixes.

⁴ Rebuilding “world”

- Documentation updates.
- Security-related fixes of any kind.
- Minor changes to device drivers, such as adding new Device IDs.
- Driver updates from the vendors.
- Any additional change that the release engineering team feels is justified, given the potential risk.

Shortly after the code freeze is started, a *BETA1* image is built and released for widespread testing. During the code freeze, at least one beta image or release candidate is released every two weeks until the final release is ready. During the days preceeding the final release, the release engineering team is in constant communication with the security-officer team, the documentation maintainers, and the port maintainers to ensure that all of the different components required for a successful release are available.

After the quality of the BETA images is satisfying enough, and no large and potentially risky changes are planned, the release branch is created and *Release Candidate* (RC) images are built from the release branch, instead of the BETA images from the STABLE branch. Also, the freeze on the STABLE branch is lifted and release branch enters a “hard code freeze” where it becomes much harder to justify new changes to the system unless a serious bug-fix or security issue is involved.

71.2.2 Final Release Checklist

When several BETA images have been made available for widespread testing and all major issues have been resolved, the final release “polishing” can begin.

Creating the Release Branch

Note

In all examples below, `$FSVN` refers to the location of the OS Subversion repository, `svn+ssh://svn.FreeBSD.org/base/`.

The layout of OS branches in Subversion is described in the Committer’s Guide. The first step in creating a branch is to identify the revision of the `stable/X` sources that you want to branch *from*.

```
PROMPT.ROOT svn log -v $FSVN/stable/9
```

The next step is to create the *release branch*

```
PROMPT.ROOT svn cp $FSVN/stable/9@REVISION $FSVN/releng/9.2
```

This branch can be checked out:

```
PROMPT.ROOT svn co $FSVN/releng/9.2 src
```

****Note****

Creating the ``releng`` branch and ``release`` tags is done by the
`Release Engineering Team <url.base;/administration.html#t-re>`__.

OS Development Branch |

OS 3.x STABLE Branch |

OS 4.x STABLE Branch |

OS 5.x STABLE Branch |

OS 6.x STABLE Branch |

OS 7.x STABLE Branch |

OS 8.x STABLE Branch |

OS 9.x STABLE Branch |

Bumping up the Version Number

Before the final release can be tagged, built, and released, the following files need to be modified to reflect the correct version of OS:

- **“doc/en_US.ISO8859-1/books/handbook/mirrors/chapter.xml “**
- **“doc/en_US.ISO8859-1/books/porters-handbook/book.xml “**
- doc/en_US.ISO8859-1/htdocs/cgi/ports.cgi
- ports/Tools/scripts/release/config
- doc/share/xml/freebsd.ent
- src/Makefile.incl
- src/UPDATING
- src/gnu/usr.bin/groff/tmac/mdoc.local
- src/release/Makefile
- src/release/doc/en_US.ISO8859-1/share/xml/release.dsl
- src/release/doc/share/examples/Makefile.relnotesng
- src/release/doc/share/xml/release.ent
- src/sys/conf/newvers.sh
- src/sys/sys/param.h
- src/usr.sbin/pkg_install/add/main.c
- doc/en_US.ISO8859-1/htdocs/search/opensource/man.xml

The release notes and errata files also need to be adjusted for the new release (on the release branch) and truncated appropriately (on the stable/current branch):

- **“src/release/doc/en_US.ISO8859-1/relnotes/common/new.xml “**
- **“src/release/doc/en_US.ISO8859-1/errata/article.xml “**

Sysinstall should be updated to note the number of available ports and the amount of disk space required for the Ports Collection.⁵ This information is currently kept in `src/usr.sbin/sysinstall/dist.c`.

After the release has been built, a number of files should be updated to announce the release to the world. These files are relative to `head/` within the `doc/` subversion tree.

- `share/images/articles/releng/branches-relengX.pic`
- `head/share/xml/release.ent`
- `en_US.ISO8859-1/htdocs/releases/*`
- `en_US.ISO8859-1/htdocs/releng/index.xml`
- `share/xml/news.xml`

Additionally, update the “BSD Family Tree” file:

- `src/share/misc/bsd-family-tree`

Creating the Release Tag

When the final release is ready, the following command will create the `release/9.2.0` tag.

```
PROMPT.ROOT svn cp $FSVN/releng/9.2 $FSVN/release/9.2.0
```

The Documentation and Ports managers are responsible for tagging their respective trees with the `tags/RELEASE_9_2_0` tag.

When the Subversion `svn cp` command is used to create a *release tag*, this identifies the source at a specific point in time. By creating tags, we ensure that future release builders will always be able to use the exact same source we used to create the official OS Project releases.

71.3 Release Building

OS “releases” can be built by anyone with a fast machine and access to a source repository. (That should be everyone, since we offer Subversion access ! See the Subversion section in the Handbook for details.) The *only* special requirement is that the MAN.MD.4 device must be available. If the device is not loaded into your kernel, then the kernel module should be automatically loaded when MAN.MDCONFIG.8 is executed during the boot media creation phase. All of the tools necessary to build a release are available from the Subversion repository in `src/release`. These tools aim to provide a consistent way to build OS releases. A complete release can actually be built with only a single command, including the creation of ISO images suitable for burning to CDROM or DVD, and an FTP install directory. MAN.RELEASE.7 fully documents the `src/release/generate-release.sh` script which is used to build a release. `generate-release.sh` is a wrapper around the Makefile target: `make release`.

71.3.1 Building a Release

MAN.RELEASE.7 documents the exact commands required to build a OS release. The following sequences of commands can build an 9.2.0 release:

```
PROMPT.ROOT cd /usr/src/release
```

```
PROMPT.ROOT sh generate-release.sh release/9.2.0 /local3/release
```

⁵ OS Ports Collection <http://www.FreeBSD.org/ports>

After running these commands, all prepared release files are available in `/local3/release/R` directory.

The release `Makefile` can be broken down into several distinct steps.

- Creation of a sanitized system environment in a separate directory hierarchy with “`make installworld`”.
- Checkout from Subversion of a clean version of the system source, documentation, and ports into the release build hierarchy.
- Population of `/etc` and `/dev` in the chrooted environment.
- chroot into the release build hierarchy, to make it harder for the outside environment to taint this build.
- `make world` in the chrooted environment.
- Build of Kerberos-related binaries.
- Build `GENERIC` kernel.
- Creation of a staging directory tree where the binary distributions will be built and packaged.
- Build and installation of the documentation toolchain needed to convert the documentation source (SGML) into HTML and text documents that will accompany the release.
- Build and installation of the actual documentation (user manuals, tutorials, release notes, hardware compatibility lists, and so on.)
- Package up distribution tarballs of the binaries and sources.
- Create FTP installation hierarchy.
- (optionally) Create ISO images for CDROM/DVD media.

For more information about the release build infrastructure, please see `MAN.RELEASE.7`.

Note

It is important to remove any site-specific settings from `/etc/make.conf`. For example, it would be unwise to distribute binaries that were built on a system with `CPU_TYPE` set to a specific processor.

71.3.2 Contributed Software (“ports”)

The [OS Ports collection](#) is a collection of over `OS.NUMPORTS` third-party software packages available for OS. The `A.PORTMGR` is responsible for maintaining a consistent ports tree that can be used to create the binary packages that accompany official OS releases.

71.3.3 Release ISOs

Starting with OS 4.4, the OS Project decided to release all four ISO images that were previously sold on the *BSDi/Wind River Systems/FreeBSD Mall* “official” CDROM distributions. Each of the four discs must contain a `README.TXT` file that explains the contents of the disc, a `CDROM.INF` file that provides meta-data for the disc so that `MAN.SYSINSTALL.8` can validate and use the contents, and a `filename.txt` file that provides a manifest for the disc. This *manifest* can be created with a simple command:

```
/stage/cdromPROMPT.ROOT find . -type f | sed -e 's/^\.\.\.\.\.' | sort > filename.txt
```

The specific requirements of each CD are outlined below.

Disc 1

The first disc is almost completely created by “`make release`”. The only changes that should be made to the `disc1`

directory are the addition of a `tools` directory, and as many popular third party software packages as will fit on the disc. The `tools` directory contains software that allow users to create installation floppies from other operating systems. This disc should be made bootable so that users of modern PCs do not need to create installation floppy disks.

If a custom kernel of OS is to be included, then `MAN.SYSINSTALL.8` and `MAN.RELEASE.7` must be updated to include installation instructions. The relevant code is contained in `src/release` and `src/usr.sbin/sysinstall`. Specifically, the file `src/release/Makefile`, and `dist.c`, `dist.h`, `menus.c`, `install.c`, and `Makefile` will need to be updated under `src/usr.sbin/sysinstall`. Optionally, you may choose to update `sysinstall.8`.

Disc 2

The second disc is also largely created by “`make release`”. This disc contains a “live filesystem” that can be used from `MAN.SYSINSTALL.8` to troubleshoot a OS installation. This disc should be bootable and should also contain a compressed copy of the CVS repository in the `CVSROOT` directory and commercial software demos in the `commerce` directory.

Multi-volume support

Sysinstall supports multiple volume package installations. This requires that each disc have an `INDEX` file containing all of the packages on all volumes of a set, along with an extra field that indicates which volume that particular package is on. Each volume in the set must also have the `CD_VOLUME` variable set in the `cdrom.inf` file so that sysinstall can tell which volume is which. When a user attempts to install a package that is not on the current disc, sysinstall will prompt the user to insert the appropriate one.

71.4 Distribution

71.4.1 FTP Sites

When the release has been thoroughly tested and packaged for distribution, the master FTP site must be updated. The official OS public FTP sites are all mirrors of a master server that is open only to other FTP sites. This site is known as `ftp-master`. When the release is ready, the following files must be modified on `ftp-master`:

`/pub/FreeBSD/releases/arch/X.Y-RELEASE/`

The installable FTP directory as output from “`make release`”.

`/pub/FreeBSD/ports/arch/packages-X.Y-release/` The complete package build for this release.

`/pub/FreeBSD/releases/arch/X.Y-RELEASE/tools` A symlink to `../../../../tools`.

`/pub/FreeBSD/releases/arch/X.Y-RELEASE/packages` A symlink to `../../../../ports/arch/packages-X.Y-release`.

`/pub/FreeBSD/releases/arch/ISO-IMAGES/X.Y/X.Y-RELEASE-arch-*.iso` The ISO images. The “*” is `disc1`, `disc2`, etc. Only if there is a `disc1` and there is an alternative first installation CD (for example a stripped-down install with no windowing system) there may be a `mini` as well.

For more information about the distribution mirror architecture of the OS FTP sites, please see the [Mirroring OS](#) article.

It may take many hours to two days after updating ftp-master before a majority of the Tier-1 FTP sites have the new software depending on whether or not a package set got loaded at the same time. It is imperative that the release engineers coordinate with the A.MIRROR-ANNOUNCE before announcing the general availability of new software on the FTP sites. Ideally the release package set should be loaded at least four days prior to release day. The release bits should be loaded between 24 and 48 hours before the planned release time with “other” file permissions turned off. This will allow the mirror sites to download it but the general public will not be able to download it from the mirror sites. Mail should be sent to A.MIRROR-ANNOUNCE at the time the release bits get posted saying the release has been staged and giving the time that the mirror sites should begin allowing access. Be sure to include a time zone with the time, for example make it relative to GMT.

71.4.2 CD-ROM Replication

Coming soon: Tips for sending OS ISOs to a replicator and quality assurance measures to be taken.

71.5 Extensibility

Although OS forms a complete operating system, there is nothing that forces you to use the system exactly as we have packaged it up for distribution. We have tried to design the system to be as extensible as possible so that it can serve as a platform that other commercial products can be built on top of. The only “rule” we have about this is that if you are going to distribute OS with non-trivial changes, we encourage you to document your enhancements! The OS community can only help support users of the software we provide. We certainly encourage innovation in the form of advanced installation and administration tools, for example, but we cannot be expected to answer questions about it.

71.5.1 Scripting `sysinstall`

The OS system installation and configuration tool, MAN.SYSINSTALL.8, can be scripted to provide automated installs for large sites. This functionality can be used in conjunction with INTEL PXE ⁶ to bootstrap systems from the network.

71.6 Lessons Learned from OS 4.4

The release engineering process for 4.4 formally began on August 1st, 2001. After that date all commits to the RELENG_4 branch of OS had to be explicitly approved by the A.RE. The first release candidate for the x86 architecture was released on August 16, followed by 4 more release candidates leading up to the final release on September 18th. The security officer was very involved in the last week of the process as several security issues were found in the earlier release candidates. A total of over 500 emails were sent to the A.RE in little over a month.

Our user community has made it very clear that the security and stability of a OS release should not be sacrificed for any self-imposed deadlines or target release dates. The OS Project has grown tremendously over its lifetime and the need for standardized release engineering procedures has never been more apparent. This will become even more important as OS is ported to new platforms.

⁶ URL.BOOKS.HANDBOOK/network-pxe-nfs.html

71.7 Future Directions

It is imperative for our release engineering activities to scale with our growing userbase. Along these lines we are working very hard to document the procedures involved in producing OS releases.

- *Parallelism* - Certain portions of the release build are actually “embarrassingly parallel”. Most of the tasks are very I/O intensive, so having multiple high-speed disk drives is actually more important than using multiple processors in speeding up the “make release” process. If multiple disks are used for different hierarchies in the MAN.CHROOT.2 environment, then the CVS checkout of the `ports` and `doc` trees can be happening simultaneously as the “make world” on another disk. Using a RAID solution (hardware or software) can significantly decrease the overall build time.
- *Cross-building releases* - Building IA-64 or Alpha release on x86 hardware? “make TARGET=ia64 release”.
- *Regression Testing* - We need better automated correctness testing for OS.
- *Installation Tools* - Our installation program has long since outlived its intended life span. Several projects are under development to provide a more advanced installation mechanism. The `libh` project was one such project that aimed to provide an intelligent new package framework and GUI installation program.

71.8 Acknowledgements

I would like to thank Jordan Hubbard for giving me the opportunity to take on some of the release engineering responsibilities for OS 4.4 and also for all of his work throughout the years making OS what it is today. Of course the release would not have been possible without all of the release-related work done by A.ASAMI.EMAIL, A.STEVE.EMAIL, A.BMAH.EMAIL, A.NIK.EMAIL, A.OBRIEN.EMAIL, A.KRIS.EMAIL, A.JHB.EMAIL and the rest of the OS development community. I would also like to thank A.RGRIMES.EMAIL, A.PHK.EMAIL, and others who worked on the release engineering tools in the very early days of OS. This article was influenced by release engineering documents from the CSRG⁷, the NetBSD Project⁸, and John Baldwin’s proposed release engineering process notes.⁹

⁷ Marshall Kirk McKusick, Michael J. Karels, and Keith Bostic: *The Release Engineering of 4.3BSD*

⁸ NetBSD Developer Documentation: Release Engineering <http://www.NetBSD.org/developers/releng/index.html>

⁹ John Baldwin’s OS Release Engineering Proposal <http://people.FreeBSD.org/~jhb/docs/releng.txt>

Remote Installation of the OS Operating System Without a Remote Console

Author Daniel Gerzo

72.1 Background

There are many server hosting providers in the world, but very few of them are officially supporting OS. They usually provide support for a LINUX distribution to be installed on the servers they offer.

In some cases, these companies will install your preferred LINUX distribution if you request it. Using this option, we will attempt to install OS. In other cases, they may offer a rescue system which would be used in an emergency. It is possible to use this for our purposes as well.

This article covers the basic installation and configuration steps required to bootstrap a remote installation of OS with RAID-1 and ZFS capabilities.

72.2 Introduction

This section will summarize the purpose of this article and better explain what is covered herein. The instructions included in this article will benefit those using services provided by colocation facilities not supporting OS.

As we have mentioned in the [Background](#) section, many of the reputable server hosting companies provide some kind of rescue system, which is booted from their LAN and accessible over SSH. They usually provide this support in order to help their customers fix broken operating systems. As this article will explain, it is possible to install OS with the help of these rescue systems.

The next section of this article will describe how to configure, and build minimalistic OS on the local machine. That version will eventually be running on the remote machine from a ramdisk, which will allow us to install a complete OS operating system from an FTP mirror using the sysinstall utility.

The rest of this article will describe the installation procedure itself, as well as the configuration of the ZFS file system.

72.2.1 Requirements

To continue successfully, you must:

- Have a network accessible operating system with SSH access
- Understand the OS installation process

- Be familiar with the MAN.SYSINSTALL.8 utility
- Have the OS installation ISO image or CD handy

72.3 Preparation - mfsBSD

Before OS may be installed on the target system, it is necessary to build the minimal OS operating system image which will boot from the hard drive. This way the new system can be accessed from the network, and the rest of the installation can be done without remote access to the system console.

The mfsBSD tool-set can be used to build a tiny OS image. As the name of mfsBSD suggests (“mfs” means “memory file system”), the resulting image runs entirely from a ramdisk. Thanks to this feature, the manipulation of hard drives will not be limited, therefore it will be possible to install a complete OS operating system. The mfsBSD home page includes pointers to the latest release of the toolset.

Please note that the internals of mfsBSD and how it all fits together is beyond the scope of this article. The interested reader should consult the original documentation of mfsBSD for more details.

Download and extract the latest mfsBSD release and change your working directory to the directory where the mfsBSD scripts will reside:

```
PROMPT.ROOT fetch http://mfsbsd.vx.sk/release/mfsbsd-2.1.tar.gz
PROMPT.ROOT tar xvzf mfsbsd-2.1.tar.gz
PROMPT.ROOT cd mfsbsd-2.1/
```

72.3.1 Configuration of mfsBSD

Before booting mfsBSD, a few important configuration options have to be set. The most important that we have to get right is, naturally, the network setup. The most suitable method to configure networking options depends on whether we know beforehand the type of the network interface we will use, and the network interface driver to be loaded for our hardware. We will see how mfsBSD can be configured in either case.

Another important thing to set is the root password. This can be done by editing `conf/loader.conf`. Please see the included comments.

The `conf/interfaces.conf` method

When the installed network interface card is unknown, it is possible to use the auto-detection features of mfsBSD. The startup scripts of mfsBSD can detect the correct driver to use, based on the MAC address of the interface, if we set the following options in `conf/interfaces.conf`:

```
mac_interfaces="ext1"
ifconfig_ext1_mac="00:00:00:00:00:00"
ifconfig_ext1="inet 192.168.0.2/24"
```

Do not forget to add the defaultrouter information to `conf/rc.conf`:

```
defaultrouter="192.168.0.1"
```

The `conf/rc.conf` Method

When the network interface driver is known, it is more convenient to use `conf/rc.conf` for networking options. The syntax of this file is the same as the one used in the standard MAN.RC.CONF.5 file of OS.

For example, if you know that a MAN.RE.4 network interface is going to be available, you can set the following options in `conf/rc.conf`:

```
defaultrouter="192.168.0.1"
ifconfig_re0="inet 192.168.0.2/24"
```

72.3.2 Building an mfsBSD Image

The process of building an mfsBSD image is pretty straightforward.

The first step is to mount the OS installation CD, or the installation ISO image to `/cdrom`. For the sake of example, in this article we will assume that you have downloaded the OS 10.1-RELEASE ISO. Mounting this ISO image to the `/cdrom` directory is easy with the MAN.MDCONFIG.8 utility:

```
PROMPT.ROOT mdconfig -a -t vnode -u 10 -f FreeBSD-10.1-RELEASE-amd64-disc1.iso
PROMPT.ROOT mount_cd9660 /dev/md10 /cdrom
```

Since the recent OS releases do not contain regular distribution sets, it is required to extract the OS distribution files from the distribution archives located on the ISO image:

```
PROMPT.ROOT mkdir DIST
PROMPT.ROOT tar -xvf /cdrom/usr/freebsd-dist/base.txz -C DIST
PROMPT.ROOT tar -xvf /cdrom/usr/freebsd-dist/kernel.txz -C DIST
```

Next, build the bootable mfsBSD image:

```
PROMPT.ROOT make BASE=DIST

**Note**

The above ``make`` has to be run from the top level of the mfsBSD
directory tree, for example ``~/mfsbsd-2.1/``.
```

72.3.3 Booting mfsBSD

Now that the mfsBSD image is ready, it must be uploaded to the remote system running a live rescue system or pre-installed LINUX distribution. The most suitable tool for this task is `scp`:

```
PROMPT.ROOT scp disk.img root@192.168.0.2:.
```

To boot mfsBSD image properly, it must be placed on the first (bootable) device of the given machine. This may be accomplished using this example providing that `sda` is the first bootable disk device:

```
PROMPT.ROOT dd if=/root/disk.img of=/dev/sda bs=1m
```

If all went well, the image should now be in the MBR of the first device and the machine can be rebooted. Watch for the machine to boot up properly with the MAN.PING.8 tool. Once it has came back on-line, it should be possible to access it over MAN.SSH.1 as user `root` with the configured password.

72.4 Installation of the OS Operating System

The mfsBSD has been successfully booted and it should be possible to log in through MAN.SSH.1. This section will describe how to create and label slices, set up `gmirror` for RAID-1, and how to use `sysinstall` to install a minimal distribution of the OS operating system.

72.4.1 Preparation of Hard Drives

The first task is to allocate disk space for OS, i.e.: to create slices and partitions. Obviously, the currently running system is fully loaded in system memory and therefore there will be no problems with manipulating hard drives. To complete this task, it is possible to use either `sysinstall` or `MAN.FDISK.8` in conjunction to `MAN.BSDLABEL.8`.

At the start, mark all system disks as empty. Repeat the following command for each hard drive:

```
PROMPT.ROOT dd if=/dev/zero of=/dev/ad0 count=2
```

Next, create slices and label them with your preferred tool. While it is considered easier to use `sysinstall`, a powerful and also probably less buggy method will be to use standard text-based UNIX tools, such as `MAN.FDISK.8` and `MAN.BSDLABEL.8`, which will also be covered in this section. The former option is well documented in the Installing OS chapter of the OS Handbook. As it was mentioned in the introduction, this article will present how to set up a system with RAID-1 and ZFS capabilities. Our set up will consist of a small `MAN.GMIRROR.8` mirrored / (root), /usr and /var dataset, and the rest of the disk space will be allocated for a `MAN.ZPOOL.8` mirrored ZFS file system. Please note, that the ZFS file system will be configured after the OS operating system is successfully installed and booted.

The following example will describe how to create slices and labels, initialize `MAN.GMIRROR.8` on each partition and how to create a UFS2 file system in each mirrored partition:

```
PROMPT.ROOT fdisk -BI /dev/ad0
PROMPT.ROOT fdisk -BI /dev/ad1
PROMPT.ROOT bsdlablel -wB /dev/ad0s1
PROMPT.ROOT bsdlablel -wB /dev/ad1s1
PROMPT.ROOT bsdlablel -e /dev/ad0s1
PROMPT.ROOT bsdlablel /dev/ad0s1 > /tmp/bsdlablel.txt && bsdlablel -R /dev/ad1s1 /tmp/bsdlablel.txt
PROMPT.ROOT gmirror label root /dev/ad[01]s1a
PROMPT.ROOT gmirror label var /dev/ad[01]s1d
PROMPT.ROOT gmirror label usr /dev/ad[01]s1e
PROMPT.ROOT gmirror label -F swap /dev/ad[01]s1b
PROMPT.ROOT newfs /dev/mirror/root
PROMPT.ROOT newfs /dev/mirror/var
PROMPT.ROOT newfs /dev/mirror/usr
```

- Create a slice covering the entire disk and initialize the boot code contained in sector 0 of the given disk. Repeat this command for all hard drives in the system.
- Write a standard label for each disk including the bootstrap code.
- Now, manually edit the label of the given disk. Refer to the `MAN.BSDLABEL.8` manual page in order to find out how to create partitions. Create partitions a for / (root) file system, b for swap, d for /var, e for /usr and finally f which will later be used for ZFS.
- Import the recently created label for the second hard drive, so both hard drives will be labeled in the same way.
- Initialize `MAN.GMIRROR.8` on each partition.
- Note that `-F` is used for the swap partition. This instructs `MAN.GMIRROR.8` to assume that the device is in the consistent state after the power/system failure.
- Create a UFS2 file system on each mirrored partition.

72.4.2 System Installation

This is the most important part. This section will describe how to actually install the minimal distribution of OS on the hard drives that we have prepared in the previous section. To accomplish this goal, all file systems need to be mounted so `sysinstall` may write the contents of OS to the hard drives:

```
PROMPT.ROOT mount /dev/mirror/root /mnt
PROMPT.ROOT mkdir /mnt/var /mnt/usr
PROMPT.ROOT mount /dev/mirror/var /mnt/var
PROMPT.ROOT mount /dev/mirror/usr /mnt/usr
```

When you are done, start `MAN.SYSINSTALL.8`. Select the Custom installation from the main menu. Select Options and press Enter. With the help of arrow keys, move the cursor on the `Install Root` item, press Space and change it to `/mnt`. Press Enter to submit your changes and exit the Options menu by pressing `q`.

Warning

Note that this step is very important and if skipped, `sysinstall` will be unable to install OS.

Go to the Distributions menu, move the cursor with the arrow keys to `Minimal`, and check it by pressing Space. This article uses the Minimal distribution in order to save network traffic, because the system itself will be installed over ftp. Exit this menu by choosing `Exit`.

Note

The Partition and Label menus will be skipped, as these are useless now.

In the Media menu, select `FTP`. Select the nearest mirror and let `sysinstall` assume that the network is already configured. You will be returned back to the Custom menu.

Finally, perform the system installation by selecting the last option, `Commit`. Exit `sysinstall` when it finishes the installation.

72.4.3 Post Installation Steps

The OS operating system should be installed now; however, the process is not finished yet. It is necessary to perform some post installation steps in order to allow OS to boot in the future and to be able to log in to the system.

You must now `MAN.CHROOT.8` into the freshly installed system in order to finish the installation. Use the following command:

```
PROMPT.ROOT chroot /mnt
```

To complete our goal, perform these steps:

- Copy the `GENERIC` kernel to the `/boot/kernel` directory:

```
PROMPT.ROOT cp -Rp /boot/GENERIC/* /boot/kernel
```

- Create the `/etc/rc.conf`, `/etc/resolv.conf` and `/etc/fstab` files. Do not forget to properly set the network information and to enable `sshd` in `/etc/rc.conf`. The contents of `/etc/fstab` will be similar to the following:

# Device	Mountpoint	FStype	Options	Dump	Pass#
/dev/mirror/swap	none	swap	sw	0	0
/dev/mirror/root	/	ufs	rw	1	1
/dev/mirror/usr	/usr	ufs	rw	2	2
/dev/mirror/var	/var	ufs	rw	2	2
/dev/cd0	/cdrom	cd9660	ro,noauto	0	0

- Create `/boot/loader.conf` with the following contents:

```
geom_mirror_load="YES"
zfs_load="YES"
```

- Perform the following command, which will make ZFS available on the next boot:

```
PROMPT.ROOT echo 'zfs_enable="YES"' >> /etc/rc.conf
```

- Add additional users to the system using the MAN.ADDUSER.8 tool. Do not forget to add a user to the wheel group so you may obtain root access after the reboot.
- Double-check all your settings.

The system should now be ready for the next boot. Use the MAN.REBOOT.8 command to reboot your system.

72.5 ZFS

If your system survived the reboot, it should now be possible to log in. Welcome to the fresh OS installation, performed remotely without the use of a remote console!

The only remaining step is to configure MAN.ZPOOL.8 and create some MAN.ZFS.8 file systems. Creating and administering ZFS is very straightforward. First, create a mirrored pool:

```
PROMPT.ROOT zpool create tank mirror /dev/ad[01]s1f
```

Next, create some file systems:

```
PROMPT.ROOT zfs create tank/ports
PROMPT.ROOT zfs create tank/src
PROMPT.ROOT zfs set compression=gzip tank/ports
PROMPT.ROOT zfs set compression=on tank/src
PROMPT.ROOT zfs set mountpoint=/usr/ports tank/ports
PROMPT.ROOT zfs set mountpoint=/usr/src tank/src
```

That is all. If you are interested in more details about ZFS on OS, please refer to the [ZFS](#) section of the OS Wiki.

Serial and UART Tutorial

Author FrankDurda

73.1 The UART: What it is and how it works

Copyright © 1996 A.UHCLEM.EMAIL, All Rights Reserved. 13 January 1996.

The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes.

Serial transmission is commonly used with modems and for non-networked communication between computers, terminals and other devices.

There are two primary forms of serial transmission: Synchronous and Asynchronous. Depending on the modes that are supported by the hardware, the name of the communication sub-system will usually include a **A** if it supports Asynchronous communications, and a **S** if it supports Synchronous communications. Both forms are described below.

Some common acronyms are:

UART Universal Asynchronous Receiver/Transmitter

USART Universal Synchronous-Asynchronous Receiver/Transmitter

73.1.1 Synchronous Serial Transmission

Synchronous serial transmission requires that the sender and receiver share a clock with one another, or that the sender provide a strobe or other timing signal so that the receiver knows when to “read” the next bit of the data. In most forms of serial Synchronous communication, if there is no data available at a given instant to transmit, a fill character must be sent instead so that data is always being transmitted. Synchronous communication is usually more efficient because only data bits are transmitted between sender and receiver, and synchronous communication can be more costly if extra wiring and circuits are required to share a clock signal between the sender and receiver.

A form of Synchronous transmission is used with printers and fixed disk devices in that the data is sent on one set of wires while a clock or strobe is sent on a different wire. Printers and fixed disk devices are not normally serial devices because most fixed disk interface standards send an entire word of data for each clock or strobe signal by using a separate wire for each bit of the word. In the PC industry, these are known as Parallel devices.

The standard serial communications hardware in the PC does not support Synchronous operations. This mode is described here for comparison purposes only.

73.1.2 Asynchronous Serial Transmission

Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, the sender and receiver must agree on timing parameters in advance and special bits are added to each word which are used to synchronize the sending and receiving units.

When a word is given to the UART for Asynchronous transmissions, a bit called the “Start Bit” is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter. These two clocks must be accurate enough to not have the frequency drift by more than 10% during the transmission of the remaining bits in the word. (This requirement was set in the days of mechanical teleprinters and is easily met by modern electronic equipment.)

After the Start Bit, the individual bits of the word of data are sent, with the Least Significant Bit (LSB) being sent first. Each bit in the transmission is transmitted for exactly the same amount of time as all of the other bits, and the receiver “looks” at the wire at approximately halfway through the period assigned to each bit to determine if the bit is a 1 or a 0. For example, if it takes two seconds to send each bit, the receiver will examine the signal to determine if it is a 1 or a 0 after one second has passed, then it will wait two seconds and then examine the value of the next bit, and so on.

The sender does not know when the receiver has “looked” at the value of the bit. The sender only knows when the clock says to begin transmitting the next bit of the word.

When the entire data word has been sent, the transmitter may add a Parity Bit that the transmitter generates. The Parity Bit may be used by the receiver to perform simple error checking. Then at least one Stop Bit is sent by the transmitter.

When the receiver has received all of the bits in the data word, it may check for the Parity Bits (both sender and receiver must agree on whether a Parity Bit is to be used), and then the receiver looks for a Stop Bit. If the Stop Bit does not appear when it is supposed to, the UART considers the entire word to be garbled and will report a Framing Error to the host processor when the data word is read. The usual cause of a Framing Error is that the sender and receiver clocks were not running at the same speed, or that the signal was interrupted.

Regardless of whether the data was received correctly or not, the UART automatically discards the Start, Parity and Stop bits. If the sender and receiver are configured identically, these bits are not passed to the host.

If another word is ready for transmission, the Start Bit for the new word can be sent as soon as the Stop Bit for the previous word has been sent.

Because asynchronous data is “self synchronizing”, if there is no data to transmit, the transmission line can be idle.

73.1.3 Other UART Functions

In addition to the basic job of converting data from parallel to serial for transmission and from serial to parallel on reception, a UART will usually provide additional circuits for signals that can be used to indicate the state of the transmission media, and to regulate the flow of data in the event that the remote device is not prepared to accept more data. For example, when the device connected to the UART is a modem, the modem may report the presence of a carrier on the phone line while the computer may be able to instruct the modem to reset itself or to not take calls by raising or lowering one more of these extra signals. The function of each of these additional signals is defined in the EIA RS232-C standard.

73.1.4 The RS232-C and V.24 Standards

In most computer systems, the UART is connected to circuitry that generates signals that comply with the EIA RS232-C specification. There is also a CCITT standard named V.24 that mirrors the specifications included in RS232-C.

RS232-C Bit Assignments (Marks and Spaces)

In RS232-C, a value of 1 is called a `Mark` and a value of 0 is called a `Space`. When a communication line is idle, the line is said to be “Marking”, or transmitting continuous 1 values.

The Start bit always has a value of 0 (a `Space`). The Stop Bit always has a value of 1 (a `Mark`). This means that there will always be a `Mark` (1) to `Space` (0) transition on the line at the start of every word, even when multiple word are transmitted back to back. This guarantees that sender and receiver can resynchronize their clocks regardless of the content of the data bits that are being transmitted.

The idle time between Stop and Start bits does not have to be an exact multiple (including zero) of the bit rate of the communication link, but most UARTs are designed this way for simplicity.

In RS232-C, the “Marking” signal (a 1) is represented by a voltage between -2 VDC and -12 VDC, and a “Spacing” signal (a 0) is represented by a voltage between 0 and +12 VDC. The transmitter is supposed to send +12 VDC or -12 VDC, and the receiver is supposed to allow for some voltage loss in long cables. Some transmitters in low power devices (like portable computers) sometimes use only +5 VDC and -5 VDC, but these values are still acceptable to a RS232-C receiver, provided that the cable lengths are short.

RS232-C Break Signal

RS232-C also specifies a signal called a `Break`, which is caused by sending continuous `Spacing` values (no Start or Stop bits). When there is no electricity present on the data circuit, the line is considered to be sending `Break`.

The `Break` signal must be of a duration longer than the time it takes to send a complete byte plus Start, Stop and Parity bits. Most UARTs can distinguish between a Framing Error and a `Break`, but if the UART cannot do this, the Framing Error detection can be used to identify `Breaks`.

In the days of teleprinters, when numerous printers around the country were wired in series (such as news services), any unit could cause a `Break` by temporarily opening the entire circuit so that no current flowed. This was used to allow a location with urgent news to interrupt some other location that was currently sending information.

In modern systems there are two types of `Break` signals. If the `Break` is longer than 1.6 seconds, it is considered a “Modem `Break`”, and some modems can be programmed to terminate the conversation and go on-hook or enter the modems’ command mode when the modem detects this signal. If the `Break` is smaller than 1.6 seconds, it signifies a `Data Break` and it is up to the remote computer to respond to this signal. Sometimes this form of `Break` is used as an Attention or Interrupt signal and sometimes is accepted as a substitute for the ASCII `CONTROL-C` character.

Marks and Spaces are also equivalent to “Holes” and “No Holes” in paper tape systems.

Note

Breaks cannot be generated from paper tape or from any other byte value, since bytes are always sent with Start and Stop bit. The UART is usually capable of generating the continuous `Spacing` signal in response to a special command from the host processor.

RS232-C DTE and DCE Devices

The RS232-C specification defines two types of equipment: the Data Terminal Equipment (DTE) and the Data Carrier Equipment (DCE). Usually, the DTE device is the terminal (or computer), and the DCE is a modem. Across the phone line at the other end of a conversation, the receiving modem is also a DCE device and the computer that is connected to that modem is a DTE device. The DCE device receives signals on the pins that the DTE device transmits on, and vice versa.

When two devices that are both DTE or both DCE must be connected together without a modem or a similar media translator between them, a `NULL` modem must be used. The `NULL` modem electrically re-arranges the cabling so that the transmitter output is connected to the receiver input on the other device, and vice versa. Similar translations

are performed on all of the control signals so that each device will see what it thinks are DCE (or DTE) signals from the other device.

The number of signals generated by the DTE and DCE devices are not symmetrical. The DTE device generates fewer signals for the DCE device than the DTE device receives from the DCE.

RS232-C Pin Assignments

The EIA RS232-C specification (and the ITU equivalent, V.24) calls for a twenty-five pin connector (usually a DB25) and defines the purpose of most of the pins in that connector.

In the IBM Personal Computer and similar systems, a subset of RS232-C signals are provided via nine pin connectors (DB9). The signals that are not included on the PC connector deal mainly with synchronous operation, and this transmission mode is not supported by the UART that IBM selected for use in the IBM PC.

Depending on the computer manufacturer, a DB25, a DB9, or both types of connector may be used for RS232-C communications. (The IBM PC also uses a DB25 connector for the parallel printer interface which causes some confusion.)

Below is a table of the RS232-C signal assignments in the DB25 and DB9 connectors.

DB25 RS232-C Pin	DB9 IBM PC Pin	EIA Circuit Symbol	CCITT Circuit Symbol	Common Name	Signal Source	Description
1	.	AA	101	PG/FG	.	Frame/Protective Ground
2	3	BA	103	TD	DTE	Transmit Data
3	2	BB	104	RD	DCE	Receive Data
4	7	CA	105	RTS	DTE	Request to Send
5	8	CB	106	CTS	DCE	Clear to Send
6	6	CC	107	DSR	DCE	Data Set Ready
7	5	AV	102	SG/GND	.	Signal Ground
8	1	CF	109	DCD/CD	DCE	Data Carrier Detect
9	Reserved for Test
10	Reserved for Test
11	Reserved for Test
12	.	CI	122	SRLSD	DCE	Sec. Recv. Line Signal Detector
13	.	SCB	121	SCTS	DCE	Secondary Clear to Send
14	.	SBA	118	STD	DTE	Secondary Transmit Data
15	.	DB	114	TSET	DCE	Trans. Sig. Element Timing
16	.	SBB	119	SRD	DCE	Secondary Received Data
17	.	DD	115	RSET	DCE	Receiver Signal Element Timing
18	.	.	141	LOOP	DTE	Local Loopback
19	.	SCA	120	SRS	DTE	Secondary Request to Send
20	4	CD	108.2	DTR	DTE	Data Terminal Ready
21	.	.	.	RDL	DTE	Remote Digital Loopback
22	9	CE	125	RI	DCE	Ring Indicator
23	.	CH	111	DSRS	DTE	Data Signal Rate Selector
24	.	DA	113	TSET	DTE	Trans. Sig. Element Timing
25	.	.	142	.	DCE	Test Mode

73.1.5 Bits, Baud and Symbols

Baud is a measurement of transmission speed in asynchronous communication. Because of advances in modem communication technology, this term is frequently misused when describing the data rates in newer devices.

Traditionally, a Baud Rate represents the number of bits that are actually being sent over the media, not the amount of data that is actually moved from one DTE device to the other. The Baud count includes the overhead bits Start, Stop and Parity that are generated by the sending UART and removed by the receiving UART. This means that seven-bit words of data actually take 10 bits to be completely transmitted. Therefore, a modem capable of moving 300 bits per second from one place to another can normally only move 30 7-bit words if Parity is used and one Start and Stop bit are present.

If 8-bit data words are used and Parity bits are also used, the data rate falls to 27.27 words per second, because it now takes 11 bits to send the eight-bit words, and the modem still only sends 300 bits per second.

The formula for converting bytes per second into a baud rate and vice versa was simple until error-correcting modems came along. These modems receive the serial stream of bits from the UART in the host computer (even when internal modems are used the data is still frequently serialized) and converts the bits back into bytes. These bytes are then combined into packets and sent over the phone line using a Synchronous transmission method. This means that the Stop, Start, and Parity bits added by the UART in the DTE (the computer) were removed by the modem before transmission by the sending modem. When these bytes are received by the remote modem, the remote modem adds Start, Stop and Parity bits to the words, converts them to a serial format and then sends them to the receiving UART in the remote computer, who then strips the Start, Stop and Parity bits.

The reason all these extra conversions are done is so that the two modems can perform error correction, which means that the receiving modem is able to ask the sending modem to resend a block of data that was not received with the correct checksum. This checking is handled by the modems, and the DTE devices are usually unaware that the process is occurring.

By stripping the Start, Stop and Parity bits, the additional bits of data that the two modems must share between themselves to perform error-correction are mostly concealed from the effective transmission rate seen by the sending and receiving DTE equipment. For example, if a modem sends ten 7-bit words to another modem without including the Start, Stop and Parity bits, the sending modem will be able to add 30 bits of its own information that the receiving modem can use to do error-correction without impacting the transmission speed of the real data.

The use of the term Baud is further confused by modems that perform compression. A single 8-bit word passed over the telephone line might represent a dozen words that were transmitted to the sending modem. The receiving modem will expand the data back to its original content and pass that data to the receiving DTE.

Modern modems also include buffers that allow the rate that bits move across the phone line (DCE to DCE) to be a different speed than the speed that the bits move between the DTE and DCE on both ends of the conversation. Normally the speed between the DTE and DCE is higher than the DCE to DCE speed because of the use of compression by the modems.

Because the number of bits needed to describe a byte varied during the trip between the two machines plus the differing bits-per-second speeds that are used present on the DTE-DCE and DCE-DCE links, the usage of the term Baud to describe the overall communication speed causes problems and can misrepresent the true transmission speed. So Bits Per Second (bps) is the correct term to use to describe the transmission rate seen at the DCE to DCE interface and Baud or Bits Per Second are acceptable terms to use when a connection is made between two systems with a wired connection, or if a modem is in use that is not performing error-correction or compression.

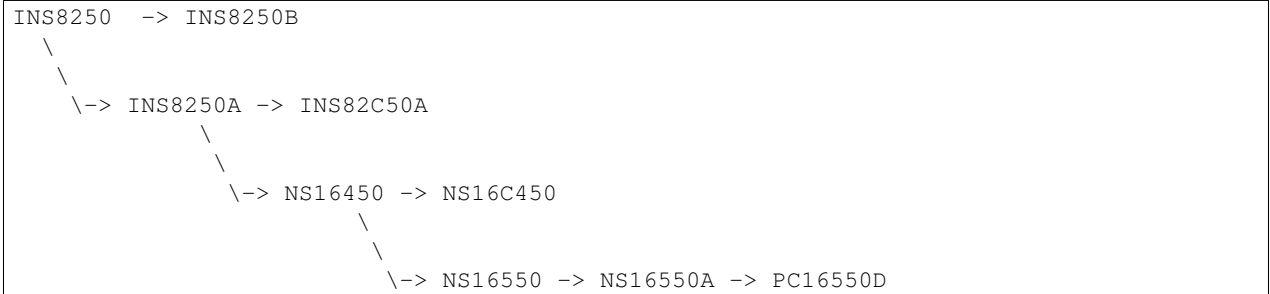
Modern high speed modems (2400, 9600, 14,400, and 19,200bps) in reality still operate at or below 2400 baud, or more accurately, 2400 Symbols per second. High speed modem are able to encode more bits of data into each Symbol using a technique called Constellation Stuffing, which is why the effective bits per second rate of the modem is higher, but the modem continues to operate within the limited audio bandwidth that the telephone system provides. Modems operating at 28,800 and higher speeds have variable Symbol rates, but the technique is the same.

73.1.6 The IBM Personal Computer UART

Starting with the original IBM Personal Computer, IBM selected the National Semiconductor INS8250 UART for use in the IBM PC Parallel/Serial Adapter. Subsequent generations of compatible computers from IBM and other vendors continued to use the INS8250 or improved versions of the National Semiconductor UART family.

National Semiconductor UART Family Tree

There have been several versions and subsequent generations of the INS8250 UART. Each major version is described below.



INS8250 This part was used in the original IBM PC and IBM PC/XT. The original name for this part was the INS8250 ACE (Asynchronous Communications Element) and it is made from NMOS technology.

The 8250 uses eight I/O ports and has a one-byte send and a one-byte receive buffer. This original UART has several race conditions and other flaws. The original IBM BIOS includes code to work around these flaws, but this made the BIOS dependent on the flaws being present, so subsequent parts like the 8250A, 16450 or 16550 could not be used in the original IBM PC or IBM PC/XT.

INS8250-B This is the slower speed of the INS8250 made from NMOS technology. It contains the same problems as the original INS8250.

INS8250A An improved version of the INS8250 using XMOS technology with various functional flaws corrected. The INS8250A was used initially in PC clone computers by vendors who used “clean” BIOS designs. Because of the corrections in the chip, this part could not be used with a BIOS compatible with the INS8250 or INS8250B.

INS82C50A This is a CMOS version (low power consumption) of the INS8250A and has similar functional characteristics.

NS16450 Same as NS8250A with improvements so it can be used with faster CPU bus designs. IBM used this part in the IBM AT and updated the IBM BIOS to no longer rely on the bugs in the INS8250.

NS16C450 This is a CMOS version (low power consumption) of the NS16450.

NS16550 Same as NS16450 with a 16-byte send and receive buffer but the buffer design was flawed and could not be reliably be used.

NS16550A Same as NS16550 with the buffer flaws corrected. The 16550A and its successors have become the most popular UART design in the PC industry, mainly due to its ability to reliably handle higher data rates on operating systems with sluggish interrupt response times.

NS16C552 This component consists of two NS16C550A CMOS UARTs in a single package.

PC16550D Same as NS16550A with subtle flaws corrected. This is revision D of the 16550 family and is the latest design available from National Semiconductor.

The NS16550AF and the PC16550D are the same thing

National reorganized their part numbering system a few years ago, and the NS16550AFN no longer exists by that name. (If you have a NS16550AFN, look at the date code on the part, which is a four digit number that usually starts with a nine. The first two digits of the number are the year, and the last two digits are the week in that year when the part was packaged. If you have a NS16550AFN, it is probably a few years old.)

The new numbers are like PC16550DV, with minor differences in the suffix letters depending on the package material and its shape. (A description of the numbering system can be found below.)

It is important to understand that in some stores, you may pay \$15(US) for a NS16550AFN made in 1990 and in the next bin are the new PC16550DN parts with minor fixes that National has made since the AFN part was in production, the PC16550DN was probably made in the past six months and it costs half (as low as \$5(US) in volume) as much as the NS16550AFN because they are readily available.

As the supply of NS16550AFN chips continues to shrink, the price will probably continue to increase until more people discover and accept that the PC16550DN really has the same function as the old part number.

National Semiconductor Part Numbering System

The older NSnnnnnrqp part numbers are now of the format PCnnnnnrqp.

The r is the revision field. The current revision of the 16550 from National Semiconductor is D.

The p is the package-type field. The types are:

“F”	QFP	(quad flat pack) L lead type
“N”	DIP	(dual inline package) through hole straight lead type
“V”	LPCC	(lead plastic chip carrier) J lead type

The g is the product grade field. If an I precedes the package-type letter, it indicates an “industrial” grade part, which has higher specs than a standard part but not as high as Military Specification (Milspec) component. This is an optional field.

So what we used to call a NS16550AFN (DIP Package) is now called a PC16550DN or PC16550DIN.

73.1.7 Other Vendors and Similar UARTs

Over the years, the 8250, 8250A, 16450 and 16550 have been licensed or copied by other chip vendors. In the case of the 8250, 8250A and 16450, the exact circuit (the “megacell”) was licensed to many vendors, including Western Digital and Intel. Other vendors reverse-engineered the part or produced emulations that had similar behavior.

In internal modems, the modem designer will frequently emulate the 8250A/16450 with the modem microprocessor, and the emulated UART will frequently have a hidden buffer consisting of several hundred bytes. Because of the size of the buffer, these emulations can be as reliable as a 16550A in their ability to handle high speed data. However, most operating systems will still report that the UART is only a 8250A or 16450, and may not make effective use of the extra buffering present in the emulated UART unless special drivers are used.

Some modem makers are driven by market forces to abandon a design that has hundreds of bytes of buffer and instead use a 16550A UART so that the product will compare favorably in market comparisons even though the effective performance may be lowered by this action.

A common misconception is that all parts with “16550A” written on them are identical in performance. There are differences, and in some cases, outright flaws in most of these 16550A clones.

When the NS16550 was developed, the National Semiconductor obtained several patents on the design and they also limited licensing, making it harder for other vendors to provide a chip with similar features. Because of the patents, reverse-engineered designs and emulations had to avoid infringing the claims covered by the patents. Subsequently,

these copies almost never perform exactly the same as the NS16550A or PC16550D, which are the parts most computer and modem makers want to buy but are sometimes unwilling to pay the price required to get the genuine part.

Some of the differences in the clone 16550A parts are unimportant, while others can prevent the device from being used at all with a given operating system or driver. These differences may show up when using other drivers, or when particular combinations of events occur that were not well tested or considered in the WINDOWS driver. This is because most modem vendors and 16550-clone makers use the Microsoft drivers from WINDOWS for Workgroups 3.11 and the MICROSOFT MS-DOS utility as the primary tests for compatibility with the NS16550A. This oversimplistic criteria means that if a different operating system is used, problems could appear due to subtle differences between the clones and genuine components.

National Semiconductor has made available a program named COMTEST that performs compatibility tests independent of any OS drivers. It should be remembered that the purpose of this type of program is to demonstrate the flaws in the products of the competition, so the program will report major as well as extremely subtle differences in behavior in the part being tested.

In a series of tests performed by the author of this document in 1994, components made by National Semiconductor, TI, StarTech, and CMD as well as megacells and emulations embedded in internal modems were tested with COMTEST. A difference count for some of these components is listed below. Because these tests were performed in 1994, they may not reflect the current performance of the given product from a vendor.

It should be noted that COMTEST normally aborts when an excessive number or certain types of problems have been detected. As part of this testing, COMTEST was modified so that it would not abort no matter how many differences were encountered.

Vendor	Part Number	Errors (aka "differences" reported)
National	(PC16550DV)	0
National	(NS16550AFN)	0
National	(NS16C552V)	0
TI	(TL16550AFN)	3
CMD	(16C550PE)	19
StarTech	(ST16C550J)	23
Rockwell	Reference modem with internal 16550 or an emulation (RC144DPi/C3000-25)	117
Sierra	Modem with an internal 16550 (SC11951/SC11351)	91

Note

To date, the author of this document has not found any non-National parts that report zero differences using the COMTEST program. It should also be noted that National has had five versions of the 16550 over the years and the newest parts behave a bit differently than the classic NS16550AFN that is considered the benchmark for functionality. COMTEST appears to turn a blind eye to the differences within the National product line and reports no errors on the National parts (except for the original 16550) even when there are official erratas that describe bugs in the A, B and C revisions of the parts, so this bias in COMTEST must be taken into account.

It is important to understand that a simple count of differences from COMTEST does not reveal a lot about what differences are important and which are not. For example, about half of the differences reported in the two modems listed above that have internal UARTs were caused by the clone UARTs not supporting five- and six-bit character modes. The real 16550, 16450, and 8250 UARTs all support these modes and COMTEST checks the functionality of these modes so over fifty differences are reported. However, almost no modern modem supports five- or six-bit characters, particularly those with error-correction and compression capabilities. This means that the differences related to five- and six-bit character modes can be discounted.

Many of the differences COMTEST reports have to do with timing. In many of the clone designs, when the host reads from one port, the status bits in some other port may not update in the same amount of time (some faster, some slower) as a *real* NS16550AFN and COMTEST looks for these differences. This means that the number of differences can be misleading in that one device may only have one or two differences but they are extremely serious, and some other device that updates the status registers faster or slower than the reference part (that would probably never affect the operation of a properly written driver) could have dozens of differences reported.

COMTEST can be used as a screening tool to alert the administrator to the presence of potentially incompatible components that might cause problems or have to be handled as a special case.

If you run COMTEST on a 16550 that is in a modem or a modem is attached to the serial port, you need to first issue a ATE0&W command to the modem so that the modem will not echo any of the test characters. If you forget to do this, COMTEST will report at least this one difference:

```
Error (6)...Timeout interrupt failed: IIR = c1  LSR = 61
```

73.1.8 8250/16450/16550 Registers

The 8250/16450/16550 UART occupies eight contiguous I/O port addresses. In the IBM PC, there are two defined locations for these eight ports and they are known collectively as COM1 and COM2. The makers of PC-clones and add-on cards have created two additional areas known as COM3 and COM4, but these extra COM ports conflict with other hardware on some systems. The most common conflict is with video adapters that provide IBM 8514 emulation.

COM1 is located from 0x3f8 to 0x3ff and normally uses IRQ 4. COM2 is located from 0x2f8 to 0x2ff and normally uses IRQ 3. COM3 is located from 0x3e8 to 0x3ef and has no standardized IRQ. COM4 is located from 0x2e8 to 0x2ef and has no standardized IRQ.

A description of the I/O ports of the 8250/16450/16550 UART is provided below.

73.1.9 Beyond the 16550A UART

Although National Semiconductor has not offered any components compatible with the 16550 that provide additional features, various other vendors have. Some of these components are described below. It should be understood that to effectively utilize these improvements, drivers may have to be provided by the chip vendor since most of the popular operating systems do not support features beyond those provided by the 16550.

ST16650 By default this part is similar to the NS16550A, but an extended 32-byte send and receive buffer can be optionally enabled. Made by StarTech.

TIL16660 By default this part behaves similar to the NS16550A, but an extended 64-byte send and receive buffer can be optionally enabled. Made by Texas Instruments.

Hayes ESP This proprietary plug-in card contains a 2048-byte send and receive buffer, and supports data rates to 230.4Kbit/sec. Made by Hayes.

In addition to these “dumb” UARTs, many vendors produce intelligent serial communication boards. This type of design usually provides a microprocessor that interfaces with several UARTs, processes and buffers the data, and then alerts the main PC processor when necessary. Because the UARTs are not directly accessed by the PC processor in this type of communication system, it is not necessary for the vendor to use UARTs that are compatible with the 8250, 16450, or the 16550 UART. This leaves the designer free to components that may have better performance characteristics.

73.2 Configuring the `sio` driver

The `sio` driver provides support for NS8250-, NS16450-, NS16550 and NS16550A-based EIA RS-232C (CCITT V.24) communications interfaces. Several multiport cards are supported as well. See the MAN.SIO.4 manual page for detailed technical documentation.

73.2.1 Digi International (DigiBoard) PC/8

Contributed by A.AWEBSTER.EMAIL. 26 August 1995.

Here is a config snippet from a machine with a Digi International PC/8 with 16550. It has 8 modems connected to these 8 lines, and they work just great. Do not forget to add `options COM_MULTIPOINT` or it will not work very well!

```
device      sio4      at isa? port 0x100 flags 0xb05
device      sio5      at isa? port 0x108 flags 0xb05
device      sio6      at isa? port 0x110 flags 0xb05
device      sio7      at isa? port 0x118 flags 0xb05
device      sio8      at isa? port 0x120 flags 0xb05
device      sio9      at isa? port 0x128 flags 0xb05
device      sio10     at isa? port 0x130 flags 0xb05
device      sio11     at isa? port 0x138 flags 0xb05 irq 9
```

The trick in setting this up is that the MSB of the flags represent the last SIO port, in this case 11 so flags are 0xb05.

73.2.2 Boca 16

Contributed by A.WHITESIDE.EMAIL. 26 August 1995.

The procedures to make a Boca 16 port board with FreeBSD are pretty straightforward, but you will need a couple things to make it work:

1. You either need the kernel sources installed so you can recompile the necessary options or you will need someone else to compile it for you. The 2.0.5 default kernel does *not* come with multiport support enabled and you will need to add a device entry for each port anyways.
2. Two, you will need to know the interrupt and IO setting for your Boca Board so you can set these options properly in the kernel.

One important note — the actual UART chips for the Boca 16 are in the connector box, not on the internal board itself. So if you have it unplugged, probes of those ports will fail. I have never tested booting with the box unplugged and plugging it back in, and I suggest you do not either.

If you do not already have a custom kernel configuration file set up, refer to Kernel Configuration chapter of the FreeBSD Handbook for general procedures. The following are the specifics for the Boca 16 board and assume you are using the kernel name MYKERNEL and editing with `vi`.

Add the line

```
options COM_MULTIPOINT
```

to the config file.

Where the current “`device sion`” lines are, you will need to add 16 more devices. The

following example is for a Boca Board with an interrupt of 3, and a base IO address 100h. The IO address for Each port is +8 hexadecimal from the previous port, thus the 100h, 108h, 110h... addresses.

```
device sio1 at isa? port 0x100 flags 0x1005
device sio2 at isa? port 0x108 flags 0x1005
device sio3 at isa? port 0x110 flags 0x1005
device sio4 at isa? port 0x118 flags 0x1005
...
device sio15 at isa? port 0x170 flags 0x1005
device sio16 at isa? port 0x178 flags 0x1005 irq 3
```

The flags entry *must* be changed from this example unless you are using the exact same sio assignments. Flags are set according to 0xMYYY where M indicates the minor number of the master port (the last port on a Boca 16) and YY indicates if FIFO is enabled or disabled(enabled), IRQ sharing is used(yes) and if there is an AST/4 compatible IRQ control register(no). In this example,

```
flags
    0x1005
```

indicates that the master port is sio16. If I added another board and assigned sio17 through sio28, the flags for all 16 ports on *that* board would be 0x1C05, where 1C indicates the minor number of the master port. Do not change the 05 setting.

Save and complete the kernel configuration, recompile, install and reboot. Presuming you have successfully installed the recompiled kernel and have it set to the correct address and IRQ, your boot message should indicate the successful probe of the Boca ports as follows: (obviously the sio numbers, IO and IRQ could be different)

```
sio1 at 0x100-0x107 flags 0x1005 on isa
sio1: type 16550A (multiport)
sio2 at 0x108-0x10f flags 0x1005 on isa
sio2: type 16550A (multiport)
sio3 at 0x110-0x117 flags 0x1005 on isa
sio3: type 16550A (multiport)
sio4 at 0x118-0x11f flags 0x1005 on isa
sio4: type 16550A (multiport)
sio5 at 0x120-0x127 flags 0x1005 on isa
sio5: type 16550A (multiport)
sio6 at 0x128-0x12f flags 0x1005 on isa
sio6: type 16550A (multiport)
sio7 at 0x130-0x137 flags 0x1005 on isa
sio7: type 16550A (multiport)
sio8 at 0x138-0x13f flags 0x1005 on isa
sio8: type 16550A (multiport)
sio9 at 0x140-0x147 flags 0x1005 on isa
sio9: type 16550A (multiport)
sio10 at 0x148-0x14f flags 0x1005 on isa
sio10: type 16550A (multiport)
sio11 at 0x150-0x157 flags 0x1005 on isa
sio11: type 16550A (multiport)
sio12 at 0x158-0x15f flags 0x1005 on isa
sio12: type 16550A (multiport)
sio13 at 0x160-0x167 flags 0x1005 on isa
sio13: type 16550A (multiport)
sio14 at 0x168-0x16f flags 0x1005 on isa
sio14: type 16550A (multiport)
sio15 at 0x170-0x177 flags 0x1005 on isa
sio15: type 16550A (multiport)
sio16 at 0x178-0x17f irq 3 flags 0x1005 on isa
sio16: type 16550A (multiport master)
```

If the messages go by too fast to see,


```
PROMPT.ROOT dmesg | more
```

will show you the boot messages.

Next, appropriate entries in `/dev` for the devices must be made using the `/dev/MAKEDEV` script. This step can be omitted if you are running FreeBSD 5.X with a kernel that has `MAN.DEVFS.5` support compiled in.

If you do need to create the `/dev` entries, run the following as root:

```
PROMPT.ROOT cd /dev
PROMPT.ROOT ./MAKEDEV tty1
PROMPT.ROOT ./MAKEDEV cua1
(everything in between)
PROMPT.ROOT ./MAKEDEV ttyg
PROMPT.ROOT ./MAKEDEV cuag
```

If you do not want or need call-out devices for some reason, you can dispense with making the `cua*` devices.

If you want a quick and sloppy way to make sure the devices are working, you can simply plug a modem into each port and (as root)

```
PROMPT.ROOT echo at > ttyd*
```

for each device you have made. You *should* see the RX lights flash for each working port.

73.2.3 Support for Cheap Multi-UART Cards

Contributed by Helge Oldach hmo@sep.hamburg.com, September 1999

Ever wondered about FreeBSD support for your 20\$ multi-I/O card with two (or more) COM ports, sharing IRQs? Here is how:

Usually the only option to support these kind of boards is to use a distinct IRQ for each port. For example, if your CPU board has an on-board COM1 port (aka `si00`—I/O address 0x3F8 and IRQ 4) and you have an extension board with two UARTs, you will commonly need to configure them as COM2 (aka `si01`—I/O address 0x2F8 and IRQ 3), and the third port (aka `si02`) as I/O 0x3E8 and IRQ 5. Obviously this is a waste of IRQ resources, as it should be basically possible to run both extension board ports using a single IRQ with the `COM_MULTIPORT` configuration described in the previous sections.

Such cheap I/O boards commonly have a 4 by 3 jumper matrix for the COM ports, similar to the following:

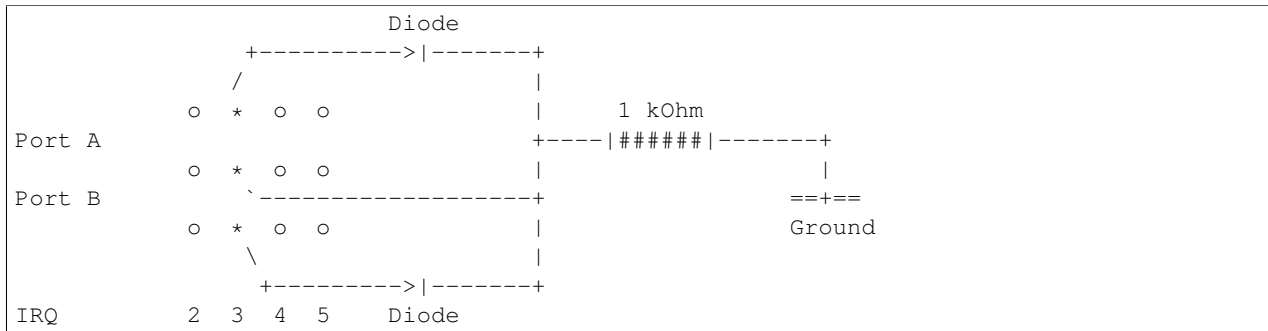
	o	o	o	*
Port A				
	o	*	o	*
Port B				
	o	*	o	o
IRQ	2	3	4	5

Shown here is port A wired for IRQ 5 and port B wired for IRQ 3. The IRQ columns on your specific board may vary—other boards may supply jumpers for IRQs 3, 4, 5, and 7 instead.

One could conclude that wiring both ports for IRQ 3 using a handcrafted wire-made jumper covering all three connection points in the IRQ 3 column would solve the issue, but no. You cannot duplicate IRQ 3 because the output drivers of each UART are wired in a “totem pole” fashion, so if one of the UARTs drives IRQ 3, the output signal will not be what you would expect. Depending on the implementation of the extension board or your motherboard, the IRQ 3 line will continuously stay up, or always stay low.

You need to decouple the IRQ drivers for the two UARTs, so that the IRQ line of the board only goes up (and only if) one of the UARTs asserts a IRQ, and stays low otherwise. The solution was proposed by Joerg Wunsch

j@ida.interface-business.de: To solder up a wired-or consisting of two diodes (Germanium or Schottky-types strongly preferred) and a 1 kOhm resistor. Here is the schematic, starting from the 4 by 3 jumper field above:



The cathodes of the diodes are connected to a common point, together with a 1 kOhm pull-down resistor. It is essential to connect the resistor to ground to avoid floating of the IRQ line on the bus.

Now we are ready to configure a kernel. Staying with this example, we would configure:

```
# standard on-board COM1 port
device      sio0      at isa? port "IO_COM1" flags 0x10
# patched-up multi-I/O extension board
options     COM_MULTIPOINT
device      sio1      at isa? port "IO_COM2" flags 0x205
device      sio2      at isa? port "IO_COM3" flags 0x205 irq 3
```

Note that the flags setting for `sio1` and `sio2` is truly essential; refer to MAN.SIO.4 for details. (Generally, the 2 in the “flags” attribute refers to `sio2` which holds the IRQ, and you surely want a 5 low nibble.) With kernel verbose mode turned on this should yield something similar to this:

```
sio0: irq maps: 0x1 0x11 0x1 0x1
sio0 at 0x3f8-0x3ff irq 4 flags 0x10 on isa
sio0: type 16550A
sio1: irq maps: 0x1 0x9 0x1 0x1
sio1 at 0x2f8-0x2ff flags 0x205 on isa
sio1: type 16550A (multiport)
sio2: irq maps: 0x1 0x9 0x1 0x1
sio2 at 0x3e8-0x3ef irq 3 flags 0x205 on isa
sio2: type 16550A (multiport master)
```

Though `/sys/i386/isa/sio.c` is somewhat cryptic with its use of the “irq maps” array above, the basic idea is that you observe 0x1 in the first, third, and fourth place. This means that the corresponding IRQ was set upon output and cleared after, which is just what we would expect. If your kernel does not display this behavior, most likely there is something wrong with your wiring.

73.3 Configuring the `cy` driver

Contributed by Alex Nash. 6 June 1996.

The Cyclades multiport cards are based on the `cy` driver instead of the usual `sio` driver used by other multiport cards. Configuration is a simple matter of:

Add the `cy` device to your kernel configuration (note that your irq and iomem settings may differ).

```
device cy0 at isa? irq 10 iomem 0xd4000 iosiz 0x2000
```

Rebuild and install the new kernel.

Make the device nodes by typing (the following example assumes an 8-port board) ¹:

```
PROMPT.ROOT cd /dev
PROMPT.ROOT for i in 0 1 2 3 4 5 6 7;do ./MAKEDEV cuac$i ttyc$i;done
```

If appropriate, add dialup entries to `/etc/ttys` by duplicating serial device (`ttyd`) entries and using `ttyc` in place of `ttyd`. For example:

```
ttyc0  "/usr/libexec/getty std.38400"  unknown on insecure
ttyc1  "/usr/libexec/getty std.38400"  unknown on insecure
ttyc2  "/usr/libexec/getty std.38400"  unknown on insecure
...
ttyc7  "/usr/libexec/getty std.38400"  unknown on insecure
```

Reboot with the new kernel.

73.4 Configuring the `si` driver

Contributed by A.NSAYER.EMAIL. 25 March 1998.

The Specialix SI/XIO and SX multiport cards use the `si` driver. A single machine can have up to 4 host cards. The following host cards are supported:

- ISA SI/XIO host card (2 versions)
- EISA SI/XIO host card
- PCI SI/XIO host card
- ISA SX host card
- PCI SX host card

Although the SX and SI/XIO host cards look markedly different, their functionality are basically the same. The host cards do not use I/O locations, but instead require a 32K chunk of memory. The factory configuration for ISA cards places this at `0xd0000-0xd7fff`. They also require an IRQ. PCI cards will, of course, auto-configure themselves.

You can attach up to 4 external modules to each host card. The external modules contain either 4 or 8 serial ports. They come in the following varieties:

- SI 4 or 8 port modules. Up to 57600 bps on each port supported.
- XIO 8 port modules. Up to 115200 bps on each port supported. One type of XIO module has 7 serial and 1 parallel port.
- SXDC 8 port modules. Up to 921600 bps on each port supported. Like XIO, a module is available with one parallel port as well.

To configure an ISA host card, add the following line to your kernel configuration file, changing the numbers as appropriate:

```
device si0 at isa? iomem 0xd0000 irq 11
```

Valid IRQ numbers are 9, 10, 11, 12 and 15 for SX ISA host cards and 11, 12 and 15 for SI/XIO ISA host cards.

To configure an EISA or PCI host card, use this line:

```
device si0
```

After adding the configuration entry, rebuild and install your new kernel.

¹ You can omit this part if you are running FreeBSD 5.X with `MAN.DEVFS.5`.

Note

The following step, is not necessary if you are using MAN.DEVFS.5 in FreeBSD 5.X.

After rebooting with the new kernel, you need to make the device nodes in /dev. The MAKEDEV script will take care of this for you. Count how many total ports you have and type:

```
PROMPT.ROOT cd /dev
PROMPT.ROOT ./MAKEDEV ttyAnn cuaAnn
```

(where nn is the number of ports)

If you want login prompts to appear on these ports, you will need to add lines like this to /etc/ttys:

```
ttyA01  "/usr/libexec/getty std.9600"  vt100  on insecure
```

Change the terminal type as appropriate. For modems, dialup or unknown is fine.

OS and Solid State Devices

Author John Kozubik

74.1 Solid State Disk Devices

The scope of this article will be limited to solid state disk devices made from flash memory. Flash memory is a solid state memory (no moving parts) that is non-volatile (the memory maintains data even after all power sources have been disconnected). Flash memory can withstand tremendous physical shock and is reasonably fast (the flash memory solutions covered in this article are slightly slower than a EIDE hard disk for write operations, and much faster for read operations). One very important aspect of flash memory, the ramifications of which will be discussed later in this article, is that each sector has a limited rewrite capacity. You can only write, erase, and write again to a sector of flash memory a certain number of times before the sector becomes permanently unusable. Although many flash memory products automatically map bad blocks, and although some even distribute write operations evenly throughout the unit, the fact remains that there exists a limit to the amount of writing that can be done to the device. Competitive units have between 1,000,000 and 10,000,000 writes per sector in their specification. This figure varies due to the temperature of the environment.

Specifically, we will be discussing ATA compatible compact-flash units, which are quite popular as storage media for digital cameras. Of particular interest is the fact that they pin out directly to the IDE bus and are compatible with the ATA command set. Therefore, with a very simple and low-cost adaptor, these devices can be attached directly to an IDE bus in a computer. Once implemented in this manner, operating systems such as OS see the device as a normal hard disk (albeit small).

Other solid state disk solutions do exist, but their expense, obscurity, and relative unease of use places them beyond the scope of this article.

74.2 Kernel Options

A few kernel options are of specific interest to those creating an embedded OS system.

All embedded OS systems that use flash memory as system disk will be interested in memory disks and memory filesystems. Because of the limited number of writes that can be done to flash memory, the disk and the filesystems on the disk will most likely be mounted read-only. In this environment, filesystems such as `/tmp` and `/var` are mounted as memory filesystems to allow the system to create logs and update counters and temporary files. Memory filesystems are a critical component to a successful solid state OS implementation.

You should make sure the following lines exist in your kernel configuration file:

options	MFS	# Memory Filesystem
options	MD_ROOT	# md device usable as a potential root device
pseudo-device	md	# memory disk

74.3 The rc Subsystem and Read-Only Filesystems

The post-boot initialization of an embedded OS system is controlled by `/etc/rc.initdiskless`.

`/etc/rc.d/var` mounts `/var` as a memory filesystem, makes a configurable list of directories in `/var` with the `MAN.MKDIR.1` command, and changes modes on some of those directories. In the execution of `/etc/rc.d/var`, one other `rc.conf` variable comes into play – `varsize`. A `/var` partition is created by `/etc/rc.d/var` based on the value of this variable in `rc.conf`:

```
varsize=8192
```

Remember that this value is in sectors by default.

The fact that `/var` is a read-write filesystem is an important distinction, as the `/` partition (and any other partitions you may have on your flash media) should be mounted read-only. Remember that in ? we detailed the limitations of flash memory - specifically the limited write capability. The importance of not mounting filesystems on flash media read-write, and the importance of not using a swap file, cannot be overstated. A swap file on a busy system can burn through a piece of flash media in less than one year. Heavy logging or temporary file creation and destruction can do the same. Therefore, in addition to removing the `swap` entry from your `/etc/fstab`, you should also change the `Options` field for each filesystem to `ro` as follows:

#	Device	Mountpoint	FStype	Options	Dump	Pass#
/	/dev/ad0s1a	/	ufs	ro	1	1

A few applications in the average system will immediately begin to fail as a result of this change. For instance, `cron` will not run properly as a result of missing `cron` tabs in the `/var` created by `/etc/rc.d/var`, and `syslog` and `dhcp` will encounter problems as well as a result of the read-only filesystem and missing items in the `/var` that `/etc/rc.d/var` has created. These are only temporary problems though, and are addressed, along with solutions to the execution of other common software packages in ?.

An important thing to remember is that a filesystem that was mounted read-only with `/etc/fstab` can be made read-write at any time by issuing the command:

```
PROMPT.ROOT /sbin/mount -uw partition
```

and can be toggled back to read-only with the command:

```
PROMPT.ROOT /sbin/mount -ur partition
```

74.4 Building a File System from Scratch

Because ATA compatible compact-flash cards are seen by OS as normal IDE hard drives, you could theoretically install OS from the network using the `kern` and `mfsroot` floppies or from a CD.

However, even a small installation of OS using normal installation procedures can produce a system in size of greater than 200 megabytes. Because most people will be using smaller flash memory devices (128 megabytes is considered fairly large - 32 or even 16 megabytes is common) an installation using normal mechanisms is not possible—there is simply not enough disk space for even the smallest of conventional installations.

The easiest way to overcome this space limitation is to install OS using conventional means to a normal hard disk. After the installation is complete, pare down the operating system to a size that will fit onto your flash media, then tar the

entire filesystem. The following steps will guide you through the process of preparing a piece of flash memory for your tarred filesystem. Remember, because a normal installation is not being performed, operations such as partitioning, labeling, file-system creation, etc. need to be performed by hand. In addition to the kern and mfsroot floppy disks, you will also need to use the fixit floppy.

After booting with the kern and mfsroot floppies, choose `custom` from the installation menu. In the custom installation menu, choose `partition`. In the partition menu, you should delete all existing partitions using `d`. After deleting all existing partitions, create a partition using `c` and accept the default value for the size of the partition. When asked for the type of the partition, make sure the value is set to `165`. Now write this partition table to the disk by pressing `w` (this is a hidden option on this screen). If you are using an ATA compatible compact flash card, you should choose the OS Boot Manager. Now press `q` to quit the partition menu. You will be shown the boot manager menu once more - repeat the choice you made earlier.

Exit the custom installation menu, and from the main installation menu choose the `fixit` option. After entering the fixit environment, enter the following command:

```
PROMPT.ROOT disklabel -e /dev/ad0c
```

At this point you will have entered the vi editor under the auspices of the disklabel command. Next, you need to add an `a:` line at the end of the file. This `a:` line should look like:

```
a:      123456  0      4.2BSD  0      0
```

Where 123456 is a number that is exactly the same as the number in the existing `c:` entry for size. Basically you are duplicating the existing `c:` line as an `a:` line, making sure that `fstype` is `4.2BSD`. Save the file and exit.

```
PROMPT.ROOT disklabel -B -r /dev/ad0c
PROMPT.ROOT newfs /dev/ad0a
```

Mount the newly prepared flash media:

```
PROMPT.ROOT mount /dev/ad0a /flash
```

Bring this machine up on the network so we may transfer our tar file and explode it onto our flash media filesystem. One example of how to do this is:

```
PROMPT.ROOT ifconfig xl0 192.168.0.10 netmask 255.255.255.0
PROMPT.ROOT route add default 192.168.0.1
```

Now that the machine is on the network, transfer your tar file. You may be faced with a bit of a dilemma at this point - if your flash memory part is 128 megabytes, for instance, and your tar file is larger than 64 megabytes, you cannot have your tar file on the flash media at the same time as you explode it - you will run out of space. One solution to this problem, if you are using FTP, is to untar the file while it is transferred over FTP. If you perform your transfer in this manner, you will never have the tar file and the tar contents on your disk at the same time:

```
ftp> get tarfile.tar "| tar xvf -"
```

If your tarfile is gzipped, you can accomplish this as well:

```
ftp> get tarfile.tar "| zcat | tar xvf -"
```

After the contents of your tarred filesystem are on your flash memory filesystem, you can unmount the flash memory and reboot:

```
PROMPT.ROOT cd /
PROMPT.ROOT umount /flash
PROMPT.ROOT exit
```

Assuming that you configured your filesystem correctly when it was built on the normal hard disk (with your filesystems mounted read-only, and with the necessary options compiled into the kernel) you should now be successfully booting your OS embedded system.

74.5 System Strategies for Small and Read Only Environments

In [?](#), it was pointed out that the `/var` filesystem constructed by `/etc/rc.d/var` and the presence of a read-only root filesystem causes problems with many common software packages used with OS. In this article, suggestions for successfully running cron, syslog, ports installations, and the Apache web server will be provided.

74.5.1 Cron

Upon boot, `/var` gets populated by `/etc/rc.d/var` using the list from `/etc/mtree/BSD.var.dist`, so the `cron`, `cron/tabs`, `at`, and a few other standard directories get created.

However, this does not solve the problem of maintaining cron tabs across reboots. When the system reboots, the `/var` filesystem that is in memory will disappear and any cron tabs you may have had in it will also disappear. Therefore, one solution would be to create cron tabs for the users that need them, mount your `/` filesystem as read-write and copy those cron tabs to somewhere safe, like `/etc/tabs`, then add a line to the end of `/etc/rc.initdiskless` that copies those crontabs into `/var/cron/tabs` after that directory has been created during system initialization. You may also need to add a line that changes modes and permissions on the directories you create and the files you copy with `/etc/rc.initdiskless`.

74.5.2 Syslog

`syslog.conf` specifies the locations of certain log files that exist in `/var/log`. These files are not created by `/etc/rc.d/var` upon system initialization. Therefore, somewhere in `/etc/rc.d/var`, after the section that creates the directories in `/var`, you will need to add something like this:

```
PROMPT.ROOT touch /var/log/security /var/log/maillog /var/log/cron /var/log/messages
PROMPT.ROOT chmod 0644 /var/log/*
```

74.5.3 Ports Installation

Before discussing the changes necessary to successfully use the ports tree, a reminder is necessary regarding the read-only nature of your filesystems on the flash media. Since they are read-only, you will need to temporarily mount them read-write using the mount syntax shown in [?](#). You should always remount those filesystems read-only when you are done with any maintenance - unnecessary writes to the flash media could considerably shorten its lifespan.

To make it possible to enter a ports directory and successfully run `make install`, we must create a packages directory on a non-memory filesystem that will keep track of our packages across reboots. Because it is necessary to mount your filesystems as read-write for the installation of a package anyway, it is sensible to assume that an area on the flash media can also be used for package information to be written to.

First, create a package database directory. This is normally in `/var/db/pkg`, but we cannot place it there as it will disappear every time the system is booted.

```
PROMPT.ROOT mkdir /etc/pkg
```

Now, add a line to `/etc/rc.d/var` that links the `/etc/pkg` directory to `/var/db/pkg`. An example:

```
PROMPT.ROOT ln -s /etc/pkg /var/db/pkg
```

Now, any time that you mount your filesystems as read-write and install a package, the `make install` will work, and package information will be written successfully to `/etc/pkg` (because the filesystem will, at that time, be mounted read-write) which will always be available to the operating system as `/var/db/pkg`.

74.5.4 Apache Web Server

Note

The steps in this section are only necessary if Apache is set up to write its pid or log information outside of `/var`. By default, Apache keeps its pid file in `/var/run/httpd.pid` and its log files in `/var/log`.

It is now assumed that Apache keeps its log files in a directory `apache_log_dir` outside of `/var`. When this directory lives on a read-only filesystem, Apache will not be able to save any log files, and may have problems working. If so, it is necessary to add a new directory to the list of directories in `/etc/rc.d/var` to create in `/var`, and to link `apache_log_dir` to `/var/log/apache`. It is also necessary to set permissions and ownership on this new directory.

First, add the directory `log/apache` to the list of directories to be created in `/etc/rc.d/var`.

Second, add these commands to `/etc/rc.d/var` after the directory creation section:

```
PROMPT.ROOT chmod 0774 /var/log/apache
PROMPT.ROOT chown nobody:nobody /var/log/apache
```

Finally, remove the existing `apache_log_dir` directory, and replace it with a link:

```
PROMPT.ROOT rm -rf apache_log_dir
PROMPT.ROOT ln -s /var/log/apache apache_log_dir
```

The `vinum` Volume Manager

Author Greg Lehey Originally written by

75.1 Synopsis

No matter the type of disks, there are always potential problems. The disks can be too small, too slow, or too unreliable to meet the system's requirements. While disks are getting bigger, so are data storage requirements. Often a file system is needed that is bigger than a disk's capacity. Various solutions to these problems have been proposed and implemented.

One method is through the use of multiple, and sometimes redundant, disks. In addition to supporting various cards and controllers for hardware Redundant Array of Independent Disks RAID systems, the base OS system includes the `vinum` volume manager, a block device driver that implements virtual disk drives and addresses these three problems. `vinum` provides more flexibility, performance, and reliability than traditional disk storage and implements RAID-0, RAID-1, and RAID-5 models, both individually and in combination.

This chapter provides an overview of potential problems with traditional disk storage, and an introduction to the `vinum` volume manager.

Note

Starting with OS 5, `vinum` has been rewritten in order to fit into the GEOM architecture, while retaining the original ideas, terminology, and on-disk metadata. This rewrite is called *gvinum* (for *GEOM vinum*). While this chapter uses the term `vinum`, any command invocations should be performed with `gvinum`. The name of the kernel module has changed from the original `vinum.ko` to `geom_vinum.ko`, and all device nodes reside under `/dev/gvinum` instead of `/dev/vinum`. As of OS 6, the original `vinum` implementation is no longer available in the code base.

75.2 Access Bottlenecks

Modern systems frequently need to access data in a highly concurrent manner. For example, large FTP or HTTP servers can maintain thousands of concurrent sessions and have multiple 100 Mbit/s connections to the outside world, well beyond the sustained transfer rate of most disks.

Current disk drives can transfer data sequentially at up to 70 MB/s, but this value is of little importance in an environment where many independent processes access a drive, and where they may achieve only a fraction of these values. In such cases, it is more interesting to view the problem from the viewpoint of the disk subsystem. The important parameter is the load that a transfer places on the subsystem, or the time for which a transfer occupies the drives involved in the transfer.

In any disk transfer, the drive must first position the heads, wait for the first sector to pass under the read head, and then perform the transfer. These actions can be considered to be atomic as it does not make any sense to interrupt them.

Consider a typical transfer of about 10 kB: the current generation of high-performance disks can position the heads in an average of 3.5 ms. The fastest drives spin at 15,000 rpm, so the average rotational latency (half a revolution) is 2 ms. At 70 MB/s, the transfer itself takes about 150 μ s, almost nothing compared to the positioning time. In such a case, the effective transfer rate drops to a little over 1 MB/s and is clearly highly dependent on the transfer size.

The traditional and obvious solution to this bottleneck is “more spindles”: rather than using one large disk, use several smaller disks with the same aggregate storage space. Each disk is capable of positioning and transferring independently, so the effective throughput increases by a factor close to the number of disks used.

The actual throughput improvement is smaller than the number of disks involved. Although each drive is capable of transferring in parallel, there is no way to ensure that the requests are evenly distributed across the drives. Inevitably the load on one drive will be higher than on another.

disk concatenation Vinum concatenation The evenness of the load on the disks is strongly dependent on the way the data is shared across the drives. In the following discussion, it is convenient to think of the disk storage as a large number of data sectors which are addressable by number, rather like the pages in a book. The most obvious method is to divide the virtual disk into groups of consecutive sectors the size of the individual physical disks and store them in this manner, rather like taking a large book and tearing it into smaller sections. This method is called *concatenation* and has the advantage that the disks are not required to have any specific size relationships. It works well when the access to the virtual disk is spread evenly about its address space. When access is concentrated on a smaller area, the improvement is less marked. ? illustrates the sequence in which storage units are allocated in a concatenated organization.

disk striping Vinum striping RAID An alternative mapping is to divide the address space into smaller, equal-sized components and store them sequentially on different devices. For example, the first 256 sectors may be stored on the first disk, the next 256 sectors on the next disk and so on. After filling the last disk, the process repeats until the disks are full. This mapping is called *striping* or RAID-0.

RAID offers various forms of fault tolerance, though RAID-0 is somewhat misleading as it provides no redundancy. Striping requires somewhat more effort to locate the data, and it can cause additional I/O load where a transfer is spread over multiple disks, but it can also provide a more constant load across the disks. ? illustrates the sequence in which storage units are allocated in a striped organization.

75.3 Data Integrity

The final problem with disks is that they are unreliable. Although reliability has increased tremendously over the last few years, disk drives are still the most likely core component of a server to fail. When they do, the results can be catastrophic and replacing a failed disk drive and restoring data can result in server downtime.

disk mirroring vinum mirroring RAID -1 One approach to this problem is *mirroring*, or RAID-1, which keeps two copies of the data on different physical hardware. Any write to the volume writes to both disks; a read can be satisfied from either, so if one drive fails, the data is still available on the other drive.

Mirroring has two problems:

- It requires twice as much disk storage as a non-redundant solution.
- Writes must be performed to both drives, so they take up twice the bandwidth of a non-mirrored volume. Reads do not suffer from a performance penalty and can even be faster.

RAID -5 An alternative solution is *parity*, implemented in RAID levels 2, 3, 4 and 5. Of these, RAID-5 is the most interesting. As implemented in `vinum`, it is a variant on a striped organization which dedicates one block of each stripe to parity one of the other blocks. As implemented by `vinum`, a RAID-5 plex is similar to a striped plex, except that it implements RAID-5 by including a parity block in each stripe. As required by RAID-5, the location of this parity block changes from one stripe to the next. The numbers in the data blocks indicate the relative block numbers.

Compared to mirroring, RAID-5 has the advantage of requiring significantly less storage space. Read access is similar to that of striped organizations, but write access is significantly slower, approximately 25% of the read performance. If one drive fails, the array can continue to operate in degraded mode where a read from one of the remaining accessible drives continues normally, but a read from the failed drive is recalculated from the corresponding block from all the remaining drives.

75.4 `vinum` Objects

In order to address these problems, `vinum` implements a four-level hierarchy of objects:

- The most visible object is the virtual disk, called a *volume*. Volumes have essentially the same properties as a UNIX disk drive, though there are some minor differences. For one, they have no size limitations.
- Volumes are composed of *plexes*, each of which represent the total address space of a volume. This level in the hierarchy provides redundancy. Think of plexes as individual disks in a mirrored array, each containing the same data.
- Since `vinum` exists within the UNIX disk storage framework, it would be possible to use UNIX partitions as the building block for multi-disk plexes. In fact, this turns out to be too inflexible as UNIX disks can have only a limited number of partitions. Instead, `vinum` subdivides a single UNIX partition, the *drive*, into contiguous areas called *subdisks*, which are used as building blocks for plexes.
- Subdisks reside on `vinum drives`, currently UNIX partitions. `vinum` drives can contain any number of subdisks. With the exception of a small area at the beginning of the drive, which is used for storing configuration and state information, the entire drive is available for data storage.

The following sections describe the way these objects provide the functionality required of `vinum`.

75.4.1 Volume Size Considerations

Plexes can include multiple subdisks spread over all drives in the `vinum` configuration. As a result, the size of an individual drive does not limit the size of a plex or a volume.

75.4.2 Redundant Data Storage

`vinum` implements mirroring by attaching multiple plexes to a volume. Each plex is a representation of the data in a volume. A volume may contain between one and eight plexes.

Although a plex represents the complete data of a volume, it is possible for parts of the representation to be physically missing, either by design (by not defining a subdisk for parts of the plex) or by accident (as a result of the failure of a drive). As long as at least one plex can provide the data for the complete address range of the volume, the volume is fully functional.

75.4.3 Which Plex Organization?

`vinum` implements both concatenation and striping at the plex level:

- A *concatenated plex* uses the address space of each subdisk in turn. Concatenated plexes are the most flexible as they can contain any number of subdisks, and the subdisks may be of different length. The plex may be extended by adding additional subdisks. They require less CPU time than striped plexes, though the difference in CPU overhead is not measurable. On the other hand, they are most susceptible to hot spots, where one disk is very active and others are idle.

- A *striped plex* stripes the data across each subdisk. The subdisks must all be the same size and there must be at least two subdisks in order to distinguish it from a concatenated plex. The greatest advantage of striped plexes is that they reduce hot spots. By choosing an optimum sized stripe, about 256 kB, the load can be evened out on the component drives. Extending a plex by adding new subdisks is so complicated that `vinum` does not implement it.

? summarizes the advantages and disadvantages of each plex organization.

Plex type	Minimum subdisks	Can add subdisks	Must be equal size	Application
con-cate-nated	1	yes	no	Large data storage with maximum placement flexibility and moderate performance
striped	2	no	yes	High performance in combination with highly concurrent access

Table: `vinum` Plex Organizations

75.5 Some Examples

`vinum` maintains a *configuration database* which describes the objects known to an individual system. Initially, the user creates the configuration database from one or more configuration files using `MAN.GVINUM.8`. `vinum` stores a copy of its configuration database on each disk *device* under its control. This database is updated on each state change, so that a restart accurately restores the state of each `vinum` object.

75.5.1 The Configuration File

The configuration file describes individual `vinum` objects. The definition of a simple volume might be:

```
drive a device /dev/da3h
volume myvol
  plex org concat
    sd length 512m drive a
```

This file describes four `vinum` objects:

- The *drive* line describes a disk partition (*drive*) and its location relative to the underlying hardware. It is given the symbolic name *a*. This separation of symbolic names from device names allows disks to be moved from one location to another without confusion.
- The *volume* line describes a volume. The only required attribute is the name, in this case *myvol*.
- The *plex* line defines a plex. The only required parameter is the organization, in this case *concat*. No name is necessary as the system automatically generates a name from the volume name by adding the suffix *.px*, where *x* is the number of the plex in the volume. Thus this plex will be called *myvol.p0*.
- The *sd* line describes a subdisk. The minimum specifications are the name of a drive on which to store it, and the length of the subdisk. No name is necessary as the system automatically assigns names derived from the plex name by adding the suffix *.sx*, where *x* is the number of the subdisk in the plex. Thus `vinum` gives this subdisk the name *myvol.p0.s0*.

After processing this file, `MAN.GVINUM.8` produces the following output:

```
PROMPT.ROOT gvinum -> create config1
Configuration summary
Drives:      1 (4 configured)
Volumes:     1 (4 configured)
```

Plexes:	1 (8 configured)				
Subdisks:	1 (16 configured)				
D a	State: up	Device /dev/da3h	Avail: 2061/2573 MB (80%)		
V myvol	State: up	Plexes:	1 Size: 512 MB		
P myvol.p0	C State: up	Subdisks:	1 Size: 512 MB		
S myvol.p0.s0	State: up	PO: 0 B	Size: 512 MB		

This output shows the brief listing format of MAN.GVINUM.8. It is represented graphically in ?.

This figure, and the ones which follow, represent a volume, which contains the plexes, which in turn contains the subdisks. In this example, the volume contains one plex, and the plex contains one subdisk.

This particular volume has no specific advantage over a conventional disk partition. It contains a single plex, so it is not redundant. The plex contains a single subdisk, so there is no difference in storage allocation from a conventional disk partition. The following sections illustrate various more interesting configuration methods.

75.5.2 Increased Resilience: Mirroring

The resilience of a volume can be increased by mirroring. When laying out a mirrored volume, it is important to ensure that the subdisks of each plex are on different drives, so that a drive failure will not take down both plexes. The following configuration mirrors a volume:

```
drive b device /dev/da4h
volume mirror
  plex org concat
    sd length 512m drive a
  plex org concat
    sd length 512m drive b
```

In this example, it was not necessary to specify a definition of drive *a* again, since *vinum* keeps track of all objects in its configuration database. After processing this definition, the configuration looks like:

Drives:	2 (4 configured)				
Volumes:	2 (4 configured)				
Plexes:	3 (8 configured)				
Subdisks:	3 (16 configured)				
D a	State: up	Device /dev/da3h	Avail: 1549/2573 MB (60%)		
D b	State: up	Device /dev/da4h	Avail: 2061/2573 MB (80%)		
V myvol	State: up	Plexes:	1 Size: 512 MB		
V mirror	State: up	Plexes:	2 Size: 512 MB		
P myvol.p0	C State: up	Subdisks:	1 Size: 512 MB		
P mirror.p0	C State: up	Subdisks:	1 Size: 512 MB		
P mirror.p1	C State: initializing	Subdisks:	1 Size: 512 MB		
S myvol.p0.s0	State: up	PO: 0 B	Size: 512 MB		
S mirror.p0.s0	State: up	PO: 0 B	Size: 512 MB		
S mirror.p1.s0	State: empty	PO: 0 B	Size: 512 MB		

? shows the structure graphically.

In this example, each plex contains the full 512 MB of address space. As in the previous example, each plex contains only a single subdisk.

75.5.3 Optimizing Performance

The mirrored volume in the previous example is more resistant to failure than an unmirrored volume, but its performance is less as each write to the volume requires a write to both drives, using up a greater proportion of the total disk bandwidth. Performance considerations demand a different approach: instead of mirroring, the data is striped across as many disk drives as possible. The following configuration shows a volume with a plex striped across four disk drives:

```
drive c device /dev/da5h
drive d device /dev/da6h
volume stripe
plex org striped 512k
  sd length 128m drive a
  sd length 128m drive b
  sd length 128m drive c
  sd length 128m drive d
```

As before, it is not necessary to define the drives which are already known to `vinum`. After processing this definition, the configuration looks like:

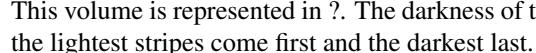
```
Drives:      4 (4 configured)
Volumes:     3 (4 configured)
Plexes:      4 (8 configured)
Subdisks:    7 (16 configured)

D a          State: up      Device /dev/da3h      Avail: 1421/2573 MB (55%)
D b          State: up      Device /dev/da4h      Avail: 1933/2573 MB (75%)
D c          State: up      Device /dev/da5h      Avail: 2445/2573 MB (95%)
D d          State: up      Device /dev/da6h      Avail: 2445/2573 MB (95%)

V myvol      State: up      Plexes:      1 Size:      512 MB
V mirror     State: up      Plexes:      2 Size:      512 MB
V striped    State: up      Plexes:      1 Size:      512 MB

P myvol.p0   C State: up      Subdisks:    1 Size:      512 MB
P mirror.p0  C State: up      Subdisks:    1 Size:      512 MB
P mirror.p1  C State: initializing Subdisks:    1 Size:      512 MB
P striped.p1 State: up      Subdisks:    1 Size:      512 MB

S myvol.p0.s0 State: up      PO:          0 B Size:      512 MB
S mirror.p0.s0 State: up      PO:          0 B Size:      512 MB
S mirror.p1.s0 State: empty   PO:          0 B Size:      512 MB
S striped.p0.s0 State: up      PO:          0 B Size:      128 MB
S striped.p0.s1 State: up      PO:          512 kB Size:      128 MB
S striped.p0.s2 State: up      PO:          1024 kB Size:      128 MB
S striped.p0.s3 State: up      PO:          1536 kB Size:      128 MB
```

This volume is represented in . The darkness of the stripes indicates the position within the plex address space, where the lightest stripes come first and the darkest last.

75.5.4 Resilience and Performance

With sufficient hardware, it is possible to build volumes which show both increased resilience and increased performance compared to standard UNIX partitions. A typical configuration file might be:

```
volume raid10
  plex org striped 512k
    sd length 102480k drive a
    sd length 102480k drive b
    sd length 102480k drive c
    sd length 102480k drive d
    sd length 102480k drive e
  plex org striped 512k
    sd length 102480k drive c
    sd length 102480k drive d
    sd length 102480k drive e
    sd length 102480k drive a
    sd length 102480k drive b
```

The subdisks of the second plex are offset by two drives from those of the first plex. This helps to ensure that writes do not go to the same subdisks even if a transfer goes over two drives.

? represents the structure of this volume.

75.6 Object Naming

vinum assigns default names to plexes and subdisks, although they may be overridden. Overriding the default names is not recommended as it does not bring a significant advantage and it can cause confusion.

Names may contain any non-blank character, but it is recommended to restrict them to letters, digits and the underscore characters. The names of volumes, plexes, and subdisks may be up to 64 characters long, and the names of drives may be up to 32 characters long.

vinum objects are assigned device nodes in the hierarchy `/dev/gvinum`. The configuration shown above would cause vinum to create the following device nodes:

- Device entries for each volume. These are the main devices used by vinum. The configuration above would include the devices `/dev/gvinum/myvol`, `/dev/gvinum/mirror`, `/dev/gvinum/striped`, `/dev/gvinum/raid5` and `/dev/gvinum/raid10`.
- All volumes get direct entries under `/dev/gvinum/`.
- The directories `/dev/gvinum/plex`, and `/dev/gvinum/sd`, which contain device nodes for each plex and for each subdisk, respectively.

For example, consider the following configuration file:

```
drive drive1 device /dev/sd1h
drive drive2 device /dev/sd2h
drive drive3 device /dev/sd3h
drive drive4 device /dev/sd4h
volume s64 setupstate
  plex org striped 64k
    sd length 100m drive drive1
    sd length 100m drive drive2
    sd length 100m drive drive3
    sd length 100m drive drive4
```

After processing this file, MAN.GVINUM.8 creates the following structure in `/dev/gvinum`:

```
drwxr-xr-x  2 root  wheel           512 Apr 13
16:46 plex
crwxr-xr--  1 root  wheel    91,    2 Apr 13 16:46 s64
drwxr-xr-x  2 root  wheel           512 Apr 13 16:46 sd

/dev/vinum/plex:
total 0
crwxr-xr--  1 root  wheel    25, 0x10000002 Apr 13 16:46 s64.p0

/dev/vinum/sd:
total 0
crwxr-xr--  1 root  wheel    91, 0x20000002 Apr 13 16:46 s64.p0.s0
crwxr-xr--  1 root  wheel    91, 0x20100002 Apr 13 16:46 s64.p0.s1
crwxr-xr--  1 root  wheel    91, 0x20200002 Apr 13 16:46 s64.p0.s2
crwxr-xr--  1 root  wheel    91, 0x20300002 Apr 13 16:46 s64.p0.s3
```

Although it is recommended that plexes and subdisks should not be allocated specific names, `vinum` drives must be named. This makes it possible to move a drive to a different location and still recognize it automatically. Drive names may be up to 32 characters long.

75.6.1 Creating File Systems

Volumes appear to the system to be identical to disks, with one exception. Unlike UNIX drives, `vinum` does not partition volumes, which thus do not contain a partition table. This has required modification to some disk utilities, notably MAN.NEWFS.8, so that it does not try to interpret the last letter of a `vinum` volume name as a partition identifier. For example, a disk drive may have a name like `/dev/ad0a` or `/dev/da2h`. These names represent the first partition (a) on the first (0) IDE disk (ad) and the eighth partition (h) on the third (2) SCSI disk (da) respectively. By contrast, a `vinum` volume might be called `/dev/gvinum/concat`, which has no relationship with a partition name.

In order to create a file system on this volume, use MAN.NEWFS.8:

```
PROMPT.ROOT newfs /dev/gvinum/concat
```

75.7 Configuring `vinum`

The GENERIC kernel does not contain `vinum`. It is possible to build a custom kernel which includes `vinum`, but this is not recommended. The standard way to start `vinum` is as a kernel module. MAN.KLDLOAD.8 is not needed because when MAN.GVINUM.8 starts, it checks whether the module has been loaded, and if it is not, it loads it automatically.

75.7.1 Startup

`vinum` stores configuration information on the disk slices in essentially the same form as in the configuration files. When reading from the configuration database, `vinum` recognizes a number of keywords which are not allowed in the configuration files. For example, a disk configuration might contain the following text:

```
volume myvol state up
volume bigraid state down
plex name myvol.p0 state up org concat vol myvol
plex name myvol.p1 state up org concat vol myvol
```

```
plex name myvol.p2 state init org striped 512b vol myvol
plex name bigraid.p0 state initializing org raid5 512b vol bigraid
sd name myvol.p0.s0 drive a plex myvol.p0 state up len 1048576b driveoffset 265b plexoffset 0b
sd name myvol.p0.s1 drive b plex myvol.p0 state up len 1048576b driveoffset 265b plexoffset 1048576b
sd name myvol.p1.s0 drive c plex myvol.p1 state up len 1048576b driveoffset 265b plexoffset 0b
sd name myvol.p1.s1 drive d plex myvol.p1 state up len 1048576b driveoffset 265b plexoffset 1048576b
sd name myvol.p2.s0 drive a plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 0b
sd name myvol.p2.s1 drive b plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 524288b
sd name myvol.p2.s2 drive c plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 1048841b
sd name myvol.p2.s3 drive d plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 1573129b
sd name bigraid.p0.s0 drive a plex bigraid.p0 state initializing len 4194304b driveoffset 1573129b plexoffset 0b
sd name bigraid.p0.s1 drive b plex bigraid.p0 state initializing len 4194304b driveoffset 1573129b plexoffset 4194304b
sd name bigraid.p0.s2 drive c plex bigraid.p0 state initializing len 4194304b driveoffset 1573129b plexoffset 8388608b
sd name bigraid.p0.s3 drive d plex bigraid.p0 state initializing len 4194304b driveoffset 1573129b plexoffset 12582912b
sd name bigraid.p0.s4 drive e plex bigraid.p0 state initializing len 4194304b driveoffset 1573129b plexoffset 16777216b
```

The obvious differences here are the presence of explicit location information and naming, both of which are allowed but discouraged, and the information on the states. `vinum` does not store information about drives in the configuration information. It finds the drives by scanning the configured disk drives for partitions with a `vinum` label. This enables `vinum` to identify drives correctly even if they have been assigned different UNIX drive IDs.

Automatic Startup

Gvinum always features an automatic startup once the kernel module is loaded, via `MAN.LOADER.CONF.5`. To load the *Gvinum* module at boot time, add `geom_vinum_load="YES"` to `/boot/loader.conf`.

When `vinum` is started with `gvinum start`, `vinum` reads the configuration database from one of the `vinum` drives. Under normal circumstances, each drive contains an identical copy of the configuration database, so it does not matter which drive is read. After a crash, however, `vinum` must determine which drive was updated most recently and read the configuration from this drive. It then updates the configuration, if necessary, from progressively older drives.

75.8 Using `vinum` for the Root File System

For a machine that has fully-mirrored file systems using `vinum`, it is desirable to also mirror the root file system. Setting up such a configuration is less trivial than mirroring an arbitrary file system because:

- The root file system must be available very early during the boot process, so the `vinum` infrastructure must already be available at this time.
- The volume containing the root file system also contains the system bootstrap and the kernel. These must be read using the host system's native utilities, such as the BIOS, which often cannot be taught about the details of `vinum`.

In the following sections, the term “root volume” is generally used to describe the `vinum` volume that contains the root file system.

75.8.1 Starting up `vinum` Early Enough for the Root File System

`vinum` must be available early in the system boot as `MAN.LOADER.8` must be able to load the `vinum` kernel module before starting the kernel. This can be accomplished by putting this line in `/boot/loader.conf`:

```
geom_vinum_load="YES"
```

75.8.2 Making a `vinum`-based Root Volume Accessible to the Bootstrap

The current OS bootstrap is only 7.5 KB of code and does not understand the internal `vinum` structures. This means that it cannot parse the `vinum` configuration data or figure out the elements of a boot volume. Thus, some workarounds are necessary to provide the bootstrap code with the illusion of a standard `a` partition that contains the root file system.

For this to be possible, the following requirements must be met for the root volume:

- The root volume must not be a stripe or RAID-5.
- The root volume must not contain more than one concatenated subdisk per plex.

Note that it is desirable and possible to use multiple plexes, each containing one replica of the root file system. The bootstrap process will only use one replica for finding the bootstrap and all boot files, until the kernel mounts the root file system. Each single subdisk within these plexes needs its own `a` partition illusion, for the respective device to be bootable. It is not strictly needed that each of these faked `a` partitions is located at the same offset within its device, compared with other devices containing plexes of the root volume. However, it is probably a good idea to create the `vinum` volumes that way so the resulting mirrored devices are symmetric, to avoid confusion.

In order to set up these `a` partitions for each device containing part of the root volume, the following is required:

The location, offset from the beginning of the device, and size of this device's subdisk that is part of the root volume needs to be examined, using the command:

```
PROMPT.ROOT gvinum l -rv root
```

`vinum` offsets and sizes are measured in bytes. They must be divided by 512 in order to obtain the block numbers that are to be used by `bsdlabel`.

Run this command for each device that participates in the root volume:

```
PROMPT.ROOT bsdlabel -e devname
```

`devname` must be either the name of the disk, like `da0` for disks without a slice table, or the name of the slice, like `ad0s1`.

If there is already an `a` partition on the device from a pre-`vinum` root file system, it should be renamed to something else so that it remains accessible (just in case), but will no longer be used by default to bootstrap the system. A currently mounted root file system cannot be renamed, so this must be executed either when being booted from a “Fixit” media, or in a two-step process where, in a mirror, the disk that is not been currently booted is manipulated first.

The offset of the `vinum` partition on this device (if any) must be added to the offset of the respective root volume subdisk on this device. The resulting value will become the `offset` value for the new `a` partition. The `size` value for this partition can be taken verbatim from the calculation above. The `fstype` should be `4.2BSD`. The `fsize`, `bsize`, and `cpg` values should be chosen to match the actual file system, though they are fairly unimportant within this context.

That way, a new `a` partition will be established that overlaps the `vinum` partition on this device. `bsdlabel` will only allow for this overlap if the `vinum` partition has properly been marked using the `vinum` `fstype`.

A faked `a` partition now exists on each device that has one replica of the root volume. It is highly recommendable to verify the result using a command like:

```
PROMPT.ROOT fsck -n /dev/devnamea
```

It should be remembered that all files containing control information must be relative to the root file system in the `vinum` volume which, when setting up a new `vinum` root volume, might not match the root file system that is currently active. So in particular, `/etc/fstab` and `/boot/loader.conf` need to be taken care of.

At next reboot, the bootstrap should figure out the appropriate control information from the new `vinum`-based root file system, and act accordingly. At the end of the kernel initialization process, after all devices have been announced, the prominent notice that shows the success of this setup is a message like:

```
Mounting root from ufs:/dev/gvinum/root
```

75.8.3 Example of a `vinum`-based Root Setup

After the `vinum` root volume has been set up, the output of “`gvinum l -rv root`” could look like:

```
...
Subdisk root.p0.s0:
  Size:          125829120 bytes (120 MB)
  State: up
  Plex root.p0 at offset 0 (0 B)
  Drive disk0 (/dev/da0h) at offset 135680 (132 kB)

Subdisk root.pl.s0:
  Size:          125829120 bytes (120 MB)
  State: up
  Plex root.pl at offset 0 (0 B)
  Drive disk1 (/dev/dal1h) at offset 135680 (132 kB)
```

The values to note are 135680 for the offset, relative to partition `/dev/da0h`. This translates to 265 512-byte disk blocks in `bsdlabeled`'s terms. Likewise, the size of this root volume is 245760 512-byte blocks. `/dev/dal1h`, containing the second replica of this root volume, has a symmetric setup.

The `bsdlabeled` for these devices might look like:

```
...
8 partitions:
#      size    offset    fstype    [fsize bsize bps/cpg]
a:   245760     281    4.2BSD    2048 16384    0   # (Cyl.  0*- 15*)
c:  71771688      0    unused      0      0      # (Cyl.  0 - 4467*)
h:  71771672     16    vinum                # (Cyl.  0*- 4467*)
```

It can be observed that the `size` parameter for the faked `a` partition matches the value outlined above, while the `offset` parameter is the sum of the offset within the `vinum` partition `h`, and the offset of this partition within the device or slice. This is a typical setup that is necessary to avoid the problem described in ?. The entire `a` partition is completely within the `h` partition containing all the `vinum` data for this device.

In the above example, the entire device is dedicated to `vinum` and there is no leftover pre-`vinum` root partition.

75.8.4 Troubleshooting

The following list contains a few known pitfalls and solutions.

System Bootstrap Loads, but System Does Not Boot

If for any reason the system does not continue to boot, the bootstrap can be interrupted by pressing space at the 10-seconds warning. The loader variable `vinum.autostart` can be examined by typing `show` and manipulated using `set` or `unset`.

If the `vinum` kernel module was not yet in the list of modules to load automatically, type `load geom_vinum`.

When ready, the boot process can be continued by typing `boot -as` which `-as` requests the kernel to ask for the root file system to mount (`-a`) and make the boot process stop in single-user mode (`-s`), where the root file system is mounted read-only. That way, even if only one plex of a multi-plex volume has been mounted, no data inconsistency between plexes is being risked.

At the prompt asking for a root file system to mount, any device that contains a valid root file system can be entered. If `/etc/fstab` is set up correctly, the default should be something like `ufs:/dev/gvinum/root`. A typical alternate choice would be something like `ufs:da0d` which could be a hypothetical partition containing the pre-`vinum` root file system. Care should be taken if one of the alias `a` partitions is entered here, that it actually references the subdisks of the `vinum` root device, because in a mirrored setup, this would only mount one piece of a mirrored root device. If this file system is to be mounted read-write later on, it is necessary to remove the other plex(es) of the `vinum` root volume since these plexes would otherwise carry inconsistent data.

Only Primary Bootstrap Loads

If `/boot/loader` fails to load, but the primary bootstrap still loads (visible by a single dash in the left column of the screen right after the boot process starts), an attempt can be made to interrupt the primary bootstrap by pressing space. This will make the bootstrap stop in stage two. An attempt can be made here to boot off an alternate partition, like the partition containing the previous root file system that has been moved away from `a`.

Nothing Boots, the Bootstrap Panics

This situation will happen if the bootstrap had been destroyed by the `vinum` installation. Unfortunately, `vinum` accidentally leaves only 4 KB at the beginning of its partition free before starting to write its `vinum` header information. However, the stage one and two bootstraps plus the `bsdlabel` require 8 KB. So if a `vinum` partition was started at offset 0 within a slice or disk that was meant to be bootable, the `vinum` setup will trash the bootstrap.

Similarly, if the above situation has been recovered, by booting from a “Fixit” media, and the bootstrap has been re-installed using `bsdlabel -B` as described in url.books.handbook/boot.html#boot-boot1, the bootstrap will trash the `vinum` header, and `vinum` will no longer find its disk(s). Though no actual `vinum` configuration data or data in `vinum` volumes will be trashed, and it would be possible to recover all the data by entering exactly the same `vinum` configuration data again, the situation is hard to fix. It is necessary to move the entire `vinum` partition by at least 4 KB, in order to have the `vinum` header and the system bootstrap no longer collide.

Design elements of the OS VM system

Author MatthewDillon

76.1 Introduction

Before moving along to the actual design let's spend a little time on the necessity of maintaining and modernizing any long-living codebase. In the programming world, algorithms tend to be more important than code and it is precisely due to BSD's academic roots that a great deal of attention was paid to algorithm design from the beginning. More attention paid to the design generally leads to a clean and flexible codebase that can be fairly easily modified, extended, or replaced over time. While BSD is considered an "old" operating system by some people, those of us who work on it tend to view it more as a "mature" codebase which has various components modified, extended, or replaced with modern code. It has evolved, and OS is at the bleeding edge no matter how old some of the code might be. This is an important distinction to make and one that is unfortunately lost to many people. The biggest error a programmer can make is to not learn from history, and this is precisely the error that many other modern operating systems have made. WINDOWSNT is the best example of this, and the consequences have been dire. Linux also makes this mistake to some degree—enough that we BSD folk can make small jokes about it every once in a while, anyway. Linux's problem is simply one of a lack of experience and history to compare ideas against, a problem that is easily and rapidly being addressed by the Linux community in the same way it has been addressed in the BSD community—by continuous code development. The WINDOWSNT folk, on the other hand, repeatedly make the same mistakes solved by UNIX decades ago and then spend years fixing them. Over and over again. They have a severe case of "not designed here" and "we are always right because our marketing department says so". I have little tolerance for anyone who cannot learn from history.

Much of the apparent complexity of the OS design, especially in the VM/Swap subsystem, is a direct result of having to solve serious performance issues that occur under various conditions. These issues are not due to bad algorithmic design but instead rise from environmental factors. In any direct comparison between platforms, these issues become most apparent when system resources begin to get stressed. As I describe OS's VM/Swap subsystem the reader should always keep two points in mind:

1. The most important aspect of performance design is what is known as "Optimizing the Critical Path". It is often the case that performance optimizations add a little bloat to the code in order to make the critical path perform better.
2. A solid, generalized design outperforms a heavily-optimized design over the long run. While a generalized design may end up being slower than an heavily-optimized design when they are first implemented, the generalized design tends to be easier to adapt to changing conditions and the heavily-optimized design winds up having to be thrown away.

Any codebase that will survive and be maintainable for years must therefore be designed properly from the beginning even if it costs some performance. Twenty years ago people were still arguing that programming in assembly was

better than programming in a high-level language because it produced code that was ten times as fast. Today, the fallibility of that argument is obvious — as are the parallels to algorithmic design and code generalization.

76.2 VM Objects

The best way to begin describing the OS VM system is to look at it from the perspective of a user-level process. Each user process sees a single, private, contiguous VM address space containing several types of memory objects. These objects have various characteristics. Program code and program data are effectively a single memory-mapped file (the binary file being run), but program code is read-only while program data is copy-on-write. Program BSS is just memory allocated and filled with zeros on demand, called demand zero page fill. Arbitrary files can be memory-mapped into the address space as well, which is how the shared library mechanism works. Such mappings can require modifications to remain private to the process making them. The fork system call adds an entirely new dimension to the VM management problem on top of the complexity already given.

A program binary data page (which is a basic copy-on-write page) illustrates the complexity. A program binary contains a preinitialized data section which is initially mapped directly from the program file. When a program is loaded into a process's VM space, this area is initially memory-mapped and backed by the program binary itself, allowing the VM system to free/reuse the page and later load it back in from the binary. The moment a process modifies this data, however, the VM system must make a private copy of the page for that process. Since the private copy has been modified, the VM system may no longer free it, because there is no longer any way to restore it later on.

You will notice immediately that what was originally a simple file mapping has become much more complex. Data may be modified on a page-by-page basis whereas the file mapping encompasses many pages at once. The complexity further increases when a process forks. When a process forks, the result is two processes—each with their own private address spaces, including any modifications made by the original process prior to the call to `fork()`. It would be silly for the VM system to make a complete copy of the data at the time of the `fork()` because it is quite possible that at least one of the two processes will only need to read from that page from then on, allowing the original page to continue to be used. What was a private page is made copy-on-write again, since each process (parent and child) expects their own personal post-fork modifications to remain private to themselves and not effect the other.

OS manages all of this with a layered VM Object model. The original binary program file winds up being the lowest VM Object layer. A copy-on-write layer is pushed on top of that to hold those pages which had to be copied from the original file. If the program modifies a data page belonging to the original file the VM system takes a fault and makes a copy of the page in the higher layer. When a process forks, additional VM Object layers are pushed on. This might make a little more sense with a fairly basic example. A `fork()` is a common operation for any *BSD system, so this example will consider a program that starts up, and forks. When the process starts, the VM system creates an object layer, let's call this A:

+-----+ | A | +-----+ |

A represents the file—pages may be paged in and out of the file's physical media as necessary. Paging in from the disk is reasonable for a program, but we really do not want to page back out and overwrite the executable. The VM system therefore creates a second layer, B, that will be physically backed by swap space:

+-----+ | B | +-----+ | A | +-----+ |

On the first write to a page after this, a new page is created in B, and its contents are initialized from A. All pages in B can be paged in or out to a swap device. When the program forks, the VM system creates two new object layers—C1

for the parent, and C2 for the child—that rest on top of B:

```
+-----+-----+ | C1 | C2 | +-----+-----+ | B |
```

```
+-----+ | A | +-----+ |
```

In this case, let's say a page in B is modified by the original parent process. The process will take a copy-on-write fault and duplicate the page in C1, leaving the original page in B untouched. Now, let's say the same page in B is modified by the child process. The process will take a copy-on-write fault and duplicate the page in C2. The original page in B is now completely hidden since both C1 and C2 have a copy and B could theoretically be destroyed if it does not represent a “real” file; however, this sort of optimization is not trivial to make because it is so fine-grained. OS does not make this optimization. Now, suppose (as is often the case) that the child process does an `exec()`. Its current address space is usually replaced by a new address space representing a new file. In this case, the C2 layer is destroyed:

```
+-----+ | C1 | +-----+-----+ | B | +-----+-----+ | A |
```

```
+-----+ |
```

In this case, the number of children of B drops to one, and all accesses to B now go through C1. This means that B and C1 can be collapsed together. Any pages in B that also exist in C1 are deleted from B during the collapse. Thus, even though the optimization in the previous step could not be made, we can recover the dead pages when either of the processes exit or `exec()`.

This model creates a number of potential problems. The first is that you can wind up with a relatively deep stack of layered VM Objects which can cost scanning time and memory when you take a fault. Deep layering can occur when processes fork and then fork again (either parent or child). The second problem is that you can wind up with dead, inaccessible pages deep in the stack of VM Objects. In our last example if both the parent and child processes modify the same page, they both get their own private copies of the page and the original page in B is no longer accessible by anyone. That page in B can be freed.

OS solves the deep layering problem with a special optimization called the “All Shadowed Case”. This case occurs if either C1 or C2 take sufficient COW faults to completely shadow all pages in B. Let's say that C1 achieves this. C1 can now bypass B entirely, so rather than have C1->B->A and C2->B->A we now have C1->A and C2->B->A. But look what also happened—now B has only one reference (C2), so we can collapse B and C2 together. The end result is that B is deleted entirely and we have C1->A and C2->A. It is often the case that B will contain a large number of pages and neither C1 nor C2 will be able to completely overshadow it. If we fork again and create a set of D layers, however, it is much more likely that one of the D layers will eventually be able to completely overshadow the much smaller dataset represented by C1 or C2. The same optimization will work at any point in the graph and the grand result of this is that even on a heavily forked machine VM Object stacks tend to not get much deeper than 4. This is true of both the parent and the children and true whether the parent is doing the forking or whether the children cascade forks.

The dead page problem still exists in the case where C1 or C2 do not completely overshadow B. Due to our other optimizations this case does not represent much of a problem and we simply allow the pages to be dead. If the system runs low on memory it will swap them out, eating a little swap, but that is it.

The advantage to the VM Object model is that `fork()` is extremely fast, since no real data copying need take place. The disadvantage is that you can build a relatively complex VM Object layering that slows page fault handling down a little, and you spend memory managing the VM Object structures. The optimizations OS makes proves to reduce the problems enough that they can be ignored, leaving no real disadvantage.

76.3 SWAP Layers

Private data pages are initially either copy-on-write or zero-fill pages. When a change, and therefore a copy, is made, the original backing object (usually a file) can no longer be used to save a copy of the page when the VM system needs to reuse it for other purposes. This is where SWAP comes in. SWAP is allocated to create backing store for memory that does not otherwise have it. OS allocates the swap management structure for a VM Object only when it is actually needed. However, the swap management structure has had problems historically:

- Under OS 3.X the swap management structure preallocates an array that encompasses the entire object requiring swap backing store—even if only a few pages of that object are swap-backed. This creates a kernel memory fragmentation problem when large objects are mapped, or processes with large runsizes (RSS) fork.
- Also, in order to keep track of swap space, a “list of holes” is kept in kernel memory, and this tends to get severely fragmented as well. Since the “list of holes” is a linear list, the swap allocation and freeing performance is a non-optimal $O(n)$ -per-page.
- It requires kernel memory allocations to take place during the swap freeing process, and that creates low memory deadlock problems.
- The problem is further exacerbated by holes created due to the interleaving algorithm.
- Also, the swap block map can become fragmented fairly easily resulting in non-contiguous allocations.
- Kernel memory must also be allocated on the fly for additional swap management structures when a swapout occurs.

It is evident from that list that there was plenty of room for improvement. For OS 4.X, I completely rewrote the swap subsystem:

- Swap management structures are allocated through a hash table rather than a linear array giving them a fixed allocation size and much finer granularity.
- Rather than using a linearly linked list to keep track of swap space reservations, it now uses a bitmap of swap blocks arranged in a radix tree structure with free-space hinting in the radix node structures. This effectively makes swap allocation and freeing an $O(1)$ operation.
- The entire radix tree bitmap is also preallocated in order to avoid having to allocate kernel memory during critical low memory swapping operations. After all, the system tends to swap when it is low on memory so we should avoid allocating kernel memory at such times in order to avoid potential deadlocks.
- To reduce fragmentation the radix tree is capable of allocating large contiguous chunks at once, skipping over smaller fragmented chunks.

I did not take the final step of having an “allocating hint pointer” that would trundle through a portion of swap as allocations were made in order to further guarantee contiguous allocations or at least locality of reference, but I ensured that such an addition could be made.

76.4 When to free a page

Since the VM system uses all available memory for disk caching, there are usually very few truly-free pages. The VM system depends on being able to properly choose pages which are not in use to reuse for new allocations. Selecting the optimal pages to free is possibly the single-most important function any VM system can perform because if it makes a poor selection, the VM system may be forced to unnecessarily retrieve pages from disk, seriously degrading system performance.

How much overhead are we willing to suffer in the critical path to avoid freeing the wrong page? Each wrong choice we make will cost us hundreds of thousands of CPU cycles and a noticeable stall of the affected processes, so we are

willing to endure a significant amount of overhead in order to be sure that the right page is chosen. This is why OS tends to outperform other systems when memory resources become stressed.

The free page determination algorithm is built upon a history of the use of memory pages. To acquire this history, the system takes advantage of a page-used bit feature that most hardware page tables have.

In any case, the page-used bit is cleared and at some later point the VM system comes across the page again and sees that the page-used bit has been set. This indicates that the page is still being actively used. If the bit is still clear it is an indication that the page is not being actively used. By testing this bit periodically, a use history (in the form of a counter) for the physical page is developed. When the VM system later needs to free up some pages, checking this history becomes the cornerstone of determining the best candidate page to reuse.

For those platforms that do not have this feature, the system actually emulates a page-used bit. It unmaps or protects a page, forcing a page fault if the page is accessed again. When the page fault is taken, the system simply marks the page as having been used and unprotects the page so that it may be used. While taking such page faults just to determine if a page is being used appears to be an expensive proposition, it is much less expensive than reusing the page for some other purpose only to find that a process needs it back and then have to go to disk.

OS makes use of several page queues to further refine the selection of pages to reuse as well as to determine when dirty pages must be flushed to their backing store. Since page tables are dynamic entities under OS, it costs virtually nothing to unmap a page from the address space of any processes using it. When a page candidate has been chosen based on the page-use counter, this is precisely what is done. The system must make a distinction between clean pages which can theoretically be freed up at any time, and dirty pages which must first be written to their backing store before being reusable. When a page candidate has been found it is moved to the inactive queue if it is dirty, or the cache queue if it is clean. A separate algorithm based on the dirty-to-clean page ratio determines when dirty pages in the inactive queue must be flushed to disk. Once this is accomplished, the flushed pages are moved from the inactive queue to the cache queue. At this point, pages in the cache queue can still be reactivated by a VM fault at relatively low cost. However, pages in the cache queue are considered to be “immediately freeable” and will be reused in an LRU (least-recently used) fashion when the system needs to allocate new memory.

It is important to note that the OS VM system attempts to separate clean and dirty pages for the express reason of avoiding unnecessary flushes of dirty pages (which eats I/O bandwidth), nor does it move pages between the various page queues gratuitously when the memory subsystem is not being stressed. This is why you will see some systems with very low cache queue counts and high active queue counts when doing a `sysstat -vm` command. As the VM system becomes more stressed, it makes a greater effort to maintain the various page queues at the levels determined to be the most effective.

An urban myth has circulated for years that Linux did a better job avoiding swapouts than OS, but this in fact is not true. What was actually occurring was that OS was proactively paging out unused pages in order to make room for more disk cache while Linux was keeping unused pages in core and leaving less memory available for cache and process pages. I do not know whether this is still true today.

76.5 Pre-Faulting and Zeroing Optimizations

Taking a VM fault is not expensive if the underlying page is already in core and can simply be mapped into the process, but it can become expensive if you take a whole lot of them on a regular basis. A good example of this is running a program such as `MAN.LS.1` or `MAN.PS.1` over and over again. If the program binary is mapped into memory but not mapped into the page table, then all the pages that will be accessed by the program will have to be faulted in every time the program is run. This is unnecessary when the pages in question are already in the VM Cache, so OS will attempt to pre-populate a process’s page tables with those pages that are already in the VM Cache. One thing that OS does not yet do is pre-copy-on-write certain pages on exec. For example, if you run the `MAN.LS.1` program while running `“vmstat`

1“ you will notice that it always takes a certain number of page

faults, even when you run it over and over again. These are zero-fill faults, not program code faults (which were pre-faulted in already). Pre-copying pages on exec or fork is an area that could use more study.

A large percentage of page faults that occur are zero-fill faults. You can usually see this by observing the `vmstat -s` output. These occur when a process accesses pages in its BSS area. The BSS area is expected to be initially zero but the VM system does not bother to allocate any memory at all until the process actually accesses it. When a fault occurs the VM system must not only allocate a new page, it must zero it as well. To optimize the zeroing operation the VM system has the ability to pre-zero pages and mark them as such, and to request pre-zeroed pages when zero-fill faults occur. The pre-zeroing occurs whenever the CPU is idle but the number of pages the system pre-zeros is limited in order to avoid blowing away the memory caches. This is an excellent example of adding complexity to the VM system in order to optimize the critical path.

76.6 Page Table Optimizations

The page table optimizations make up the most contentious part of the OS VM design and they have shown some strain with the advent of serious use of `mmap()`. I think this is actually a feature of most BSDs though I am not sure when it was first introduced. There are two major optimizations. The first is that hardware page tables do not contain persistent state but instead can be thrown away at any time with only a minor amount of management overhead. The second is that every active page table entry in the system has a governing `pv_entry` structure which is tied into the `vm_page` structure. OS can simply iterate through those mappings that are known to exist while Linux must check all page tables that *might* contain a specific mapping to see if it does, which can achieve $O(n^2)$ overhead in certain situations. It is because of this that OS tends to make better choices on which pages to reuse or swap when memory is stressed, giving it better performance under load. However, OS requires kernel tuning to accommodate large-shared-address-space situations such as those that can occur in a news system because it may run out of `pv_entry` structures.

Both Linux and OS need work in this area. OS is trying to maximize the advantage of a potentially sparse active-mapping model (not all processes need to map all pages of a shared library, for example), whereas Linux is trying to simplify its algorithms. OS generally has the performance advantage here at the cost of wasting a little extra memory, but OS breaks down in the case where a large file is massively shared across hundreds of processes. Linux, on the other hand, breaks down in the case where many processes are sparsely-mapping the same shared library and also runs non-optimally when trying to determine whether a page can be reused or not.

76.7 Page Coloring

We will end with the page coloring optimizations. Page coloring is a performance optimization designed to ensure that accesses to contiguous pages in virtual memory make the best use of the processor cache. In ancient times (i.e. 10+ years ago) processor caches tended to map virtual memory rather than physical memory. This led to a huge number of problems including having to clear the cache on every context switch in some cases, and problems with data aliasing in the cache. Modern processor caches map physical memory precisely to solve those problems. This means that two side-by-side pages in a processes address space may not correspond to two side-by-side pages in the cache. In fact, if you are not careful side-by-side pages in virtual memory could wind up using the same page in the processor cache—leading to cacheable data being thrown away prematurely and reducing CPU performance. This is true even with multi-way set-associative caches (though the effect is mitigated somewhat).

OS's memory allocation code implements page coloring optimizations, which means that the memory allocation code will attempt to locate free pages that are contiguous from the point of view of the cache. For example, if page 16 of physical memory is assigned to page 0 of a process's virtual memory and the cache can hold 4 pages, the page coloring code will not assign page 20 of physical memory to page 1 of a process's virtual memory. It would, instead, assign page 21 of physical memory. The page coloring code attempts to avoid assigning page 20 because this maps over the same cache memory as page 16 and would result in non-optimal caching. This code adds a significant amount of complexity to the VM memory allocation subsystem as you can well imagine, but the result is well worth the effort. Page Coloring makes VM memory as deterministic as physical memory in regards to cache performance.

76.8 Conclusion

Virtual memory in modern operating systems must address a number of different issues efficiently and for many different usage patterns. The modular and algorithmic approach that BSD has historically taken allows us to study and understand the current implementation as well as relatively cleanly replace large sections of the code. There have been a number of improvements to the OS VM system in the last several years, and work is ongoing.

76.9 Bonus QA session by Allen Briggs briggs@ninthwonder.com

Q: What is “the interleaving algorithm” that you refer to in your listing of the ills of the OS 3.X swap arrangements?

A: OS uses a fixed swap interleave which defaults to 4. This means that OS reserves space for four swap areas even if you only have one, two, or three. Since swap is interleaved the linear address space representing the “four swap areas” will be fragmented if you do not actually have four swap areas. For example, if you have two swap areas A and B OS’s address space representation for that swap area will be interleaved in blocks of 16 pages:

A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

OS 3.X uses a “sequential list of free regions” approach to accounting for the free swap areas. The idea is that large blocks of free linear space can be represented with a single list node (`kern/subr_rlist.c`). But due to the fragmentation the sequential list winds up being insanely fragmented. In the above example, completely unused swap will have A and B shown as “free” and C and D shown as “all allocated”. Each A-B sequence requires a list node to account for because C and D are holes, so the list node cannot be combined with the next A-B sequence.

Why do we interleave our swap space instead of just tack swap areas onto the end and do something fancier? Because it is a whole lot easier to allocate linear swaths of an address space and have the result automatically be interleaved across multiple disks than it is to try to put that sophistication elsewhere.

The fragmentation causes other problems. Being a linear list under 3.X, and having such a huge amount of inherent fragmentation, allocating and freeing swap winds up being an $O(N)$ algorithm instead of an $O(1)$ algorithm. Combined with other factors (heavy swapping) and you start getting into $O(N^2)$ and $O(N^3)$ levels of overhead, which is bad. The 3.X system may also need to allocate KVM during a swap operation to create a new list node which can lead to a deadlock if the system is trying to pageout pages in a low-memory situation.

Under 4.X we do not use a sequential list. Instead we use a radix tree and bitmaps of swap blocks rather than ranged list nodes. We take the hit of preallocating all the bitmaps required for the entire swap area up front but it winds up wasting less memory due to the use of a bitmap (one bit per block) instead of a linked list of nodes. The use of a radix tree instead of a sequential list gives us nearly $O(1)$ performance no matter how fragmented the tree becomes.

Q: How is the separation of clean and dirty (inactive) pages related to the situation where you see low cache queue counts and high active queue counts in `systat -vm`? Do the `systat` stats roll the active and dirty pages together for the active queue count?

I do not get the following:

It is important to note that the OS VM system attempts to separate clean and dirty pages for the express reason of avoiding unnecessary flushes of dirty pages (which eats I/O bandwidth), nor does it move pages between the various page queues gratuitously when the memory subsystem is not being stressed. This is why you will see some systems with very low cache queue counts and high active queue counts when doing a `systat -vm` command.

A: Yes, that is confusing. The relationship is “goal” verses “reality”. Our goal is to separate the pages but the reality is that if we are not in a memory crunch, we do not really have to.

What this means is that OS will not try very hard to separate out dirty pages (inactive queue) from clean pages (cache queue) when the system is not being stressed, nor will it try to deactivate pages (active queue -> inactive queue) when the system is not being stressed, even if they are not being used.

Q: In the MAN.LS.1 / `vmstat 1` example, would not some of the page faults be data page faults (COW from executable file to private page)? I.e., I would expect the page faults to be some zero-fill and some program data. Or are you implying that OS does do pre-COW for the program data?

A: A COW fault can be either zero-fill or program-data. The mechanism is the same either way because the backing program-data is almost certainly already in the cache. I am indeed lumping the two together. OS does not pre-COW program data or zero-fill, but it *does* pre-map pages that exist in its cache.

Q: In your section on page table optimizations, can you give a little more detail about `pv_entry` and `vm_page` (or should `vm_page` be `vm_pmap`—as in 4.4, cf. pp. 180-181 of McKusick, Bostic, Karel, Quarterman)? Specifically, what kind of operation/reaction would require scanning the mappings?

How does Linux do in the case where OS breaks down (sharing a large file mapping over many processes)?

A: A `vm_page` represents an (object,index#) tuple. A `pv_entry` represents a hardware page table entry (pte). If you have five processes sharing the same physical page, and three of those processes's page tables actually map the page, that page will be represented by a single `vm_page` structure and three `pv_entry` structures.

`pv_entry` structures only represent pages mapped by the MMU (one `pv_entry` represents one pte). This means that when we need to remove all hardware references to a `vm_page` (in order to reuse the page for something else, page it out, clear it, dirty it, and so forth) we can simply scan the linked list of `pv_entry`'s associated with that `vm_page` to remove or modify the pte's from their page tables.

Under Linux there is no such linked list. In order to remove all the hardware page table mappings for a `vm_page` linux must index into every VM object that *might* have mapped the page. For example, if you have 50 processes all mapping the same shared library and want to get rid of page X in that library, you need to index into the page table for each of those 50 processes even if only 10 of them have actually mapped the page. So Linux is trading off the simplicity of its design against performance. Many VM algorithms which are $O(1)$ or (small N) under OS wind up being $O(N)$, $O(N^2)$, or worse under Linux. Since the pte's representing a particular page in an object tend to be at the same offset in all the page tables they are mapped in, reducing the number of accesses into the page tables at the same pte offset will often avoid blowing away the L1 cache line for that offset, which can lead to better performance.

OS has added complexity (the `pv_entry` scheme) in order to increase performance (to limit page table accesses to *only* those pte's that need to be modified).

But OS has a scaling problem that Linux does not in that there are a limited number of `pv_entry` structures and this causes problems when you have massive sharing of data. In this case you may run out of `pv_entry` structures even though there is plenty of free memory available. This can be fixed easily enough by bumping up the number of `pv_entry` structures in the kernel config, but we really need to find a better way to do it.

In regards to the memory overhead of a page table verses the `pv_entry` scheme: Linux uses “permanent” page tables that are not throw away, but does not need a `pv_entry` for each potentially mapped pte. OS uses “throw away” page tables but adds in a `pv_entry` structure for each actually-mapped pte. I think memory utilization winds up being about the same, giving OS an algorithmic advantage with its ability to throw away page tables at will with very low overhead.

Q: Finally, in the page coloring section, it might help to have a little more description of what you mean here. I did not quite follow it.

A: Do you know how an L1 hardware memory cache works? I will explain: Consider a machine with 16MB of main memory but only 128K of L1 cache. Generally the way this cache works is that each 128K block of main memory uses the *same* 128K of cache. If you access offset 0 in main memory and then offset 128K in main memory you can wind up throwing away the cached data you read from offset 0!

Now, I am simplifying things greatly. What I just described is what is called a “direct mapped” hardware memory cache. Most modern caches are what are called 2-way-set-associative or 4-way-set-associative caches. The set-associatively allows you to access up to N different memory regions that overlap the same cache memory without destroying the previously cached data. But only N .

So if I have a 4-way set associative cache I can access offset 0, offset 128K, 256K and offset 384K and still be able

to access offset 0 again and have it come from the L1 cache. If I then access offset 512K, however, one of the four previously cached data objects will be thrown away by the cache.

It is extremely important... *extremely* important for most of a processor's memory accesses to be able to come from the L1 cache, because the L1 cache operates at the processor frequency. The moment you have an L1 cache miss and have to go to the L2 cache or to main memory, the processor will stall and potentially sit twiddling its fingers for *hundreds* of instructions worth of time waiting for a read from main memory to complete. Main memory (the dynamic ram you stuff into a computer) is *slow*, when compared to the speed of a modern processor core.

Ok, so now onto page coloring: All modern memory caches are what are known as *physical* caches. They cache physical memory addresses, not virtual memory addresses. This allows the cache to be left alone across a process context switch, which is very important.

But in the UNIX world you are dealing with virtual address spaces, not physical address spaces. Any program you write will see the virtual address space given to it. The actual *physical* pages underlying that virtual address space are not necessarily physically contiguous! In fact, you might have two pages that are side by side in a processes address space which wind up being at offset 0 and offset 128K in *physical* memory.

A program normally assumes that two side-by-side pages will be optimally cached. That is, that you can access data objects in both pages without having them blow away each other's cache entry. But this is only true if the physical pages underlying the virtual address space are contiguous (insofar as the cache is concerned).

This is what Page coloring does. Instead of assigning *random* physical pages to virtual addresses, which may result in non-optimal cache performance, Page coloring assigns *reasonably-contiguous* physical pages to virtual addresses. Thus programs can be written under the assumption that the characteristics of the underlying hardware cache are the same for their virtual address space as they would be if the program had been run directly in a physical address space.

Note that I say "reasonably" contiguous rather than simply "contiguous". From the point of view of a 128K direct mapped cache, the physical address 0 is the same as the physical address 128K. So two side-by-side pages in your virtual address space may wind up being offset 128K and offset 132K in physical memory, but could also easily be offset 128K and offset 4K in physical memory and still retain the same cache performance characteristics. So page-coloring does *not* have to assign truly contiguous pages of physical memory to contiguous pages of virtual memory, it just needs to make sure it assigns contiguous pages from the point of view of cache performance and operation.

OS Architecture Handbook

Author The FreeBSD Documentation Project

CHAP.BOOT CHAP.LOCKING CHAP.KOBJ CHAP.JAIL CHAP.SYSINIT CHAP.MAC CHAP.VM CHAP.SMP
CHAP.DRIVERBASICS CHAP.ISA CHAP.PCI CHAP.SCSI CHAP.USB CHAP.NEWBUS CHAP.SND
CHAP.PCCARD

MarshallKirkMcKusick KeithBostic MichaelJKarels JohnSQuarterman 1996 Addison-Wesley Publishing Company,
Inc. 0-201-54979-4 Addison-Wesley Publishing Company, Inc. The Design and Implementation of the 4.4 BSD
Operating System 1-2

CHAP.INDEX

FreeBSD Bibliography

Author The FreeBSD Documentation Project
BIBLIOGRAPHY

The Design and Implementation of the 4.4BSD Operating System

Author MarshallKirkMcKusick

Author KeithBostic

Author MichaelJ.Karels

Author JohnS.Quarterman

79.1 Design Overview of 4.4BSD

79.1.1 4.4BSD Facilities and the Kernel

The 4.4BSD kernel provides four basic facilities: processes, a filesystem, communications, and system startup. This section outlines where each of these four basic services is described in this book.

1. Processes constitute a thread of control in an address space. Mechanisms for creating, terminating, and otherwise controlling processes are described in Chapter 4. The system multiplexes separate virtual-address spaces for each process; this memory management is discussed in Chapter 5.
2. The user interface to the filesystem and devices is similar; common aspects are discussed in Chapter 6. The filesystem is a set of named files, organized in a tree-structured hierarchy of directories, and of operations to manipulate them, as presented in Chapter 7. Files reside on physical media such as disks. 4.4BSD supports several organizations of data on the disk, as set forth in Chapter 8. Access to files on remote machines is the subject of Chapter 9. Terminals are used to access the system; their operation is the subject of Chapter 10.
3. Communication mechanisms provided by traditional UNIX systems include simplex reliable byte streams between related processes (see pipes, Section 11.1), and notification of exceptional events (see signals, Section 4.7). 4.4BSD also has a general interprocess-communication facility. This facility, described in Chapter 11, uses access mechanisms distinct from those of the filesystem, but, once a connection is set up, a process can access it as though it were a pipe. There is a general networking framework, discussed in Chapter 12, that is normally used as a layer underlying the IPC facility. Chapter 13 describes a particular networking implementation in detail.
4. Any real operating system has operational issues, such as how to start it running. Startup and operational issues are described in Chapter 14.

Sections 2.3 through 2.14 present introductory material related to Chapters 3 through 14. We shall define terms, mention basic system calls, and explore historical developments. Finally, we shall give the reasons for many major design decisions.

The Kernel

The *kernel* is the part of the system that runs in protected mode and mediates access by all user programs to the underlying hardware (e.g., CPU, disks, terminals, network links) and software constructs (e.g., filesystem, network protocols). The kernel provides the basic system facilities; it creates and manages processes, and provides functions to access the filesystem and communication facilities. These functions, called *system calls* appear to user processes as library subroutines. These system calls are the only interface that processes have to these facilities. Details of the system-call mechanism are given in Chapter 3, as are descriptions of several kernel mechanisms that do not execute as the direct result of a process doing a system call.

A *kernel* in traditional operating-system terminology, is a small nucleus of software that provides only the minimal facilities necessary for implementing additional operating-system services. In contemporary research operating systems – such as Chorus ?, Mach ?, Tunis ?, and the V Kernel ? – this division of functionality is more than just a logical one. Services such as filesystems and networking protocols are implemented as client application processes of the nucleus or kernel.

The 4.4BSD kernel is not partitioned into multiple processes. This basic design decision was made in the earliest versions of UNIX. The first two implementations by Ken Thompson had no memory mapping, and thus made no hardware-enforced distinction between user and kernel space ?. A message-passing system could have been implemented as readily as the actually implemented model of kernel and user processes. The monolithic kernel was chosen for simplicity and performance. And the early kernels were small; the inclusion of facilities such as networking into the kernel has increased its size. The current trend in operating-systems research is to reduce the kernel size by placing such services in user space.

Users ordinarily interact with the system through a command-language interpreter, called a *shell*, and perhaps through additional user application programs. Such programs and the shell are implemented with processes. Details of such programs are beyond the scope of this book, which instead concentrates almost exclusively on the kernel.

Sections 2.3 and 2.4 describe the services provided by the 4.4BSD kernel, and give an overview of the latter's design. Later chapters describe the detailed design and implementation of these services as they appear in 4.4BSD.

79.1.2 Kernel Organization

In this section, we view the organization of the 4.4BSD kernel in two ways:

1. As a static body of software, categorized by the functionality offered by the modules that make up the kernel
2. By its dynamic operation, categorized according to the services provided to users

The largest part of the kernel implements the system services that applications access through system calls. In 4.4BSD, this software has been organized according to the following:

- Basic kernel facilities: timer and system-clock handling, descriptor management, and process management
- Memory-management support: paging and swapping
- Generic system interfaces: the I/O, control, and multiplexing operations performed on descriptors
- The filesystem: files, directories, pathname translation, file locking, and I/O buffer management
- Terminal-handling support: the terminal-interface driver and terminal line disciplines
- Interprocess-communication facilities: sockets
- Support for network communication: communication protocols and generic network facilities, such as routing

Category	Lines of code	Percentage of kernel
headers	9,393	4.6
initialization	1,107	0.6
kernel facilities	8,793	4.4
generic interfaces	4,782	2.4
interprocess communication	4,540	2.2
terminal handling	3,911	1.9
virtual memory	11,813	5.8
vnode management	7,954	3.9
filesystem naming	6,550	3.2
fast filestore	4,365	2.2
log-structure filestore	4,337	2.1
memory-based filestore	645	0.3
cd9660 filesystem	4,177	2.1
miscellaneous filesystems (10)	12,695	6.3
network filesystem	17,199	8.5
network communication	8,630	4.3
internet protocols	11,984	5.9
ISO protocols	23,924	11.8
X.25 protocols	10,626	5.3
XNS protocols	5,192	2.6

Table: Machine-independent software in the 4.4BSD kernel

Most of the software in these categories is machine independent and is portable across different hardware architectures.

The machine-dependent aspects of the kernel are isolated from the mainstream code. In particular, none of the machine-independent code contains conditional code for specific architecture. When an architecture-dependent action is needed, the machine-independent code calls an architecture-dependent function that is located in the machine-dependent code. The software that is machine dependent includes

- Low-level system-startup actions
- Trap and fault handling
- Low-level manipulation of the run-time context of a process
- Configuration and initialization of hardware devices
- Run-time support for I/O devices

Category	Lines of code	Percentage of kernel
machine dependent headers	1,562	0.8
device driver headers	3,495	1.7
device driver source	17,506	8.7
virtual memory	3,087	1.5
other machine dependent	6,287	3.1
routines in assembly language	3,014	1.5
HP/UX compatibility	4,683	2.3

Table: Machine-dependent software for the HP300 in the 4.4BSD kernel

? summarizes the machine-independent software that constitutes the 4.4BSD kernel for the HP300. The numbers in column 2 are for lines of C source code, header files, and assembly language. Virtually all the software in the kernel is written in the C programming language; less than 2 percent is written in assembly language. As the statistics in ? show, the machine-dependent software, excluding HP/UX and device support, accounts for a minuscule 6.9 percent of the kernel.

Only a small part of the kernel is devoted to initializing the system. This code is used when the system is *bootstrapped* into operation and is responsible for setting up the kernel hardware and software environment (see Chapter 14). Some operating systems (especially those with limited physical memory) discard or *overlay* the software that performs these functions after that software has been executed. The 4.4BSD kernel does not reclaim the memory used by the startup code because that memory space is barely 0.5 percent of the kernel resources used on a typical machine. Also, the startup code does not appear in one place in the kernel – it is scattered throughout, and it usually appears in places logically associated with what is being initialized.

79.1.3 Kernel Services

The boundary between the kernel- and user-level code is enforced by hardware-protection facilities provided by the underlying hardware. The kernel operates in a separate address space that is inaccessible to user processes. Privileged operations – such as starting I/O and halting the central processing unit (CPU) – are available to only the kernel. Applications request services from the kernel with *system calls*. System calls are used to cause the kernel to execute complicated operations, such as writing data to secondary storage, and simple operations, such as returning the current time of day. All system calls appear *synchronous* to applications: The application does not run while the kernel does the actions associated with a system call. The kernel may finish some operations associated with a system call after it has returned. For example, a *write* system call will copy the data to be written from the user process to a kernel buffer while the process waits, but will usually return from the system call before the kernel buffer is written to the disk.

A system call usually is implemented as a hardware trap that changes the CPU's execution mode and the current address-space mapping. Parameters supplied by users in system calls are validated by the kernel before being used. Such checking ensures the integrity of the system. All parameters passed into the kernel are copied into the kernel's address space, to ensure that validated parameters are not changed as a side effect of the system call. System-call results are returned by the kernel, either in hardware registers or by their values being copied to user-specified memory addresses. Like parameters passed into the kernel, addresses used for the return of results must be validated to ensure that they are part of an application's address space. If the kernel encounters an error while processing a system call, it returns an error code to the user. For the C programming language, this error code is stored in the global variable *errno*, and the function that executed the system call returns the value -1.

User applications and the kernel operate independently of each other. 4.4BSD does not store I/O control blocks or other operating-system-related data structures in the application's address space. Each user-level application is provided an independent address space in which it executes. The kernel makes most state changes, such as suspending a process while another is running, invisible to the processes involved.

79.1.4 Process Management

4.4BSD supports a multitasking environment. Each task or thread of execution is termed a *process*. The *context* of a 4.4BSD process consists of user-level state, including the contents of its address space and the run-time environment, and kernel-level state, which includes scheduling parameters, resource controls, and identification information. The context includes everything used by the kernel in providing services for the process. Users can create processes, control the processes' execution, and receive notification when the processes' execution status changes. Every process is assigned a unique value, termed a *process identifier* (PID). This value is used by the kernel to identify a process when reporting status changes to a user, and by a user when referencing a process in a system call.

The kernel creates a process by duplicating the context of another process. The new process is termed a *child process* of the original *parent process*. The context duplicated in process creation includes both the user-level execution state of the process and the process's system state managed by the kernel. Important components of the kernel state are described in Chapter 4.

Fig. 79.1: Process lifecycle

The process lifecycle is depicted in ?. A process may create a new process that is a copy of the original by using the *fork* system call. The *fork* call returns twice: once in the parent process, where the return value is the process identifier of the child, and once in the child process, where the return value is 0. The parent-child relationship induces a hierarchical structure on the set of processes in the system. The new process shares all its parent's resources, such as file descriptors, signal-handling status, and memory layout.

Although there are occasions when the new process is intended to be a copy of the parent, the loading and execution of a different program is a more useful and typical action. A process can overlay itself with the memory image of another program, passing to the newly created image a set of parameters, using the system call *execve*. One parameter is the name of a file whose contents are in a format recognized by the system – either a binary-executable file or a file that causes the execution of a specified interpreter program to process its contents.

A process may terminate by executing an *exit* system call, sending 8 bits of exit status to its parent. If a process wants to communicate more than a single byte of information with its parent, it must either set up an interprocess-communication channel using pipes or sockets, or use an intermediate file. Interprocess communication is discussed extensively in Chapter 11.

A process can suspend execution until any of its child processes terminate using the *wait* system call, which returns the PID and exit status of the terminated child process. A parent process can arrange to be notified by a signal when a child process exits or terminates abnormally. Using the *wait4* system call, the parent can retrieve information about the event that caused termination of the child process and about resources consumed by the process during its lifetime. If a process is orphaned because its parent exits before it is finished, then the kernel arranges for the child's exit status to be passed back to a special system process *init*: see Sections 3.1 and 14.6).

The details of how the kernel creates and destroys processes are given in Chapter 5.

Processes are scheduled for execution according to a *process-priority* parameter. This priority is managed by a kernel-based scheduling algorithm. Users can influence the scheduling of a process by specifying a parameter (*nice*) that weights the overall scheduling priority, but are still obligated to share the underlying CPU resources according to the kernel's scheduling policy.

Signals

The system defines a set of *signals* that may be delivered to a process. Signals in 4.4BSD are modeled after hardware interrupts. A process may specify a user-level subroutine to be a *handler* to which a signal should be delivered. When a signal is generated, it is blocked from further occurrence while it is being *caught* by the handler. Catching a signal involves saving the current process context and building a new one in which to run the handler. The signal is then delivered to the handler, which can either abort the process or return to the executing process (perhaps after setting a global variable). If the handler returns, the signal is unblocked and can be generated (and caught) again.

Alternatively, a process may specify that a signal is to be *ignored*, or that a default action, as determined by the kernel, is to be taken. The default action of certain signals is to terminate the process. This termination may be accompanied by creation of a *core file* that contains the current memory image of the process for use in postmortem debugging.

Some signals cannot be caught or ignored. These signals include *SIGKILL*, which kills runaway processes, and the job-control signal *SIGSTOP*.

A process may choose to have signals delivered on a special stack so that sophisticated software stack manipulations are possible. For example, a language supporting coroutines needs to provide a stack for each coroutine. The language run-time system can allocate these stacks by dividing up the single stack provided by 4.4BSD. If the kernel does not support a separate signal stack, the space allocated for each coroutine must be expanded by the amount of space required to catch a signal.

All signals have the same *priority*. If multiple signals are pending simultaneously, the order in which signals are delivered to a process is implementation specific. Signal handlers execute with the signal that caused their invocation to be blocked, but other signals may yet occur. Mechanisms are provided so that processes can protect critical sections of code against the occurrence of specified signals.

The detailed design and implementation of signals is described in Section 4.7.

Process Groups and Sessions

Processes are organized into *process groups*. Process groups are used to control access to terminals and to provide a means of distributing signals to collections of related processes. A process inherits its process group from its parent process. Mechanisms are provided by the kernel to allow a process to alter its process group or the process group of its descendants. Creating a new process group is easy; the value of a new process group is ordinarily the process identifier of the creating process.

The group of processes in a process group is sometimes referred to as a *job* and is manipulated by high-level system software, such as the shell. A common kind of job created by a shell is a *pipeline* of several processes connected by pipes, such that the output of the first process is the input of the second, the output of the second is the input of the third, and so forth. The shell creates such a job by forking a process for each stage of the pipeline, then putting all those processes into a separate process group.

A user process can send a signal to each process in a process group, as well as to a single process. A process in a specific process group may receive software interrupts affecting the group, causing the group to suspend or resume execution, or to be interrupted or terminated.

A terminal has a process-group identifier assigned to it. This identifier is normally set to the identifier of a process group associated with the terminal. A job-control shell may create a number of process groups associated with the same terminal; the terminal is the *controlling terminal* for each process in these groups. A process may read from a descriptor for its controlling terminal only if the terminal's process-group identifier matches that of the process. If the identifiers do not match, the process will be blocked if it attempts to read from the terminal. By changing the process-group identifier of the terminal, a shell can arbitrate a terminal among several different jobs. This arbitration is called *job control* and is described, with process groups, in Section 4.8.

Just as a set of related processes can be collected into a process group, a set of process groups can be collected into a *session*. The main uses for sessions are to create an isolated environment for a daemon process and its children, and to collect together a user's login shell and the jobs that that shell spawns.

79.1.5 Memory Management

Each process has its own private address space. The address space is initially divided into three logical segments: *text*, *data*, and *stack*. The text segment is read-only and contains the machine instructions of a program. The data and stack segments are both readable and writable. The data segment contains the initialized and uninitialized data portions of a program, whereas the stack segment holds the application's run-time stack. On most machines, the stack segment is extended automatically by the kernel as the process executes. A process can expand or contract its data segment by making a system call, whereas a process can change the size of its text segment only when the segment's contents are overlaid with data from the filesystem, or when debugging takes place. The initial contents of the segments of a child process are duplicates of the segments of a parent process.

The entire contents of a process address space do not need to be resident for a process to execute. If a process references a part of its address space that is not resident in main memory, the system *pages* the necessary information into memory. When system resources are scarce, the system uses a two-level approach to maintain available resources. If a modest amount of memory is available, the system will take memory resources away from processes if these resources have not been used recently. Should there be a severe resource shortage, the system will resort to *swapping* the entire context of a process to secondary storage. The *demand paging* and *swapping* done by the system are effectively transparent to processes. A process may, however, advise the system about expected future memory utilization as a performance aid.

BSD Memory-Management Design Decisions

The support of large sparse address spaces, mapped files, and shared memory was a requirement for 4.2BSD. An interface was specified, called *mmap*, that allowed unrelated processes to request a shared mapping of a file into their address spaces. If multiple processes mapped the same file into their address spaces, changes to the file's portion of an address space by one process would be reflected in the area mapped by the other processes, as well as in the file itself. Ultimately, 4.2BSD was shipped without the *mmap* interface, because of pressure to make other features, such as networking, available.

Further development of the *mmap* interface continued during the work on 4.3BSD. Over 40 companies and research groups participated in the discussions leading to the revised architecture that was described in the Berkeley Software Architecture Manual ?. Several of the companies have implemented the revised interface ?.

Once again, time pressure prevented 4.3BSD from providing an implementation of the interface. Although the latter could have been built into the existing 4.3BSD virtual-memory system, the developers decided not to put it in because that implementation was nearly 10 years old. Furthermore, the original virtual-memory design was based on the assumption that computer memories were small and expensive, whereas disks were locally connected, fast, large, and inexpensive. Thus, the virtual-memory system was designed to be frugal with its use of memory at the expense of generating extra disk traffic. In addition, the 4.3BSD implementation was riddled with VAX memory-management hardware dependencies that impeded its portability to other computer architectures. Finally, the virtual-memory system was not designed to support the tightly coupled multiprocessors that are becoming increasingly common and important today.

Attempts to improve the old implementation incrementally seemed doomed to failure. A completely new design, on the other hand, could take advantage of large memories, conserve disk transfers, and have the potential to run on multiprocessors. Consequently, the virtual-memory system was completely replaced in 4.4BSD. The 4.4BSD virtual-memory system is based on the Mach 2.0 VM system ? with updates from Mach 2.5 and Mach 3.0. It features efficient support for sharing, a clean separation of machine-independent and machine-dependent features, as well as (currently unused) multiprocessor support. Processes can map files anywhere in their address space. They can share parts of their address space by doing a shared mapping of the same file. Changes made by one process are visible in the address space of the other process, and also are written back to the file itself. Processes can also request private mappings of a file, which prevents any changes that they make from being visible to other processes mapping the file or being written back to the file itself.

Another issue with the virtual-memory system is the way that information is passed into the kernel when a system call is made. 4.4BSD always copies data from the process address space into a buffer in the kernel. For read or write operations that are transferring large quantities of data, doing the copy can be time consuming. An alternative to doing the copying is to remap the process memory into the kernel. The 4.4BSD kernel always copies the data for several reasons:

- Often, the user data are not page aligned and are not a multiple of the hardware page length.
- If the page is taken away from the process, it will no longer be able to reference that page. Some programs depend on the data remaining in the buffer even after those data have been written.
- If the process is allowed to keep a copy of the page (as it is in current 4.4BSD semantics), the page must be made *copy-on-write*. A copy-on-write page is one that is protected against being written by being made read-only. If the process attempts to modify the page, the kernel gets a write fault. The kernel then makes a copy of the page that the process can modify. Unfortunately, the typical process will immediately try to write new data to its output buffer, forcing the data to be copied anyway.
- When pages are remapped to new virtual-memory addresses, most memory-management hardware requires that the hardware address-translation cache be purged selectively. The cache purges are often slow. The net effect is that remapping is slower than copying for blocks of data less than 4 to 8 Kbyte.

The biggest incentives for memory mapping are the needs for accessing big files and for passing large quantities of data between processes. The *mmap* interface provides a way for both of these tasks to be done without copying.

Memory Management Inside the Kernel

The kernel often does allocations of memory that are needed for only the duration of a single system call. In a user process, such short-term memory would be allocated on the run-time stack. Because the kernel has a limited run-time stack, it is not feasible to allocate even moderate-sized blocks of memory on it. Consequently, such memory must be allocated through a more dynamic mechanism. For example, when the system must translate a pathname, it must allocate a 1-Kbyte buffer to hold the name. Other blocks of memory must be more persistent than a single system call, and thus could not be allocated on the stack even if there was space. An example is protocol-control blocks that remain throughout the duration of a network connection.

Demands for dynamic memory allocation in the kernel have increased as more services have been added. A generalized memory allocator reduces the complexity of writing code inside the kernel. Thus, the 4.4BSD kernel has a single memory allocator that can be used by any part of the system. It has an interface similar to the C library routines *malloc* and *free* that provide memory allocation to application programs ?. Like the C library interface, the allocation routine takes a parameter specifying the size of memory that is needed. The range of sizes for memory requests is not constrained; however, physical memory is allocated and is not paged. The *free* routine takes a pointer to the storage being freed, but does not require the size of the piece of memory being freed.

79.1.6 I/O System

The basic model of the UNIX I/O system is a sequence of bytes that can be accessed either randomly or sequentially. There are no *access methods* and no *control blocks* in a typical UNIX user process.

Different programs expect various levels of structure, but the kernel does not impose structure on I/O. For instance, the convention for text files is lines of ASCII characters separated by a single newline character (the ASCII line-feed character), but the kernel knows nothing about this convention. For the purposes of most programs, the model is further simplified to being a stream of data bytes, or an *I/O stream*. It is this single common data form that makes the characteristic UNIX tool-based approach work ?. An I/O stream from one program can be fed as input to almost any other program. (This kind of traditional UNIX I/O stream should not be confused with the Eighth Edition stream I/O system or with the System V, Release 3 STREAMS, both of which can be accessed as traditional I/O streams.)

Descriptors and I/O

UNIX processes use *descriptors* to reference I/O streams. Descriptors are small unsigned integers obtained from the *open* and *socket* system calls. The *open* system call takes as arguments the name of a file and a permission mode to specify whether the file should be open for reading or for writing, or for both. This system call also can be used to create a new, empty file. A *read* or *write* system call can be applied to a descriptor to transfer data. The *close* system call can be used to deallocate any descriptor.

Descriptors represent underlying objects supported by the kernel, and are created by system calls specific to the type of object. In 4.4BSD, three kinds of objects can be represented by descriptors: files, pipes, and sockets.

- A *file* is a linear array of bytes with at least one name. A file exists until all its names are deleted explicitly and no process holds a descriptor for it. A process acquires a descriptor for a file by opening that file's name with the *open* system call. I/O devices are accessed as files.
- A *pipe* is a linear array of bytes, as is a file, but it is used solely as an I/O stream, and it is unidirectional. It also has no name, and thus cannot be opened with *open*. Instead, it is created by the *pipe* system call, which returns two descriptors, one of which accepts input that is sent to the other descriptor reliably, without duplication, and in order. The system also supports a named pipe or FIFO. A FIFO has properties identical to a pipe, except that it appears in the filesystem; thus, it can be opened using the *open* system call. Two processes that wish to communicate each open the FIFO: One opens it for reading, the other for writing.
- A *socket* is a transient object that is used for interprocess communication; it exists only as long as some process holds a descriptor referring to it. A socket is created by the *socket* system call, which returns a descriptor for it.

There are different kinds of sockets that support various communication semantics, such as reliable delivery of data, preservation of message ordering, and preservation of message boundaries.

In systems before 4.2BSD, pipes were implemented using the filesystem; when sockets were introduced in 4.2BSD, pipes were reimplemented as sockets.

The kernel keeps for each process a *descriptor table*, which is a table that the kernel uses to translate the external representation of a descriptor into an internal representation. (The descriptor is merely an index into this table.) The descriptor table of a process is inherited from that process's parent, and thus access to the objects to which the descriptors refer also is inherited. The main ways that a process can obtain a descriptor are by opening or creation of an object, and by inheritance from the parent process. In addition, socket IPC allows passing of descriptors in messages between unrelated processes on the same machine.

Every valid descriptor has an associated *file offset* in bytes from the beginning of the object. Read and write operations start at this offset, which is updated after each data transfer. For objects that permit random access, the file offset also may be set with the *lseek* system call. Ordinary files permit random access, and some devices do, as well. Pipes and sockets do not.

When a process terminates, the kernel reclaims all the descriptors that were in use by that process. If the process was holding the final reference to an object, the object's manager is notified so that it can do any necessary cleanup actions, such as final deletion of a file or deallocation of a socket.

Descriptor Management

Most processes expect three descriptors to be open already when they start running. These descriptors are 0, 1, 2, more commonly known as *standard input*, *standard output*, and *standard error*, respectively. Usually, all three are associated with the user's terminal by the login process (see Section 14.6) and are inherited through *fork* and *exec* by processes run by the user. Thus, a program can read what the user types by reading standard input, and the program can send output to the user's screen by writing to standard output. The standard error descriptor also is open for writing and is used for error output, whereas standard output is used for ordinary output.

These (and other) descriptors can be mapped to objects other than the terminal; such mapping is called *I/O redirection*, and all the standard shells permit users to do it. The shell can direct the output of a program to a file by closing descriptor 1 (standard output) and opening the desired output file to produce a new descriptor 1. It can similarly redirect standard input to come from a file by closing descriptor 0 and opening the file.

Pipes allow the output of one program to be input to another program without rewriting or even relinking of either program. Instead of descriptor 1 (standard output) of the source program being set up to write to the terminal, it is set up to be the input descriptor of a pipe. Similarly, descriptor 0 (standard input) of the sink program is set up to reference the output of the pipe, instead of the terminal keyboard. The resulting set of two processes and the connecting pipe is known as a *pipeline*. Pipelines can be arbitrarily long series of processes connected by pipes.

The *open*, *pipe*, and *socket* system calls produce new descriptors with the lowest unused number usable for a descriptor. For pipelines to work, some mechanism must be provided to map such descriptors into 0 and 1. The *dup* system call creates a copy of a descriptor that points to the same file-table entry. The new descriptor is also the lowest unused one, but if the desired descriptor is closed first, *dup* can be used to do the desired mapping. Care is required, however: If descriptor 1 is desired, and descriptor 0 happens also to have been closed, descriptor 0 will be the result. To avoid this problem, the system provides the *dup2* system call; it is like *dup*, but it takes an additional argument specifying the number of the desired descriptor (if the desired descriptor was already open, *dup2* closes it before reusing it).

Devices

Hardware devices have filenames, and may be accessed by the user via the same system calls used for regular files. The kernel can distinguish a *device special file* or *special file*, and can determine to what device it refers, but most processes do not need to make this determination. Terminals, printers, and tape drives are all accessed as though they

were streams of bytes, like 4.4BSD disk files. Thus, device dependencies and peculiarities are kept in the kernel as much as possible, and even in the kernel most of them are segregated in the device drivers.

Hardware devices can be categorized as either *structured* or *unstructured*; they are known as *block* or *character* devices, respectively. Processes typically access devices through *special files* in the filesystem. I/O operations to these files are handled by kernel-resident software modules termed *device drivers*. Most network-communication hardware devices are accessible through only the interprocess-communication facilities, and do not have special files in the filesystem name space, because the *raw-socket* interface provides a more natural interface than does a special file.

Structured or block devices are typified by disks and magnetic tapes, and include most random-access devices. The kernel supports read-modify-write-type buffering actions on block-oriented structured devices to allow the latter to be read and written in a totally random byte-addressed fashion, like regular files. Filesystems are created on block devices.

Unstructured devices are those devices that do not support a block structure. Familiar unstructured devices are communication lines, raster plotters, and unbuffered magnetic tapes and disks. Unstructured devices typically support large block I/O transfers.

Unstructured files are called *character devices* because the first of these to be implemented were terminal device drivers. The kernel interface to the driver for these devices proved convenient for other devices that were not block structured.

Device special files are created by the *mknod* system call. There is an additional system call, *ioctl*, for manipulating the underlying device parameters of special files. The operations that can be done differ for each device. This system call allows the special characteristics of devices to be accessed, rather than overloading the semantics of other system calls. For example, there is an *ioctl* on a tape drive to write an end-of-tape mark, instead of there being a special or modified version of *write*.

Socket IPC

The 4.2BSD kernel introduced an IPC mechanism more flexible than pipes, based on *sockets*. A socket is an endpoint of communication referred to by a descriptor, just like a file or a pipe. Two processes can each create a socket, and then connect those two endpoints to produce a reliable byte stream. Once connected, the descriptors for the sockets can be read or written by processes, just as the latter would do with a pipe. The transparency of sockets allows the kernel to redirect the output of one process to the input of another process residing on another machine. A major difference between pipes and sockets is that pipes require a common parent process to set up the communications channel. A connection between sockets can be set up by two unrelated processes, possibly residing on different machines.

System V provides local interprocess communication through FIFOs (also known as *named pipes*). FIFOs appear as an object in the filesystem that unrelated processes can open and send data through in the same way as they would communicate through a pipe. Thus, FIFOs do not require a common parent to set them up; they can be connected after a pair of processes are up and running. Unlike sockets, FIFOs can be used on only a local machine; they cannot be used to communicate between processes on different machines. FIFOs are implemented in 4.4BSD only because they are required by the POSIX.1 standard. Their functionality is a subset of the socket interface.

The socket mechanism requires extensions to the traditional UNIX I/O system calls to provide the associated naming and connection semantics. Rather than overloading the existing interface, the developers used the existing interfaces to the extent that the latter worked without being changed, and designed new interfaces to handle the added semantics. The *read* and *write* system calls were used for byte-stream type connections, but six new system calls were added to allow sending and receiving addressed messages such as network datagrams. The system calls for writing messages include *send*, *sendto*, and *sendmsg*. The system calls for reading messages include *recv*, *recvfrom*, and *recvmsg*. In retrospect, the first two in each class are special cases of the others; *recvfrom* and *sendto* probably should have been added as library interfaces to *recvmsg* and *sendmsg*, respectively.

Scatter/Gather I/O

In addition to the traditional *read* and *write* system calls, 4.2BSD introduced the ability to do scatter/gather I/O. Scatter input uses the *readv* system call to allow a single read to be placed in several different buffers. Conversely, the *writev* system call allows several different buffers to be written in a single atomic write. Instead of passing a single buffer and length parameter, as is done with *read* and *write*, the process passes in a pointer to an array of buffers and lengths, along with a count describing the size of the array.

This facility allows buffers in different parts of a process address space to be written atomically, without the need to copy them to a single contiguous buffer. Atomic writes are necessary in the case where the underlying abstraction is record based, such as tape drives that output a tape block on each write request. It is also convenient to be able to read a single request into several different buffers (such as a record header into one place and the data into another). Although an application can simulate the ability to scatter data by reading the data into a large buffer and then copying the pieces to their intended destinations, the cost of memory-to-memory copying in such cases often would more than double the running time of the affected application.

Just as *send* and *recv* could have been implemented as library interfaces to *sendto* and *recvfrom*, it also would have been possible to simulate *read* with *readv* and *write* with *writev*. However, *read* and *write* are used so much more frequently that the added cost of simulating them would not have been worthwhile.

Multiple Filesystem Support

With the expansion of network computing, it became desirable to support both local and remote filesystems. To simplify the support of multiple filesystems, the developers added a new virtual node or *vnode* interface to the kernel. The set of operations exported from the *vnode* interface appear much like the filesystem operations previously supported by the local filesystem. However, they may be supported by a wide range of filesystem types:

- Local disk-based filesystems
- Files imported using a variety of remote filesystem protocols
- Read-only CD-ROM filesystems
- Filesystems providing special-purpose interfaces – for example, the */proc* filesystem

A few variants of 4.4BSD, such as FreeBSD, allow filesystems to be loaded dynamically when the filesystems are first referenced by the *mount* system call. The *vnode* interface is described in Section 6.5; its ancillary support routines are described in Section 6.6; several of the special-purpose filesystems are described in Section 6.7.

79.1.7 Filesystems

A regular file is a linear array of bytes, and can be read and written starting at any byte in the file. The kernel distinguishes no record boundaries in regular files, although many programs recognize line-feed characters as distinguishing the ends of lines, and other programs may impose other structure. No system-related information about a file is kept in the file itself, but the filesystem stores a small amount of ownership, protection, and usage information with each file.

A *filename* component is a string of up to 255 characters. These filenames are stored in a type of file called a *directory*. The information in a directory about a file is called a *directory entry* and includes, in addition to the filename, a pointer to the file itself. Directory entries may refer to other directories, as well as to plain files. A hierarchy of directories and files is thus formed, and is called a *filesystem*;

Fig. 79.2: A small filesystem

a small one is shown in ?. Directories may contain subdirectories, and there is no inherent limitation to the depth with which directory nesting may occur. To protect the consistency of the filesystem, the kernel does not permit processes

to write directly into directories. A filesystem may include not only plain files and directories, but also references to other objects, such as devices and sockets.

The filesystem forms a tree, the beginning of which is the *root directory*, sometimes referred to by the name *slash*, spelled with a single solidus character (/). The root directory contains files; in our example in Fig 2.2, it contains `vmunix`, a copy of the kernel-executable object file. It also contains directories; in this example, it contains the `usr` directory. Within the `usr` directory is the `bin` directory, which mostly contains executable object code of programs, such as the files `ls` and `vi`.

A process identifies a file by specifying that file's *pathname*, which is a string composed of zero or more filenames separated by slash (/) characters. The kernel associates two directories with each process for use in interpreting pathnames. A process's *root directory* is the topmost point in the filesystem that the process can access; it is ordinarily set to the root directory of the entire filesystem. A pathname beginning with a slash is called an *absolute pathname*, and is interpreted by the kernel starting with the process's root directory.

A pathname that does not begin with a slash is called a *relative pathname*, and is interpreted relative to the *current working directory* of the process. (This directory also is known by the shorter names *current directory* or *working directory*.) The current directory itself may be referred to directly by the name *dot*, spelled with a single period (.). The filename *dot-dot* (..) refers to a directory's parent directory. The root directory is its own parent.

A process may set its root directory with the *chroot* system call, and its current directory with the *chdir* system call. Any process may do *chdir* at any time, but *chroot* is permitted only a process with superuser privileges. *Chroot* is normally used to set up restricted access to the system.

Using the filesystem shown in Fig. 2.2, if a process has the root of the filesystem as its root directory, and has `/usr` as its current directory, it can refer to the file `vi` either from the root with the absolute pathname `/usr/bin/vi`, or from its current directory with the relative pathname `bin/vi`.

System utilities and databases are kept in certain well-known directories. Part of the well-defined hierarchy includes a directory that contains the *home directory* for each user – for example, `/usr/staff/mckusick` and `/usr/staff/karels` in Fig. 2.2. When users log in, the current working directory of their shell is set to the home directory. Within their home directories, users can create directories as easily as they can regular files. Thus, a user can build arbitrarily complex subhierarchies.

The user usually knows of only one filesystem, but the system may know that this one virtual filesystem is really composed of several physical filesystems, each on a different device. A physical filesystem may not span multiple hardware devices. Since most physical disk devices are divided into several logical devices, there may be more than one filesystem per physical device, but there will be no more than one per logical device. One filesystem – the filesystem that anchors all absolute pathnames – is called the *root filesystem*, and is always available. Others may be mounted; that is, they may be integrated into the directory hierarchy of the root filesystem. References to a directory that has a filesystem mounted on it are converted transparently by the kernel into references to the root directory of the mounted filesystem.

The *link* system call takes the name of an existing file and another name to create for that file. After a successful *link*, the file can be accessed by either filename. A filename can be removed with the *unlink* system call. When the final name for a file is removed (and the final process that has the file open closes it), the file is deleted.

Files are organized hierarchically in *directories*. A directory is a type of file, but, in contrast to regular files, a directory has a structure imposed on it by the system. A process can read a directory as it would an ordinary file, but only the kernel is permitted to modify a directory. Directories are created by the *mkdir* system call and are removed by the *rmdir* system call. Before 4.2BSD, the *mkdir* and *rmdir* system calls were implemented by a series of *link* and *unlink* system calls being done. There were three reasons for adding systems calls explicitly to create and delete directories:

1. The operation could be made atomic. If the system crashed, the directory would not be left half-constructed, as could happen when a series of link operations were used.
2. When a networked filesystem is being run, the creation and deletion of files and directories need to be specified atomically so that they can be serialized.

3. When supporting non-UNIX filesystems, such as an MS-DOS filesystem, on another partition of the disk, the other filesystem may not support link operations. Although other filesystems might support the concept of directories, they probably would not create and delete the directories with links, as the UNIX filesystem does. Consequently, they could create and delete directories only if explicit directory create and delete requests were presented.

The *chown* system call sets the owner and group of a file, and *chmod* changes protection attributes. *Stat* applied to a filename can be used to read back such properties of a file. The *fchown*, *fchmod*, and *fstat* system calls are applied to a descriptor, instead of to a filename, to do the same set of operations. The *rename* system call can be used to give a file a new name in the filesystem, replacing one of the file's old names. Like the directory-creation and directory-deletion operations, the *rename* system call was added to 4.2BSD to provide atomicity to name changes in the local filesystem. Later, it proved useful explicitly to export renaming operations to foreign filesystems and over the network.

The *truncate* system call was added to 4.2BSD to allow files to be shortened to an arbitrary offset. The call was added primarily in support of the Fortran run-time library, which has the semantics such that the end of a random-access file is set to be wherever the program most recently accessed that file. Without the *truncate* system call, the only way to shorten a file was to copy the part that was desired to a new file, to delete the old file, then to rename the copy to the original name. As well as this algorithm being slow, the library could potentially fail on a full filesystem.

Once the filesystem had the ability to shorten files, the kernel took advantage of that ability to shorten large empty directories. The advantage of shortening empty directories is that it reduces the time spent in the kernel searching them when names are being created or deleted.

Newly created files are assigned the user identifier of the process that created them and the group identifier of the directory in which they were created. A three-level access-control mechanism is provided for the protection of files. These three levels specify the accessibility of a file to

1. The user who owns the file
2. The group that owns the file
3. Everyone else

Each level of access has separate indicators for read permission, write permission, and execute permission.

Files are created with zero length, and may grow when they are written. While a file is open, the system maintains a pointer into the file indicating the current location in the file associated with the descriptor. This pointer can be moved about in the file in a random-access fashion. Processes sharing a file descriptor through a *fork* or *dup* system call share the current location pointer. Descriptors created by separate *open* system calls have separate current location pointers. Files may have *holes* in them. Holes are void areas in the linear extent of the file where data have never been written. A process can create these holes by positioning the pointer past the current end-of-file and writing. When read, holes are treated by the system as zero-valued bytes.

Earlier UNIX systems had a limit of 14 characters per filename component. This limitation was often a problem. For example, in addition to the natural desire of users to give files long descriptive names, a common way of forming filenames is as *basename.extension*, where the extension (indicating the kind of file, such as *.c* for C source or *.o* for intermediate binary object) is one to three characters, leaving 10 to 12 characters for the basename. Source-code-control systems and editors usually take up another two characters, either as a prefix or a suffix, for their purposes, leaving eight to 10 characters. It is easy to use 10 or 12 characters in a single English word as a basename (e.g., “multiplexer”).

It is possible to keep within these limits, but it is inconvenient or even dangerous, because other UNIX systems accept strings longer than the limit when creating files, but then *truncate* to the limit. A C language source file named *multiplexer.c* (already 13 characters) might have a source-code-control file with *s.* prepended, producing a filename *s.multiplexer* that is indistinguishable from the source-code-control file for *multiplexer.ms*, a file containing *troff* source for documentation for the C program. The contents of the two original files could easily get confused with no warning from the source-code-control system. Careful coding can detect this problem, but the long filenames first introduced in 4.2BSD practically eliminate it.

79.1.8 Filestores

The operations defined for local filesystems are divided into two parts. Common to all local filesystems are hierarchical naming, locking, quotas, attribute management, and protection. These features are independent of how the data will be stored. 4.4BSD has a single implementation to provide these semantics.

The other part of the local filesystem is the organization and management of the data on the storage media. Laying out the contents of files on the storage media is the responsibility of the filestore. 4.4BSD supports three different filestore layouts:

- The traditional Berkeley Fast Filesystem
- The log-structured filesystem, based on the Sprite operating-system design ?
- A memory-based filesystem

Although the organizations of these filestores are completely different, these differences are indistinguishable to the processes using the filestores.

The Fast Filesystem organizes data into cylinder groups. Files that are likely to be accessed together, based on their locations in the filesystem hierarchy, are stored in the same cylinder group. Files that are not expected to be accessed together are moved into different cylinder groups. Thus, files written at the same time may be placed far apart on the disk.

The log-structured filesystem organizes data as a log. All data being written at any point in time are gathered together, and are written at the same disk location. Data are never overwritten; instead, a new copy of the file is written that replaces the old one. The old files are reclaimed by a garbage-collection process that runs when the filesystem becomes full and additional free space is needed.

The memory-based filesystem is designed to store data in virtual memory. It is used for filesystems that need to support fast but temporary data, such as `/tmp`. The goal of the memory-based filesystem is to keep the storage packed as compactly as possible to minimize the usage of virtual-memory resources.

79.1.9 Network Filesystem

Initially, networking was used to transfer data from one machine to another. Later, it evolved to allowing users to log in remotely to another machine. The next logical step was to bring the data to the user, instead of having the user go to the data – and network filesystems were born. Users working locally do not experience the network delays on each keystroke, so they have a more responsive environment.

Bringing the filesystem to a local machine was among the first of the major client-server applications. The *server* is the remote machine that exports one or more of its filesystems. The *client* is the local machine that imports those filesystems. From the local client's point of view, a remotely mounted filesystem appears in the file-tree name space just like any other locally mounted filesystem. Local clients can change into directories on the remote filesystem, and can read, write, and execute binaries within that remote filesystem identically to the way that they can do these operations on a local filesystem.

When the local client does an operation on a remote filesystem, the request is packaged and is sent to the server. The server does the requested operation and returns either the requested information or an error indicating why the request was denied. To get reasonable performance, the client must cache frequently accessed data. The complexity of remote filesystems lies in maintaining cache consistency between the server and its many clients.

Although many remote-filesystem protocols have been developed over the years, the most pervasive one in use among UNIX systems is the Network Filesystem (NFS), whose protocol and most widely used implementation were done by Sun Microsystems. The 4.4BSD kernel supports the NFS protocol, although the implementation was done independently from the protocol specification ?. The NFS protocol is described in Chapter 9.

79.1.10 Terminals

Terminals support the standard system I/O operations, as well as a collection of terminal-specific operations to control input-character editing and output delays. At the lowest level are the terminal device drivers that control the hardware terminal ports. Terminal input is handled according to the underlying communication characteristics, such as baud rate, and according to a set of software-controllable parameters, such as parity checking.

Layered above the terminal device drivers are line disciplines that provide various degrees of character processing. The default line discipline is selected when a port is being used for an interactive login. The line discipline is run in *canonical mode*; input is processed to provide standard line-oriented editing functions, and input is presented to a process on a line-by-line basis.

Screen editors and programs that communicate with other computers generally run in *noncanonical mode* (also commonly referred to as *raw mode* or *character-at-a-time mode*). In this mode, input is passed through to the reading process immediately and without interpretation. All special-character input processing is disabled, no erase or other line editing processing is done, and all characters are passed to the program that is reading from the terminal.

It is possible to configure the terminal in thousands of combinations between these two extremes. For example, a screen editor that wanted to receive user interrupts asynchronously might enable the special characters that generate signals and enable output flow control, but otherwise run in noncanonical mode; all other characters would be passed through to the process uninterpreted.

On output, the terminal handler provides simple formatting services, including

- Converting the line-feed character to the two-character carriage-return-line-feed sequence
- Inserting delays after certain standard control characters
- Expanding tabs
- Displaying echoed nongraphic ASCII characters as a two-character sequence of the form “^C” (i.e., the ASCII caret character followed by the ASCII character that is the character’s value offset from the ASCII “@” character).

Each of these formatting services can be disabled individually by a process through control requests.

79.1.11 Interprocess Communication

Interprocess communication in 4.4BSD is organized in *communication domains*. Domains currently supported include the *local domain*, for communication between processes executing on the same machine; the *internet domain*, for communication between processes using the TCP/IP protocol suite (perhaps within the Internet); the ISO/OSI protocol family for communication between sites required to run them; and the *XNS domain*, for communication between processes using the XEROX Network Systems (XNS) protocols.

Within a domain, communication takes place between communication endpoints known as *sockets*. As mentioned in Section 2.6, the *socket* system call creates a socket and returns a descriptor; other IPC system calls are described in Chapter 11. Each socket has a type that defines its communications semantics; these semantics include properties such as reliability, ordering, and prevention of duplication of messages.

Each socket has associated with it a *communication protocol*. This protocol provides the semantics required by the socket according to the latter’s type. Applications may request a specific protocol when creating a socket, or may allow the system to select a protocol that is appropriate for the type of socket being created.

Sockets may have addresses bound to them. The form and meaning of socket addresses are dependent on the communication domain in which the socket is created. Binding a name to a socket in the local domain causes a file to be created in the filesystem.

Normal data transmitted and received through sockets are untyped. Data-representation issues are the responsibility of libraries built on top of the interprocess-communication facilities. In addition to transporting normal data, commu-

nication domains may support the transmission and reception of specially typed data, termed *access rights*. The local domain, for example, uses this facility to pass descriptors between processes.

Networking implementations on UNIX before 4.2BSD usually worked by overloading the character-device interfaces. One goal of the socket interface was for naive programs to be able to work without change on stream-style connections. Such programs can work only if the *read* and *write* systems calls are unchanged. Consequently, the original interfaces were left intact, and were made to work on stream-type sockets. A new interface was added for more complicated sockets, such as those used to send datagrams, with which a destination address must be presented with each *send* call.

Another benefit is that the new interface is highly portable. Shortly after a test release was available from Berkeley, the socket interface had been ported to System III by a UNIX vendor (although AT&T did not support the socket interface until the release of System V Release 4, deciding instead to use the Eighth Edition stream mechanism). The socket interface was also ported to run in many Ethernet boards by vendors, such as Excelan and Interlan, that were selling into the PC market, where the machines were too small to run networking in the main processor. More recently, the socket interface was used as the basis for Microsoft's Winsock networking interface for Windows.

79.1.12 Network Communication

Some of the communication domains supported by the *socket* IPC mechanism provide access to network protocols. These protocols are implemented as a separate software layer logically below the socket software in the kernel. The kernel provides many ancillary services, such as buffer management, message routing, standardized interfaces to the protocols, and interfaces to the network interface drivers for the use of the various network protocols.

At the time that 4.2BSD was being implemented, there were many networking protocols in use or under development, each with its own strengths and weaknesses. There was no clearly superior protocol or protocol suite. By supporting multiple protocols, 4.2BSD could provide interoperability and resource sharing among the diverse set of machines that was available in the Berkeley environment. Multiple-protocol support also provides for future changes. Today's protocols designed for 10- to 100-Mbit-per-second Ethernets are likely to be inadequate for tomorrow's 1- to 10-Gbit-per-second fiber-optic networks. Consequently, the network-communication layer is designed to support multiple protocols. New protocols are added to the kernel without the support for older protocols being affected. Older applications can continue to operate using the old protocol over the same physical network as is used by newer applications running with a newer network protocol.

79.1.13 Network Implementation

The first protocol suite implemented in 4.2BSD was DARPA's Transmission Control Protocol/Internet Protocol (TCP/IP). The CSRG chose TCP/IP as the first network to incorporate into the socket IPC framework, because a 4.1BSD-based implementation was publicly available from a DARPA-sponsored project at Bolt, Beranek, and Newman (BBN). That was an influential choice: The 4.2BSD implementation is the main reason for the extremely widespread use of this protocol suite. Later performance and capability improvements to the TCP/IP implementation have also been widely adopted. The TCP/IP implementation is described in detail in Chapter 13.

The release of 4.3BSD added the Xerox Network Systems (XNS) protocol suite, partly building on work done at the University of Maryland and at Cornell University. This suite was needed to connect isolated machines that could not communicate using TCP/IP.

The release of 4.4BSD added the ISO protocol suite because of the latter's increasing visibility both within and outside the United States. Because of the somewhat different semantics defined for the ISO protocols, some minor changes were required in the socket interface to accommodate these semantics. The changes were made such that they were invisible to clients of other existing protocols. The ISO protocols also required extensive addition to the two-level routing tables provided by the kernel in 4.3BSD. The greatly expanded routing capabilities of 4.4BSD include arbitrary levels of routing with variable-length addresses and network masks.

79.1.14 System Operation

Bootstrapping mechanisms are used to start the system running. First, the 4.4BSD kernel must be loaded into the main memory of the processor. Once loaded, it must go through an initialization phase to set the hardware into a known state. Next, the kernel must do autoconfiguration, a process that finds and configures the peripherals that are attached to the processor. The system begins running in single-user mode while a start-up script does disk checks and starts the accounting and quota checking. Finally, the start-up script starts the general system services and brings up the system to full multiuser operation.

During multiuser operation, processes wait for login requests on the terminal lines and network ports that have been configured for user access. When a login request is detected, a login process is spawned and user validation is done. When the login validation is successful, a login shell is created from which the user can run additional processes.

79.2 References

Accetta et al, 1986 Mach: A New Kernel Foundation for UNIX Development” M.Accetta R.Baron W.Bolosky D.Golub R.Rashid A.Tevanian M.Young 93-113 USENIX Association Conference Proceedings USENIX Association June 1986

Cheriton, 1988 The V Distributed System D. R.Cheriton 314-333 Comm ACM, 31, 3 March 1988

Ewens et al, 1985 Tunis: A Distributed Multiprocessor Operating System P.Ewens D. R.Blythe M.Funkenhauser R. C.Holt 247-254 USENIX Association Conference Proceedings USENIX Association June 1985

Gingell et al, 1987 Virtual Memory Architecture in SunOS R.Gingell J.Moran W.Shannon 81-94 USENIX Association Conference Proceedings USENIX Association June 1987

Kernighan & Pike, 1984 The UNIX Programming Environment B. W.Kernighan R.Pike Prentice-Hall Englewood Cliffs NJ 1984

Macklem, 1994 The 4.4BSD NFS Implementation R.Macklem 6:1-14 4.4BSD System Manager’s Manual O’Reilly & Associates, Inc. Sebastopol CA 1994

McKusick & Karels, 1988 Design of a General Purpose Memory Allocator for the 4.3BSD UNIX Kernel M. K.McKusick M. J.Karels 295-304 USENIX Association Conference Proceedings USENIX Association June 1988

McKusick et al, 1994 Berkeley Software Architecture Manual, 4.4BSD Edition M. K.McKusick M. J.Karels S. J.Leffler W. N.Joy R. S.Faber 5:1-42 4.4BSD Programmer’s Supplementary Documents O’Reilly & Associates, Inc. Sebastopol CA 1994

Ritchie, 1988 Early Kernel Design private communication D. M.Ritchie March 1988

Rosenblum & Ousterhout, 1992 The Design and Implementation of a Log-Structured File System M.Rosenblum K.Ousterhout 26-52 ACM Transactions on Computer Systems, 10, 1 Association for Computing Machinery February 1992

Rozier et al, 1988 Chorus Distributed Operating Systems M.Rozier V.Abrossimov F.Armand I.Boule M.Gien M.Guillemont F.Herrmann C.Kaiser S.Langlois P.Leonard W.Neuhauser 305-370 USENIX Computing Systems, 1, 4 Fall 1988

Tevanian, 1987 Architecture-Independent Virtual Memory Management for Parallel and Distributed Environments: The Mach Approach Technical Report CMU-CS-88-106, A.Tevanian Department of Computer Science, Carnegie-Mellon University Pittsburgh PA December 1987

A project model for the FreeBSD Project

Author NiklasSaers

Date February 1st, 2003

80.1 Foreword

Up until now, the FreeBSD project has released a number of described techniques to do different parts of work. However, a project model summarising how the project is structured is needed because of the increasing amount of project members.¹ This paper will provide such a project model and is donated to the FreeBSD Documentation project where it can evolve together with the project so that it can at any point in time reflect the way the project works. It is based on ?.

I would like to thank the following people for taking the time to explain things that were unclear to me and for proofreading the document.

- Andrey A. Chernov ache@freebsd.org
- Bruce A. Mah bmah@freebsd.org
- Dag-Erling Smørgrav des@freebsd.org
- Giorgos Keramidas keramida@freebsd.org
- Ingvil Hovig ingvil.hovig@skatteetaten.no
- Jesper Holck jeh.inf@cbs.dk
- John Baldwin jhb@freebsd.org
- John Polstra jdp@freebsd.org
- Kirk McKusick mckusick@freebsd.org
- Mark Linimon linimon@freebsd.org
- Marleen Devos
- Niels Jørgenssen nielsj@ruc.dk
- Nik Clayton nik@freebsd.org
- Poul-Henning Kamp phk@freebsd.org
- Simon L. Nielsen simon@freebsd.org

¹ This goes hand-in-hand with Brooks' law that "adding another person to a late project will make it later" since it will increase the communication needs ?. A project model is a tool to reduce the communication needs.

80.2 Overview

A project model is a means to reduce the communications overhead in a project. As shown by ?, increasing the number of project participants increases the communication in the project exponentially. FreeBSD has during the past few years increased both its mass of active users and committers, and the communication in the project has risen accordingly. This project model will serve to reduce this overhead by providing an up-to-date description of the project.

During the Core elections in 2002, Mark Murray stated “I am opposed to a long rule-book, as that satisfies lawyer-tendencies, and is counter to the technocentricity that the project so badly needs.” ?. This project model is not meant to be a tool to justify creating impositions for developers, but as a tool to facilitate coordination. It is meant as a description of the project, with an overview of how the different processes are executed. It is an introduction to how the FreeBSD project works.

The FreeBSD project model will be described as of July 1st, 2004. It is based on the Niels Jørgensen’s paper ?, FreeBSD’s official documents, discussions on FreeBSD mailing lists and interviews with developers.

After providing definitions of terms used, this document will outline the organisational structure (including role descriptions and communication lines), discuss the methodology model and after presenting the tools used for process control, it will present the defined processes. Finally it will outline major sub-projects of the FreeBSD project.

?, Section 1.2 and 1.3 give the vision and the architectural guidelines for the project. The vision is “To produce the best UNIX-like operating system package possible, with due respect to the original software tools ideology as well as usability, performance and stability.” The architectural guidelines help determine whether a problem that someone wants to be solved is within the scope of the project

80.3 Definitions

80.3.1 Activity

An “activity” is an element of work performed during the course of a project ?. It has an output and leads towards an outcome. Such an output can either be an input to another activity or a part of the process’ delivery.

80.3.2 Process

A “process” is a series of activities that lead towards a particular outcome. A process can consist of one or more sub-processes. An example of a process is software design.

80.3.3 Hat

A “hat” is synonymous with role. A hat has certain responsibilities in a process and for the process outcome. The hat executes activities. It is well defined what issues the hat should be contacted about by the project members and people outside the project.

80.3.4 Outcome

An “outcome” is the final output of the process. This is synonymous with deliverable, that is defined as “any measurable, tangible, verifiable outcome, result or item that must be produced to complete a project or part of a project. Often used more narrowly in reference to an external deliverable, which is a deliverable that is subject to approval by the project sponsor or customer” by ?. Examples of outcomes are a piece of software, a decision made or a report written.

80.3.5 FreeBSD

When saying “FreeBSD” we will mean the BSD derivative UNIX-like operating system FreeBSD, whereas when saying “the FreeBSD Project” we will mean the project organisation.

80.4 Organisational structure

While no-one takes ownership of FreeBSD, the FreeBSD organisation is divided into core, committers and contributors and is part of the FreeBSD community that lives around it.

Number of committers has been determined by going through CVS logs from January 1st, 2004 to December 31st, 2004 and contributors by going through the list of contributions and problem reports.

The main resource in the FreeBSD community is its developers: the committers and contributors. It is with their contributions that the project can move forward. Regular developers are referred to as contributors. As by January 1st, 2003, there are an estimated 5500 contributors on the project.

Committers are developers with the privilege of being able to commit changes. These are usually the most active developers who are willing to spend their time not only integrating their own code but integrating code submitted by the developers who do not have this privilege. They are also the developers who elect the core team, and they have access to closed discussions.

The project can be grouped into four distinct separate parts, and most developers will focus their involvement in one part of FreeBSD. The four parts are kernel development, userland development, ports and documentation. When referring to the base system, both kernel and userland is meant.

This split changes our triangle to look like this:

Number of committers per area has been determined by going through CVS logs from January 1st, 2004 to December 31st, 2004. Note that many committers work in multiple areas, making the total number higher than the real number of committers. The total number of committers at that time was 269.

Committers fall into three groups: committers who are only concerned with one area of the project (for instance file systems), committers who are involved only with one sub-project and committers who commit to different parts of the code, including sub-projects. Because some committers work on different parts, the total number in the committers section of the triangle is higher than in the above triangle.

The kernel is the main building block of FreeBSD. While the userland applications are protected against faults in other userland applications, the entire system is vulnerable to errors in the kernel. This, combined with the vast amount of dependencies in the kernel and that it is not easy to see all the consequences of a kernel change, demands developers with a relative full understanding of the kernel. Multiple development efforts in the kernel also requires a closer coordination than userland applications do.

The core utilities, known as userland, provide the interface that identifies FreeBSD, both user interface, shared libraries and external interfaces to connecting clients. Currently, 162 people are involved in userland development and maintenance, many being maintainers for their own part of the code. Maintainership will be discussed in the ? section.

Documentation is handled by ? and includes all documents surrounding the FreeBSD project, including the web pages. There were during 2004 101 people making commits to the FreeBSD Documentation Project.

Ports is the collection of meta-data that is needed to make software packages build correctly on FreeBSD. An example of a port is the port for the web-browser Mozilla. It contains information about where to fetch the source, what patches to apply and how, and how the package should be installed on the system. This allows automated tools to fetch, build and install the package. As of this writing, there are more than 12600 ports available. ², ranging from web servers to games, programming languages and most of the application types that are in use on modern computers. Ports will be discussed further in the section ?.

² Statistics are generated by counting the number of entries in the file fetched by portsdb by April 1st, 2005. portsdb is a part of the port sysutils/portupgrade.

80.5 Methodology model

80.5.1 Development model

There is no defined model for how people write code in FreeBSD. However, Niels Jørgenssen has suggested a model of how written code is integrated into the project.

The “development release” is the FreeBSD-CURRENT (“-CURRENT”) branch and the “production release” is the FreeBSD-STABLE branch (“-STABLE”) ?.

This is a model for one change, and shows that after coding, developers seek community review and try integrating it with their own systems. After integrating the change into the development release, called FreeBSD-CURRENT, it is tested by many users and developers in the FreeBSD community. After it has gone through enough testing, it is merged into the production release, called FreeBSD-STABLE. Unless each stage is finished successfully, the developer needs to go back and make modifications in the code and restart the process. To integrate a change with either -CURRENT or -STABLE is called making a commit.

Jørgensen found that most FreeBSD developers work individually, meaning that this model is used in parallel by many developers on the different ongoing development efforts. A developer can also be working on multiple changes, so that while he is waiting for review or people to test one or more of his changes, he may be writing another change.

As each commit represents an increment, this is a massively incremental model. The commits are in fact so frequent that during one year ³, 85427 commits were made, making a daily average of 233 commits.

Within the “code” bracket in Jørgensen’s figure, each programmer has his own working style and follows his own development models. The bracket could very well have been called “development” as it includes requirements gathering and analysis, system and detailed design, implementation and verification. However, the only output from these stages is the source code or system documentation.

From a stepwise model’s perspective (such as the waterfall model), the other brackets can be seen as further verification and system integration. This system integration is also important to see if a change is accepted by the community. Up until the code is committed, the developer is free to choose how much to communicate about it to the rest of the project. In order for -CURRENT to work as a buffer (so that bright ideas that had some undiscovered drawbacks can be backed out) the minimum time a commit should be in -CURRENT before merging it to -STABLE is 3 days. Such a merge is referred to as an MFC (Merge From Current).

It is important to notice the word “change”. Most commits do not contain radical new features, but are maintenance updates.

The only exceptions from this model are security fixes and changes to features that are deprecated in the -CURRENT branch. In these cases, changes can be committed directly to the -STABLE branch.

In addition to many people working on the project, there are many related projects to the FreeBSD Project. These are either projects developing brand new features, sub-projects or projects whose outcome is incorporated into FreeBSD ⁴. These projects fit into the FreeBSD Project just like regular development efforts: they produce code that is integrated with the FreeBSD Project. However, some of them (like Ports and Documentation) have the privilege of being applicable to both branches or commit directly to both -CURRENT and -STABLE.

There is no standards to how design should be done, nor is design collected in a centralised repository. The main design is that of 4.4BSD. ⁵ As design is a part of the “Code” bracket in Jørgenssen’s model, it is up to every developer or sub-project how this should be done. Even if the design should be stored in a central repository, the output from the design stages would be of limited use as the differences of methodologies would make them poorly if at all interoperable. For

³ The period from January 1st, 2004 to December 31st, 2004 was examined to find this number.

⁴ For instance, the development of the Bluetooth stack started as a sub-project until it was deemed stable enough to be merged into the -CURRENT branch. Now it is a part of the core FreeBSD system.

⁵ According to Kirk McKusick, after 20 years of developing UNIX operating systems, the interfaces are for the most part figured out. There is therefore no need for much design. However, new applications of the system and new hardware leads to some implementations being more beneficial than those that used to be preferred. One example is the introduction of web browsing that made the normal TCP/IP connection a short burst of data rather than a steady stream over a longer period of time.

the overall design of the project, the project relies on the sub-projects to negotiate fit interfaces between each other rather than to dictate interfacing.

80.5.2 Release branches

The releases of FreeBSD is best illustrated by a tree with many branches where each major branch represents a major version. Minor versions are represented by branches of the major branches.

In the following release tree, arrows that follow one-another in a particular direction represent a branch. Boxes with full lines and diamonds represent official releases. Boxes with dotted lines represent the development branch at that time. Security branches are represented by ovals. Diamonds differ from boxes in that they represent a fork, meaning a place where a branch splits into two branches where one of the branches becomes a sub-branch. For example, at 4.0-RELEASE the 4.0-CURRENT branch split into 4-STABLE and 5.0-CURRENT. At 4.5-RELEASE, the branch forked off a security branch called RELENG_4_5.

The latest -CURRENT version is always referred to as -CURRENT, while the latest -STABLE release is always referred to as -STABLE. In this figure, -STABLE refers to 4-STABLE while -CURRENT refers to 5.0-CURRENT following 5.0-RELEASE. ?

A “major release” is always made from the -CURRENT branch. However, the -CURRENT branch does not need to fork at that point in time, but can focus on stabilising. An example of this is that following 3.0-RELEASE, 3.1-RELEASE was also a continuation of the -CURRENT-branch, and -CURRENT did not become a true development branch until this version was released and the 3-STABLE branch was forked. When -CURRENT returns to becoming a development branch, it can only be followed by a major release. 5-STABLE is predicted to be forked off 5.0-CURRENT at around 5.3-RELEASE. It is not until 5-STABLE is forked that the development branch will be branded 6.0-CURRENT.

A “minor release” is made from the -CURRENT branch following a major release, or from the -STABLE branch.

Following and including, 4.3-RELEASE ⁶, when a minor release has been made, it becomes a “security branch”. This is meant for organisations that do not want to follow the -STABLE branch and the potential new/changed features it offers, but instead require an absolutely stable environment, only updating to implement security updates. ⁷

Each update to a security branch is called a “patchlevel”. For every security enhancement that is done, the patchlevel number is increased, making it easy for people tracking the branch to see what security enhancements they have implemented. In cases where there have been especially serious security flaws, an entire new release can be made from a security branch. An example of this is 4.6.2-RELEASE.

80.5.3 Model summary

To summarise, the development model of FreeBSD can be seen as the following tree:

The tree of the FreeBSD development with ongoing development efforts and continuous integration.

The tree symbolises the release versions with major versions spawning new main branches and minor versions being versions of the main branch. The top branch is the -CURRENT branch where all new development is integrated, and the -STABLE branch is the branch directly below it.

Clouds of development efforts hang over the project where developers use the development models they see fit. The product of their work is then integrated into -CURRENT where it undergoes parallel debugging and is finally merged from -CURRENT into -STABLE. Security fixes are merged from -STABLE to the security branches.

⁶ The first release this actually happened for was 4.5-RELEASE, but security branches were at the same time created for 4.3-RELEASE and 4.4-RELEASE.

⁷ There is a terminology overlap with respect to the word “stable”, which leads to some confusion. The -STABLE branch is still a development branch, whose goal is to be useful for most people. If it is never acceptable for a system to get changes that are not announced at the time it is deployed, that system should run a security branch.

80.6 Hats

Many committers have a special area of responsibility. These roles are called hats. These hats can be either project roles, such as public relations officer, or maintainer for a certain area of the code. Because this is a project where people give voluntarily of their spare time, people with assigned hats are not always available. They must therefore appoint a deputy that can perform the hat's role in his or her absence. The other option is to have the role held by a group.

Many of these hats are not formalised. Formalised hats have a charter stating the exact purpose of the hat along with its privileges and responsibilities. The writing of such charters is a new part of the project, and has thus yet to be completed for all hats. These hat descriptions are not such a formalisation, rather a summary of the role with links to the charter where available and contact addresses.

80.6.1 General Hats

Contributor

A Contributor contributes to the FreeBSD project either as a developer, as an author, by sending problem reports, or in other ways contributing to the progress of the project. A contributor has no special privileges in the FreeBSD project. ?

Committer

A person who has the required privileges to add his code or documentation to the repository. A committer has made a commit within the past 12 months. ? An active committer is a committer who has made an average of one commit per month during that time.

It is worth noting that there are no technical barriers to prevent someone, once having gained commit privileges to the main- or a sub-project, to make commits in parts of that project's source the committer did not specifically get permission to modify. However, when wanting to make modifications to parts a committer has not been involved in before, he/she should read the logs to see what has happened in this area before, and also read the MAINTAINER file to see if the maintainer of this part has any special requests on how changes in the code should be made

Core Team

The core team is elected by the committers from the pool of committers and serves as the board of directors of the FreeBSD project. It promotes active contributors to committers, assigns people to well-defined hats, and is the final arbiter of decisions involving which way the project should be heading. As by July 1st, 2004, core consisted of 9 members. Elections are held every two years.

Maintainership

Maintainership means that that person is responsible for what is allowed to go into that area of the code and has the final say should disagreements over the code occur. This involves proactive work aimed at stimulating contributions and reactive work in reviewing commits.

With the FreeBSD source comes the MAINTAINERS file that contains a one-line summary of how each maintainer would like contributions to be made. Having this notice and contact information enables developers to focus on the development effort rather than being stuck in a slow correspondence should the maintainer be unavailable for some time.

If the maintainer is unavailable for an unreasonably long period of time, and other people do a significant amount of work, maintainership may be switched without the maintainer's approval. This is based on the stance that maintainership should be demonstrated, not declared.

Maintainership of a particular piece of code is a hat that is not held as a group.

80.6.2 Official Hats

The official hats in the FreeBSD Project are hats that are more or less formalised and mainly administrative roles. They have the authority and responsibility for their area. The following illustration shows the responsibility lines. After this follows a description of each hat, including who it is held by.

All boxes consist of groups of committers, except for the dotted boxes where the holders are not necessarily committers. The flattened circles are sub-projects and consist of both committers and non-committers of the main project.

Documentation project manager

? architect is responsible for defining and following up documentation goals for the committers in the Documentation project.

Hat held by: The DocEng team doceng@FreeBSD.org. The [DocEng Charter](#).

Postmaster

The Postmaster is responsible for mail being correctly delivered to the committers' email address. He is also responsible for ensuring that the mailing lists work and should take measures against possible disruptions of mail such as having troll-, spam- and virus-filters.

Hat currently held by: the Postmaster Team postmaster@FreeBSD.org.

Release Coordination

The responsibilities of the Release Engineering Team are

- Setting, publishing and following a release schedule for official releases
- Documenting and formalising release engineering procedures
- Creation and maintenance of code branches
- Coordinating with the Ports and Documentation teams to have an updated set of packages and documentation released with the new releases
- Coordinating with the Security team so that pending releases are not affected by recently disclosed vulnerabilities.

Further information about the development process is available in the ? section.

Hat held by: the Release Engineering team re@FreeBSD.org. The [Release Engineering Charter](#).

Public Relations & Corporate Liaison

The Public Relations & Corporate Liaison's responsibilities are:

- Making press statements when happenings that are important to the FreeBSD Project happen.
- Being the official contact person for corporations that are working close with the FreeBSD Project.

- Take steps to promote FreeBSD within both the Open Source community and the corporate world.
- Handle the “freebsd-advocacy” mailing list.

This hat is currently not occupied.

Security Officer

The Security Officer’s main responsibility is to coordinate information exchange with others in the security community and in the FreeBSD project. The Security Officer is also responsible for taking action when security problems are reported and promoting proactive development behaviour when it comes to security.

Because of the fear that information about vulnerabilities may leak out to people with malicious intent before a patch is available, only the Security Officer, consisting of an officer, a deputy and two ? members, receive sensitive information about security issues. However, to create or implement a patch, the Security Officer has the Security Officer Team security-team@FreeBSD.org to help do the work.

Source Repository Manager

The Source Repository Manager is the only one who is allowed to directly modify the repository without using the ? tool. It is his/her responsibility to ensure that technical problems that arise in the repository are resolved quickly. The source repository manager has the authority to back out commits if this is necessary to resolve a SVN technical problem.

Hat held by: the Source Repository Manager clusteradm@FreeBSD.org.

Election Manager

The Election Manager is responsible for the ? process. The manager is responsible for running and maintaining the election system, and is the final authority should minor unforeseen events happen in the election process. Major unforeseen events have to be discussed with the ?

Hat held only during elections.

Web site Management

The Web site Management hat is responsible for coordinating the rollout of updated web pages on mirrors around the world, for the overall structure of the primary web site and the system it is running upon. The management needs to coordinate the content with ? and acts as maintainer for the “www” tree.

Hat held by: the FreeBSD Webmasters www@FreeBSD.org.

Ports Manager

The Ports Manager acts as a liaison between ? and the core project, and all requests from the project should go to the ports manager.

Hat held by: the Ports Management Team portmgr@FreeBSD.org. The [Portmgr charter](#).

Standards

The Standards hat is responsible for ensuring that FreeBSD complies with the standards it is committed to , keeping up to date on the development of these standards and notifying FreeBSD developers of important changes that allows them to take a proactive role and decrease the time between a standards update and FreeBSD's compliancy.

Hat currently held by: Garrett Wollman wollman@FreeBSD.org.

Core Secretary

The Core Secretary's main responsibility is to write drafts to and publish the final Core Reports. The secretary also keeps the core agenda, thus ensuring that no balls are dropped unresolved.

Hat currently held by: A.MATTHEW.EMAIL.

Bugmeister

The Bugmeister is responsible for ensuring that the maintenance database is in working order, that the entries are correctly categorised and that there are no invalid entries.

Hat currently held by: the Bugmeister Team bugmeister@FreeBSD.org.

Donations Liaison Officer

The task of the donations liaison officer is to match the developers with needs with people or organisations willing to make a donation. The Donations Liaison Charter is available [here](#)

Hat held by: the Donations Liaison Office donations@FreeBSD.org.

Admin

(Also called "FreeBSD Cluster Admin")

The admin team consists of the people responsible for administrating the computers that the project relies on for its distributed work and communication to be synchronised. It consists mainly of those people who have physical access to the servers.

Hat held by: the Admin team admin@FreeBSD.org.

80.6.3 Process dependent hats

Report originator

The person originally responsible for filing a Problem Report.

Bugbuster

A person who will either find the right person to solve the problem, or close the PR if it is a duplicate or otherwise not an interesting one.

Mentor

A mentor is a committer who takes it upon him/her to introduce a new committer to the project, both in terms of ensuring the new committers setup is valid, that the new committer knows the available tools required in his/her work and that the new committer knows what is expected of him/her in terms of behaviour.

Vendor

The person(s) or organisation whom external code comes from and whom patches are sent to.

Reviewers

People on the mailing list where the request for review is posted.

80.7 Processes

The following section will describe the defined project processes. Issues that are not handled by these processes happen on an ad-hoc basis based on what has been customary to do in similar cases.

80.7.1 Adding new and removing old committers

The Core team has the responsibility of giving and removing commit privileges to contributors. This can only be done through a vote on the core mailing list. The ports and documentation sub-projects can give commit privileges to people working on these projects, but have to date not removed such privileges.

Normally a contributor is recommended to core by a committer. For contributors or outsiders to contact core asking to be a committer is not well thought of and is usually rejected.

If the area of particular interest for the developer potentially overlaps with other committers' area of maintainership, the opinion of those maintainers is sought. However, it is frequently this committer that recommends the developer.

When a contributor is given committer status, he is assigned a mentor. The committer who recommended the new committer will, in the general case, take it upon himself to be the new committers mentor.

When a contributor is given his commit bit, a ?-signed email is sent from either ?, ? or nik@freebsd.org to both admins@freebsd.org, the assigned mentor, the new committer and core confirming the approval of a new account. The mentor then gathers a password line, ? public key and PGP key from the new committer and sends them to ?. When the new account is created, the mentor activates the commit bit and guides the new committer through the rest of the initial process.

When a contributor sends a piece of code, the receiving committer may choose to recommend that the contributor is given commit privileges. If he recommends this to core, they will vote on this recommendation. If they vote in favour, a mentor is assigned the new committer and the new committer has to email his details to the administrators for an account to be created. After this, the new committer is all set to make his first commit. By tradition, this is by adding his name to the committers list.

Recall that a committer is considered to be someone who has committed code during the past 12 months. However, it is not until after 18 months of inactivity have passed that commit privileges are eligible to be revoked. ? There are, however, no automatic procedures for doing this. For reactions concerning commit privileges not triggered by time, see *section 1.5.8*.

When Core decides to clean up the committers list, they check who has not made a commit for the past 18 months. Committers who have not done so have their commit bits revoked.

It is also possible for committers to request that their commit bit be retired if for some reason they are no longer going to be actively committing to the project. In this case, it can also be restored at a later time by core, should the committer ask.

Roles in this process:

1. ?
 2. ?
 3. ?
 4. ?
 5. ?
- ???

80.7.2 Committing code

The committing of new or modified code is one of the most frequent processes in the FreeBSD project and will usually happen many times a day. Committing of code can only be done by a “committer”. Committers commit either code written by themselves, code submitted to them or code submitted through a *problem report*.

When code is written by the developer that is non-trivial, he should seek a code review from the community. This is done by sending mail to the relevant list asking for review. Before submitting the code for review, he should ensure it compiles correctly with the entire tree and that all relevant tests run. This is called “pre-commit test”. When contributed code is received, it should be reviewed by the committer and tested the same way.

When a change is committed to a part of the source that has been contributed from an outside ?, the maintainer should ensure that the patch is contributed back to the vendor. This is in line with the open source philosophy and makes it easier to stay in sync with outside projects as the patches do not have to be reapplied every time a new release is made.

After the code has been available for review and no further changes are necessary, the code is committed into the development branch, -CURRENT. If the change applies for the -STABLE branch or the other branches as well, a “Merge From Current” (“MFC”) countdown is set by the committer. After the number of days the committer chose when setting the MFC have passed, an email will automatically be sent to the committer reminding him to commit it to the -STABLE branch (and possibly security branches as well). Only security critical changes should be merged to security branches.

Delaying the commit to -STABLE and other branches allows for “parallel debugging” where the committed code is tested on a wide range of configurations. This makes changes to -STABLE to contain fewer faults and thus giving the branch its name.

When a committer has written a piece of code and wants to commit it, he first needs to determine if it is trivial enough to go in without prior review or if it should first be reviewed by the developer community. If the code is trivial or has been reviewed and the committer is not the maintainer, he should consult the maintainer before proceeding. If the code is contributed by an outside vendor, the maintainer should create a patch that is sent back to the vendor. The code is then committed and the deployed by the users. Should they find problems with the code, this will be reported and the committer can go back to writing a patch. If a vendor is affected, he can choose to implement or ignore the patch.

The difference when a contributor makes a code contribution is that he submits the code through the Bugzilla interface. This report is picked up by the maintainer who reviews the code and commits it.

Hats included in this process are:

1. ?
2. ?
3. ?

4. ?

? ?

80.7.3 Core election

Core elections are held at least every two years.⁸ Nine core members are elected. New elections are held if the number of core members drops below seven. New elections can also be held should at least 1/3 of the active committers demand this.

When an election is to take place, core announces this at least 6 weeks in advance, and appoints an election manager to run the elections.

Only committers can be elected into core. The candidates need to submit their candidacy at least one week before the election starts, but can refine their statements until the voting starts. They are presented in the [candidates list](#). When writing their election statements, the candidates must answer a few standard questions submitted by the election manager.

During elections, the rule that a committer must have committed during the 12 past months is followed strictly. Only these committers are eligible to vote.

When voting, the committer may vote once in support of up to nine nominees. The voting is done over a period of four weeks with reminders being posted on “developers” mailing list that is available to all committers.

The election results are released one week after the election ends, and the new core team takes office one week after the results have been posted.

Should there be a voting tie, this will be resolved by the new, unambiguously elected core members.

Votes and candidate statements are archived, but the archives are not publicly available.

Core announces the election and selects an election manager. He prepares the elections, and when ready, candidates can announce their candidacies through submitting their statements. The committers then vote. After the vote is over, the election results are announced and the new core team takes office.

Hats in core elections are:

- ?
- ?
- ?

? ? ?

80.7.4 Development of new features

Within the project there are sub-projects that are working on new features. These projects are generally done by one person ?. Every project is free to organise development as it sees fit. However, when the project is merged to the -CURRENT branch it must follow the project guidelines. When the code has been well tested in the -CURRENT branch and deemed stable enough and relevant to the -STABLE branch, it is merged to the -STABLE branch.

The requirements of the project are given by developer wishes, requests from the community in terms of direct requests by mail, Problem Reports, commercial funding for the development of features, or contributions by the scientific community. The wishes that come within the responsibility of a developer are given to that developer who prioritises his time between the request and his wishes. A common way to do this is maintain a TODO-list maintained by the project. Items that do not come within someone’s responsibility are collected on TODO-lists unless someone volunteers to take the responsibility. All requests, their distribution and follow-up are handled by the ? tool.

⁸ The first Core election was held September 2000

Requirements analysis happens in two ways. The requests that come in are discussed on mailing lists, both within the main project and in the sub-project that the request belongs to or is spawned by the request. Furthermore, individual developers on the sub-project will evaluate the feasibility of the requests and determine the prioritisation between them. Other than archives of the discussions that have taken place, no outcome is created by this phase that is merged into the main project.

As the requests are prioritised by the individual developers on the basis of doing what they find interesting, necessary or are funded to do, there is no overall strategy or prioritisation of what requests to regard as requirements and following up their correct implementation. However, most developers have some shared vision of what issues are more important, and they can ask for guidelines from the release engineering team.

The verification phase of the project is two-fold. Before committing code to the current-branch, developers request their code to be reviewed by their peers. This review is for the most part done by functional testing, but also code review is important. When the code is committed to the branch, a broader functional testing will happen, that may trigger further code review and debugging should the code not behave as expected. This second verification form may be regarded as structural verification. Although the sub-projects themselves may write formal tests such as unit tests, these are usually not collected by the main project and are usually removed before the code is committed to the current branch.⁹

80.7.5 Maintenance

It is an advantage to the project to for each area of the source have at least one person that knows this area well. Some parts of the code have designated maintainers. Others have de-facto maintainers, and some parts of the system do not have maintainers. The maintainer is usually a person from the sub-project that wrote and integrated the code, or someone who has ported it from the platform it was written for.¹⁰ The maintainer's job is to make sure the code is in sync with the project the code comes from if it is contributed code, and apply patches submitted by the community or write fixes to issues that are discovered.

The main bulk of work that is put into the FreeBSD project is maintenance. ? has made a figure showing the life cycle of changes.

Here “development release” refers to the -CURRENT branch while “production release” refers to the -STABLE branch. The “pre-commit test” is the functional testing by peer developers when asked to do so or trying out the code to determine the status of the sub-project. “Parallel debugging” is the functional testing that can trigger more review, and debugging when the code is included in the -CURRENT branch.

As of this writing, there were 269 committers in the project. When they commit a change to a branch, that constitutes a new release. It is very common for users in the community to track a particular branch. The immediate existence of a new release makes the changes widely available right away and allows for rapid feedback from the community. This also gives the community the response time they expect on issues that are of importance to them. This makes the community more engaged, and thus allows for more and better feedback that again spurs more maintenance and ultimately should create a better product.

Before making changes to code in parts of the tree that has a history unknown to the committer, the committer is required to read the commit logs to see why certain features are implemented the way they are in order not to make mistakes that have previously either been thought through or resolved.

80.7.6 Problem reporting

Before OS 10, OS included a problem reporting tool called `send-pr`. Problems include bug reports, feature requests, feature enhancements and notices of new versions of external software that are included in the project. Although `send-pr` is available, users and developers are encouraged to submit issues using our [problem report form](#).

⁹ More and more tests are however performed when building the system (“make world”). These tests are however a very new addition and no systematic framework for these tests have yet been created.

¹⁰ `sendmail` and `named` are examples of code that has been merged from other platforms.

Problem reports are sent to an email address where it is inserted into the Problem Reports maintenance database. A ? classifies the problem and sends it to the correct group or maintainer within the project. After someone has taken responsibility for the report, the report is being analysed. This analysis includes verifying the problem and thinking out a solution for the problem. Often feedback is required from the report originator or even from the FreeBSD community. Once a patch for the problem is made, the originator may be asked to try it out. Finally, the working patch is integrated into the project, and documented if applicable. It then goes through the regular maintenance cycle as described in section ?. These are the states a problem report can be in: open, analyzed, feedback, patched, suspended and closed. The suspended state is for when further progress is not possible due to the lack of information or for when the task would require so much work that nobody is working on it at the moment.

A problem is reported by the report originator. It is then classified by a bugbuster and handed to the correct maintainer. He verifies the problem and discusses the problem with the originator until he has enough information to create a working patch. This patch is then committed and the problem report is closed.

The roles included in this process are:

1. ?
 2. ?
 3. ?
- ?. ?

80.7.7 Reacting to misbehaviour

? has a number of rules that committers should follow. However, it happens that these rules are broken. The following rules exist in order to be able to react to misbehaviour. They specify what actions will result in how long a suspension the committer's commit privileges.

- Committing during code freezes without the approval of the Release Engineering team - 2 days
- Committing to a security branch without approval - 2 days
- Commit wars - 5 days to all participating parties
- Impolite or inappropriate behaviour - 5 days

?

For the suspensions to be efficient, any single core member can implement a suspension before discussing it on the “core” mailing list. Repeat offenders can, with a 2/3 vote by core, receive harsher penalties, including permanent removal of commit privileges. (However, the latter is always viewed as a last resort, due to its inherent tendency to create controversy). All suspensions are posted to the “developers” mailing list, a list available to committers only.

It is important that you cannot be suspended for making technical errors. All penalties come from breaking social etiquette.

Hats involved in this process:

- ?
- ?

80.7.8 Release engineering

The FreeBSD project has a Release Engineering team with a principal release engineer that is responsible for creating releases of FreeBSD that can be brought out to the user community via the net or sold in retail outlets. Since FreeBSD is available on multiple platforms and releases for the different architectures are made available at the same time, the team has one person in charge of each architecture. Also, there are roles in the team responsible for coordinating

quality assurance efforts, building a package set and for having an updated set of documents. When referring to the release engineer, a representative for the release engineering team is meant.

When a release is coming, the FreeBSD project changes shape somewhat. A release schedule is made containing feature- and code-freezes, release of interim releases and the final release. A feature-freeze means no new features are allowed to be committed to the branch without the release engineers' explicit consent. Code-freeze means no changes to the code (like bugs-fixes) are allowed to be committed without the release engineers explicit consent. This feature- and code-freeze is known as stabilising. During the release process, the release engineer has the full authority to revert to older versions of code and thus "back out" changes should he find that the changes are not suitable to be included in the release.

There are three different kinds of releases:

1. .0 releases are the first release of a major version. These are branched of the -CURRENT branch and have a significantly longer release engineering cycle due to the unstable nature of the -CURRENT branch
2. .X releases are releases of the -STABLE branch. They are scheduled to come out every 4 months.
3. .X.Y releases are security releases that follow the .X branch. These come out only when sufficient security fixes have been merged since the last release on that branch. New features are rarely included, and the security team is far more involved in these than in regular releases.

For releases of the -STABLE-branch, the release process starts 45 days before the anticipated release date. During the first phase, the first 15 days, the developers merge what changes they have had in -CURRENT that they want to have in the release to the release branch. When this period is over, the code enters a 15 day code freeze in which only bug fixes, documentation updates, security-related fixes and minor device driver changes are allowed. These changes must be approved by the release engineer in advance. At the beginning of the last 15 day period a release candidate is created for widespread testing. Updates are less likely to be allowed during this period, except for important bug fixes and security updates. In this final period, all releases are considered release candidates. At the end of the release process, a release is created with the new version number, including binary distributions on web sites and the creation of a CD-ROM images. However, the release is not considered "really released" until a ?-signed message stating exactly that, is sent to the mailing list freebsd-announce; anything labelled as a "release" before that may well be in-process and subject to change before the PGP-signed message is sent. ¹¹.

The releases of the -CURRENT-branch (that is, all releases that end with ".0") are very similar, but with twice as long timeframe. It starts 8 weeks prior to the release with announcement of the release time line. Two weeks into the release process, the feature freeze is initiated and performance tweaks should be kept to a minimum. Four weeks prior to the release, an official beta version is made available. Two weeks prior to release, the code is officially branched into a new version. This version is given release candidate status, and as with the release engineering of -STABLE, the code freeze of the release candidate is hardened. However, development on the main development branch can continue. Other than these differences, the release engineering processes are alike.

.0 releases go into their own branch and are aimed mainly at early adopters. The branch then goes through a period of stabilisation, and it is not until the ? decides the demands to stability have been satisfied that the branch becomes -STABLE and -CURRENT targets the next major version. While this for the majority has been with .1 versions, this is not a demand.

Most releases are made when a given date that has been deemed a long enough time since the previous release comes. A target is set for having major releases every 18 months and minor releases every 4 months. The user community has made it very clear that security and stability cannot be sacrificed by self-imposed deadlines and target release dates. For slips of time not to become too long with regards to security and stability issues, extra discipline is required when committing changes to -STABLE.

These are the stages in the release engineering process. Multiple release candidates may be created until the release is deemed stable enough to be released.

?

¹¹ Many commercial vendors use these images to create CD-ROMs that are sold in retail outlets.

80.8 Tools

The major support tools for supporting the development process are Perforce, Bugzilla, Mailman, and OpenSSH. These are externally developed tools and are commonly used in the open source world.

80.8.1 Subversion (SVN)

Subversion (“SVN”) is a system to handle multiple versions of text files and tracking who committed what changes and why. A project lives within a “repository” and different versions are considered different “branches”.

80.8.2 Bugzilla

Bugzilla is a maintenance database consisting of a set of tools to track bugs at a central site. It supports the bug tracking process for sending and handling bugs as well as querying and updating the database and editing bug reports. The project uses its web interface to send “Problem Reports” to the projects central Bugzilla server. The committers also have web and command-line clients.

80.8.3 Mailman

Mailman is a program that automates the management of mailing lists. The FreeBSD Project uses it to run 16 general lists, 60 technical lists, 4 limited lists and 5 lists with CVS commit logs. It is also used for many mailing lists set up and used by other people and projects in the FreeBSD community. General lists are lists for the general public, technical lists are mainly for the development of specific areas of interest, and closed lists are for internal communication not intended for the general public. The majority of all the communication in the project goes through these 85 lists ?, Appendix C.

80.8.4 Perforce

Perforce is a commercial software configuration management system developed by Perforce Systems that is available on over 50 operating systems. It is a collection of clients built around the Perforce server that contains the central file repository and tracks the operations done upon it. The clients are both clients for accessing the repository and administration of its configuration.

80.8.5 Pretty Good Privacy

Pretty Good Privacy, better known as PGP, is a cryptosystem using a public key architecture to allow people to digitally sign and/or encrypt information in order to ensure secure communication between two parties. A signature is used when sending information out many recipients, enabling them to verify that the information has not been tampered with before they received it. In the FreeBSD Project this is the primary means of ensuring that information has been written by the person who claims to have written it, and not altered in transit.

80.8.6 Secure Shell

Secure Shell is a standard for securely logging into a remote system and for executing commands on the remote system. It allows other connections, called tunnels, to be established and protected between the two involved systems. This standard exists in two primary versions, and only version two is used for the FreeBSD Project. The most common implementation of the standard is OpenSSH that is a part of the project’s main distribution. Since its source is updated more often than FreeBSD releases, the latest version is also available in the ports tree.

80.9 Sub-projects

Sub-projects are formed to reduce the amount of communication needed to coordinate the group of developers. When a problem area is sufficiently isolated, most communication would be within the group focusing on the problem, requiring less communication with the groups they communicate with than were the group not isolated.

80.9.1 The Ports Subproject

A “port” is a set of meta-data and patches that are needed to fetch, compile and install correctly an external piece of software on a FreeBSD system. The amount of ports have grown at a tremendous rate, as shown by the following figure.

? is taken from [the FreeBSD web site](#). It shows the number of ports available to FreeBSD in the period 1995 to 2005. It looks like the curve has first grown exponentially, and then since the middle of 2001 grown linearly.

As the external software described by the port often is under continued development, the amount of work required to maintain the ports is already large, and increasing. This has led to the ports part of the FreeBSD project gaining a more empowered structure, and is more and more becoming a sub-project of the FreeBSD project.

Ports has its own core team with the ? as its leader, and this team can appoint committers without FreeBSD Core’s approval. Unlike in the FreeBSD Project, where a lot of maintenance frequently is rewarded with a commit bit, the ports sub-project contains many active maintainers that are not committers.

Unlike the main project, the ports tree is not branched. Every release of FreeBSD follows the current ports collection and has thus available updated information on where to find programs and how to build them. This, however, means that a port that makes dependencies on the system may need to have variations depending on what version of FreeBSD it runs on.

With an unbranched ports repository it is not possible to guarantee that any port will run on anything other than -CURRENT and -STABLE, in particular older, minor releases. There is neither the infrastructure nor volunteer time needed to guarantee this.

For efficiency of communication, teams depending on Ports, such as the release engineering team, have their own ports liaisons.

80.9.2 The FreeBSD Documentation Project

The FreeBSD Documentation project was started January 1995. From the initial group of a project leader, four team leaders and 16 members, they are now a total of 44 committers. The documentation mailing list has just under 300 members, indicating that there is quite a large community around it.

The goal of the Documentation project is to provide good and useful documentation of the FreeBSD project, thus making it easier for new users to get familiar with the system and detailing advanced features for the users.

The main tasks in the Documentation project are to work on current projects in the “FreeBSD Documentation Set”, and translate the documentation to other languages.

Like the FreeBSD Project, documentation is split in the same branches. This is done so that there is always an updated version of the documentation for each version. Only documentation errors are corrected in the security branches.

Like the ports sub-project, the Documentation project can appoint documentation committers without FreeBSD Core’s approval. ?.

The Documentation project has a primer. This is used both to introduce new project members to the standard tools and syntaxes and acts as a reference when working on the project.

80.10 References

- Frederick P. Brooks 1975/1995 Pearson Education Limited 0201835959 Addison-Wesley Pub Co The Mythical Man-Month Essays on Software Engineering, Anniversary Edition (2nd Edition)
- Niklas Saers 2003 A project model for the FreeBSD Project Candidatus Scientiarum thesis <http://niklas.saers.com/thesis>
- Niels Jørgensen 2001 Putting it All in the Trunk Incremental Software Development in the FreeBSD Open Source Project <http://www.dat.ruc.dk/~nielsj/research/papers/freebsd.pdf>
- Project Management Institute 1996/2000 Project Management Institute 1-880410-23-0 Project Management Institute Newtown Square Pennsylvania USA PMBOK Guide A Guide to the Project Management Body of Knowledge, 2000 Edition
- 2002 The FreeBSD Project Core Bylaws <http://www.freebsd.org/internal/bylaws.html>
- 2002 The FreeBSD Documentation Project FreeBSD Developer's Handbook http://www.freebsd.org/doc/en_US.ISO8859-1/books/developers-handbook/
- 2002 The FreeBSD Project Core team election 2002 <http://election.uk.freebsd.org/candidates.html>
- Dag-Erling Smørgrav Hiten Pandya 2002 The FreeBSD Documentation Project The FreeBSD Documentation Project Problem Report Handling Guidelines <http://www.freebsd.org/doc/en/articles/pr-guidelines/article.html>
- Dag-Erling Smørgrav 2002 The FreeBSD Documentation Project The FreeBSD Documentation Project Writing FreeBSD Problem Reports <http://www.freebsd.org/doc/en/articles/problem-reports/article.html>
- 2001 The FreeBSD Documentation Project The FreeBSD Documentation Project Committers Guide <http://www.freebsd.org/doc/en/articles/committers-guide/article.html>
- Murray Stokely 2002 The FreeBSD Documentation Project The FreeBSD Documentation Project FreeBSD Release Engineering http://www.freebsd.org/doc/en_US.ISO8859-1/articles/releeng/article.html
- The FreeBSD Documentation Project FreeBSD Handbook http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook
- 2002 The FreeBSD Documentation Project The FreeBSD Documentation Project Contributors to FreeBSD http://www.freebsd.org/doc/en_US.ISO8859-1/articles/contributors/article.html
- 2002 The FreeBSD Project The FreeBSD Project Core team elections 2002 <http://election.uk.freebsd.org>
- 2002 The FreeBSD Project The FreeBSD Project Commit Bit Expiration Policy 2002/04/06 15:35:30 <http://www.freebsd.org/internal/expire-bits.html>
- 2002 The FreeBSD Project The FreeBSD Project New Account Creation Procedure 2002/08/19 17:11:27 <http://www.freebsd.org/internal/new-account.html>
- 2002 The FreeBSD Documentation Project The FreeBSD Documentation Project FreeBSD DocEng Team Charter 2003/03/16 12:17 <http://www.freebsd.org/internal/doceng.html>
- Greg Lehey 2002 Greg Lehey Greg Lehey Two years in the trenches The evolution of a software project <http://www.lemis.com/grog/In-the-trenches.pdf>

FreeBSD Developers' Handbook

Author The FreeBSD Documentation Project

CHAP.INTRODUCTION CHAP.TOOLS CHAP.SECURE CHAP.L10N CHAP.POLICIES CHAP.TESTING
CHAP.SOCKETS CHAP.IPV6 CHAP.KERNELBUILD CHAP.KERNELDEBUG CHAP.X86

DaveAPatterson JohnLHennessy 1998Morgan Kaufmann Publishers, Inc. 1-55860-428-6 Morgan Kaufmann Publishers, Inc. Computer Organization and Design The Hardware / Software Interface 1-2

W.RichardStevens 1993Addison Wesley Longman, Inc. 0-201-56317-7 Addison Wesley Longman, Inc. Advanced Programming in the Unix Environment 1-2

MarshallKirkMcKusick GeorgeNeville-Neil 2004Addison-Wesley 0-201-70245-2 Addison-Wesley The Design and Implementation of the FreeBSD Operating System 1-2

AlephOne Phrack 49; "Smashing the Stack for Fun and Profit"

ChrispinCowan CaltonPu DaveMaier StackGuard; Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks

ToddMiller Theode Raadt strlcpy and strlcat – consistent, safe string copy and concatenation.

CHAP.INDEX

Installation

Author NikClayton

82.1 Introduction

Q: What is OS?

A: OS is a modern operating system for desktops, laptops, servers, and embedded systems with support for a large number of [platforms](#).

It is based on U.C. Berkeley's "4.4BSD-Lite" release, with some "4.4BSD-Lite2" enhancements. It is also based indirectly on William Jolitz's port of U.C. Berkeley's "Net/2" to the I386, known as "386BSD", though very little of the 386BSD code remains.

OS is used by companies, Internet Service Providers, researchers, computer professionals, students and home users all over the world in their work, education and recreation.

For more detailed information on OS, refer to the OS Handbook.

Q: What is the goal of the OS Project?

A: The goal of the OS Project is to provide a stable and fast general purpose operating system that may be used for any purpose without strings attached.

Q: Does the OS license have any restrictions?

A: Yes. Those restrictions do not control how the code is used, but how to treat the OS Project itself. The license itself is available at [license](#) and can be summarized like this:

- Do not claim that you wrote this.
- Do not sue us if it breaks.
- Do not remove or modify the license.

Many of us have a significant investment in the project and would certainly not mind a little financial compensation now and then, but we definitely do not insist on it. We believe that our first and foremost "mission" is to provide code to any and all comers, and for whatever purpose, so that the code gets the widest possible use and provides the widest possible benefit. This, we believe, is one of the most fundamental goals of Free Software and one that we enthusiastically support.

Code in our source tree which falls under the [GNU General Public License \(GPL\)](#) or [GNU Library General Public License \(LGPL\)](#) comes with slightly more strings attached, though at least on the side of enforced access rather than the usual opposite. Due to the additional complexities that can evolve in the commercial use of GPL software, we do, however, endeavor to replace such software with submissions under the more relaxed [OS license](#) whenever possible.

Q: Can OS replace my current operating system?

A: For most people, yes. But this question is not quite that cut-and-dried.

Most people do not actually use an operating system. They use applications. The applications are what really use the operating system. OS is designed to provide a robust and full-featured environment for applications. It supports a wide variety of web browsers, office suites, email readers, graphics programs, programming environments, network servers, and much more. Most of these applications can be managed through the [Ports Collection](#).

If an application is only available on one operating system, that operating system cannot just be replaced. Chances are, there is a very similar application on OS, however. As a solid office or Internet server or a reliable workstation, OS will almost certainly do everything you need. Many computer users across the world, including both novices and experienced UNIX administrators, use OS as their only desktop operating system.

Users migrating to OS from another UNIX-like environment will find OS to be similar. WINDOWS and MACOS users may be interested in instead using [PC-BSD](#), a OS-based desktop distribution. Non-UNIX users should expect to invest some additional time learning the UNIX way of doing things. This FAQ and the OS Handbook are excellent places to start.

Q: Why is it called OS?

- It may be used free of charge, even by commercial users.
- Full source for the operating system is freely available, and the minimum possible restrictions have been placed upon its use, distribution and incorporation into other work (commercial or non-commercial).
- Anyone who has an improvement or bug fix is free to submit their code and have it added to the source tree (subject to one or two obvious provisions).

It is worth pointing out that the word “free” is being used in two ways here: one meaning “at no cost” and the other meaning “do whatever you like”. Apart from one or two things you *cannot* do with the OS code, for example pretending you wrote it, you can really do whatever you like with it.

Q: What are the differences between OS and NetBSD, OpenBSD, and other open source BSD operating systems?

A: James Howard wrote a good explanation of the history and differences between the various projects, called [The BSD Family Tree](#) which goes a fair way to answering this question. Some of the information is out of date, but the history portion in particular remains accurate.

Most of the BSDs share patches and code, even today. All of the BSDs have common ancestry.

The design goals of OS are described in ?, above. The design goals of the other most popular BSDs may be summarized as follows:

- OpenBSD aims for operating system security above all else. The OpenBSD team wrote MAN.SSH.1 and MAN.PF.4, which have both been ported to OS.
- NetBSD aims to be easily ported to other hardware platforms.
- DragonFly BSD is a fork of OS 4.8 that has since developed many interesting features of its own, including the HAMMER file system and support for user-mode “vkernels”.

Q: What is the latest version of OS?

A: At any point in the development of OS, there can be multiple parallel branches. REL.RELX releases are made from the REL.STABLE branch, and REL2.RELX releases are made from the REL2.STABLE branch.

Up until the release of 9.0, the REL2.RELX series was the one known as *-STABLE*. However, as of REL.HEAD.RELX, the REL2.RELX branch will be designated for an “extended support” status and receive only fixes for major problems, such as security-related fixes.

Version [REL.CURRENT](#) is the latest release from the REL.STABLE branch; it was released in REL.CURRENT.DATE. Version [REL2.CURRENT](#) is the latest release from the REL2.STABLE branch; it was released in REL2.CURRENT.DATE.

Briefly, *-STABLE* is aimed at the ISP, corporate user, or any user who wants stability and a minimal number of changes compared to the new (and possibly unstable) features of the latest *-CURRENT* snapshot. Releases can come from either branch, but *-CURRENT* is meant for users who are prepared for its increased volatility, relative to *-STABLE*.

Releases are made *every few months*. While many people stay more up-to-date with the OS sources (see the questions on *OS.CURRENT* and *OS.STABLE*) than that, doing so is more of a commitment, as the sources are a moving target.

More information on OS releases can be found on the [Release Engineering page](#) and in MAN.RELEASE.7.

Q: What is *OS-CURRENT*?

A: *OS.CURRENT* is the development version of the operating system, which will in due course become the new *OS.STABLE* branch. As such, it is really only of interest to developers working on the system and die-hard hobbyists. See the relevant section in the Handbook for details on running *-CURRENT*.

Users not familiar with OS should not use *OS.CURRENT*. This branch sometimes evolves quite quickly and due to mistake can be un-buildable at times. People that use *OS.CURRENT* are expected to be able to analyze, debug, and report problems.

OS snapshot releases are made based on the current state of the *-CURRENT* and *-STABLE* branches. The goals behind each snapshot release are:

- To test the latest version of the installation software.
- To give people who would like to run *-CURRENT* or *-STABLE* but who do not have the time or bandwidth to follow it on a day-to-day basis an easy way of bootstrapping it onto their systems.
- To preserve a fixed reference point for the code in question, just in case we break something really badly later. (Although Subversion normally prevents anything horrible like this happening.)
- To ensure that all new features and fixes in need of testing have the greatest possible number of potential testers.

No claims are made that any *-CURRENT* snapshot can be considered “production quality” for any purpose. If a stable and fully tested system is needed, stick to full releases or use the *-STABLE* snapshots.

Snapshot releases are directly available from snapshot.

Official snapshots are generated on a regular basis for all actively developed branches.

Q: What is the *OS-STABLE* concept?

A: Back when OS 2.0.5 was released, OS development branched in two. One branch was named *-STABLE*, one *-CURRENT*. *OS-STABLE* is intended for Internet Service Providers and other commercial enterprises for whom sudden shifts or experimental features are quite undesirable. It receives only well-tested bug fixes and other small incremental enhancements. *OS-CURRENT*, on the other hand, has been one unbroken line since 2.0 was released, leading towards REL.CURRENT-RELEASE and beyond. For more detailed information on branches see “OS Release Engineering: Creating the Release Branch”, the status of the branches and the upcoming release schedule can be found on the [Release Engineering Information page](#).

REL.CURRENT-STABLE is the actively developed *-STABLE* branch. The latest release on the REL.CURRENT-STABLE branch is REL.CURRENT-RELEASE, which was released in REL.CURRENT.DATE.

The REL.HEAD branch is the actively developed *-CURRENT* branch toward the next generation of OS. See *What is OS-CURRENT?* for more information on this branch.

Q: When are OS releases made?

A: The A.RE releases a new major version of OS about every 18 months and a new minor version about every 8 months, on average. Release dates are announced well in advance, so that the people working on the system know when their projects need to be finished and tested. A testing period precedes each release, to ensure that the addition of new features does not compromise the stability of the release. Many users regard this caution as one of the best things about OS, even though waiting for all the latest goodies to reach *-STABLE* can be a little frustrating.

More information on the release engineering process (including a schedule of upcoming releases) can be found on the [release engineering](#) pages on the OS Web site.

For people who need or want a little more excitement, binary snapshots are made weekly as discussed above.

Q: Who is responsible for OS?

A: The key decisions concerning the OS project, such as the overall direction of the project and who is allowed to add code to the source tree, are made by a core team of 9 people. There is a much larger team of more than 350 committers who are authorized to make changes directly to the OS source tree.

However, most non-trivial changes are discussed in advance in the *mailing lists*, and there are no restrictions on who may take part in the discussion.

Q: Where can I get OS?

A: Every significant release of OS is available via anonymous FTP from the [OS FTP site](#):

- The latest REL.STABLE release, REL.CURRENT-RELEASE can be found in the [REL.CURRENT-RELEASE directory](#).
- Snapshot releases are made monthly for the *-CURRENT* and *-STABLE* branch, these being of service purely to bleeding-edge testers and developers.
- The latest REL2.STABLE release, REL2.CURRENT-RELEASE can be found in the [REL2.CURRENT-RELEASE directory](#).

Information about obtaining OS on CD, DVD, and other media can be found in the Handbook.

Q: How do I access the Problem Report database?

A: The Problem Report database of all user change requests may be queried by using our web-based PR [query](#) interface.

The web-based problem report submission interface can be used to submit problem reports through a web browser.

Before submitting a problem report, read *Writing OS Problem Reports*, an article on how to write good problem reports.

82.2 Documentation and Support

Q: What good books are there about OS?

A: The project produces a wide range of documentation, available online from this link: <http://www.FreeBSD.org/docs.html>. In addition, *the Bibliography* at the end of this FAQ, and the one in the Handbook reference other recommended books.

Q: Is the documentation available in other formats, such as plain text (ASCII), or POSTSCRIPT?

A: Yes. The documentation is available in a number of different formats and compression schemes on the OS FTP site, in the [/pub/FreeBSD/doc/](#) directory.

The documentation is categorized in a number of different ways. These include:

- The document's name, such as *faq*, or *handbook*.
- The document's language and encoding. These are based on the locale names found under [/usr/share/locale](#) on a OS system. The current languages and encodings are as follows:

Name	Meaning
en_US.ISO8859-1	English (United States)
bn_BD.ISO10646-1	Bengali or Bangla (Bangladesh)
da_DK.ISO8859-1	Danish (Denmark)
de_DE.ISO8859-1	German (Germany)
el_GR.ISO8859-7	Greek (Greece)
es_ES.ISO8859-1	Spanish (Spain)
fr_FR.ISO8859-1	French (France)
hu_HU.ISO8859-2	Hungarian (Hungary)
it_IT.ISO8859-15	Italian (Italy)
ja_JP.eucJP	Japanese (Japan, EUC encoding)
mn_MN.UTF-8	Mongolian (Mongolia, UTF-8 encoding)
nl_NL.ISO8859-1	Dutch (Netherlands)
no_NO.ISO8859-1	Norwegian (Norway)
pl_PL.ISO8859-2	Polish (Poland)
pt_BR.ISO8859-1	Portuguese (Brazil)
ru_RU.KOI8-R	Russian (Russia, KOI8-R encoding)
sr_YU.ISO8859-2	Serbian (Serbia)
tr_TR.ISO8859-9	Turkish (Turkey)
zh_CN.UTF-8	Simplified Chinese (China, UTF-8 encoding)
zh_TW.UTF-8	Traditional Chinese (Taiwan, UTF-8 encoding)

Note

Some documents may not be available in all languages.

- The document's format. We produce the documentation in a number of different output formats. Each format has its own advantages and disadvantages. Some formats are better suited for online reading, while others are meant to be aesthetically pleasing when printed on paper. Having the documentation available in any of these formats ensures that our readers will be able to read the parts they are interested in, either on their monitor, or on paper after printing the documents. The currently available formats are:

Format	Meaning
html-split	A collection of small, linked, HTML files.
html	One large HTML file containing the entire document
pdf	Adobe's Portable Document Format
ps	POSTSCRIPT
rtf	MICROSOFT's Rich Text Format
txt	Plain text

Note

Page numbers are not automatically updated when loading Rich Text Format into Word. Press Ctrl+A, Ctrl+End, F9 after loading the document, to update the page numbers.

- The compression and packaging scheme.
 1. Where the format is `html-split`, the files are bundled up using `MAN.TAR.1`. The resulting `.tar` file is then compressed using the compression schemes detailed in the next point.
 2. All the other formats generate one file. For example, `article.pdf`, `book.html`, and so on.

These files are then compressed using either the `zip` or `bz2` compression schemes. `MAN.TAR.1` can be used to uncompress these files.

So the POSTSCRIPT version of the Handbook, compressed using `bzip2` will be stored in a file called `book.ps.bz2` in the `handbook/` directory.

After choosing the format and compression mechanism, download the compressed files, uncompress them, and then copy the appropriate documents into place.

For example, the split HTML version of the FAQ, compressed using MAN.BZIP2.1, can be found in `doc/en_US.ISO8859-1/books/faq/book.html-split.tar.bz2`. To download and uncompress that file, type:

```
PROMPT.ROOT fetch ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/en_US.ISO8859-1/books/faq/book.html-split.tar.bz2
PROMPT.ROOT tar xvf book.html-split.tar.bz2
```

If the file is compressed, tar will automatically detect the appropriate format and decompress it correctly, resulting in a collection of `.html` files. The main one is called `index.html`, which will contain the table of contents, introductory material, and links to the other parts of the document.

Q: Where do I find info on the OS mailing lists? What OS news groups are available?

A: Refer to the Handbook entry on mailing-lists and the Handbook entry on newsgroups.

Q: Are there OS IRC (Internet Relay Chat) channels?

A: Yes, most major IRC networks host a OS chat channel:

- Channel `#FreeBSDhelp` on [EFNet](#) is a channel dedicated to helping OS users.
- Channel `#FreeBSD` on [Freenode](#) is a general help channel with many users at any time. The conversations have been known to run off-topic for a while, but priority is given to users with OS questions. Other users can help with the basics, referring to the Handbook whenever possible and providing links for learning more about a particular topic. This is primarily an English speaking channel, though it does have users from all over the world. Non-native English speakers should try to ask the question in English first and then relocate to `##freebsd-lang` as appropriate.
- Channel `#FreeBSD` on [DALNET](#) is available at `irc.dal.net` in the US and `irc.eu.dal.net` in Europe.
- Channel `#FreeBSD` on [UNDERNET](#) is available at `us.undernet.org` in the US and `eu.undernet.org` in Europe. Since it is a help channel, be prepared to read the documents you are referred to.
- Channel `#FreeBSD` on [RUSNET](#) is a Russian language channel dedicated to helping OS users. This is also good place for non-technical discussions.
- Channel `#bsdchat` on [Freenode](#) is a Traditional Chinese (UTF-8 encoding) language channel dedicated to helping OS users. This is also good place for non-technical discussions.

The OS wiki has a [good list](#) of IRC channels.

Each of these channels are distinct and are not connected to each other. Since their chat styles differ, try each to find one suited to your chat style.

Q: Are there any web based forums to discuss OS?

A: The official OS forums are located at <https://forums.FreeBSD.org/>.

Q: Where can I get commercial OS training and support?

A: [iXsystems, Inc.](#), parent company of the [OS Mall](#), provides commercial OS and PC-BSD software [support](#), in addition to OS development and tuning solutions.

BSD Certification Group, Inc. provides system administration certifications for DragonFly BSD, OS, NetBSD, and OpenBSD. Refer to [their site](#) for more information.

Any other organizations providing training and support should contact the Project to be listed here.

82.3 Installation

Q: Which platform should I download? I have a 64 bit capable INTEL CPU, but I only see amd64.

A: ARCH.AMD64 is the term OS uses for 64-bit compatible x86 architectures (also known as “x86-64” or “x64”). Most modern computers should use ARCH.AMD64. Older hardware should use ARCH.I386. When installing on a non-x86-compatible architecture, select the platform which best matches the hardware.

Q: Which file do I download to get OS?

A: On the [Getting OS](#) page, select [iso] next to the architecture that matches the hardware.

Any of the following can be used:

file	description
disc1.iso	Contains enough to install OS and a minimal set of packages.
dvd1.iso	Similar to disc1.iso but with additional packages.
memstick.img	A bootable image sufficient for writing to a USB stick.
bootonly.iso	A minimal image that requires network access during installation to completely install OS.

ARCH.PC98 users require these floppy images: floppies/boot.flp, floppies/kern1.flp, floppies/kern2.flp, and floppies/mfsroot1.flp. These images need to be written onto floppies by tools like MAN.DD.1.

Full instructions on this procedure and a little bit more about installation issues in general can be found in the Handbook entry on installing OS.

Q: What do I do if the install image does not boot?

A: This can be caused by not downloading the image in *binary* mode when using FTP.

Some FTP clients default their transfer mode to *ascii* and attempt to change any end-of-line characters received to match the conventions used by the client’s system. This will almost invariably corrupt the boot image. Check the SHA-256 checksum of the downloaded boot image: if it is not *exactly* that on the server, then the download process is suspect.

When using a command line FTP client, type *binary* at the FTP command prompt after getting connected to the server and before starting the download of the image.

Q: Where are the instructions for installing OS?

A: Installation instructions for versions since OS 9.0 can be found at Handbook entry on installing OS. Older instructions can be found in the legacy entry on installing OS.

Q: What are the minimum requirements to run OS?

A: OS requires a 486 or better PC, 64 MB or more of RAM, and at least 1.1 GB of hard disk space.

Q: How can I make my own custom release or install disk?

A: Customized OS installation media can be created by building a custom release. Follow the instructions in the Release Engineering article.

Q: Can WINDOWS co-exist with OS?

A: If WINDOWS is installed first, then yes. OS’s boot manager will then manage to boot WINDOWS and OS. If WINDOWS is installed afterwards, it will overwrite the boot manager. If that happens, see the next section.

Q: Another operating system destroyed my Boot Manager. How do I get it back?

A: This depends upon the boot manager. The OS boot selection menu can be reinstalled using MAN.BOOT0CFG.8. For example, to restore the boot menu onto the disk ada0:

```
PROMPT.ROOT boot0cfg -B ada0
```

The non-interactive MBR bootloader can be installed using MAN.GPART.8:

```
PROMPT.ROOT gpart bootcode -b /boot/mbr ada0
```

For more complex situations, including GPT disks, see MAN.GPART.8.

Q: I booted from a CD, but the install program says no CD-ROM is found. Where did it go?

A: The usual cause of this problem is a mis-configured CD-ROM drive. Many PCs now ship with the CD-ROM as the slave device on the secondary IDE controller, with no master device on that controller. This is illegal according to the ATAPI specification, but WINDOWS plays fast and loose with the specification, and the BIOS ignores it when booting. This is why the BIOS was able to see the CD-ROM to boot from it, but why OS cannot see it to complete the install.

Reconfigure the system so that the CD-ROM is either the master device on the IDE controller it is attached to, or make sure that it is the slave on an IDE controller that also has a master device.

Q: Do I need to install the source?

A: In general, no. There is nothing in the base system which requires the presence of the source to operate. Some ports, like sysutils/lsof, will not build unless the source is installed. In particular, if the port builds a kernel module or directly operates on kernel structures, the source must be installed.

Q: Do I need to build a kernel?

A: Usually not. The supplied `GENERIC` kernel contains the drivers an ordinary computer will need. MAN.FREEBSD-UPDATE.8, the OS binary upgrade tool, cannot upgrade custom kernels, another reason to stick with the `GENERIC` kernel when possible. For computers with very limited RAM, such as embedded systems, it may be worthwhile to build a smaller custom kernel containing just the required drivers.

Q: Should I use DES, Blowfish, or MD5 passwords and how do I specify which form my users receive?

A: OS 9 and later use *SHA512* by default. DES passwords are still available for backwards compatibility with legacy operating systems which still use the less secure password format. OS also supports the Blowfish and MD5 password formats. Which password format to use for new passwords is controlled by the `passwd_format` login capability in `/etc/login.conf`, which takes values of `des`, `blf` (if these are available) or `md5`. See the MAN.LOGIN.CONF.5 manual page for more information about login capabilities.

Q: What are the limits for FFS file systems?

A: For FFS file systems, the largest file system is practically limited by the amount of memory required to MAN.FSCK.8 the file system. MAN.FSCK.8 requires one bit per fragment, which with the default fragment size of 4 KB equates to 32 MB of memory per TB of disk. This does mean that on architectures which limit userland processes to 2 GB (e.g., I386), the maximum MAN.FSCK.8'able filesystem is ~60 TB.

If there was not a MAN.FSCK.8 memory limit the maximum filesystem size would be 2^{64} (blocks) * 32 KB => 16 Exa * 32 KB => 512 ZettaBytes.

The maximum size of a single FFS file is approximately 2 PB with the default block size of 32 KB. Each 32 KB block can point to 4096 blocks. With triple indirect blocks, the calculation is $32\text{ KB} * 12 + 32\text{ KB} * 4096 + 32\text{ KB} * 4096^2 + 32\text{ KB} * 4096^3$. Increasing the block size to 64 KB will increase the max file size by a factor of 16.

Q: Why do I get an error message, readin failed after compiling and booting a new kernel?

A: The world and kernel are out of sync. This is not supported. Be sure to use “make
 buildworld” and make `buildkernel` to update the kernel.

Boot the system by specifying the kernel directly at the second stage, pressing any key when the `|` shows up before loader is started.

Q: Is there a tool to perform post-installation configuration tasks?

A: Yes. `bsdconfig` provides a nice interface to configure OS post-installation.

82.4 Hardware Compatibility

82.4.1 General

Q: I want to get a piece of hardware for my OS system. Which model/brand/type is best?

A: This is discussed continually on the OS mailing lists but is to be expected since hardware changes so quickly. Read through the Hardware Notes for OS REL.CURRENT or REL2.CURRENT and search the mailing list [archives](#) before asking about the latest and greatest hardware. Chances are a discussion about that type of hardware took place just last week.

Before purchasing a laptop, check the archives for A.MOBILE and A.QUESTIONS, or possibly a specific mailing list for a particular hardware type.

Q: What are the limits for memory? Does OS support more than 4 GB of memory (RAM)? More than 16 GB? More than 48 GB?

A: OS as an operating system generally supports as much physical memory (RAM) as the platform it is running on does. Keep in mind that different platforms have different limits for memory; for example I386 without PAE supports at most 4 GB of memory (and usually less than that because of PCI address space) and I386 with PAE supports at most 64 GB memory. As of OS 10, AMD64 platforms support up to 4 TB of physical memory.

Q: Why does OS report less than 4 GB memory when installed on an I386 machine?

A: The total address space on I386 machines is 32-bit, meaning that at most 4 GB of memory is addressable (can be accessed). Furthermore, some addresses in this range are reserved by hardware for different purposes, for example for using and controlling PCI devices, for accessing video memory, and so on. Therefore, the total amount of memory usable by the operating system for its kernel and applications is limited to significantly less than 4 GB. Usually, 3.2 GB to 3.7 GB is the maximum usable physical memory in this configuration.

To access more than 3.2 GB to 3.7 GB of installed memory (meaning up to 4 GB but also more than 4 GB), a special tweak called PAE must be used. PAE stands for Physical Address Extension and is a way for 32-bit x86 CPUs to address more than 4 GB of memory. It remaps the memory that would otherwise be overlaid by address reservations for hardware devices above the 4 GB range and uses it as additional physical memory (see MAN.PAE.4). Using PAE has some drawbacks; this mode of memory access is a little bit slower than the normal (without PAE) mode and loadable modules (see MAN.KLD.4) are not supported. This means all drivers must be compiled into the kernel.

The most common way to enable PAE is to build a new kernel with the special ready-provided kernel configuration file called `PAE`, which is already configured to build a safe kernel. Note that some entries in this kernel configuration file are too conservative and some drivers marked as unready to be used with PAE are actually usable. A rule of thumb is that if the driver is usable on 64-bit architectures (like AMD64), it is also usable with PAE. When creating a custom kernel configuration file, PAE can be enabled by adding the following line:

<code>options</code>	<code>PAE</code>
----------------------	------------------

PAE is not much used nowadays because most new x86 hardware also supports running in 64-bit mode, known as AMD64 or INTEL 64. It has a much larger address space and does not need such tweaks. OS supports AMD64 and it is recommended that this version of OS be used instead of the I386 version if 4 GB or more memory is required.

82.4.2 Architectures and Processors

Q: Does OS support architectures other than the x86?

A: Yes. OS divides support into multiple tiers. Tier 1 architectures, such as i386 or amd64; are fully supported. Tiers 2 and 3 are supported on an if-possible basis. A full explanation of the tier system is available in the Committer's Guide.

A complete list of supported architectures can be found on the [platforms page](#).

Q: Does OS support Symmetric Multiprocessing (SMP)?

A: OS supports symmetric multi-processor (SMP) on all non-embedded platforms (e.g, ARCH.I386, ARCH.AMD64, etc.). SMP is also supported in arm and MIPS kernels, although some CPUs may not support this. OS's SMP implementation uses fine-grained locking, and performance scales nearly linearly with number of CPUs.

MAN.SMP.4 has more details.

Q: What is microcode? How do I install INTEL CPU microcode updates?

A: Microcode is a method of programmatically implementing hardware level instructions. This allows for CPU bugs to be fixed without replacing the on board chip.

Install sysutils/devcpu-data, then add:

```
microcode_update_enable="YES"
```

to /etc/rc.conf

82.4.3 Hard Drives, Tape Drives, and CD and DVD Drives

Q: What kind of hard drives does OS support?

A: OS supports EIDE, SATA, SCSI, and SAS drives (with a compatible controller; see the next section), and all drives using the original "Western Digital" interface (MFM, RLL, ESDI, and of course IDE). A few ESDI controllers that use proprietary interfaces may not work: stick to WD1002/3/6/7 interfaces and clones.

Q: Which SCSI or SAS controllers are supported?

A: See the complete list in the Hardware Notes for OS REL.CURRENT or REL2.CURRENT.

Q: What types of tape drives are supported?

A: OS supports all standard SCSI tape interfaces.

Q: Does OS support tape changers?

A: OS supports SCSI changers using the MAN.CH.4 device and the MAN.CHIO.1 command. The details of how to control the changer can be found in MAN.CHIO.1.

While AMANDA and some other products already understands changers, other applications only know how to move a tape from one point to another/ In this case, keep track of which slot a tape is in and which slot the tape currently in the drive needs to go back to.

Q: Which CD-ROM and CD-RW drives are supported by OS?

A: Any SCSI drive connected to a supported controller is supported. Most ATAPI compatible IDE CD-ROMs are supported.

OS supports any ATAPI-compatible IDE CD-R or CD-RW drive. See MAN.BURNCD.8 for details.

OS also supports any SCSI CD-R or CD-RW drives. Install the sysutils/cdrtools port or package, then use `cdrecord`.

82.4.4 Keyboards and Mice

Q: Is it possible to use a mouse outside the X Window system?

A: The default console driver, MAN.SYSCONS.4, provides the ability to use a mouse pointer in text consoles to cut & paste text. Run the mouse daemon, MAN.MOUSED.8, and turn on the mouse pointer in the virtual console:

```
PROMPT.ROOT moused -p /dev/xxxx -t yyyy
PROMPT.ROOT vidcontrol -m on
```

Where `xxxx` is the mouse device name and `yyyy` is a protocol type for the mouse. The mouse daemon can automatically determine the protocol type of most mice, except old serial mice. Specify the `auto` protocol to invoke automatic detection. If automatic detection does not work, see the MAN.MOUSED.8 manual page for a list of supported protocol types.

For a PS/2 mouse, add `moused_enable="YES"` to `/etc/rc.conf` to start the mouse daemon at boot time. Additionally, to use the mouse daemon on all virtual terminals instead of just the console, add `"allscreens_flags="-m on"` to `/etc/rc.conf`.

When the mouse daemon is running, access to the mouse must be coordinated between the mouse daemon and other programs such as X Windows. Refer to the FAQ *Why does my mouse not work with X?* for more details on this issue.

Q: How do I cut and paste text with a mouse in the text console?

A: It is not possible to remove data using the mouse. However, it is possible to copy and paste. Once the mouse daemon is running as described in the *previous question*, hold down button 1 (left button) and move the mouse to select a region of text. Then, press button 2 (middle button) to paste it at the text cursor. Pressing button 3 (right button) will “extend” the selected region of text.

If the mouse does not have a middle button, it is possible to emulate one or remap buttons using mouse daemon options. See the MAN.MOUSED.8 manual page for details.

Q: My mouse has a fancy wheel and buttons. Can I use them in OS?

A: The answer is, unfortunately, “It depends”. These mice with additional features require specialized driver in most cases. Unless the mouse device driver or the user program has specific support for the mouse, it will act just like a standard two, or three button mouse.

For the possible usage of wheels in the X Window environment, refer to *that section*.

Q: How do I use my delete key in `sh` and `csh`?

A: For the Bourne Shell, add the following lines to `~/ .shrc`. See MAN.SH.1 and MAN.EDITRC.5.

```
bind ^? ed-delete-next-char # for console
bind ^[[3~ ed-delete-next-char # for xterm
```

For the C Shell, add the following lines to `~/ .cshrc`. See MAN.CSH.1.

```
bindkey ^? delete-char # for console
bindkey ^[[3~ delete-char # for xterm
```

For more information, see [this page](#).

82.4.5 Other Hardware

Q: Workarounds for no sound from my MAN.PCM.4 sound card?

A: Some sound cards set their output volume to 0 at every boot. Run the following command every time the machine boots:

```
PROMPT.ROOT mixer pcm 100 vol 100 cd 100
```

Q: Does OS support power management on my laptop?

A: OS supports the ACPI features found in modern hardware. Further information can be found in MAN.ACPI.4.

82.5 Troubleshooting

Q: Why is OS finding the wrong amount of memory on I386 hardware?

A: The most likely reason is the difference between physical memory addresses and virtual addresses.

The convention for most PC hardware is to use the memory area between 3.5 GB and 4 GB for a special purpose (usually for PCI). This address space is used to access PCI hardware. As a result real, physical memory cannot be accessed by that address space.

What happens to the memory that should appear in that location is hardware dependent. Unfortunately, some hardware does nothing and the ability to use that last 500 MB of RAM is entirely lost.

Luckily, most hardware remaps the memory to a higher location so that it can still be used. However, this can cause some confusion when watching the boot messages.

On a 32-bit version of OS, the memory appears lost, since it will be remapped above 4 GB, which a 32-bit kernel is unable to access. In this case, the solution is to build a PAE enabled kernel. See the entry on memory limits for more information.

On a 64-bit version of OS, or when running a PAE-enabled kernel, OS will correctly detect and remap the memory so it is usable. During boot, however, it may seem as if OS is detecting more memory than the system really has, due to the described remapping. This is normal and the available memory will be corrected as the boot process completes.

Q: Why do my programs occasionally die with Signal 11 errors?

A: Signal 11 errors are caused when a process has attempted to access memory which the operating system has not granted it access to. If something like this is happening at seemingly random intervals, start investigating the cause.

These problems can usually be attributed to either:

1. If the problem is occurring only in a specific custom application, it is probably a bug in the code.
2. If it is a problem with part of the base OS system, it may also be buggy code, but more often than not these problems are found and fixed long before us general FAQ readers get to use these bits of code (that is what -CURRENT is for).

It is probably not a OS bug if the problem occurs compiling a program, but the activity that the compiler is carrying out changes each time.

For example, if “make buildworld” fails while trying to compile `ls.c` into

`ls.o` and, when run again, it fails in the same place, this is a broken build. Try updating source and try again. If the compile fails elsewhere, it is almost certainly due to hardware.

In the first case, use a debugger such as MAN.GDB.1 to find the point in the program which is attempting to access a bogus address and fix it.

In the second case, verify which piece of hardware is at fault.

Common causes of this include:

1. The hard disks might be overheating: Check that the fans are still working, as the disk and other hardware might be overheating.

2. The processor running is overheating: This might be because the processor has been overclocked, or the fan on the processor might have died. In either case, ensure that the hardware is running at what it is specified to run at, at least while trying to solve this problem. If it is not, clock it back to the default settings.)

Regarding overclocking, it is far cheaper to have a slow system than a fried system that needs replacing! Also the community is not sympathetic to problems on overclocked systems.

3. Dodgy memory: if multiple memory SIMMS/DIMMS are installed, pull them all out and try running the machine with each SIMM or DIMM individually to narrow the problem down to either the problematic DIMM/SIMM or perhaps even a combination.
4. Over-optimistic motherboard settings: the BIOS settings, and some motherboard jumpers, provide options to set various timings. The defaults are often sufficient, but sometimes setting the wait states on RAM too low, or setting the “RAM Speed: Turbo” option will cause strange behavior. A possible idea is to set to BIOS defaults, after noting the current settings first.
5. Unclean or insufficient power to the motherboard. Remove any unused I/O boards, hard disks, or CD-ROMs, or disconnect the power cable from them, to see if the power supply can manage a smaller load. Or try another power supply, preferably one with a little more power. For instance, if the current power supply is rated at 250 Watts, try one rated at 300 Watts.

Read the section on *Signal 11* for a further explanation and a discussion on how memory testing software or hardware can still pass faulty memory. There is an extensive FAQ on this at [the SIG11 problem FAQ](#).

Finally, if none of this has helped, it is possibly a bug in OS. Follow *these instructions* to send a problem report.

Q: My system crashes with either Fatal trap 12: page fault in kernel mode, or panic:, and spits out a bunch of information. What should I do?

A: The OS developers are interested in these errors, but need more information than just the error message. Copy the full crash message. Then consult the FAQ section on *kernel panics*, build a debugging kernel, and get a backtrace. This might sound difficult, but does not require any programming skills. Just follow the instructions.

Q: What is the meaning of the error maxproc limit exceeded by uid %i, please see tuning(7) and login.conf(5)?

A: The OS kernel will only allow a certain number of processes to exist at one time. The number is based on the `kern.maxusers` `MAN.SYSCTL.8` variable. `kern.maxusers` also affects various other in-kernel limits, such as network buffers. If the machine is heavily loaded, increase `kern.maxusers`. This will increase these other system limits in addition to the maximum number of processes.

To adjust the `kern.maxusers` value, see the File/Process Limits section of the Handbook. While that section refers to open files, the same limits apply to processes.

If the machine is lightly loaded but running a very large number of processes, adjust the `kern.maxproc` tunable by defining it in `/boot/loader.conf`. The tunable will not get adjusted until the system is rebooted. For more information about tuning tunables, see `MAN.LOADER.CONF.5`. If these processes are being run by a single user, adjust `kern.maxprocperuid` to be one less than the new `kern.maxproc` value. It must be at least one less because one system program, `MAN.INIT.8`, must always be running.

Q: Why do full screen applications on remote machines misbehave?

A: The remote machine may be setting the terminal type to something other than `xterm` which is required by the OS console. Alternatively the kernel may have the wrong values for the width and height of the terminal.

Check the value of the `TERM` environment variable is `xterm`. If the remote machine does not support that try `vt100`.

Run `stty -a` to check what the kernel thinks the terminal dimensions are. If they are incorrect, they can be changed by running “`stty rows RR cols`”

“`CC`”.

Alternatively, if the client machine has `x11/xterm` installed, then running `resize` will query the terminal for the correct dimensions and set them.

Q: Why does it take so long to connect to my computer via `ssh` or `telnet`?

A: The symptom: there is a long delay between the time the TCP connection is established and the time when the client software asks for a password (or, in `MAN.TELNET.1`'s case, when a login prompt appears).

The problem: more likely than not, the delay is caused by the server software trying to resolve the client's IP address into a hostname. Many servers, including the Telnet and SSH servers that come with OS, do this to store the hostname in a log file for future reference by the administrator.

The remedy: if the problem occurs whenever connecting the client computer to any server, the problem is with the client. If the problem only occurs when someone connects to the server computer, the problem is with the server.

If the problem is with the client, the only remedy is to fix the DNS so the server can resolve it. If this is on a local network, consider it a server problem and keep reading. If this is on the Internet, contact your ISP.

If the problem is with the server on a local network, configure the server to resolve address-to-hostname queries for the local address range. See `MAN.HOSTS.5` and `MAN.NAMED.8` for more information. If this is on the Internet, the problem may be that the local server's resolver is not functioning correctly. To check, try to look up another host such as `www.yahoo.com`. If it does not work, that is the problem.

Following a fresh install of OS, it is also possible that domain and name server information is missing from `/etc/resolv.conf`. This will often cause a delay in SSH, as the option `UseDNS` is set to `yes` by default in `/etc/ssh/sshd_config`. If this is causing the problem, either fill in the missing information in `/etc/resolv.conf` or set `UseDNS` to `no` in `sshd_config` as a temporary workaround.

Q: Why does `file: table is full` show up repeatedly in `MAN.DMESG.8`?

A: This error message indicates that the number of available file descriptors have been exhausted on the system. Refer to the `kern.maxfiles` section of the Tuning Kernel Limits section of the Handbook for a discussion and solution.

Q: Why does the clock on my computer keep incorrect time?

A: The computer has two or more clocks, and OS has chosen to use the wrong one.

Run `MAN.DMESG.8`, and check for lines that contain `Timecounter`. The one with the highest quality value that OS chose.

```
PROMPT.ROOT dmesg | grep Timecounter
Timecounter "i8254" frequency 1193182 Hz quality 0
Timecounter "ACPI-fast" frequency 3579545 Hz quality 1000
Timecounter "TSC" frequency 2998570050 Hz quality 800
Timecounters tick every 1.000 msec
```

Confirm this by checking the `kern.timecounter.hardware` `MAN.SYSCTL.3`.

```
PROMPT.ROOT sysctl kern.timecounter.hardware
kern.timecounter.hardware: ACPI-fast
```

It may be a broken ACPI timer. The simplest solution is to disable the ACPI timer in `/boot/loader.conf`:

```
debug.acpi.disabled="timer"
```

Or the BIOS may modify the TSC clock—perhaps to change the speed of the processor when running from batteries, or going into a power saving mode, but OS is unaware of these adjustments, and appears to gain or lose time.

In this example, the `i8254` clock is also available, and can be selected by writing its name to the `kern.timecounter.hardware` `MAN.SYSCTL.3`.

```
PROMPT.ROOT sysctl kern.timecounter.hardware=i8254
kern.timecounter.hardware: TSC -> i8254
```

The computer should now start keeping more accurate time.

To have this change automatically run at boot time, add the following line to `/etc/sysctl.conf`:


```
kern.timecounter.hardware=i8254
```

Q: What does the error `swap_pager: indefinite wait buffer: mean?`

A: This means that a process is trying to page memory to disk, and the page attempt has hung trying to access the disk for more than 20 seconds. It might be caused by bad blocks on the disk drive, disk wiring, cables, or any other disk I/O-related hardware. If the drive itself is bad, disk errors will appear in `/var/log/messages` and in the output of `dmesg`. Otherwise, check the cables and connections.

Q: What is a lock order reversal?

A: The OS kernel uses a number of resource locks to arbitrate contention for certain resources. When multiple kernel threads try to obtain multiple resource locks, there's always the potential for a deadlock, where two threads have each obtained one of the locks and blocks forever waiting for the other thread to release one of the other locks. This sort of locking problem can be avoided if all threads obtain the locks in the same order.

A run-time lock diagnostic system called `MAN.WITNESS.4`, enabled in `OS.CURRENT` and disabled by default for stable branches and releases, detects the potential for deadlocks due to locking errors, including errors caused by obtaining multiple resource locks with a different order from different parts of the kernel. The `MAN.WITNESS.4` framework tries to detect this problem as it happens, and reports it by printing a message to the system console about a lock order reversal (often referred to also as `LOR`).

It is possible to get false positives, as `MAN.WITNESS.4` is conservative. A true positive report *does not* mean that a system is dead-locked; instead it should be understood as a warning that a deadlock could have happened here.

Note

Problematic `LORs` tend to get fixed quickly, so check `A.CURRENT.URL` before posting to the mailing lists.

Q: What does `Called ... with the following non-sleepable locks held` mean?

A: This means that a function that may sleep was called while a mutex (or other unsleepable) lock was held.

The reason this is an error is because mutexes are not intended to be held for long periods of time; they are supposed to only be held to maintain short periods of synchronization. This programming contract allows device drivers to use mutexes to synchronize with the rest of the kernel during interrupts. Interrupts (under OS) may not sleep. Hence it is imperative that no subsystem in the kernel block for an extended period while holding a mutex.

To catch such errors, assertions may be added to the kernel that interact with the `MAN.WITNESS.4` subsystem to emit a warning or fatal error (depending on the system configuration) when a potentially blocking call is made while holding a mutex.

In summary, such warnings are non-fatal, however with unfortunate timing they could cause undesirable effects ranging from a minor blip in the system's responsiveness to a complete system lockup.

For additional information about locking in OS see `MAN.LOCKING.9`.

Q: Why does `buildworld/installworld` die with the message `touch: not found?`

A: This error does not mean that the `MAN.TOUCH.1` utility is missing. The error is instead probably due to the dates of the files being set sometime in the future. If the CMOS clock is set to local time, run `adjkerntz -i` to adjust the kernel clock when booting into single-user mode.

82.6 User Applications

Q: Where are all the user applications?

A: Refer to the ports page for info on software packages ported to OS. The list currently tops `OS.NUMPORTS` and is growing daily, so come back to check often or subscribe to the `A.ANNOUNCE` for periodic updates on new entries.

Most ports should work on all supported versions of OS. Those that do not are specifically marked as such. Each time a OS release is made, a snapshot of the ports tree at the time of release is also included in the `ports/` directory.

OS supports compressed binary packages to easily install and uninstall ports. Use `MAN.PKG.7` to control the installation of packages.

Q: How do I download the Ports tree? Should I be using SVN?

A: Any of the methods listed here work:

- Use `portsnap` for most use cases. Refer to `Using the Ports Collection` for instructions on how to use this tool.
- Use SVN if custom patches to the ports tree are needed. Refer to `Using Subversion` for details.
- Use CTM, as described in `Using CTM` to receive patches by email over an unreliable Internet connection.

Q: Does OS support JAVA?

A: Yes. Refer to <http://www.FreeBSD.org/java/> for more information.

Q: Why can I not build this port on my `REL2.RELX -`, or `REL.RELX -STABLE` machine?

A: If the installed OS version lags significantly behind `-CURRENT` or `-STABLE`, update the Ports Collection using the instructions in `Using the Ports Collection`. If the system is up-to-date, someone might have committed a change to the port which works for `-CURRENT` but which broke the port for `-STABLE`. [Submit](#) a bug report, since the Ports Collection is supposed to work for both the `-CURRENT` and `-STABLE` branches.

Q: I just tried to build `INDEX` using `make index`, and it failed. Why?

A: First, make sure that the Ports Collection is up-to-date. Errors that affect building `INDEX` from an up-to-date copy of the Ports Collection are high-visibility and are thus almost always fixed immediately.

There are rare cases where `INDEX` will not build due to odd cases involving `WITH_*` or `WITHOUT_*` variables being set in `make.conf`. If you suspect that this is the case, try to make `INDEX` with those make variables turned off before reporting it to `A.PORTS`.

Q: I updated the sources, now how do I update my installed ports?

A: OS does not include a port upgrading tool, but it does have some tools to make the upgrade process somewhat easier. Additional tools are available to simplify port handling and are described the `Upgrading Ports` section in the OS Handbook.

Q: Do I need to recompile every port each time I perform a major version update?

A: Yes! While a recent system will run with software compiled under an older release, things will randomly crash and fail to work once other ports are installed or updated.

When the system is upgraded, various shared libraries, loadable modules, and other parts of the system will be replaced with newer versions. Applications linked against the older versions may fail to start or, in other cases, fail to function properly.

For more information, see the section on upgrades in the OS Handbook.

Q: Do I need to recompile every port each time I perform a minor version update?

A: In general, no. OS developers do their utmost to guarantee binary compatibility across all releases with the same major version number. Any exceptions will be documented in the Release Notes, and advice given there should be followed.

Q: Why is `/bin/sh` so minimal? Why does OS not use `bash` or another shell?

A: Many people need to write shell scripts which will be portable across many systems. That is why POSIX specifies the shell and utility commands in great detail. Most scripts are written in Bourne shell (`MAN.SH.1`), and because several important programming interfaces (`MAN.MAKE.1`, `MAN.SYSTEM.3`, `MAN.POPEN.3`, and analogues in higher-level scripting languages like Perl and Tcl) are specified to use the Bourne shell to interpret commands. Because

the Bourne shell is so often and widely used, it is important for it to be quick to start, be deterministic in its behavior, and have a small memory footprint.

The existing implementation is our best effort at meeting as many of these requirements simultaneously as we can. To keep `/bin/sh` small, we have not provided many of the convenience features that other shells have. That is why other more featureful shells like `bash`, `scsh`, `MAN.TCSH.1`, and `zsh` are available. Compare the memory utilization of these shells by looking at the “VSZ” and “RSS” columns in a `ps -u` listing.

Q: How do I create audio CDs from my MIDI files?

A: To create audio CDs from MIDI files, first install `audio/timidity++` from ports then install manually the GUS patches set by Eric A. Welsh, available at <http://alleg.sourceforge.net/digmid.html>. After `TiMidity++` has been installed properly, MIDI files may be converted to WAV files with the following command line:

```
PROMPT.USER timidity -Ow -s 44100 -o /tmp/juke/01.wav 01.mid
```

The WAV files can then be converted to other formats or burned onto audio CDs, as described in the OS Handbook.

82.7 Kernel Configuration

Q: I would like to customize my kernel. Is it difficult?

A: Not at all! Check out the kernel config section of the Handbook.

Note

The new `kernel` will be installed to the `/boot/kernel` directory along with its modules, while the old kernel and its modules will be moved to the `/boot/kernel.old` directory. If a mistake is made in the configuration, simply boot the previous version of the kernel.

Q: Why is my kernel so big?

A: `GENERIC` kernels shipped with OS and later are compiled in *debug mode*. Kernels built in debug mode contain many symbols in separate files that are used for debugging, thus greatly increasing the size of `/boot/kernel/`. Note that there will be little or no performance loss from running a debug kernel, and it is useful to keep one around in case of a system panic.

However, when running low on disk space, there are different options to reduce the size of `/boot/kernel/`.

To not install the symbol files, make sure the following line exists in `/etc/src.conf`:

```
WITHOUT_KERNEL_SYMBOLS=yes
```

For more information see `MAN.SRC.CONF.5`.

If you do not want to build a debug kernel, make sure that both of the following are true:

- This line does not exist in the kernel configuration file:

```
makeoptions DEBUG=-g
```

- Do not run `MAN.CONFIG.8` with `-g`.

Either of the above settings will cause the kernel to be built in debug mode.

To build and install only the specified modules, list them in `/etc/make.conf`:

```
MODULES_OVERRIDE= accf_http ipfw
```

Replace `accf_http ipfw` with a list of needed modules. Only the listed modules will be built. This reduces the size of the kernel directory and decreases the amount of time needed to build the kernel. For more information, read `/usr/share/examples/etc/make.conf`.

Unneeded devices can be removed from the kernel to further reduce the size. See ? for more information.

To put any of these options into effect, follow the instructions to build and install the new kernel.

Most kernels (`/boot/kernel/kernel`) tend to be around 12 MB to 16 MB.

Q: Why does every kernel I try to build fail to compile, even `GENERIC`?

A: There are a number of possible causes for this problem:

- The source tree is different from the one used to build the currently running system. When attempting an upgrade, read `/usr/src/UPDATING`, paying particular attention to the “COMMON ITEMS” section at the end.
- The “`make buildkernel`” command did not complete successfully. The “`make buildkernel`” target relies on files generated by the `make buildworld` target to complete its job correctly.
- Even when building *OS-STABLE*, it is possible that the source tree was fetched at a time when it was either being modified or it was broken. Only releases are guaranteed to be buildable, although *OS-STABLE* builds fine the majority of the time. Try re-fetching the source tree and see if the problem goes away. Try using a different mirror in case the previous one is having problems.

Q: Which scheduler is in use on a running system?

A: The name of the scheduler currently being used is directly available as the value of the `kern.sched.name` sysctl:

```
PROMPT.USER sysctl kern.sched.name
kern.sched.name: ULE
```

Q: What is `kern.sched.quantum`?

A: `kern.sched.quantum` is the maximum number of ticks a process can run without being preempted in the 4BSD scheduler.

82.8 Disks, File Systems, and Boot Loaders

Q: How can I add my new hard disk to my OS system?

A: See the Adding Disks section in the OS Handbook.

Q: How do I move my system over to my huge new disk?

A: The best way is to reinstall the operating system on the new disk, then move the user data over. This is highly recommended when tracking *-STABLE* for more than one release or when updating a release instead of installing a new one. Install `booteasy` on both disks with `MAN.BOOT0CFG.8` and dual boot until you are happy with the new configuration. Skip the next paragraph to find out how to move the data after doing this.

Alternatively, partition and label the new disk with either `MAN.SADE.8` or `MAN.GPART.8`. If the disks are MBR-formatted, `booteasy` can be installed on both disks with `MAN.BOOT0CFG.8` so that the computer can dual boot to the old or new system after the copying is done.

Once the new disk set up, the data cannot just be copied. Instead, use tools that understand device files and sysctl flags, such as `MAN.DUMP.8`. Although it is recommended to move the data while in single-user mode, it is not required.

When the disks are formatted with UFS, never use anything but `MAN.DUMP.8` and `MAN.RESTORE.8` to move the root file system. These commands should also be used when moving a single partition to another empty partition. The sequence of steps to use `dump` to move the data from one UFS partitions to a new partition is:

`newfs` the new partition.

mount it on a temporary mount point.

cd to that directory.

dump the old partition, piping output to the new one.

For example, to move `/dev/adals1a` with `/mnt` as the temporary mount point, type:

```
PROMPT.ROOT newfs /dev/adals1a
PROMPT.ROOT mount /dev/adals1a /mnt
PROMPT.ROOT cd /mnt
PROMPT.ROOT dump 0af - / | restore rf -
```

Rearranging partitions with `dump` takes a bit more work. To merge a partition like `/var` into its parent, create the new partition large enough for both, move the parent partition as described above, then move the child partition into the empty directory that the first move created:

```
PROMPT.ROOT newfs /dev/adals1a
PROMPT.ROOT mount /dev/adals1a /mnt
PROMPT.ROOT cd /mnt
PROMPT.ROOT dump 0af - / | restore rf -
PROMPT.ROOT cd var
PROMPT.ROOT dump 0af - /var | restore rf -
```

To split a directory from its parent, say putting `/var` on its own partition when it was not before, create both partitions, then mount the child partition on the appropriate directory in the temporary mount point, then move the old single partition:

```
PROMPT.ROOT newfs /dev/adals1a
PROMPT.ROOT newfs /dev/adals1d
PROMPT.ROOT mount /dev/adals1a /mnt
PROMPT.ROOT mkdir /mnt/var
PROMPT.ROOT mount /dev/adals1d /mnt/var
PROMPT.ROOT cd /mnt
PROMPT.ROOT dump 0af - / | restore rf -
```

The `MAN.CPIO.1` and `MAN.PAX.1` utilities are also available for moving user data. These are known to lose file flag information, so use them with caution.

Q: Which partitions can safely use Soft Updates? I have heard that Soft Updates on `/` can cause problems. What about Journaled Soft Updates?

A: Short answer: Soft Updates can usually be safely used on all partitions.

Long answer: Soft Updates has two characteristics that may be undesirable on certain partitions. First, a Soft Updates partition has a small chance of losing data during a system crash. The partition will not be corrupted as the data will simply be lost. Second, Soft Updates can cause temporary space shortages.

When using Soft Updates, the kernel can take up to thirty seconds to write changes to the physical disk. When a large file is deleted the file still resides on disk until the kernel actually performs the deletion. This can cause a very simple race condition. Suppose one large file is deleted and another large file is immediately created. The first large file is not yet actually removed from the physical disk, so the disk might not have enough room for the second large file. This will produce an error that the partition does not have enough space, even though a large chunk of space has just been released. A few seconds later, the file creation works as expected.

If a system should crash after the kernel accepts a chunk of data for writing to disk, but before that data is actually written out, data could be lost. This risk is extremely small, but generally manageable.

These issues affect all partitions using Soft Updates. So, what does this mean for the root partition?

Vital information on the root partition changes very rarely. If the system crashed during the thirty-second window after such a change is made, it is possible that data could be lost. This risk is negligible for most applications, but be

aware that it exists. If the system cannot tolerate this much risk, do not use Soft Updates on the root file system!

`/` is traditionally one of the smallest partitions. If `/tmp` is on `/`, there may be intermittent space problems. Symlinking `/tmp` to `/var/tmp` will solve this problem.

Finally, `MAN.DUMP.8` does not work in live mode (`-L`) on a filesystem, with Journaled Soft Updates (`SU+J`).

Q: Can I mount other foreign file systems under OS?

A: OS supports a variety of other file systems.

UFS UFS CD-ROMs can be mounted directly on OS. Mounting disk partitions from Digital UNIX and other systems that support UFS may be more complex, depending on the details of the disk partitioning for the operating system in question.

ext2/ext3 OS supports `ext2fs` and `ext3fs` partitions. See `MAN.EXT2FS.5` for more information.

NTFS FUSE based NTFS support is available as a port (`sysutils/fusefs-ntfs`). For more information see [ntfs-3g](#).

FAT OS includes a read-write FAT driver. For more information, see `MAN.MOUNT.MSDOSFS.8`.

ZFS OS includes a port of SUN's ZFS driver. The current recommendation is to use it only on `ARCH.AMD64` platforms with sufficient memory. For more information, see `MAN.ZFS.8`.

OS includes the Network File System NFS and the OS Ports Collection provides several FUSE applications to support many other file systems.

Q: How do I mount a secondary DOS partition?

A: The secondary DOS partitions are found after *all* the primary partitions. For example, if E is the second DOS partition on the second SCSI drive, there will be a device file for "slice 5" in `/dev`. To mount it:

```
PROMPT.ROOT mount -t msdosfs /dev/dals5 /dos/e
```

Q: Is there a cryptographic file system for OS?

A: Yes, `MAN.GBDE.8` and `MAN.GELI.8`. See the Encrypting Disk Partitions section of the OS Handbook.

Q: How do I boot OS and LINUX using GRUB?

A: To boot OS using GRUB, add the following to either `/boot/grub/menu.lst` or `/boot/grub/grub.conf`, depending upon which is used by the LINUX distribution.

```
title OS 9.1
    root (hd0,a)
    kernel /boot/loader
```

Where `hd0,a` points to the root partition on the first disk. To specify the slice number, use something like this (`hd0,2,a`). By default, if the slice number is omitted, GRUB searches the first slice which has the `a` partition.

Q: How do I boot OS and LINUX using BootEasy?

A: Install LILO at the start of the LINUX boot partition instead of in the Master Boot Record. You can then boot LILO from BootEasy.

This is recommended when running WINDOWS and LINUX as it makes it simpler to get LINUX booting again if WINDOWS is reinstalled.

Q: How do I change the boot prompt from `???` to something more meaningful?

A: This cannot be accomplished with the standard boot manager without rewriting it. There are a number of other boot managers in the `sysutils` category of the Ports Collection.

Q: How do I use a new removable drive?

A: If the drive already has a file system on it, use a command like this:

```
PROMPT.ROOT mount -t msdosfs /dev/da0s1 /mnt
```

If the drive will only be used with OS systems, partition it with UFS or ZFS. This will provide long filename support, improvement in performance, and stability. If the drive will be used by other operating systems, a more portable choice, such as msdosfs, is better.

```
PROMPT.ROOT dd if=/dev/zero of=/dev/da0 count=2
PROMPT.ROOT gpart create -s GPT /dev/da0
PROMPT.ROOT gpart add -t freebsd-ufs /dev/da0
```

Finally, create a new file system:

```
PROMPT.ROOT newfs /dev/da0p1
```

and mount it:

```
PROMPT.ROOT mount /dev/da0s1 /mnt
```

It is a good idea to add a line to `/etc/fstab` (see MAN.FSTAB.5) so you can just type `mount /mnt` in the future:

```
/dev/da0p1 /mnt ufs rw,noauto 0 0
```

Q: Why do I get Incorrect super block when mounting a CD?

A: The type of device to mount must be specified. This is described in the Handbook section on Using Data CDs.

Q: Why do I get Device not configured when mounting a CD?

A: This generally means that there is no CD in the drive, or the drive is not visible on the bus. Refer to the Using Data CDs section of the Handbook for a detailed discussion of this issue.

Q: Why do all non-English characters in filenames show up as “?” on my CDs when mounted in OS?

A: The CD probably uses the “Joliet” extension for storing information about files and directories. This is discussed in the Handbook section on Using Data CD-ROMs.

Q: A CD burned under OS cannot be read under any other operating system. Why?

A: This means a raw file was burned to the CD, rather than creating an ISO 9660 file system. Take a look at the Handbook section on Using Data CDs.

Q: How can I create an image of a data CD?

A: This is discussed in the Handbook section on Writing Data to an ISO File System. For more on working with CD-ROMs, see the Creating CDs Section in the Storage chapter in the Handbook.

Q: Why can I not mount an audio CD?

A: Trying to mount an audio CD will produce an error like `cd9660: /dev/acd0c: Invalid argument`. This is because `mount` only works on file systems. Audio CDs do not have file systems; they just have data. Instead, use a program that reads audio CDs, such as the `audio/xmcd` package or `port`.

Q: How do I mount a multi-session CD?

A: By default, MAN.MOUNT.8 will attempt to mount the last data track (session) of a CD. To load an earlier session, use the `-s` command line argument. Refer to MAN.MOUNT.CD9660.8 for specific examples.

Q: How do I let ordinary users mount CD-ROMs, DVDs, USB drives, and other removable media?

A: As root set the `sysctl` variable `vfs.usermount` to 1.

```
PROMPT.ROOT sysctl vfs.usermount=1
```

To make this persist across reboots, add the line `vfs.usermount=1` to `/etc/sysctl.conf` so that it is reset at system boot time.

Users can only mount devices they have read permissions to. To allow users to mount a device permissions must be set in `/etc/devfs.conf`.

For example, to allow users to mount the first USB drive add:

```
# Allow all users to mount a USB drive.
    own      /dev/da0      root:operator
    perm     /dev/da0      0666
```

All users can now mount devices they could read onto a directory that they own:

```
PROMPT.USER mkdir ~/my-mount-point
PROMPT.USER mount -t msdosfs /dev/da0 ~/my-mount-point
```

Unmounting the device is simple:

```
PROMPT.USER umount ~/my-mount-point
```

Enabling `vfs.usermount`, however, has negative security implications. A better way to access MS-DOS formatted media is to use the emulators/mttools package in the Ports Collection.

Note

The device name used in the previous examples must be changed according to the configuration.

Q: The `du` and `df` commands show different amounts of disk space available. What is going on?

A: This is due to how these commands actually work. `du` goes through the directory tree, measures how large each file is, and presents the totals. `df` just asks the file system how much space it has left. They seem to be the same thing, but a file without a directory entry will affect `df` but not `du`.

When a program is using a file, and the file is deleted, the file is not really removed from the file system until the program stops using it. The file is immediately deleted from the directory listing, however. As an example, consider a file that is large enough that its presence affects the output of `du` and `df`. If this file is deleted while using `more` on it, `more` does not immediately choke and complain that it cannot view the file. The entry is removed from the directory so no other program or user can access it. However, `du` shows that it is gone as it has walked the directory tree and the file is not listed. `df` shows that it is still there, as the file system knows that `more` is still using that space. Once the `more` session ends, `du` and `df` will agree.

This situation is common on web servers. Many people set up a OS web server and forget to rotate the log files. The access log fills up `/var`. The new administrator deletes the file, but the system still complains that the partition is full. Stopping and restarting the web server program would free the file, allowing the system to release the disk space. To prevent this from happening, set up `MAN.NEWSYSLOG.8`.

Note that Soft Updates can delay the freeing of disk space and it can take up to 30 seconds for the change to be visible.

Q: How can I add more swap space?

A: This section of the Handbook describes how to do this.

Q: Why does OS see my disk as smaller than the manufacturer says it is?

A: Disk manufacturers calculate gigabytes as a billion bytes each, whereas OS calculates them as 1,073,741,824 bytes each. This explains why, for example, OS's boot messages will report a disk that supposedly has 80 GB as holding 76,319 MB.

Also note that OS will (by default) *reserve* 8% of the disk space.

Q: How is it possible for a partition to be more than 100% full?

A: A portion of each UFS partition (8%, by default) is reserved for use by the operating system and the root user. `MAN.DF.1` does not count that space when calculating the `Capacity` column, so it can exceed 100%. Notice that the `Blocks` column is always greater than the sum of the `Used` and `Avail` columns, usually by a factor of 8%.

For more details, look up `-m` in `MAN.TUNEF.8`.

Q: Why does OS pause for a long time at boot when the system has large amounts of ram?

A: OS does a short memory test early in the boot process. This test usually only takes several seconds, however if the system has many 10s or 100s of gigabytes of memory it can take up to a few minutes. This test can be disabled by setting `hw.memtest.tests` to 0 in `/boot/loader.conf`.

For more details, see `MAN.LOADER.CONF.5`.

82.9 ZFS

Q: What is the minimum amount of RAM one should have to run ZFS?

A: A minimum of 4GB of RAM is required for comfortable usage, but individual workloads can vary widely.

Q: What is the ZIL and when does it get used?

A: The ZIL ((ZFS intent log) is a write log used to implement posix write commitment semantics across crashes. Normally writes are bundled up into transaction groups and written to disk when filled (“Transaction Group Commit”). However syscalls like `MAN.FSYNC.2` require a commitment that the data is written to stable storage before returning. The ZIL is needed for writes that have been acknowledged as written but which are not yet on disk as part of a transaction. The transaction groups are timestamped. In the event of a crash the last valid timestamp is found and missing data is merged in from the ZIL.

Q: Do I need a SSD for ZIL?

A: By default, ZFS stores the ZIL in the pool with all the data. If an application has a heavy write load, storing the ZIL in a separate device that has very fast synchronous, sequential write performance can improve overall system. For other workloads, a SSD is unlikely to make much of an improvement.

Q: What is the L2ARC?

A: The L2ARC is a read cache stored on a fast device such as an SSD. This cache is not persistent across reboots. Note that RAM is used as the first layer of cache and the L2ARC is only needed if there is insufficient RAM.

L2ARC needs space in the ARC to index it. So, perversely, a working set that fits perfectly in the ARC will not fit perfectly any more if a L2ARC is used because part of the ARC is holding the L2ARC index, pushing part of the working set into the L2ARC which is slower than RAM.

Q: Is enabling deduplication advisable?

A: Generally speaking, no.

Deduplication takes up a significant amount of RAM and may slow down read and write disk access times. Unless one is storing data that is very heavily duplicated, such as virtual machine images or user backups, it is possible that deduplication will do more harm than good. Another consideration is the inability to revert deduplication status. If data is written when deduplication is enabled, disabling dedup will not cause those blocks which were deduplicated to be replicated until they are next modified.

Deduplication can also lead to some unexpected situations. In particular, deleting files may become much slower.

Q: I cannot delete or create files on my ZFS pool. How can I fix this?

A: This could happen because the pool is 100% full. ZFS requires space on the disk to write transaction metadata. To restore the pool to a usable state, truncate the file to delete:

```
PROMPT.USER truncate -s 0 unimportant-file
```

File truncation works because a new transaction is not started, new spare blocks are created instead.

Note

On systems with additional ZFS dataset tuning, such as deduplication, the space may not be immediately available

Q: Does ZFS support TRIM for Solid State Drives?

A: ZFS TRIM support was added to OS 10-CURRENT with revision r240868. ZFS TRIM support was added to all OS-STABLE branches in r252162 and r251419, respectively.

ZFS TRIM is enabled by default, and can be turned off by adding this line to `/etc/sysctl.conf`:

```
vfs.zfs.trim_disable=1

**Note**

ZFS TRIM may not work with all configurations, such as a ZFS
filesystem on a GELI-backed device.
```

82.10 System Administration

Q: Where are the system start-up configuration files?

A: The primary configuration file is `/etc/defaults/rc.conf` which is described in MAN.RC.CONF.5. System startup scripts such as `/etc/rc` and `/etc/rc.d`, which are described in MAN.RC.8, include this file. *Do not edit this file!* Instead, to edit an entry in `/etc/defaults/rc.conf`, copy the line into `/etc/rc.conf` and change it there.

For example, if to start MAN.NAMED.8, the included DNS server:

```
PROMPT.ROOT echo 'named_enable="YES"' >> /etc/rc.conf
```

To start up local services, place shell scripts in the `/usr/local/etc/rc.d` directory. These shell scripts should be set executable, the default file mode is 555.

Q: How do I add a user easily?

A: Use the MAN.ADDUSER.8 command, or the MAN.PW.8 command for more complicated situations.

To remove the user, use the MAN.RMUSER.8 command or, if necessary, MAN.PW.8.

Q: Why do I keep getting messages like root: not found after editing `/etc/crontab`?

A: This is normally caused by editing the system crontab. This is not the correct way to do things as the system crontab has a different format to the per-user crontabs. The system crontab has an extra field, specifying which user to run the command as. MAN.CRON.8 assumes this user is the first word of the command to execute. Since no such command exists, this error message is displayed.

To delete the extra, incorrect crontab:

```
PROMPT.ROOT crontab -r
```

Q: Why do I get the error, you are not in the correct group to su root when I try to `su` to root?

A: This is a security feature. In order to `su` to root, or any other account with superuser privileges, the user account must be a member of the wheel group. If this feature were not there, anybody with an account on a system who also found out root's password would be able to gain superuser level access to the system.

To allow someone to `su` to root, put them in the wheel group using `pw`:

```
PROMPT.ROOT pw groupmod wheel -m lisa
```

The above example will add user `lisa` to the group `wheel`.

Q: I made a mistake in `rc.conf`, or another startup file, and now I cannot edit it because the file system is read-only. What should I do?

A: Restart the system using “`boot -s`” at the loader prompt to enter single-user mode. When prompted for a shell pathname, press Enter and run “`mount -urw /`” to re-mount the root file system in read/write mode.

You may also need to run “`mount -a -t ufs`” to mount the file system where your favorite

editor is defined. If that editor is on a network file system, either configure the network manually before mounting the network file systems, or use an editor which resides on a local file system, such as `MAN.ED.1`.

In order to use a full screen editor such as `MAN.VI.1` or `MAN.EMACS.1`, run `export TERM=xterm` on OS 9.0+, or `export TERM=cons25` on OS 8.X so that these editors can load the correct data from the `MAN.TERMCAP.5` database.

After performing these steps, edit `/etc/rc.conf` to fix the syntax error. The error message displayed immediately after the kernel boot messages should indicate the number of the line in the file which is at fault.

Q: Why am I having trouble setting up my printer?

A: See the Handbook entry on printing for troubleshooting tips.

Q: How can I correct the keyboard mappings for my system?

A: Refer to the Handbook section on using localization, specifically the section on console setup.

Q: Why can I not get user quotas to work properly?

1. It is possible that the kernel is not configured to use quotas. In this case, add the following line to the kernel configuration file and recompile the kernel:

```
options QUOTA
```

Refer to the Handbook entry on quotas for full details.

2. Do not turn on quotas on `/`.
3. Put the quota file on the file system that the quotas are to be enforced on:

File System	Quota file
<code>/usr</code>	<code>/usr/admin/quotas</code>
<code>/home</code>	<code>/home/admin/quotas</code>
...	...

Q: Does OS support System V IPC primitives?

A: Yes, OS supports System V-style IPC, including shared memory, messages and semaphores, in the `GENERIC` kernel. With a custom kernel, support may be loaded with the `sysvshm.ko`, `sysvsem.ko` and `sysvmsg.ko` kernel modules, or enabled in the custom kernel by adding the following lines to the kernel configuration file:

```
options    SYSVSHM        # enable shared memory
options    SYSVSEM        # enable for semaphores
options    SYSVMSG        # enable for messaging
```

Recompile and install the kernel.

Q: What other mail-server software can I use instead of Sendmail?

A: The [Sendmail](#) server is the default mail-server software for OS, but it can be replaced with another MTA installed from the Ports Collection. Available ports include mail/exim, mail/postfix, and mail/qmail. Search the mailing lists for discussions regarding the advantages and disadvantages of the available MTAs.

Q: I have forgotten the root password! What do I do?

A: Do not panic! Restart the system, type `boot -s` at the `Boot :` prompt to enter single-user mode. At the question about the shell to use, hit Enter which will display a `PROMPT.ROOT` prompt. Enter `“mount`

`-urw /“` to remount the root file system read/write, then run

mount -a to remount all the file systems. Run “passwd root“ to change the root password then run `MAN.EXIT.1` to

continue booting.

Note

If you are still prompted to give the root password when entering the single-user mode, it means that the console has been marked as `insecure` in `/etc/ttys`. In this case, it will be required to boot from a OS installation disk, choose the Live CD or Shell at the beginning of the install process and issue the commands mentioned above. Mount the specific partition in this case and then `chroot` to it. For example, replace `“mount`

`-urw /“` with `mount /dev/ada0p1 /mnt; chroot /mnt` for a system on `ada0p1`.

Note

If the root partition cannot be mounted from single-user mode, it is possible that the partitions are encrypted and it is impossible to mount them without the access keys. For more information see the section about encrypted disks in the OS Handbook.

Q: How do I keep Control+Alt+Delete from rebooting the system?

A: When using `MAN.SYSCONS.4`, the default console driver, build and install a new kernel with this line in the configuration file:

```
options SC_DISABLE_REBOOT
```

This can also be done by setting the following `MAN.SYSCTL.8` which does not require a reboot or kernel recompile:

```
PROMPT.ROOT sysctl hw.syscons.kbd_reboot=0
```

****Note****

The above two methods are exclusive: The `MAN.SYSCTL.8` does not exist if the kernel is compiled with `“SC_DISABLE_REBOOT“`.

Q: How do I reformat DOS text files to UNIX ones?

A: Use this `MAN.PERL.1` command:

```
PROMPT.USER perl -i.bak -npe 's/\r\n/\n/g' file(s)
```

where `file(s)` is one or more files to process. The modification is done in-place, with the original file stored with a `.bak` extension.

Alternatively, use `MAN.TR.1`:

```
PROMPT.USER tr -d '\r' < dos-text-file > unix-file
```

`dos-text-file` is the file containing DOS text while `unix-file` will contain the converted output. This can be quite a bit faster than using `perl`.

Yet another way to reformat DOS text files is to use the converters/dosunix port from the Ports Collection. Consult its documentation about the details.

Q: How do I re-read `/etc/rc.conf` and re-start `/etc/rc` without a reboot?

A: Go into single-user mode and then back to multi-user mode:

```
PROMPT.ROOT shutdown now
PROMPT.ROOT return
PROMPT.ROOT exit
```

Q: I tried to update my system to the latest *-STABLE*, but got *-BETAx*, *-RC* or *-PRERELEASE*! What is going on?

A: Short answer: it is just a name. *RC* stands for “Release Candidate”. It signifies that a release is imminent. In OS, *-PRERELEASE* is typically synonymous with the code freeze before a release. (For some releases, the *-BETA* label was used in the same way as *-PRERELEASE*.)

Long answer: OS derives its releases from one of two places. Major, dot-zero, releases, such as 9.0-RELEASE are branched from the head of the development stream, commonly referred to as *-CURRENT*. Minor releases, such as 6.3-RELEASE or 5.2-RELEASE, have been snapshots of the active *-STABLE* branch. Starting with 4.3-RELEASE, each release also now has its own branch which can be tracked by people requiring an extremely conservative rate of development (typically only security advisories).

When a release is about to be made, the branch from which it will be derived from has to undergo a certain process. Part of this process is a code freeze. When a code freeze is initiated, the name of the branch is changed to reflect that it is about to become a release. For example, if the branch used to be called 6.2-STABLE, its name will be changed to 6.3-PRERELEASE to signify the code freeze and signify that extra pre-release testing should be happening. Bug fixes can still be committed to be part of the release. When the source code is in shape for the release the name will be changed to 6.3-RC to signify that a release is about to be made from it. Once in the RC stage, only the most critical bugs found can be fixed. Once the release (6.3-RELEASE in this example) and release branch have been made, the branch will be renamed to 6.3-STABLE.

For more information on version numbers and the various Subversion branches, refer to the Release Engineering article.

Q: I tried to install a new kernel, and the `MAN.CHFLAGS.1` failed. How do I get around this?

A: Short answer: the security level is greater than 0. Reboot directly to single-user mode to install the kernel.

Long answer: OS disallows changing system flags at security levels greater than 0. To check the current security level:

```
PROMPT.ROOT sysctl kern.securelevel
```

The security level cannot be lowered in multi-user mode, so boot to single-user mode to install the kernel, or change the security level in `/etc/rc.conf` then reboot. See the `MAN.INIT.8` manual page for details on `securelevel`, and see `/etc/defaults/rc.conf` and the `MAN.RC.CONF.5` manual page for more information on `rc.conf`.

Q: I cannot change the time on my system by more than one second! How do I get around this?

A: Short answer: the system is at a security level greater than 1. Reboot directly to single-user mode to change the date.

Long answer: OS disallows changing the time by more than one second at security levels greater than 1. To check the security level:

```
PROMPT.ROOT sysctl kern.securelevel
```

The security level cannot be lowered in multi-user mode. Either boot to single-user mode to change the date or change the security level in `/etc/rc.conf` and reboot. See the `MAN.INIT.8` manual page for details on `securelevel`, and see `/etc/defaults/rc.conf` and the `MAN.RC.CONF.5` manual page for more information on `rc.conf`.

Q: Why is `rpc.statd` using 256 MB of memory?

A: No, there is no memory leak, and it is not using 256 MB of memory. For convenience, `rpc.statd` maps an obscene amount of memory into its address space. There is nothing terribly wrong with this from a technical standpoint; it just throws off things like `MAN.TOP.1` and `MAN.PS.1`.

`MAN.RPC.STATD.8` maps its status file (resident on `/var`) into its address space; to save worrying about remapping it later when it needs to grow, it maps it with a generous size. This is very evident from the source code, where one can see that the length argument to `MAN.MMAP.2` is `0x10000000`, or one sixteenth of the address space on an IA32, or exactly 256 MB.

Q: Why can I not unset the `schg` file flag?

A: The system is running a `securelevel` greater than 0. Lower the `securelevel` and try again. For more information, see the *FAQ entry on `securelevel`* and the `MAN.INIT.8` manual page.

Q: What is `vnlr`?

A: `vnlr` flushes and frees `vnodes` when the system hits the `kern.maxvnodes` limit. This kernel thread sits mostly idle, and only activates when there is a huge amount of RAM and users are accessing tens of thousands of tiny files.

Q: What do the various memory states displayed by `top` mean?

- **Active:** pages recently statistically used.
- **Inactive:** pages recently statistically unused.
- **Cache:** (most often) pages that have percolated from inactive to a status where they maintain their data, but can often be immediately reused (either with their old association, or reused with a new association). There can be certain immediate transitions from `active` to `cache` state if the page is known to be clean (unmodified), but that transition is a matter of policy, depending upon the algorithm choice of the VM system maintainer.
- **Free:** pages without data content, and can be immediately used in certain circumstances where cache pages might be ineligible. Free pages can be reused at interrupt or process state.
- **Wired:** pages that are fixed into memory, usually for kernel purposes, but also sometimes for special use in processes.

Pages are most often written to disk (sort of a VM sync) when they are in the inactive state, but active pages can also be synced. This depends upon the CPU tracking of the modified bit being available, and in certain situations there can be an advantage for a block of VM pages to be synced, whether they are active or inactive. In most common cases, it is best to think of the inactive queue to be a queue of relatively unused pages that might or might not be in the process of being written to disk. Cached pages are already synced, not mapped, but available for immediate process use with their old association or with a new association. Free pages are available at interrupt level, but cached or free pages can be used at process state for reuse. Cache pages are not adequately locked to be available at interrupt level.

There are some other flags (e.g., busy flag or busy count) that might modify some of the described rules.

Q: How much free memory is available?

A: There are a couple of kinds of “free memory”. One kind is the amount of memory immediately available without paging anything else out. That is approximately the size of cache queue + size of free queue (with a derating factor, depending upon system tuning). Another kind of “free memory” is the total amount of VM space. That can be complex, but is dependent upon the amount of swap space and memory. Other kinds of “free memory” descriptions are also possible, but it is relatively useless to define these, but rather it is important to make sure that the paging rate is kept low, and to avoid running out of swap space.

Q: What is `/var/empty`?

A: `/var/empty` is a directory that the `MAN.SSHD.8` program uses when performing privilege separation. The `/var/empty` directory is empty, owned by root and has the `schg` flag set. This directory should not be deleted.

Q: I just changed `/etc/newsyslog.conf`. How can I check if it does what I expect?

A: To see what `MAN.NEWSYSLOG.8` will do, use the following:

```
PROMPT.USER newsyslog -nrvv
```

Q: My time is wrong, how can I change the timezone?

A: Use MAN.TZSETUP.8.

82.11 The X Window System and Virtual Consoles

Q: What is the X Window System?

A: The X Window System (commonly X11) is the most widely available windowing system capable of running on UNIX or UNIX like systems, including OS. The [X.Org Foundation](#) administers the [X protocol standards](#), with the current reference implementation, version 11 release XORG.VERSION, so references are often shortened to X11.

Many implementations are available for different architectures and operating systems. An implementation of the server-side code is properly known as an “X

server”.

Q: I want to run XORG, how do I go about it?

A: To install XORG do one of the following:

Use the x11/xorg meta-port, which builds and installs every XORG component.

Use x11/xorg-minimal, which builds and installs only the necessary XORG components.

Install XORG from OS packages:

```
PROMPT.ROOT pkg install xorg
```

After the installation of XORG, follow the instructions from the X11 Configuration section of the OS Handbook.

Q: I *tried* to run X, but I get a No devices detected. error when I type `startx`. What do I do now?

A: The system is probably running at a raised `securelevel`. It is not possible to start X at a raised `securelevel` because X requires write access to MAN.IO.4. For more information, see at the MAN.INIT.8 manual page.

There are two solutions to the problem: set the `securelevel` back down to zero or run MAN.XDM.1 (or an alternative display manager) at boot time before the `securelevel` is raised.

See ? for more information about running MAN.XDM.1 at boot time.

Q: Why does my mouse not work with X?

A: When using MAN.SYSCONS.4, the default console driver, OS can be configured to support a mouse pointer on each virtual screen. To avoid conflicting with X, MAN.SYSCONS.4 supports a virtual device called `/dev/sysmouse`. All mouse events received from the real mouse device are written to the MAN.SYSMOUSE.4 device via MAN.MOUSED.8. To use the mouse on one or more virtual consoles, *and* use X, see ? and set up MAN.MOUSED.8.

Then edit `/etc/X11/xorg.conf` and make sure the following lines exist:

```
Section "InputDevice"
    Option          "Protocol" "SysMouse"
    Option          "Device"  "/dev/sysmouse"
    . . . . .
```

Starting with XORG version 7.4, the `InputDevice` sections in `xorg.conf` are ignored in favor of autodetected devices. To restore the old behavior, add the following line to the `ServerLayout` or `ServerFlags` section:

```
Option "AutoAddDevices" "false"
```

Some people prefer to use `/dev/mouse` under X. To make this work, `/dev/mouse` should be linked to `/dev/sysmouse` (see `MAN.SYSMOUSE.4`) by adding the following line to `/etc/devfs.conf` (see `MAN.DEVFS.CONF.5`):

```
link    sysmouse    mouse
```

This link can be created by restarting `MAN.DEVFS.5` with the following command (as root):

```
PROMPT.ROOT service devfs restart
```

Q: My mouse has a fancy wheel. Can I use it in X?

A: Yes, if X is configured for a 5 button mouse. To do this, add the lines `Buttons 5` and `ZAxisMapping 4 5` to the “`InputDevice`” section of `/etc/X11/xorg.conf`, as seen in this example:

```
Section "InputDevice"
    Identifier      "Mouse1"
    Driver          "mouse"
    Option          "Protocol" "auto"
    Option          "Device"   "/dev/sysmouse"
    Option          "Buttons"  "5"
    Option          "ZAxisMapping" "4 5"
EndSection
```

To use the mouse in Emacs, also add the following lines to `~/.emacs`:

```
;; wheel mouse
(global-set-key [mouse-4] 'scroll-down)
(global-set-key [mouse-5] 'scroll-up)
```

Q: My laptop has a Synaptics touchpad. Can I use it in X?

A: Yes, after configuring a few things to make it work.

In order to use the Xorg synaptics driver, first remove `mouse_enable` from `rc.conf`.

To enable synaptics, add the following line to `/boot/loader.conf`:

```
hw.psm.synaptics_support="1"
```

Add the following to `/etc/X11/xorg.conf`:

```
Section "InputDevice"
Identifier  "Touchpad0"
Driver     "synaptics"
Option     "Protocol"  "psm"
Option     "Device"    "/dev/psm0"
EndSection
```

And be sure to add the following into the “`ServerLayout`” section:

```
InputDevice      "Touchpad0" "SendCoreEvents"
```

Q: How do I use remote X displays?

A: For security reasons, the default setting is to not allow a machine to remotely open a window.

To enable this feature, start X with the optional `-listen_tcp` argument:

```
PROMPT.USER startx -listen_tcp
```


Q: What is a virtual console and how do I make more?

A: Virtual consoles provide several simultaneous sessions on the same machine without doing anything complicated like setting up a network or running X.

When the system starts, it will display a login prompt on the monitor after displaying all the boot messages. Type in your login name and password to start working on the first virtual console.

To start another session, perhaps to look at documentation for a program or to read mail while waiting for an FTP transfer to finish, hold down Alt and press F2. This will display the login prompt for the second virtual console. To go back to the original session, press Alt+F1.

The default OS installation has eight virtual consoles enabled. Alt+F1, Alt+F2, Alt+F3, and so on will switch between these virtual consoles.

To enable more of virtual consoles, edit `/etc/ttys` (see MAN.TTYS.5) and add entries for `ttyv8` to `ttyvc`, after the comment on “Virtual terminals”:

```
# Edit the existing entry for ttyv8 in /etc/ttys and change
# "off" to "on".
ttyv8  "/usr/libexec/getty Pc"      xterm  on secure
ttyv9  "/usr/libexec/getty Pc"      xterm  on secure
ttyva  "/usr/libexec/getty Pc"      xterm  on secure
ttyvb  "/usr/libexec/getty Pc"      xterm  on secure
```

The more virtual terminals, the more resources that are used. This can be problematic on systems with 8 MB RAM or less. Consider changing `secure` to `insecure`.

Note

Versions of OS prior to 9.0 used the “cons25” terminal type, and not “xterm”. Use the format of existing entries in when adding entries to `/etc/ttys`.

Important

In order to run an X server, at least one virtual terminal must be left to `off` for it to use. This means that only eleven of the Alt-function keys can be used as virtual consoles so that one is left for the X server.

For example, to run X and eleven virtual consoles, the setting for virtual terminal 12 should be:

```
ttyvb  "/usr/libexec/getty Pc"      xterm  off secure
```

The easiest way to activate the virtual consoles is to reboot.

Q: How do I access the virtual consoles from X?

A: Use Ctrl+Alt+F to switch back to a virtual console. Press Ctrl+Alt+F1 to return to the first virtual console.

Once at a text console, use Alt+F to move between them.

To return to the X session, switch to the virtual console running X. If X was started from the command line using `startx`, the X session will attach to the next unused virtual console, not the text console from which it was invoked. For eight active virtual terminals, X will run on the ninth, so use Alt+F9.

Q: How do I start XDM on boot?

A: There are two schools of thought on how to start MAN.XDM.1. One school starts `xdm` from `/etc/ttys` (see MAN.TTYS.5) using the supplied example, while the other runs `xdm` from `rc.local` (see MAN.RC.8) or from an X script in `/usr/local/etc/rc.d`. Both are equally valid, and one may work in situations where the other does not. In both cases the result is the same: X will pop up a graphical login prompt.

The MAN.TTYS.5 method has the advantage of documenting which vty X will start on and passing the responsibility of restarting the X server on logout to MAN.INIT.8. The MAN.RC.8 method makes it easy to `kill xdm` if there is a problem starting the X server.

If loaded from MAN.RC.8, `xm` should be started without any arguments. `xm` must start *after* MAN.GETTY.8 runs, or else `getty` and `xm` will conflict, locking out the console. The best way around this is to have the script sleep 10 seconds or so then launch `xm`.

When starting `xm` from `/etc/ttys`, there still is a chance of conflict between `xm` and MAN.GETTY.8. One way to avoid this is to add the `vt` number in `/usr/local/lib/X11/xm/Xservers`:

```
:0 local /usr/local/bin/X vt4
```

The above example will direct the X server to run in `/dev/ttyv3`. Note the number is offset by one. The X server counts the `vt`y from one, whereas the OS kernel numbers the `vt`y from zero.

Q: Why do I get Couldn't open console when I run `xconsole`?

A: When X is started with `startx`, the permissions on `/dev/console` will *not* get changed, resulting in things like `xterm -C` and `xconsole` not working.

This is because of the way console permissions are set by default. On a multi-user system, one does not necessarily want just any user to be able to write on the system console. For users who are logging directly onto a machine with a VTY, the MAN.FBTAB.5 file exists to solve such problems.

In a nutshell, make sure an uncommented line of the form is in `/etc/fstab` (see MAN.FBTAB.5):

```
/dev/ttyv0 0600 /dev/console
```

It will ensure that whomever logs in on `/dev/ttyv0` will own the console.

Q: Why does my PS/2 mouse misbehave under X?

A: The mouse and the mouse driver may have become out of synchronization. In rare cases, the driver may also erroneously report synchronization errors:

```
psmintr: out of sync (xxxx != yyyy)
```

If this happens, disable the synchronization check code by setting the driver flags for the PS/2 mouse driver to `0x100`. This can be easiest achieved by adding `hint.psm.0.flags="0x100"` to `/boot/loader.conf` and rebooting.

Q: How do I reverse the mouse buttons?

A: Type `xmodmap -e "pointer = 3 2 1"`. Add this command to `~/.xinitrc` or `~/.xsession` to make it happen automatically.

Q: How do I install a splash screen and where do I find them?

A: The detailed answer for this question can be found in the Boot Time Splash Screens section of the OS Handbook.

Q: Can I use the Windows keys on my keyboard in X?

A: Yes. Use MAN.XMODMAP.1 to define which functions the keys should perform.

Assuming all Windows keyboards are standard, the keycodes for these three keys are the following:

- 115 — Windows key, between the left-hand Ctrl and Alt keys
- 116 — Windows key, to the right of AltGr
- 117 — Menu, to the left of the right-hand Ctrl

To have the left Windows key print a comma, try this.

```
PROMPT.ROOT xmodmap -e "keycode 115 = comma"
```

To have the Windows key-mappings enabled automatically every time X is started, either put the `xmodmap` commands in `~/.xinitrc` or, preferably, create a `~/.xmodmaprc` and include the `xmodmap` options, one per line, then add the following line to `~/.xinitrc`:

```
xmodmap $HOME/.xmodmaprc
```

For example, to map the 3 keys to be F13, F14, and F15, respectively. This would make it easy to map them to useful functions within applications or the window manager.

To do this, put the following in `~/.xmodmaprc`.

```
keycode 115 = F13
keycode 116 = F14
keycode 117 = F15
```

For the `x11-wm/fvwm2` desktop manager, one could map the keys so that F13 iconifies or de-iconifies the window the cursor is in, F14 brings the window the cursor is in to the front or, if it is already at the front, pushes it to the back, and F15 pops up the main Workplace menu even if the cursor is not on the desktop, which is useful when no part of the desktop is visible.

The following entries in `~/.fvwmrc` implement the aforementioned setup:

Key F13	FTIWS	A	Iconify
Key F14	FTIWS	A	RaiseLower
Key F15	A	A	Menu Workplace Nop

Q: How can I get 3D hardware acceleration for OpenGL?

A: The availability of 3D acceleration depends on the version of XORG and the type of video chip. For an nVidia chip, use the binary drivers provided for OS by installing one of the following ports:

The latest versions of nVidia cards are supported by the `x11/nvidia-driver` port.

Older drivers are available as `x11/nvidia-driver-###`

nVidia provides detailed information on which card is supported by which driver on their web site: http://www.nvidia.com/object/IO_32667.html.

For Matrox G200/G400, check the `x11-servers/mga_hal` port.

For ATI Rage 128 and Radeon see `MAN.ATI.4X`, `MAN.R128.4X` and `MAN.RADEON.4X`.

82.12 Networking

Q: Where can I get information on “diskless booting”?

A: “Diskless booting” means that the OS box is booted over a network, and reads the necessary files from a server instead of its hard disk. For full details, see the Handbook entry on diskless booting.

Q: Can a OS box be used as a dedicated network router?

A: Yes. Refer to the Handbook entry on advanced networking, specifically the section on routing and gateways.

Q: Can I connect my WINDOWS box to the Internet via OS?

A: Typically, people who ask this question have two PCs at home, one with OS and one with some version of WINDOWS the idea is to use the OS box to connect to the Internet and then be able to access the Internet from the WINDOWS box through the OS box. This is really just a special case of the previous question and works perfectly well.

Dialup users must use `-nat` and set `gateway_enable` to *YES* in `/etc/rc.conf`. For more information, refer to `MAN.PPP.8` or the Handbook entry on user PPP.

If the connection to the Internet is over Ethernet, use `MAN.NATD.8`. A tutorial can be found in the `natd` section of the Handbook.

Q: Does OS support PPP?

A: Yes. `MAN.PPP.8` provides support for both incoming and outgoing connections.

For more information on how to use this, refer to the Handbook chapter on PPP.

Q: Does OS support NAT or Masquerading?

A: Yes. For instructions on how to use NAT over a PPP connection, see the Handbook entry on PPP. To use NAT over some other sort of network connection, look at the `natd` section of the Handbook.

Q: How can I set up Ethernet aliases?

A: If the alias is on the same subnet as an address already configured on the interface, add `netmask 0xffffffff` to this command:

```
PROMPT.ROOT ifconfig ed0 alias 192.0.2.2 netmask 0xffffffff
```

Otherwise, specify the network address and netmask as usual:

```
PROMPT.ROOT ifconfig ed0 alias 172.16.141.5 netmask 0xffffffff00
```

More information can be found in the OS Handbook.

Q: Why can I not NFS-mount from a LINUX box?

A: Some versions of the LINUX NFS code only accept mount requests from a privileged port; try to issue the following command:

```
PROMPT.ROOT mount -o -P linuxbox:/blah /mnt
```

Q: Why does `mountd` keep telling me it can't change attributes and that I have a bad exports list on my OS NFS server?

A: The most frequent problem is not understanding the correct format of `/etc/exports`. Review `MAN.EXPORTS.5` and the NFS entry in the Handbook, especially the section on configuring NFS.

Q: How do I enable IP multicast support?

A: Install the `net/mrouted` package or port and add `mrouted_enable="YES"` to `/etc/rc.conf` start this service at boot time.

Q: Why do I have to use the FQDN for hosts on my site?

A: See the answer in the OS Handbook.

Q: Why do I get an error, Permission denied, for all networking operations?

A: If the kernel is compiled with the `IPFIREWALL` option, be aware that the default policy is to deny all packets that are not explicitly allowed.

If the firewall is unintentionally misconfigured, restore network operability by typing the following as root:

```
PROMPT.ROOT ipfw add 65534 allow all from any to any
```

Consider setting `firewall_type="open"` in `/etc/rc.conf`.

For further information on configuring this firewall, see the Handbook chapter.

Q: Why is my `ipfw` “fwd” rule to redirect a service to another machine not working?

A: Possibly because network address translation (NAT) is needed instead of just forwarding packets. A “fwd” rule only forwards packets, it does not actually change the data inside the packet. Consider this rule:

```
01000 fwd 10.0.0.1 from any to foo 21
```

When a packet with a destination address of foo arrives at the machine with this rule, the packet is forwarded to 10.0.0.1, but it still has the destination address of foo. The destination address of the packet is not changed to 10.0.0.1. Most machines would probably drop a packet that they receive with a destination address that is not their own. Therefore, using a “fwd” rule does not often work the way the user expects. This behavior is a feature and not a bug.

See the *FAQ about redirecting services*, the MAN.NATD.8 manual, or one of the several port redirecting utilities in the Ports Collection for a correct way to do this.

Q: How can I redirect service requests from one machine to another?

A: FTP and other service requests can be redirected with the sysutils/socket package or port. Replace the entry for the service in /etc/inetd.conf to call socket, as seen in this example for ftpd:

```
ftp stream tcp nowait nobody /usr/local/bin/socket socket ftp.example.com ftp
```

where ftp.example.com and ftp are the host and port to redirect to, respectively.

Q: Where can I get a bandwidth management tool?

A: There are three bandwidth management tools available for OS. MAN.DUMMYNET.4 is integrated into OS as part of MAN.IPFW.4. ALTQ has been integrated into OS as part of MAN.PF.4. Bandwidth Manager from [Emerging Technologies](#) is a commercial product.

Q: Why do I get /dev/bpf0: device not configured?

A: The running application requires the Berkeley Packet Filter (MAN.BPF.4), but it was removed from a custom kernel. Add this to the kernel config file and build a new kernel:

```
device bpf          # Berkeley Packet Filter
```

Q: How do I mount a disk from a WINDOWS machine that is on my network, like smbmount in LINUX?

A: Use the SMBFS toolset. It includes a set of kernel modifications and a set of userland programs. The programs and information are available as MAN.MOUNT.SMBFS.8 in the base system.

Q: What are these messages about: Limiting icmp/open port/closed port response in my log files?

A: This kernel message indicates that some activity is provoking it to send a large amount of ICMP or TCP reset (RST) responses. ICMP responses are often generated as a result of attempted connections to unused UDP ports. TCP resets are generated as a result of attempted connections to unopened TCP ports. Among others, these are the kinds of activities which may cause these messages:

- Brute-force denial of service (DoS) attacks (as opposed to single-packet attacks which exploit a specific vulnerability).
- Port scans which attempt to connect to a large number of ports (as opposed to only trying a few well-known ports).

The first number in the message indicates how many packets the kernel would have sent if the limit was not in place, and the second indicates the limit. This limit is controlled using net.inet.icmp.icmplim. This example sets the limit to 300 packets per second:

```
PROMPT.ROOT sysctl net.inet.icmp.icmplim=300
```

To disable these messages without disabling response limiting, use net.inet.icmp.icmplim_output to disable the output:

```
PROMPT.ROOT sysctl net.inet.icmp.icmplim_output=0
```

Finally, to disable response limiting completely, set `net.inet.icmp.icmplim` to 0. Disabling response limiting is discouraged for the reasons listed above.

Q: What are these arp: unknown hardware address format error messages?

A: This means that some device on the local Ethernet is using a MAC address in a format that OS does not recognize. This is probably caused by someone experimenting with an Ethernet card somewhere else on the network. This is most commonly seen on cable modem networks. It is harmless, and should not affect the performance of the OS system.

Q: Why do I keep seeing messages like: 192.168.0.10 is on fxp1 but got reply from 00:15:17:67:cf:82 on rl0, and how do I disable it?

A: Because a packet is coming from outside the network unexpectedly. To disable them, set `net.link.ether.inet.log_arp_wrong_iface` to 0.

82.13 Security

Q: What is a sandbox?

A: “Sandbox” is a security term. It can mean two things:

- A process which is placed inside a set of virtual walls that are designed to prevent someone who breaks into the process from being able to break into the wider system.

The process is be able to run inside the walls. Since nothing the process does in regards to executing code is supposed to be able to breach the walls, a detailed audit of its code is not needed in order to be able to say certain things about its security.

The walls might be a user ID, for example. This is the definition used in the `MAN.SECURITY.7` and `MAN.NAMED.8` man pages.

Take the `ntalk` service, for example (see `MAN.INETD.8`). This service used to run as user ID `root`. Now it runs as user ID `tty`. The `tty` user is a sandbox designed to make it more difficult for someone who has successfully hacked into the system via `ntalk` from being able to hack beyond that user ID.

- A process which is placed inside a simulation of the machine. It means that someone who is able to break into the process may believe that he can break into the wider machine but is, in fact, only breaking into a simulation of that machine and not modifying any real data.

The most common way to accomplish this is to build a simulated environment in a subdirectory and then run the processes in that directory chrooted so that `/` for that process is this directory, not the real `/` of the system).

Another common use is to mount an underlying file system read-only and then create a file system layer on top of it that gives a process a seemingly writeable view into that file system. The process may believe it is able to write to those files, but only the process sees the effects — other processes in the system do not, necessarily.

An attempt is made to make this sort of sandbox so transparent that the user (or hacker) does not realize that he is sitting in it.

UNIX implements two core sandboxes. One is at the process level, and one is at the `userid` level.

Every UNIX process is completely firewalled off from every other UNIX process. One process cannot modify the address space of another.

A UNIX process is owned by a particular `userid`. If the user ID is not the `root` user, it serves to firewall the process off from processes owned by other users. The user ID is also used to firewall off on-disk data.

Q: What is `securelevel`?

A: `securelevel` is a security mechanism implemented in the kernel. When the `securelevel` is positive, the kernel restricts certain tasks; not even the superuser (root) is allowed to do them. The `securelevel` mechanism limits the ability to:

- Unset certain file flags, such as `schg` (the system immutable flag).
- Write to kernel memory via `/dev/mem` and `/dev/kmem`.
- Load kernel modules.
- Alter firewall rules.

To check the status of the `securelevel` on a running system:

```
PROMPT.ROOT sysctl -n kern.securelevel
```

The output contains the current value of the `securelevel`. If it is greater than 0, at least some of the `securelevel`'s protections are enabled.

The `securelevel` of a running system cannot be lowered as this would defeat its purpose. If a task requires that the `securelevel` be non-positive, change the `kern_securelevel` and `kern_securelevel_enable` variables in `/etc/rc.conf` and reboot.

For more information on `securelevel` and the specific things all the levels do, consult `MAN.INIT.8`.

Warning

`Securelevel` is not a silver bullet; it has many known deficiencies. More often than not, it provides a false sense of security.

One of its biggest problems is that in order for it to be at all effective, all files used in the boot process until the `securelevel` is set must be protected. If an attacker can get the system to execute their code prior to the `securelevel` being set (which happens quite late in the boot process since some things the system must do at start-up cannot be done at an elevated `securelevel`), its protections are invalidated. While this task of protecting all files used in the boot process is not technically impossible, if it is achieved, system maintenance will become a nightmare since one would have to take the system down, at least to single-user mode, to modify a configuration file.

This point and others are often discussed on the mailing lists, particularly the `A.SECURITY`. Search the archives here for an extensive discussion. A more fine-grained mechanism is preferred.

Q: `BIND` (`named`) is listening on some high-numbered ports. What is going on?

A: `BIND` uses a random high-numbered port for outgoing queries. Recent versions of it choose a new, random UDP port for each query. This may cause problems for some network configurations, especially if a firewall blocks incoming UDP packets on particular ports. To get past that firewall, try the `avoid-v4-udp-ports` and `avoid-v6-udp-ports` options to avoid selecting random port numbers within a blocked range.

Warning

If a port number (like 53) is specified via the `query-source` or `query-source-v6` options in `/etc/namedb/named.conf`, randomized port selection will not be used. It is strongly recommended that these options not be used to specify fixed port numbers.

Congratulations, by the way. It is good practice to read `MAN.SOCKSTAT.1` output and notice odd things!

Q: The Sendmail daemon is listening on port 587 as well as the standard port 25! What is going on?

A: Recent versions of Sendmail support a mail submission feature that runs over port 587. This is not yet widely supported, but is growing in popularity.

Q: What is this UID 0 `toor` account? Have I been compromised?

A: Do not worry. `toor` is an “alternative” superuser account, where `toor` is root spelled backwards. It is intended to be used with a non-standard shell so the default shell for root does not need to change. This is important as shells which are

not part of the base distribution, but are instead installed from ports or packages, are installed in `/usr/local/bin` which, by default, resides on a different file system. If root's shell is located in `/usr/local/bin` and the file system containing `/usr/local/bin` is not mounted, root will not be able to log in to fix a problem and will have to reboot into single-user mode in order to enter the path to a shell.

Some people use `toor` for day-to-day root tasks with a non-standard shell, leaving root, with a standard shell, for single-user mode or emergencies. By default, a user cannot log in using `toor` as it does not have a password, so log in as root and set a password for `toor` before using it to login.

82.14 PPP

Q: I cannot make MAN.PPP.8 work. What am I doing wrong?

A: First, read MAN.PPP.8 and the PPP section of the Handbook. To assist in troubleshooting, enable logging with the following command:

```
set log Phase Chat Connect Carrier lcp ipcp ccp command
```

This command may be typed at the MAN.PPP.8 command prompt or it may be entered at the start of the default section in `/etc/ppp/ppp.conf`. Make sure that `/etc/syslog.conf` contains the lines below and the file `/var/log/ppp.log` exists:

```
!ppp
*. *      /var/log/ppp.log
```

A lot about what is going can be learned from the log file. Do not worry if it does not all make sense as it may make sense to someone else.

Q: Why does MAN.PPP.8 hang when I run it?

A: This is usually because the hostname will not resolve. The best way to fix this is to make sure that `/etc/hosts` is read first by the by ensuring that the `hosts` line is listed first in `/etc/host.conf`. Then, put an entry in `/etc/hosts` for the local machine. If there is no local network, change the `localhost` line:

```
127.0.0.1      foo.example.com foo localhost
```

Otherwise, add another entry for the host. Consult the relevant manual pages for more details.

When finished, verify that this command is successful: `ping -c1 `hostname``.

Q: Why will MAN.PPP.8 not dial in `-auto` mode?

A: First, check that a default route exists. This command should display two entries:

Destination	Gateway	Flags	Refs	Use	Netif	Expire
default	10.0.0.2	UGSc	0	0	tun0	
10.0.0.2	10.0.0.1	UH	0	0	tun0	

If a default route is not listed, make sure that the `HISADDR` line has been added to `/etc/ppp/ppp.conf`.

Another reason for the default route line being missing is that a default route has been added to `/etc/rc.conf` and this line is missing from `/etc/ppp/ppp.conf`:

```
delete ALL
```

If this is the case, go back to the Final System Configuration section of the Handbook.

Q: What does No route to host mean?

A: This error is usually because the following section is missing in `/etc/ppp/ppp.linkup`:


```
MYADDR:
  delete ALL
  add 0 0 HISADDR
```

This is only necessary for a dynamic IP address or when the address of the default gateway is unknown. When using interactive mode, the following can be typed in after entering packet mode. Packet mode is indicated by the capitalized PPP in the prompt:

```
delete ALL
add 0 0 HISADDR
```

Refer to the PPP and Dynamic IP addresses section of the Handbook for further details.

Q: Why does my connection drop after about 3 minutes?

A: The default PPP timeout is 3 minutes. This can be adjusted with the following line:

```
set timeout NNN
```

where NNN is the number of seconds of inactivity before the connection is closed. If NNN is zero, the connection is never closed due to a timeout. It is possible to put this command in `ppp.conf`, or to type it at the prompt in interactive mode. It is also possible to adjust it on the fly while the line is active by connecting to ppp's server socket using MAN.TELNET.1 or MAN.PPPCTL.8. Refer to the MAN.PPP.8 man page for further details.

Q: Why does my connection drop under heavy load?

A: If Link Quality Reporting (LQR) is configured, it is possible that too many LQR packets are lost between the OS system and the peer. MAN.PPP.8 deduces that the line must therefore be bad, and disconnects. LQR is disabled by default and can be enabled with the following line:

```
enable lqr
```

Q: Why does my connection drop after a random amount of time?

A: Sometimes, on a noisy phone line or even on a line with call waiting enabled, the modem may hang up because it incorrectly thinks that it lost carrier.

There is a setting on most modems for determining how tolerant it should be to temporary losses of carrier. Refer to the modem manual for details.

Q: Why does my connection hang after a random amount of time?

A: Many people experience hung connections with no apparent explanation. The first thing to establish is which side of the link is hung.

When using an external modem, try using MAN.PING.8 to see if the TD light is flashing when data is transmitted. If it flashes but the RD light does not, the problem is with the remote end. If TD does not flash, the problem is local. With an internal modem, use the “set

server“ command in `ppp.conf`. When the hang occurs,

connect to MAN.PPP.8 using MAN.PPPCTL.8. If the network connection suddenly revives due to the activity on the diagnostic socket, or if it will not connect but the `set socket` command succeeded at startup time, the problem is local. If it can connect but things are still hung, enable local logging with `set log local async` and use MAN.PING.8 from another window or terminal to make use of the link. The async logging will show the data being transmitted and received on the link. If data is going out and not coming back, the problem is remote.

Having established whether the problem is local or remote, there are now two possibilities:

- If the problem is remote, read on entry ?.
- If the problem is local, read on entry ?.

Q: The remote end is not responding. What can I do?

A: There is very little that can be done about this. Many ISPs will refuse to help users not running a MICROSOFT OS. Add `enable lqr` to `/etc/ppp/ppp.conf`, allowing MAN.PPP.8 to detect the remote failure and hang up. This detection is relatively slow and therefore not that useful.

First, try disabling all local compression by adding the following to the configuration:

```
disable predl deflate deflate24 protocomp acfcomp shortseq vj
deny predl deflate deflate24 protocomp acfcomp shortseq vj
```

Then reconnect to ensure that this makes no difference. If things improve or if the problem is solved completely, determine which setting makes the difference through trial and error. This is good information for the ISP, although it may make it apparent that it is not a MICROSOFT system.

Before contacting the ISP, enable async logging locally and wait until the connection hangs again. This may use up quite a bit of disk space. The last data read from the port may be of interest. It is usually ASCII data, and may even describe the problem (Memory fault, Core dumped).

If the ISP is helpful, they should be able to enable logging on their end, then when the next link drop occurs, they may be able to tell why their side is having a problem.

Q: MAN.PPP.8 has hung. What can I do?

A: In this case, rebuild MAN.PPP.8 with debugging information, and then use MAN.GDB.1 to grab a stack trace from the ppp process that is stuck. To rebuild the ppp utility with debugging information, type:

```
PROMPT.ROOT cd /usr/src/usr.sbin/ppp
PROMPT.ROOT env DEBUG_FLAGS='-g' make clean
PROMPT.ROOT env DEBUG_FLAGS='-g' make install
```

Then, restart ppp and wait until it hangs again. When the debug build of ppp hangs, start gdb on the stuck process by typing:

```
PROMPT.ROOT gdb ppp `pgrep ppp`
```

At the gdb prompt, use the `bt` or `where` commands to get a stack trace. Save the output of the gdb session, and “detach” from the running process by typing `quit`.

Q: I keep seeing errors about magic being the same. What does it mean?

A: Occasionally, just after connecting, there may be messages in the log that say Magic is same. Sometimes, these messages are harmless, and sometimes one side or the other exits. Most PPP implementations cannot survive this problem, and even if the link seems to come up, there will be repeated configure requests and configure acknowledgments in the log file until MAN.PPP.8 eventually gives up and closes the connection.

This normally happens on server machines with slow disks that are spawning a MAN.GETTY.8 on the port, and executing MAN.PPP.8 from a login script or program after login. There were reports of it happening consistently when using slirp. The reason is that in the time taken between MAN.GETTY.8 exiting and MAN.PPP.8 starting, the client-side MAN.PPP.8 starts sending Line Control Protocol (LCP) packets. Because ECHO is still switched on for the port on the server, the client MAN.PPP.8 sees these packets “reflect” back.

One part of the LCP negotiation is to establish a magic number for each side of the link so that “reflections” can be detected. The protocol says that when the peer tries to negotiate the same magic number, a NAK should be sent and a new magic number should be chosen. During the period that the server port has ECHO turned on, the client MAN.PPP.8 sends LCP packets, sees the same magic in the reflected packet and NAKs it. It also sees the NAK reflect (which also means MAN.PPP.8 must change its magic). This produces a potentially enormous number of magic number changes, all of which are happily piling into the server’s tty buffer. As soon as MAN.PPP.8 starts on the server, it is flooded with magic number changes and almost immediately decides it has tried enough to negotiate LCP and gives up. Meanwhile, the client, who no longer sees the reflections, becomes happy just in time to see a hangup from the server.

This can be avoided by allowing the peer to start negotiating with the following line in `ppp.conf`:

```
set openmode passive
```

This tells MAN.PPP.8 to wait for the server to initiate LCP negotiations. Some servers however may never initiate negotiations. In this case, try something like:

```
set openmode active 3
```

This tells MAN.PPP.8 to be passive for 3 seconds, and then to start sending LCP requests. If the peer starts sending requests during this period, MAN.PPP.8 will immediately respond rather than waiting for the full 3 second period.

Q: LCP negotiations continue until the connection is closed. What is wrong?

A: There is currently an implementation mis-feature in MAN.PPP.8 where it does not associate LCP, CCP & IPCP responses with their original requests. As a result, if one PPP implementation is more than 6 seconds slower than the other side, the other side will send two additional LCP configuration requests. This is fatal.

Consider two implementations, A and B. A starts sending LCP requests immediately after connecting and B takes 7 seconds to start. When B starts, A has sent 3 LCP REQs. We are assuming the line has ECHO switched off, otherwise we would see magic number problems as described in the previous section. B sends a REQ, then an ACK to the first of A's REQs. This results in A entering the OPENED state and sending an ACK (the first) back to B. In the meantime, B sends back two more ACKs in response to the two additional REQs sent by A before B started up. B then receives the first ACK from A and enters the OPENED state. A receives the second ACK from B and goes back to the REQ-SENT state, sending another (forth) REQ as per the RFC. It then receives the third ACK and enters the OPENED state. In the meantime, B receives the forth REQ from A, resulting in it reverting to the ACK-SENT state and sending another (second) REQ and (forth) ACK as per the RFC. A gets the REQ, goes into REQ-SENT and sends another REQ. It immediately receives the following ACK and enters OPENED.

This goes on until one side figures out that they are getting nowhere and gives up.

The best way to avoid this is to configure one side to be `passive` — that is, make one side wait for the other to start negotiating. This can be done with the following command:

```
set openmode passive
```

Care should be taken with this option. This command can also be used to limit the amount of time that MAN.PPP.8 waits for the peer to begin negotiations:

```
set stopped N
```

Alternatively, the following command (where N is the number of seconds to wait before starting negotiations) can be used:

```
set openmode active N
```

Check the manual page for details.

Q: Why does MAN.PPP.8 lock up when I shell out to test it?

A: When using `shell` or `!`, MAN.PPP.8 executes a shell or the passed arguments. The `ppp` program will wait for the command to complete before continuing. Any attempt to use the PPP link while running the command will appear as a frozen link. This is because MAN.PPP.8 is waiting for the command to complete.

To execute commands like this, use `!bg` instead. This will execute the given command in the background, and MAN.PPP.8 can continue to service the link.

Q: Why does MAN.PPP.8 over a null-modem cable never exit?

A: There is no way for MAN.PPP.8 to automatically determine that a direct connection has been dropped. This is due to the lines that are used in a null-modem serial cable. When using this sort of connection, LQR should always be enabled with the following line:

```
enable lqr
```

LQR is accepted by default if negotiated by the peer.

Q: Why does MAN.PPP.8 dial for no reason in `-auto` mode?

A: If MAN.PPP.8 is dialing unexpectedly, determine the cause, and set up dial filters to prevent such dialing.

To determine the cause, use the following line:

```
set log +tcp/ip
```

This will log all traffic through the connection. The next time the line comes up unexpectedly, the reason will be logged with a convenient timestamp next to it.

Next, disable dialing under these circumstances. Usually, this sort of problem arises due to DNS lookups. To prevent DNS lookups from establishing a connection (this will *not* prevent MAN.PPP.8 from passing the packets through an established connection), use the following:

```
set dfilter 1 deny udp src eq 53
set dfilter 2 deny udp dst eq 53
set dfilter 3 permit 0/0 0/0
```

This is not always suitable, as it will effectively break demand-dial capabilities. Most programs will need a DNS lookup before doing any other network related things.

In the DNS case, try to determine what is actually trying to resolve a host name. A lot of the time, Sendmail is the culprit. Make sure to configure Sendmail not to do any DNS lookups in its configuration file. See the section on using email with a dialup connection in the OS Handbook for details. You may also want to add the following line to `.mc`:

```
define(`confDELIVERY_MODE', `d')dnl
```

This will make Sendmail queue everything until the queue is run, usually, every 30 minutes, or until a “`sendmail -q`” is done, perhaps from `/etc/ppp/ppp.linkup`.

Q: What do these CCP errors mean?

A: I keep seeing the following errors in my log file:

```
CCP: CcpSendConfigReq
CCP: Received Terminate Ack (1) state = Req-Sent (6)
```

This is because MAN.PPP.8 is trying to negotiate Predictor1 compression, but the peer does not want to negotiate any compression at all. The messages are harmless, but can be silenced by disabling the compression:

```
disable pred1
```

Q: Why does MAN.PPP.8 not log my connection speed?

A: To log all lines of the modem conversation, enable the following:

```
set log +connect
```

This will make MAN.PPP.8 log everything up until the last requested “expect” string.

To see the connect speed when using PAP or CHAP, make sure to configure MAN.PPP.8 to expect the whole CONNECT line, using something like this:

```
set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 4 \
\\\" ATZ OK-ATZ-OK ATDT\\T TIMEOUT 60 CONNECT \\c \\n"
```

This gets the CONNECT, sends nothing, then expects a line-feed, forcing MAN.PPP.8 to read the whole CONNECT response.

Q: Why does MAN.PPP.8 ignore the \ character in my chat script?

A: The ppp utility parses each line in its configuration files so that it can interpret strings such as `set phone "123 456 789"` correctly and realize that the number is actually only one argument. To specify a " character, escape it using a backslash (\).

When the chat interpreter parses each argument, it re-interprets the argument to find any special escape sequences such as \P or \T. As a result of this double-parsing, remember to use the correct number of escapes.

To actually send a \ character, do something like:

```
set dial "\" ATZ OK-ATZ-OK AT\\X OK"
```

It will result in the following sequence:

```
ATZ
OK
AT\X
OK
```

Or:

```
set phone 1234567
set dial "\" ATZ OK ATDT\T"
```

It will result in the following sequence:

```
ATZ
OK
ATDT1234567
```

Q: What are FCS errors?

A: FCS stands for Frame Check Sequence. Each PPP packet has a checksum attached to ensure that the data being received is the data being sent. If the FCS of an incoming packet is incorrect, the packet is dropped and the HDLC FCS count is increased. The HDLC error values can be displayed using the `show hdlc` command.

If the link is bad or if the serial driver is dropping packets, it will produce the occasional FCS error. This is not usually worth worrying about although it does slow down the compression protocols substantially.

If the link freezes as soon as it connects and produces a large number of FCS errors, make sure the modem is not using software flow control (XON/XOFF). If the link must use software flow control, use `set accmap 0x000a0000` to tell MAN.PPP.8 to escape the ^Q and ^S characters.

Another reason for too many FCS errors may be that the remote end has stopped talking PPP. In this case, enable `async` logging to determine if the incoming data is actually a login or shell prompt. If it is a shell prompt at the remote end, it is possible to terminate MAN.PPP.8 without dropping the line by using `close lcp` followed by `term`) to reconnect to the shell on the remote machine.

If nothing in the log file indicates why the link was terminated, ask the remote administrator or ISP why the session was terminated.

Q: None of this helps — I am desperate! What can I do?

A: If all else fails, send the details of the error, the configuration files, how MAN.PPP.8 is being started, the relevant parts of the log file, and the output of `netstat -rn`, before and after connecting, to the A.QUESTIONS.

82.15 Serial Communications

This section answers common questions about serial communications with OS. PPP is covered in the [Networking](#) section.

Q: Which multi-port serial cards are supported by OS?

A: There is a list of these in the Serial Communications chapter of the Handbook.

Most multi-port PCI cards that are based on 16550 or clones are supported with no extra effort.

Some unnamed clone cards have also been known to work, especially those that claim to be AST compatible.

Check MAN.UART.4 and MAN.SIO.4 to get more information on configuring such cards.

Q: How do I get the boot: prompt to show on the serial console?

A: See this section of the Handbook.

Q: How do I tell if OS found my serial ports or modem cards?

A: As the OS kernel boots, it will probe for the serial ports for which the kernel is configured. Either watch the boot messages closely or run this command after the system is up and running:

```
PROMPT.USER dmesg | grep -E "^sio[0-9]"
sio0: <16550A-compatible COM port> port 0x3f8-0x3ff irq 4 flags 0x10 on acpi0
sio0: type 16550A
sio1: <16550A-compatible COM port> port 0x2f8-0x2ff irq 3 on acpi0
sio1: type 16550A
```

This example shows two serial ports. The first is on IRQ4, port address 0x3f8, and has a 16550A-type UART chip. The second uses the same kind of chip but is on IRQ3 and is at port address 0x2f8. Internal modem cards are treated just like serial ports, except that they always have a modem attached to the port.

The GENERIC kernel includes support for two serial ports using the same IRQ and port address settings in the above example. If these settings are not right for the system, or if there are more modem cards or serial ports than the kernel is configured for, reconfigure using the instructions in *building a kernel* for more details.

Q: How do I access the serial ports on OS?

A: The third serial port, sio2, or COM3, is on /dev/cuad2 for dial-out devices, and on /dev/ttyd2 for dial-in devices. What is the difference between these two classes of devices?

When opening /dev/ttydX in blocking mode, a process will wait for the corresponding cuadX device to become inactive, and then wait for the carrier detect line to go active. When the cuadX device is opened, it makes sure the serial port is not already in use by the ttydX device. If the port is available, it steals it from the ttydX device. Also, the cuadX device does not care about carrier detect. With this scheme and an auto-answer modem, remote users can log in and local users can still dial out with the same modem and the system will take care of all the conflicts.

Q: How do I enable support for a multi-port serial card?

A: The section on kernel configuration provides information about configuring the kernel. For a multi-port serial card, place an MAN.SIO.4 line for each serial port on the card in the MAN.DEVICE.HINTS.5 file. But place the IRQ specifiers on only one of the entries. All of the ports on the card should share one IRQ. For consistency, use the last serial port to specify the IRQ. Also, specify the following option in the kernel configuration file:

```
options COM_MULTIPORT
```

The following /boot/device.hints example is for an AST 4-port serial card on IRQ 12:

```
hint.sio.4.at="isa"
hint.sio.4.port="0x2a0"
hint.sio.4.flags="0x701"
```

```
hint.sio.5.at="isa"
hint.sio.5.port="0x2a8"
hint.sio.5.flags="0x701"
hint.sio.6.at="isa"
hint.sio.6.port="0x2b0"
hint.sio.6.flags="0x701"
hint.sio.7.at="isa"
hint.sio.7.port="0x2b8"
hint.sio.7.flags="0x701"
hint.sio.7.irq="12"
```

The flags indicate that the master port has minor number 7 (0x700), and all the ports share an IRQ (0x001).

Q: Can I set the default serial parameters for a port?

A: See the Serial Communications section in the OS Handbook.

Q: How can I enable dialup logins on my modem?

A: Refer to the section about Dial-in Services in the OS Handbook.

Q: How can I connect a dumb terminal to my OS box?

A: This information is in the Terminals section of the OS Handbook.

Q: Why can I not run `tip` or `cu`?

A: The built-in `MAN.TIP.1` and `MAN.CU.1` utilities can only access the `/var/spool/lock` directory via user `uucp` and group `dialer`. Use the dialer group to control who has access to the modem or remote systems by adding user accounts to `dialer`.

Alternatively, everyone can be configured to run `MAN.TIP.1` and `MAN.CU.1` by typing:

```
PROMPT.ROOT chmod 4511 /usr/bin/cu
PROMPT.ROOT chmod 4511 /usr/bin/tip
```

82.16 Miscellaneous Questions

Q: OS uses a lot of swap space even when the computer has free memory left. Why?

A: OS will proactively move entirely idle, unused pages of main memory into swap in order to make more main memory available for active use. This heavy use of swap is balanced by using the extra free memory for caching.

Note that while OS is proactive in this regard, it does not arbitrarily decide to swap pages when the system is truly idle. Thus, the system will not be all paged out after leaving it idle overnight.

Q: Why does `top` show very little free memory even when I have very few programs running?

A: The simple answer is that free memory is wasted memory. Any memory that programs do not actively allocate is used within the OS kernel as disk cache. The values shown by `MAN.TOP.1` labeled as `Inact`, `Cache`, and `Buf` are all cached data at different aging levels. This cached data means the system does not have to access a slow disk again for data it has accessed recently, thus increasing overall performance. In general, a low value shown for `Free` memory in `MAN.TOP.1` is good, provided it is not *very* low.

Q: Why will `chmod` not change the permissions on symlinks?

A: Symlinks do not have permissions, and by default, `MAN.CHMOD.1` will follow symlinks to change the permissions on the source file, if possible. For the file, `foo` with a symlink named `bar`, this command will always succeed.

```
PROMPT.USER chmod g-w bar
```

However, the permissions on `bar` will not have changed.

When changing modes of the file hierarchies rooted in the files instead of the files themselves, use either `-H` or `-L` together with `-R` to make this work. See `MAN.CHMOD.1` and `MAN.SYMLINK.7` for more information.

Warning

`-R` does a *recursive* `MAN.CHMOD.1`. Be careful about specifying directories or symlinks to directories to `MAN.CHMOD.1`. To change the permissions of a directory referenced by a symlink, use `MAN.CHMOD.1` without any options and follow the symlink with a trailing slash (`/`). For example, if `foo` is a symlink to directory `bar`, to change the permissions of `foo` (actually `bar`), do something like:

```
PROMPT.USER chmod 555 foo/
```

With the trailing slash, `MAN.CHMOD.1` will follow the symlink, `foo`, to change the permissions of the directory, `bar`.

Q: Can I run DOS binaries under OS?

A: Yes. A DOS emulation program, `emulators/doscmd`, is available in the OS Ports Collection.

If `doscmd` will not suffice, `emulators/pccemu` emulates an 8088 and enough BIOS services to run many DOS text-mode applications. It requires the X Window System.

The Ports Collection also has `emulators/dosbox`. The main focus of this application is emulating old DOS games using the local file system for files.

Q: What do I need to do to translate a OS document into my native language?

A: See the Translation FAQ in the OS Documentation Project Primer.

Q: Why does my email to any address at FreeBSD.org bounce?

A: The FreeBSD.org mail system implements some Postfix checks on incoming mail and rejects mail that is either from misconfigured relays or otherwise appears likely to be spam. Some of the specific requirements are:

- The IP address of the SMTP client must “reverse-resolve” to a forward confirmed hostname.
- The fully-qualified hostname given in the SMTP conversation (either HELO or EHLO) must resolve to the IP address of the client.

Other advice to help mail reach its destination include:

- Mail should be sent in plain text, and messages sent to mailing lists should generally be no more than 200KB in length.
- Avoid excessive cross posting. Choose *one* mailing list which seems most relevant and send it there.

If you still have trouble with email infrastructure at FreeBSD.org, send a note with the details to postmaster@freebsd.org; Include a date/time interval so that logs may be reviewed — and note that we only keep one week’s worth of mail logs. (Be sure to specify the time zone or offset from UTC.)

Q: Where can I find a free OS account?

A: While OS does not provide open access to any of their servers, others do provide open access UNIX systems. The charge varies and limited services may be available.

[Arboret, Inc.](#), also known as *M-Net*, has been providing open access to UNIX systems since 1983. Starting on an Altos running System III, the site switched to BSD/OS in 1991. In June of 2000, the site switched again to OS. *M-Net* can be accessed via telnet and SSH and provides basic access to the entire OS software suite. However, network access is limited to members and patrons who donate to the system, which is run as a non-profit organization. *M-Net* also provides an bulletin board system and interactive chat.

Q: What is the cute little red guy’s name?

A: He does not have one, and is just called “the BSD daemon”. If you insist upon using a name, call him “beastie”. Note that “beastie” is pronounced “BSD”.

More about the BSD daemon is available on his [home page](#).

Q: Can I use the BSD daemon image?

A: Perhaps. The BSD daemon is copyrighted by Marshall Kirk McKusick. Check his [Statement on the Use of the BSD Daemon Figure](#) for detailed usage terms.

In summary, the image can be used in a tasteful manner, for personal use, so long as appropriate credit is given. Before using the logo commercially, contact A.MCKUSICK.EMAIL for permission. More details are available on the [BSD Daemon’s home page](#).

Q: Do you have any BSD daemon images I could use?

A: Xfig and eps drawings are available under `/usr/share/examples/BSD_daemon/`.

Q: I have seen an acronym or other term on the mailing lists and I do not understand what it means. Where should I look?

A: Refer to the OS Glossary.

Q: Why should I care what color the bikeshed is?

A: The really, really short answer is that you should not. The somewhat longer answer is that just because you are capable of building a bikeshed does not mean you should stop others from building one just because you do not like the color they plan to paint it. This is a metaphor indicating that you need not argue about every little feature just because you know enough to do so. Some people have commented that the amount of noise generated by a change is inversely proportional to the complexity of the change.

The longer and more complete answer is that after a very long argument about whether MAN.SLEEP.1 should take fractional second arguments, A.PHK.EMAIL posted a long message entitled “[A bike shed \(any color will do\) on greener grass...](#)”. The appropriate portions of that message are quoted below.

“What is it about this bike shed?” Some of you have asked me.

It is a long story, or rather it is an old story, but it is quite short actually. C. Northcote Parkinson wrote a book in the early 1960s, called “Parkinson’s Law”, which contains a lot of insight into the dynamics of management.

[snip a bit of commentary on the book]

In the specific example involving the bike shed, the other vital component is an atomic power-plant, I guess that illustrates the age of the book.

Parkinson shows how you can go into the board of directors and get approval for building a multi-million or even billion dollar atomic power plant, but if you want to build a bike shed you will be tangled up in endless discussions.

Parkinson explains that this is because an atomic plant is so vast, so expensive and so complicated that people cannot grasp it, and rather than try, they fall back on the assumption that somebody else checked all the details before it got this far. Richard P. Feynmann gives a couple of interesting, and very much to the point, examples relating to Los Alamos in his books.

A bike shed on the other hand. Anyone can build one of those over a weekend, and still have time to watch the game on TV. So no matter how well prepared, no matter how reasonable you are with your proposal, somebody will seize the chance to show that he is doing his job, that he is paying attention, that he is *here*.

In Denmark we call it “setting your fingerprint”. It is about personal pride and prestige, it is about being able to point somewhere and say “There! *I* did that.” It is a strong trait in politicians, but present in most people given the chance. Just think about footsteps in wet cement.

—A.PHK.EMAIL on A.HACKERS.NAME, October 2, 1999

82.17 The OS Funnies

Q: How cool is OS?

A: Q. Has anyone done any temperature testing while running OS? I know LINUX runs cooler than DOS, but have never seen a mention of OS. It seems to run really hot.

A. No, but we have done numerous taste tests on blindfolded volunteers who have also had 250 micrograms of LSD-25 administered beforehand. 35% of the volunteers said that OS tasted sort of orange, whereas LINUX tasted like purple haze. Neither group mentioned any significant variances in temperature. We eventually had to throw the results of this survey out entirely anyway when we found that too many volunteers were wandering out of the room during the tests, thus skewing the results. We think most of the volunteers are at Apple now, working on their new “scratch and sniff” GUI. It is a funny old business we are in!

Seriously, OS uses the HLT (halt) instruction when the system is idle thus lowering its energy consumption and therefore the heat it generates. Also if you have ACPI (Advanced Configuration and Power Interface) configured, then OS can also put the CPU into a low power mode.

Q: Who is scratching in my memory banks??

A: Q. Is there anything “odd” that OS does when compiling the kernel which would cause the memory to make a scratchy sound? When compiling (and for a brief moment after recognizing the floppy drive upon startup, as well), a strange scratchy sound emanates from what appears to be the memory banks.

A. Yes! You will see frequent references to “daemons” in the BSD documentation, and what most people do not know is that this refers to genuine, non-corporeal entities that now possess your computer. The scratchy sound coming from your memory is actually high-pitched whispering exchanged among the daemons as they best decide how to deal with various system administration tasks.

If the noise gets to you, a good “fdisk -mbr” from DOS will get rid of them, but do not be surprised

if they react adversely and try to stop you. In fact, if at any point during the exercise you hear the satanic voice of Bill Gates coming from the built-in speaker, take off running and do not ever look back! Freed from the counterbalancing influence of the BSD daemons, the twin demons of DOS and WINDOWS are often able to re-assert total control over your machine to the eternal damnation of your soul. Now that you know, given a choice you would probably prefer to get used to the scratchy noises, no?

Q: How many OS hackers does it take to change a lightbulb?

A: One thousand, one hundred and sixty-nine:

Twenty-three to complain to -CURRENT about the lights being out;

Four to claim that it is a configuration problem, and that such matters really belong on -questions;

Three to submit PRs about it, one of which is misfiled under doc and consists only of “it’s dark”;

One to commit an untested lightbulb which breaks buildworld, then back it out five minutes later;

Eight to flame the PR originators for not including patches in their PRs;

Five to complain about buildworld being broken;

Thirty-one to answer that it works for them, and they must have updated at a bad time;

One to post a patch for a new lightbulb to -hackers;

One to complain that he had patches for this three years ago, but when he sent them to -CURRENT they were just ignored, and he has had bad experiences with the PR system; besides, the proposed new lightbulb is non-reflexive;

Thirty-seven to scream that lightbulbs do not belong in the base system, that committers have no right to do things like this without consulting the Community, and WHAT IS -CORE DOING ABOUT IT!?

Two hundred to complain about the color of the bicycle shed;

Three to point out that the patch breaks MAN.STYLE.9;

Seventeen to complain that the proposed new lightbulb is under GPL;

Five hundred and eighty-six to engage in a flame war about the comparative advantages of the GPL, the BSD license, the MIT license, the NPL, and the personal hygiene of unnamed FSF founders;

Seven to move various portions of the thread to -chat and -advocacy;

One to commit the suggested lightbulb, even though it shines dimmer than the old one;

Two to back it out with a furious flame of a commit message, arguing that OS is better off in the dark than with a dim lightbulb;

Forty-six to argue vociferously about the backing out of the dim lightbulb and demanding a statement from -core;

Eleven to request a smaller lightbulb so it will fit their Tamagotchi if we ever decide to port OS to that platform;

Seventy-three to complain about the SNR on -hackers and -chat and unsubscribe in protest;

Thirteen to post “unsubscribe”, “How do I unsubscribe?”, or “Please remove me from the list”, followed by the usual footer;

One to commit a working lightbulb while everybody is too busy flaming everybody else to notice;

Thirty-one to point out that the new lightbulb would shine 0.364% brighter if compiled with TenDRA (although it will have to be reshaped into a cube), and that OS should therefore switch to TenDRA instead of GCC;

One to complain that the new lightbulb lacks fairings;

Nine (including the PR originators) to ask “what is MFC?”;

Fifty-seven to complain about the lights being out two weeks after the bulb has been changed.

A.NIK.EMAIL adds:

I was laughing quite hard at this.

And then I thought, “Hang on, shouldn’t there be ‘I to document it.’ in that list somewhere?”

And then I was enlightened :-)

A.TABTHORPE.EMAIL says: “None, real OS hackers are not afraid of the dark!”

Q: Where does data written to `/dev/null` go?

A: It goes into a special data sink in the CPU where it is converted to heat which is vented through the heatsink / fan assembly. This is why CPU cooling is increasingly important; as people get used to faster processors, they become careless with their data and more and more of it ends up in `/dev/null`, overheating their CPUs. If you delete `/dev/null` (which effectively disables the CPU data sink) your CPU may run cooler but your system will quickly become constipated with all that excess data and start to behave erratically. If you have a fast network connection you can cool down your CPU by reading data out of `/dev/random` and sending it off somewhere; however you run the risk of overheating your network connection and / or angering your ISP, as most of the data will end up getting converted to heat by their equipment, but they generally have good cooling, so if you do not overdo it you should be OK.

Paul Robinson adds:

There are other methods. As every good sysadmin knows, it is part of standard practice to send data to the screen of interesting variety to keep all the pixies that make up your picture happy. Screen pixies (commonly mis-typed or re-named as “pixels”) are categorized by the type of hat they wear (red, green or blue) and will hide or appear (thereby showing the color of their hat) whenever they receive a little piece of food. Video cards turn data into pixie-food, and then send them to the pixies — the more expensive the card, the better the food, so the better behaved the pixies are. They also need constant stimulation — this is why screen savers exist.

To take your suggestions further, you could just throw the random data to console, thereby letting the pixies consume it. This causes no heat to be produced at all, keeps the pixies happy and gets rid of your data quite quickly, even if it does make things look a bit messy on your screen.

Incidentally, as an ex-admin of a large ISP who experienced many problems attempting to maintain a stable temperature in a server room, I would strongly discourage people sending the data they do not want out to the network. The fairies who do the packet switching and routing get annoyed by it as well.

Q: My colleague sits at the computer too much, how can I prank her?

A: Install games/sl and wait for her to mistype `sl` for `ls`.

82.18 Advanced Topics

Q: How can I learn more about OS's internals?

A: See the OS Architecture Handbook.

Additionally, much general UNIX knowledge is directly applicable to OS.

Q: How can I contribute to OS?

A: See the article on Contributing to OS for specific advice on how to do this. Assistance is more than welcome!

Q: What are snapshots and releases?

A: There are currently REL.NUMBRANCH active/semi-active branches in the OS [Subversion Repository](#). (Earlier branches are only changed very rarely, which is why there are only REL.NUMBRANCH active branches of development):

- REL3.RELENG AKA REL3.STABLE
- REL2.RELENG AKA REL2.STABLE
- REL.RELENG AKA REL.STABLE
- REL.HEAD.RELENG AKA *-CURRENT* AKA REL.HEAD

HEAD is not an actual branch tag. It is a symbolic constant for the current, non-branched development stream known as *-CURRENT*.

Right now, *-CURRENT* is the REL.HEAD.RELX development stream; the REL.STABLE branch, REL.RELENG, forked off from *-CURRENT* in REL.RELENGDATE and the REL2.STABLE branch, REL2.RELENG, forked off from *-CURRENT* in REL2.RELENGDATE.

Q: Can I follow *-CURRENT* with limited Internet access?

A: Yes, this can be done *without* downloading the whole source tree by using the CTM facility.

Q: I have written a kernel extension, who do I send it to?

A: Take a look at the article on Contributing to OS to learn how to submit code.

And thanks for the thought!

Q: How can I make the most of the data I see when my kernel panics?

A: Here is typical kernel panic:

```
Fatal trap 12: page fault while in kernel mode
fault virtual address  = 0x40
fault code             = supervisor read, page not present
instruction pointer    = 0x8:0xf014a7e5
stack pointer          = 0x10:0xf4ed6f24
```

```

frame pointer      = 0x10:0xf4ed6f28
code segment       = base 0x0, limit 0xffffffff, type 0x1b
                  = DPL 0, pres 1, def32 1, gran 1
processor eflags    = interrupt enabled, resume, IOPL = 0
current process     = 80 (mount)
interrupt mask      =
trap number         = 12
panic: page fault

```

This message is not enough. While the instruction pointer value is important, it is also configuration dependent as it varies depending on the kernel image. If it is a GENERIC kernel image from one of the snapshots, it is possible for somebody else to track down the offending function, but for a custom kernel, only you can tell us where the fault occurred.

To proceed:

Write down the instruction pointer value. Note that the 0x8: part at the beginning is not significant in this case: it is the 0xf0xxxxxx part that we want.

When the system reboots, do the following:

```
PROMPT.USER nm -n kernel.that.caused.the.panic | grep f0xxxxxx
```

where f0xxxxxx is the instruction pointer value. The odds are you will not get an exact match since the symbols in the kernel symbol table are for the entry points of functions and the instruction pointer address will be somewhere inside a function, not at the start. If you do not get an exact match, omit the last digit from the instruction pointer value and try again:

```
PROMPT.USER nm -n kernel.that.caused.the.panic | grep f0xxxxx
```

If that does not yield any results, chop off another digit. Repeat until there is some sort of output. The result will be a possible list of functions which caused the panic. This is a less than exact mechanism for tracking down the point of failure, but it is better than nothing.

However, the best way to track down the cause of a panic is by capturing a crash dump, then using MAN.KGDB.1 to generate a stack trace on the crash dump.

In any case, the method is this:

Make sure that the following line is included in the kernel configuration file:

```
makeoptions      DEBUG=-g          # Build kernel with gdb(1) debug symbols
```

Change to the /usr/src directory:

```
PROMPT.ROOT cd /usr/src
```

Compile the kernel:

```
PROMPT.ROOT make buildkernel KERNCONF=MYKERNEL
```

Wait for MAN.MAKE.1 to finish compiling.

```
PROMPT.ROOT make installkernel KERNCONF=MYKERNEL
```

Reboot.

Note

If KERNCONF is not included, the GENERIC kernel will instead be built and installed.

The MAN.MAKE.1 process will have built two kernels. `/usr/obj/usr/src/sys/MYKERNEL/kernel` and `/usr/obj/usr/src/sys/MYKERNEL/kernel.debug`. `kernel` was installed as `/boot/kernel/kernel`, while `kernel.debug` can be used as the source of debugging symbols for MAN.KGDB.1.

To capture a crash dump, edit `/etc/rc.conf` and set `dumpdev` to point to either the swap partition or `AUTO`. This will cause the MAN.RC.8 scripts to use the MAN.DUMPON.8 command to enable crash dumps. This command can also be run manually. After a panic, the crash dump can be recovered using MAN.SAVECORE.8; if `dumpdev` is set in `/etc/rc.conf`, the MAN.RC.8 scripts will run MAN.SAVECORE.8 automatically and put the crash dump in `/var/crash`.

Note

OS crash dumps are usually the same size as physical RAM. Therefore, make sure there is enough space in `/var/crash` to hold the dump. Alternatively, run MAN.SAVECORE.8 manually and have it recover the crash dump to another directory with more room. It is possible to limit the size of the crash dump by using “options

MAXMEM=N” where N is the size of kernel’s memory usage in

KBs. For example, for 1 GB of RAM, limit the kernel’s memory usage to 128 MB, so that the crash dump size will be 128 MB instead of 1 GB.

Once the crash dump has been recovered, get a stack trace as follows:

```
PROMPT.USER kgdb /usr/obj/usr/src/sys/MYKERNEL/kernel.debug /var/crash/vmcore.0
(kgdb) backtrace
```

Note that there may be several screens worth of information. Ideally, use MAN.SCRIPT.1 to capture all of them. Using the unstripped kernel image with all the debug symbols should show the exact line of kernel source code where the panic occurred. The stack trace is usually read from the bottom up to trace the exact sequence of events that lead to the crash. MAN.KGDB.1 can also be used to print out the contents of various variables or structures to examine the system state at the time of the crash.

Tip

If a second computer is available, MAN.KGDB.1 can be configured to do remote debugging, including setting breakpoints and single-stepping through the kernel code.

Note

If DDB is enabled and the kernel drops into the debugger, a panic and a crash dump can be forced by typing `panic` at the `ddb` prompt. It may stop in the debugger again during the panic phase. If it does, type `continue` and it will finish the crash dump.

Q: Why has `dlsym()` stopped working for ELF executables?

A: The ELF toolchain does not, by default, make the symbols defined in an executable visible to the dynamic linker. Consequently `dlsym()` searches on handles obtained from calls to `dlopen(NULL, flags)` will fail to find such symbols.

To search, using `dlsym()`, for symbols present in the main executable of a process, link the executable using the `--export-dynamic` option to the ELF linker (MAN.LD.1).

Q: How can I increase or reduce the kernel address space on i386?

A: By default, the kernel address space is 1 GB (2 GB for PAE) for i386. When running a network-intensive server or using ZFS, this will probably not be enough.

Add the following line to the kernel configuration file to increase available space and rebuild the kernel:

```
options KVA_PAGES=N
```

To find the correct value of N, divide the desired address space size (in megabytes) by four. (For example, it is 512 for 2 GB.)

82.19 Acknowledgments

This innocent little Frequently Asked Questions document has been written, rewritten, edited, folded, spindled, mutilated, eviscerated, contemplated, discombobulated, cogitated, regurgitated, rebuilt, castigated, and reinvigorated over the last decade, by a cast of hundreds if not thousands. Repeatedly.

We wish to thank every one of the people responsible, and we encourage you to join them in making this FAQ even better.

BIBLIOGRAPHY

FreeBSD Documentation Project Primer for New Contributors

Author The FreeBSD Documentation Project

83.1 Preface

83.1.1 Shell Prompts

This table shows the default system prompt and superuser prompt. The examples use these prompts to indicate which type of user is running the example.

User	Prompt
Normal user	PROMPT.USER
root	PROMPT.ROOT

83.1.2 Typographic Conventions

This table describes the typographic conventions used in this book.

Meaning	Examples
The names of commands.	Use <code>ls -l</code> to list all files.
The names of files.	Edit <code>.login</code> .
On-screen computer output.	You have mail.
What the user types, contrasted with on-screen computer output.	PROMPT.USER date +"The time is %H:%M" The time is 09:18
Manual page references.	Use MAN.SU.1 to change user identity.
User and group names.	Only root can do this.
Emphasis.	The user <i>must</i> do this.
Text that the user is expected to replace with the actual text.	To search for a keyword in the manual pages, type “<code>man -k keyword</code>”
Environment variables.	\$HOME is set to the user’s home directory.

83.1.3 Notes, Tips, Important Information, Warnings, and Examples

Notes, warnings, and examples appear within the text.

Note

Notes are represented like this, and contain information to take note of, as it may affect what the user does.

Tip

Tips are represented like this, and contain information helpful to the user, like showing an easier way to do something.

Important

Important information is represented like this. Typically, these show extra steps the user may need to take.

Warning

Warnings are represented like this, and contain information warning about possible damage if the instructions are not followed. This damage may be physical, to the hardware or the user, or it may be non-physical, such as the inadvertent deletion of important files.

Examples are represented like this, and typically contain examples showing a walkthrough, or the results of a particular action.

83.1.4 Acknowledgments

My thanks to Sue Blake, Patrick Durusau, Jon Hamilton, Peter Flynn, and Christopher Maden, who took the time to read early drafts of this document and offer many valuable comments and criticisms.

CHAP.OVERVIEW CHAP.TOOLS CHAP.WORKING-COPY CHAP.STRUCTURE CHAP.DOC-BUILD
CHAP.THE-WEBSITE CHAP.XML-PRIMER CHAP.XHTML-MARKUP CHAP.DOCBOOK-MARKUP
CHAP.STYLESHEETS CHAP.TRANSLATIONS CHAP.WRITING-STYLE CHAP.EDITOR-CONFIG CHAP.SEE-
ALSO APP.EXAMPLES

OS Porter's Handbook

Author The OS Documentation Project

CHAP.PORTING-WHY CHAP.NEW-PORT CHAP.QUICK-PORTING CHAP.SLOW-PORTING
 CHAP.MAKEFILES CHAP.SPECIAL CHAP.PLIST CHAP.PKG-FILES CHAP.TESTING CHAP.UPGRADING
 CHAP.SECURITY CHAP.PORTING-DADS CHAP.PORTING-SAMPLEM CHAP.KEEPING-UP CHAP.USES
 CHAP.VERSIONS

13 May 2005 2005 FreeBSD Mall, Inc.

- Overview of FreeBSD Documentation Architecture
- How does DocBook Slides fit in?
- Why is it useful?
- XSLT Tools
- XSL-FO
- Questions
- Documentation Set:
 - 40 articles.
 - 9 books.
 - Hundreds of man pages.
 - Available in a dozen languages.
- 157 developers have made a commit to doc/ or www/ in last 12 months.
- Books and articles authored in structured SGML/XML DocBook DTD.
- Robust makefile infrastructure allows one to build PDF, HTML, or ASCII output with simple 'make' command.
- Structured documentation format that allows one to specify semantics of a document rather than the presentation.
- Stylesheets take care of details such as always printing commands in a monospace font, etc. Stylesheets can make different presentation decisions based on the output format.
- For example, HTML stylesheets may use a CSS mouseover on acronyms so that the full technical term can be displayed.
- Links to online man pages can be automatically generated for <command> s in HTML output.
- DocBook is available as both an SGML DTD and as an XML Schema. For various reasons, most of our documentation is in SGML format, but can easily be converted to XML for processing with XML tools.

- Both SGML and XML allow include files, so we can share content across the release notes, all 40 articles, and 9 books in the documentation set.
- Since the content is structured, intelligent search engines could differentiate between, say, touch , the Unix Command, and touch , the feeling. Search engines, agents, and other information processing tools have information about the meaning of the content.
- Traditionally, we have used the Jade DSSSL Engine to convert the DocBook SGML files into HTML, text, and PostScript formats.
- Jade can output HTML directly with the help of the DSSSL stylesheets and extensive FreeBSD customizations. The text formats can then be converted from the HTML with the help of a text based web browser.
- For Print output, we use the TeX backend of Jade and then rely on TeX to generate the PostScript output.
- The makefiles handle all of this, so make FORMATS=html or make FORMATS=ps is all you really need to know about if you have all of the tools installed.
- The DSSSL stylesheet specification proved difficult to implement in practice, and so Jade was the only widely distributed implementation.
- The XSLT language has been much more widely adopted, and the goal is to eventually transition the FreeBSD Documentation Set to XML/XSLT.
- We already use XSLT extensively for the web site builds, and it possible to build parts of the documentation set with XSLT as well.
- The new slides infrastructure relies solely on XSLT and XSL-FO, without any DSSSL stylesheets.
- A DTD based on Simplified DocBook XML Schema.
- Provides a collection of tags useful for structuring content into slides, lists, and paragraphs for presentations.
- Also provides tags for describing technical content such as commands, variables, etc..
- Stylesheets can create HTML or PDF output by default.
- OpenOffice Impress output should also be possible.

The preamble:

```
<?xml version='1.0'?>
<!DOCTYPE slides SYSTEM
"/usr/.../schema/dtd/slides.dtd" [
<!ENTITY % freebsd SYSTEM
    "../.../share/xml/freebsd.ent">
%freebsd;
]>
```

Note that the default `freebsd.ent` file from the FreeBSD Documentation Project is brought in. This provides entities to represent the newest release of FreeBSD, the number of ports in the Ports Collection, etc.

```
<slides>
<slidesinfo>
  <title>DocBook Slides, XSLT, and XSL-FO</title>
  <titleabbrev>DocBook Slides</titleabbrev>

  <author>
    <firstname>Murray</firstname>
    <surname>Stokely</surname>
  </author>
  <pubdate>13 May 2005</pubdate>
  <copyright>
    <year>2005</year>
```

```
<holder>FreeBSD Mall, Inc.</holder>
</copyright>
</slidesinfo>
```

```
<foil><title>My Title</title>

<para>Creating slides is easy.</para>

<itemizedlist>
  <listitem>Point 1.</listitem>
  <listitem>Point 2.</listitem>
  <listitem>Point 3.</listitem>
  <listitem>Point 4.</listitem>
</itemizedlist>

<para>
  There are &os.num; ports in &rel.current;.
</para>
</foil>
```

- As with other SGML/XML document types in the tree, DocBook slides are supported by a robust Makefile infrastructure to allow the building of HTML or PDF output.
- Additional options exist to use different XSLT or XSL-FO processors, or to specify alternate stylesheets.
- FORMATS=openoffice support would be really cool. Any takers?
- All of the relevant software is installed with the textproc/docproj port.

```
$ make USE_XEP=1 FORMATS=pdf
```

An XSLT processor is required to transform the source XML document for both HTML or PDF output. The Open Source `xsltproc` processor is fastest and supports the basics necessary for using the Slides DTD.

Other alternatives

- Saxon
- XT (James Clark)
- Xalan (Apache XML Project)
- MSXSLT
- See ports collection.

Used extensively in the `www/` tree, but for various reasons we haven't fully migrated to building the FreeBSD `doc/` tree with it.

Stylesheets can be layered to arbitrary depth. Templates at layer N take precedence over those at level N-1.

```
<xsl:template match="command">
  <tt><xsl:value-of select="."></tt>
</xsl:template>
```

Numerous examples in the `doc/` of layered stylesheets built on top of default DocBook XSLT stylesheets.

The default XSL stylesheets for HTML output use chunking to generate one HTML page per foil with navigation icons to allow easy hypertext browsing of presentations.

!image0!

- Great for advocacy: Allows the FreeBSD Project to get “more mileage” out of presentations by providing a space to archive them in HTML form on <http://www.FreeBSD.org>.

- More amenable to searching.
- Recent presentations can be indexed together rather than visiting different personal homepages or conference websites looking for most recent FreeBSD presentations.
- Text formats easier for translation teams to work with.
- Easier for documentation teams to find relevant information from HTML slides and incorporate them into the Handbook and other FreeBSD documentation sources.
- For print output, the XSLT processor transforms the XML document into an intermediate XSL-FO document.
- XSL-FO is an XML format that contains detailed presentation information about the content.
- There are many open source and commercial XSL-FO processors.
 - FOP (Apache XML Project)
 - RenderX XEP (commercial)
 - PassiveTeX

What does `file.fo` look like?

XSL-FO files not really meant for hand-generation, but you asked for it :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="my-page">
      <fo:region-body margin="1in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
```

The layout-master-set contains one or more declarations of page masters and page sequence masters elements that define layouts of single pages and page sequences. In the example, the area should have a 1 inch margin from all sides of the page.

```
<fo:page-sequence master-reference="my-page">
  <fo:flow flow-name="xsl-region-body">
    <fo:block>Hello, world!</fo:block>
  </fo:flow>
</fo:page-sequence>
</fo:root>
```

- Pages in the document are grouped into sequences; each sequence starts from a new page.
- Flow is the container object for all user text in the document. Everything contained in the flow will be formatted into regions on pages generated inside the page sequence.
- `fo:block` objects roughly correspond to `<DIV>` in HTML, and normally include a paragraph of text.

An open source implementation based on TeX similar to the JadeTeX macro package used for creating print output from DSSSL stylesheets.

Pros:

- Open source.

Cons:

- Does not implement many features of XSL-FO that are difficult to implement in TeX, such as background images. (The daemon in the background of this slide would be impossible to produce with PassiveTeX).
- Requires a full TeX installation rather than generating PDF directly. Difficult to debug.

Open source implementation in Java from the Apache Project.

Pros:

- Open source.
- Handles some implementation features that PassiveTeX does not.

Cons:

- Not as conformant as the commercial processors. Many features are missing.
- A Work in Progress.
- This presentation will be committed to CVS later today so that it appears on the FreeBSD web site as an example.
- Implement OpenOffice Impress output format.
- Make better use of XML Documentation sources such as release notes to create more dynamic slides.
- Add more stylesheet options for PDF output.
- Find/write better open source XSL-FO toolchain.
- DocBook SourceForge Project Page
- doc/share/mk/doc.slides.mk
- docbook-apps@ mailing list.
- freebsd-doc@ mailing list.
- This example presentation.
- If all else fails, send me an email and I'd be happy to help you design a presentation in DocBook Slides.

Thursday, 3 Jan 2008 2004-2008 FreeBSD Mall, Inc.

- What is FreeBSD?
- Who uses FreeBSD?
- FreeBSD Development Model
- FreeBSD Release / Branch Terminology
- Recent FreeBSD Releases

SLIDES.WHAT-IS-FREEBSD SLIDES.FREEBSD-USERS SLIDES.FREEBSD-DEV-MODEL
SLIDES.FREEBSD-ORGANIZATION SLIDES.FREEBSD-RELEASE-PROCESS SLIDES.FREEBSD-RECENT-
RELEASES SLIDES.FREEBSD-MORE-INFORMATION

Indices and tables

- `genindex`
- `modindex`
- `search`