

# **FreeBSD Handbook**

**The FreeBSD Documentation Project**

## **FreeBSD Handbook**

by The FreeBSD Documentation Project

Published February 1999

Copyright © 1995, 1996, 1997, 1998, 1999 by The FreeBSD Documentation Project, FreeBSD Inc.

Welcome to FreeBSD! This handbook covers the installation and day to day use of *FreeBSD Release 3.1*. This manual is a *work in progress* and is the work of many individuals. Many sections do not yet exist and some of those that do exist need to be updated. If you are interested in helping with this project, send email to the FreeBSD documentation project mailing list <freebsd-doc@FreeBSD.ORG>. The latest version of this document is always available from the FreeBSD World Wide Web server (<http://www.FreeBSD.ORG/>). It may also be downloaded in plain text (handbook.latin1), postscript (handbook.ps) or HTML (handbook-html.tar.gz) with HTTP or gzip'd from the FreeBSD FTP server (<ftp://ftp.FreeBSD.ORG/pub/FreeBSD/doc/>) or one of the numerous mirror sites. You may also want to Search the Handbook (<http://www.FreeBSD.ORG/search.html>).

# Table of Contents

<b>I. Getting Started .....</b>	<b>24</b>
1. Introduction.....	25
FreeBSD in a Nutshell.....	25
A Brief History of FreeBSD.....	27
FreeBSD Project Goals.....	29
The FreeBSD Development Model .....	29
About the Current Release.....	31
2. Installing FreeBSD .....	33
Supported Configurations.....	35
Disk Controllers .....	35
Ethernet cards.....	37
Miscellaneous devices .....	39
Preparing for the Installation .....	39
Before installing from CDROM .....	39
Before installing from Floppy.....	40
Before installing from a MS-DOS partition.....	41
Before installing from QIC/SCSI Tape.....	41
Before installing over a network.....	42
Preparing for NFS installation.....	43
Preparing for FTP Installation.....	43
Installing FreeBSD .....	44
MS-DOS User's Questions and Answers .....	45
3. Unix Basics.....	47
The Online Manual.....	47
GNU Info Files.....	48
4. Installing Applications: The Ports collection .....	49
Why Have a Ports Collection?.....	49
How Does the Ports Collection Work?.....	50
Getting a FreeBSD Port.....	52
Compiling ports from CDROM .....	52
Compiling ports from the Internet .....	52
Skeletons.....	54
Makefile .....	54
The files directory.....	55
The patches directory.....	55
The pkg directory .....	55
What to do when a port does not work.....	56
Some Questions and Answers .....	56

Making a port yourself .....	62
Quick Porting .....	62
Writing the Makefile .....	62
Writing the description files .....	63
COMMENT .....	63
DESCR.....	63
PLIST.....	64
Creating the checksum file .....	64
Testing the port.....	64
Checking your port with portlint .....	65
Submitting the port.....	65
Slow Porting.....	66
How things work .....	66
Getting the original sources.....	67
Modifying the port .....	68
Patching .....	68
Configuring.....	69
Handling user input .....	69
Configuring the Makefile .....	69
The original source.....	69
DISTNAME.....	70
PKGNAME .....	70
CATEGORIES .....	70
MASTER_SITES .....	71
PATCHFILES .....	71
MAINTAINER .....	72
Dependencies .....	72
LIB_DEPENDS.....	72
RUN_DEPENDS.....	73
BUILD_DEPENDS .....	73
FETCH_DEPENDS .....	73
DEPENDS .....	74
Common dependency variables .....	74
Notes on dependencies .....	74
Building mechanisms .....	75
Special considerations.....	75
ldconfig.....	75
ELF support .....	76
Moving a.out libraries out of the way .....	76
Format .....	76

PORTOBJFORMAT.....	77
Building shared libraries .....	77
LIB_DEPENDS .....	77
PLIST.....	77
ldconfig.....	78
MASTERDIR .....	78
Shared library versions .....	79
Manpages .....	80
Ports that require Motif.....	80
REQUIRES_MOTIF.....	81
MOTIFLIB.....	81
X11 fonts.....	81
Info files .....	81
The pkg/ subdirectory.....	86
MESSAGE.....	86
INSTALL.....	86
REQ .....	87
Changing PLIST based on make variables .....	87
Changing the names of files in the pkg subdirectory.....	88
Licensing Problems.....	88
Upgrading .....	89
Do's and Dont's.....	89
Strip Binaries.....	90
INSTALL_* macros .....	90
WRKDIR .....	90
WRKDIRPREFIX.....	91
Differentiating operating systems and OS versions .....	91
Writing something after bsd.port.mk.....	94
Install additional documentation .....	95
DIST_SUBDIR.....	96
Package information.....	96
RCS strings.....	96
Recursive diff .....	96
PREFIX.....	97
Subdirectories.....	97
Cleaning up empty directories.....	97
UIDs .....	98
Do things rationally .....	99
Respect CFLAGS .....	99
Configuration files .....	99

Portlint.....	99
Feedback.....	99
Miscellanea.....	99
If you are stuck... ..	100
A Sample Makefile .....	100
Package Names .....	102
Categories .....	103
Current list of categories .....	104
Choosing the right category .....	106
Changes to this document and the ports system .....	107
That is It, Folks! .....	107
<b>II. System Administration.....</b>	<b>108</b>
5. Configuring the FreeBSD Kernel.....	109
Why Build a Custom Kernel?.....	109
Building and Installing a Custom Kernel .....	109
The Configuration File .....	111
Mandatory Keywords.....	111
General Options .....	113
Filesystem Options .....	114
Basic Controllers and Devices .....	116
SCSI Device Support .....	117
Console, Bus Mouse, and X Server Support.....	119
Serial and Parallel Ports .....	120
Networking .....	121
Sound cards .....	124
Pseudo-devices.....	125
Joystick, PC Speaker, Miscellaneous.....	126
Making Device Nodes .....	127
If Something Goes Wrong.....	128
6. Security .....	130
DES, MD5, and Crypt .....	130
Recognizing your crypt mechanism .....	130
S/Key .....	131
Secure connection initialization .....	132
Insecure connection initialization .....	133
Diversion: a login prompt.....	133
Generating a single one-time password .....	134
Generating multiple one-time passwords.....	135
Restricting use of UNIX passwords.....	135
Kerberos.....	136

Creating the initial database.....	136
Making it all run.....	138
Creating the server file .....	139
Populating the database.....	140
Testing it all out .....	141
Adding <code>su</code> privileges.....	142
Using other commands .....	144
Firewalls .....	144
What is a firewall?.....	145
Packet filtering routers.....	145
Proxy servers .....	146
What does IPFW allow me to do? .....	146
Enabling IPFW on FreeBSD.....	146
Configuring IPFW.....	147
Altering the IPFW rules .....	148
Listing the IPFW rules .....	151
Flushing the IPFW rules.....	152
Clearing the IPFW packet counters.....	152
Example commands for <code>ipfw</code> .....	152
Building a packet filtering firewall.....	153
7. Printing.....	155
What the Spooler Does .....	155
Why You Should Use the Spooler .....	155
Setting Up the Spooling System.....	156
Simple Printer Setup.....	156
Hardware Setup.....	156
Ports and Cables .....	157
Parallel Ports.....	157
Serial Ports .....	158
Software Setup .....	158
Kernel Configuration .....	159
Adding <code>/dev</code> Entries for the Ports.....	159
Setting the Communication Mode for the Parallel Port.....	160
Checking Printer Communications.....	161
Checking a Parallel Printer.....	162
Checking a Serial Printer .....	162
Enabling the Spooler: The <code>/etc/printcap</code> File .....	163
Naming the Printer.....	164
Suppressing Header Pages .....	165
Making the Spooling Directory .....	166

Identifying the Printer Device.....	167
Configuring Spooler Communication Parameters .....	167
Installing the Text Filter.....	169
Trying It Out .....	170
Troubleshooting.....	170
Using Printers .....	174
Printing Jobs .....	175
Checking Jobs .....	176
Removing Jobs.....	177
Beyond Plain Text: Printing Options .....	178
Formatting and Conversion Options.....	178
Job Handling Options.....	180
Header Page Options .....	181
Administrating Printers .....	181
Advanced Printer Setup .....	183
Filters .....	184
How Filters Work .....	185
Accommodating Plain Text Jobs on PostScript Printers .....	187
Simulating PostScript on Non-PostScript Printers.....	188
Conversion Filters .....	190
Why Install Conversion Filters? .....	190
Which Conversions Filters Should I Install? .....	190
Installing Conversion Filters.....	191
More Conversion Filter Examples .....	192
Automated Conversion: An Alternative To Conversion Filters.....	195
Output Filters.....	196
lpf: a Text Filter.....	197
Header Pages.....	197
Enabling Header Pages.....	198
Controlling Header Pages.....	199
Accounting for Header Pages.....	200
Header Pages on PostScript Printers .....	201
Networked Printing .....	204
Printers Installed on Remote Hosts .....	205
Printers with Networked Data Stream Interfaces .....	207
Restricting Printer Usage .....	208
Restricting Multiple Copies .....	208
Restricting Access To Printers .....	209
Controlling Sizes of Jobs Submitted .....	210
Restricting Jobs from Remote Printers.....	211

Accounting for Printer Usage .....	213
Quick and Dirty Printer Accounting .....	213
How Can You Count Pages Printed? .....	216
Alternatives to the Standard Spooler .....	216
Acknowledgments .....	217
8. Disks .....	219
Using sysinstall.....	219
Using command line utilities .....	220
* Using Slices .....	220
Dedicated .....	220
* Non-traditional Drives .....	220
* Zip Drives .....	221
* Jazz Drives .....	221
* Sequest Drives.....	221
9. Backups.....	222
* What about backups to floppies?.....	222
Tape Media .....	222
4mm (DDS: Digital Data Storage).....	222
8mm (Exabyte).....	223
QIC.....	223
* Mini-Cartridge .....	224
DLT .....	224
Using a new tape for the first time .....	224
Backup Programs.....	225
Dump and Restore.....	225
Tar .....	225
Cpio.....	226
Pax.....	226
Amanda .....	226
Do nothing .....	226
Which Backup Program is Best? .....	227
Emergency Restore Procedure .....	227
Before the Disaster .....	227
After the Disaster .....	231
* I did not prepare for the Disaster, What Now?.....	232
10. Disk Quotas.....	233
Configuring Your System to Enable Disk Quotas .....	233
Setting Quota Limits.....	234
Checking Quota Limits and Disk Usage .....	235
* Quotas over NFS .....	236

11. The X Window System .....	237
12. PC Hardware compatibility.....	238
Resources on the Internet .....	238
Sample Configurations .....	238
Jordan's Picks.....	238
Motherboards .....	239
Disk Controllers .....	239
Disk drives.....	240
CDROM drives.....	240
CD Recordable (WORM) drives .....	240
Tape drives.....	240
Video Cards .....	241
Monitors .....	241
Networking.....	241
Serial.....	242
Audio .....	242
Video .....	242
Core/Processing.....	242
Motherboards, busses, and chipsets .....	243
* ISA .....	243
* EISA.....	242
* VLB .....	243
PCI.....	243
CPUs/FPUs .....	244
P6 class (Pentium Pro/Pentium II) .....	244
Pentium class.....	244
Clock speeds .....	245
The AMD K6 Bug .....	246
* 486 class .....	246
* 386 class .....	246
286 class .....	246
* Memory.....	246
* BIOS .....	246
Input/Output Devices.....	246
* Video cards.....	247
* Sound cards.....	247
Serial ports and multiport cards .....	247
The UART: What it is and how it works .....	247
Synchronous Serial Transmission.....	247
Asynchronous Serial Transmission .....	248

Other UART Functions .....	249
The RS232-C and V.24 Standards .....	249
RS232-C Bit Assignments (Marks and Spaces) .....	249
RS232-C Break Signal .....	250
RS232-C DTE and DCE Devices .....	250
RS232-C Pin Assignments .....	251
Bits, Baud and Symbols.....	253
The IBM Personal Computer UART .....	254
National Semiconductor UART Family Tree.....	255
The NS16550AF and the PC16550D are the same thing.....	256
National Semiconductor Part Numbering System .....	257
Other Vendors and Similar UARTs .....	257
8250/16450/16550 Registers .....	260
Beyond the 16550A UART.....	261
Configuring the <code>sio</code> driver.....	262
Digi International (DigiBoard) PC/8 .....	262
Boca 16.....	263
Configuring the <code>cy</code> driver .....	265
* Parallel ports .....	266
* Modems.....	266
* Network cards .....	266
* Keyboards .....	266
* Mice .....	266
* Other .....	266
Storage Devices .....	266
Using ESDI hard disks .....	267
Concepts of ESDI.....	267
Physical connections.....	267
Device addressing .....	267
Termination.....	268
Using ESDI disks with FreeBSD .....	268
ESDI speed variants.....	268
Stay on track .....	268
Hard or soft sectoring .....	269
Low level formatting.....	269
Translations.....	269
Spare sectoring.....	270
Bad block handling .....	271
Kernel configuration .....	271
Particulars on ESDI hardware .....	272

Adaptec 2320 controllers.....	272
Western Digital WD1007 controllers .....	272
Ultrastor U14F controllers.....	272
Further reading .....	272
Thanks to.....	273
What is SCSI?.....	273
Components of SCSI.....	274
SCSI bus types.....	275
Single ended buses.....	275
Differential buses .....	276
Terminators .....	276
Terminator power.....	278
Device addressing .....	278
Bus layout .....	279
Using SCSI with FreeBSD.....	279
About translations, BIOSes and magic.....	280
SCSI subsystem design.....	281
Kernel configuration .....	281
Tuning your SCSI kernel setup.....	284
Rogue SCSI devices.....	284
Multiple LUN devices .....	285
Tagged command queueing .....	286
Busmaster host adapters .....	286
Tracking down problems .....	286
Further reading .....	287
* Disk/tape controllers .....	289
* SCSI .....	289
* IDE .....	289
* Floppy.....	289
Hard drives .....	289
SCSI hard drives.....	289
Rotational speed.....	289
Form factor .....	290
Interface .....	290
* IDE hard drives.....	291
Tape drives .....	291
General tape access commands .....	291
Controller Interfaces.....	291
SCSI drives.....	292
4mm (DAT: Digital Audio Tape).....	292

8mm (Exabyte) .....	292
QIC (Quarter-Inch Cartridge) .....	292
DLT (Digital Linear Tape) .....	293
Mini-Cartridge .....	293
Autoloaders/Changers .....	293
* IDE drives .....	293
Floppy drives .....	293
* Parallel port drives .....	293
Detailed Information .....	293
Archive Anaconda 2750 .....	293
Archive Python .....	294
Archive Viper 60 .....	295
Archive Viper 150 .....	295
Archive Viper 2525 .....	295
Conner 420R .....	296
Conner CTMS 3200 .....	296
DEC TZ87	
( <a href="http://www.digital.com/info/Customer-Update/931206004.txt.html">http://www.digital.com/info/Customer-Update/931206004.txt.html</a> )	
296	
Exabyte EXB-2501	
( <a href="http://www.Exabyte.COM:80/Products/Minicartridge/2501/Rfeatures.html">http://www.Exabyte.COM:80/Products/Minicartridge/2501/Rfeatures.html</a> )	
297	
Exabyte EXB-8200 .....	297
Exabyte EXB-8500 .....	298
Exabyte EXB-8505	
( <a href="http://www.Exabyte.COM:80/Products/8mm/8505XL/Rfeatures.html">http://www.Exabyte.COM:80/Products/8mm/8505XL/Rfeatures.html</a> )	
298	
Hewlett-Packard HP C1533A .....	298
Hewlett-Packard HP 1534A .....	299
Hewlett-Packard HP C1553A Autoloading DDS2 .....	300
Hewlett-Packard HP 35450A .....	301
Hewlett-Packard HP 35470A .....	301
Hewlett-Packard HP 35480A .....	302
Sony SDT-5000	
( <a href="http://www.sel.sony.com/SEL/ccpg/storage/tape/t5000.html">http://www.sel.sony.com/SEL/ccpg/storage/tape/t5000.html</a> ) .....	302
Tandberg TDC 3600 .....	303
Tandberg TDC 3620 .....	303
Tandberg TDC 4222 .....	303
Wangtek 5525ES .....	304
Wangtek 6200 .....	304

* Problem drives.....	304
CD-ROM drives .....	304
* Other .....	305
* Other .....	305
* PCMCIA .....	305
13. Localization .....	306
Russian Language (KOI8-R encoding) .....	306
Console Setup .....	306
Locale Setup.....	306
Login Class Method .....	307
How to do it with vipw(8).....	307
How to do it with adduser(8) .....	307
How to do it with pw(8).....	308
Shell Startup Files Method .....	308
Printer Setup.....	308
MSDOS FS and Russian file names .....	308
X Window Setup.....	309
German Language (ISO 8859-1).....	310
<b>III. Network Communications.....</b>	<b>311</b>
14. Serial Communications.....	312
Serial Basics .....	312
Terminals .....	312
Uses and Types of Terminals .....	313
Dumb Terminals .....	313
PCs Acting As Terminals .....	314
X Terminals .....	314
Cables and Ports.....	314
Cables.....	314
Null-modem cables .....	314
Standard RS-232C Cables .....	315
Ports.....	315
Kinds of Ports .....	315
Port Names.....	316
Configuration .....	316
Adding an Entry to /etc/ttys.....	317
Specifying the <i>getty</i> Type .....	317
Specifying the Default Terminal Type .....	318
Enabling the Port.....	319
Specifying Secure Ports .....	319
Force init to Reread /etc/ttys .....	320

Debugging your connection.....	320
Dialin Service .....	321
Prerequisites.....	321
FreeBSD Version.....	322
Terminology .....	322
External vs. Internal Modems .....	322
Modems and Cables .....	323
Serial Interface Considerations .....	324
Quick Overview .....	324
Kernel Configuration .....	324
Device Special Files.....	326
Making Device Special Files.....	326
Configuration Files .....	327
/etc/gettytab.....	327
Locked-Speed Config .....	328
Matching-Speed Config.....	328
/etc/ttys .....	329
Locked-Speed Config .....	330
Matching-Speed Config.....	330
/etc/rc.serial or /etc/rc.local .....	331
Modem Settings .....	331
Locked-speed Config.....	332
Matching-speed Config .....	333
Checking the Modem's Configuration .....	333
Troubleshooting .....	333
Checking out the FreeBSD system.....	333
Try Dialing In.....	334
Acknowledgments.....	335
Dialout Service .....	335
Why cannot I run <code>tip</code> or <code>cu</code> ?.....	335
My stock Hayes modem is not supported, what can I do?.....	336
How am I expected to enter these AT commands? .....	336
The @ sign for the pn capability does not work! .....	337
How can I dial a phone number on the command line?.....	337
Do I have to type in the bps rate every time I do that? .....	337
I access a number of hosts through a terminal server. ....	337
Can <code>tip</code> try more than one line for each site? .....	338
Why do I have to hit CTRL+P twice to send CTRL+P once?.....	338
Suddenly everything I type is in UPPER CASE?? .....	339
How can I do file transfers with <code>tip</code> ?.....	339

How can I run zmodem with <code>tip</code> ?	339
15. PPP and SLIP	340
Setting up User PPP	340
Before you start	340
Building a ppp ready kernel	341
Check the tun device	342
Name Resolution Configuration	343
Edit the <code>/etc/host.conf</code> file	343
Edit the <code>/etc/hosts(5)</code> file	343
Edit the <code>/etc/resolv.conf</code> file	344
ppp Configuration	344
PPP and Static IP addresses	344
PPP and Dynamic IP addresses	346
Receiving incoming calls with <code>ppp</code>	348
Which <code>getty</code> ?	348
PPP permissions	348
Setting up a PPP shell for dynamic-IP users	349
Setting up a PPP shell for static-IP users	350
Setting up <code>ppp.conf</code> for dynamic-IP users	350
Setting up <code>ppp.conf</code> for static-IP users	350
More on <code>mgetty</code> , AutoPPP, and MS extensions	351
<code>mgetty</code> and AutoPPP	351
MS extentions	352
PAP and CHAP authentication	352
Changing your <code>ppp</code> configuration on the fly	353
Final system configuration	354
Summary	355
Acknowledgments	356
Setting up Kernel PPP	356
Working as a PPP client	357
Working as a PPP server	360
Setting up a SLIP Client	365
Things you have to do only once	365
Making a SLIP connection	367
How to shutdown the connection	367
Troubleshooting	368
Setting up a SLIP Server	369
Prerequisites	369
Quick Overview	370
An Example of a SLIP Server Login	370

Kernel Configuration .....	370
Sliplogin Configuration.....	371
slip.hosts Configuration .....	372
slip.login Configuration .....	373
slip.logout Configuration .....	374
Routing Considerations.....	375
Static Routes.....	375
Running gated .....	375
Acknowledgments.....	377
16. Advanced Networking .....	378
Gateways and Routes.....	378
An example .....	378
Default routes.....	380
Dual homed hosts.....	381
Routing propagation.....	381
Troubleshooting .....	382
NFS.....	382
Diskless Operation.....	383
Setup Instructions.....	384
Using Shared / and /usr filesystems .....	386
Compiling netboot for specific setups.....	386
ISDN.....	386
ISDN Cards .....	387
ISDN Terminal Adapters .....	387
Standalone ISDN Bridges/Routers .....	388
17. Electronic Mail .....	391
Basic Information .....	391
User program .....	391
Mailhost Server Daemon .....	391
DNS — Name Service .....	391
POP Servers .....	392
Configuration .....	392
Basic.....	392
Mail for your Domain (Network). .....	393
Setting up UUCP.....	394
FAQ.....	397
Why do I have to use the FQDN for hosts on my site? .....	397
Sendmail says mail loops back to myself .....	398
How can I do E-Mail with a dialup PPP host?.....	398
<b>IV. Advanced topics.....</b>	<b>400</b>

18. The Cutting Edge: FreeBSD-current and FreeBSD-stable.....	401
Staying Current with FreeBSD.....	401
What is FreeBSD-current?.....	401
Who needs FreeBSD-current?.....	401
What is FreeBSD-current <i>not</i> ?.....	401
Using FreeBSD-current.....	402
Staying Stable with FreeBSD.....	403
What is FreeBSD-stable?.....	403
Who needs FreeBSD-stable?.....	403
Using FreeBSD-stable.....	404
Synchronizing Source Trees over the Internet.....	405
Anonymous CVS.....	406
Introduction.....	406
Using Anonymous CVS.....	407
Examples.....	408
Other Resources.....	409
<b>CTM</b> .....	409
Why should I use <b>CTM</b> ?.....	410
What do I need to use <b>CTM</b> ?.....	410
Starting off with <b>CTM</b> for the first time.....	411
Using <b>CTM</b> in your daily life.....	411
Keeping your local changes.....	412
Other interesting <b>CTM</b> options.....	412
Finding out exactly what would be touched by an update.....	412
Making backups before updating.....	412
Restricting the files touched by an update.....	413
Future plans for <b>CTM</b> .....	413
Miscellaneous stuff.....	413
Thanks!.....	413
<b>CVSup</b> .....	414
Introduction.....	414
Installation.....	415
Configuration.....	416
Running <b>CVSup</b> .....	421
<b>CVSup</b> File Collections.....	422
For more information.....	429
Using <code>make world</code> to rebuild your system.....	429
19. Contributing to FreeBSD.....	431
What Is Needed.....	431
High priority tasks.....	431

Medium priority tasks .....	433
Low priority tasks .....	434
Smaller tasks .....	434
How to Contribute .....	435
Bug reports and general commentary .....	435
Changes to the documentation .....	436
Changes to existing source code .....	436
New code or major value-added packages .....	437
Money, Hardware or Internet access .....	438
Donating funds .....	439
Donating hardware .....	439
Donating Internet access .....	440
Donors Gallery .....	440
Core Team Alumnus .....	443
Derived Software Contributors .....	443
Additional FreeBSD Contributors .....	444
386BSD Patch Kit Patch Contributors .....	479
20. Source Tree Guidelines and Policies .....	483
MAINTAINER on Makefiles .....	483
Contributed Software .....	483
Shared Libraries .....	486
21. Adding New Kernel Configuration Options .....	488
What's a <i>Kernel Option</i> , Anyway? .....	488
Now What Do I Have to Do for it? .....	489
22. Kernel Debugging .....	491
Debugging a Kernel Crash Dump with <i>kgdb</i> .....	491
Debugging a crash dump with <i>DDD</i> .....	494
Post-mortem Analysis of a Dump .....	495
On-line Kernel Debugging Using <i>DDB</i> .....	495
On-line Kernel Debugging Using Remote <i>GDB</i> .....	499
Debugging a Console Driver .....	500
23. Linux Emulation .....	501
How to Install the Linux Emulator .....	501
Installing Linux Emulation in 2.1-STABLE .....	501
Installing Linux Emulation in 2.2.2-RELEASE and later .....	502
Installing Linux Runtime Libraries .....	503
Installing using the <i>linux_lib</i> port .....	503
Installing libraries manually .....	503
How to install additional shared libraries .....	504
Configuring the <i>ld.so</i> — for FreeBSD 2.2-RELEASE and later .....	505

Installing Linux ELF binaries .....	507
Configuring the host name resolver .....	507
Finding the necessary files .....	508
How to Install Mathematica on FreeBSD.....	509
Unpacking the Mathematica distribution.....	510
Obtaining your Mathematica Password.....	510
Bugs .....	512
Acknowledgments.....	512
24. FreeBSD Internals.....	513
The FreeBSD Booting Process.....	513
Loading a kernel .....	513
Determine the root filesystem .....	513
Initialize user-land things.....	514
Interesting combinations.....	514
PC Memory Utilization .....	515
DMA: What it Is and How it Works.....	516
A Sample DMA transfer .....	517
DMA Page Registers and 16Meg address space limitations.....	519
DMA Operational Modes and Settings.....	520
Programming the DMA .....	522
DMA Port Map .....	523
0x00–0x1f DMA Controller #1 (Channels 0, 1, 2 and 3) .....	523
0xc0–0xdf DMA Controller #2 (Channels 4, 5, 6 and 7).....	524
0x80–0x9f DMA Page Registers.....	525
0x400–0x4ff 82374 Enhanced DMA Registers .....	526
The FreeBSD VM System.....	530
Management of physical memory—vm_page_t .....	530
The unified buffer cache—vm_object_t.....	531
Filesystem I/O—struct buf .....	531
Mapping Page Tables - vm_map_t, vm_entry_t.....	532
KVM Memory Mapping.....	532
Tuning the FreeBSD VM system.....	532
<b>V. Appendices.....</b>	<b>535</b>
25. Obtaining FreeBSD.....	536
CD-ROM Publishers.....	536
FTP Sites .....	536
CTM Sites.....	543
CVSup Sites.....	544
AFS Sites .....	549
26. Bibliography .....	550

Books & Magazines Specific to FreeBSD.....	550
Users' Guides .....	550
Administrators' Guides .....	551
Programmers' Guides .....	551
Operating System Internals .....	552
Security Reference.....	553
Hardware Reference .....	553
UNIX History .....	554
Magazines and Journals.....	554
27. Resources on the Internet.....	556
Mailing lists .....	556
List summary .....	556
How to subscribe.....	558
List charters.....	559
Usenet newsgroups .....	565
BSD specific newsgroups .....	565
Other Unix newsgroups of interest .....	566
X Window System .....	566
World Wide Web servers .....	567
28. FreeBSD Project Staff.....	569
The FreeBSD Core Team .....	569
The FreeBSD Developers .....	569
The FreeBSD Documentation Project.....	574
Who Is Responsible for What.....	575
29. PGP keys.....	578
Officers .....	578
FreeBSD Security Officer <security-officer@freebsd.org>.....	578
Warner Losh <imp@FreeBSD.ORG>.....	578
Core Team members.....	579
Satoshi Asami <asami@FreeBSD.ORG> .....	579
Jonathan M. Bresler <jmb@FreeBSD.ORG> .....	580
Andrey A. Chernov <ache@FreeBSD.ORG>.....	581
Jordan K. Hubbard <jkh@FreeBSD.ORG> .....	582
Poul-Henning Kamp <phk@FreeBSD.ORG> .....	583
Rich Murphey <rich@FreeBSD.ORG> .....	584
John Polstra <jdp@FreeBSD.ORG>.....	584
Guido van Rooij <guido@FreeBSD.ORG>.....	585
Peter Wemm <peter@FreeBSD.ORG>.....	586
Jörg Wunsch <joerg@FreeBSD.ORG>.....	587
Developers.....	588

Wolfram Schneider <wosch@FreeBSD.ORG> .....	588
Brian Somers <brian@FreeBSD.ORG> .....	589

# List of Examples

16-1. Branch office or Home network .....	389
16-2. Head office or other lan .....	389
18-1. Checking out something from -current (ls(1)) and deleting it again:.....	409
18-2. Checking out the version of ls(1) in the 2.2-stable branch: .....	409
18-3. Creating a list of changes (as unidiffs) to ls(1) between FreeBSD 2.2.2 and FreeBSD 2.2.6: .....	409
18-4. Finding out what other module names can be used: .....	409

# **I. Getting Started**

# Chapter 1. Introduction

FreeBSD is a 4.4BSD-Lite based operating system for Intel architecture (x86) based PCs. For an overview of FreeBSD, see FreeBSD in a nutshell. For a history of the project, read a brief history of FreeBSD. To see a description of the latest release, read about the current release. If you're interested in contributing something to the FreeBSD project (code, equipment, sacks of unmarked bills), please see about contributing to FreeBSD.

## FreeBSD in a Nutshell

FreeBSD is a state of the art operating system for personal computers based on the Intel CPU architecture, which includes the 386, 486 and Pentium processors (both SX and DX versions). Intel compatible CPUs from AMD and Cyrix are supported as well. FreeBSD provides you with many advanced features previously available only on much more expensive computers. These features include:

- *Preemptive multitasking* with dynamic priority adjustment to ensure smooth and fair sharing of the computer between applications and users.
- *Multuser* access means that many people can use a FreeBSD system simultaneously for a variety of things. System peripherals such as printers and tape drives are also properly SHARED BETWEEN ALL users on the system.
- Complete *TCP/IP networking* including SLIP, PPP, NFS and NIS support. This means that your FreeBSD machine can inter-operate easily with other systems as well act as an enterprise server, providing vital functions such as NFS (remote file access) and e-mail services or putting your organization on the Internet with WWW, ftp, routing and firewall (security) services.
- *Memory protection* ensures that applications (or users) cannot interfere with each other. One application crashing will not affect others in any way.
- FreeBSD is a *32-bit* operating system and was designed as such from the ground up.
- The industry standard *X Window System (X11R6)* provides a graphical user interface (GUI) for the cost of a common VGA card and monitor and comes with full sources.
- *Binary compatibility* with many programs built for SCO, BSDI, NetBSD, Linux and 386BSD.
- Hundreds of *ready-to-run* applications are available from the FreeBSD *ports* and *packages* collection. Why search the net when you can find it all right here?
- Thousands of additional and *easy-to-port* applications available on the Internet. FreeBSD is source code compatible with most popular commercial Unix systems and thus most applications require few, if any, changes to compile.

- Demand paged *virtual memory* and “merged VM/buffer cache” design efficiently satisfies applications with large appetites for memory while still maintaining interactive response to other users.
- *Shared libraries* (the Unix equivalent of MS-Windows DLLs) provide for efficient use of disk space and memory.
- A full complement of *C*, *C++* and *Fortran* development tools. Many additional languages for advanced research and development are also available in the ports and packages collection.
- *Source code* for the entire system means you have the greatest degree of control over your environment. Why be locked into a proprietary solution and at the mercy of your vendor when you can have a truly Open System?
- Extensive *on-line documentation*.
- *And many more!*

FreeBSD is based on the 4.4BSD-Lite release from Computer Systems Research Group (CSRG) at the University of California at Berkeley, and carries on the distinguished tradition of BSD systems development. In addition to the fine work provided by CSRG, the FreeBSD Project has put in many thousands of hours in fine tuning the system for maximum performance and reliability in real-life load situations. As many of the commercial giants struggle to field PC operating systems with such features, performance and reliability, FreeBSD can offer them *now!*

The applications to which FreeBSD can be put are truly limited only by your own imagination. From software development to factory automation, inventory control to azimuth correction of remote satellite antennae; if it can be done with a commercial UNIX product then it is more than likely that you can do it with FreeBSD, too! FreeBSD also benefits significantly from the literally thousands of high quality applications developed by research centers and universities around the world, often available at little to no cost. Commercial applications are also available and appearing in greater numbers every day.

Because the source code for FreeBSD itself is generally available, the system can also be customized to an almost unheard of degree for special applications or projects, and in ways not generally possible with operating systems from most major commercial vendors. Here is just a sampling of some of the applications in which people are currently using FreeBSD:

- *Internet Services*: The robust TCP/IP networking built into FreeBSD makes it an ideal platform for a variety of Internet services such as:
  - FTP servers
  - World Wide Web servers
  - Gopher servers
  - Electronic Mail servers
  - USENET News

- Bulletin Board Systems
- And more...

You can easily start out small with an inexpensive 386 class PC and upgrade as your enterprise grows.

- *Education:* Are you a student of computer science or a related engineering field? There is no better way of learning about operating systems, computer architecture and networking than the hands on, under the hood experience that FreeBSD can provide. A number of freely available CAD, mathematical and graphic design packages also make it highly useful to those whose primary interest in a computer is to get *other* work done!
- *Research:* With source code for the entire system available, FreeBSD is an excellent platform for research in operating systems as well as other branches of computer science. FreeBSD's freely available nature also makes it possible for remote groups to collaborate on ideas or shared development without having to worry about special licensing agreements or limitations on what may be discussed in open forums.
- *Networking:* Need a new router? A name server (DNS)? A firewall to keep people out of your internal network? FreeBSD can easily turn that unused 386 or 486 PC sitting in the corner into an advanced router with sophisticated packet filtering capabilities.
- *X Window workstation:* FreeBSD is a fine choice for an inexpensive X terminal solution, either using the freely available XFree86 server or one of the excellent commercial servers provided by X Inside. Unlike an X terminal, FreeBSD allows many applications to be run locally, if desired, thus relieving the burden on a central server. FreeBSD can even boot "diskless", making individual workstations even cheaper and easier to administer.
- *Software Development:* The basic FreeBSD system comes with a full complement of development tools including the renowned GNU C/C++ compiler and debugger.

FreeBSD is available in both source and binary form on CDROM and via anonymous ftp. See Obtaining FreeBSD for more details.

## A Brief History of FreeBSD

*Contributed by Jordan K. Hubbard <jkh@FreeBSD.ORG>.*

The FreeBSD project had its genesis in the early part of 1993, partially as an outgrowth of the "Unofficial 386BSD Patchkit" by the patchkit's last 3 coordinators: Nate Williams, Rod Grimes and myself.

Our original goal was to produce an intermediate snapshot of 386BSD in order to fix a number of problems with it that the patchkit mechanism just was not capable of solving. Some of you may remember the early working title for the project being "386BSD 0.5" or "386BSD Interim" in reference to that fact.

386BSD was Bill Jolitz's operating system, which had been up to that point suffering rather severely from almost a year's worth of neglect. As the patchkit swelled ever more uncomfortably with each passing day, we were in unanimous agreement that something had to be done and decided to try and assist Bill by providing this interim "cleanup" snapshot. Those plans came to a rude halt when Bill Jolitz suddenly decided to withdraw his sanction from the project and without any clear indication of what would be done instead.

It did not take us long to decide that the goal remained worthwhile, even without Bill's support, and so we adopted the name "FreeBSD", coined by David Greenman. Our initial objectives were set after consulting with the system's current users and, once it became clear that the project was on the road to perhaps even becoming a reality, I contacted Walnut Creek CDROM with an eye towards improving FreeBSD's distribution channels for those many unfortunates without easy access to the Internet. Walnut Creek CDROM not only supported the idea of distributing FreeBSD on CD but went so far as to provide the project with a machine to work on and a fast Internet connection. Without Walnut Creek CDROM's almost unprecedented degree of faith in what was, at the time, a completely unknown project, it is quite unlikely that FreeBSD would have gotten as far, as fast, as it has today.

The first CDROM (and general net-wide) distribution was FreeBSD 1.0, released in December of 1993. This was based on the 4.3BSD-Lite ("Net/2") tape from U.C. Berkeley, with many components also provided by 386BSD and the Free Software Foundation. It was a fairly reasonable success for a first offering, and we followed it with the highly successful FreeBSD 1.1 release in May of 1994.

Around this time, some rather unexpected storm clouds formed on the horizon as Novell and U.C. Berkeley settled their long-running lawsuit over the legal status of the Berkeley Net/2 tape. A condition of that settlement was U.C. Berkeley's concession that large parts of Net/2 were "encumbered" code and the property of Novell, who had in turn acquired it from AT&T some time previously. What Berkeley got in return was Novell's "blessing" that the 4.4BSD-Lite release, when it was finally released, would be declared unencumbered and all existing Net/2 users would be strongly encouraged to switch. This included FreeBSD, and the project was given until the end of July 1994 to stop shipping its own Net/2 based product. Under the terms of that agreement, the project was allowed one last release before the deadline, that release being FreeBSD 1.1.5.1.

FreeBSD then set about the arduous task of literally re-inventing itself from a completely new and rather incomplete set of 4.4BSD-Lite bits. The "Lite" releases were light in part because Berkeley's CSRG had removed large chunks of code required for actually constructing a bootable running system (due to various legal requirements) and the fact that the Intel port of 4.4 was highly incomplete. It took the project until December of 1994 to make this transition, and in January of 1995 it released FreeBSD 2.0 to the net and on CDROM. Despite being still more than a little rough around the edges, the release was a significant success and was followed by the more robust and easier to install FreeBSD 2.0.5 release in June of 1995.

We released FreeBSD 2.1.5 in August of 1996, and it appeared to be popular enough among the ISP and commercial communities that another release along the 2.1-stable branch was merited. This was

FreeBSD 2.1.7.1, released in February 1997 and capping the end of mainstream development on 2.1-stable. Now in maintenance mode, only security enhancements and other critical bug fixes will be done on this branch (RELENG\_2\_1\_0).

FreeBSD 2.2 was branched from the development mainline (“-current”) in November 1996 as the RELENG\_2\_2 branch, and the first full release (2.2.1) was released in April, 1997. Further releases along the 2.2 branch were done in the Summer and Fall of ’97, the latest being 2.2.7 which appeared in late July of ’98. The first official 3.0 release appeared in October, 1998 and the last release on the 2.2 branch, 2.2.8, appeared in November, 1998.

The tree branched again on Jan 20, 1999. This led to 4.0-current and a 3.x-stable branch, from which 3.1 will be released on February 15th, 1999.

Long term development projects will continue to take place in the 4.0-current branch and SNAPSHOT releases of 4.0 on CDROM (and, of course, on the net).

## FreeBSD Project Goals

*Contributed by Jordan K. Hubbard <jkh@FreeBSD.ORG>.*

The goals of the FreeBSD Project are to provide software that may be used for any purpose and without strings attached. Many of us have a significant investment in the code (and project) and would certainly not mind a little financial compensation now and then, but we’re definitely not prepared to 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 is, I believe, one of the most fundamental goals of Free Software and one that we enthusiastically support.

That code in our source tree which falls under the GNU Public License (GPL) or GNU Library Public License (GLPL) 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 BSD copyright whenever possible.

## The FreeBSD Development Model

*Contributed by Satoshi Asami <asami@FreeBSD.ORG>.*

The development of FreeBSD is a very open and flexible process, FreeBSD being literally built from the contributions of hundreds of people around the world, as can be seen from our list of contributors. We are constantly on the lookout for new developers and ideas, and those interested in becoming more closely involved with the project need simply contact us at the FreeBSD technical discussions mailing

list <freebsd-hackers@FreeBSD.ORG>. Those who prefer to work more independently are also accommodated, and they are free to use our FTP facilities at ftp.freebsd.org (ftp://ftp.freebsd.org/pub/FreeBSD/incoming) to distribute their own patches or work-in-progress sources. The FreeBSD announcements mailing list <freebsd-announce@FreeBSD.ORG> is also available to those wishing to make other FreeBSD users aware of major areas of work.

Useful things to know about the FreeBSD project and its development process, whether working independently or in close cooperation:

#### The CVS repository

The central source tree for FreeBSD is maintained by CVS (<http://www.cyclic.com/cyclic-pages/CVS-sheet.html>) (Concurrent Version System), a freely available source code control tool which comes bundled with FreeBSD. The primary CVS repository (<http://www.freebsd.org/cgi/cvsweb.cgi>) resides on a machine in Concord CA, USA from where it is replicated to numerous mirror machines throughout the world. The CVS tree, as well as the -current and -stable trees which are checked out of it, can be easily replicated to your own machine as well. Please refer to the Synchronizing your source tree section for more information on doing this.

#### The committers list

The committers are the people who have *write* access to the CVS tree, and are thus authorized to make modifications to the FreeBSD source (the term “committer” comes from the `cvs(1) commit` command, which is used to bring new changes into the CVS repository). The best way of making submissions for review by the committers list is to use the `send-pr(1)` command, though if something appears to be jammed in the system then you may also reach them by sending mail to <committers@freebsd.org>.

#### The FreeBSD core team

The FreeBSD core team would be equivalent to the board of directors if the FreeBSD Project were a company. The primary task of the core team is to make sure the project, as a whole, is in good shape and is heading in the right directions. Inviting dedicated and responsible developers to join our group of committers is one of the functions of the core team, as is the recruitment of new core team members as others move on. Most current members of the core team started as committers who's addition to the project got the better of them.

Some core team members also have specific areas of responsibility, meaning that they are committed to ensuring that some large portion of the system works as advertised.

**Note:** Most members of the core team are volunteers when it comes to FreeBSD development and do not benefit from the project financially, so “commitment” should also not be

misconstrued as meaning “guaranteed support.” The “board of directors” analogy above is not actually very accurate, and it may be more suitable to say that these are the people who gave up their lives in favor of FreeBSD against their better judgement! ;)

#### Outside contributors

Last, but definitely not least, the largest group of developers are the users themselves who provide feedback and bug-fixes to us on an almost constant basis. The primary way of keeping in touch with FreeBSD’s more non-centralized development is to subscribe to the FreeBSD technical discussions mailing list <freebsd-hackers@FreeBSD.ORG> (see mailing list info) where such things are discussed.

The list of those who have contributed something which made its way into our source tree is a long and growing one, so why not join it by contributing something back to FreeBSD today? :-)

Providing code is not the only way of contributing to the project; for a more complete list of things that need doing, please refer to the how to contribute section in this handbook.

In summary, our development model is organized as a loose set of concentric circles. The centralized model is designed for the convenience of the *users* of FreeBSD, who are thereby provided with an easy way of tracking one central code base, not to keep potential contributors out! Our desire is to present a stable operating system with a large set of coherent application programs that the users can easily install and use, and this model works very well in accomplishing that.

All we ask of those who would join us as FreeBSD developers is some of the same dedication its current people have to its continued success!

## About the Current Release

FreeBSD is a freely available, full source 4.4BSD-Lite based release for Intel i386/i486/Pentium/PentiumPro/Pentium II (or compatible) based PC’s. It is based primarily on software from U.C. Berkeley’s CSRG group, with some enhancements from NetBSD, OpenBSD, 386BSD, and the Free Software Foundation.

Since our release of FreeBSD 2.0 in January of 95, the performance, feature set, and stability of FreeBSD has improved dramatically. The largest change is a revamped virtual memory system with a merged VM/file buffer cache that not only increases performance, but reduces FreeBSD’s memory footprint, making a 5MB configuration a more acceptable minimum. Other enhancements include full NIS client and server support, transaction TCP support, dial-on-demand PPP, an improved SCSI subsystem, early ISDN support, support for FDDI and Fast Ethernet (100Mbit) adapters, improved support for the Adaptec 2940 (WIDE and narrow) and many hundreds of bug fixes.

We have also taken the comments and suggestions of many of our users to heart and have attempted to provide what we hope is a more sane and easily understood installation process. Your feedback on this (constantly evolving) process is especially welcome!

In addition to the base distributions, FreeBSD offers a new ported software collection with hundreds of commonly sought-after programs. At the end of August 1998 there were more than 1700 ports! The list of ports ranges from http (WWW) servers, to games, languages, editors and almost everything in between. The entire ports collection requires approximately 26MB of storage, all ports being expressed as “deltas” to their original sources. This makes it much easier for us to update ports, and greatly reduces the disk space demands made by the older 1.0 ports collection. To compile a port, you simply change to the directory of the program you wish to install, type `make all` followed by `make install` after successful compilation and let the system do the rest. The full original distribution for each port you build is retrieved dynamically off the CDRom or a local ftp site, so you need only enough disk space to build the ports you want. (Almost) every port is also provided as a pre-compiled “package” which can be installed with a simple command (`pkg_add`) by those who do not wish to compile their own ports from source.

A number of additional documents which you may find very helpful in the process of installing and using FreeBSD may now also be found in the `/usr/share/doc` directory on any machine running FreeBSD 2.1 or later. You may view the locally installed manuals with any HTML capable browser using the following URLs:

The FreeBSD handbook

`file:/usr/share/doc/handbook/handbook.html`

The FreeBSD FAQ

`file:/usr/share/doc/FAQ/FAQ.html`

You can also visit the master (and most frequently updated) copies at <http://www.freebsd.org>.

The core of FreeBSD does not contain DES code which would inhibit its being exported outside the United States. There is an add-on package to the core distribution, for use only in the United States, that contains the programs that normally use DES. The auxiliary packages provided separately can be used by anyone. A freely (from outside the U.S.) exportable European distribution of DES for our non-U.S. users also exists and is described in the FreeBSD FAQ (`../FAQ/FAQ.html`).

If password security for FreeBSD is all you need, and you have no requirement for copying encrypted passwords from different hosts (Suns, DEC machines, etc) into FreeBSD password entries, then FreeBSD’s MD5 based security may be all you require! We feel that our default security model is more than a match for DES, and without any messy export issues to deal with. If you are outside (or even inside) the U.S., give it a try!

## Chapter 2. Installing FreeBSD

So, you would like to try out FreeBSD on your system? This section is a quick-start guide for what you need to do. FreeBSD can be installed from a variety of media including CD-ROM, floppy disk, magnetic tape, an MS-DOS partition and, if you have a network connection, via anonymous ftp or NFS.

Regardless of the installation media you choose, you can get started by creating the *installation disks* as described below. Booting your computer into the FreeBSD installer, even if you aren't planning on installing FreeBSD right away, will provide important information about compatibility between FreeBSD and your hardware which may, in turn, dictate which installation options are even possible. It can also provide early clues to any compatibility problems which could prevent FreeBSD running on your system at all.

If you plan on installing via anonymous FTP then the installation floppies are all you need to download and create—the installation program itself will handle any further required downloading directly (using an ethernet connection, a modem and ppp dialip #, etc).

For more information on obtaining the latest FreeBSD distributions, please see Obtaining FreeBSD in the Appendix.

So, to get the show on the road, follow these steps:

1. Review the supported configurations section of this installation guide to be sure that your hardware is supported by FreeBSD. It may be helpful to make a list of any special cards you have installed, such as SCSI controllers, Ethernet adapters or sound cards. This list should include relevant configuration parameters such as interrupts (IRQ) and IO port addresses.
2. If you're installing FreeBSD from CDROM media then you have several different installation options:
  - If the CD has been mastered with El Torrito boot support and your system supports direct booting from CDROM (and many older systems do *not*), simply insert the CD into the drive and boot directly from it.
  - If you're running DOS and have the proper drivers to access your CD, run the install.bat script provided on the CD. This will attempt to boot into the FreeBSD installation straight from DOS.

**Note:** You must do this from actual DOS and not a Windows DOS box.

If you also want to install FreeBSD from your DOS partition (perhaps because your CDROM drive is completely unsupported by FreeBSD) then run the setup program first to copy the appropriate files from the CD to your DOS partition, afterwards running install.

- If either of the two preceding methods work then you can simply skip the rest of this section, otherwise your final option is to create a set of boot floppies from the `floppies\kern.flp` and `floppies\mfsroot.flp` images—proceed to step 4 for instructions on how to do this.
3. If you don't have a CDROM distribution then simply read the installation boot image information (<ftp://ftp.FreeBSD.ORG/pub/FreeBSD/3.1-RELEASE/floppies/README.TXT>) to find out what files you need to download first.
  4. Make the installation boot disks from the image files:
    - If you are using MS-DOS then download `fdimage.exe` (<ftp://ftp.FreeBSD.ORG/pub/FreeBSD/tools/fdimage.exe>) or get it from `tools\fdimage.exe` on the CDROM and then run it like so:
 

```
E:\> tools\fdimage floppies\kern.flp a:
```

The `fdimage` program will format the A: drive and then copy the `kern.flp` image onto it (assuming that you're at the top level of a FreeBSD distribution and the floppy images live in the `floppies` subdirectory, as is typically the case).
    - If you are using a UNIX system to create the floppy images:
 

```
# dd if=kern.flp of=disk_device
```

`disk_device` is the `/dev` entry for the floppy drive. On FreeBSD systems, this is `/dev/rfd0` for the A: drive and `/dev/rfd1` for the B: drive.
  5. With the `kern.flp` in the A: drive, reboot your computer. The next request you should get is for the `mfsroot.flp` floppy, after which the installation will proceed normally.
 

If you do *not* type anything at the boot prompt which appears during this process, FreeBSD will automatically boot with its default configuration after a delay of about five seconds. As FreeBSD boots, it probes your computer to determine what hardware is installed. The results of this probing is displayed on the screen.
  6. When the booting process is finished, The main FreeBSD installation menu will be displayed.

*If something goes wrong...*

Due to limitations of the PC architecture, it is impossible for probing to be 100 percent reliable. In the event that your hardware is incorrectly identified, or that the probing causes your computer to lock up, first check the supported configurations section of this installation guide to be sure that your hardware is indeed supported by FreeBSD.

If your hardware is supported, reset the computer and when the visual kernel configuration choice is presented, take it. This puts FreeBSD into a configuration mode where you can supply hints about your hardware. The FreeBSD kernel on the installation disk is configured assuming that most hardware devices are in their factory default configuration in terms of IRQs, IO addresses and DMA channels. If

your hardware has been reconfigured, you will most likely need to use the configuration editor to tell FreeBSD where things are.

It is also possible that a probe for a device not present will cause a later probe for another device that is present to fail. In that case, the probes for the conflicting driver(s) should be disabled.

**Warning:** Do not disable any device you will need during installation, such as your screen (`sc0`). If the installation wedges or fails mysteriously after leaving the configuration editor, you have probably removed or changed something that you should not have. Simply reboot and try again.

In the configuration mode, you can:

- List the device drivers installed in the kernel.
- Disable device drivers for hardware not present in your system.
- Change the IRQ, DRQ, and IO port addresses used by a device driver.

After adjusting the kernel to match how you have your hardware configured, type `Q` to continue booting with the new settings.

After FreeBSD has been installed, changes made in the configuration mode will be permanent so you do not have to reconfigure every time you boot. Even so, it is likely that you will want to build a custom kernel to optimize the performance of your system. See Kernel configuration for more information on creating custom kernels.

## Supported Configurations

FreeBSD currently runs on a wide variety of ISA, VLB, EISA and PCI bus based PC's, ranging from 386sx to Pentium class machines (though the 386sx is not recommended). Support for generic IDE or ESDI drive configurations, various SCSI controller, network and serial cards is also provided.

A minimum of four megabytes of RAM is required to run FreeBSD. To run the X Window System, eight megabytes of RAM is the recommended minimum.

Following is a list of all disk controllers and Ethernet cards currently known to work with FreeBSD. Other configurations may very well work, and we have simply not received any indication of this.

## Disk Controllers

- WD1003 (any generic MFM/RLL)

- WD1007 (any generic IDE/ESDI)
- IDE
- ATA
- Adaptec 1505 ISA SCSI controller
- Adaptec 152x series ISA SCSI controllers
- Adaptec 1535 ISA SCSI controllers
- Adaptec 154x series ISA SCSI controllers
- Adaptec 174x series EISA SCSI controller in standard and enhanced mode.
- Adaptec 274x/284x/2940/2940U/3940 (Narrow/Wide/Twin) series EISA/VLB/PCI SCSI controllers
- Adaptec AIC7850, 7890, 7891, 7895, 7896, and 7897 on-board SCSI controllers
- Adaptec AIC-6360 based boards, which includes the AHA-152x and SoundBlaster SCSI cards.

**Note:** You cannot boot from the SoundBlaster cards as they have no on-board BIOS, which is necessary for mapping the boot device into the system BIOS I/O vectors. They are perfectly usable for external tapes, CDROMs, etc, however. The same goes for any other AIC-6x60 based card without a boot ROM. Some systems DO have a boot ROM, which is generally indicated by some sort of message when the system is first powered up or reset. Check your system/board documentation for more details.

- Buslogic 545S & 545c

**Note:** Buslogic was formerly known as “Bustek”.

- Buslogic 445S/445c VLB SCSI controller
- Buslogic 742A/747S/747c EISA SCSI controller.
- Buslogic 946c PCI SCSI controller
- Buslogic 956c PCI SCSI controller
- NCR 53C810/53C815/53C825/53C860/53C875 PCI SCSI controller.
- NCR5380/NCR53400 (“ProAudio Spectrum”) SCSI controller.
- DTC 3290 EISA SCSI controller in 1542 emulation mode.
- UltraStor 14F/24F/34F SCSI controllers.
- Seagate ST01/02 SCSI controllers.

- Future Domain 8xx/950 series SCSI controllers.
- WD7000 SCSI controllers.

With all supported SCSI controllers, full support is provided for SCSI-I & SCSI-II peripherals, including Disks, tape drives (including DAT) and CD ROM drives.

The following CD-ROM type systems are supported at this time:

- SoundBlaster SCSI and ProAudio Spectrum SCSI (`cd`)
- Mitsumi (all models) proprietary interface (`mcd`)
- Matsushita/Panasonic (Creative) CR-562/CR-563 proprietary interface (`matcd`)
- Sony proprietary interface (`scd`)
- ATAPI IDE interface (experimental and should be considered ALPHA quality!) (`wcd`)

## Ethernet cards

- Allied-Telesis AT1700 and RE2000 cards
- SMC Elite 16 WD8013 Ethernet interface, and most other WD8003E, WD8003EBT, WD8003W, WD8013W, WD8003S, WD8003SBT and WD8013EBT based clones. SMC Elite Ultra and 9432TX based cards are also supported.
- DEC EtherWORKS III NICs (DE203, DE204, and DE205)
- DEC EtherWORKS II NICs (DE200, DE201, DE202, and DE422)
- DEC DC21040/DC21041/DC21140 based NICs:
  - ASUS PCI-L101-TB
  - Accton ENI1203
  - Cogent EM960PCI
  - Compex CPXPCI/32C
  - D-Link DE-530
  - DEC DE435
  - Danpex EN-9400P3
  - JCIS Condor JC1260
  - Kingston KNE100TX

- Linksys EtherPCI
- Mylex LNP101
- SMC EtherPower 10/100 (Model 9332)
- SMC EtherPower (Model 8432)
- SMC EtherPower (2)
- Zynx ZX314
- Zynx ZX342
  
- DEC FDDI (DEFPA/DEFEA) NICs
- Fujitsu FMV-181 and FMV-182
- Fujitsu MB86960A/MB86965A
- Intel EtherExpress
- Intel EtherExpress Pro/100B 100Mbit.
- Isolan AT 4141-0 (16 bit)
- Isolink 4110 (8 bit)
- Lucent WaveLAN wireless networking interface.
- Novell NE1000, NE2000, and NE2100 ethernet interface.
- 3Com 3C501 cards
- 3Com 3C503 Etherlink II
- 3Com 3c505 Etherlink/+
- 3Com 3C507 Etherlink 16/TP
- 3Com 3C509, 3C579, 3C589 (PCMCIA) Etherlink III
- 3Com 3C590, 3C595 Etherlink III
- 3Com 3C90x cards.
- HP PC Lan Plus (27247B and 27252A)
- Toshiba ethernet cards
- PCMCIA ethernet cards from IBM and National Semiconductor are also supported.

**Note:** FreeBSD does not currently support PnP (plug-n-play) features present on some ethernet cards. If your card has PnP and is giving you problems, try disabling its PnP features.

## Miscellaneous devices

- AST 4 port serial card using shared IRQ.
- ARNET 8 port serial card using shared IRQ.
- BOCA IOAT66 6 port serial card using shared IRQ.
- BOCA 2016 16 port serial card using shared IRQ.
- Cyclades Cyclom-y Serial Board.
- STB 4 port card using shared IRQ.
- SDL Communications Riscom/8 Serial Board.
- SDL Communications RISCom/N2 and N2pci sync serial cards.
- Digiboard Sync/570i high-speed sync serial card.
- Decision-Computer Intl. "Eight-Serial" 8 port serial cards using shared IRQ.
- Adlib, SoundBlaster, SoundBlaster Pro, ProAudioSpectrum, Gravis UltraSound, Gravis UltraSound MAX and Roland MPU-401 sound cards.
- Matrox Meteor video frame grabber.
- Creative Labs Video spigot frame grabber.
- Omnimedia Talisman frame grabber.
- Brooktree BT848 chip based frame grabbers.
- X-10 power controllers.
- PC joystick and speaker.

FreeBSD does not currently support IBM's microchannel (MCA) bus.

## Preparing for the Installation

There are a number of different methods by which FreeBSD can be installed. The following describes what preparation needs to be done for each type.

### Before installing from CDROM

If your CDROM is of an unsupported type, then please skip to MS-DOS Preparation.

There is not a lot of preparatory work that needs to be done to successfully install from one of Walnut Creek's FreeBSD CDROMs (other CDROM distributions may work as well, though we cannot say for certain as we have no hand or say in how they are created). You can either boot into the CD installation directly from DOS using Walnut Creek's supplied `install.bat` batch file or you can make boot floppies with the `makefloppy.bat` command.

For the easiest interface of all (from DOS), type `view`. This will bring up a DOS menu utility that leads you through all the available options.

If you are creating the boot floppies from a UNIX machine, see the beginning of this guide for examples of how to create the boot floppies.

Once you have booted from DOS or floppy, you should then be able to select CDROM as the media type in the Media menu and load the entire distribution from CDROM. No other types of installation media should be required.

After your system is fully installed and you have rebooted from the hard disk, you can mount the CDROM at any time by typing: `mount /cdrom`

Before removing the CD again, also note that it is necessary to first type: `umount /cdrom`. Do not just remove it from the drive!

**Note:** Before invoking the installation, be sure that the CDROM is in the drive so that the install probe can find it. This is also true if you wish the CDROM to be added to the default system configuration automatically during the install (whether or not you actually use it as the installation media).

Finally, if you would like people to be able to FTP install FreeBSD directly from the CDROM in your machine, you will find it quite easy. After the machine is fully installed, you simply need to add the following line to the password file (using the `vipw` command):

```
ftp:*:99:99::0:0:FTP:/cdrom:/nonexistent
```

Anyone with network connectivity to your machine (and permission to log into it) can now choose a Media type of FTP and type in: **ftp://your machine** after picking "Other" in the ftp sites menu.

## Before installing from Floppy

If you must install from floppy disks, either due to unsupported hardware or simply because you enjoy doing things the hard way, you must first prepare some floppies for the install.

You will need, at minimum, as many 1.44MB or 1.2MB floppies as it takes to hold all files in the `bin` (binary distribution) directory. If you are preparing these floppies under DOS, then THESE floppies *must* be formatted using the MS-DOS `FORMAT` command. If you are using Windows, use the Windows File Manager format command.

Do *not* trust Factory Preformatted floppies! Format them again yourself, just to make sure. Many problems reported by our users in the past have resulted from the use of improperly formatted media, which is why I am taking such special care to mention it here!

If you are creating the floppies from another FreeBSD machine, a format is still not a bad idea though you do not need to put a DOS filesystem on each floppy. You can use the `disklabel` and `newfs` commands to put a UFS filesystem on them instead, as the following sequence of commands (for a 3.5" 1.44MB floppy disk) illustrates:

```
# fdformat -f 1440 fd0.1440
# disklabel -w -r fd0.1440 floppy3
# newfs -t 2 -u 18 -l 1 -i 65536 /dev/rfd0
```

**Note:** Use `fd0.1200` and `floppy5` for 5.25" 1.2MB disks.

Then you can mount and write to them like any other file system.

After you have formatted the floppies, you will need to copy the files onto them. The distribution files are split into chunks conveniently so that 5 of them will fit on a conventional 1.44MB floppy. Go through all your floppies, packing as many files as will fit on each one, until you have got all the distributions you want packed up in this fashion. Each distribution should go into a subdirectory on the floppy, e.g.: `a:\bin\bin.aa`, `a:\bin\bin.ab`, and so on.

Once you come to the Media screen of the install, select “Floppy” and you will be prompted for the rest.

## Before installing from a MS-DOS partition

To prepare for installation from an MS-DOS partition, copy the files from the distribution into a directory called `C:\FREEBSD`. The directory tree structure of the CDROM must be partially reproduced within this directory so we suggest using the DOS `xcopy` command. For example, to prepare for a minimal installation of FreeBSD:

```
C:\> MD C:\FREEBSD
C:\> XCOPY /S E:\BIN C:\FREEBSD\BIN\
C:\> XCOPY /S E:\MANPAGES C:\FREEBSD\MANPAGES\
```

Assuming that `C:` is where you have free space and `E:` is where your CDROM is mounted.

For as many DISTs you wish to install from MS-DOS (and you have free space for), install each one under `C:\FREEBSD` — the BIN dist is only the minimal requirement.

## Before installing from QIC/SCSI Tape

Installing from tape is probably the easiest method, short of an on-line install using FTP or a CDROM install. The installation program expects the files to be simply tar'ed onto the tape, so after getting all of the files for distribution you are interested in, simply tar them onto the tape with a command like:

```
# cd /freebsd/distdir
# tar cvf /dev/rwt0 dist1 ... dist2
```

When you go to do the installation, you should also make sure that you leave enough room in some temporary directory (which you will be allowed to choose) to accommodate the *full* contents of the tape you have created. Due to the non-random access nature of tapes, this method of installation requires quite a bit of temporary storage. You should expect to require as much temporary storage as you have stuff written on tape.

**Note:** When going to do the installation, the tape must be in the drive *before* booting from the boot floppy. The installation probe may otherwise fail to find it.

## Before installing over a network

You can do network installations over 3 types of communications links:

Serial port

SLIP or PPP

Parallel port

PLIP (laplink cable)

Ethernet

A standard ethernet controller (includes some PCMCIA).

SLIP support is rather primitive, and limited primarily to hard-wired links, such as a serial cable running between a laptop computer and another computer. The link should be hard-wired as the SLIP installation does not currently offer a dialing capability; that facility is provided with the PPP utility, which should be used in preference to SLIP whenever possible.

If you are using a modem, then PPP is almost certainly your only choice. Make sure that you have your service provider's information handy as you will need to know it fairly soon in the installation process. You will need to know how to dial your ISP using the "AT commands" specific to your modem, as the

PPP dialer provides only a very simple terminal emulator. If you're using PAP or CHAP, you'll need to type the necessary `set authname` and `set authkey` commands before typing `term`. Refer to the user-ppp handbook and FAQ ([../FAQ/userppp.html](#)) entries for further information. If you have problems, logging can be directed to the screen using the command `set log local ...`.

If a hard-wired connection to another FreeBSD (2.0R or later) machine is available, you might also consider installing over a "laplink" parallel port cable. The data rate over the parallel port is much higher than what is typically possible over a serial line (up to 50k/sec), thus resulting in a quicker installation.

Finally, for the fastest possible network installation, an ethernet adaptor is always a good choice! FreeBSD supports most common PC ethernet cards, a table of supported cards (and their required settings) is provided in Supported Hardware. If you are using one of the supported PCMCIA ethernet cards, also be sure that it is plugged in *before* the laptop is powered on! FreeBSD does not, unfortunately, currently support hot insertion of PCMCIA cards during installation.

You will also need to know your IP address on the network, the netmask value for your address class, and the name of your machine. Your system administrator can tell you which values to use for your particular network setup. If you will be referring to other hosts by name rather than IP address, you will also need a name server and possibly the address of a gateway (if you are using PPP, it is your provider's IP address) to use in talking to it. If you do not know the answers to all or most of these questions, then you should really probably talk to your system administrator *first* before trying this type of installation.

Once you have a network link of some sort working, the installation can continue over NFS or FTP.

## Preparing for NFS installation

NFS installation is fairly straight-forward: Simply copy the FreeBSD distribution files you want onto a server somewhere and then point the NFS media selection at it.

If this server supports only "privileged port" access (as is generally the default for Sun workstations), you will need to set this option in the Options menu before installation can proceed.

If you have a poor quality ethernet card which suffers from very slow transfer rates, you may also wish to toggle the appropriate Options flag.

In order for NFS installation to work, the server must support subdir mounts, e.g., if your FreeBSD 3.1 distribution directory lives on: `ziggy:/usr/archive/stuff/FreeBSD` Then `ziggy` will have to allow the direct mounting of `/usr/archive/stuff/FreeBSD`, not just `/usr` or `/usr/archive/stuff`.

In FreeBSD's `/etc/exports` file, this is controlled by the `-alldirs` option. Other NFS servers may have different conventions. If you are getting Permission Denied messages from the server then it is likely that you do not have this enabled properly.

## Preparing for FTP Installation

FTP installation may be done from any mirror site containing a reasonably up-to-date version of FreeBSD 3.1. A full menu of reasonable choices from almost anywhere in the world is provided by the FTP site menu.

If you are installing from some other FTP site not listed in this menu, or you are having troubles getting your name server configured properly, you can also specify your own URL by selecting the “Other” choice in that menu. A URL can also be a direct IP address, so the following would work in the absence of a name server:

```
ftp://165.113.121.81/pub/FreeBSD/3.1-RELEASE
```

There are two FTP installation modes you can use:

### FTP Active

For all FTP transfers, use “Active” mode. This will not work through firewalls, but will often work with older ftp servers that do not support passive mode. If your connection hangs with passive mode (the default), try active!

### FTP Passive

For all FTP transfers, use “Passive” mode. This allows the user to pass through firewalls that do not allow incoming connections on random port addresses.

**Note:** Active and passive modes are not the same as a “proxy” connection, where a proxy FTP server is listening and forwarding FTP requests!

For a proxy FTP server, you should usually give name of the server you really want as a part of the username, after an @-sign. The proxy server then ‘fakes’ the real server. An example: Say you want to install from `ftp.freebsd.org`, using the proxy FTP server `foo.bar.com`, listening on port 1234.

In this case, you go to the options menu, set the FTP username to `ftp@ftp.freebsd.org`, and the password to your e-mail address. As your installation media, you specify FTP (or passive FTP, if the proxy support it), and the URL `ftp://foo.bar.com:1234/pub/FreeBSD`

`/pub/FreeBSD` from `ftp.freebsd.org` is proxied under `foo.bar.com`, allowing you to install from *that* machine (which fetch the files from `ftp.freebsd.org` as your installation requests them).

## Installing FreeBSD

Once you have taken note of the appropriate preinstallation steps, you should be able to install FreeBSD without any further trouble.

Should this not be true, then you may wish to go back and re-read the relevant preparation section above for the installation media type you are trying to use, perhaps there is a helpful hint there that you missed the first time? If you are having hardware trouble, or FreeBSD refuses to boot at all, read the Hardware Guide provided on the boot floppy for a list of possible solutions.

The FreeBSD boot floppies contain all the on-line documentation you should need to be able to navigate through an installation and if it does not then we would like to know what you found most confusing.

Send your comments to the FreeBSD documentation project mailing list

<freebsd-doc@FreeBSD.ORG>. It is the objective of the FreeBSD installation program (sysinstall) to be self-documenting enough that painful “step-by-step” guides are no longer necessary. It may take us a little while to reach that objective, but that is the objective!

Meanwhile, you may also find the following “typical installation sequence” to be helpful:

1. Boot the `kern.flp` floppy and, when asked, remove it and insert the `mfsroot.flp` floppy and hit return. After a boot sequence which can take anywhere from 30 seconds to 3 minutes, depending on your hardware, you should be presented with a menu of initial choices. If the `kern.flp` floppy does not boot at all, or the boot hangs at some stage, go read the Q&A section of the Hardware Guide for possible causes.
2. Press F1. You should see some basic usage instructions on the menu system and general navigation. If you have not used this menu system before then *please* read this thoroughly!
3. Select the Options item and set any special preferences you may have.
4. Select a Novice, Custom or Express install, depending on whether or not you would like the installation to help you through a typical installation, give you a high degree of control over each step of the installation or simply whizz through it (using reasonable defaults when possible) as fast as possible. If you have never used FreeBSD before then the Novice installation method is most recommended.
5. The final configuration menu choice allows you to further configure your FreeBSD installation by giving you menu-driven access to various system defaults. Some items, like networking, may be especially important if you did a CDROM/Tape/Floppy installation and have not yet configured your network interfaces (assuming you have any). Properly configuring such interfaces here will allow FreeBSD to come up on the network when you first reboot from the hard disk.

## MS-DOS User's Questions and Answers

Many FreeBSD users wish to install FreeBSD on PCs inhabited by MS-DOS. Here are some commonly asked questions about installing FreeBSD on such systems.

*Help! I have no space! Do I need to delete everything first?*

If your machine is already running MS-DOS and has little or no free space available for FreeBSD's installation, all is not lost! You may find the FIPS utility, provided in the `tools` directory on the FreeBSD CDROM or on the various FreeBSD ftp sites, to be quite useful.

FIPS allows you to split an existing MS-DOS partition into two pieces, preserving the original partition and allowing you to install onto the second free piece. You first defragment your MS-DOS partition, using the DOS 6.xx DEFRAG utility or the Norton Disk tools, then run FIPS. It will prompt you for the rest of the information it needs. Afterwards, you can reboot and install FreeBSD on the new free slice. See the *Distributions* menu for an estimation of how much free space you will need for the kind of installation you want.

*Can I use compressed MS-DOS filesystems from FreeBSD?*

No. If you are using a utility such as Stacker(tm) or DoubleSpace(tm), FreeBSD will only be able to use whatever portion of the filesystem you leave uncompressed. The rest of the filesystem will show up as one large file (the stacked/dblspaced file!). *Do not remove that file!* You will probably regret it greatly!

It is probably better to create another uncompressed MS-DOS primary partition and use this for communications between MS-DOS and FreeBSD.

*Can I mount my MS-DOS extended partitions?*

Yes. DOS extended partitions are mapped in at the end of the other "slices" in FreeBSD, e.g. your `D:` drive might be `/dev/sd0s5`, your `E:` drive `/dev/sd0s6`, and so on. This example assumes, of course, that your extended partition is on SCSI drive 0. For IDE drives, substitute `wd` for `sd` appropriately. You otherwise mount extended partitions exactly like you would mount any other DOS drive, e.g.:

```
# mount -t msdos /dev/sd0s5 /dos_d
```

# Chapter 3. Unix Basics

## The Online Manual

The most comprehensive documentation on FreeBSD is in the form of *man pages*. Nearly every program on the system comes with a short reference manual explaining the basic operation and various arguments. These manuals can be view with the `man` command. Use of the `man` command is simple:

```
% man command
```

*command* is the name of the command you wish to learn about. For example, to learn more about `ls` command type:

```
% man ls
```

The online manual is divided up into numbered sections:

1. User commands
2. System calls and error numbers
3. Functions in the C libraries
4. Device drivers
5. File formats
6. Games and other diversions
7. Miscellaneous information
8. System maintenance and operation commands

In some cases, the same topic may appear in more than one section of the on-line manual. For example, there is a `chmod` user command and a `chmod()` system call. In this case, you can tell the `man` command which one you want by specifying the section:

```
% man 1 chmod
```

This will display the manual page for the user command `chmod`. References to a particular section of the on-line manual are traditionally placed in parenthesis in written documentation, so `chmod(1)` refers to the `chmod` user command and `chmod(2)` refers to the system call.

This is fine if you know the name of the command and simply wish to know how to use it, but what if you cannot recall the command name? You can use `man` to search for keywords in the command *descriptions* by using the `-k` switch:

```
% man -k mail
```

With this command you will be presented with a list of commands that have the keyword “mail” in their descriptions. This is actually functionally equivalent to using the `apropos` command.

So, you are looking at all those fancy commands in `/usr/bin` but do not even have the faintest idea what most of them actually do? Simply do a

```
% cd /usr/bin; man -f *
```

or

```
% cd /usr/bin; whatis *
```

which does the same thing.

## GNU Info Files

FreeBSD includes many applications and utilities produced by the Free Software Foundation (FSF). In addition to man pages, these programs come with more extensive hypertext documents called “info” files which can be viewed with the `info` command or, if you installed `emacs`, the info mode of `emacs`.

To use the `info(1)` command, simply type:

```
% info
```

For a brief introduction, type `h`. For a quick command reference, type `?`.

# Chapter 4. Installing Applications: The Ports collection

*Contributed by James Raynard <jraynard@freebsd.org>.*

The FreeBSD Ports collection allows you to compile and install a very wide range of applications with a minimum of effort.

For all the hype about open standards, getting a program to work on different versions of Unix in the real world can be a tedious and tricky business, as anyone who has tried it will know. You may be lucky enough to find that the program you want will compile cleanly on your system, install itself in all the right places and run flawlessly “out of the box”, but this is unfortunately rather rare. With most programs, you will find yourself doing a fair bit of head-scratching, and there are quite a few programs that will result in premature greying, or even chronic alopecia...

Some software distributions have attacked this problem by providing configuration scripts. Some of these are very clever, but they have an unfortunate tendency to triumphantly announce that your system is something you have never heard of and then ask you lots of questions that sound like a final exam in system-level Unix programming (Does your system’s `gethitlist` function return a `const` pointer to a `fromboz` or a pointer to a `const fromboz`? Do you have Foonix style unacceptable exception handling? And if not, why not?).

Fortunately, with the Ports collection, all the hard work involved has already been done, and you can just type `make install` and get a working program.

## Why Have a Ports Collection?

The base FreeBSD system comes with a very wide range of tools and system utilities, but a lot of popular programs are not in the base system, for good reasons:-

1. Programs that some people cannot live without and other people cannot stand, such as a certain Lisp-based editor.
2. Programs which are too specialised to put in the base system (CAD, databases).
3. Programs which fall into the “I must have a look at that when I get a spare minute” category, rather than system-critical ones (some languages, perhaps).
4. Programs that are far too much fun to be supplied with a serious operating system like FreeBSD ;-)
5. However many programs you put in the base system, people will always want more, and a line has to be drawn somewhere (otherwise FreeBSD distributions would become absolutely enormous).

Obviously it would be unreasonable to expect everyone to port their favourite programs by hand (not to mention a tremendous amount of duplicated work), so the FreeBSD Project came up with an ingenious way of using standard tools that would automate the process.

Incidentally, this is an excellent illustration of how “the Unix way” works in practice by combining a set of simple but very flexible tools into something very powerful.

## How Does the Ports Collection Work?

Programs are typically distributed on the Internet as a tarball consisting of a Makefile and the source code for the program and usually some instructions (which are unfortunately not always as instructive as they could be), with perhaps a configuration script.

The standard scenario is that you FTP down the tarball, extract it somewhere, glance through the instructions, make any changes that seem necessary, run the configure script to set things up and use the standard make program to compile and install the program from the source.

FreeBSD ports still use the tarball mechanism, but use a skeleton to hold the “knowledge” of how to get the program working on FreeBSD, rather than expecting the user to be able to work it out. They also supply their own customised Makefile, so that almost every port can be built in the same way.

If you look at a port skeleton (either on your FreeBSD system (file://localhost/usr/ports/devel/ElectricFence) or the FTP site (ftp://ftp.freebsd.org/pub/FreeBSD/ports/ports/devel/ElectricFence)) and expect to find all sorts of pointy-headed rocket science lurking there, you may be disappointed by the one or two rather unexciting-looking files and directories you find there. (We will discuss in a minute how to go about Getting a port).

“How on earth can this do anything?” I hear you cry. “There is no source code there!”

Fear not, gentle reader, all will become clear (hopefully). Let’s see what happens if we try and install a port. I have chosen **ElectricFence**, a useful tool for developers, as the skeleton is more straightforward than most.

**Note:** If you are trying this at home, you will need to be root.

```
# cd /usr/ports/devel/ElectricFence
# make install
>> Checksum OK for ElectricFence-2.0.5.tar.gz.
===> Extracting for ElectricFence-2.0.5
===> Patching for ElectricFence-2.0.5
===> Applying FreeBSD patches for ElectricFence-2.0.5
===> Configuring for ElectricFence-2.0.5
```

```
==> Building for ElectricFence-2.0.5
[lots of compiler output...]
==> Installing for ElectricFence-2.0.5
==> Warning: your umask is "0002". If this is not desired, set it to
      an appropriate value and install this port again by "make reinstall".
install -c -o bin -g bin -
m 444 /usr/ports/devel/ElectricFence/work/ElectricFence-
2.0.5/libefence.a /usr/local/lib
install -c -o bin -g bin -
m 444 /usr/ports/devel/ElectricFence/work/ElectricFence-
2.0.5/libefence.3 /usr/local/man/man3
==> Compressing manual pages for ElectricFence-2.0.5
==> Registering installation for ElectricFence-2.0.5
```

To avoid confusing the issue, I have completely removed the build output.

If you tried this yourself, you may well have got something like this at the start:-

```
# make install
>> ElectricFence-2.0.5.tar.gz doesn't seem to exist on this system.
>> Attempt-
ing to fetch from ftp://ftp.doc.ic.ac.uk/Mirrors/sunsite.unc.edu/pub/Linux/devel/lang/c/.
```

The make program has noticed that you did not have a local copy of the source code and tried to FTP it down so it could get the job done. I already had the source handy in my example, so it did not need to fetch it.

Let's go through this and see what the make program was doing.

1. Locate the source code tarball. If it is not available locally, try to grab it from an FTP site.
2. Run a checksum test on the tarball to make sure it has not been tampered with, accidentally truncated, downloaded in ASCII mode, struck by neutrinos while in transit, etc.
3. Extract the tarball into a temporary work directory.
4. Apply any patches needed to get the source to compile and run under FreeBSD.
5. Run any configuration script required by the build process and correctly answer any questions it asks.
6. (Finally!) Compile the code.
7. Install the program executable and other supporting files, man pages, etc. under the `/usr/local` hierarchy, where they will not get mixed up with system programs. This also makes sure that all the ports you install will go in the same place, instead of being flung all over your system.

8. Register the installation in a database. This means that, if you do not like the program, you can cleanly remove all traces of it from your system.

Scroll up to the `make` output and see if you can match these steps to it. And if you were not impressed before, you should be by now!

## Getting a FreeBSD Port

There are two ways of getting hold of the FreeBSD port for a program. One requires a FreeBSD CDROM, the other involves using an Internet Connection.

## Compiling ports from CDROM

Assuming that your FreeBSD CDROM is in the drive and mounted on `/cdrom` (and the mount point *must* be `/cdrom`), you should then be able to build ports just as you normally do and the port collection's built in search path should find the tarballs in `/cdrom/ports/distfiles/` (if they exist there) rather than downloading them over the net.

Another way of doing this, if you want to just use the port skeletons on the CDROM, is to set these variables in `/etc/make.conf`:

```
PORTSDIR=      /cdrom/ports
DISTDIR=       /tmp/distfiles
WRKDIRPREFIX=  /tmp
```

Substitute `/tmp` for any place you have enough free space. Then, just `cd` to the appropriate subdirectory under `/cdrom/ports` and type `make install` as usual. `WRKDIRPREFIX` will cause the port to be built under `/tmp/cdrom/ports`; for instance, `games/oneko` will be built under `/tmp/cdrom/ports/games/oneko`.

**Note:** There are some ports for which we cannot provide the original source in the CDROM due to licensing limitations. In that case, you will need to look at the section on Compiling ports using an Internet connection.

## Compiling ports from the Internet

If you do not have a CDROM, or you want to make sure you get the very latest version of the port you want, you will need to download the skeleton for the port. Now this might sound like rather a fiddly job

full of pitfalls, but it is actually very easy.

First, if you are running a release version of FreeBSD, make sure you get the appropriate “upgradekit” for your release from the ports web page (<http://www.freebsd.org/ports/>). These packages include files that have been updated since the release that you may need to compile new ports.

The key to the skeletons is that the FreeBSD FTP server can create on-the-fly tarballs for you. Here is how it works, with the gnats program in the databases directory as an example (the bits in square brackets are comments. Do not type them in if you are trying this yourself!):-

```
# cd /usr/ports
# mkdir databases
# cd databases
# ftp ftp.freebsd.org
[log in as 'ftp' and give your email address when asked for a
password. Remember to use binary (also known as image) mode!]
> cd /pub/FreeBSD/ports/ports/databases
> get gnats.tar
[tars up the gnats skeleton for us]
> quit
# tar xf gnats.tar
[extract the gnats skeleton]
# cd gnats
# make install
[build and install gnats]
```

What happened here? We connected to the FTP server in the usual way and went to its databases sub-directory. When we gave it the command `get gnats.tar`, the FTP server tarred up the gnats directory for us.

We then extracted the gnats skeleton and went into the gnats directory to build the port. As we explained earlier, the make process noticed we did not have a copy of the source locally, so it fetched one before extracting, patching and building it.

Let’s try something more ambitious now. Instead of getting a single port skeleton, let’s get a whole sub-directory, for example all the database skeletons in the ports collection. It looks almost the same:-

```
# cd /usr/ports
# ftp ftp.freebsd.org
[log in as 'ftp' and give your email address when asked for a
password. Remember to use binary (also known as image) mode!]
> cd /pub/FreeBSD/ports/ports
> get databases.tar
[tars up the databases directory for us]
> quit
# tar xf databases.tar
```

```
[extract all the database skeletons]
# cd databases
# make install
[build and install all the database ports]
```

With half a dozen straightforward commands, we have now got a set of database programs on our FreeBSD machine! All we did that was different from getting a single port skeleton and building it was that we got a whole directory at once, and compiled everything in it at once. Pretty impressive, no?

If you expect to be installing many ports, it is probably worth downloading all the ports directories.

## Skeletons

A team of compulsive hackers who have forgotten to eat in a frantic attempt to make a deadline? Something unpleasant lurking in the FreeBSD attic? No, a skeleton here is a minimal framework that supplies everything needed to make the ports magic work.

## Makefile

The most important component of a skeleton is the Makefile. This contains various statements that specify how the port should be compiled and installed. Here is the Makefile for ElectricFence:-

```
# New ports collection makefile for: Electric Fence
# Version required: 2.0.5
# Date created: 13 November 1997
# Whom: jraynard
#
# $Id$
#

DISTNAME=      ElectricFence-2.0.5
CATEGORIES=    devel
MASTER_SITES=  ${MASTER_SITE_SUNSITE}
MASTER_SITE_SUBDIR= devel/lang/c

MAINTAINER=   jraynard@freebsd.org

MAN3=         libefence.3

do-install:
    ${INSTALL_DATA} ${WRKSRC}/libefence.a ${PREFIX}/lib
```

```
    ${INSTALL_MAN} ${WRKSRV}/libefence.3 ${PREFIX}/man/man3  
  
.include <bsd.port.mk>
```

The lines beginning with a "#" sign are comments for the benefit of human readers (as in most Unix script files).

`DISTNAME` specifies the name of the tarball, but without the extension.

`CATEGORIES` states what kind of program this is. In this case, a utility for developers. See the categories section of this handbook for a complete list.

`MASTER_SITES` is the URL(s) of the master FTP site, which is used to retrieve the tarball if it is not available on the local system. This is a site which is regarded as reputable, and is normally the one from which the program is officially distributed (in so far as any software is "officially" distributed on the Internet).

`MAINTAINER` is the email address of the person who is responsible for updating the skeleton if, for example a new version of the program comes out.

Skipping over the next few lines for a minute, the line `.include <bsd.port.mk>` says that the other statements and commands needed for this port are in a standard file called `bsd.port.mk`. As these are the same for all ports, there is no point in duplicating them all over the place, so they are kept in a single standard file.

This is probably not the place to go into a detailed examination of how Makefiles work; suffice it to say that the line starting with `MAN3` ensures that the ElectricFence man page is compressed after installation, to help conserve your precious disk space. The original port did not provide an `install` target, so the three lines from `do-install` ensure that the files produced by this port are placed in the correct destination.

## The files directory

The file containing the checksum for the port is called `md5`, after the MD5 algorithm used for ports checksums. It lives in a directory with the slightly confusing name of `files`.

This directory can also contain other miscellaneous files that are required by the port and do not belong anywhere else.

## The patches directory

This directory contains the patches needed to make everything work properly under FreeBSD.

## The `pkg` directory

This program contains three quite useful files:-

- `COMMENT` — a one-line description of the program.
- `DESCR` — a more detailed description.
- `PLIST` — a list of all the files that will be created when the program is installed.

## What to do when a port does not work.

Oh. You can do one of four (4) things :

1. Fix it yourself. Technical details on how ports work can be found in Porting applications.
2. Gripe. This is done by e-mail *only!* Send such e-mail to the FreeBSD ports mailing list `<freebsd-ports@FreeBSD.ORG>` and please include the name/version of the port, where you got both the port source & distfile(s) from, and what the text of the error was.
3. Forget it. This is the easiest for most — very few of the programs in ports can be classified as essential!
4. Grab the pre-compiled package from a ftp server. The “master” package collection is on FreeBSD’s FTP server in the packages directory (`ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/packages/`), though check your local mirror first, please! These are more likely to work (on the whole) than trying to compile from source and a lot faster besides! Use the `pkg.add(1)` program to install a package file on your system.

## Some Questions and Answers

- Q. I thought this was going to be a discussion about modems??!  
A. Ah. You must be thinking of the serial ports on the back of your computer. We are using “port” here to mean the result of “porting” a program from one version of Unix to another. (It is an unfortunate bad habit of computer people to use the same word to refer to several completely different things).
- Q. I thought you were supposed to use packages to install extra programs?  
A. Yes, that is usually the quickest and easiest way of doing it.

- Q. So why bother with ports then?

A. Several reasons:-

1. The licensing conditions on some software distributions require that they be distributed as source code, not binaries.
2. Some people do not trust binary distributions. At least with source code you can (in theory) read through it and look for potential problems yourself.
3. If you have some local patches, you will need the source to add them yourself.
4. You might have opinions on how a program should be compiled that differ from the person who did the package — some people have strong views on what optimisation setting should be used, whether to build debug versions and then strip them or not, etc. etc.
5. Some people like having code around, so they can read it if they get bored, hack around with it, borrow from it (licence terms permitting, of course!) and so on.
6. If you ain't got the source, it ain't software! ;-)

- Q. What is a patch?

A. A patch is a small (usually) file that specifies how to go from one version of a file to another. It contains text that says, in effect, things like “delete line 23”, “add these two lines after line 468” or “change line 197 to this”. Also known as a “diff”, since it is generated by a program of that name.

- Q. What is all this about tarballs?

A. It is a file ending in `.tar` or `.tar.gz` (with variations like `.tar.Z`, or even `.tgz` if you are trying to squeeze the names into a DOS filesystem).

Basically, it is a directory tree that has been archived into a single file (`.tar`) and optionally compressed (`.gz`). This technique was originally used for *Tape ARchives* (hence the name `tar`), but it is a widely used way of distributing program source code around the Internet.

You can see what files are in them, or even extract them yourself, by using the standard Unix `tar` program, which comes with the base FreeBSD system, like this:-

```
% tar tvzf foobar.tar.gz
% tar xzvf foobar.tar.gz
% tar tvf foobar.tar
% tar xvf foobar.tar
```

- Q. And a checksum?

A. It is a number generated by adding up all the data in the file you want to check. If any of the characters change, the checksum will no longer be equal to the total, so a simple comparison will

allow you to spot the difference. (In practice, it is done in a more complicated way to spot problems like position-swapping, which will not show up with a simplistic addition).

- Q. I did what you said for compiling ports from a CDROM and it worked great until I tried to install the kermit port:-

```
# make install
>> ckul90.tar.gz doesn't seem to exist on this system.
>> Attempting to fetch from ftp://kermit.columbia.edu/kermit/archives/.
```

Why can it not be found? Have I got a dud CDROM?

A. The licensing terms for kermit do not allow us to put the tarball for it on the CDROM, so you will have to fetch it by hand — sorry! The reason why you got all those error messages was because you were not connected to the Internet at the time. Once you have downloaded it from any of the sites above, you can re-start the process (try and choose the nearest site to you, though, to save your time and the Internet's bandwidth).

- Q. I did that, but when I tried to put it into `/usr/ports/distfiles` I got some error about not having permission.

A. The ports mechanism looks for the tarball in `/usr/ports/distfiles`, but you will not be able to copy anything there because it is sym-linked to the CDROM, which is read-only. You can tell it to look somewhere else by doing

```
# make DISTDIR=/where/you/put/it install
```

- Q. Does the ports scheme only work if you have everything in `/usr/ports`? My system administrator says I must put everything under `/u/people/guests/wurzbürger`, but it does not seem to work.

A. You can use the `PORTSDIR` and `PREFIX` variables to tell the ports mechanism to use different directories. For instance,

```
# make PORTSDIR=/u/people/guests/wurzbürger/ports install
```

will compile the port in `/u/people/guests/wurzbürger/ports` and install everything under `/usr/local`.

```
# make PREFIX=/u/people/guests/wurzbürger/local install
```

will compile it in `/usr/ports` and install it in `/u/people/guests/wurzbürger/local`.

And of course

```
# make PORTSDIR=.../ports PREFIX=.../local install
```

will combine the two (it is too long to fit on the page if I write it in full, but I am sure you get the idea).

If you do not fancy typing all that in every time you install a port (and to be honest, who would?), it is a good idea to put these variables into your environment.

- Q. I do not have a FreeBSD CDROM, but I would like to have all the tarballs handy on my system so I do not have to wait for a download every time I install a port. Is there an easy way to get them all at once?

A. To get every single tarball for the ports collection, do

```
# cd /usr/ports
# make fetch
```

For all the tarballs for a single ports directory, do

```
# cd /usr/ports/directory
# make fetch
```

and for just one port — well, I think you have guessed already.

- Q. I know it is probably faster to fetch the tarballs from one of the FreeBSD mirror sites close by. Is there any way to tell the port to fetch them from servers other than ones listed in the MASTER\_SITES?

A. Yes. If you know, for example, `ftp.FreeBSD.ORG` is much closer than sites listed in `MASTER_SITES`, do as following example.

```
# cd /usr/ports/directory
# make MAS-
TER_SITE_OVERRIDE=ftp://ftp.FreeBSD.ORG/pub/FreeBSD/ports/distfiles/ fetch
```

- Q. I want to know what files make is going to need before it tries to pull them down.

A. `make fetch-list` will display a list of the files needed for a port.

- Q. Is there any way to stop the port from compiling? I want to do some hacking on the source before I install it, but it is a bit tiresome having to watch it and hit control-C every time.

A. Doing `make extract` will stop it after it has fetched and extracted the source code.

- Q. I am trying to make my own port and I want to be able to stop it compiling until I have had a chance to see if my patches worked properly. Is there something like `make extract`, but for patches?

A. Yep, `make patch` is what you want. You will probably find the `PATCH_DEBUG` option useful as well. And by the way, thank you for your efforts!

- Q. I have heard that some compiler options can cause bugs. Is this true? How can I make sure that I compile ports with the right settings?

A. Yes, with version 2.6.3 of `gcc` (the version shipped with FreeBSD 2.1.0 and 2.1.5), the `-O2` option could result in buggy code unless you used the `-fno-strength-reduce` option as well. (Most of the ports don't use `-O2`). You *should* be able to specify the compiler options used by something like

```
# make CFLAGS='-O2 -fno-strength-reduce' install
```

or by editing `/etc/make.conf`, but unfortunately not all ports respect this. The surest way is to do `make configure`, then go into the source directory and inspect the Makefiles by hand, but this can get tedious if the source has lots of sub-directories, each with their own Makefiles.

- Q. There are so many ports it is hard to find the one I want. Is there a list anywhere of what ports are available?

A. Look in the `INDEX` file in `/usr/ports`. If you would like to search the ports collection for a keyword, you can do that too. For example, you can find ports relevant to the LISP programming language using:

```
% cd /usr/ports
% make search key=lisp
```

- Q. I went to install the `foo` port but the system suddenly stopped compiling it and starting compiling the `bar` port. What's going on?

A. The `foo` port needs something that is supplied with `bar` — for instance, if `foo` uses graphics, `bar` might have a library with useful graphics processing routines. Or `bar` might be a tool that is needed to compile the `foo` port.

- Q. I installed the `grizzle` program from the ports and frankly it is a complete waste of disk space. I want to delete it but I do not know where it put all the files. Any clues?

A. No problem, just do

```
# pkg_delete grizzle-6.5
```

- Q. Hang on a minute, you have to know the version number to use that command. You do not seriously expect me to remember that, do you??

A. Not at all, you can find it out by doing

```
# pkg_info -a | grep grizzle
Information for grizzle-6.5:
grizzle-6.5 -
the combined piano tutorial, LOGO interpreter and shoot 'em up arcade game.
```

- Q. Talking of disk space, the ports directory seems to be taking up an awful lot of room. Is it safe to go in there and delete things?

A. Yes, if you have installed the program and are fairly certain you will not need the source again, there is no point in keeping it hanging around. The best way to do this is

```
# cd /usr/ports
# make clean
```

which will go through all the ports subdirectories and delete everything except the skeletons for each port.

- Q. I tried that and it still left all those tarballs or whatever you called them in the `distfiles` directory. Can I delete those as well?

A. Yes, if you are sure you have finished with them, those can go as well.

- Q. I like having lots and lots of programs to play with. Is there any way of installing all the ports in one go?

A. Just do

```
# cd /usr/ports
# make install
```

- Q. OK, I tried that, but I thought it would take a very long time so I went to bed and left it to get on with it. When I looked at the computer this morning, it had only done three and a half ports. Did something go wrong?

A. No, the problem is that some of the ports need to ask you questions that we cannot answer for you (eg “Do you want to print on A4 or US letter sized paper?”) and they need to have someone on hand to answer them.

- Q. I really do not want to spend all day staring at the monitor. Any better ideas?

A. OK, do this before you go to bed/work/the local park:-

```
# cd /usr/ports
# make -DBATCH install
```

This will install every port that does *not* require user input. Then, when you come back, do

```
# cd /usr/ports
# make -DIS_INTERACTIVE install
```

to finish the job.

- Q. At work, we are using `frobbles`, which is in your ports collection, but we have altered it quite a bit to get it to do what we need. Is there any way of making our own packages, so we can distribute it more easily around our sites?

A. No problem, assuming you know how to make patches for your changes:-

```
# cd /usr/ports/somewhere/frobbles
# make extract
# cd work/frobbles-2.8
[Apply your patches]
# cd ../../
# make package
```

- Q. This ports stuff is really clever. I am desperate to find out how you did it. What is the secret?

A. Nothing secret about it at all, just look at the `bsd.ports.mk` and `bsd.ports.subdir.mk` files in your makefiles directory. (`file://localhost/usr/share/mk/`)

**Note:** Readers with an aversion to intricate shell-scripts are advised not to follow this link...)

## Making a port yourself

*Contributed by Jordan K. Hubbard <jkh@FreeBSD.ORG>, Gary Palmer <gpalmer@FreeBSD.ORG>, Satoshi Asami <asami@FreeBSD.ORG> David O'Brien <obrien@FreeBSD.ORG> and Tim Vanderhoek <hoek@FreeBSD.ORG>. 28 August 1996.*

So, now you are interested in making your own port? Great!

What follows are some guidelines for creating a new port for FreeBSD. The bulk of the work is done by `/usr/share/mk/bsd.port.mk`, which all port Makefiles include. Please refer to that file for more details on the inner workings of the ports collection. Even if you don't hack Makefiles daily, it is well commented, and you will still gain much knowledge from it.

**Note:** Only a fraction of the overridable variables (`VAR`) are mentioned in this document. Most (if not all) are documented at the start of `bsd.port.mk`. This file uses a non-standard tab setting. **Emacs** and **Vim** should recognise the setting on loading the file. `vi` or `ex` can be set to use the correct value by typing `:set tabstop=4` once the file has been loaded.

## Quick Porting

This section tells you how to do a quick port. In many cases, it is not enough, but we will see.

First, get the original tarball and put it into `DISTDIR`, which defaults to `/usr/ports/distfiles`.

**Note:** The following assumes that the software compiled out-of-the-box, i.e., there was absolutely no change required for the port to work on your FreeBSD box. If you needed to change something, you will have to refer to the next section too.

## Writing the Makefile

The minimal Makefile would look something like this:

```
# New ports collection makefile for:   oneko
# Version required:                   1.1b
```

```
# Date created:      5 December 1994
# Whom:             asami
#
# $Id$
#

DISTNAME=          oneko-1.1b
CATEGORIES=        games
MASTER_SITES=      ftp://ftp.cs.columbia.edu/archives/X11R5/contrib/

MAINTAINER=        asami@FreeBSD.ORG

MAN1=              oneko.1
MANCOMPRESSED=     yes
USE_IMAKE=          yes

.include <bsd.port.mk>
```

See if you can figure it out. Do not worry about the contents of the `$Id$` line, it will be filled in automatically by CVS when the port is imported to our main ports tree. You can find a more detailed example in the sample Makefile section.

## Writing the description files

There are three description files that are required for any port, whether they actually package or not. They are `COMMENT`, `DESCR`, and `PLIST`, and reside in the `pkg` subdirectory.

### **COMMENT**

This is the one-line description of the port. *Please* do not include the package name (or version number of the software) in the comment. Here is an example:

```
A cat chasing a mouse all over the screen.
```

### **DESCR**

This is a longer description of the port. One to a few paragraphs concisely explaining what the port does is sufficient.

**Note:** This is *not* a manual or an in-depth description on how to use or compile the port! *Please be careful if you are copying from the `README` or `manpage`*; too often they are not a concise description

of the port or are in an awkward format (e.g., manpages have justified spacing). If the ported software has an official WWW homepage, you should list it here.

It is recommended that you sign your name at the end of this file, as in:

```
This is a port of oneko, in which a cat chases a poor mouse all over
the screen.
```

```
:
(etc.)
```

```
http://www.oneko.org/
```

```
- Satoshi
asami@cs.berkeley.edu
```

#### **PLIST**

This file lists all the files installed by the port. It is also called the “packing list” because the package is generated by packing the files listed here. The pathnames are relative to the installation prefix (usually `/usr/local` or `/usr/X11R6`). If you are using the `MANn` variables (as you should be), do not list any manpages here.

Here is a small example:

```
bin/oneko
lib/X11/app-defaults/Oneko
lib/X11/oneko/cat1.xpm
lib/X11/oneko/cat2.xpm
lib/X11/oneko/mouse.xpm
@dirrm lib/X11/oneko
```

Refer to the `pkg.create(1)` man page for details on the packing list.

**Note:** You should list all the files, but not the name directories, in the list. Also, if the port creates directories for itself during installation, make sure to add `@dirrm` lines as necessary to remove them when the port is deleted.

It is recommended that you keep all the filenames in this file sorted alphabetically. It will make verifying the changes when you upgrade the port much easier.

## Creating the checksum file

Just type `make makesum`. The ports make rules will automatically generate the file `files/md5`.

## Testing the port

You should make sure that the port rules do exactly what you want it to do, including packaging up the port. These are the important points you need to verify.

- `PLIST` does not contain anything not installed by your port
- `PLIST` contains everything that is installed by your port
- Your port can be installed multiple times using the `reinstall` target
- Your port cleans up after itself upon `deinstall`

## Recommended test ordering

1. `make install`
2. `make package`
3. `make deinstall`
4. `pkg_add `make package-name``
5. `make deinstall`
6. `make reinstall`
7. `make package`

Make sure that there aren't any warnings issued in any of the `package` and `deinstall` stages, After step 3, check to see if all the new directories are correctly deleted. Also, try using the software after step 4, to ensure that it works correctly when installed from a package.

## Checking your port with `portlint`

Please use `portlint` to see if your port conforms to our guidelines. The `portlint` program is part of the ports collection. In particular, you may want to check if the Makefile is in the right shape and the package is named appropriately.

## Submitting the port

First, make sure you have read the Do's and Dont's section.

Now that you are happy with your port, the only thing remaining is to put it in the main FreeBSD ports tree and make everybody else happy about it too. We do not need your `work` directory or the `pkgname.tgz` package, so delete them now. Next, simply include the output of `shar `find port_dir`` in a bug report and send it with the `send-pr(1)` program (see Bug Reports and General Commentary for more information about `send-pr(1)`). If the uncompressed port is larger than 20KB, you should compress it into a tarfile and use `uuencode(1)` before including it in the bug report (uuencoded tarfiles are acceptable even if the bug report is smaller than 20KB but are not preferred). Be sure to classify the bug report as category `ports` and class `change-request`. (Do not mark the report `confidential!`)

One more time, *do not include the original source distfile, the work directory, or the package you built with `make package`.*

**Note:** In the past, we asked you to upload new port submissions in our ftp site (`ftp.freebsd.org`). This is no longer recommended as read access is turned off on that `incoming/` directory of that site due to the large amount of pirated software showing up there.

We will look at your port, get back to you if necessary, and put it in the tree. Your name will also appear in the list of “Additional FreeBSD contributors” on the FreeBSD Handbook and other files. Isn't that great!?!? :)

## Slow Porting

Ok, so it was not that simple, and the port required some modifications to get it to work. In this section, we will explain, step by step, how to modify it to get it to work with the ports paradigm.

### How things work

First, this is the sequence of events which occurs when the user first types `make` in your port's directory, and you may find that having `bsd.port.mk` in another window while you read this really helps to understand it.

But do not worry if you do not really understand what `bsd.port.mk` is doing, not many people do... :>

1. The `fetch` target is run. The `fetch` target is responsible for making sure that the tarball exists locally in `DISTDIR`. If `fetch` cannot find the required files in `DISTDIR` it will look up the URL `MASTER_SITES`, which is set in the Makefile, as well as our main ftp site at

`ftp://ftp.freebsd.org/pub/FreeBSD/ports/distfiles/`, where we put sanctioned distfiles as backup. It will then attempt to fetch the named distribution file with `FETCH`, assuming that the requesting site has direct access to the Internet. If that succeeds, it will save the file in `DISTDIR` for future use and proceed.

2. The `extract` target is run. It looks for your port's distribution file (typically a gzip'd tarball) in `DISTDIR` and unpacks it into a temporary subdirectory specified by `WRKDIR` (defaults to `work`).
3. The `patch` target is run. First, any patches defined in `PATCHFILES` are applied. Second, if any patches are found in `PATCHDIR` (defaults to the `patches` subdirectory), they are applied at this time in alphabetical order.
4. The `configure` target is run. This can do any one of many different things.
  1. If it exists, `scripts/configure` is run.
  2. If `HAS_CONFIGURE` or `GNU_CONFIGURE` is set, `WRKSRV/configure` is run.
  3. If `USE_IMAKE` is set, `XMKMF` (default: `xmkmf -a`) is run.
5. The `build` target is run. This is responsible for descending into the port's private working directory (`WRKSRV`) and building it. If `USE_GMAKE` is set, GNU `make` will be used, otherwise the system `make` will be used.

The above are the default actions. In addition, you can define targets `pre-something` or `post-something`, or put scripts with those names, in the `scripts` subdirectory, and they will be run before or after the default actions are done.

For example, if you have a `post-extract` target defined in your Makefile, and a file `pre-build` in the `scripts` subdirectory, the `post-extract` target will be called after the regular extraction actions, and the `pre-build` script will be executed before the default build rules are done. It is recommended that you use Makefile targets if the actions are simple enough, because it will be easier for someone to figure out what kind of non-default action the port requires.

The default actions are done by the `bsd.port.mk` targets `do-something`. For example, the commands to extract a port are in the target `do-extract`. If you are not happy with the default target, you can fix it by redefining the `do-something` target in your Makefile.

**Note:** The "main" targets (e.g., `extract`, `configure`, etc.) do nothing more than make sure all the stages up to that one are completed and call the real targets or scripts, and they are not intended to be changed. If you want to fix the extraction, fix `do-extract`, but never ever touch `extract`!

Now that you understand what goes on when the user types `make`, let us go through the recommended steps to create the perfect port.

## Getting the original sources

Get the original sources (normally) as a compressed tarball (`foo.tar.gz` or `foo.tar.Z`) and copy it into `DISTDIR`. Always use *mainstream* sources when and where you can.

If you cannot find a ftp/http site that is well-connected to the net, or can only find sites that have irritatingly non-standard formats, you might want to put a copy on a reliable ftp or http server that you control (e.g., your home page). Make sure you set `MASTER_SITES` to reflect your choice.

If you cannot find somewhere convenient and reliable to put the distfile (if you are a FreeBSD committer, you can just put it in your `public_html/` directory on `freefall`), we can “house” it ourselves by putting it on `ftp://ftp.freebsd.org/pub/FreeBSD/ports/distfiles/LOCAL_PORTS/` as the last resort. Please refer to this location as `MASTER_SITE_LOCAL`. Send mail to the FreeBSD ports mailing list `<freebsd-ports@FreeBSD.ORG>` if you are not sure what to do.

If your port’s distfile changes all the time for no good reason, consider putting the distfile in your home page and listing it as the first `MASTER_SITES`. This will prevent users from getting checksum mismatch errors, and also reduce the workload of maintainers of our ftp site. Also, if there is only one master site for the port, it is recommended that you house a backup at your site and list it as the second `MASTER_SITES`.

If your port requires some additional ‘patches’ that are available on the Internet, fetch them too and put them in `DISTDIR`. Do not worry if they come from a site other than where you got the main source tarball, we have a way to handle these situations (see the description of `PATCHFILES` below).

## Modifying the port

Unpack a copy of the tarball in a private directory and make whatever changes are necessary to get the port to compile properly under the current version of FreeBSD. Keep *careful track* of everything you do, as you will be automating the process shortly. Everything, including the deletion, addition or modification of files should be doable using an automated script or patch file when your port is finished.

If your port requires significant user interaction/customization to compile or install, you should take a look at one of Larry Wall’s classic **Configure** scripts and perhaps do something similar yourself. The goal of the new ports collection is to make each port as “plug-and-play” as possible for the end-user while using a minimum of disk space.

**Note:** Unless explicitly stated, patch files, scripts, and other files you have created and contributed to the FreeBSD ports collection are assumed to be covered by the standard BSD copyright conditions.

## Patching

In the preparation of the port, files that have been added or changed can be picked up with a recursive

diff for later feeding to patch. Each set of patches you wish to apply should be collected into a file named `patch-xx` where `xx` denotes the sequence in which the patches will be applied — these are done in *alphabetical order*, thus `aa` first, `ab` second and so on. These files should be stored in `PATCHDIR`, from where they will be automatically applied. All patches should be relative to `WRKSRCDIR` (generally the directory your port's tarball unpacks itself into, that being where the build is done). To make fixes and upgrades easier, you should avoid having more than one patch fix the same file (e.g., `patch-aa` and `patch-ab` both changing `WRKSRCDIR/fooobar.c`).

## Configuring

Include any additional customization commands to your `configure` script and save it in the `scripts` subdirectory. As mentioned above, you can also do this as `Makefile` targets and/or scripts with the name `pre-configure` or `post-configure`.

## Handling user input

If your port requires user input to build, configure or install, then set `IS_INTERACTIVE` in your `Makefile`. This will allow “overnight builds” to skip your port if the user sets the variable `BATCH` in his environment (and if the user sets the variable `INTERACTIVE`, then *only* those ports requiring interaction are built).

It is also recommended that if there are reasonable default answers to the questions, you check the `PACKAGE_BUILDING` variable and turn off the interactive script when it is set. This will allow us to build the packages for CD-ROMs and ftp.

## Configuring the Makefile

Configuring the `Makefile` is pretty simple, and again we suggest that you look at existing examples before starting. Also, there is a sample `Makefile` in this handbook, so take a look and please follow the ordering of variables and sections in that template to make your port easier for others to read.

Now, consider the following problems in sequence as you design your new `Makefile`:

### The original source

Does it live in `DISTDIR` as a standard gzip'd tarball? If so, you can go on to the next step. If not, you should look at overriding any of the `EXTRACT_CMD`, `EXTRACT_BEFORE_ARGS`, `EXTRACT_AFTER_ARGS`, `EXTRACT_SUFFIX`, or `DISTFILES` variables, depending on how alien a format your port's distribution file

is. (The most common case is `EXTRACT_SUFFIX=.tar.Z`, when the tarball is condensed by regular compress, not gzip.)

In the worst case, you can simply create your own `do-extract` target to override the default, though this should be rarely, if ever, necessary.

### **DISTNAME**

You should set `DISTNAME` to be the base name of your port. The default rules expect the distribution file list (`DISTFILES`) to be named `DISTNAMEEXTRACT_SUFFIX` which, if it is a normal tarball, is going to be something like `foozolix-1.0.tar.gz` for a setting of `DISTNAME=foozolix-1.0`.

The default rules also expect the tarball(s) to extract into a subdirectory called `work/DISTNAME`, e.g. `work/foozolix-1.0/`.

All this behavior can be overridden, of course; it simply represents the most common time-saving defaults. For a port requiring multiple distribution files, simply set `DISTFILES` explicitly. If only a subset of `DISTFILES` are actual extractable archives, then set them up in `EXTRACT_ONLY`, which will override the `DISTFILES` list when it comes to extraction, and the rest will be just left in `DISTDIR` for later use.

### **PKGNAME**

If `DISTNAME` does not conform to our guidelines for a good package name, you should set the `PKGNAME` variable to something better. See the abovementioned guidelines for more details.

### **CATEGORIES**

When a package is created, it is put under `/usr/ports/packages/All` and links are made from one or more subdirectories of `/usr/ports/packages`. The names of these subdirectories are specified by the variable `CATEGORIES`. It is intended to make life easier for the user when he is wading through the pile of packages on the ftp site or the CD-ROM. Please take a look at the existing categories and pick the ones that are suitable for your port.

This list also determines where in the ports tree the port is imported. If you put more than one category here, it is assumed that the port files will be put in the subdirectory with the name in the first category. See the categories section for more discussion about how to pick the right categories.

If your port truly belongs to something that is different from all the existing ones, you can even create a new category name. In that case, please send mail to the FreeBSD ports mailing list `<freebsd-ports@FreeBSD.ORG>` to propose a new category.

**Note:** There is no error checking for category names. `make package` will happily create a new directory if you mustype the category name, so be careful!

### MASTER\_SITES

Record the directory part of the ftp/http-URL pointing at the original tarball in `MASTER_SITES`. Do not forget the trailing slash (/)!

The `make` macros will try to use this specification for grabbing the distribution file with `FETCH` if they cannot find it already on the system.

It is recommended that you put multiple sites on this list, preferably from different continents. This will safeguard against wide-area network problems, and we are even planning to add support for automatically determining the closest master site and fetching from there!

If the original tarball is part of one of the following popular archives: X-contrib, GNU, Perl CPAN, TeX CTAN, or Linux Sunsite, you refer to those sites in an easy compact form using `MASTER_SITE_XCONTRIB`, `MASTER_SITE_GNU`, `MASTER_SITE_PERL_CPAN`, `MASTER_SITE_TEX_CTAN`, and `MASTER_SITE_SUNSITE`. Simply set `MASTER_SITE_SUBDIR` to the path within the archive. Here is an example:

```
MASTER_SITES=      ${MASTER_SITE_XCONTRIB}
MASTER_SITE_SUBDIR= applications
```

The user can also set the `MASTER_SITE_*` variables in `/etc/make.conf` to override our choices, and use their favorite mirrors of these popular archives instead.

### PATCHFILES

If your port requires some additional patches that are available by ftp or http, set `PATCHFILES` to the names of the files and `PATCH_SITES` to the URL of the directory that contains them (the format is the same as `MASTER_SITES`).

If the patch is not relative to the top of the source tree (i.e., `WKR SRC`) because it contains some extra pathnames, set `PATCH_DIST_STRIP` accordingly. For instance, if all the pathnames in the patch have an extra `foozolicx-1.0/` in front of the filenames, then set `PATCH_DIST_STRIP=-p1`.

Do not worry if the patches are compressed, they will be decompressed automatically if the filenames end with `.gz` or `.Z`.

If the patch is distributed with some other files, such as documentation, in a gzip'd tarball, you can't just use `PATCHFILES`. If that is the case, add the name and the location of the patch tarball to `DISTFILES`

and `MASTER_SITES`. Then, from the `pre-patch` target, apply the patch either by running the `patch` command from there, or copying the patch file into the `PATCHDIR` directory and calling it `patch-xx`.

**Note:** Note the tarball will have been extracted alongside the regular source by then, so there is no need to explicitly extract it if it is a regular gzip'd or compress'd tarball. If you do the latter, take extra care not to overwrite something that already exists in that directory. Also do not forget to add a command to remove the copied patch in the `pre-clean` target.

## MAINTAINER

Set your mail-address here. Please. :)

For detailed description of the responsibility of maintainers, refer to `MAINTAINER` on Makefiles section.

## Dependencies

Many ports depend on other ports. There are five variables that you can use to ensure that all the required bits will be on the user's machine. There are also some pre-supported dependency variables for common cases, plus a few more to control the behaviour of dependencies.

### LIB\_DEPENDS

This variable specifies the shared libraries this port depends on. It is a list of `lib:dir[:target]` tuples where `lib` is the name of the shared library, and `dir` is the directory in which to find it in case it is not available, and `target` is the target to call in that directory. For example,

```
LIB_DEPENDS=
    jpeg\\.9\\. :${PORTSDIR}/graphics/jpeg:install
```

will check for a shared jpeg library with major version 9, and descend into the `graphics/jpeg` subdirectory of your ports tree to build and install it if it is not found. The `target` part can be omitted if it is equal to `DEPENDS_TARGET` (which defaults to `install`).

**Note:** The `lib` part is an argument given to `ldconfig -r | grep -wF`. There shall be no regular expressions in this variable.

The dependency is checked twice, once from within the `extract` target and then from within the `install` target. Also, the name of the dependency is put in to the package so that `pkg_add` will automatically install it if it is not on the user's system.

**RUN\_DEPENDS**

This variable specifies executables or files this port depends on during run-time. It is a list of *path:dir[:target]* tuples where *path* is the name of the executable or file, and *dir* is the directory in which to find it in case it is not available, and *target* is the target to call in that directory. If *path* starts with a slash (/), it is treated as a file and its existence is tested with `test -e`; otherwise, it is assumed to be an executable, and `which -s` is used to determine if the program exists in the user's search path.

For example,

```
RUN_DEPENDS=  ${PREFIX}/etc/innd:${PORTSDIR}/news/inn \
              wish8.0:${PORTSDIR}/x11-toolkits/tk80
```

will check if the file or directory `/usr/local/etc/innd` exists, and build and install it from the `news/inn` subdirectory of the ports tree if it is not found. It will also see if an executable called `wish8.0` is in your search path, and descend into the `x11-toolkits/tk80` subdirectory of your ports tree to build and install it if it is not found.

**Note:** In this case, `innd` is actually an executable; if an executable is in a place that is not expected to be in a normal user's search path, you should use the full pathname.

The dependency is checked from within the `install` target. Also, the name of the dependency is put in to the package so that `pkg_add` will automatically install it if it is not on the user's system. The *target* part can be omitted if it is the same `DEPENDS_TARGET`.

**BUILD\_DEPENDS**

This variable specifies executables or files this port requires to build. Like `RUN_DEPENDS`, it is a list of *path:dir[:target]* tuples. For example,

```
BUILD_DEPENDS=
              unzip:${PORTSDIR}/archivers/unzip
```

will check for an executable called `unzip`, and descend into the `archivers/unzip` subdirectory of your ports tree to build and install it if it is not found.

**Note:** "build" here means everything from extracting to compilation. The dependency is checked from within the `extract` target. The *target* part can be omitted if it is the same as `DEPENDS_TARGET`

### **FETCH\_DEPENDS**

This variable specifies executables or files this port requires to fetch. Like the previous two, it is a list of *path:dir[:target]* tuples. For example,

```
FETCH_DEPENDS=  
ncftp2:${PORTSDIR}/net/ncftp2
```

will check for an executable called `ncftp2`, and descend into the `net/ncftp2` subdirectory of your ports tree to build and install it if it is not found.

The dependency is checked from within the `fetch` target. The *target* part can be omitted if it is the same as `DEPENDS_TARGET`.

### **DEPENDS**

If there is a dependency that does not fall into either of the above four categories, or your port requires to have the source of the other port extracted in addition to having them installed, then use this variable.

This is a list of *dir[:target]*, as there is nothing to check, unlike the previous four. The *target* part can be omitted if it is the same as `DEPENDS_TARGET`.

### **Common dependency variables**

Define `USE_XLIB=yes` if your port requires the X Window System to be installed (it is implied by `USE_IMAKE`). Define `USE_GMAKE=yes` if your port requires GNU `make` instead of BSD `make`. Define `USE_AUTOCONF=yes` if your port requires GNU `autoconf` to be run. Define `USE_QT=yes` if your port uses the latest `qt` toolkit. Use `USE_PERL5=yes` if your port requires version 5 of the perl language. (The last is especially important since some versions of FreeBSD has `perl5` as part of the base system while others don't.)

### **Notes on dependencies**

As mentioned above, the default target to call when a dependency is required is `DEPENDS_TARGET`. It defaults to `install`. This is a user variable; is is never defined in a port's `Makefile`. If your port needs a special way to handle a dependency, use the *:target* part of the `*_DEPENDS` variables instead of redefining `DEPENDS_TARGET`.

When you type `make clean`, its dependencies are automatically cleaned too. If you do not wish this to happen, define the variable `NOCLEANDEPENDS` in your environment.

To depend on another port unconditionally, it is customary to use the string `nonexistent` as the first field of `BUILD_DEPENDS` or `RUN_DEPENDS`. Use this only when you need to the to get to the source of the other port. You can often save compilation time by specifying the target too. For instance

```
BUILD_DEPENDS= /nonexistent:${PORTSDIR}/graphics/jpeg:extract
```

will always descend to the JPEG port and extract it.

Do not use `DEPENDS` unless there is no other way the behaviour you want can be accomplished. It will cause the other port to be always build (and installed, by default), and the dependency will go into the packages as well. If this is really what you need, I recommend you write it as `BUILD_DEPENDS` and `RUN_DEPENDS` instead—at least the intention will be clear.

## Building mechanisms

If your package uses GNU make, set `USE_GMAKE=yes`. If your package uses `configure`, set `HAS_CONFIGURE=yes`. If your package uses GNU `configure`, set `GNU_CONFIGURE=yes` (this implies `HAS_CONFIGURE`). If you want to give some extra arguments to `configure` (the default argument list `-prefix=${PREFIX}` for GNU `configure` and empty for non-GNU `configure`), set those extra arguments in `CONFIGURE_ARGS`. If your package uses GNU `autoconf`, set `USE_AUTOCONF=yes`. This implies `GNU_CONFIGURE`, and will cause `autoconf` to be run before `configure`.

If your package is an X application that creates `Makefiles` from `Imakefiles` using `imake`, then set `USE_IMAKE=yes`. This will cause the `configure` stage to automatically do an `xmkmf -a`. If the `-a` flag is a problem for your port, set `XMKMF=xmkmf`. If the port uses `imake` but does not understand the `install.man` target, `NO_INSTALL_MANPAGES=yes` should be set. In addition, the author of the original port should be shot. :>

If your port's source `Makefile` has something else than `all` as the main build target, set `ALL_TARGET` accordingly. Same goes for `install` and `INSTALL_TARGET`.

## Special considerations

There are some more things you have to take into account when you create a port. This section explains the most common of those.

### `ldconfig`

If your port installs a shared library, add a `post-install` target to your `Makefile` that runs `${LDCONFIG} -m` on the directory where the new library is installed (usually `PREFIX/lib`) to register it into the shared library cache.

Also, add a matching `@exec /sbin/ldconfig -m` and `@unexec /sbin/ldconfig -R` pair to your `pkg/PLIST` file so that a user who installed the package can start using the shared library immediately

and deinstallation will not cause the system to still believe the library is there. These lines should immediately follow the line for the shared library itself, as in:

```
lib/libtv180.so.1
@exec /sbin/ldconfig -m %D/lib
@unexec /sbin/ldconfig -R
```

Never, ever, *ever* add a line that says `ldconfig` without any arguments to your `Makefile` or `pkg/PLIST`. This will reset the shared library cache to the contents of `/usr/lib` only, and will royally screw up the user's machine ("Help, xinit does not run anymore after I install this port!"). Anybody who does this will be shot and cut in 65,536 pieces by a rusty knife and have his liver chopped out by a bunch of crows and will eternally rot to death in the deepest bowels of hell (not necessarily in that order...)

## ELF support

Since FreeBSD is moving to ELF shortly after 3.0-RELEASE, we need to convert many ports that build shared libraries to support ELF. Complicating this task is that a 3.0 system can run as both ELF and a.out, and we wish to unofficially support the 2.2 as long as possible. Below are the guidelines on how to convert a.out only ports to support both a.out and ELF compilation.

Some part of this list is only applicable during the conversion, but will be left here for awhile for reference in case you have come across some old port you wish to upgrade.

### Moving a.out libraries out of the way

A.out libraries should be moved out of `/usr/local/lib` and similar to an a.out subdirectory. (If you don't move them out of the way, ELF ports will happily overwrite a.out libraries.) The `move-aout-libs` target in the 3.0-CURRENT `src/Makefile` (called from `aout-to-elf`) will do this for you. It will only move a.out libs so it is safe to call it on a system with both ELF and a.out libs in the standard directories.

### Format

The ports tree will build packages in the format the machine is in. This means a.out for 2.2 and a.out or ELF for 3.0 depending on what `objformat` returns. Also, once users move a.out libraries to a subdirectory, building a.out libraries will be unsupported. (I.e., it may still work if you know what you are doing, but you are on your own.)

**Note:** If a port only works for a.out, set `BROKEN_ELF` to a string describing the reason why. Such ports will be skipped during a build on an ELF system.

### PORTOBJFORMAT

`bsd.port.mk` will set `PORTOBJFORMAT` to `aout` or `elf` and export it in the environments `CONFIGURE_ENV`, `SCRIPTS_ENV` and `MAKE_ENV`. (It's always going to be `aout` in 2.2-STABLE). It is also passed to `PLIST_SUB` as `PORTOBJFORMAT=${PORTOBJFORMAT}`. (See comment on `ldconfig` lines below.)

The variable is set using this line in `bsd.port.mk`:

```
PORTOBJFORMAT!= test -x /usr/bin/objformat && /usr/bin/objformat || echo aout
```

Ports' make processes should use this variable to decide what to do. However, if the port's configure script already automatically detects an ELF system, it is not necessary to refer to `PORTOBJFORMAT`.

## Building shared libraries

The following are differences in handling shared libraries for a.out and ELF.

- Shared library versions

An ELF shared library should be called `libfoo.so.M` where `M` is the single version number, and an a.out library should be called `libfoo.so.M.N` where `M` is the major version and `N` is the minor version number. Do not mix those; *never* install an ELF shared library called `libfoo.so.N.M` or an a.out shared library (or symlink) called `libfoo.so.N`.

- Linker command lines

Assuming `cc -shared` is used rather than `ld` directly, the only difference is that you need to add `-Wl,-soname,libfoo.so.M` on the command line for ELF.

You need to install a symlink from `libfoo.so` to `libfoo.so.N` to make ELF linkers happy. Since it should be listed in `PLIST` too, and it won't hurt in the a.out case (some ports even require the link for dynamic loading), you should just make this link regardless of the setting of `PORTOBJFORMAT`.

### LIB\_DEPENDS

All port Makefiles are edited to remove minor numbers from `LIB_DEPENDS`, and also to have the regexp support removed. (E.g., `foo\1\.\(33|40\)` becomes `foo.2`.) They will be matched using `grep`

-wF.

## PLIST

PLIST should contain the short (ELF) shlib names if the a.out minor number is zero, and the long (a.out) names otherwise. `bsd.port.mk` will automatically add `.0` to the end of short shlib lines if `PORTOBJFORMAT` equals `aout`, and will delete the minor number from long shlib names if `PORTOBJFORMAT` equals `elf`.

In cases where you really need to install shlibs with two versions on an ELF system or those with one version on an a.out system (for instance, ports that install compatibility libraries for other operating systems), define the variable `NO_FILTER_SHLIBS`. This will turn off the editing of `PLIST` mentioned in the previous paragraph.

## ldconfig

The `ldconfig` line in Makefiles should read:

```
${SETENV} OBJFORMAT=${PORTOBJFORMAT} ${LDCONFIG} -m ....
```

In `PLIST` it should read;

```
@exec /usr/bin/env OBJFORMAT=%PORTOBJFORMAT% /sbin/ldconfig -m ...  
@unexec /usr/bin/env OBJFORMAT=%PORTOBJFORMAT% /sbin/ldconfig -R
```

This is to ensure that the correct `ldconfig` will be called depending on the format of the package, not the default format of the system.

## MASTERDIR

If your port needs to build slightly different versions of packages by having a variable (for instance, resolution, or paper size) take different values, create one subdirectory per package to make it easier for users to see what to do, but try to share as many files as possible between ports. Typically you only need a very short `Makefile` in all but one of the directories if you use variables cleverly. In the sole `Makefiles`, you can use `MASTERDIR` to specify the directory where the rest of the files are. Also, use a variable as part of `PKGNAME` so the packages will have different names.

This will be best demonstrated by an example. This is part of `japanese/xdvi300/Makefile`;

```
PKGNAME=          ja-xdvi${RESOLUTION}-17
```

```

:
# default
RESOLUTION?= 300
.if ${RESOLUTION} != 118 && ${RESOLUTION} != 240 && \
    ${RESOLUTION} != 300 && ${RESOLUTION} != 400
    @${ECHO} "Error: invalid value for RESOLUTION: \"${RESOLUTION}\""
    @${ECHO} "Possible values are: 118, 240, 300 (default) and 400."
    @${FALSE}
.endif

```

japanese/xdvi300 also has all the regular patches, package files, etc. If you type make there, it will take the default value for the resolution (300) and build the port normally.

As for other resolutions, this is the *entire* xdvi118/Makefile;

```

RESOLUTION= 118
MASTERDIR= ${.CURDIR}/../xdvi300

.include ${MASTERDIR}/Makefile

```

(xdvi240/Makefile and xdvi400/Makefile are similar). The MASTERDIR definition tells bsd.port.mk that the regular set of subdirectories like PATCHDIR and PKGDIR are to be found under xdvi300. The RESOLUTION=118 line will override the RESOLUTION=300 line in xdvi300/Makefile and the port will be built with resolution set to 118.

## Shared library versions

First, please read our policy on shared library versioning to understand what to do with shared library versions in general. Do not blindly assume software authors know what they are doing; many of them do not. It is very important that these details are carefully considered, as we have quite a unique situation where we are trying to have dozens of potentially incompatible software pairs co-exist. Careless port imports have caused great trouble regarding shared libraries in the past (ever wondered why the port jpeg-6b has a shared library version of 9.0?). If in doubt, send a message to the FreeBSD ports mailing list <freebsd-ports@FreeBSD.ORG>. Most of the time, your job ends by determining the right shared library version and making appropriate patches to implement it.

However, if there is a port which is a different version of the same software already in the tree, the situation is much more complex. In short, the FreeBSD implementation does not allow the user to specify to the linker which version of shared library to link against (the linker will always pick the highest numbered version). This means, if there is a libfoo.so.3.2 and libfoo.so.4.0 in the system, there is no way to tell the linker to link a particular application to libfoo.so.3.2. It is essentially completely overshadowed in terms of compilation-time linkage. In this case, the only solution

is to rename the *base* part of the shared library. For instance, change `libfoo.so.4.0` to `libfoo4.so.1.0` so both version 3.2 and 4.0 can be linked from other ports.

## Manpages

The `MAN[1-9LN]` variables will automatically add any manpages to `pkg/PLIST` (this means you must *not* list manpages in the `PLIST`—see generating `PLIST` for more). It also makes the install stage automatically compress or uncompress manpages depending on the setting of `NOMANCOMPRESS` in `/etc/make.conf`.

To specify whether the manpages are compressed upon installation, use the `MANCOMPRESSED` variable. This variable can take three values, `yes`, `no` and `maybe`. `yes` means manpages are already installed compressed, `no` means they are not, and `maybe` means the software already respects the value of `NOMANCOMPRESS` so `bsd.port.mk` does not have to do anything special.

`MANCOMPRESSED` is automatically set to `yes` if `USE_IMAKE` is set and `NO_INSTALL_MANPAGES` is not set, and to `no` otherwise. You don't have to explicitly define it unless the default is not suitable for your port.

If your port anchors its man tree somewhere other than `PREFIX`, you can use the `MANPREFIX` to set it. Also, if only manpages in certain sections go in a non-standard place, such as some Perl modules ports, you can set individual man paths using `MANsectPREFIX` (where *sect* is one of 1-9, L or N).

If your manpages go to language-specific subdirectories, set the name of the languages to `MANLANG`. The value of this variable defaults to `" "` (i.e., English only).

Here is an example that puts it all together.

```
MAN1=          foo.1
MAN3=          bar.3
MAN4=          baz.4
MANLANG=       " " ja
MAN3PREFIX=    ${PREFIX}/share/foobar
MANCOMPRESSED= yes
```

This states that six files are installed by this port;

```
${PREFIX}/man/man1/foo.1.gz
${PREFIX}/man/ja/man1/foo.1.gz
${PREFIX}/share/foobar/man/man3/bar.3.gz
${PREFIX}/share/foobar/man/ja/man3/bar.3.gz
${PREFIX}/man/man4/baz.4.gz
${PREFIX}/man/ja/man4/baz.4.gz
```

## Ports that require Motif

There are many programs that require a Motif library (available from several commercial vendors, while there is a free clone reported to be able to run many applications in `x11-toolkits/lesstif`) to compile. Since it is a popular toolkit and their licenses usually permit redistribution of statically linked binaries, we have made special provisions for handling ports that require Motif in a way that we can easily compile binaries linked either dynamically (for people who are compiling from the port) or statically (for people who distribute packages).

### **REQUIRES\_MOTIF**

If your port requires Motif, define this variable in the Makefile. This will prevent people who don't own a copy of Motif from even attempting to build it.

### **MOTIFLIB**

This variable will be set by `bsd.port.mk` to be the appropriate reference to the Motif library. Please patch the source to use this wherever the Motif library is referenced in the Makefile or Imakefile.

There are two common cases:

- If the port refers to the Motif library as `-lXm` in its Makefile or Imakefile, simply substitute `${MOTIFLIB}` for it.
- If the port uses `XmClientLibs` in its Imakefile, change it to `${MOTIFLIB} ${XTOOLLIB} ${XLIB}`.

Note that `MOTIFLIB` (usually) expands to `-L/usr/X11R6/lib -lXm` or `/usr/X11R6/lib/libXm.a`, so there is no need to add `-L` or `-l` in front.

## X11 fonts

If your port installs fonts for the X Window system, put them in `X11BASE/lib/X11/fonts/local`. This directory is new to XFree86 release 3.3.3. If it does not exist, please create it, and print out a message urging the user to update their XFree86 to 3.3.3 or newer, or at least add this directory to the font path in `/etc/XF86Config`.

## Info files

The new version of texinfo (included in 2.2.2-RELEASE and onwards) contains a utility called `install-info` to add and delete entries to the `dir` file. If your port installs any info documents, please follow this instructions so your port/package will correctly update the user's `PREFIX/info/dir` file. (Sorry for the length of this section, but is it imperative to weave all the info files together. If done correctly, it will produce a *beautiful* listing, so please bear with me!

First, this is what you (as a porter) need to know

```
% install-info -help
install-info [OPTION]... [INFO-FILE [DIR-FILE]]
  Install INFO-FILE in the Info directory file DIR-FILE.

Options:
-delete          Delete existing entries in INFO-FILE;
                  don't insert any new entries.
:
-entry=TEXT      Insert TEXT as an Info directory entry.
:
-section=SEC     Put this file's entries in section SEC of the directory. :
```

**Note:** This program will not actually *install* info files; it merely inserts or deletes entries in the `dir` file.

Here's a seven-step procedure to convert ports to use `install-info`. I will use `editors/emacs` as an example.

1. Look at the texinfo sources and make a patch to insert `@dircategory` and `@direntry` statements to files that don't have them. This is part of my patch:

```
-- ./man/vip.texi.org  Fri Jun 16 15:31:11 1995
+++ ./man/vip.texi      Tue May 20 01:28:33 1997
@@ -2,6 +2,10 @@

    @setfilename ../info/vip
    @settitle VIP
+@dircategory The Emacs editor and associated tools
+@direntry
+* VIP: (vip).          A VI-emulation for Emacs.
+@end direntry

@iftex
@finalout
:
```

The format should be self-explanatory. Many authors leave a `dir` file in the source tree that contains all the entries you need, so look around before you try to write your own. Also, make sure you look into related ports and make the section names and entry indentations consistent (we recommend that all entry text start at the 4th tab stop).

**Note:** Note that you can put only one info entry per file because of a bug in `install-info -delete` that deletes only the first entry if you specify multiple entries in the `<@direntry>` section.

You can give the `dir` entries to `install-info` as arguments (`-section` and `-entry`) instead of patching the texinfo sources. I do not think this is a good idea for ports because you need to duplicate the same information in *three* places (`Makefile` and `@exec/@unexec` of `PLIST`; see below). However, if you have a Japanese (or other multibyte encoding) info files, you will have to use the extra arguments to `install-info` because `makeinfo` can't handle those texinfo sources. (See `Makefile` and `PLIST` of `japanese/skk` for examples on how to do this).

2. Go back to the port directory and do a `make clean; make` and verify that the info files are regenerated from the texinfo sources. Since the texinfo sources are newer than the info files, they should be rebuilt when you type `make`; but many `Makefiles` don't include correct dependencies for info files. In `emacs`' case, I had to patch the main `Makefile.in` so it will descend into the `man` subdirectory to rebuild the info pages.

```
-- ./Makefile.in.org   Mon Aug 19 21:12:19 1996
+++ ./Makefile.in     Tue Apr 15 00:15:28 1997
@@ -184,7 +184,7 @@
 # Subdirectories to make recursively.  'lisp' is not included
 # because the compiled lisp files are part of the distribution
 # and you cannot remake them without installing Emacs first.
-SUBDIR = lib-src src
+SUBDIR = lib-src src man

# The makefiles of the directories in $SUBDIR.
SUBDIR_MAKEFILES = lib-
src/Makefile man/Makefile src/Makefile oldXMenu/Makefile lwlib/Makefile
-- ./man/Makefile.in.org   Thu Jun 27 15:27:19 1996
+++ ./man/Makefile.in     Tue Apr 15 00:29:52 1997
@@ -66,6 +66,7 @@
 ${srcdir}/gnul.texi \
 ${srcdir}/glossary.texi

+all: info
 info: $(INFO_TARGETS)

 dvi: $(DVI_TARGETS)
```

The second hunk was necessary because the default target in the man subdir is called `info`, while the main `Makefile` wants to call `all`. I also deleted the installation of the `info` info file because we already have one with the same name in `/usr/share/info` (that patch is not shown here).

3. If there is a place in the `Makefile` that is installing the `dir` file, delete it. Your port may not be doing it. Also, remove any commands that are otherwise mucking around with the `dir` file.

```
-- ./Makefile.in.org   Mon Aug 19 21:12:19 1996
+++ ./Makefile.in     Mon Apr 14 23:38:07 1997
@@ -368,14 +368,8 @@
         if [ `(cd ${srcdir}/info && /bin/pwd)` != `(cd ${infodir} && /bin/pwd)` ]; \
         then \
             (cd ${infodir}; \
-             if [ -f dir ]; then \
-                 if [ ! -f dir.old ]; then mv -f dir dir.old; \
-                 else mv -f dir dir.bak; fi; \
-             fi; \
             cd ${srcdir}/info ; \
-
             (cd ${thisdir}; ${INSTALL_DATA} ${srcdir}/info/dir ${infodir}/dir); \
-             (cd ${thisdir}; chmod a+r ${infodir}/dir); \
             for f in ccmode* cl* dired-
x* ediff* emacs* forms* gnus* info* message* mh-e* sc* vip*; do \
                (cd ${thisdir}; \
                    ${INSTALL_DATA} ${srcdir}/info/$f ${infodir}/$f; \
                    chmod a+r ${infodir}/$f); \
```

4. (This step is only necessary if you are modifying an existing port.) Take a look at `pkg/PLIST` and delete anything that is trying to patch up `info/dir`. They may be in `pkg/INSTALL` or some other file, so search extensively.

```
Index: pkg/PLIST
=====
RCS file: /usr/cvs/ports/editors/emacs/pkg/PLIST,v
retrieving revision 1.15
diff -u -r1.15 PLIST
-- PLIST          1997/03/04 08:04:00      1.15
+++ PLIST          1997/04/15 06:32:12
@@ -15,9 +15,6 @@
     man/man1/emacs.1.gz
     man/man1/etags.1.gz
     man/man1/ctags.1.gz
-@unexec cp %D/info/dir %D/info/dir.bak
-info/dir
-@unexec cp %D/info/dir.bak %D/info/dir
```

```

info/cl
info/cl-1
info/cl-2

```

5. Add a `post-install` target to the Makefile to create a `dir` file if it is not there. Also, call `install-info` with the installed info files.

```

Index: Makefile
=====
RCS file: /usr/cvs/ports/editors/emacs/Makefile,v
retrieving revision 1.26
diff -u -r1.26 Makefile
-- Makefile      1996/11/19 13:14:40      1.26
+++ Makefile      1997/05/20 10:25:09      1.28
@@ -20,5 +20,11 @@
  post-install:
    .for file in emacs-19.34 emacsclient etags ctags b2m
      strip ${PREFIX}/bin/${file}
    .endfor
+   if [ ! -f ${PREFIX}/info/dir ]; then \
+     ${SED} -
ne '1,/Menu:/p' /usr/share/info/dir > ${PREFIX}/info/dir; \
+   fi
+.for info in emacs vip viper forms gnus mh-e cl sc dired-x ediff ccmode
+  install-info ${PREFIX}/info/${info} ${PREFIX}/info/dir
+.endfor

  .include <bsd.port.mk>

```

Do not use anything other than `/usr/share/info/dir` and the above command to create a new info file. In fact, I'd add the first three lines of the above patch to `bsd.port.mk` if you (the porter) wouldn't have to do it in `PLIST` by yourself anyway.

6. Edit `PLIST` and add equivalent `@exec` statements and also `@unexec` for `pkg_delete`. You do not need to delete `info/dir` with `@unexec`.

```

Index: pkg/PLIST
=====
RCS file: /usr/cvs/ports/editors/emacs/pkg/PLIST,v
retrieving revision 1.15
diff -u -r1.15 PLIST
-- PLIST          1997/03/04 08:04:00      1.15
+++ PLIST          1997/05/20 10:25:12      1.17
@@ -16,7 +14,15 @@
  man/man1/etags.1.gz
  man/man1/ctags.1.gz
+@unexec install-info -delete %D/info/emacs %D/info/dir

```

```
:
+@unexec install-info -delete %D/info/ccmode %D/info/dir
info/cl
info/cl-1
@@ -87,6 +94,18 @@
info/viper-3
info/viper-4
+@exec [ -f %D/info/dir ] || sed -
ne 'l,/Menu:/p' /usr/share/info/dir > %D/info/dir
+@exec install-info %D/info/emacs %D/info/dir
:
+@exec install-info %D/info/ccmode %D/info/dir
libexec/emacs/19.34/i386-freebsd/cvtmail
libexec/emacs/19.34/i386-freebsd/digest-doc
```

**Note:** The `@unexec install-info -delete` commands have to be listed before the info files themselves so they can read the files. Also, the `@exec install-info` commands have to be after the info files and the `@exec` command that creates the `dir` file.

7. Test and admire your work. :). Check the `dir` file before and after each step.

## The `pkg/` subdirectory

There are some tricks we haven't mentioned yet about the `pkg/` subdirectory that come in handy sometimes.

### MESSAGE

If you need to display a message to the installer, you may place the message in `pkg/MESSAGE`. This capability is often useful to display additional installation steps to be taken after a `pkg_add` or to display licensing information.

**Note:** The `pkg/MESSAGE` file does not need to be added to `pkg/PLIST`. Also, it will not get automatically printed if the user is using the port, not the package, so you should probably display it from the `post-install` target yourself.

## INSTALL

If your port needs to execute commands when the binary package is installed with `pkg_add` you can do this via the `pkg/INSTALL` script. This script will automatically be added to the package, and will be run twice by `pkg_add`. The first time will as `INSTALL ${PKGNAME} PRE-INSTALL` and the second time as `INSTALL ${PKGNAME} POST-INSTALL`. `$2` can be tested to determine which mode the script is being run in. The `PKG_PREFIX` environmental variable will be set to the package installation directory. See `pkg.add(1)` for additional information.

**Note:** This script is not run automatically if you install the port with `make install`. If you are depending on it being run, you will have to explicitly call it from your port's `Makefile`.

## REQ

If your port needs to determine if it should install or not, you can create a `pkg/REQ` “requirements” script. It will be invoked automatically at installation/deinstallation time to determine whether or not installation/deinstallation should proceed.

## Changing `PLIST` based on make variables

Some ports, particularly the p5- ports, need to change their `PLIST` depending on what options they are configured with (or version of perl, in the case of p5- ports). To make this easy, any instances in the `PLIST` of `%%OSREL%%`, `%%PERL_VER%%`, and `%%PERL_VERSION%%` will be substituted for appropriately. The value of `%%OSREL%%` is the numeric revision of the operating system (e.g., 2.2.7). `%%PERL_VERSION%%` is the full version number of perl (e.g., 5.00502) and `%%PERL_VER%%` is the perl version number minus the patchlevel (e.g., 5.005).

If you need to make other substitutions, you can set the `PLIST_SUB` variable with a list of `VAR=VALUE` pairs and instances of `%%VAR%%` will be substituted with `VALUE` in the `PLIST`.

For instance, if you have a port that installs many files in a version-specific subdirectory, you can put something like

```
OCTAVE_VERSION= 2.0.13
PLIST_SUB=      OCTAVE_VERSION=${OCTAVE_VERSION}
```

in the `Makefile` and use `%%OCTAVE_VERSION%%` wherever the version shows up in `PLIST`. That way, when you upgrade the port, you will not have to change dozens (or in some cases, hundreds) of lines in the `PLIST`.

This substitution (as well as addition of any man pages) will be done between the `do-install` and `post-install` targets, by reading from `PLIST` and writing to `TMPPLIST` (default:

`WRKDIR/.PLIST.mktmp`). So if your port builds `PLIST` on the fly, do so in or before `do-install`. Also, if your port needs to edit the resulting file, do so in `post-install` to a file named `TMPPLIST`.

## Changing the names of files in the `pkg` subdirectory

All the filenames in the `pkg` subdirectory are defined using variables so you can change them in your `Makefile` if need be. This is especially useful when you are sharing the same `pkg` subdirectory among several ports or have to write to one of the above files (see writing to places other than `WRKDIR` for why it is a bad idea to write directly in to the `pkg` subdirectory).

Here is a list of variable names and their default values.

Variable	Default value
<code>COMMENT</code>	<code>\${PKGDIR}/DESCR</code>
<code>DESCR</code>	<code>\${PKGDIR}/DESCR</code>
<code>PLIST</code>	<code>\${PKGDIR}/PLIST</code>
<code>PKGINSTALL</code>	<code>\${PKGDIR}/PKGINSTALL</code>
<code>PKGDEINSTALL</code>	<code>\${PKGDIR}/PKGDEINSTALL</code>
<code>PKGREQ</code>	<code>\${PKGDIR}/REQ</code>
<code>PKGMESSAGE</code>	<code>\${PKGDIR}/MESSAGE</code>

Please change these variables rather than overriding `PKG_ARGS`. If you change `PKG_ARGS`, those files will not correctly be installed in `/var/db/pkg` upon install from a port.

## Licensing Problems

Some software packages have restrictive licenses or can be in violation to the law (PKP's patent on public key crypto, ITAR (export of crypto software) to name just two of them). What we can do with them varies a lot, depending on the exact wordings of the respective licenses.

**Note:** It is your responsibility as a porter to read the licensing terms of the software and make sure that the FreeBSD project will not be held accountable of violating them by redistributing the source or compiled binaries either via ftp or CD-ROM. If in doubt, please contact the FreeBSD ports mailing list [<freebsd-ports@FreeBSD.ORG>](mailto:freebsd-ports@FreeBSD.ORG).

There are two variables you can set in the `Makefile` to handle the situations that arise frequently:

1. If the port has a “do not sell for profit” type of license, set the variable `NO_CDROM` to a string describing the reason why. We will make sure such ports won’t go into the CD-ROM come release time. The distfile and package will still be available via ftp.
2. If the resulting package needs to be built uniquely for each site, or the resulting binary package can’t be distributed due to licensing; set the variable `NO_PACKAGE` to a string describing the reason why. We will make sure such packages won’t go on the ftp site, nor into the CD-ROM come release time. The distfile will still be included on both however.
3. If the port has legal restrictions on who can use it (e.g., crypto stuff) or has a “no commercial use” license, set the variable `RESTRICTED` to be the string describing the reason why. For such ports, the distfiles/packages will not be available even from our ftp sites.

**Note:** The GNU General Public License (GPL), both version 1 and 2, should not be a problem for ports.

**Note:** If you are a committer, make sure you update the `ports/LEGAL` file too.

## Upgrading

When you notice that a port is out of date compared to the latest version from the original authors, first make sure you have the latest port. You can find them in the `ports/ports-current` directory of the ftp mirror sites.

The next step is to send a mail to the maintainer, if one is listed in the port’s `Makefile`. That person may already be working on an upgrade, or have a reason to not upgrade the port right now (because of, for example, stability problems of the new version).

If the maintainer asks you to do the upgrade or there isn’t any such person to begin with, please make the upgrade and send the recursive diff (either unified or context diff is fine, but port committers appear to prefer unified diff more) of the new and old ports directories to us (e.g., if your modified port directory is called `superedit` and the original as in our tree is `superedit.bak`, then send us the result of `diff -ruN superedit.bak superedit`). Please examine the output to make sure all the changes make sense. The best way to send us the diff is by including it to `send-pr(1)` (category `ports`). Please mention any added or deleted files in the message, as they have to be explicitly specified to CVS when doing a commit. If the diff is more than about 20KB, please compress and uuencode it; otherwise, just include it in as is in the PR.

Once again, please use `diff(1)` and not `shar(1)` to send updates to ports.

## Do's and Dont's

Here is a list of common do's and dont's that you encounter during the porting process. You should check your own port against this list, but you can also check ports in the PR database that others have submitted. Submit any comments on ports you check as described in Bug Reports and General Commentary. Checking ports in the PR database will both make it faster for us to commit them, and prove that you know what you are doing.

### Strip Binaries

Do strip binaries. If the original source already strips the binaries, fine; otherwise you should add a `post-install` rule to it yourself. Here is an example;

```
post-install:
    strip ${PREFIX}/bin/xdl
```

Use the `file(1)` command on the installed executable to check whether the binary is stripped or not. If it does not say `not stripped`, it is stripped.

### INSTALL\_\* macros

Do use the macros provided in `bsd.port.mk` to ensure correct modes and ownership of files in your own `*-install` targets. They are:

- `INSTALL_PROGRAM` is a command to install binary executables.
- `INSTALL_SCRIPT` is a command to install executable scripts.
- `INSTALL_DATA` is a command to install sharable data.
- `INSTALL_MAN` is a command to install manpages and other documentation (it doesn't compress anything).

These are basically the `install` command with all the appropriate flags. See below for an example on how to use them.

### WRKDIR

Do not write anything to files outside `WRKDIR`. `WRKDIR` is the only place that is guaranteed to be writable during the port build (see compiling ports from CDROM for an example of building ports from a read-only tree). If you need to modify some file in `PKGDIR`, do so by redefining a variable, not by writing over it.

## WRKDIRPREFIX

Make sure your port honors WRKDIRPREFIX. Most ports don't have to worry about this. In particular, if you are referring to a WRKDIR of another port, note that the correct location is

```
WRKDIRPREFIXPORTSDIR/subdir/name/work not PORTSDIR/subdir/name/work or  
.CURDIR/../../subdir/name/work or some such.
```

Also, if you are defining WRKDIR yourself, make sure you prepend `${WRKDIRPREFIX}${.CURDIR}` in the front.

## Differentiating operating systems and OS versions

You may come across code that needs modifications or conditional compilation based upon what version of UNIX it is running under. If you need to make such changes to the code for conditional compilation, make sure you make the changes as general as possible so that we can back-port code to FreeBSD 1.x systems and cross-port to other BSD systems such as 4.4BSD from CSRG, BSD/386, 386BSD, NetBSD, and OpenBSD.

The preferred way to tell 4.3BSD/Reno (1990) and newer versions of the BSD code apart is by using the BSD macro defined in `<sys/param.h>`. Hopefully that file is already included; if not, add the code:

```
#if (defined(__unix__) || defined(unix)) && !defined(USG)  
#include <sys/param.h>  
#endif
```

to the proper place in the `.c` file. We believe that every system that defines these two symbols has `sys/param.h`. If you find a system that doesn't, we would like to know. Please send mail to the FreeBSD ports mailing list `<freebsd-ports@FreeBSD.ORG>`.

Another way is to use the GNU Autoconf style of doing this:

```
#ifdef HAVE_SYS_PARAM_H  
#include <sys/param.h>  
#endif
```

Don't forget to add `-DHAVE_SYS_PARAM_H` to the `CFLAGS` in the `Makefile` for this method.

Once you have `sys/param.h` included, you may use:

```
#if (defined(BSD) && (BSD >= 199103))
```

to detect if the code is being compiled on a 4.3 Net2 code base or newer (e.g. FreeBSD 1.x, 4.3/Reno, NetBSD 0.9, 386BSD, BSD/386 1.1 and below).

Use:

```
#if (defined(BSD) && (BSD >= 199306))
```

to detect if the code is being compiled on a 4.4 code base or newer (e.g. FreeBSD 2.x, 4.4, NetBSD 1.0, BSD/386 2.0 or above).

The value of the `BSD` macro is 199506 for the 4.4BSD-Lite2 code base. This is stated for informational purposes only. It should not be used to distinguish between versions of FreeBSD based only on 4.4-Lite vs. versions that have merged in changes from 4.4-Lite2. The `__FreeBSD__` macro should be used instead.

Use sparingly:

- `__FreeBSD__` is defined in all versions of FreeBSD. Use it if the change you are making *only* affects FreeBSD. Porting gotchas like the use of `sys_errlist[]` vs `strerror()` are Berkeleyisms, not FreeBSD changes.
- In FreeBSD 2.x, `__FreeBSD__` is defined to be 2. In earlier versions, it is 1. Later versions will bump it to match their major version number.
- If you need to tell the difference between a FreeBSD 1.x system and a FreeBSD 2.x or 3.x system, usually the right answer is to use the `BSD` macros described above. If there actually is a FreeBSD specific change (such as special shared library options when using `ld`) then it is OK to use `__FreeBSD__` and `#if __FreeBSD__ > 1` to detect a FreeBSD 2.x and later system. If you need more granularity in detecting FreeBSD systems since 2.0-RELEASE you can use the following:

```
#if __FreeBSD__ >= 2
#include <osreldate.h>
#   if __FreeBSD_version >= 199504
        /* 2.0.5+ release specific code here */
#   endif
#endif
```

<b>Release</b>	<b>__FreeBSD_version</b>
2.0-RELEASE	119411
2.1-CURRENTs	199501, 199503
2.0.5-RELEASE	199504
2.2-CURRENT before 2.1	199508
2.1.0-RELEASE	199511
2.2-CURRENT before 2.1.5	199512
2.1.5-RELEASE	199607
2.2-CURRENT before 2.1.6	199608
2.1.6-RELEASE	199612

2.1.7-RELEASE	199612
2.2-RELEASE	220000
2.2.1-RELEASE	220000 (no change)
2.2-STABLE after 2.2.1-RELEASE	220000 (no change)
2.2-STABLE after texinfo-3.9	221001
2.2-STABLE after top	221002
2.2.2-RELEASE	222000
2.2-STABLE after 2.2.2-RELEASE	222001
2.2.5-RELEASE	225000
2.2-STABLE after 2.2.5-RELEASE	225001
2.2-STABLE after ldconfig -R merge	225002
2.2.6-RELEASE	226000
2.2.7-RELEASE	227000
2.2-STABLE after 2.2.7-RELEASE	227001
2.2-STABLE after semctl(2) change	227002
2.2.8-RELEASE	228000
2.2-STABLE after 2.2.8-RELEASE	228001
3.0-CURRENT before mount(2) change	300000
3.0-CURRENT after mount(2) change	300001
3.0-CURRENT after semctl(2) change	300002
3.0-CURRENT after ioctl arg changes	300003
3.0-CURRENT after ELF conversion	300004
3.0-RELEASE	300005
3.0-CURRENT after 3.0-RELEASE	300006
3.0-STABLE after 3/4 branch	300007
3.1-RELEASE	310000
3.1-STABLE after 3.1-RELEASE	310001
4.0-CURRENT after 3/4 branch	400000

**Note:** Note that 2.2-STABLE sometimes identifies itself as “2.2.5-STABLE” after the 2.2.5-RELEASE. The pattern used to be year followed by the month, but we decided to change it to a more straightforward major/minor system starting from 2.2. This is because the parallel development on several branches made it infeasible to classify the releases simply by their real release dates. If

you are making a port now, you don't have to worry about old `-CURRENTs`; they are listed here just for your reference.

In the hundreds of ports that have been done, there have only been one or two cases where `__FreeBSD__` should have been used. Just because an earlier port screwed up and used it in the wrong place does not mean you should do so too.

### Writing something after `bsd.port.mk`

Do not write anything after the `.include <bsd.port.mk>` line. It usually can be avoided by including `bsd.port.pre.mk` somewhere in the middle of your Makefile and `bsd.port.post.mk` at the end.

**Note:** You need to include either the `pre.mk/post.mk` pair or `bsd.port.mk` only; don't mix these two.

`bsd.port.pre.mk` only defines a few variables, which can be used in tests in the Makefile, `bsd.port.post.mk` defines the rest.

Here are some important variables defined in `bsd.port.pre.mk` (this is not the complete list, please read `bsd.port.mk` for the complete list).

Variable	Description
ARCH	The architecture as returned by <code>uname -m</code> (e.g., <code>i386</code> )
OPSYS	The operating system type, as returned by <code>uname -s</code> (e.g., <code>FreeBSD</code> )
OSREL	The release version of the operating system (e.g., <code>2.1.5</code> or <code>2.2.7</code> )
OSVERSION	The numeric version of the operating system, same as <code>__FreeBSD_version</code> .
PORTOBJFORMAT	The object format of the system ( <code>aout</code> or <code>elf</code> )
LOCALBASE	The base of the "local" tree (e.g., <code>/usr/local/</code> )
X11BASE	The base of the "X11" tree (e.g., <code>/usr/X11R6</code> )
PREFIX	Where the port installs itself (see more on <code>PREFIX</code> ).

**Note:** If you have to define the variables `USE_IMAKE`, `USE_X_PREFIX`, or `MASTERDIR`, do so before including `bsd.port.pre.mk`.

Here are some examples of things you can write after `bsd.port.pre.mk`;

```
# no need to compile lang/perl5 if perl5 is already in system
.if ${OSVERSION} > 300003
BROKEN= perl is in system
.endif

# only one shlib version number for ELF
.if ${PORTOBJFORMAT} == "elf"
TCL_LIB_FILE= ${TCL_LIB}.${SHLIB_MAJOR}
.else
TCL_LIB_FILE= ${TCL_LIB}.${SHLIB_MAJOR}.${SHLIB_MINOR}
.endif

# software already makes link for ELF, but not for a.out
post-install:
.if ${PORTOBJFORMAT} == "aout"
${LN} -sf liblinpack.so.1.0 ${PREFIX}/lib/liblinpack.so
.endif
```

## Install additional documentation

If your software has some documentation other than the standard man and info pages that you think is useful for the user, install it under `PREFIX/share/doc`. This can be done, like the previous item, in the `post-install` target.

Create a new directory for your port. The directory name should reflect what the port is. This usually means `PKGNAME` minus the version part. However, if you think the user might want different versions of the port to be installed at the same time, you can use the whole `PKGNAME`.

Make the installation dependent to the variable `NOPORTDOCS` so that users can disable it in `/etc/make.conf`, like this:

```
post-install:
.if !defined(NOPORTDOCS)
${MKDIR} ${PREFIX}/share/doc/xv
${INSTALL_MAN} ${WRKSRCS}/docs/xvdocs.ps ${PREFIX}/share/doc/xv
.endif
```

Do not forget to add them to `pkg/PLIST` too! (Do not worry about `NOPORTDOCS` here; there is currently no way for the packages to read variables from `/etc/make.conf`.)

Also you can use the `pkg/MESSAGE` file to display messages upon installation. See the `using pkg/MESSAGE` section for details.

**Note:** `MESSAGE` does not need to be added to `pkg/PLIST`).

### **DIST\_SUBDIR**

Do not let your port clutter `/usr/ports/distfiles`. If your port requires a lot of files to be fetched, or contains a file that has a name that might conflict with other ports (e.g., `Makefile`), set `DIST_SUBDIR` to the name of the port (`PKGNAME` without the version part should work fine). This will change `DISTDIR` from the default `/usr/ports/distfiles` to `/usr/ports/distfiles/DIST_SUBDIR`, and in effect puts everything that is required for your port into that subdirectory.

It will also look at the subdirectory with the same name on the backup master site at `ftp.freebsd.org`. (Setting `DISTDIR` explicitly in your `Makefile` will not accomplish this, so please use `DIST_SUBDIR`.)

**Note:** This does not affect the `MASTER_SITES` you define in your `Makefile`.

### **Package information**

Do include package information, i.e. `COMMENT`, `DESCR`, and `PLIST`, in `pkg`.

**Note:** Note that these files are not used only for packaging anymore, and are *mandatory* now, even if `NO_PACKAGE` is set.

### **RCS strings**

Do not put RCS strings in patches. CVS will mangle them when we put the files into the ports tree, and when we check them out again, they will come out different and the patch will fail. RCS strings are surrounded by dollar (\$) signs, and typically start with `$Id` or `$RCS`.

### **Recursive diff**

Using the `recurse (-r)` option to `diff` to generate patches is fine, but please take a look at the resulting patches to make sure you don't have any unnecessary junk in there. In particular, diffs between two backup files, `Makefiles` when the port uses `Imake` or `GNU configure`, etc., are unnecessary and should be deleted. If you had to edit `configure.in` and run `autoconf` to regenerate `configure`, do not take the diffs of `configure` (it often grows to a few thousand lines!); define `USE_AUTOCONF=yes` and take the diffs of `configure.in`.

Also, if you had to delete a file, then you can do it in the `post-extract` target rather than as part of the patch. Once you are happy with the resulting diff, please split it up into one source file per patch file.

## **PREFIX**

Do try to make your port install relative to `PREFIX`. (The value of this variable will be set to `LOCALBASE` (default `/usr/local`), unless `USE_X_PREFIX` or `USE_IMAKE` is set, in which case it will be `X11BASE` (default `/usr/X11R6`).)

Not hard-coding `/usr/local` or `/usr/X11R6` anywhere in the source will make the port much more flexible and able to cater to the needs of other sites. For X ports that use `imake`, this is automatic; otherwise, this can often be done by simply replacing the occurrences of `/usr/local` (or `/usr/X11R6` for X ports that do not use `imake`) in the various scripts/Makefiles in the port to read `PREFIX`, as this variable is automatically passed down to every stage of the build and install processes.

Do not set `USE_X_PREFIX` unless your port truly require it (i.e., it links against X libs or it needs to reference files in `X11BASE`).

The variable `PREFIX` can be reassigned in your `Makefile` or in the user's environment. However, it is strongly discouraged for individual ports to set this variable explicitly in the `Makefiles`.

Also, refer to programs/files from other ports with the variables mentioned above, not explicit pathnames. For instance, if your port requires a macro `PAGER` to be the full pathname of `less`, use the compiler flag:

```
-DPAGER="\${PREFIX}/bin/less\ "
```

or

```
-DPAGER="\${LOCALBASE}/bin/less\ "
```

if this is an X port, instead of `-DPAGER="/usr/local/bin/less"`. This way it will have a better chance of working if the system administrator has moved the whole `'/usr/local'` tree somewhere else.

## **Subdirectories**

Try to let the port put things in the right subdirectories of `PREFIX`. Some ports lump everything and put it in the subdirectory with the port's name, which is incorrect. Also, many ports put everything except binaries, header files and manual pages in the a subdirectory of `lib`, which does not bode well with the BSD paradigm. Many of the files should be moved to one of the following: `etc` (setup/configuration files), `libexec` (executables started internally), `sbin` (executables for superusers/managers), `info` (documentation for info browser) or `share` (architecture independent files). See `man hier(7)` for details, the rules governing `/usr` pretty much apply to `/usr/local` too. The exception are ports dealing with USENET "news". They may use `PREFIX/news` as a destination for their files.

## Cleaning up empty directories

Do make your ports clean up after themselves when they are deinstalled. This is usually accomplished by adding `@dirrm` lines for all directories that are specifically created by the port. You need to delete subdirectories before you can delete parent directories.

```
:
lib/X11/oneko/pixmaps/cat.xpm
lib/X11/oneko/sounds/cat.au
:
@dirrm lib/X11/oneko/pixmaps
@dirrm lib/X11/oneko/sounds
@dirrm lib/X11/oneko
```

However, sometimes `@dirrm` will give you errors because other ports also share the same subdirectory. You can call `rmdir` from `@unexec` to remove only empty directories without warning.

```
@unexec rmdir %D/share/doc/gimp 2>/dev/null || true
```

This will neither print any error messages nor cause `pkg_delete` to exit abnormally even if `PREFIX/share/doc/gimp` is not empty due to other ports installing some files in there.

## UIDs

If your port requires a certain user to be on the installed system, let the `pkg/INSTALL` script call `pw` to create it automatically. Look at `net/cvsup-mirror` for an example.

If your port must use the same user/group ID number when it is installed a binarypackage as when it was compiled, then you must choose a free UID from 50 to 99 and register it below. Look at `japanese/Wnn` for an example.

Make sure you don't use a UID already used by the system or other ports. This is the current list of UIDs between 50 and 99.

```
majordom:*:54:54:Majordomo Pseudo User:/usr/local/majordomo:/nonexistent
cyrus:*:60:60:the cyrus mail server:/nonexistent:/nonexistent
gnats:*:61:1:GNATS database owner:/usr/local/share/gnats/gnats-db:/bin/sh
uucp:*:66:66:UUCP pseudo-user:/var/spool/uucppublic:/usr/libexec/uucp/uucico
xten:*:67:67:X-10 daemon:/usr/local/xten:/nonexistent
pop:*:68:6:Post Office Owner (popper):/nonexistent:/nonexistent
wnn:*:69:7:Wnn:/nonexistent:/nonexistent
ifmail:*:70:66:Ifmail user:/nonexistent:/nonexistent
pgsql:*:70:70:PostgreSQL pseudo-user:/usr/local/pgsql:/bin/sh
ircd:*:72:72:IRCd hybrid:/nonexistent:/nonexistent
alias:*:81:81:QMail user:/var/qmail/alias:/nonexistent
```

```
qmail:*:83:81:QMail user:/var/qmail:/nonexistent
qmaild:*:82:81:QMail user:/var/qmail:/nonexistent
qmailq:*:85:82:QMail user:/var/qmail:/nonexistent
qmails:*:87:82:QMail user:/var/qmail:/nonexistent
qmailp:*:84:81:QMail user:/var/qmail:/nonexistent
qmailr:*:86:82:QMail user:/var/qmail:/nonexistent
msql:*:87:87:mSQL-2 pseudo-user:/var/db/msqldb:/bin/sh
```

Please include a notice when you submit a port (or an upgrade) that reserves a new UID or GID in this range. This allows us to keep the list of reserved IDs up to date.

## Do things rationally

The Makefile should do things simply and reasonably. If you can make it a couple of lines shorter or more readable, then do so. Examples include using a `make .if` construct instead of a shell `if` construct, not redefining `do-extract` if you can redefine `EXTRACT*` instead, and using `GNU_CONFIGURE` instead of `CONFIGURE_ARGS += -prefix=${PREFIX}`.

## Respect CFLAGS

The port should respect the `CFLAGS` variable. If it doesn't, please add `NO_PACKAGE=ignores cflags` to the Makefile.

## Configuration files

If your port requires some configuration files in `PREFIX/etc`, do *not* just install them and list them in `pkg/PLIST`. That will cause `pkg_delete` to delete files carefully edited by the user and a new installation to wipe them out.

Instead, install sample files with a suffix (`filename.sample` will work well) and print out a message pointing out that the user has to copy and edit the file before the software can be made to work.

## Portlint

Do check your work with `portlint` before you submit or commit it.

## Feedback

Do send applicable changes/patches to the original author/maintainer for inclusion in next release of the code. This will only make your job that much easier for the next release.

## Miscellanea

The files `pkg/DESCR`, `pkg/COMMENT`, and `pkg/PLIST` should each be double-checked. If you are reviewing a port and feel they can be worded better, do so.

Don't copy more copies of the GNU General Public License into our system, please.

Please be careful to note any legal issues! Don't let us illegally distribute software!

## If you are stuck...

Do look at existing examples and the `bsd.port.mk` file before asking us questions! ;)

Do ask us questions if you have any trouble! Do not just beat your head against a wall! :)

## A Sample Makefile

Here is a sample Makefile that you can use to create a new port. Make sure you remove all the extra comments (ones between brackets)!

It is recommended that you follow this format (ordering of variables, empty lines between sections, etc.). This format is designed so that the most important information is easy to locate. We recommend that you use `portlint` to check the Makefile.

```
[the header...just to make it easier for us to identify the ports.]
# New ports collection makefile for:  xdvi
[the version required header should updated when upgrading a port.]
# Version required:    pl18 [things like "1.5alpha" are fine here too]
[this is the date when the first version of this Makefile was created.
Never change this when doing an update of the port.]
# Date created:                26 May 1995
[this is the person who did the original port to FreeBSD, in particular, the
person who wrote the first version of this Makefile. Remember, this should
not be changed when upgrading the port later.]
# Whom:                        Satoshi Asami <asami@FreeBSD.ORG>
#
# $Id$
```

```
[ ^^^ This will be automatically replaced with RCS ID string by CVS
when it is committed to our repository.]
#

[section to describe the port itself and the master site - DISTNAME
 is always first, followed by PKGNAME (if necessary), CATEGORIES,
 and then MASTER_SITES, which can be followed by MASTER_SITE_SUBDIR.
 After those, one of EXTRACT_SUFX or DISTFILES can be specified too.]
DISTNAME=      xdvi
PKGNAME=       xdvi-pl18
CATEGORIES=    print
[do not forget the trailing slash ("/")!
 if you aren't using MASTER_SITE_* macros]
MASTER_SITES=  ${MASTER_SITE_XCONTRIB}
MASTER_SITE_SUBDIR= applications
[set this if the source is not in the standard ".tar.gz" form]
EXTRACT_SUFX= .tar.Z

[section for distributed patches - can be empty]
PATCH_SITES=  ftp://ftp.sra.co.jp/pub/X11/japanese/
PATCHFILES=  xdvi-18.patch1.gz xdvi-18.patch2.gz

[maintainer; *mandatory*! This is the person (preferably with commit
 privileges) who a user can contact for questions and bug reports - this
 person should be the porter or someone who can forward questions to the
 original porter reasonably promptly. If you really do not want to have
 your address here, set it to "ports@FreeBSD.ORG".]
MAINTAINER=    asami@FreeBSD.ORG

[dependencies - can be empty]
RUN_DEPENDS=   gs:${PORTSDIR}/print/ghostscript
LIB_DEPENDS=   Xpm.5:${PORTSDIR}/graphics/xpm

[this section is for other standard bsd.port.mk variables that do not
 belong to any of the above]
[If it asks questions during configure, build, install...]
IS_INTERACTIVE=      yes
[If it extracts to a directory other than ${DISTNAME}...]
WRKSRC=              ${WRKDIR}/xdvi-new
[If the distributed patches were not made relative to ${WRKSRC}, you
 may need to tweak this]
PATCH_DIST_STRIP=   -p1
[If it requires a "configure" script generated by GNU autoconf to be run]
GNU_CONFIGURE=      yes
[If it requires GNU make, not /usr/bin/make, to build...]
```

```
USE_GMAKE=      yes
[If it is an X application and requires "xmkmf -a" to be run...]
USE_IMAKE=      yes
[et cetera.]

[non-standard variables to be used in the rules below]
MY_FAVORITE_RESPONSE= "yeah, right"

[then the special rules, in the order they are called]
pre-fetch:
    i go fetch something, yeah

post-patch:
    i need to do something after patch, great

pre-install:
    and then some more stuff before installing, wow

[and then the epilogue]
.include <bsd.port.mk>
```

## Package Names

The following are the conventions you should follow in naming your packages. This is to have our package directory easy to scan, as there are already lots and lots of packages and users are going to turn away if they hurt their eyes!

The package name should look like *language-name-compiled.specifics-version.numbers*.

If your `DISTNAME` doesn't look like that, set `PKGNAME` to something in that format.

1. FreeBSD strives to support the native language of its users. The *language-* part should be a two letter abbreviation of the natural language defined by ISO-639 if the port is specific to a certain language. Examples are `ja` for Japanese, `ru` for Russian, `vi` for Vietnamese, `zh` for Chinese, `ko` for Korean and `de` for German.
2. The name part should be all lowercases, except for a really large package (with lots of programs in it). Things like XFree86 (yes there really is a port of it, check it out) and ImageMagick fall into this category. Otherwise, convert the name (or at least the first letter) to lowercase. If the capital letters are important to the name (for example, with one-letter names like `R` or `V`) you may use capital letters at your discretion. There is a tradition of naming Perl 5 modules by prepending `p5-` and converting the double-colon separator to a hyphen; for example, the `Data::Dumper` module

becomes `p5-Data-Dumper`. If the software in question has numbers, hyphens, or underscores in its name, you may include them as well (like `kinput2`).

3. If the port can be built with different hardcoded defaults (usually part of the directory name in a family of ports), the `-compiled.specifics` part should state the compiled-in defaults (the hyphen is optional). Examples are `papersize` and `font units`.
4. The version string should be a period-separated list of integers and single lowercase alphabets. The only exception is the string `p1` (meaning ‘patchlevel’), which can be used *only* when there are no major and minor version numbers in the software.

Here are some (real) examples on how to convert a `DISTNAME` into a suitable `PKGNAME`:

Distribution Name	Package Name	Reason
<code>mule-2.2.2.</code>	<code>mule-2.2.2</code>	No changes required
<code>XFree86-3.1.2</code>	<code>XFree86-3.1.2</code>	No changes required
<code>EmiClock-1.0.2</code>	<code>emiclock-1.0.2</code>	No uppercase names for single programs
<code>gmod1.4</code>	<code>gmod-1.4</code>	Need a hyphen before version numbers
<code>xmris.4.0.2</code>	<code>xmris-4.0.2</code>	Need a hyphen before version numbers
<code>rdist-1.3alpha</code>	<code>rdist-1.3a</code>	No strings like <code>alpha</code> allowed
<code>es-0.9-beta1</code>	<code>es-0.9b1</code>	No strings like <code>beta</code> allowed
<code>v3.3beta021.src</code>	<code>tiff-3.3</code>	What the heck was that anyway?
<code>tvtwm</code>	<code>tvtwm-pl11</code>	Version string always required
<code>piewm</code>	<code>piewm-1.0</code>	Version string always required
<code>xvgr-2.10pl1</code>	<code>xvgr-2.10.1</code>	<code>p1</code> allowed only when no major/minor version numbers
<code>gawk-2.15.6</code>	<code>ja-gawk-2.15.6</code>	Japanese language version
<code>psutils-1.13</code>	<code>psutils-letter-1.13</code>	Papersize hardcoded at package build time
<code>pkfonts</code>	<code>pkfonts300-1.0</code>	Package for 300dpi fonts

If there is absolutely no trace of version information in the original source and it is unlikely that the original author will ever release another version, just set the version string to `1.0` (like the `piewm` example above). Otherwise, ask the original author or use the date string (`yy.mm.dd`) as the version.

## Categories

As you already know, ports are classified in several categories. But for this to work, it is important that porters and users understand what each category and how we decide what to put in each category.

### Current list of categories

First, this is the current list of port categories. Those marked with an asterisk (\*) are *virtual* categories—those that do not have a corresponding subdirectory in the ports tree.

**Note:** For non-virtual categories, you will find a one-line description in the `pkg/COMMENT` file in that subdirectory (e.g., `archivers/pkg/COMMENT`).

Category	Description
afterstep*	Ports to support AfterStep window manager
archivers	Archiving tools.
astro	Astronomical ports.
audio	Sound support.
benchmarks	Benchmarking utilities.
biology	Biology-related software.
cad	Computer aided design tools.
chinese	Chinese language support.
comms	Communication software. Mostly software to talk to your serial port.
converters	Character code converters.
databases	Databases.
deskutils	Things that used to be on the desktop before computers were invented.
devel	Development utilities. Do not put libraries here just because they are libraries—unless they truly don't belong to anywhere else, they shouldn't be in this category.
editors	General editors. Specialized editors go in the section for those tools (e.g., a mathematical-formula editor will go in <code>math</code> ).
elisp	Emacs-lisp ports.

emulators	Emulators for other operating systems. Terminal emulators do <i>not</i> belong here—X-based ones should go to <code>x11</code> and text-based ones to either <code>comms</code> or <code>misc</code> , depending on the exact functionality.
games	Games.
german	German language support.
graphics	Graphics utilities.
japanese	Japanese language support.
kde*	Ports that form the K Desktop Environment ( <code>kde</code> ).
korean	Korean language support.
lang	Programming languages.
mail	Mail software.
math	Numerical computation software and other utilities for mathematics.
mbone	MBone applications.
misc	Miscellaneous utilities—basically things that doesn't belong to anywhere else. This is the only category that should not appear with any other non-virtual category. If you have <code>misc</code> with something else in your <code>CATEGORIES</code> line, that means you can safely delete <code>misc</code> and just put the port in that other subdirectory!
net	Miscellaneous networking software.
news	USENET news software.
offix*	Ports from the OffiX suite.
palm	Software support for the 3Com Palm(tm) series.
perl5*	Ports that require perl version 5 to run.
plan9*	Various programs from Plan9.
print	Printing software. Desktop publishing tools (previewers, etc.) belong here too.
python*	Software written in python.
russian	Russian language support.
security	Security utilities.
shells	Command line shells.
sysutils	System utilities.

tcl175*	Ports that use tcl version 7.5 to run.
tcl176*	Ports that use tcl version 7.6 to run.
tcl180*	Ports that use tcl version 8.0 to run.
tcl181*	Ports that use tcl version 8.1 to run.
textproc	Text processing utilities. It does not include desktop publishing tools, which go to print/.
tk41*	Ports that use tk version 4.1 to run.
tk42*	Ports that use tk version 4.2 to run.
tk80*	Ports that use tk version 8.0 to run.
tk81*	Ports that use tk version 8.1 to run.
vietnamese	Vietnamese language support.
windowmaker*	Ports to support the WindowMaker window manager
www	Software related to the World Wide Web. HTML language support belong here too.
x11	The X window system and friends. This category is only for software that directly support the window system. Do not put regular X applications here. If your port is an X application, define <code>USE_XLIB</code> (implied by <code>USE_IMAKE</code> ) and put it in appropriate categories. Also, many of them go into other <code>x11-*</code> categories (see below).
x11-clocks	X11 clocks.
x11-fm	X11 file managers.
x11-fonts	X11 fonts and font utilities.
x11-toolkits	X11 toolkits.
x11-wm	X11 window managers.

## Choosing the right category

As many of the categories overlap, you often have to choose which of the categories should be the primary category of your port. There are several rules that govern this use. Here is the list of priorities, in decreasing order of precedence.

- Language specific categories always come first. For example, if your port installs Japanese X11 fonts,

then your CATEGORIES line would read `japanese x11`.

- Specific categories win over less-specific ones. For instance, an HTML editor should be listed as `www editors`, not the other way around. Also, you don't need to list `net` when the port belongs to either of `mail`, `mbone`, `news`, `security`, or `www`.
- `x11` is used as a secondary category only when the primary category is a natural language. In particular, you should not put `x11` in the category line for X applications.
- If your port truly does not belong anywhere else, put it in `misc`.

If you are not sure about the category, please put a comment to that effect in your `send-pr` submission so we can discuss it before import it. (If you are a committer, send a note FreeBSD ports mailing list `<freebsd-ports@FreeBSD.ORG>` so we can discuss it first—too often new ports are imported to a wrong category only to be moved right away.)

## Changes to this document and the ports system

If you maintain a lot of ports, you should consider following the FreeBSD ports mailing list `<freebsd-ports@FreeBSD.ORG>`. Important changes to the way ports work will be announced there. You can always find more detailed information on the latest changes by looking at the `bsd.port.mk` CVS log (<http://www.FreeBSD.ORG/cgi/cvsweb.cgi/src/share/mk/bsd.port.mk>).

## That is It, Folks!

Boy, this sure was a long tutorial, wasn't it? Thanks for following us to here, really.

Well, now that you know how to do a port, let us go at it and convert everything in the world into ports! That is the easiest way to start contributing to the FreeBSD Project! :)

## **II. System Administration**

# Chapter 5. Configuring the FreeBSD Kernel

*Contributed by Jake Hamby <jehamby@lightside.com>. 6 October 1995.*

This large section of the handbook discusses the basics of building your own custom kernel for FreeBSD. This section is appropriate for both novice system administrators and those with advanced Unix experience.

## Why Build a Custom Kernel?

Building a custom kernel is one of the most important rites of passage every Unix system administrator must endure. This process, while time-consuming, will provide many benefits to your FreeBSD system. Unlike the `GENERIC` kernel, which must support every possible SCSI and network card, along with tons of other rarely used hardware support, a custom kernel only contains support for *your* PC's hardware. This has a number of benefits:

- It will take less time to boot because it does not have to spend time probing for hardware which you do not have.
- A custom kernel often uses less memory, which is important because the kernel is the one process which must always be present in memory, and so all of that unused code ties up pages of RAM that your programs would otherwise be able to use. Therefore, on a system with limited RAM, building a custom kernel is of critical importance.
- Finally, there are several kernel options which you can tune to fit your needs, and device driver support for things like sound cards which you can include in your kernel but are *not* present in the `GENERIC` kernel.

## Building and Installing a Custom Kernel

First, let us take a quick tour of the kernel build directory. All directories mentioned will be relative to the main `/usr/src/sys` directory, which is also accessible through `/sys`. There are a number of subdirectories here representing different parts of the kernel, but the most important, for our purposes, are `i386/conf`, where you will edit your custom kernel configuration, and `compile`, which is the staging area where your kernel will be built. Notice the logical organization of the directory tree, with each supported device, filesystem, and option in its own subdirectory. Also, anything inside the `i386` directory deals with PC hardware only, while everything outside the `i386` directory is common to all platforms which FreeBSD could potentially be ported to.

**Note:** If there is *not* a `/usr/src/sys` directory on your system, then the kernel source has not been installed. The easiest way to do this is by running `/stand/sysinstall` as `root`, choosing `Configure`, then `Distributions`, then `src`, then `sys`.

Next, move to the `i386/conf` directory and copy the `GENERIC` configuration file to the name you want to give your kernel. For example:

```
# cd /usr/src/sys/i386/conf
# cp GENERIC MYKERNEL
```

Traditionally, this name is in all capital letters and, if you are maintaining multiple FreeBSD machines with different hardware, it is a good idea to name it after your machine's hostname. We will call it `MYKERNEL` for the purpose of this example.

**Note:** You must execute these and all of the following commands under the `root` account or you will get permission denied errors.

Now, edit `MYKERNEL` with your favorite text editor. If you are just starting out, the only editor available will probably be `vi`, which is too complex to explain here, but is covered well in many books in the bibliography. Feel free to change the comment lines at the top to reflect your configuration or the changes you have made to differentiate it from `GENERIC`.

If you have build a kernel under SunOS or some other BSD operating system, much of this file will be very familiar to you. If you are coming from some other operating system such as DOS, on the other hand, the `GENERIC` configuration file might seem overwhelming to you, so follow the descriptions in the Configuration File section slowly and carefully.

**Note:** If you are trying to upgrade your kernel from an older version of FreeBSD, you will probably have to get a new version of `config(8)` from the same place you got the new kernel sources. It is located in `/usr/src/usr.sbin`, so you will need to download those sources as well. Re-build and install it before running the next commands.

When you are finished, type the following to compile and install your kernel:

```
# /usr/sbin/config MYKERNEL
# cd ../../compile/MYKERNEL
# make depend
# make
# make install
```

The new kernel will be copied to the root directory as `/kernel` and the old kernel will be moved to `/kernel.old`. Now, shutdown the system and reboot to use your kernel. In case something goes wrong,

there are some troubleshooting instructions at the end of this document. Be sure to read the section which explains how to recover in case your new kernel does not boot.

**Note:** If you have added any new devices (such as sound cards) you may have to add some device nodes to your `/dev` directory before you can use them.

## The Configuration File

The general format of a configuration file is quite simple. Each line contains a keyword and one or more arguments. For simplicity, most lines only contain one argument. Anything following a `#` is considered a comment and ignored. The following sections describe each keyword, generally in the order they are listed in `GENERIC`, although some related keywords have been grouped together in a single section (such as Networking) even though they are actually scattered throughout the `GENERIC` file. An exhaustive list of options and more detailed explanations of the device lines is present in the `LINT` configuration file, located in the same directory as `GENERIC`. If you are in doubt as to the purpose or necessity of a line, check first in `LINT`.

The kernel is currently being moved to a better organization of the option handling. Traditionally, each option in the config file was simply converted into a `-D` switch for the `CFLAGS` line of the kernel Makefile. Naturally, this caused a creeping optionism, with nobody really knowing which option has been referenced in what files.

In the new scheme, every `#ifdef` that is intended to be dependent upon an option gets this option out of an `opt_foo.h` declaration file created in the compile directory by `config`. The list of valid options for `config` lives in two files: options that do not depend on the architecture are listed in `/sys/conf/options`, architecture-dependent ones in `/sys/arch/conf/options.arch`, with `arch` being for example `i386`.

## Mandatory Keywords

These keywords are required in every kernel you build.

```
machine "i386"
```

The first keyword is `machine`, which, since FreeBSD only runs on Intel 386 and compatible chips, is `i386`.

**Note:** Any keyword which contains numbers used as text must be enclosed in quotation marks, otherwise `config` gets confused and thinks you mean the actual number 386.

```
cpu "cpu_type"
```

The next keyword is `cpu`, which includes support for each CPU supported by FreeBSD. The possible values of `cpu_type` include:

- I386\_CPU
- I486\_CPU
- I586\_CPU
- I686\_CPU

Multiple instances of the `cpu` line may be present with different values of `cpu_type` as are present in the `GENERIC` kernel. For a custom kernel, it is best to specify only the `cpu` you have. If, for example, you have an Intel Pentium, use `I586_CPU` for `cpu_type`.

```
ident machine_name
```

Next, we have `ident`, which is the identification of the kernel. You should change this from `GENERIC` to whatever you named your kernel, in this example, `MYKERNEL`. The value you put in `ident` will print when you boot up the kernel, so it is useful to give a kernel a different name if you want to keep it separate from your usual kernel (if you want to build an experimental kernel, for example). Note that, as with `machine` and `cpu`, enclose your kernel's name in quotation marks if it contains any numbers.

Since this name is passed to the C compiler as a `-D` switch, do not use names like `DEBUG`, or something that could be confused with another machine or CPU name, like `vax`.

```
maxusers number
```

This file sets the size of a number of important system tables. This number is supposed to be roughly equal to the number of simultaneous users you expect to have on your machine. However, under normal circumstances, you will want to set `maxusers` to at least 4, especially if you are using the X Window System or compiling software. The reason is that the most important table set by `maxusers` is the maximum number of processes, which is set to  $20 + 16 * \text{maxusers}$ , so if you set `maxusers` to 1, then you can only have 36 simultaneous processes, including the 18 or so that the system starts up at boot time, and the 15 or so you will probably create when you start the X Window System. Even a simple task like reading a man page will start up nine processes to filter, decompress, and view it. Setting `maxusers` to 4 will allow you to have up to 84 simultaneous processes, which should be enough for anyone. If, however, you see the dreaded `proc table full` error when trying to start another program, or are running a server with a large number of simultaneous users (like Walnut Creek CDROM's FTP site), you can always increase this number and rebuild.

**Note:** `maxuser` does *not* limit the number of users which can log into your machine. It simply sets various table sizes to reasonable values considering the maximum number of users you will likely have on your system and how many processes each of them will be running. One keyword which *does* limit the number of simultaneous *remote logins* is pseudo-device `pty 16`.

```
config kernel_name root on root_device
```

This line specifies the location and name of the kernel. Traditionally the kernel is called `vmunix` but in FreeBSD, it is aptly named `kernel`. You should always use `kernel` for `kernel_name` because changing it will render numerous system utilities inoperative. The second part of the line specifies the disk and partition where the root filesystem and kernel can be found. Typically this will be `wd0` for systems with non-SCSI drives, or `sd0` for systems with SCSI drives.

## General Options

These lines provide kernel support for various filesystems and other options.

```
options MATH_EMULATE
```

This line allows the kernel to simulate a math co-processor if your computer does not have one (386 or 486SX). If you have a Pentium, a 486DX, or a 386 or 486SX with a separate 387 or 487 chip, you can comment this line out.

**Note:** The normal math co-processor emulation routines that come with FreeBSD are *not* very accurate. If you do not have a math co-processor, and you need the best accuracy, I recommend that you change this option to `GPL_MATH_EMULATE` to use the superior GNU math support, which is not included by default for licensing reasons.

```
options "COMPAT_43"
```

Compatibility with 4.3BSD. Leave this in; some programs will act strangely if you comment this out.

```
options BOUNCE_BUFFERS
```

ISA devices and EISA devices operating in an ISA compatibility mode can only perform DMA (Direct Memory Access) to memory below 16 megabytes. This option enables such devices to work in systems with more than 16 megabytes of memory.

options UCONSOLE

Allow users to grab the console, useful for X Windows. For example, you can create a console xterm by typing `xterm -C`, which will display any `write`, `talk`, and other messages you receive, as well as any console messages sent by the kernel.

options SYSVSHM

This option provides for System V shared memory. The most common use of this is the XSHM extension in X Windows, which many graphics-intensive programs (such as the movie player XAnim, and Linux DOOM) will automatically take advantage of for extra speed. If you use the X Window System, you will definitely want to include this.

options SYSVSEM

Support for System V semaphores. Less commonly used but only adds a few hundred bytes to the kernel.

options SYSVMSG

Support for System V messages. Again, only adds a few hundred bytes to the kernel.

**Note:** The `ipcs(1)` command will tell will list any processes using each of these System V facilities.

## Filesystem Options

These options add support for various filesystems. You must include at least one of these to support the device you boot from; typically this will be FFS if you boot from a hard drive, or NFS if you are booting a diskless workstation from Ethernet. You can include other commonly-used filesystems in the kernel, but feel free to comment out support for filesystems you use less often (perhaps the MS-DOS filesystem?), since they will be dynamically loaded from the Loadable Kernel Module directory `/lkm` the first time you mount a partition of that type.

options FFS

The basic hard drive filesystem; leave it in if you boot from the hard disk.

options NFS

Network Filesystem. Unless you plan to mount partitions from a Unix file server over Ethernet, you can comment this out.

options MSDOSFS

MS-DOS Filesystem. Unless you plan to mount a DOS formatted hard drive partition at boot time, you can safely comment this out. It will be automatically loaded the first time you mount a DOS partition, as described above. Also, the excellent **mttools** software (in the ports collection) allows you to access DOS floppies without having to mount and unmount them (and does not require MSDOSFS at all).

options "CD9660"

ISO 9660 filesystem for CD-ROMs. Comment it out if you do not have a CD-ROM drive or only mount data CD's occasionally (since it will be dynamically loaded the first time you mount a data CD). Audio CD's do not need this filesystem.

options PROCFS

Process filesystem. This is a pretend filesystem mounted on `/proc` which allows programs like `ps(1)` to give you more information on what processes are running.

options MFS

Memory-mapped file system. This is basically a RAM disk for fast storage of temporary files, useful if you have a lot of swap space that you want to take advantage of. A perfect place to mount an MFS partition is on the `/tmp` directory, since many programs store temporary data here. To mount an MFS RAM disk on `/tmp`, add the following line to `/etc/fstab` and then reboot or type `mount /tmp`:

```
/dev/wd1s2b /tmp mfs rw 0 0
```

**Note:** Replace the `/dev/wd1s2b` with the name of your swap partition, which will be listed in your `/etc/fstab` as follows:

```
/dev/wd1s2b none swap sw 0 0
```

**Note:** Also, the MFS filesystem can *not* be dynamically loaded, so you *must* compile it into your kernel if you want to experiment with it.

```
options "EXT2FS"
```

Linux's native file system. With ext2fs support you are able to read and write to Linux partitions. This is useful if you dual-boot FreeBSD and Linux and want to share data between the two systems.

```
options QUOTA
```

Enable disk quotas. If you have a public access system, and do not want users to be able to overflow the /home partition, you can establish disk quotas for each user. Refer to the Disk Quotas section for more information.

## Basic Controllers and Devices

These sections describe the basic disk, tape, and CD-ROM controllers supported by FreeBSD. There are separate sections for SCSI controllers and network cards.

```
controller isa0
```

All PC's supported by FreeBSD have one of these. If you have an IBM PS/2 (Micro Channel Architecture), then you cannot run FreeBSD at this time.

```
controller pci0
```

Include this if you have a PCI motherboard. This enables auto-detection of PCI cards and gatewaying from the PCI to the ISA bus.

```
controller fdc0
```

Floppy drive controller: `fd0` is the A: floppy drive, and `fd1` is the B: drive. `ft0` is a QIC-80 tape drive attached to the floppy controller. Comment out any lines corresponding to devices you do not have.

**Note:** QIC-80 tape support requires a separate filter program called `ft(8)`, see the manual page for details.

```
controller wdc0
```

This is the primary IDE controller. `wd0` and `wd1` are the master and slave hard drive, respectively. `wdc1` is a secondary IDE controller where you might have a third or fourth hard drive, or an IDE

CD-ROM. Comment out the lines which do not apply (if you have a SCSI hard drive, you will probably want to comment out all six lines, for example).

```
device wcd0
```

This device provides IDE CD-ROM support. Be sure to leave `wcd0` uncommented, and `wcd1` if you have more than one IDE controller and your CD-ROM is on the second one card. To use this, you must also include the line `options ATAPI`.

```
device npx0 at isa? port "IO_NPX" irq 13 vector npxintr
```

`npx0` is the interface to the floating point math unit in FreeBSD, either the hardware co-processor or the software math emulator. It is *not* optional.

```
device wt0 at isa? port 0x300 bio irq 5 drq 1 vector wtintr
```

Wangtek and Archive QIC-02/QIC-36 tape drive support

#### Proprietary CD-ROM support

The following drivers are for the so-called *proprietary* CD-ROM drives. These drives have their own controller card or might plug into a sound card such as the SoundBlaster 16. They are *not* IDE or SCSI. Most older single-speed and double-speed CD-ROMs use these interfaces, while newer quad-speeds are likely to be IDE or SCSI.

```
device mcd0 at isa? port 0x300 bio irq 10 vector mcdintr
```

Mitsumi CD-ROM (LU002, LU005, FX001D).

```
device scd0 at isa? port 0x230 bio
```

Sony CD-ROM (CDU31, CDU33A).

```
controller matcd0 at isa? port ? bio
```

Matsushita/Panasonic CD-ROM (sold by Creative Labs for SoundBlaster).

## SCSI Device Support

This section describes the various SCSI controllers and devices supported by FreeBSD.

## SCSI Controllers

The next ten or so lines include support for different kinds of SCSI controllers. Comment out all except for the one(s) you have:

```
controller bt0 at isa? port "IO_BT0" bio irq ? vector btintr
```

Most Buslogic controllers

```
controller uha0 at isa? port "IO_UHA0" bio irq ? drq 5 vector uhaintr
```

UltraStor 14F and 34F

```
controller ahc0
```

Adaptec 274x/284x/294x

```
controller ahb0 at isa? bio irq ? vector ahbintr
```

Adaptec 174x

```
controller aha0 at isa? port "IO_AHA0" bio irq ? drq 5 vector ahaintr
```

Adaptec 154x

```
controller aic0 at isa? port 0x340 bio irq 11 vector aicintr
```

Adaptec 152x and sound cards using Adaptec AIC-6360 (slow!)

```
controller nca0 at isa? port 0x1f88 bio irq 10 vector ncaintr
```

ProAudioSpectrum cards using NCR 5380 or Trantor T130

```
controller sea0 at isa? bio irq 5 iomem 0xc8000 iosiz 0x2000 vector  
seaintr
```

Seagate ST01/02 8 bit controller (slow!)

```
controller wds0 at isa? port 0x350 bio irq 15 drq 6 vector wdsintr
```

Western Digital WD7000 controller

```
controller ncr0
```

NCR 53C810, 53C815, 53C825, 53C860, 53C875 PCI SCSI controller

```
options "SCSI_DELAY=15"
```

This causes the kernel to pause 15 seconds before probing each SCSI device in your system. If you only have IDE hard drives, you can ignore this, otherwise you will probably want to lower this number, perhaps to 5 seconds, to speed up booting. Of course if you do this, and FreeBSD has trouble recognizing your SCSI devices, you will have to raise it back up.

```
controller scbus0
```

If you have any SCSI controllers, this line provides generic SCSI support. If you do not have SCSI, you can comment this, and the following three lines, out.

```
device sd0
```

Support for SCSI hard drives.

```
device st0
```

Support for SCSI tape drives.

```
device cd0
```

Support for SCSI CD-ROM drives.

Note that the number 0 in the above entries is slightly misleading: all these devices are automatically configured as they are found, regardless of how many of them are hooked up to the SCSI bus(es), and which target IDs they have.

If you want to “wire down” specific target IDs to particular devices, refer to the appropriate section of the `LINT` kernel config file.

## Console, Bus Mouse, and X Server Support

You must choose one of these two console types, and, if you plan to use the X Window System with the vt220 console, enable the `XSERVER` option and optionally, a bus mouse or PS/2 mouse device.

```
device sc0 at isa? port "IO_KBD" tty irq 1 vector scintr
```

`sc0` is the default console driver, which resembles an SCO console. Since most full-screen programs access the console through a terminal database library like `termcap`, it should not matter much whether you use this or `vt0`, the VT220 compatible console driver. When you log in, set your `TERM` variable to “`scoansi`” if full-screen programs have trouble running under this console.

```
device vt0 at isa? port "IO_KBD" tty irq 1 vector pcrnt
```

This is a VT220-compatible console driver, backwards compatible to VT100/102. It works well on some laptops which have hardware incompatibilities with `sc0`. Also, set your `TERM` variable to `vt100` or `vt220` when you log in. This driver might also prove useful when connecting to a large number of different machines over the network, where the `termcap` or `terminfo` entries for the `sc0` device are often not available — `vt100` should be available on virtually any platform.

```
options "PCVT_FREEBSD=210"
```

Required with the `vt0` console driver.

```
options XSERVER
```

Only applicable with the `vt0` console driver. This includes code required to run the **XFree86** X Window Server under the `vt0` console driver.

```
device mse0 at isa? port 0x23c tty irq 5 vector ms
```

Use this device if you have a Logitech or ATI InPort bus mouse card.

**Note:** If you have a serial mouse, ignore these two lines, and instead, make sure the appropriate serial port is enabled (probably `COM1`).

```
device psm0 at isa? port "IO_KBD" conflicts tty irq 12 vector psmintr
```

Use this device if your mouse plugs into the PS/2 mouse port.

## Serial and Parallel Ports

Nearly all systems have these. If you are attaching a printer to one of these ports, the Printing section of the handbook is very useful. If you are using modem, Dialup access provides extensive detail on serial port configuration for use with such devices.

```
device sio0 at isa? port "IO_COM1" tty irq 4 vector siointr
```

`sio0` through `sio3` are the four serial ports referred to as `COM1` through `COM4` in the MS-DOS world. Note that if you have an internal modem on `COM4` and a serial port at `COM2` you will have to change the `IRQ` of the modem to 2 (for obscure technical reasons `IRQ 2 = IRQ 9`) in order to

access it from FreeBSD. If you have a multiport serial card, check the manual page for `sio(4)` for more information on the proper values for these lines. Some video cards (notably those based on S3 chips) use IO addresses of the form `0x*2e8`, and since many cheap serial cards do not fully decode the 16-bit IO address space, they clash with these cards, making the COM4 port practically unavailable.

Each serial port is required to have a unique IRQ (unless you are using one of the multiport cards where shared interrupts are supported), so the default IRQs for COM3 and COM4 cannot be used.

```
device lpt0 at isa? port? tty irq 7 vector lptintr
```

`lpt0` through `lpt2` are the three printer ports you could conceivably have. Most people just have one, though, so feel free to comment out the other two lines if you do not have them.

## Networking

FreeBSD, as with Unix in general, places a *big* emphasis on networking. Therefore, even if you do not have an Ethernet card, pay attention to the mandatory options and the dial-up networking support.

```
options INET
```

Networking support. Leave it in even if you do not plan to be connected to a network. Most programs require at least loopback networking (i.e. making network connections within your PC) so this is essentially mandatory.

### Ethernet cards

The next lines enable support for various Ethernet cards. If you do not have a network card, you can comment out all of these lines. Otherwise, you will want to leave in support for your particular Ethernet card(s):

```
device de0
```

Ethernet adapters based on Digital Equipment DC21040, DC21041 or DC21140 chips

```
device fxp0
```

Intel EtherExpress Pro/100B

```
device vx0
    3Com 3C590 and 3C595 (buggy)

device cx0 at isa? port 0x240 net irq 15 drq 7 vector cxintr
    Cronyx/Sigma multiport sync/async (with Cisco or PPP framing)

device ed0 at isa? port 0x280 net irq 5 iomem 0xd8000 vector edintr
    Western Digital and SMC 80xx and 8216; Novell NE1000 and NE2000; 3Com 3C503; HP PC
    Lan Plus (HP27247B and HP27252A)

device el0 at isa? port 0x300 net irq 9 vector elintr
    3Com 3C501 (slow!)

device eg0 at isa? port 0x310 net irq 5 vector egintr
    3Com 3C505

device ep0 at isa? port 0x300 net irq 10 vector epintr
    3Com 3C509 (buggy)

device fe0 at isa? port 0x240 net irq ? vector feintr
    Fujitsu MB86960A/MB86965A Ethernet

device fea0 at isa? net irq ? vector feaintr
    DEC DEFEA EISA FDDI adapter

device ie0 at isa? port 0x360 net irq 7 iomem 0xd0000 vector ieintr
    AT&T StarLAN 10 and EN100; 3Com 3C507; unknown NI5210; Intel EtherExpress 16

device le0 at isa? port 0x300 net irq 5 iomem 0xd0000 vector le_intr
    Digital Equipment EtherWorks 2 and EtherWorks 3 (DEPCA, DE100, DE101, DE200, DE201,
    DE202, DE203, DE204, DE205, DE422)

device lnc0 at isa? port 0x300 net irq 10 drq 0 vector lncintr
    Lance/PCnet cards (Isolan, Novell NE2100, NE32-VL)
```

device xl0

3Com Etherlink XL series PCI ethernet controllers (3C905B and related).

device ze0 at isa? port 0x300 net irq 5 iomem 0xd8000 vector zeintr

IBM/National Semiconductor PCMCIA ethernet controller.

device zp0 at isa? port 0x300 net irq 10 iomem 0xd8000 vector zpintr

3Com PCMCIA Etherlink III

**Note:** With certain cards (notably the NE2000) you will have to change the port and/or IRQ since there is no “standard” location for these cards.

pseudo-device loop

loop is the generic loopback device for TCP/IP. If you telnet or FTP to localhost (a.k.a. 127.0.0.1) it will come back at you through this pseudo-device. Mandatory.

pseudo-device ether

ether is only needed if you have an Ethernet card and includes generic Ethernet protocol code.

pseudo-device sl *number*

sl is for SLIP (Serial Line Internet Protocol) support. This has been almost entirely supplanted by PPP, which is easier to set up, better suited for modem-to-modem connections, as well as more powerful. The *number* after sl specifies how many simultaneous SLIP sessions to support. This handbook has more information on setting up a SLIP client or server.

pseudo-device ppp *number*

ppp is for kernel-mode PPP (Point-to-Point Protocol) support for dial-up Internet connections. There is also version of PPP implemented as a user application that uses the tun and offers more flexibility and features such as demand dialing. If you still want to use this PPP driver, read the kernel-mode PPP section of the handbook. As with the sl device, *number* specifies how many simultaneous PPP connections to support.

pseudo-device tun *number*

tun is used by the user-mode PPP software. This program is easy to set up and very fast. It also has special features such as automatic dial-on-demand. The number after tun specifies the number of

simultaneous PPP sessions to support. See the user-mode PPP section of the handbook for more information.

```
pseudo-device bpfiler number
```

Berkeley packet filter. This pseudo-device allows network interfaces to be placed in promiscuous mode, capturing every packet on a broadcast network (e.g. an ethernet). These packets can be captured to disk and/or examined with the `tcpdump(1)` program. Note that implementation of this capability can seriously compromise your overall network security. The *number* after `bpfiler` is the number of interfaces that can be examined simultaneously. Optional, not recommended except for those who are fully aware of the potential pitfalls. Not all network cards support this capability.

## Sound cards

This is the first section containing lines that are not in the GENERIC kernel. To include sound card support, you will have to copy the appropriate lines from the LINT kernel (which contains support for every device) as follows:

```
controller snd0
```

Generic sound driver code. Required for all of the following sound cards except `pca`.

```
device pas0 at isa? port 0x388 irq 10 drq 6 vector pasintr
```

ProAudioSpectrum digital audio and MIDI.

```
device sb0 at isa? port 0x220 irq 7 conflicts drq 1 vector sbintr
```

SoundBlaster digital audio.

**Note:** If your SoundBlaster is on a different IRQ (such as 5), change `irq 7` to, for example, `irq 5` and remove the `conflicts` keyword. Also, you must add the line: `options "SBC_IRQ=5"`

```
device sbxvi0 at isa? drq 5
```

SoundBlaster 16 digital 16-bit audio.

**Note:** If your SB16 is on a different 16-bit DMA channel (such as 6 or 7), change the `drq 5` keyword appropriately, and then add the line: `options "SB16_DMA=6"`

```
device sbmidi0 at isa? port 0x330
```

SoundBlaster 16 MIDI interface. If you have a SoundBlaster 16, you must include this line, or the kernel will not compile.

```
device gus0 at isa? port 0x220 irq 10 drq 1 vector gusintr
```

Gravis Ultrasound.

```
device mss0 at isa? port 0x530 irq 10 drq 1 vector adintr
```

Microsoft Sound System.

```
device opl0 at isa? port 0x388 conflicts
```

AdLib FM-synthesis audio. Include this line for AdLib, SoundBlaster, and ProAudioSpectrum users, if you want to play MIDI songs with a program such as `playmidi` (in the ports collection).

```
device mpu0 at isa? port 0x330 irq 6 drq 0
```

Roland MPU-401 stand-alone card.

```
device uart0 at isa? port 0x330 irq 5 vector "m6850intr"
```

Stand-alone 6850 UART for MIDI.

```
device pca0 at isa? port "IO_TIMER1" tty
```

Digital audio through PC speaker. This is going to be very poor sound quality and quite CPU-intensive, so you have been warned (but it does not require a sound card).

**Note:** There is some additional documentation in `/usr/src/sys/i386/isa/sound/sound.doc`. Also, if you add any of these devices, be sure to create the sound device nodes.

## Pseudo-devices

Pseudo-device drivers are parts of the kernel that act like device drivers but do not correspond to any actual hardware in the machine. The network-related pseudo-devices are in that section, while the remainder are here.

`pseudo-device gzip`

`gzip` allows you to run FreeBSD programs that have been compressed with `gzip`. The programs in `/stand` are compressed so it is a good idea to have this option in your kernel.

`pseudo-device log`

`log` is used for logging of kernel error messages. Mandatory.

`pseudo-device pty number`

`pty` is a “pseudo-terminal” or simulated login port. It is used by incoming `telnet` and `rlogin` sessions, `xterm`, and some other applications such as `emacs`. The *number* indicates the number of `ptys` to create. If you need more than `GENERIC` default of 16 simultaneous `xterm` windows and/or remote logins, be sure to increase this number accordingly, up to a maximum of 256.

`pseudo-device snp number`

Snoop device. This pseudo-device allows one terminal session to watch another using the `watch(8)` command. Note that implementation of this capability has important security and privacy implications. The *number* after `snp` is the total number of simultaneous snoop sessions. Optional.

`pseudo-device vn`

Vnode driver. Allows a file to be treated as a device after being set up with the `vnconfig(8)` command. This driver can be useful for manipulating floppy disk images and using a file as a swap device (e.g. an MS Windows swap file). Optional.

`pseudo-device ccd number`

Concatenated disks. This pseudo-device allows you to concatenate multiple disk partitions into one large “meta”-disk. The *number* after `ccd` is the total number of concatenated disks (not total number of disks that can be concatenated) that can be created. (See `ccd(4)` and `ccdconfig(8)` man pages for more details.) Optional.

## Joystick, PC Speaker, Miscellaneous

This section describes some miscellaneous hardware devices supported by FreeBSD. Note that none of these lines are included in the `GENERIC` kernel, you will have to copy them from this handbook or the `LINT` kernel (which contains support for *every* device):

```
device joy0 at isa? port "IO_GAME"
```

PC joystick device.

```
pseudo-device speaker
```

Supports IBM BASIC-style noises through the PC speaker. Some fun programs which use this are `/usr/sbin/spkrtest`, which is a shell script that plays some simple songs, and `/usr/games/piano` which lets you play songs using the keyboard as a simple piano (this file only exists if you have installed the `games` package). Also, the excellent text role-playing game **NetHack** (in the `ports` collection) can be configured to use this device to play songs when you play musical instruments in the game.

See also the `pca0` device.

## Making Device Nodes

Almost every device in the kernel has a corresponding “node” entry in the `/dev` directory. These nodes look like regular files, but are actually special entries into the kernel which programs use to access the device. The shell script `/dev/MAKEDEV`, which is executed when you first install the operating system, creates nearly all of the device nodes supported. However, it does not create *all* of them, so when you add support for a new device, it pays to make sure that the appropriate entries are in this directory, and if not, add them. Here is a simple example:

Suppose you add the IDE CD-ROM support to the kernel. The line to add is:

```
controller wcd0
```

This means that you should look for some entries that start with `wcd0` in the `/dev` directory, possibly followed by a letter, such as `c`, or preceded by the letter `r`, which means a “raw” device. It turns out that those files are not there, so I must change to the `/dev` directory and type:

```
# sh MAKEDEV wcd0
```

When this script finishes, you will find that there are now `wcd0c` and `rwcd0c` entries in `/dev` so you know that it executed correctly.

For sound cards, the command:

```
# sh MAKEDEV snd0
```

creates the appropriate entries.

**Note:** When creating device nodes for devices such as sound cards, if other people have access to your machine, it may be desirable to protect the devices from outside access by adding them to the `/etc/fstab` file. See `man fstab` for more information.

Follow this simple procedure for any other non-`GENERIC` devices which do not have entries.

**Note:** All SCSI controllers use the same set of `/dev` entries, so you do not need to create these. Also, network cards and SLIP/PPP pseudo-devices do not have entries in `/dev` at all, so you do not have to worry about these either.

## If Something Goes Wrong

There are four categories of trouble that can occur when building a custom kernel. They are:

### Config command fails

If the `config` command fails when you give it your kernel description, you have probably made a simple error somewhere. Fortunately, `config` will print the line number that it had trouble with, so you can quickly skip to it with `vi`. For example, if you see:

```
config: line 17: syntax error
```

you can skip to the problem in `vi` by typing `17G` in command mode. Make sure the keyword is typed correctly, by comparing it to the `GENERIC` kernel or another reference.

### Make command fails

If the `make` command fails, it usually signals an error in your kernel description, but not severe enough for `config` to catch it. Again, look over your configuration, and if you still cannot resolve the problem, send mail to the FreeBSD general questions mailing list `<freebsd-questions@FreeBSD.ORG>` with your kernel configuration, and it should be diagnosed very quickly.

### Kernel will not boot

If your new kernel does not boot, or fails to recognize your devices, do not panic! Fortunately, BSD has an excellent mechanism for recovering from incompatible kernels. Simply type the name of the kernel you want to boot from (i.e. `kernel.old`) at the FreeBSD boot prompt instead of pressing return. When reconfiguring a kernel, it is always a good idea to keep a kernel that is known to work on hand.

After booting with a good kernel you can check over your configuration file and try to build it again. One helpful resource is the `/var/log/messages` file which records, among other things, all of the kernel messages from every successful boot. Also, the `dmesg(8)` command will print the kernel messages from the current boot.

**Note:** If you are having trouble building a kernel, make sure to keep a `GENERIC`, or some other kernel that is known to work on hand as a different name that will not get erased on the next build. You cannot rely on `kernel.old` because when installing a new kernel, `kernel.old` is overwritten with the last installed kernel which may be non-functional. Also, as soon as possible, move the working kernel to the proper `kernel` location or commands such as `ps(1)` will not work properly. The proper command to “unlock” the kernel file that `make` installs (in order to move another kernel back permanently) is:

```
# chflags noschg /kernel
```

And, if you want to “lock” your new kernel into place, or any file for that matter, so that it cannot be moved or tampered with:

```
# chflags schg /kernel
```

Kernel works, but `ps` does not work any more!

If you have installed a different version of the kernel from the one that the system utilities have been built with, for example, an experimental “2.2.0” kernel on a 2.1.0-RELEASE system, many system-status commands like `ps(1)` and `vmstat(8)` will not work any more. You must recompile the `libkvm` library as well as these utilities. This is one reason it is not normally a good idea to use a different version of the kernel from the rest of the operating system.

# Chapter 6. Security

## DES, MD5, and Crypt

*Contributed by Garrett Wollman <wollman@FreeBSD.ORG> 24 September 1995.*

In order to protect the security of passwords on UN\*X systems from being easily exposed, passwords have traditionally been scrambled in some way. Starting with Bell Labs' Seventh Edition Unix, passwords were encrypted using what the security people call a "one-way hash function". That is to say, the password is transformed in such a way that the original password cannot be regained except by brute-force searching the space of possible passwords. Unfortunately, the only secure method that was available to the AT&T researchers at the time was based on DES, the Data Encryption Standard. This causes only minimal difficulty for commercial vendors, but is a serious problem for an operating system like FreeBSD where all the source code is freely available, because national governments in many places like to place restrictions on cross-border transport of DES and other encryption software.

So, the FreeBSD team was faced with a dilemma: how could we provide compatibility with all those UNIX systems out there while still not running afoul of the law? We decided to take a dual-track approach: we would make distributions which contained only a non-regulated password scrambler, and then provide as a separate add-on library the DES-based password hash. The password-scrambling function was moved out of the C library to a separate library, called `libcrypt` because the name of the C function to implement it is `crypt`. In FreeBSD 1.x and some pre-release 2.0 snapshots, the non-regulated scrambler uses an insecure function written by Nate Williams; in subsequent releases this was replaced by a mechanism using the RSA Data Security, Inc., MD5 one-way hash function. Because neither of these functions involve encryption, they are believed to be exportable from the US and importable into many other countries.

Meanwhile, work was also underway on the DES-based password hash function. First, a version of the `crypt` function which was written outside the US was imported, thus synchronizing the US and non-US code. Then, the library was modified and split into two; the DES `libcrypt` contains only the code involved in performing the one-way password hash, and a separate `libcipher` was created with the entry points to actually perform encryption. The code was partitioned in this way to make it easier to get an export license for the compiled library.

### Recognizing your `crypt` mechanism

It is fairly easy to recognize whether a particular password string was created using the DES- or MD5-based hash function. MD5 password strings always begin with the characters `$1$`. DES password strings do not have any particular identifying characteristics, but they are shorter than MD5 passwords,

and are coded in a 64-character alphabet which does not include the \$ character, so a relatively short string which doesn't begin with a dollar sign is very likely a DES password.

Determining which library is being used on your system is fairly easy for most programs, except for those like `init` which are statically linked. (For those programs, the only way is to try them on a known password and see if it works.) Programs which use `crypt` are linked against `libcrypt`, which for each type of library is a symbolic link to the appropriate implementation. For example, on a system using the DES versions:

```
% cd /usr/lib
% ls -l /usr/lib/libcrypt*
lrwxr-xr-x 1 bin bin 13 Sep  5 12:50 libcrypt.a -> libdescript.a
lrwxr-xr-x 1 bin bin 18 Sep  5 12:50 libcrypt.so.2.0 -> libdescript.so.2.0
lrwxr-xr-x 1 bin bin 15 Sep  5 12:50 libcrypt_p.a -> libdescript_p.a
```

On a system using the MD5-based libraries, the same links will be present, but the target will be `libcrypt` rather than `libdescript`.

## S/Key

*Contributed by Garrett Wollman <wollman@FreeBSD.ORG> 25 September 1995.*

S/Key is a one-time password scheme based on a one-way hash function (in our version, this is MD4 for compatibility; other versions have used MD5 and DES-MAC). S/Key has been a standard part of all FreeBSD distributions since version 1.1.5, and is also implemented on a large and growing number of other systems. S/Key is a registered trademark of Bell Communications Research, Inc.

There are three different sorts of passwords which we will talk about in the discussion below. The first is your usual UNIX-style or Kerberos password; we will call this a “UNIX password”. The second sort is the one-time password which is generated by the S/Key `key` program and accepted by the `keyinit` program and the login prompt; we will call this a “one-time password”. The final sort of password is the secret password which you give to the `key` program (and sometimes the `keyinit` program) which it uses to generate one-time passwords; we will call it a “secret password” or just unqualified “password”.

The secret password does not necessarily have anything to do with your UNIX password (while they can be the same, this is not recommended). While UNIX passwords are limited to eight characters in length, your S/Key secret password can be as long as you like; I use seven-word phrases. In general, the S/Key system operates completely independently of the UNIX password system.

There are in addition two other sorts of data involved in the S/Key system; one is called the “seed” or (confusingly) “key”, and consists of two letters and five digits, and the other is the “iteration count” and is a number between 100 and 1. S/Key constructs a one-time password from these components by

concatenating the seed and the secret password, then applying a one-way hash (the RSA Data Security, Inc., MD4 secure hash function) iteration-count times, and turning the result into six short English words. The `login` and `su` programs keep track of the last one-time password used, and the user is authenticated if the hash of the user-provided password is equal to the previous password. Because a one-way hash function is used, it is not possible to generate future one-time passwords having overheard one which was successfully used; the iteration count is decremented after each successful login to keep the user and login program in sync. (When you get the iteration count down to 1, it is time to reinitialize S/Key.)

There are four programs involved in the S/Key system which we will discuss below. The `key` program accepts an iteration count, a seed, and a secret password, and generates a one-time password. The `keyinit` program is used to initialize S/Key, and to change passwords, iteration counts, or seeds; it takes either a secret password, or an iteration count, seed, and one-time password. The `keyinfo` program examines the `/etc/skeykeys` file and prints out the invoking user's current iteration count and seed. Finally, the `login` and `su` programs contain the necessary logic to accept S/Key one-time passwords for authentication. The `login` program is also capable of disallowing the use of UNIX passwords on connections coming from specified addresses.

There are four different sorts of operations we will cover. The first is using the `keyinit` program over a secure connection to set up S/Key for the first time, or to change your password or seed. The second operation is using the `keyinit` program over an insecure connection, in conjunction with the `key` program over a secure connection, to do the same. The third is using the `key` program to log in over an insecure connection. The fourth is using the `key` program to generate a number of keys which can be written down or printed out to carry with you when going to some location without secure connections to anywhere (like at a conference).

## Secure connection initialization

To initialize S/Key, change your password, or change your seed while logged in over a secure connection (e.g., on the console of a machine), use the `keyinit` command without any parameters while logged in as yourself:

```
% keyinit
Updating wollman: ) these will not appear if you
Old key: ha73895 ) have not used S/Key before
Reminder - Only use this method if you are directly connected.
If you are using telnet or rlogin exit with no password and use keyinit -s.
Enter secret password: ) I typed my pass phrase here
Again secret password: ) I typed it again ID

wollman s/key is 99 ha73896 ) discussed below SAG
HAS FONT GOUT FATE BOOM )
```

There is a lot of information here. At the `Enter secret password:` prompt, you should enter some password or phrase (I use phrases of minimum seven words) which will be needed to generate login keys. The line starting 'ID' gives the parameters of your particular S/Key instance: your login name, the iteration count, and seed. When logging in with S/Key, the system will remember these parameters and present them back to you so you do not have to remember them. The last line gives the particular one-time password which corresponds to those parameters and your secret password; if you were to re-login immediately, this one-time password is the one you would use.

## Insecure connection initialization

To initialize S/Key or change your password or seed over an insecure connection, you will need to already have a secure connection to some place where you can run the `key` program; this might be in the form of a desk accessory on a Macintosh, or a shell prompt on a machine you trust (we will show the latter). You will also need to make up an iteration count (100 is probably a good value), and you may make up your own seed or use a randomly-generated one. Over on the insecure connection (to the machine you are initializing), use the `keyinit -s` command:

```
% keyinit -s
Updating wollman: Old key: kh94741
Reminder you need the 6 English words from the skey command.
Enter sequence count from 1 to 9999: 100 ) I typed this
Enter new key [default kh94742]:
s/key 100 kh94742
```

To accept the default seed (which the `keyinit` program confusingly calls a `key`), press return. Then move over to your secure connection or S/Key desk accessory, and give it the same parameters:

```
% key 100 kh94742
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password:                ) I typed my secret password
HULL NAY YANG TREE TOUT VETO
```

Now switch back over to the insecure connection, and copy the one-time password generated by `key` over to the `keyinit` program:

```
s/key access password: HULL NAY YANG TREE TOUT VETO
ID wollman s/key is 100 kh94742
HULL NAY YANG TREE TOUT VETO
```

The rest of the description from the previous section applies here as well.

## Diversion: a login prompt

Before explaining how to generate one-time passwords, we should go over an S/Key login prompt:

```
% telnet himalia
Trying 18.26.0.186...
Connected to himalia.lcs.mit.edu.
Escape character is '^]'.
s/key 92 hi52030
Password:
```

Note that, before prompting for a password, the login program prints out the iteration number and seed which you will need in order to generate the appropriate key. You will also find a useful feature (not shown here): if you press return at the password prompt, the login program will turn echo on, so you can see what you are typing. This can be extremely useful if you are attempting to type in an S/Key by hand, such as from a printout.

If this machine were configured to disallow UNIX passwords over a connection from my machine, the prompt would have also included the annotation (*s/key required*), indicating that only S/Key one-time passwords will be accepted.

## Generating a single one-time password

Now, to generate the one-time password needed to answer this login prompt, we use a trusted machine and the `key` program. (There are versions of the `key` program from DOS and Windows machines, and there is an S/Key desk accessory for Macintosh computers as well.) The command-line `key` program takes as its parameters the iteration count and seed; you can cut-and-paste right from the login prompt starting at `key` to the end of the line. Thus:

```
% key 92 hi52030 ) pasted from previous section
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password: ) I typed my secret password
ADEN BED WOLF HAW HOT STUN
```

And in the other window:

```
s/key 92 hi52030 ) from previous section
Password:
  (turning echo on)
Password:ADEN BED WOLF HAW HOT STUN
Last login: Wed Jun 28 15:31:00 from halloran-eldar.1
[etc.]
```

This is the easiest mechanism *if* you have a trusted machine. There is a Java S/Key `key` applet, The Java OTP Calculator (<http://www.cs.umd.edu/~harry/jotp/src.html>), that you can download and run locally on any Java supporting browser.

## Generating multiple one-time passwords

Sometimes we have to go places where no trusted machines or connections are available. In this case, it is possible to use the `key` command to generate a number of one-time passwords in the same command; these can then be printed out. For example:

```
% key -n 25 57 zz99999
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password:
33: WALT THY MALI DARN NIT HEAD
34: ASK RICE BEAU GINA DOUR STAG
...
56: AMOS BOWL LUG FAT CAIN INCH
57: GROW HAYS TUN DISH CAR BALM
```

The `-n 25` requests twenty-five keys in sequence; the `57` indicates the *ending* iteration number; and the rest is as before. Note that these are printed out in *reverse* order of eventual use. If you are really paranoid, you might want to write the results down by hand; otherwise you can cut-and-paste into `lpr`. Note that each line shows both the iteration count and the one-time password; you may still find it handy to scratch off passwords as you use them.

## Restricting use of UNIX passwords

The configuration file `/etc/skey.access` can be used to configure restrictions on the use of UNIX passwords based on the host name, user name, terminal port, or IP address of a login session. The complete format of the file is documented in the `skey.access(5)` manual page; there are also some security cautions there which should be read before depending on this file for security.

If there is no `/etc/skey.access` file (which is the default state as FreeBSD is shipped), then all users will be allowed to use UNIX passwords. If the file exists, however, then all users will be required to use S/Key unless explicitly permitted to do otherwise by configuration statements in the `skey.access` file. In all cases, UNIX passwords are permitted on the console.

Here is a sample configuration file which illustrates the three most common sorts of configuration statements:

```
permit internet 18.26.0.0 255.255.0.0
```

```

permit user jrl
permit port ttyd0

```

The first line (`permit internet`) allows users whose IP source address (which is vulnerable to spoofing) matches the specified value and mask, to use UNIX passwords. This should not be considered a security mechanism, but rather, a means to remind authorized users that they are using an insecure network and need to use S/Key for authentication.

The second line (`permit user`) allows the specified user to use UNIX passwords at any time. Generally speaking, this should only be used for people who are either unable to use the key program, like those with dumb terminals, or those who are uneducable.

The third line (`permit port`) allows all users logging in on the specified terminal line to use UNIX passwords; this would be used for dial-ups.

## Kerberos

*Contributed by Mark Murray <markm@FreeBSD.ORG> (based on contribution by Mark Dapoz <md@bsc.no>).*

Kerberos is a network add-on system/protocol that allows users to authenticate themselves through the services of a secure server. Services such as remote login, remote copy, secure inter-system file copying and other high-risk tasks are made considerably safer and more controllable.

The following instructions can be used as a guide on how to set up Kerberos as distributed for FreeBSD. However, you should refer to the relevant manual pages for a complete description.

In FreeBSD, the Kerberos is not that from the original 4.4BSD-Lite, distribution, but eBones, which had been previously ported to FreeBSD 1.1.5.1, and was sourced from outside the USA/Canada, and is thus available to system owners outside those countries.

For those needing to get a legal foreign distribution of this software, please *do not* get it from a USA or Canada site. You will get that site in *big* trouble! A legal copy of this is available from `ftp.internat.freebsd.org`, which is in South Africa and an official FreeBSD mirror site.

## Creating the initial database

This is done on the Kerberos server only. First make sure that you do not have any old Kerberos databases around. You should change to the directory `/etc/kerberosIV` and check that only the following files are present:

```
# cd /etc/kerberosIV
```

```
# ls
README krb.conf          krb.realms
```

If any additional files (such as `principal.*` or `master_key`) exist, then use the `kdb_destroy` command to destroy the old Kerberos database, or if Kerberos is not running, simply delete the extra files.

You should now edit the `krb.conf` and `krb.realms` files to define your Kerberos realm. In this case the realm will be `GRONDAR.ZA` and the server is `grunt.grondar.za`. We edit or create the `krb.conf` file:

```
# cat krb.conf
GRONDAR.ZA
GRONDAR.ZA grunt.grondar.za admin server
CS.BERKELEY.EDU okeeffe.berkeley.edu
ATHENA.MIT.EDU kerberos.mit.edu
ATHENA.MIT.EDU kerberos-1.mit.edu
ATHENA.MIT.EDU kerberos-2.mit.edu
ATHENA.MIT.EDU kerberos-3.mit.edu
LCS.MIT.EDU kerberos.lcs.mit.edu
TELECOM.MIT.EDU bitsy.mit.edu
ARC.NASA.GOV trident.arc.nasa.gov
```

In this case, the other realms do not need to be there. They are here as an example of how a machine may be made aware of multiple realms. You may wish to not include them for simplicity.

The first line names the realm in which this system works. The other lines contain realm/host entries. The first item on a line is a realm, and the second is a host in that realm that is acting as a “key distribution centre”. The words `admin server` following a hosts name means that host also provides an administrative database server. For further explanation of these terms, please consult the Kerberos man pages.

Now we have to add `grunt.grondar.za` to the `GRONDAR.ZA` realm and also add an entry to put all hosts in the `.grondar.za` domain in the `GRONDAR.ZA` realm. The `krb.realms` file would be updated as follows:

```
# cat krb.realms
grunt.grondar.za GRONDAR.ZA
.grondar.za GRONDAR.ZA
.berkeley.edu CS.BERKELEY.EDU
.MIT.EDU ATHENA.MIT.EDU
.mit.edu ATHENA.MIT.EDU
```

Again, the other realms do not need to be there. They are here as an example of how a machine may be made aware of multiple realms. You may wish to remove them to simplify things.

The first line puts the *specific* system into the named realm. The rest of the lines show how to default systems of a particular subdomain to a named realm.

Now we are ready to create the database. This only needs to run on the Kerberos server (or Key Distribution Centre). Issue the `kdb_init` command to do this:

```
# kdb_init
Realm name [default ATHENA.MIT.EDU ]: GRONDAR.ZA
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
```

```
Enter Kerberos master key:
```

Now we have to save the key so that servers on the local machine can pick it up. Use the `kstash` command to do this.

```
# kstash
Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered. BEWARE!
```

This saves the encrypted master password in `/etc/kerberosIV/master_key`.

## Making it all run

Two principals need to be added to the database for *each* system that will be secured with Kerberos. Their names are `kpasswd` and `rcmd`. These two principals are made for each system, with the instance being the name of the individual system.

These daemons, `kpasswd` and `rcmd` allow other systems to change Kerberos passwords and run commands like `rcp`, `rlogin` and `rsh`.

Now let's add these entries:

```
# kdb_edit
Opening database...

Enter Kerberos master key:

Current Kerberos master key version is 1.
```

```
Master key entered.  BEWARE!
Previous or default values are in [brackets] ,
enter return to leave the same, or new value.

Principal name: passwd
Instance: grunt

<Not found>, Create [y] ? y

Principal: passwd, Instance: grunt, kdc_key_ver: 1
New Password:          <--- enter RANDOM here
Verifying password

New Password: <--- enter RANDOM here

Random password [y] ? y

Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name: rcmd
Instance: grunt

<Not found>, Create [y] ?

Principal: rcmd, Instance: grunt, kdc_key_ver: 1
New Password: <--- enter RANDOM here
Verifying password

New Password:          <--- enter RANDOM here

Random password [y] ?

Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name:          <--- null entry here will cause an exit
```

## Creating the server file

We now have to extract all the instances which define the services on each machine. For this we use the `ext_srvtab` command. This will create a file which must be copied or moved *by secure means* to each Kerberos client's `/etc/kerberosIV` directory. This file must be present on each server and client, and is crucial to the operation of Kerberos.

```
# ext_srvtab grunt
Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered. BEWARE!
Generating 'grunt-new-srvtab'....
```

Now, this command only generates a temporary file which must be renamed to `srvtab` so that all the server can pick it up. Use the `mv` command to move it into place on the original system:

```
# mv grunt-new-srvtab srvtab
```

If the file is for a client system, and the network is not deemed safe, then copy the `client-new-srvtab` to removable media and transport it by secure physical means. Be sure to rename it to `srvtab` in the client's `/etc/kerberosIV` directory, and make sure it is mode 600:

```
# mv grumble-new-srvtab srvtab
# chmod 600 srvtab
```

## Populating the database

We now have to add some user entries into the database. First let's create an entry for the user `jane`. Use the `kdb_edit` command to do this:

```
# kdb_edit
Opening database...

Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered. BEWARE!
Previous or default values are in [brackets] ,
enter return to leave the same, or new value.
```

```

Principal name: jane
Instance:

<Not found>, Create [y] ? y

Principal: jane, Instance: , kdc_key_ver: 1
New Password:          <--- enter a secure password here
Verifying password

New Password:          <--- re-enter the password here
Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name:       <--- null entry here will cause an exit

```

## Testing it all out

First we have to start the Kerberos daemons. NOTE that if you have correctly edited your `/etc/rc.conf` then this will happen automatically when you reboot. This is only necessary on the Kerberos server. Kerberos clients will automatically get what they need from the `/etc/kerberosIV` directory.

```

# kerberos &
Kerberos server starting
Sleep forever on error
Log file is /var/log/kerberos.log
Current Kerberos master key version is 1.

Master key entered. BEWARE!

Current Kerberos master key version is 1
Local realm: GRONDAR.ZA
# kadmind -n &
KADM Server KADM0.0A initializing
Please do not use 'kill -9' to kill this job, use a
regular kill instead

Current Kerberos master key version is 1.

Master key entered. BEWARE!

```

Now we can try using the `kinit` command to get a ticket for the id `jane` that we created above:

```
% kinit jane
MIT Project Athena (grunt.grondar.za)
Kerberos Initialization for "jane"
Password:
```

Try listing the tokens using `klist` to see if we really have them:

```
% klist
Ticket file:      /tmp/tkt245
Principal:       jane@GRONDAR.ZA

    Issued                Expires                Principal
Apr 30 11:23:22  Apr 30 19:23:22  krbtgt.GRONDAR.ZA@GRONDAR.ZA
```

Now try changing the password using `passwd` to check if the `kpasswd` daemon can get authorization to the Kerberos database:

```
% passwd
realm GRONDAR.ZA
Old password for jane:
New Password for jane:
Verifying password
New Password for jane:
Password changed.
```

## Adding `su` privileges

Kerberos allows us to give *each* user who needs root privileges their own *separate* `supassword`. We could now add an id which is authorized to `su` to `root`. This is controlled by having an instance of `root` associated with a principal. Using `kdb_edit` we can create the entry `jane.root` in the Kerberos database:

```
# kdb_edit
Opening database...

Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered.  BEWARE!
Previous or default values are in [brackets] ,
```

enter return to leave the same, or new value.

Principal name: **jane**  
Instance: **root**

<Not found>, Create [y] ? y

Principal: jane, Instance: root, kdc\_key\_ver: 1  
New Password: <--- enter a SECURE password here  
Verifying password

New Password: <--- re-enter the password here

Principal's new key version = 1  
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?  
Max ticket lifetime (\*5 minutes) [ 255 ] ? **12** <-- Keep this short!  
Attributes [ 0 ] ?  
Edit O.K.  
Principal name: <--- null entry here will cause an exit

Now try getting tokens for it to make sure it works:

```
# kinit jane.root
MIT Project Athena (grunt.grondar.za)
Kerberos Initialization for "jane.root"
Password:
```

Now we need to add the user to root's .klogin file:

```
# cat /root/.klogin
jane.root@GRONDAR.ZA
```

Now try doing the su:

```
% su
Password:
```

and take a look at what tokens we have:

```
# klist
Ticket file: /tmp/tkt_root_245
Principal: jane.root@GRONDAR.ZA

    Issued                Expires                Principal
May  2 20:43:12  May  3 04:43:12  krbtgt.GRONDAR.ZA@GRONDAR.ZA
```

## Using other commands

In an earlier example, we created a principal called `jane` with an instance `root`. This was based on a user with the same name as the principal, and this is a Kerberos default; that a `<principal>.<instance>` of the form `<username>.root` will allow that `<username>` to `su` to `root` if the necessary entries are in the `.klogin` file in `root`'s home directory:

```
# cat /root/.klogin
jane.root@GRONDAR.ZA
```

Likewise, if a user has in their own home directory lines of the form:

```
% cat ~/.klogin
jane@GRONDAR.ZA
jack@GRONDAR.ZA
```

This allows anyone in the `GRONDAR.ZA` realm who has authenticated themselves to `jane` or `jack` (via `kinit`, see above) access to `rlogin` to `jane`'s account or files on this system (`grunt`) via `rlogin`, `rsh` or `rcp`.

For example, Jane now logs into another system, using Kerberos:

```
% kinit
MIT Project Athena (grunt.grondar.za)
Password:
%prompt.user; rlogin grunt
Last login: Mon May  1 21:14:47 from grumble
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
    The Regents of the University of California.  All rights reserved.

FreeBSD BUILT-19950429 (GR386) #0: Sat Apr 29 17:50:09 SAT 1995
```

Or Jack logs into Jane's account on the same machine (Jane having set up the `.klogin` file as above, and the person in charge of Kerberos having set up principal `jack` with a null instance:

```
% kinit
% rlogin grunt -l jane
MIT Project Athena (grunt.grondar.za)
Password:
Last login: Mon May  1 21:16:55 from grumble
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
    The Regents of the University of California.  All rights reserved.

FreeBSD BUILT-19950429 (GR386) #0: Sat Apr 29 17:50:09 SAT 1995
```

## Firewalls

*Contributed by Gary Palmer <gpalmer@FreeBSD.ORG> and Alex Nash <alex@freebsd.org>.*

Firewalls are an area of increasing interest for people who are connected to the Internet, and are even finding applications on private networks to provide enhanced security. This section will hopefully explain what firewalls are, how to use them, and how to use the facilities provided in the FreeBSD kernel to implement them.

**Note:** People often think that having a firewall between your companies internal network and the “Big Bad Internet” will solve all your security problems.

It may help, but a poorly setup firewall system is more of a security risk than not having one at all. A firewall can only add another layer of security to your systems, but they will not be able to stop a really determined cracker from penetrating your internal network. If you let internal security lapse because you believe your firewall to be impenetrable, you have just made the crackers job that bit easier.

### What is a firewall?

There are currently two distinct types of firewalls in common use on the Internet today. The first type is more properly called a *packet filtering router*, where the kernel on a multi-homed machine chooses whether to forward or block packets based on a set of rules. The second type, known as *proxy servers*, rely on daemons to provide authentication and to forward packets, possibly on a multi-homed machine which has kernel packet forwarding disabled.

Sometimes sites combine the two types of firewalls, so that only a certain machine (known as a *bastion host*) is allowed to send packets through a packet filtering router onto an internal network. Proxy services are run on the bastion host, which are generally more secure than normal authentication mechanisms.

FreeBSD comes with a kernel packet filter (known as **IPFW**), which is what the rest of this section will concentrate on. Proxy servers can be built on FreeBSD from third party software, but there is such a variety of proxy servers available that it would be impossible to cover them in this document.

### Packet filtering routers

A router is a machine which forwards packets between two or more networks. A packet filtering router has an extra piece of code in its kernel, which compares each packet to a list of rules before deciding if it should be forwarded or not. Most modern IP routing software has packet filtering code in it, which defaults to forwarding all packets. To enable the filters, you need to define a set of rules for the filtering code, so that it can decide if the packet should be allowed to pass or not.

To decide if a packet should be passed on or not, the code looks through its set of rules for a rule which matches the contents of this packets headers. Once a match is found, the rule action is obeyed. The rule

action could be to drop the packet, to forward the packet, or even to send an ICMP message back to the originator. Only the first match counts, as the rules are searched in order. Hence, the list of rules can be referred to as a “rule chain”.

The packet matching criteria varies depending on the software used, but typically you can specify rules which depend on the source IP address of the packet, the destination IP address, the source port number, the destination port number (for protocols which support ports), or even the packet type (UDP, TCP, ICMP, etc).

## Proxy servers

Proxy servers are machines which have had the normal system daemons (telnetd, ftpd, etc) replaced with special servers. These servers are called *proxy servers* as they normally only allow onward connections to be made. This enables you to run (for example) a proxy telnet server on your firewall host, and people can telnet in to your firewall from the outside, go through some authentication mechanism, and then gain access to the internal network (alternatively, proxy servers can be used for signals coming from the internal network and heading out).

Proxy servers are normally more secure than normal servers, and often have a wider variety of authentication mechanisms available, including “one-shot” password systems so that even if someone manages to discover what password you used, they will not be able to use it to gain access to your systems as the password instantly expires. As they do not actually give users access to the host machine, it becomes a lot more difficult for someone to install backdoors around your security system.

Proxy servers often have ways of restricting access further, so that only certain hosts can gain access to the servers, and often they can be set up so that you can limit which users can talk to which destination machine. Again, what facilities are available depends largely on what proxy software you choose.

## What does IPFW allow me to do?

**IPFW**, the software supplied with FreeBSD, is a packet filtering and accounting system which resides in the kernel, and has a user-land control utility, `ipfw(8)`. Together, they allow you to define and query the rules currently used by the kernel in its routing decisions.

There are two related parts to **IPFW**. The firewall section allows you to perform packet filtering. There is also an IP accounting section which allows you to track usage of your router, based on similar rules to the firewall section. This allows you to see (for example) how much traffic your router is getting from a certain machine, or how much WWW (World Wide Web) traffic it is forwarding.

As a result of the way that **IPFW** is designed, you can use **IPFW** on non-router machines to perform packet filtering on incoming and outgoing connections. This is a special case of the more general use of

**IPFW**, and the same commands and techniques should be used in this situation.

## Enabling IPFW on FreeBSD

As the main part of the **IPFW** system lives in the kernel, you will need to add one or more options to your kernel configuration file, depending on what facilities you want, and recompile your kernel. See reconfiguring the kernel for more details on how to recompile your kernel.

There are currently three kernel configuration options relevant to **IPFW**:

```
options IPFIREWALL
```

Compiles into the kernel the code for packet filtering.

```
options IPFIREWALL_VERBOSE
```

Enables code to allow logging of packets through `syslogd(8)`. Without this option, even if you specify that packets should be logged in the filter rules, nothing will happen.

```
options IPFIREWALL_VERBOSE_LIMIT=10
```

Limits the number of packets logged through `syslogd(8)` on a per entry basis. You may wish to use this option in hostile environments in which you want to log firewall activity, but do not want to be open to a denial of service attack via `syslog` flooding.

When a chain entry reaches the packet limit specified, logging is turned off for that particular entry. To resume logging, you will need to reset the associated counter using the `ipfw(8)` utility:

```
# ipfw zero 4500
```

Where 4500 is the chain entry you wish to continue logging.

Previous versions of FreeBSD contained an `IPFIREWALL_ACCT` option. This is now obsolete as the firewall code automatically includes accounting facilities.

## Configuring IPFW

The configuration of the **IPFW** software is done through the `ipfw(8)` utility. The syntax for this command looks quite complicated, but it is relatively simple once you understand its structure.

There are currently four different command categories used by the utility: addition/deletion, listing, flushing, and clearing. Addition/deletion is used to build the rules that control how packets are accepted,

rejected, and logged. Listing is used to examine the contents of your rule set (otherwise known as the chain) and packet counters (accounting). Flushing is used to remove all entries from the chain. Clearing is used to zero out one or more accounting entries.

## Altering the IPFW rules

The syntax for this form of the command is:

```
ipfw [-N] command [index] action [log] protocol addresses [options]
```

There is one valid flag when using this form of the command:

-N

Resolve addresses and service names in output.

The *command* given can be shortened to the shortest unique form. The valid *commands* are:

add

Add an entry to the firewall/accounting rule list

delete

Delete an entry from the firewall/accounting rule list

Previous versions of **IPFW** used separate firewall and accounting entries. The present version provides packet accounting with each firewall entry.

If an *index* value is supplied, it used to place the entry at a specific point in the chain. Otherwise, the entry is placed at the end of the chain at an index 100 greater than the last chain entry (this does not include the default policy, rule 65535, deny).

The `log` option causes matching rules to be output to the system console if the kernel was compiled with `IPFIREWALL_VERBOSE`.

Valid *actions* are:

reject

Drop the packet, and send an ICMP host or port unreachable (as appropriate) packet to the source.

allow

Pass the packet on as normal. (aliases: `pass` and `accept`)

deny

Drop the packet. The source is not notified via an ICMP message (thus it appears that the packet never arrived at the destination).

count

Update packet counters but do not allow/deny the packet based on this rule. The search continues with the next chain entry.

Each *action* will be recognized by the shortest unambiguous prefix.

The *protocols* which can be specified are:

all

Matches any IP packet

icmp

Matches ICMP packets

tcp

Matches TCP packets

udp

Matches UDP packets

The *address* specification is:

from *address/mask* [*port*] to *address/mark* [*port*] [*via interface*]

You can only specify *port* in conjunction with *protocols* which support ports (UDP and TCP).

The *via* is optional and may specify the IP address or domain name of a local IP interface, or an interface name (e.g. `ed0`) to match only packets coming through this interface. Interface unit numbers can be specified with an optional wildcard. For example, `ppp*` would match all kernel PPP interfaces.

The syntax used to specify an *address/mask* is:

*address*

or

*address/mask-bits*

or

*address:mask-pattern*

A valid hostname may be specified in place of the IP address. *mask-bits* is a decimal number representing how many bits in the address mask should be set. e.g. specifying 192.216.222.1/24 will create a mask which will allow any address in a class C subnet (in this case, 192.216.222) to be matched. *mask-pattern* is an IP address which will be logically AND'ed with the address given. The keyword *any* may be used to specify "any IP address".

The port numbers to be blocked are specified as:

*port* [,*port* [,*port* [...]]]

to specify either a single port or a list of ports, or

*port-port*

to specify a range of ports. You may also combine a single range with a list, but the range must always be specified first.

The *options* available are:

**frag**

Matches if the packet is not the first fragment of the datagram.

**in**

Matches if the packet is on the way in.

**out**

Matches if the packet is on the way out.

**ipoptions *spec***

Matches if the IP header contains the comma separated list of options specified in *spec*. The supported list of IP options are: *ssrr* (strict source route), *lsrr* (loose source route), *rr* (record packet route), and *ts* (timestamp). The absence of a particular option may be denoted with a leading *!*.

**established**

Matches if the packet is part of an already established TCP connection (i.e. it has the RST or ACK bits set). You can optimize the performance of the firewall by placing *established* rules early in the chain.

**setup**

Matches if the packet is an attempt to establish a TCP connection (the SYN bit set is set but the ACK bit is not).

**tcpflags *flags***

Matches if the TCP header contains the comma separated list of *flags*. The supported flags are *fin*, *syn*, *rst*, *psh*, *ack*, and *urg*. The absence of a particular flag may be indicated by a leading *!*.

**icmptypes *types***

Matches if the ICMP type is present in the list *types*. The list may be specified as any combination of ranges and/or individual types separated by commas. Commonly used ICMP types are: 0 echo reply (ping reply), 3 destination unreachable, 5 redirect, 8 echo request (ping request), and 11 time exceeded (used to indicate TTL expiration as with *traceroute(8)*).

**Listing the IPFW rules**

The syntax for this form of the command is:

```
ipfw [-a] [-t] [-N] l
```

There are three valid flags when using this form of the command:

**-a**

While listing, show counter values. This option is the only way to see accounting counters.

-t

Display the last match times for each chain entry. The time listing is incompatible with the input syntax used by the `ipfw(8)` utility.

-N

Attempt to resolve given addresses and service names.

## Flushing the IPFW rules

The syntax for flushing the chain is:

```
ipfw flush
```

This causes all entries in the firewall chain to be removed except the fixed default policy enforced by the kernel (index 65535). Use caution when flushing rules, the default deny policy will leave your system cut off from the network until allow entries are added to the chain.

## Clearing the IPFW packet counters

The syntax for clearing one or more packet counters is:

```
ipfw zero [index]
```

When used without an *index* argument, all packet counters are cleared. If an *index* is supplied, the clearing operation only affects a specific chain entry.

## Example commands for ipfw

This command will deny all packets from the host `evil.crackers.org` to the telnet port of the host `nice.people.org` by being forwarded by the router:

```
# ipfw add deny tcp from evil.crackers.org to nice.people.org 23
```

The next example denies and logs any TCP traffic from the entire `crackers.org` network (a class C) to the `nice.people.org` machine (any port).

```
# ipfw add deny log tcp from evil.crackers.org/24 to nice.people.org
```

If you do not want people sending X sessions to your internal network (a subnet of a class C), the following command will do the necessary filtering:

```
# ipfw add deny tcp from any to my.org/28 6000 setup
```

To see the accounting records:

```
# ipfw -a list
```

or in the short form

```
# ipfw -a 1
```

You can also see the last time a chain entry was matched with:

```
# ipfw -at 1
```

## Building a packet filtering firewall

**Note:** The following suggestions are just that: suggestions. The requirements of each firewall are different and I cannot tell you how to build a firewall to meet your particular requirements.

When initially setting up your firewall, unless you have a test bench setup where you can configure your firewall host in a controlled environment, I strongly recommend you use the logging version of the commands and enable logging in the kernel. This will allow you to quickly identify problem areas and cure them without too much disruption. Even after the initial setup phase is complete, I recommend using the logging for of 'deny' as it allows tracing of possible attacks and also modification of the firewall rules if your requirements alter.

**Note:** If you use the logging versions of the `accept` command, it can generate *large* amounts of log data as one log line will be generated for every packet that passes through the firewall, so large ftp/http transfers, etc, will really slow the system down. It also increases the latencies on those packets as it requires more work to be done by the kernel before the packet can be passed on. `syslogd` will also start using up a lot more processor time as it logs all the extra data to disk, and it could quite easily fill the partition `/var/log` is located on.

As currently supplied, FreeBSD does not have the ability to load firewall rules at boot time. My suggestion is to put a call to a shell script in the `/etc/netstart` script. Put the call early enough in the

netstart file so that the firewall is configured before any of the IP interfaces are configured. This means that there is no window during which time your network is open.

The actual script used to load the rules is entirely up to you. There is currently no support in the `ipfw` utility for loading multiple rules in the one command. The system I use is to use the command:

```
# ipfw list
```

to write a list of the current rules out to a file, and then use a text editor to prepend `ipfw` before all the lines. This will allow the script to be fed into `/bin/sh` and reload the rules into the kernel. Perhaps not the most efficient way, but it works.

The next problem is what your firewall should actually *do*! This is largely dependent on what access to your network you want to allow from the outside, and how much access to the outside world you want to allow from the inside. Some general rules are:

- Block all incoming access to ports below 1024 for TCP. This is where most of the security sensitive services are, like finger, SMTP (mail) and telnet.
- Block *all* incoming UDP traffic. There are very few useful services that travel over UDP, and what useful traffic there is normally a security threat (e.g. Sun's RPC and NFS protocols). This has its disadvantages also, since UDP is a connectionless protocol, denying incoming UDP traffic also blocks the replies to outgoing UDP traffic. This can cause a problem for people (on the inside) using external archie (prospero) servers. If you want to allow access to archie, you'll have to allow packets coming from ports 191 and 1525 to any internal UDP port through the firewall. ntp is another service you may consider allowing through, which comes from port 123.
- Block traffic to port 6000 from the outside. Port 6000 is the port used for access to X11 servers, and can be a security threat (especially if people are in the habit of doing `xhost +` on their workstations). X11 can actually use a range of ports starting at 6000, the upper limit being how many X displays you can run on the machine. The upper limit as defined by RFC 1700 (Assigned Numbers) is 6063.
- Check what ports any internal servers use (e.g. SQL servers, etc). It is probably a good idea to block those as well, as they normally fall outside the 1-1024 range specified above.

Another checklist for firewall configuration is available from CERT at [ftp://ftp.cert.org/pub/tech\\_tips/packet\\_filtering](ftp://ftp.cert.org/pub/tech_tips/packet_filtering)

As I said above, these are only *guidelines*. You will have to decide what filter rules you want to use on your firewall yourself. I cannot accept ANY responsibility if someone breaks into your network, even if you follow the advice given above.

# Chapter 7. Printing

*Contributed by Sean Kelly <kelly@fs1.noaa.gov> 30 September 1995*

In order to use printers with FreeBSD, you will need to set them up to work with the Berkeley line printer spooling system, also known as the LPD spooling system. It is the standard printer control system in FreeBSD. This section introduces the LPD spooling system, often simply called LPD.

If you are already familiar with LPD or another printer spooling system, you may wish to skip to section [Setting up the spooling system](#).

## What the Spooler Does

LPD controls everything about a host's printers. It is responsible for a number of things:

- It controls access to attached printers and printers attached to other hosts on the network.
- It enables users to submit files to be printed; these submissions are known as *jobs*.
- It prevents multiple users from accessing a printer at the same time by maintaining a *queue* for each printer.
- It can print *header pages* (also known as *banner* or *burst* pages) so users can easily find jobs they have printed in a stack of printouts.
- It takes care of communications parameters for printers connected on serial ports.
- It can send jobs over the network to another LPD spooler on another host.
- It can run special filters to format jobs to be printed for various printer languages or printer capabilities.
- It can account for printer usage.

Through a configuration file, and by providing the special filter programs, you can enable the LPD system to do all or some subset of the above for a great variety of printer hardware.

## Why You Should Use the Spooler

If you are the sole user of your system, you may be wondering why you should bother with the spooler when you do not need access control, header pages, or printer accounting. While it is possible to enable direct access to a printer, you should use the spooler anyway since

- LPD prints jobs in the background; you do not have to wait for data to be copied to the printer.

- LPD can conveniently run a job to be printed through filters to add date/time headers or convert a special file format (such as a TeX DVI file) into a format the printer will understand. You will not have to do these steps manually.
- Many free and commercial programs that provide a print feature usually expect to talk to the spooler on your system. By setting up the spooling system, you will more easily support other software you may later add or already have.

## Setting Up the Spooling System

To use printers with the LPD spooling system, you will need to set up both your printer hardware and the LPD software. This document describes two levels of setup:

- See section Simple Printer Setup to learn how to connect a printer, tell LPD how to communicate with it, and print plain text files to the printer.
- See section Advanced Printer Setup to find out how to print a variety of special file formats, to print header pages, to print across a network, to control access to printers, and to do printer accounting.

## Simple Printer Setup

This section tells how to configure printer hardware and the LPD software to use the printer. It teaches the basics:

- Section Hardware Setup gives some hints on connecting the printer to a port on your computer.
- Section Software Setup shows how to setup the LPD spooler configuration file `/etc/printcap`.

If you are setting up a printer that uses a network protocol to accept data to print instead of a serial or parallel interface, see [Printers With Networked Data Stream Interfaces](#).

Although this section is called “Simple Printer Setup,” it is actually fairly complex. Getting the printer to work with your computer and the LPD spooler is the hardest part. The advanced options like header pages and accounting are fairly easy once you get the printer working.

## Hardware Setup

This section tells about the various ways you can connect a printer to your PC. It talks about the kinds of ports and cables, and also the kernel configuration you may need to enable FreeBSD to speak to the

printer.

If you have already connected your printer and have successfully printed with it under another operating system, you can probably skip to section Software Setup.

## Ports and Cables

Nearly all printers you can get for a PC today support one or both of the following interfaces:

- *Serial* interfaces use a serial port on your computer to send data to the printer. Serial interfaces are common in the computer industry and cables are readily available and also easy to construct. Serial interfaces sometimes need special cables and might require you to configure somewhat complex communications options.
- *Parallel* interfaces use a parallel port on your computer to send data to the printer. Parallel interfaces are common in the PC market. Cables are readily available but more difficult to construct by hand. There are usually no communications options with parallel interfaces, making their configuration exceedingly simple.

Parallel interfaces are sometimes known as “Centronics” interfaces, named after the connector type on the printer.

In general, serial interfaces are slower than parallel interfaces. Parallel interfaces usually offer just one-way communication (computer to printer) while serial gives you two-way. Many newer parallel ports can also receive data from the printer, but only few printers need to send data back to the computer. And FreeBSD does not support two-way parallel communication yet.

Usually, the only time you need two-way communication with the printer is if the printer speaks PostScript. PostScript printers can be very verbose. In fact, PostScript jobs are actually programs sent to the printer; they need not produce paper at all and may return results directly to the computer. PostScript also uses two-way communication to tell the computer about problems, such as errors in the PostScript program or paper jams. Your users may be appreciative of such information. Furthermore, the best way to do effective accounting with a PostScript printer requires two-way communication: you ask the printer for its page count (how many pages it has printed in its lifetime), then send the user’s job, then ask again for its page count. Subtract the two values and you know how much paper to charge the user.

So, which interface should you use?

- If you need two-way communication, use a serial port. FreeBSD does not yet support two-way communication over a parallel port.
- If you do not need two-way communication and can pick parallel or serial, prefer the parallel interface. It keeps a serial port free for other peripherals—such as a terminal or a modem—and is faster most of the time. It is also easier to configure.

- Finally, use whatever works.

## Parallel Ports

To hook up a printer using a parallel interface, connect the Centronics cable between the printer and the computer. The instructions that came with the printer, the computer, or both should give you complete guidance.

Remember which parallel port you used on the computer. The first parallel port is `/dev/lpt0` to FreeBSD; the second is `/dev/lpt1`, and so on.

## Serial Ports

To hook up a printer using a serial interface, connect the proper serial cable between the printer and the computer. The instructions that came with the printer, the computer, or both should give you complete guidance.

If you are unsure what the “proper serial cable” is, you may wish to try one of the following alternatives:

- A *modem* cable connects each pin of the connector on one end of the cable straight through to its corresponding pin of the connector on the other end. This type of cable is also known as a “DTE-to-DCE” cable.
- A *null-modem* cable connects some pins straight through, swaps others (send data to receive data, for example), and shorts some internally in each connector hood. This type of cable is also known as a “DTE-to-DTE” cable.
- A *serial printer* cable, required for some unusual printers, is like the null modem cable, but sends some signals to their counterparts instead of being internally shorted.

You should also set up the communications parameters for the printer, usually through front-panel controls or DIP switches on the printer. Choose the highest bps (bits per second, sometimes *baud rate*) rate that both your computer and the printer can support. Choose 7 or 8 data bits; none, even, or odd parity; and 1 or 2 stop bits. Also choose a flow control protocol: either none, or XON/XOFF (also known as “in-band” or “software”) flow control. Remember these settings for the software configuration that follows.

## Software Setup

This section describes the software setup necessary to print with the LPD spooling system in FreeBSD.

Here is an outline of the steps involved:

1. Configure your kernel, if necessary, for the port you are using for the printer; section Kernel Configuration tells you what you need to do.
2. Set the communications mode for the parallel port, if you are using a parallel port; section Setting the Communication Mode for the Parallel Port gives details.
3. Test if the operating system can send data to the printer. Section Checking Printer Communications gives some suggestions on how to do this.
4. Set up LPD for the printer by modifying the file `/etc/printcap`. Section The `/etc/printcap` File shows you how.

## Kernel Configuration

The operating system kernel is compiled to work with a specific set of devices. The serial or parallel interface for your printer is a part of that set. Therefore, it might be necessary to add support for an additional serial or parallel port if your kernel is not already configured for one.

To find out if the kernel you are currently using supports a serial interface, type:

```
# dmesg | grep sioN
```

Where *N* is the number of the serial port, starting from zero. If you see output similar to the following:

```
sio2 at 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
```

then the kernel supports the port.

To find out if the kernel supports a parallel interface, type:

```
# dmesg | grep lptN
```

Where *N* is the number of the parallel port, starting from zero. If you see output similar to the following

```
lpt0 at 0x378-0x37f on isa
```

then the kernel supports the port.

You might have to reconfigure your kernel in order for the operating system to recognize and use the parallel or serial port you are using for the printer.

To add support for a serial port, see the section on kernel configuration. To add support for a parallel port, see that section *and* the section that follows.

### Adding `/dev` Entries for the Ports

Even though the kernel may support communication along a serial or parallel port, you will still need a software interface through which programs running on the system can send and receive data. That is what entries in the `/dev` directory are for.

*To add a `/dev` entry for a port:*

1. Become root with the `su(1)` command. Enter the root password when prompted.
2. Change to the `/dev` directory:

```
# cd /dev
```

3. Type:

```
# ./MAKEDEV
  port
```

Where *port* is the device entry for the port you want to make. Use `lpt0` for the first parallel port, `lpt1` for the second, and so on; use `ttyd0` for the first serial port, `ttyd1` for the second, and so on.

4. Type:

```
# ls -l
  port
```

to make sure the device entry got created.

### Setting the Communication Mode for the Parallel Port

When you are using the parallel interface, you can choose whether FreeBSD should use interrupt-driven or polled communication with the printer.

- The *interrupt-driven* method is the default with the GENERIC kernel. With this method, the operating system uses an IRQ line to determine when the printer is ready for data.
- The *polled* method directs the operating system to repeatedly ask the printer if it is ready for more data. When it responds ready, the kernel sends more data.

The interrupt-driven method is somewhat faster but uses up a precious IRQ line. You should use whichever one works.

You can set the communications mode in two ways: by configuring the kernel or by using the `lptcontrol(8)` program.

*To set the communications mode by configuring the kernel:*

1. Edit your kernel configuration file. Look for or add an `lpt0` entry. If you are setting up the second parallel port, use `lpt1` instead. Use `lpt2` for the third port, and so on.
  - If you want interrupt-driven mode, add the `irq` specifier:
 

```
device lpt0 at isa? port? tty irq N vector lptintr
```

 Where *N* is the IRQ number for your computer's parallel port.
  - If you want polled mode, do not add the `irq` specifier:
 

```
device lpt0 at isa? port? tty vector lptintr
```
2. Save the file. Then configure, build, and install the kernel, then reboot. See kernel configuration for more details.

To set the communications mode with `lptcontrol(8)`:

1. Type:
 

```
# lptcontrol -i -u N
```

 to set interrupt-driven mode for `lptN`.
2. Type:
 

```
# lptcontrol -p -u N
```

 to set polled-mode for `lptN`.

You could put these commands in your `/etc/rc.local` file to set the mode each time your system boots. See `lptcontrol(8)` for more information.

### Checking Printer Communications

Before proceeding to configure the spooling system, you should make sure the operating system can successfully send data to your printer. It is a lot easier to debug printer communication and the spooling system separately.

To test the printer, we will send some text to it. For printers that can immediately print characters sent to them, the program `lptest(1)` is perfect: it generates all 96 printable ASCII characters in 96 lines.

For a PostScript (or other language-based) printer, we will need a more sophisticated test. A small PostScript program, such as the following, will suffice:

```
%!PS
100 100 moveto 300 300 lineto stroke
310 310 moveto /Helvetica findfont 12 scalefont setfont
```

```
(Is this thing working?) show
showpage
```

**Note:** When this document refers to a printer language, I am assuming a language like PostScript, and not Hewlett Packard's PCL. Although PCL has great functionality, you can intermingle plain text with its escape sequences. PostScript cannot directly print plain text, and that is the kind of printer language for which we must make special accommodations.

### Checking a Parallel Printer

This section tells you how to check if FreeBSD can communicate with a printer connected to a parallel port.

*To test a printer on a parallel port:*

1. Become root with `su(1)`.
2. Send data to the printer.
  - If the printer can print plain text, then use `lptest(1)`. Type:

```
# lptest > /dev/lptN
```

Where *N* is the number of the parallel port, starting from zero.

- If the printer understands PostScript or other printer language, then send a small program to the printer. Type:

```
# cat > /dev/lptN
```

Then, line by line, type the program *carefully* as you cannot edit a line once you have pressed RETURN or ENTER. When you have finished entering the program, press CONTROL+D, or whatever your end of file key is.

Alternatively, you can put the program in a file and type:

```
# cat file > /dev/lptN
```

Where *file* is the name of the file containing the program you want to send to the printer.

You should see something print. Do not worry if the text does not look right; we will fix such things later.

### Checking a Serial Printer

This section tells you how to check if FreeBSD can communicate with a printer on a serial port.

To test a printer on a serial port:

1. Become root with `su(1)`.
2. Edit the file `/etc/remote`. Add the following entry:

```
printer:dv=/dev/port:br#bps-rate:pa=parity
```

Where *port* is the device entry for the serial port (`tttyd0`, `tttyd1`, etc.), *bps-rate* is the bits-per-second rate at which the printer communicates, and *parity* is the parity required by the printer (either *even*, *odd*, *none*, or *zero*).

Here is a sample entry for a printer connected via a serial line to the third serial port at 19200 bps with no parity:

```
printer:dv=/dev/ttyd2:br#19200:pa=none
```

3. Connect to the printer with `tip(1)`. Type:

```
# tip printer
```

If this step does not work, edit the file `/etc/remote` again and try using `/dev/cuaaN` instead of `/dev/ttydN`.

4. Send data to the printer.

- If the printer can print plain text, then use `lptest(1)`. Type:

```
~$lptest
```

- If the printer understands PostScript or other printer language, then send a small program to the printer. Type the program, line by line, *very carefully* as backspacing or other editing keys may be significant to the printer. You may also need to type a special end-of-file key for the printer so it knows it received the whole program. For PostScript printers, press `CONTROL+D`.

Alternatively, you can put the program in a file and type:

```
~>file
```

Where *file* is the name of the file containing the program. After `tip(1)` sends the file, press any required end-of-file key.

You should see something print. Do not worry if the text does not look right; we will fix that later.

## Enabling the Spooler: The `/etc/printcap` File

At this point, your printer should be hooked up, your kernel configured to communicate with it (if necessary), and you have been able to send some simple data to the printer. Now, we are ready to configure LPD to control access to your printer.

You configure LPD by editing the file `/etc/printcap`. The LPD spooling system reads this file each time the spooler is used, so updates to the file take immediate effect.

The format of the `printcap(5)` file is straightforward. Use your favorite text editor to make changes to `/etc/printcap`. The format is identical to other capability files like `/usr/share/misc/termcap` and `/etc/remote`. For complete information about the format, see the `cgetent(3)`.

The simple spooler configuration consists of the following steps:

1. Pick a name (and a few convenient aliases) for the printer, and put them in the `/etc/printcap` file; see Naming the Printer.
2. Turn off header pages (which are on by default) by inserting the `sh` capability; see Suppressing Header Pages.
3. Make a spooling directory, and specify its location with the `sd` capability; see Making the Spooling Directory.
4. Set the `/dev` entry to use for the printer, and note it in `/etc/printcap` with the `lp` capability; see Identifying the Printer Device. Also, if the printer is on a serial port, set up the communication parameters with the `fs`, `fc`, `xs`, and `xc` capabilities; see Configuring Spooler Communications Parameters.
5. Install a plain text input filter; see Installing the Text Filter
6. Test the setup by printing something with the `lpr(1)` command; see Trying It Out and Troubleshooting.

**Note:** Language-based printers, such as PostScript printers, cannot directly print plain text. The simple setup outlined above and described in the following sections assumes that if you are installing such a printer you will print only files that the printer can understand.

Users often expect that they can print plain text to any of the printers installed on your system. Programs that interface to LPD to do their printing usually make the same assumption. If you are installing such a printer and want to be able to print jobs in the printer language *and* print plain text jobs, you are strongly urged to add an additional step to the simple setup outlined above: install an automatic plain-text-to-PostScript (or other printer language) conversion program. Section Accommodating Plain Text Jobs on PostScript Printers tells how to do this.

## Naming the Printer

The first (easy) step is to pick a name for your printer. It really does not matter whether you choose functional or whimsical names since you can also provide a number aliases for the printer.

At least one of the printers specified in the `/etc/printcap` should have the alias `lp`. This is the default printer's name. If users do not have the `PRINTER` environment variable nor specify a printer name on the command line of any of the LPD commands, then `lp` will be the default printer they get to use.

Also, it is common practice to make the last alias for a printer be a full description of the printer, including make and model.

Once you have picked a name and some common aliases, put them in the `/etc/printcap` file. The name of the printer should start in the leftmost column. Separate each alias with a vertical bar and put a colon after the last alias.

In the following example, we start with a skeletal `/etc/printcap` that defines two printers (a Diablo 630 line printer and a Panasonic KX-P4455 PostScript laser printer):

```
#
# /etc/printcap for host rose
#
rattan|line|diablo|lp|Diablo 630 Line Printer:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:
```

In this example, the first printer is named `rattan` and has as aliases `line`, `diablo`, `lp`, and `Diablo 630 Line Printer`. Since it has the alias `lp`, it is also the default printer. The second is named `bamboo`, and has as aliases `ps`, `PS`, `S`, `panasonic`, and `Panasonic KX-P4455 PostScript v51.4`.

## Suppressing Header Pages

The LPD spooling system will by default print a *header page* for each job. The header page contains the user name who requested the job, the host from which the job came, and the name of the job, in nice large letters. Unfortunately, all this extra text gets in the way of debugging the simple printer setup, so we will suppress header pages.

To suppress header pages, add the `sh` capability to the entry for the printer in `/etc/printcap`. Here is the example `/etc/printcap` with `sh` added:

```
#
# /etc/printcap for host rose - no header pages anywhere
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
      :sh:
```

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
:sh:
```

Note how we used the correct format: the first line starts in the leftmost column, and subsequent lines are indented with a single TAB. Every line in an entry except the last ends in a backslash character.

### Making the Spooling Directory

The next step in the simple spooler setup is to make a *spooling directory*, a directory where print jobs reside until they are printed, and where a number of other spooler support files live.

Because of the variable nature of spooling directories, it is customary to put these directories under `/var/spool`. It is not necessary to backup the contents of spooling directories, either. Recreating them is as simple as running `mkdir(1)`.

It is also customary to make the directory with a name that is identical to the name of the printer, as shown below:

```
# mkdir /var/spool/printer-name
```

However, if you have a lot of printers on your network, you might want to put the spooling directories under a single directory that you reserve just for printing with LPD. We will do this for our two example printers `rattan` and `bamboo`:

```
# mkdir /var/spool/lpd
# mkdir /var/spool/lpd/rattan
# mkdir /var/spool/lpd/bamboo
```

**Note:** If you are concerned about the privacy of jobs that users print, you might want to protect the spooling directory so it is not publicly accessible. Spooling directories should be owned and be readable, writable, and searchable by user `daemon` and group `daemon`, and no one else. We will do this for our example printers:

```
# chown daemon.daemon /var/spool/lpd/rattan
# chown daemon.daemon /var/spool/lpd/bamboo
# chmod 770 /var/spool/lpd/rattan
# chmod 770 /var/spool/lpd/bamboo
```

Finally, you need to tell LPD about these directories using the `/etc/printcap` file. You specify the pathname of the spooling directory with the `sd` capability:

```
#
# /etc/printcap for host rose - added spooling directories
#
```

```
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:
```

Note that the name of the printer starts in the first column but all other entries describing the printer should be indented with a tab and each line escaped with a backslash.

If you do not specify a spooling directory with `sd`, the spooling system will use `/var/spool/lpd` as a default.

### Identifying the Printer Device

In section Adding `/dev` Entries for the Ports, we identified which entry in the `/dev` directory FreeBSD will use to communicate with the printer. Now, we tell LPD that information. When the spooling system has a job to print, it will open the specified device on behalf of the filter program (which is responsible for passing data to the printer).

List the `/dev` entry pathname in the `/etc/printcap` file using the `lp` capability.

In our running example, let us assume that `rattan` is on the first parallel port, and `bamboo` is on a sixth serial port; here are the additions to `/etc/printcap`:

```
#
# /etc/printcap for host rose - identified what devices to use
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:\
    :lp=/dev/ttyd5:
```

If you do not specify the `lp` capability for a printer in your `/etc/printcap` file, LPD uses `/dev/lp` as a default. `/dev/lp` currently does not exist in FreeBSD.

If the printer you are installing is connected to a parallel port, skip to the section Installing the Text Filter. Otherwise, be sure to follow the instructions in the next section.

## Configuring Spooler Communication Parameters

For printers on serial ports, LPD can set up the bps rate, parity, and other serial communication parameters on behalf of the filter program that sends data to the printer. This is advantageous since:

- It lets you try different communication parameters by simply editing the `/etc/printcap` file; you do not have to recompile the filter program.
- It enables the spooling system to use the same filter program for multiple printers which may have different serial communication settings.

The following `/etc/printcap` capabilities control serial communication parameters of the device listed in the `lp` capability:

`br#bps-rate`

Sets the communications speed of the device to `bps-rate`, where `bps-rate` can be 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, or 38400 bits-per-second.

`fc#clear-bits`

Clears the flag bits `clear-bits` in the `sgttyb` structure after opening the device.

`fs#set-bits`

Sets the flag bits `set-bits` in the `sgttyb` structure.

`xc#clear-bits`

Clears local mode bits `clear-bits` after opening the device.

`xs#set-bits`

Sets local mode bits `set-bits`.

For more information on the bits for the `fc`, `fs`, `xc`, and `xs` capabilities, see the file `/usr/include/sys/ioctl_compat.h`.

When LPD opens the device specified by the `lp` capability, it reads the flag bits in the `sgttyb` structure; it clears any bits in the `fc` capability, then sets bits in the `fs` capability, then applies the resultant setting. It does the same for the local mode bits as well.

Let us add to our example printer on the sixth serial port. We will set the bps rate to 38400. For the flag bits, we will set the TANDEM, ANYP, LITOUT, FLUSHO, and PASS8 flags. For the local mode bits, we will set the LITOUT and PASS8 flags:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
      :sh:sd=/var/spool/lpd/bamboo:\
      :lp=/dev/ttyd5:fs#0x82000c1:xs#0x820:
```

## Installing the Text Filter

We are now ready to tell LPD what text filter to use to send jobs to the printer. A *text filter*, also known as an *input filter*, is a program that LPD runs when it has a job to print. When LPD runs the text filter for a printer, it sets the filter's standard input to the job to print, and its standard output to the printer device specified with the `lp` capability. The filter is expected to read the job from standard input, perform any necessary translation for the printer, and write the results to standard output, which will get printed. For more information on the text filter, see section Filters.

For our simple printer setup, the text filter can be a small shell script that just executes `/bin/cat` to send the job to the printer. FreeBSD comes with another filter called `lpf` that handles backspacing and underlining for printers that might not deal with such character streams well. And, of course, you can use any other filter program you want. The filter `lpf` is described in detail in section `lpf: a Text Filter`.

First, let us make the shell script `/usr/local/libexec/if-simple` be a simple text filter. Put the following text into that file with your favorite text editor:

```
#!/bin/sh
#
# if-simple - Simple text input filter for lpd
# Installed in /usr/local/libexec/if-simple
#
# Simply copies stdin to stdout. Ignores all filter arguments.

/bin/cat && exit 0
exit 2
```

Make the file executable:

```
# chmod 555 /usr/local/libexec/if-simple
```

And then tell LPD to use it by specifying it with the `if` capability in `/etc/printcap`. We will add it to the two printers we have so far in the example `/etc/printcap`:

```
#
# /etc/printcap for host rose - added text filter
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
      :sh:sd=/var/spool/lpd/rattan:\ :lp=/dev/lpt0:\
      :if=/usr/local/libexec/if-simple:
```

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
      :sh:sd=/var/spool/lpd/bamboo:\
      :lp=/dev/ttyd5:fs#0x82000e1:xs#0x820:\
      :if=/usr/local/libexec/if-simple:
```

### Trying It Out

You have reached the end of the simple LPD setup. Unfortunately, congratulations are not quite yet in order, since we still have to test the setup and correct any problems. To test the setup, try printing something. To print with the LPD system, you use the command `lpr(1)`, which submits a job for printing.

You can combine `lpr(1)` with the `lptest(1)` program, introduced in section Checking Printer Communications to generate some test text.

*To test the simple LPD setup:*

Type:

```
# lptest 20 5 | lpr -Pprinter-name
```

Where *printer-name* is the name of a printer (or an alias) specified in `/etc/printcap`. To test the default printer, type `lpr(1)` without any `-P` argument. Again, if you are testing a printer that expects PostScript, send a PostScript program in that language instead of using `lptest(1)`. You can do so by putting the program in a file and typing `lpr file`.

For a PostScript printer, you should get the results of the program. If you are using `lptest(1)`, then your results should look like the following:

```
! "$%&'()*+,-./01234
"# $%&'()*+,-./012345
# $%&'()*+,-./0123456
$%&'()*+,-./01234567
%&'()*+,-./012345678
```

To further test the printer, try downloading larger programs (for language-based printers) or running `lptest(1)` with different arguments. For example, `lptest 80 60` will produce 60 lines of 80 characters each.

If the printer did not work, see the next section, Troubleshooting.

## Troubleshooting

After performing the simple test with `lptest(1)`, you might have gotten one of the following results instead of the correct printout:

It worked, after awhile; or, it did not eject a full sheet.

The printer printed the above, but it sat for awhile and did nothing. In fact, you might have needed to press a PRINT REMAINING or FORM FEED button on the printer to get any results to appear.

If this is the case, the printer was probably waiting to see if there was any more data for your job before it printed anything. To fix this problem, you can have the text filter send a FORM FEED character (or whatever is necessary) to the printer. This is usually sufficient to have the printer immediately print any text remaining in its internal buffer. It is also useful to make sure each print job ends on a full sheet, so the next job does not start somewhere on the middle of the last page of the previous job.

The following replacement for the shell script `/usr/local/libexec/if-simple` prints a form feed after it sends the job to the printer:

```
#!/bin/sh
#
# if-simple - Simple text input filter for lpd
# Installed in /usr/local/libexec/if-simple
#
# Simply copies stdin to stdout. Ignores all filter arguments.
# Writes a form feed character (\f) after printing job.

/bin/cat && printf "\f" && exit 0
exit 2
```

It produced the “staircase effect.”

You got the following on paper:

```
! "#$%&'()*+,-./01234
  "#$%&'()*+,-./012345
                                "#$%&'()*+,-./0123456
```

You have become another victim of the *staircase effect*, caused by conflicting interpretations of what characters should indicate a new-line. UNIX-style operating systems use a single character: ASCII code 10, the line feed (LF). MS-DOS, OS/2, and others uses a pair of characters, ASCII code 10 *and* ASCII code 13 (the carriage return or CR). Many printers use the MS-DOS convention for representing new-lines.

When you print with FreeBSD, your text used just the line feed character. The printer, upon seeing a line feed character, advanced the paper one line, but maintained the same horizontal position on the page for the next character to print. That is what the carriage return is for: to move the location of the next character to print to the left edge of the paper.

Here is what FreeBSD wants your printer to do:

Printer received CR	Printer prints CR
Printer received LF	Printer prints CR + LF

Here are some ways to achieve this:

- Use the printer's configuration switches or control panel to alter its interpretation of these characters. Check your printer's manual to find out how to do this.

**Note:** If you boot your system into other operating systems besides FreeBSD, you may have to *reconfigure* the printer to use a an interpretation for CR and LF characters that those other operating systems use. You might prefer one of the other solutions, below.

- Have FreeBSD's serial line driver automatically convert LF to CR+LF. Of course, this works with printers on serial ports *only*. To enable this feature, set the CRMOD bit in `fs` capability in the `/etc/printcap` file for the printer.
- Send an *escape code* to the printer to have it temporarily treat LF characters differently. Consult your printer's manual for escape codes that your printer might support. When you find the proper escape code, modify the text filter to send the code first, then send the print job.

Here is an example text filter for printers that understand the Hewlett-Packard PCL escape codes. This filter makes the printer treat LF characters as a LF and CR; then it sends the job; then it sends a form feed to eject the last page of the job. It should work with nearly all Hewlett Packard printers.

```
#!/bin/sh
#
# hpif - Simple text input filter for lpd for HP-PCL based printers
# Installed in /usr/local/libexec/hpif
#
# Simply copies stdin to stdout. Ignores all filter arguments.
# Tells printer to treat LF as CR+LF. Writes a form feed character
# after printing job.

printf "\033&k2G" && cat && printf "\033&l0H" && exit 0
exit 2
```

Here is an example `/etc/printcap` from a host called `orchid`. It has a single printer attached to its first parallel port, a Hewlett Packard LaserJet 3Si named `teak`. It is using the above script as its text filter:

```
#
# /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/hpif:
```

It overprinted each line.

The printer never advanced a line. All of the lines of text were printed on top of each other on one line.

This problem is the “opposite” of the staircase effect, described above, and is much rarer. Somewhere, the LF characters that FreeBSD uses to end a line are being treated as CR characters to return the print location to the left edge of the paper, but not also down a line.

Use the printer’s configuration switches or control panel to enforce the following interpretation of LF and CR characters:

Printer receives	Printer prints
CR	CR
LF	CR + LF

The printer lost characters.

While printing, the printer did not print a few characters in each line. The problem might have gotten worse as the printer ran, losing more and more characters.

The problem is that the printer cannot keep up with the speed at which the computer sends data over a serial line. (This problem should not occur with printers on parallel ports.) There are two ways to overcome the problem:

- If the printer supports XON/XOFF flow control, have FreeBSD use it by specifying the TANDEM bit in the `fs` capability.
- If the printer supports carrier flow control, specify the MDMBUF bit in the `fs` capability. Make sure the cable connecting the printer to the computer is correctly wired for carrier flow control.
- If the printer does not support any flow control, use some combination of the NLDELAY,

TBDELAY, CRDELAY, VTDELAY, and BSDELAY bits in the `fs` capability to add appropriate delays to the stream of data sent to the printer.

It printed garbage.

The printer printed what appeared to be random garbage, but not the desired text.

This is usually another symptom of incorrect communications parameters with a serial printer. Double-check the `bps` rate in the `br` capability, and the parity bits in the `fs` and `fc` capabilities; make sure the printer is using the same settings as specified in the `/etc/printcap` file.

Nothing happened.

If nothing happened, the problem is probably within FreeBSD and not the hardware. Add the log file (`lf`) capability to the entry for the printer you are debugging in the `/etc/printcap` file. For example, here is the entry for `rattan`, with the `lf` capability:

```
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:\
    :lf=/var/log/rattan.log
```

Then, try printing again. Check the log file (in our example, `/var/log/rattan.log`) to see any error messages that might appear. Based on the messages you see, try to correct the problem.

If you do not specify a `lf` capability, LPD uses `/dev/console` as a default.

## Using Printers

This section tells you how to use printers you have setup with FreeBSD. Here is an overview of the user-level commands:

`lpr(1)`

Print jobs

`lpq(1)`

Check printer queues

`lprm(1)`

Remove jobs from a printer's queue

There is also an administrative command, `lpc(8)`, described in the section *Administrating the LPD Spooler*, used to control printers and their queues.

All three of the commands `lpr(1)`, `lprm(1)`, and `lpq(1)` accept an option `-P printer-name` to specify on which printer/queue to operate, as listed in the `/etc/printcap` file. This enables you to submit, remove, and check on jobs for various printers. If you do not use the `-P` option, then these commands use the printer specified in the `PRINTER` environment variable. Finally, if you do not have a `PRINTER` environment variable, these commands default to the printer named `lp`.

Hereafter, the terminology *default printer* means the printer named in the `PRINTER` environment variable, or the printer named `lp` when there is no `PRINTER` environment variable.

## Printing Jobs

To print files, type:

```
% lpr filename ...
```

This prints each of the listed files to the default printer. If you list no files, `lpr(1)` reads data to print from standard input. For example, this command prints some important system files:

```
% lpr /etc/host.conf /etc/hosts.equiv
```

To select a specific printer, type:

```
% lpr -P printer-name filename ...
```

This example prints a long listing of the current directory to the printer named `rattan`:

```
% ls -l | lpr -P rattan
```

Because no files were listed for the `lpr(1)` command, `lpr` read the data to print from standard input, which was the output of the `ls -l` command.

The `lpr(1)` command can also accept a wide variety of options to control formatting, apply file conversions, generate multiple copies, and so forth. For more information, see the section *Printing Options*.

## Checking Jobs

When you print with `lpr(1)`, the data you wish to print is put together in a package called a “print job”, which is sent to the LPD spooling system. Each printer has a queue of jobs, and your job waits in that queue along with other jobs from yourself and from other users. The printer prints those jobs in a first-come, first-served order.

To display the queue for the default printer, type `lpq(1)`. For a specific printer, use the `-P` option. For example, the command

```
% lpq -P bamboo
```

shows the queue for the printer named `bamboo`. Here is an example of the output of the `lpq` command:

```
bamboo is ready and printing
Rank  Owner   Job  Files                                Total Size
active kelly   9    /etc/host.conf, /etc/hosts.equiv    88 bytes
2nd   kelly   10   (standard input)                    1635 bytes
3rd   mary    11   ...                                  78519 bytes
```

This shows three jobs in the queue for `bamboo`. The first job, submitted by user `kelly`, got assigned “job number” 9. Every job for a printer gets a unique job number. Most of the time you can ignore the job number, but you will need it if you want to cancel the job; see section [Removing Jobs](#) for details.

Job number nine consists of two files; multiple files given on the `lpr(1)` command line are treated as part of a single job. It is the currently active job (note the word `active` under the “Rank” column), which means the printer should be currently printing that job. The second job consists of data passed as the standard input to the

`lpr(1)` command. The third job came from user `mary`; it is a much larger job. The pathname of the files she’s trying to print is too long to fit, so the `lpq(1)` command just shows three dots.

The very first line of the output from `lpq(1)` is also useful: it tells what the printer is currently doing (or at least what LPD thinks the printer is doing).

The `lpq(1)` command also support a `-l` option to generate a detailed long listing. Here is an example of `lpq -l`:

```
waiting for bamboo to become ready (offline ?)
kelly: 1st [job 009rose]
        /etc/host.conf                73 bytes
        /etc/hosts.equiv              15 bytes

kelly: 2nd [job 010rose]
        (standard input)              1635 bytes
```

```
mary: 3rd [job 011rose]
/home/orchid/mary/research/venus/alpha-regio/mapping 78519 bytes
```

## Removing Jobs

If you change your mind about printing a job, you can remove the job from the queue with the `lprm(1)` command. Often, you can even use `lprm(1)` to remove an active job, but some or all of the job might still get printed.

To remove a job from the default printer, first use `lpq(1)` to find the job number. Then type:

```
% lprm job-number
```

To remove the job from a specific printer, add the `-P` option. The following command removes job number 10 from the queue for the printer `bamboo`:

```
% lprm -P bamboo 10
```

The `lprm(1)` command has a few shortcuts:

`lprm -`

Removes all jobs (for the default printer) belonging to you.

`lprm user`

Removes all jobs (for the default printer) belonging to `user`. The superuser can remove other users' jobs; you can remove only your own jobs.

`lprm`

With no job number, user name, or `-` appearing on the command line, `lprm(1)` removes the currently active job on the default printer, if it belongs to you. The superuser can remove any active job.

Just use the `-P` option with the above shortcuts to operate on a specific printer instead of the default. For example, the following command removes all jobs for the current user in the queue for the printer named `rattan`:

```
% lprm -P rattan -
```

**Note:** If you are working in a networked environment, `lprm(1)` will let you remove jobs only from the host from which the jobs were submitted, even if the same printer is available from other hosts. The following command sequence demonstrates this:

```
% lpr -P rattan myfile
% rlogin orchid
% lpq -P rattan
Rank  Owner  Job  Files                Total Size
active seeyan  12  ...                49123 bytes
2nd   kelly   13  myfile                12 bytes
% lprm -P rattan 13
rose: Permission denied
% logout
% lprm -P rattan 13
dfA013rose dequeued
cfA013rose dequeued
```

## Beyond Plain Text: Printing Options

The `lpr(1)` command supports a number of options that control formatting text, converting graphic and other file formats, producing multiple copies, handling of the job, and more. This section describes the options.

### Formatting and Conversion Options

The following `lpr(1)` options control formatting of the files in the job. Use these options if the job does not contain plain text or if you want plain text formatted through the `pr(1)` utility.

For example, the following command prints a DVI file (from the TeX typesetting system) named `fish-report.dvi` to the printer named `bamboo`:

```
% lpr -P bamboo -d fish-report.dvi
```

These options apply to every file in the job, so you cannot mix (say) DVI and ditroff files together in a job. Instead, submit the files as separate jobs, using a different conversion option for each job.

**Note:** All of these options except `-p` and `-T` require conversion filters installed for the destination printer. For example, the `-d` option requires the DVI conversion filter. Section Conversion Filters gives details.

-c

Print cifplot files.

-d

Print DVI files.

-f

Print FORTRAN text files.

-g

Print plot data.

-i *number*

Indent the output by *number* columns; if you omit *number*, indent by 8 columns. This option works only with certain conversion filters.

**Note:** Do not put any space between the -i and the number.

-l

Print literal text data, including control characters.

-n

Print ditroff (device independent troff) data.

-p

Format plain text with pr(1) before printing. See pr(1) for more information.

-T *title*

Use *title* on the pr(1) header instead of the file name. This option has effect only when used with the -p option.

-t

Print troff data.

-v

Print raster data.

Here is an example: this command prints a nicely formatted version of the `ls(1)` manual page on the default printer:

```
% zcat /usr/share/man/man1/ls.1.gz | troff -t -man | lpr -t
```

The `zcat(1)` command uncompresses the source of the

`ls(1)` manual page and passes it to the `troff(1)` command, which formats that source and makes GNU `troff` output and passes it to `lpr(1)`, which submits the job to the LPD spooler. Because we used the `-t` option to

`lpr(1)`, the spooler will convert the GNU `troff` output into a format the default printer can understand when it prints the job.

## Job Handling Options

The following options to `lpr(1)` tell LPD to handle the job specially:

*-#copies*

Produce a number of *copies* of each file in the job instead of just one copy. An administrator may disable this option to reduce printer wear-and-tear and encourage photocopier usage. See section [Restricting Multiple Copies](#).

This example prints three copies of `parser.c` followed by three copies of `parser.h` to the default printer:

```
% lpr -#3 parser.c parser.h
```

*-m*

Send mail after completing the print job. With this option, the LPD system will send mail to your account when it finishes handling your job. In its message, it will tell you if the job completed successfully or if there was an error, and (often) what the error was.

*-s*

Do not copy the files to the spooling directory, but make symbolic links to them instead.

If you are printing a large job, you probably want to use this option. It saves space in the spooling directory (your job might overflow the free space on the filesystem where the spooling directory

resides). It saves time as well since LPD will not have to copy each and every byte of your job to the spooling directory.

There is a drawback, though: since LPD will refer to the original files directly, you cannot modify or remove them until they have been printed.

**Note:** If you are printing to a remote printer, LPD will eventually have to copy files from the local host to the remote host, so the `-s` option will save space only on the local spooling directory, not the remote. It is still useful, though.

`-r`

Remove the files in the job after copying them to the spooling directory, or after printing them with the `-s` option. Be careful with this option!

## Header Page Options

These options to `lpr(1)` adjust the text that normally appears on a job's header page. If header pages are suppressed for the destination printer, these options have no effect. See section Header Pages for information about setting up header pages.

`-C text`

Replace the hostname on the header page with *text*. The hostname is normally the name of the host from which the job was submitted.

`-J text`

Replace the job name on the header page with *text*. The job name is normally the name of the first file of the job, or `stdin` if you are printing standard input.

`-h`

Do not print any header page.

**Note:** At some sites, this option may have no effect due to the way header pages are generated. See Header Pages for details.

## Administrating Printers

As an administrator for your printers, you have had to install, set up, and test them. Using the `lpc(8)` command, you can interact with your printers in yet more ways. With `lpc(8)`, you can

- Start and stop the printers
- Enable and disable their queues
- Rearrange the order of the jobs in each queue.

First, a note about terminology: if a printer is *stopped*, it will not print anything in its queue. Users can still submit jobs, which will wait in the queue until the printer is *started* or the queue is cleared.

If a queue is *disabled*, no user (except root) can submit jobs for the printer. An *enabled* queue allows jobs to be submitted. A printer can be *started* for a disabled queue, in which case it will continue to print jobs in the queue until the queue is empty.

In general, you have to have root privileges to use the `lpc(8)` command. Ordinary users can use the `lpc(8)` command to get printer status and to restart a hung printer only.

Here is a summary of the `lpc(8)` commands. Most of the commands takes a *printer-name* argument to tell on which printer to operate. You can use `all` for the *printer-name* to mean all printers listed in `/etc/printcap`.

`abort printer-name`

Cancel the current job and stop the printer. Users can still submit jobs if the queue's enabled.

`clean printer-name`

Remove old files from the printer's spooling directory. Occasionally, the files that make up a job are not properly removed by LPD, particularly if there have been errors during printing or a lot of administrative activity. This command finds files that do not belong in the spooling directory and removes them.

`disable printer-name`

Disable queuing of new jobs. If the printer's started, it will continue to print any jobs remaining in the queue. The superuser (root) can always submit jobs, even to a disabled queue.

This command is useful while you are testing a new printer or filter installation: disable the queue and submit jobs as root. Other users will not be able to submit jobs until you complete your testing and re-enable the queue with the `enable` command.

`down printer-name message`

Take a printer down. Equivalent to `disable` followed by `stop`. The *message* appears as the printer's status whenever a user checks the printer's queue with `lpq(1)` or `status` with `lpc status`.

`enable printer-name`

Enable the queue for a printer. Users can submit jobs but the printer will not print anything until it is started.

`help command-name`

Print help on the command *command-name*. With no *command-name*, print a summary of the commands available.

`restart printer-name`

Start the printer. Ordinary users can use this command if some extraordinary circumstance hangs LPD, but they cannot start a printer stopped with either the `stop` or `down` commands. The `restart` command is equivalent to `abort` followed by `start`.

`start printer-name`

Start the printer. The printer will print jobs in its queue.

`stop printer-name`

Stop the printer. The printer will finish the current job and will not print anything else in its queue. Even though the printer is stopped, users can still submit jobs to an enabled queue.

`topq printer-name job-or-username`

Rearrange the queue for *printer-name* by placing the jobs with the listed *job* numbers or the jobs belonging to *username* at the top of the queue. For this command, you cannot use `all` as the *printer-name*.

`up printer-name`

Bring a printer up; the opposite of the `down` command. Equivalent to `start` followed by `enable`.

`lpc(8)` accepts the above commands on the command line. If you do not enter any commands, `lpc(8)` enters an interactive mode, where you can enter commands until you type `exit`, `quit`, or end-of-file.

## Advanced Printer Setup

This section describes filters for printing specially formatted files, header pages, printing across networks, and restricting and accounting for printer usage.

### Filters

Although LPD handles network protocols, queuing, access control, and other aspects of printing, most of the *real* work happens in the *filters*. Filters are programs that communicate with the printer and handle its device dependencies and special requirements. In the simple printer setup, we installed a plain text filter—an extremely simple one that should work with most printers (section Installing the Text Filter).

However, in order to take advantage of format conversion, printer accounting, specific printer quirks, and so on, you should understand how filters work. It will ultimately be the filter’s responsibility to handle these aspects. And the bad news is that most of the time *you* have to provide filters yourself. The good news is that many are generally available; when they are not, they are usually easy to write.

Also, FreeBSD comes with one, `/usr/libexec/lpr/lpf`, that works with many printers that can print plain text. (It handles backspacing and tabs in the file, and does accounting, but that is about all it does.) There are also several filters and filter components in the FreeBSD ports collection.

Here is what you will find in this section:

- Section How Filters Work, tries to give an overview of a filter’s role in the printing process. You should read this section to get an understanding of what is happening “under the hood” when LPD uses filters. This knowledge could help you anticipate and debug problems you might encounter as you install more and more filters on each of your printers.
- LPD expects every printer to be able to print plain text by default. This presents a problem for PostScript (or other language-based printers) which cannot directly print plain text. Section Accommodating Plain Text Jobs on PostScript Printers tells you what you should do to overcome this problem. I recommend reading this section if you have a PostScript printer.
- PostScript is a popular output format for many programs. Even some people (myself included) write PostScript code directly. But PostScript printers are expensive. Section Simulating PostScript on Non-PostScript Printers tells how you can further modify a printer’s text filter to accept and print PostScript data on a *non-PostScript* printer. I recommend reading this section if you do not have a PostScript printer.
- Section Conversion Filters tells about a way you can automate the conversion of specific file formats, such as graphic or typesetting data, into formats your printer can understand. After reading this section, you should be able to set up your printers such that users can type `lpr -t` to print troff data,

or `lpr -d` to print TeX DVI data, or `lpr -v` to print raster image data, and so forth. I recommend reading this section.

- Section Output Filters tells all about a not often used feature of LPD: output filters. Unless you are printing header pages (see Header Pages), you can probably skip that section altogether.
- Section `lpf`: a Text Filter describes `lpf`, a fairly complete if simple text filter for line printers (and laser printers that act like line printers) that comes with FreeBSD. If you need a quick way to get printer accounting working for plain text, or if you have a printer which emits smoke when it sees backspace characters, you should definitely consider `lpf`.

## How Filters Work

As mentioned before, a filter is an executable program started by LPD to handle the device-dependent part of communicating with the printer.

When LPD wants to print a file in a job, it starts a filter program. It sets the filter's standard input to the file to print, its standard output to the printer, and its standard error to the error logging file (specified in the `lf` capability in `/etc/printcap`, or `/dev/console` by default).

Which filter LPD starts and the filter's arguments depend on what is listed in the `/etc/printcap` file and what arguments the user specified for the job on the `lpr(1)` command line. For example, if the user typed `lpr -t`, LPD would start the `troff` filter, listed in the `tf` capability for the destination printer. If the user wanted to print plain text, it would start the `if` filter (this is mostly true: see Output Filters for details).

There are three kinds of filters you can specify in `/etc/printcap`:

- The *text filter*, confusingly called the *input filter* in LPD documentation, handles regular text printing. Think of it as the default filter. LPD expects every printer to be able to print plain text by default, and it is the text filter's job to make sure backspaces, tabs, or other special characters do not confuse the printer. If you are in an environment where you have to account for printer usage, the text filter must also account for pages printed, usually by counting the number of lines printed and comparing that to the number of lines per page the printer supports. The text filter is started with the following argument list:

```
filter-name [-c] -wwidth -llength -iindent -n login -h host acct-file
```

where

`-c`

appears if the job's submitted with `lpr -l`

*width*

is the value from the `pw` (page width) capability specified in `/etc/printcap`, default 132

*length*

is the value from the `pl` (page length) capability, default 66

*indent*

is the amount of the indentation from `lpr -i`, default 0

*login*

is the account name of the user printing the file

*host*

is the host name from which the job was submitted

*acct-file*

is the name of the accounting file from the `af` capability.

- A *conversion filter* converts a specific file format into one the printer can render onto paper. For example, ditroff typesetting data cannot be directly printed, but you can install a conversion filter for ditroff files to convert the ditroff data into a form the printer can digest and print. Section Conversion Filters tells all about them. Conversion filters also need to do accounting, if you need printer accounting. Conversion filters are started with the following arguments:

```
filter-name -xpixel-width -ypixel-height -n login -h host acct-file
```

where *pixel-width* is the value from the `px` capability (default 0) and *pixel-height* is the value from the `py` capability (default 0).

- The *output filter* is used only if there is no text filter, or if header pages are enabled. In my experience, output filters are rarely used. Section Output Filters describe them. There are only two arguments to an output filter:

```
filter-name -wwidth -llength
```

which are identical to the text filters `-w` and `-l` arguments.

Filters should also *exit* with the following exit status:

exit 0

If the filter printed the file successfully.

exit 1

If the filter failed to print the file but wants LPD to try to print the file again. LPD will restart a filter if it exits with this status.

exit 2

If the filter failed to print the file and does not want LPD to try again. LPD will throw out the file.

The text filter that comes with the FreeBSD release, `/usr/libexec/lpr/lpf`, takes advantage of the page width and length arguments to determine when to send a form feed and how to account for printer usage. It uses the login, host, and accounting file arguments to make the accounting entries.

If you are shopping for filters, see if they are LPD-compatible. If they are, they must support the argument lists described above. If you plan on writing filters for general use, then have them support the same argument lists and exit codes.

## Accommodating Plain Text Jobs on PostScript Printers

If you are the only user of your computer and PostScript (or other language-based) printer, and you promise to never send plain text to your printer and to never use features of various programs that will want to send plain text to your printer, then you do not need to worry about this section at all.

But, if you would like to send both PostScript and plain text jobs to the printer, then you are urged to augment your printer setup. To do so, we have the text filter detect if the arriving job is plain text or PostScript. All PostScript jobs must start with `%!` (for other printer languages, see your printer documentation). If those are the first two characters in the job, we have PostScript, and can pass the rest of the job directly. If those are not the first two characters in the file, then the filter will convert the text into PostScript and print the result.

How do we do this?

If you have got a serial printer, a great way to do it is to install `lprps`. `lprps` is a PostScript printer filter which performs two-way communication with the printer. It updates the printer's status file with verbose information from the printer, so users and administrators can see exactly what the state of the printer is (such as toner low or paper jam). But more importantly, it includes a program called `psif` which detects whether the incoming job is plain text and calls `textps` (another program that comes with `lprps`) to convert it to PostScript. It then uses `lprps` to send the job to the printer.

`lprps` is part of the FreeBSD ports collection (see The Ports Collection). You can fetch, build and install it yourself, of course. After installing `lprps`, just specify the pathname to the `psif` program that is part

of `lprps`. If you installed `lprps` from the ports collection, use the following in the serial PostScript printer's entry in `/etc/printcap`:

```
:if=/usr/local/libexec/psif:
```

You should also specify the `rw` capability; that tells LPD to open the printer in read-write mode.

If you have a parallel PostScript printer (and therefore cannot use two-way communication with the printer, which `lprps` needs), you can use the following shell script as the text filter:

```
#!/bin/sh
#
# psif - Print PostScript or plain text on a PostScript printer
# Script version; NOT the version that comes with lprps
# Installed in /usr/local/libexec/psif
#

read first_line
first_two_chars=`expr "$first_line" : '\(..\)`

if [ "$first_two_chars" = "%!" ]; then
    #
    # PostScript job, print it.
    #
    echo "$first_line" && cat && printf "\004" && exit 0
    exit 2
else
    #
    # Plain text, convert it, then print it.
    #
    ( echo "$first_line"; cat ) | /usr/local/bin/textps && printf "\004" && exit 0
    exit 2
fi
```

In the above script, `textps` is a program we installed separately to convert plain text to PostScript. You can use any text-to-PostScript program you wish. The FreeBSD ports collection (see The Ports Collection) includes a full featured text-to-PostScript program called `a2ps` that you might want to investigate.

## Simulating PostScript on Non-PostScript Printers

PostScript is the *de facto* standard for high quality typesetting and printing. PostScript is, however, an *expensive* standard. Thankfully, Alladin Enterprises has a free PostScript work-alike called **Ghostscript** that runs with FreeBSD. Ghostscript can read most PostScript files and can render their pages onto a

variety of devices, including many brands of non-PostScript printers. By installing Ghostscript and using a special text filter for your printer, you can make your non-PostScript printer act like a real PostScript printer.

Ghostscript should be in the FreeBSD ports collection, if you would like to install it from there. You can fetch, build, and install it quite easily yourself, as well.

To simulate PostScript, we have the text filter detect if it is printing a PostScript file. If it is not, then the filter will pass the file directly to the printer; otherwise, it will use Ghostscript to first convert the file into a format the printer will understand.

Here is an example: the following script is a text filter for Hewlett Packard DeskJet 500 printers. For other printers, substitute the `-sDEVICE` argument to the `gs` (Ghostscript) command. (Type `gs -h` to get a list of devices the current installation of Ghostscript supports.)

```
#!/bin/sh
#
# ifhp - Print Ghostscript-simulated PostScript on a DeskJet 500
# Installed in /usr/local/libexec/hpif
#
# Treat LF as CR+LF:
#
printf "\033&k2G" || exit 2
#
# Read first two characters of the file
#
read first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

if [ "$first_two_chars" = "%!" ]; then
#
# It is PostScript; use Ghostscript to scan-convert and print it
#
/usr/local/bin/gs -dSAFER -dNOPAUSE -q -sDEVICE=djet500 -sOutputFile=- -
\
    && exit 0
else
#
# Plain text or HP/PCL, so just print it directly; print a form
# at the end to eject the last page.
#
echo "$first_line" && cat && printf "\f" && exit 0
fi
```

```
exit 2
```

Finally, you need to notify LPD of the filter via the `if` capability:

```
:if=/usr/local/libexec/hpif:
```

That is it. You can type `lpr plain.text` and `lpr whatever.ps` and both should print successfully.

## Conversion Filters

After completing the simple setup described in Simple Printer Setup, the first thing you will probably want to do is install conversion filters for your favorite file formats (besides plain ASCII text).

### Why Install Conversion Filters?

Conversion filters make printing various kinds of files easy. As an example, suppose we do a lot of work with the TeX typesetting system, and we have a PostScript printer. Every time we generate a DVI file from TeX, we cannot print it directly until we convert the DVI file into PostScript. The command sequence goes like this:

```
% dvips seaweed-analysis.dvi
% lpr seaweed-analysis.ps
```

By installing a conversion filter for DVI files, we can skip the hand conversion step each time by having LPD do it for us. Now, each time we get a DVI file, we are just one step away from printing it:

```
% lpr -d seaweed-analysis.dvi
```

We got LPD to do the DVI file conversion for us by specifying the `-d` option. Section Formatting and Conversion Options lists the conversion options.

For each of the conversion options you want a printer to support, install a *conversion filter* and specify its pathname in `/etc/printcap`. A conversion filter is like the text filter for the simple printer setup (see section Installing the Text Filter) except that instead of printing plain text, the filter converts the file into a format the printer can understand.

### Which Conversions Filters Should I Install?

You should install the conversion filters you expect to use. If you print a lot of DVI data, then a DVI conversion filter is in order. If you have got plenty of troff to print out, then you probably want a troff filter.

The following table summarizes the filters that LPD works with, their capability entries for the `/etc/printcap` file, and how to invoke them with the `lpr` command:

File type	<code>/etc/printcap</code> capability	<code>lpr</code> option
cifplot	<code>cf</code>	<code>-c</code>
DVI	<code>df</code>	<code>-d</code>
plot	<code>gf</code>	<code>-g</code>
ditroff	<code>nf</code>	<code>-n</code>
FORTTRAN text	<code>rf</code>	<code>-f</code>
troff	<code>rf</code>	<code>-f</code>
raster	<code>vf</code>	<code>-v</code>
plain text	<code>if</code>	none, <code>-p</code> , or <code>-l</code>

In our example, using `lpr -d` means the printer needs a `df` capability in its entry in `/etc/printcap`.

Despite what others might contend, formats like FORTRAN text and plot are probably obsolete. At your site, you can give new meanings to these or any of the formatting options just by installing custom filters. For example, suppose you would like to directly print Printerleaf files (files from the Interleaf desktop publishing program), but will never print plot files. You could install a Printerleaf conversion filter under the `gf` capability and then educate your users that `lpr -g` mean “print Printerleaf files.”

### Installing Conversion Filters

Since conversion filters are programs you install outside of the base FreeBSD installation, they should probably go under `/usr/local`. The directory `/usr/local/libexec` is a popular location, since they are specialized programs that only LPD will run; regular users should not ever need to run them.

To enable a conversion filter, specify its pathname under the appropriate capability for the destination printer in `/etc/printcap`.

In our example, we will add the DVI conversion filter to the entry for the printer named `bamboo`. Here is the example `/etc/printcap` file again, with the new `df` capability for the printer `bamboo`.

```
#
# /etc/printcap for host rose - added df filter for bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:
```

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
      :sh:sd=/var/spool/lpd/bamboo:\
      :lp=/dev/ttyd5:fs#0x82000e1:xs#0x820:rw:\
      :if=/usr/local/libexec/psif:\
      :df=/usr/local/libexec/psdf:
```

The DVI filter is a shell script named `/usr/local/libexec/psdf`. Here is that script:

```
#!/bin/sh
#
# psdf - DVI to PostScript printer filter
# Installed in /usr/local/libexec/psdf
#
# Invoked by lpd when user runs lpr -d
#
exec /usr/local/bin/dvips -f | /usr/local/libexec/lprps "$@"
```

This script runs `dvips` in filter mode (the `-f` argument) on standard input, which is the job to print. It then starts the PostScript printer filter `lprps` (see section Accommodating Plain Text Jobs on PostScript Printers) with the arguments `LPD` passed to this script. `lprps` will use those arguments to account for the pages printed.

### More Conversion Filter Examples

Since there is no fixed set of steps to install conversion filters, let me instead provide more examples. Use these as guidance to making your own filters. Use them directly, if appropriate.

This example script is a raster (well, GIF file, actually) conversion filter for a Hewlett Packard LaserJet III-Si printer:

```
#!/bin/sh
#
# hpvf - Convert GIF files into HP/PCL, then print
# Installed in /usr/local/libexec/hpvf

PATH=/usr/X11R6/bin:$PATH; export PATH giftopnm | ppmtopgm | pgmtopbm | pbm-
tolj -resolution 300 \
    && exit 0 \
    || exit 2
```

It works by converting the GIF file into a portable anymap, converting that into a portable graymap, converting that into a portable bitmap, and converting that into LaserJet/PCL-compatible data.

Here is the `/etc/printcap` file with an entry for a printer using the above filter:

```
#
# /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
      :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
      :if=/usr/local/libexec/hpif:\
      :vf=/usr/local/libexec/hpvf:
```

The following script is a conversion filter for troff data from the groff typesetting system for the PostScript printer named bamboo:

```
#!/bin/sh
#
# pstf - Convert groff's troff data into PS, then print.
# Installed in /usr/local/libexec/pstf
#
exec grops | /usr/local/libexec/lprps "$@"
```

The above script makes use of `lprps` again to handle the communication with the printer. If the printer were on a parallel port, we would use this script instead:

```
#!/bin/sh
#
# pstf - Convert groff's troff data into PS, then print.
# Installed in /usr/local/libexec/pstf
#
exec grops
```

That is it. Here is the entry we need to add to `/etc/printcap` to enable the filter:

```
:tf=/usr/local/libexec/pstf:
```

Here is an example that might make old hands at FORTRAN blush. It is a FORTRAN-text filter for any printer that can directly print plain text. We will install it for the printer `teak`:

```
#!/bin/sh
#
# hprf - FORTRAN text filter for LaserJet 3si:
# Installed in /usr/local/libexec/hprf
#

printf "\033&k2G" && fpr && printf "\f" && exit 0
exit 2
```

And we will add this line to the `/etc/printcap` for the printer `teak` to enable this filter:

```
:rf=/usr/local/libexec/hprf:
```

Here is one final, somewhat complex example. We will add a DVI filter to the LaserJet printer `teak` introduced earlier. First, the easy part: updating `/etc/printcap` with the location of the DVI filter:

```
:df=/usr/local/libexec/hpdf:
```

Now, for the hard part: making the filter. For that, we need a DVI-to-LaserJet/PCL conversion program. The FreeBSD ports collection (see [The Ports Collection](#)) has one: `dvi2xx` is the name of the package. Installing this package gives us the program we need, `dvilj2p`, which converts DVI into LaserJet IIp, LaserJet III, and LaserJet 2000 compatible codes.

`dvilj2p` makes the filter `hpdf` quite complex since `dvilj2p` cannot read from standard input. It wants to work with a filename. What is worse, the filename has to end in `.dvi` so using `/dev/fd/0` for standard input is problematic. We can get around that problem by linking (symbolically) a temporary file name (one that ends in `.dvi`) to `/dev/fd/0`, thereby forcing `dvilj2p` to read from standard input.

The only other fly in the ointment is the fact that we cannot use `/tmp` for the temporary link. Symbolic links are owned by user and group `bin`. The filter runs as user `daemon`. And the `/tmp` directory has the sticky bit set. The filter can create the link, but it will not be able clean up when done and remove it since the link will belong to a different user.

Instead, the filter will make the symbolic link in the current working directory, which is the spooling directory (specified by the `sd` capability in `/etc/printcap`). This is a perfect place for filters to do their work, especially since there is (sometimes) more free disk space in the spooling directory than under `/tmp`.

Here, finally, is the filter:

```
#!/bin/sh
#
# hpdf - Print DVI data on HP/PCL printer
# Installed in /usr/local/libexec/hpdf

PATH=/usr/local/bin:$PATH; export PATH

#
# Define a function to clean up our temporary files. These exist
# in the current directory, which will be the spooling directory
# for the printer.
#
cleanup() {
    rm -f hpdf$$$.dvi
}

#
```

```

# Define a function to handle fatal errors: print the given message
# and exit 2. Exiting with 2 tells LPD to do not try to reprint the
# job.
#
fatal() {
    echo "$@" 1>&2
    cleanup
    exit 2
}

#
# If user removes the job, LPD will send SIGINT, so trap SIGINT
# (and a few other signals) to clean up after ourselves.
#
trap cleanup 1 2 15

#
# Make sure we are not colliding with any existing files.
#
cleanup

#
# Link the DVI input file to standard input (the file to print).
#
ln -s /dev/fd/0 hpdf$$ .dvi || fatal "Cannot symlink /dev/fd/0"

#
# Make LF = CR+LF
#
printf "\033&k2G" || fatal "Cannot initialize printer"

#
# Convert and print. Return value from dvi2ps does not seem to be
# reliable, so we ignore it.
#
dvi2ps -M1 -q -e- dfhp$$ .dvi

#
# Clean up and exit
#
cleanup
exit 0

```

### Automated Conversion: An Alternative To Conversion Filters

All these conversion filters accomplish a lot for your printing environment, but at the cost forcing the user to specify (on the `lpr(1)` command line) which one to use. If your users are not particularly computer literate, having to specify a filter option will become annoying. What is worse, though, is that an incorrectly specified filter option may run a filter on the wrong type of file and cause your printer to spew out hundreds of sheets of paper.

Rather than install conversion filters at all, you might want to try having the text filter (since it is the default filter) detect the type of file it has been asked to print and then automatically run the right conversion filter. Tools such as `file` can be of help here. Of course, it will be hard to determine the differences between *some* file types—and, of course, you can still provide conversion filters just for them.

The FreeBSD ports collection has a text filter that performs automatic conversion called `apsfilter`. It can detect plain text, PostScript, and DVI files, run the proper conversions, and print.

### Output Filters

The LPD spooling system supports one other type of filter that we have not yet explored: an output filter. An output filter is intended for printing plain text only, like the text filter, but with many simplifications. If you are using an output filter but no text filter, then:

- LPD starts an output filter once for the entire job instead of once for each file in the job.
- LPD does not make any provision to identify the start or the end of files within the job for the output filter.
- LPD does not pass the user's login or host to the filter, so it is not intended to do accounting. In fact, it gets only two arguments:

```
filter-name -width -length
```

Where *width* is from the `pw` capability and *length* is from the `p1` capability for the printer in question.

Do not be seduced by an output filter's simplicity. If you would like each file in a job to start on a different page an output filter *will not work*. Use a text filter (also known as an input filter); see section Installing the Text Filter. Furthermore, an output filter is actually *more complex* in that it has to examine the byte stream being sent to it for special flag characters and must send signals to itself on behalf of LPD.

However, an output filter is *necessary* if you want header pages and need to send escape sequences or other initialization strings to be able to print the header page. (But it is also *futile* if you want to charge

header pages to the requesting user's account, since LPD does not give any user or host information to the output filter.)

On a single printer, LPD allows both an output filter and text or other filters. In such cases, LPD will start the output filter to print the header page (see section Header Pages) only. LPD then expects the output filter to *stop itself* by sending two bytes to the filter: ASCII 031 followed by ASCII 001. When an output filter sees these two bytes (031, 001), it should stop by sending SIGSTOP to itself. When LPD's done running other filters, it will restart the output filter by sending SIGCONT to it.

If there is an output filter but *no* text filter and LPD is working on a plain text job, LPD uses the output filter to do the job. As stated before, the output filter will print each file of the job in sequence with no intervening form feeds or other paper advancement, and this is probably *not* what you want. In almost all cases, you need a text filter.

The program `lpf`, which we introduced earlier as a text filter, can also run as an output filter. If you need a quick-and-dirty output filter but do not want to write the byte detection and signal sending code, try `lpf`. You can also wrap `lpf` in a shell script to handle any initialization codes the printer might require.

### **lpf: a Text Filter**

The program `/usr/libexec/lpr/lpf` that comes with FreeBSD binary distribution is a text filter (input filter) that can indent output (job submitted with `lpr -i`), allow literal characters to pass (job submitted with `lpr -l`), adjust the printing position for backspaces and tabs in the job, and account for pages printed. It can also act like an output filter.

`lpf` is suitable for many printing environments. And although it has no capability to send initialization sequences to a printer, it is easy to write a shell script to do the needed initialization and then execute `lpf`.

In order for `lpf` to do page accounting correctly, it needs correct values filled in for the `pw` and `p1` capabilities in the `/etc/printcap` file. It uses these values to determine how much text can fit on a page and how many pages were in a user's job. For more information on printer accounting, see [Accounting for Printer Usage](#).

## **Header Pages**

If you have *lots* of users, all of them using various printers, then you probably want to consider *header pages* as a necessary evil.

Header pages, also known as *banner* or *burst pages* identify to whom jobs belong after they are printed. They are usually printed in large, bold letters, perhaps with decorative borders, so that in a stack of printouts they stand out from the real documents that comprise users' jobs. They enable users to locate

their jobs quickly. The obvious drawback to a header page is that it is yet one more sheet that has to be printed for every job, their ephemeral usefulness lasting not more than a few minutes, ultimately finding themselves in a recycling bin or rubbish heap. (Note that header pages go with each job, not each file in a job, so the paper waste might not be that bad.)

The LPD system can provide header pages automatically for your printouts *if* your printer can directly print plain text. If you have a PostScript printer, you will need an external program to generate the header page; see Header Pages on PostScript Printers.

## Enabling Header Pages

In the Simple Printer Setup, we turned off header pages by specifying `sh` (meaning “suppress header”) in the `/etc/printcap` file. To enable header pages for a printer, just remove the `sh` capability.

Sounds too easy, right?

You are right. You *might* have to provide an output filter to send initialization strings to the printer. Here is an example output filter for Hewlett Packard PCL-compatible printers:

```
#!/bin/sh
#
# hpof - Output filter for Hewlett Packard PCL-compatible printers
# Installed in /usr/local/libexec/hpof

printf "\033&k2G" || exit 2 exec
/usr/libexec/lpr/lpf
```

Specify the path to the output filter in the `of` capability. See Output Filters for more information.

Here is an example `/etc/printcap` file for the printer `teak` that we introduced earlier; we enabled header pages and added the above output filter:

```
#
# /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/hpif:\
    :vf=/usr/local/libexec/hpvf:\
    :of=/usr/local/libexec/hpof:
```

Now, when users print jobs to `teak`, they get a header page with each job. If users want to spend time searching for their printouts, they can suppress header pages by submitting the job with `lpr -h`; see Header Page Options for more `lpr(1)` options.

**Note:** LPD prints a form feed character after the header page. If your printer uses a different character or sequence of characters to eject a page, specify them with the `ff` capability in `/etc/printcap`.

## Controlling Header Pages

By enabling header pages, LPD will produce a *long header*, a full page of large letters identifying the user, host, and job. Here is an example (kelly printed the job named `outline` from host `rose`):

```

k          ll      ll
k          l       l
k          l       l
k  k      eeee    l   l   y   y
k  k      e  e    l   l   y   y
k  k      eeeee   l   l   y   y
kk  k     e       l   l   y   y
k  k     e  e    l   l   y  yy
k  k     eeee    ll    ll   yyy y
                    Y
                    Y
                    Y
                    Y
                    Y

```

```

                    ll
                    t       l       i
                    t       l
oooo  u  u  ttttt  l   ii  n nnn  eeee
o  o  u  u  t      l   i  nn  n  e  e
o  o  u  u  t      l   i  n  n  e
o  o  u  uu  t  t   l   i  n  n  e  e
oooo  uuu u  tt    ll    iii  n  n  eeee

```

```

r rrr  oooo  ssss  eeee
rr  r  o  o  s  s  e  e

```

```

r      o   o   ss      eeeee
r      o   o       ss    e
r      o   o   s   s   e   e
r      oooo   ssss   eeee

```

```

Job:  outline
Date: Sun Sep 17 11:04:58 1995

```

LPD appends a form feed after this text so the job starts on a new page (unless you have `sf` (suppress form feeds) in the destination printer's entry in `/etc/printcap`).

If you prefer, LPD can make a *short header*; specify `sb` (short banner) in the `/etc/printcap` file. The header page will look like this:

```

rose:kelly Job:  outline Date: Sun Sep 17 11:07:51 1995

```

Also by default, LPD prints the header page first, then the job. To reverse that, specify `hl` (header last) in `/etc/printcap`.

## Accounting for Header Pages

Using LPD's built-in header pages enforces a particular paradigm when it comes to printer accounting: header pages must be *free of charge*.

Why?

Because the output filter is the only external program that will have control when the header page is printed that could do accounting, and it is not provided with any *user or host* information or an accounting file, so it has no idea whom to charge for printer use. It is also not enough to just "add one page" to the text filter or any of the conversion filters (which do have user and host information) since users can suppress header pages with `lpr -h`. They could still be charged for header pages they did not print. Basically, `lpr -h` will be the preferred option of environmentally-minded users, but you cannot offer any incentive to use it.

It is *still not enough* to have each of the filters generate their own header pages (thereby being able to charge for them). If users wanted the option of suppressing the header pages with `lpr -h`, they will still get them and be charged for them since LPD does not pass any knowledge of the `-h` option to any of the filters.

So, what are your options?

You can:

- Accept LPD's paradigm and make header pages free.
- Install an alternative to LPD, such as LPDng or PLP. Section Alternatives to the Standard Spooler tells more about other spooling software you can substitute for LPD.
- Write a *smart* output filter. Normally, an output filter is not meant to do anything more than initialize a printer or do some simple character conversion. It is suited for header pages and plain text jobs (when there is no text (input) filter). But, if there is a text filter for the plain text jobs, then LPD will start the output filter only for the header pages. And the output filter can parse the header page text that LPD generates to determine what user and host to charge for the header page. The only other problem with this method is that the output filter still does not know what accounting file to use (it is not passed the name of the file from the `af` capability), but if you have a well-known accounting file, you can hard-code that into the output filter. To facilitate the parsing step, use the `sh` (short header) capability in `/etc/printcap`. Then again, all that might be too much trouble, and users will certainly appreciate the more generous system administrator who makes header pages free.

## Header Pages on PostScript Printers

As described above, LPD can generate a plain text header page suitable for many printers. Of course, PostScript cannot directly print plain text, so the header page feature of LPD is useless—or mostly so.

One obvious way to get header pages is to have every conversion filter and the text filter generate the header page. The filters should use the user and host arguments to generate a suitable header page. The drawback of this method is that users will always get a header page, even if they submit jobs with `lpr -h`.

Let us explore this method. The following script takes three arguments (user login name, host name, and job name) and makes a simple PostScript header page:

```
#!/bin/sh
#
# make-ps-header - make a PostScript header page on stdout
# Installed in /usr/local/libexec/make-ps-header
#
#
# These are PostScript units (72 to the inch).  Modify for A4 or
# whatever size paper you are using:
#
page_width=612
```

```

page_height=792
border=72

#
# Check arguments
#
if [ $# -ne 3 ]; then
    echo "Usage: `basename $0` <user> <host> <job>" 1>&2
    exit 1
fi

#
# Save these, mostly for readability in the PostScript, below.
#
user=$1
host=$2
job=$3
date=`date`

#
# Send the PostScript code to stdout.
#
exec cat <<EOF
%!PS

%
% Make sure we do not interfere with user's job that will follow
%
save

%
% Make a thick, unpleasant border around the edge of the paper.
%
$border $border moveto
$page_width $border 2 mul sub 0 rlineto
0 $page_height $border 2 mul sub rlineto
currentscreen 3 -1 roll pop 100 3 1 roll setscreen
$border 2 mul $page_width sub 0 rlineto closepath
0.8 setgray 10 setlinewidth stroke 0 setgray

%
% Display user's login name, nice and large and prominent
%
/Helvetica-Bold findfont 64 scalefont setfont
$page_width ($user) stringwidth pop sub 2 div $page_height 200 sub moveto

```

```

($user) show

%
% Now show the boring particulars
%
/Helvetica findfont 14 scalefont setfont
/y 200 def
[ (Job:) (Host:) (Date:) ] {
200 y moveto show /y y 18 sub def }
forall

/Helvetica-Bold findfont 14 scalefont setfont
/y 200 def
[ ($job) ($host) ($date) ] {
    270 y moveto show /y y 18 sub def
} forall

%
% That is it
%
restore
showpage
EOF

```

Now, each of the conversion filters and the text filter can call this script to first generate the header page, and then print the user's job. Here is the DVI conversion filter from earlier in this document, modified to make a header page:

```

#!/bin/sh
#
# psdf - DVI to PostScript printer filter
# Installed in /usr/local/libexec/psdf
#
# Invoked by lpd when user runs lpr -d
#

orig_args="$@"

fail() {
    echo "$@" 1>&2
    exit 2
}

while getopts "x:y:n:h:" option; do
    case $option in

```

```

        x|y) ;; # Ignore
        n)  login=$OPTARG ;;
        h)  host=$OPTARG ;;
        *)  echo "LPD started `basename $0` wrong." 1>&2
            exit 2
            ;;
    esac
done

[ "$login" ] || fail "No login name"
[ "$host" ] || fail "No host name"

( /usr/local/libexec/make-ps-header $login $host "DVI File"
  /usr/local/bin/dvips -f ) | eval /usr/local/libexec/lprps $orig_args

```

Notice how the filter has to parse the argument list in order to determine the user and host name. The parsing for the other conversion filters is identical. The text filter takes a slightly different set of arguments, though (see section How Filters Work).

As we have mentioned before, the above scheme, though fairly simple, disables the “suppress header page” option (the `-h` option) to `lpr`. If users wanted to save a tree (or a few pennies, if you charge for header pages), they would not be able to do so, since every filter’s going to print a header page with every job.

To allow users to shut off header pages on a per-job basis, you will need to use the trick introduced in section Accounting for Header Pages: write an output filter that parses the LPD-generated header page and produces a PostScript version. If the user submits the job with `lpr -h`, then LPD will not generate a header page, and neither will your output filter. Otherwise, your output filter will read the text from LPD and send the appropriate header page PostScript code to the printer.

If you have a PostScript printer on a serial line, you can make use of `lprps`, which comes with an output filter, `psof`, which does the above. Note that `psof` does not charge for header pages.

## Networked Printing

FreeBSD supports networked printing: sending jobs to remote printers. Networked printing generally refers to two different things:

- Accessing a printer attached to a remote host. You install a printer that has a conventional serial or parallel interface on one host. Then, you set up LPD to enable access to the printer from other hosts on the network. Section Printers Installed on Remote Hosts tells how to do this.

- Accessing a printer attached directly to a network. The printer has a network interface in addition (or in place of) a more conventional serial or parallel interface. Such a printer might work as follows:
  - It might understand the LPD protocol and can even queue jobs from remote hosts. In this case, it acts just like a regular host running LPD. Follow the same procedure in section Printers Installed on Remote Hosts to set up such a printer.
  - It might support a data stream network connection. In this case, you “attach” the printer to one host on the network by making that host responsible for spooling jobs and sending them to the printer. Section Printers with Networked Data Stream Interfaces gives some suggestions on installing such printers.

## Printers Installed on Remote Hosts

The LPD spooling system has built-in support for sending jobs to other hosts also running LPD (or are compatible with LPD). This feature enables you to install a printer on one host and make it accessible from other hosts. It also works with printers that have network interfaces that understand the LPD protocol.

To enable this kind of remote printing, first install a printer on one host, the *printer host*, using the simple printer setup described in Simple Printer Setup. Do any advanced setup in Advanced Printer Setup that you need. Make sure to test the printer and see if it works with the features of LPD you have enabled. Also ensure that the *local host* has authorization to use the LPD service in the *remote host* (see Restricting Jobs from Remote Printers).

If you are using a printer with a network interface that is compatible with LPD, then the *printer host* in the discussion below is the printer itself, and the *printer name* is the name you configured for the printer. See the documentation that accompanied your printer and/or printer-network interface.

Then, on the other hosts you want to have access to the printer, make an entry in their `/etc/printcap` files with the following:

1. Name the entry anything you want. For simplicity, though, you probably want to use the same name and aliases as on the printer host.
2. Leave the `lp` capability blank, explicitly (`:lp=:`).
3. Make a spooling directory and specify its location in the `sd` capability. LPD will store jobs here before they get sent to the printer host.
4. Place the name of the printer host in the `rm` capability.
5. Place the printer name on the *printer host* in the `rp` capability.

That is it. You do not need to list conversion filters, page dimensions, or anything else in the `/etc/printcap` file.

Here is an example. The host `rose` has two printers, `bamboo` and `rattan`. We will enable users on the host `orchid` to print to those printers. Here is the `/etc/printcap` file for `orchid` (back from section Enabling Header Pages). It already had the entry for the printer `teak`; we have added entries for the two printers on the host `rose`:

```
#
# /etc/printcap for host orchid - added (remote) printers on rose
#

#
# teak is local; it is connected directly to orchid:
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
      :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
      :if=/usr/local/libexec/ifhp:\
      :vf=/usr/local/libexec/vfhp:\
      :of=/usr/local/libexec/ofhp:

#
# rattan is connected to rose; send jobs for rattan to rose:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
      :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

#
# bamboo is connected to rose as well:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
      :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:
```

Then, we just need to make spooling directories on `orchid`:

```
# mkdir -p /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chmod 770 /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chown daemon.daemon /var/spool/lpd/rattan /var/spool/lpd/bamboo
```

Now, users on `orchid` can print to `rattan` and `bamboo`. If, for example, a user on `orchid` typed

```
% lpr -P bamboo -d sushi-review.dvi
```

the LPD system on `orchid` would copy the job to the spooling directory `/var/spool/lpd/bamboo` and note that it was a DVI job. As soon as the host `rose` has room in its `bamboo` spooling directory, the two

LPDs would transfer the file to rose. The file would wait in rose's queue until it was finally printed. It would be converted from DVI to PostScript (since bamboo is a PostScript printer) on rose.

## Printers with Networked Data Stream Interfaces

Often, when you buy a network interface card for a printer, you can get two versions: one which emulates a spooler (the more expensive version), or one which just lets you send data to it as if you were using a serial or parallel port (the cheaper version). This section tells how to use the cheaper version. For the more expensive one, see the previous section Printers Installed on Remote Hosts.

The format of the `/etc/printcap` file lets you specify what serial or parallel interface to use, and (if you are using a serial interface), what baud rate, whether to use flow control, delays for tabs, conversion of newlines, and more. But there is no way to specify a connection to a printer that is listening on a TCP/IP or other network port.

To send data to a networked printer, you need to develop a communications program that can be called by the text and conversion filters. Here is one such example: the script `netprint` takes all data on standard input and sends it to a network-attached printer. We specify the hostname of the printer as the first argument and the port number to which to connect as the second argument to `netprint`. Note that this supports one-way communication only (FreeBSD to printer); many network printers support two-way communication, and you might want to take advantage of that (to get printer status, perform accounting, etc.).

```
#!/usr/bin/perl
#
# netprint - Text filter for printer attached to network
# Installed in /usr/local/libexec/netprint
#
$#ARGV eq 1 || die "Usage: $0 <printer-hostname> <port-number>";

$printer_host = $ARGV[0];
$printer_port = $ARGV[1];

require 'sys/socket.ph';

($ignore, $ignore, $protocol) = getprotobyname('tcp');
($ignore, $ignore, $ignore, $ignore, $address)
    = gethostbyname($printer_host);

$sockaddr = pack('S n a4 x8', &AF_INET, $printer_port, $address);

socket(PRINTER, &PF_INET, &SOCK_STREAM, $protocol)
    || die "Can't create TCP/IP stream socket: $!";
```

```
connect(PRINTER, $sockaddr) || die "Can't contact $printer_host: $!";
while (<STDIN>) { print PRINTER; }
exit 0;
```

We can then use this script in various filters. Suppose we had a Diablo 750-N line printer connected to the network. The printer accepts data to print on port number 5100. The host name of the printer is scrivener. Here is the text filter for the printer:

```
#!/bin/sh
#
# diablo-if-net - Text filter for Diablo printer 'scrivener' listening
# on port 5100. Installed in /usr/local/libexec/diablo-if-net # exec
/usr/libexec/lpr/lpf "$@" | /usr/local/libexec/netprint scrivener 5100
```

## Restricting Printer Usage

This section gives information on restricting printer usage. The LPD system lets you control who can access a printer, both locally or remotely, whether they can print multiple copies, how large their jobs can be, and how large the printer queues can get.

### Restricting Multiple Copies

The LPD system makes it easy for users to print multiple copies of a file. Users can print jobs with `lpr -#5` (for example) and get five copies of each file in the job. Whether this is a good thing is up to you.

If you feel multiple copies cause unnecessary wear and tear on your printers, you can disable the `-#` option to `lpr(1)` by adding the `sc` capability to the `/etc/printcap` file. When users submit jobs with the `-#` option, they will see:

```
lpr: multiple copies are not allowed
```

Note that if you have set up access to a printer remotely (see section [Printers Installed on Remote Hosts](#)), you need the `sc` capability on the remote `/etc/printcap` files as well, or else users will still be able to submit multiple-copy jobs by using another host.

Here is an example. This is the `/etc/printcap` file for the host `rose`. The printer `rattan` is quite hearty, so we will allow multiple copies, but the laser printer `bamboo`'s a bit more delicate, so we will disable multiple copies by adding the `sc` capability:

```
#
# /etc/printcap for host rose - restrict multiple copies on bamboo
#
```

```
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:sc:\
    :lp=/dev/ttyd5:fs#0x8200e1:xs#0x820:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

Now, we also need to add the `sc` capability on the host orchid's `/etc/printcap` (and while we are at it, let us disable multiple copies for the printer teak):

```
#
# /etc/printcap for host orchid - no multiple copies for local
# printer teak or remote printer bamboo
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:sc:\
    :if=/usr/local/libexec/ifhp:\
    :vf=/usr/local/libexec/vfhp:\
    :of=/usr/local/libexec/ofhp:

rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:sc:
```

By using the `sc` capability, we prevent the use of `lpr -#`, but that still does not prevent users from running `lpr(1)` multiple times, or from submitting the same file multiple times in one job like this:

```
% lpr forsale.sign forsale.sign forsale.sign forsale.sign forsale.sign
```

There are many ways to prevent this abuse (including ignoring it) which you are free to explore.

## Restricting Access To Printers

You can control who can print to what printers by using the UNIX group mechanism and the `rg` capability in `/etc/printcap`. Just place the users you want to have access to a printer in a certain group, and then name that group in the `rg` capability.

Users outside the group (including root) will be greeted with `lpr`: Not a member of the restricted group if they try to print to the controlled printer.

As with the `sc` (suppress multiple copies) capability, you need to specify `rg` on remote hosts that also have access to your printers, if you feel it is appropriate (see section Printers Installed on Remote Hosts).

For example, we will let anyone access the printer `rattan`, but only those in group `artists` can use `bamboo`. Here is the familiar `/etc/printcap` for host `rose`:

```
#
# /etc/printcap for host rose - restricted group for bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:\
    :lp=/dev/ttyd5:fs#0x8200e1:xs#0x820:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

Let us leave the other example `/etc/printcap` file (for the host `orchid`) alone. Of course, anyone on `orchid` can print to `bamboo`. It might be the case that we only allow certain logins on `orchid` anyway, and want them to have access to the printer. Or not.

**Note:** There can be only one restricted group per printer.

## Controlling Sizes of Jobs Submitted

If you have many users accessing the printers, you probably need to put an upper limit on the sizes of the files users can submit to print. After all, there is only so much free space on the filesystem that houses the spooling directories, and you also need to make sure there is room for the jobs of other users.

LPD enables you to limit the maximum byte size a file in a job can be with the `mx` capability. The units are in BUFSIZ blocks, which are 1024 bytes. If you put a zero for this capability, there will be no limit on file size.

**Note:** The limit applies to *files* in a job, and *not* the total job size.

LPD will not refuse a file that is larger than the limit you place on a printer. Instead, it will queue as much of the file up to the limit, which will then get printed. The rest will be discarded. Whether this is correct behavior is up for debate.

Let us add limits to our example printers `rattan` and `bamboo`. Since those artists' PostScript files tend to be large, we will limit them to five megabytes. We will put no limit on the plain text line printer:

```
#
# /etc/printcap for host rose
#

#
# No limit on job size:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

#
# Limit of five megabytes:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
    :lp=/dev/ttyd5:fs#0x8200e1:xs#0x820:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

Again, the limits apply to the local users only. If you have set up access to your printers remotely, remote users will not get those limits. You will need to specify the `mx` capability in the remote `/etc/printcap` files as well. See section [Printers Installed on Remote Hosts](#) for more information on remote printing.

There is another specialized way to limit job sizes from remote printers; see section [Restricting Jobs from Remote Printers](#).

## Restricting Jobs from Remote Printers

The LPD spooling system provides several ways to restrict print jobs submitted from remote hosts:

### Host restrictions

You can control from which remote hosts a local LPD accepts requests with the files `/etc/hosts.equiv` and `/etc/hosts.lpd`. LPD checks to see if an incoming request is from a host listed in either one of these files. If not, LPD refuses the request.

The format of these files is simple: one host name per line. Note that the file `/etc/hosts.equiv` is also used by the `ruserok(3)` protocol, and affects programs like `rsh(1)` and `rcp(1)`, so be careful.

For example, here is the `/etc/hosts.lpd` file on the host `rose`:

```
orchid
violet
madrival.fishbaum.de
```

This means `rose` will accept requests from the hosts `orchid`, `violet`, and `madrival.fishbaum.de`. If any other host tries to access `rose`'s LPD, LPD will refuse them.

### Size restrictions

You can control how much free space there needs to remain on the filesystem where a spooling directory resides. Make a file called `minfree` in the spooling directory for the local printer. Insert in that file a number representing how many disk blocks (512 bytes) of free space there has to be for a remote job to be accepted.

This lets you insure that remote users will not fill your filesystem. You can also use it to give a certain priority to local users: they will be able to queue jobs long after the free disk space has fallen below the amount specified in the `minfree` file.

For example, let us add a `minfree` file for the printer `bamboo`. We examine `/etc/printcap` to find the spooling directory for this printer; here is `bamboo`'s entry:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
    :lp=/dev/ttyd5:fs#0x8200e1:xs#0x820:rw:mx#5000:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

The spooling directory is the given in the `sd` capability. We will make three megabytes (which is 6144 disk blocks) the amount of free disk space that must exist on the filesystem for LPD to accept remote jobs:

```
# echo 6144 > /var/spool/lpd/bamboo/minfree
```

### User restrictions

You can control which remote users can print to local printers by specifying the `rs` capability in `/etc/printcap`. When `rs` appears in the entry for a locally-attached printer, LPD will accept jobs from remote hosts *if* the user submitting the job also has an account of the same login name on the local host. Otherwise, LPD refuses the job.

This capability is particularly useful in an environment where there are (for example) different departments sharing a network, and some users transcend departmental boundaries. By giving them accounts on your systems, they can use your printers from their own departmental systems. If you

would rather allow them to use *only* your printers and not your compute resources, you can give them “token” accounts, with no home directory and a useless shell like `/usr/bin/false`.

## Accounting for Printer Usage

So, you need to charge for printouts. And why not? Paper and ink cost money. And then there are maintenance costs—printers are loaded with moving parts and tend to break down. You have examined your printers, usage patterns, and maintenance fees and have come up with a per-page (or per-foot, per-meter, or per-whatever) cost. Now, how do you actually start accounting for printouts?

Well, the bad news is the LPD spooling system does not provide much help in this department. Accounting is highly dependent on the kind of printer in use, the formats being printed, and *your* requirements in charging for printer usage.

To implement accounting, you have to modify a printer’s text filter (to charge for plain text jobs) and the conversion filters (to charge for other file formats), to count pages or query the printer for pages printed. You cannot get away with using the simple output filter, since it cannot do accounting. See section Filters.

Generally, there are two ways to do accounting:

- *Periodic accounting* is the more common way, possibly because it is easier. Whenever someone prints a job, the filter logs the user, host, and number of pages to an accounting file. Every month, semester, year, or whatever time period you prefer, you collect the accounting files for the various printers, tally up the pages printed by users, and charge for usage. Then you truncate all the logging files, starting with a clean slate for the next period.
- *Timely accounting* is less common, probably because it is more difficult. This method has the filters charge users for printouts as soon as they use the printers. Like disk quotas, the accounting is immediate. You can prevent users from printing when their account goes in the red, and might provide a way for users to check and adjust their “print quotas.” But this method requires some database code to track users and their quotas.

The LPD spooling system supports both methods easily: since you have to provide the filters (well, most of the time), you also have to provide the accounting code. But there is a bright side: you have enormous flexibility in your accounting methods. For example, you choose whether to use periodic or timely accounting. You choose what information to log: user names, host names, job types, pages printed, square footage of paper used, how long the job took to print, and so forth. And you do so by modifying the filters to save this information.

## Quick and Dirty Printer Accounting

FreeBSD comes with two programs that can get you set up with simple periodic accounting right away. They are the text filter `lpf`, described in section `lpf: a Text Filter`, and `pac(8)`, a program to gather and total entries from printer accounting files.

As mentioned in the section on filters (Filters), LPD starts the text and the conversion filters with the name of the accounting file to use on the filter command line. The filters can use this argument to know where to write an accounting file entry. The name of this file comes from the `af` capability in `/etc/printcap`, and if not specified as an absolute path, is relative to the spooling directory.

LPD starts `lpf` with page width and length arguments (from the `pw` and `pl` capabilities). `lpf` uses these arguments to determine how much paper will be used. After sending the file to the printer, it then writes an accounting entry in the accounting file. The entries look like this:

```
2.00 rose:andy
3.00 rose:kelly
3.00 orchid:mary
5.00 orchid:mary
2.00 orchid:zhang
```

You should use a separate accounting file for each printer, as `lpf` has no file locking logic built into it, and two `lpfs` might corrupt each other's entries if they were to write to the same file at the same time. A easy way to insure a separate accounting file for each printer is to use `af=acct` in `/etc/printcap`. Then, each accounting file will be in the spooling directory for a printer, in a file named `acct`.

When you are ready to charge users for printouts, run the `pac(8)` program. Just change to the spooling directory for the printer you want to collect on and type `pac`. You will get a dollar-centric summary like the following:

Login	pages/feet	runs	price
orchid:kelly	5.00	1	\$ 0.10
orchid:mary	31.00	3	\$ 0.62
orchid:zhang	9.00	1	\$ 0.18
rose:andy	2.00	1	\$ 0.04
rose:kelly	177.00	104	\$ 3.54
rose:mary	87.00	32	\$ 1.74
rose:root	26.00	12	\$ 0.52
total	337.00	154	\$ 6.74

These are the arguments `pac(8)` expects:

`-Pprinter`

Which *printer* to summarize. This option works only if there is an absolute path in the `af` capability in `/etc/printcap`.

`-c`

Sort the output by cost instead of alphabetically by user name.

`-m`

Ignore host name in the accounting files. With this option, user `smith` on host `alpha` is the same user `smith` on host `gamma`. Without, they are different users.

`-pprice`

Compute charges with *price* dollars per page or per foot instead of the price from the `pc` capability in `/etc/printcap`, or two cents (the default). You can specify *price* as a floating point number.

`-r`

Reverse the sort order.

`-s`

Make an accounting summary file and truncate the accounting file.

*name . . .*

Print accounting information for the given user *names* only.

In the default summary that `pac(8)` produces, you see the number of pages printed by each user from various hosts. If, at your site, host does not matter (because users can use any host), run `pac -m`, to produce the following summary:

Login	pages/foot	runs	price
andy	2.00	1	\$ 0.04
kelly	182.00	105	\$ 3.64
mary	118.00	35	\$ 2.36
root	26.00	12	\$ 0.52
zhang	9.00	1	\$ 0.18
total	337.00	154	\$ 6.74

To compute the dollar amount due, `pac(8)` uses the `pc` capability in the `/etc/printcap` file (default of 200, or 2 cents per page). Specify, in hundredths of cents, the price per page or per foot you want to charge for printouts in this capability. You can override this value when you run `pac(8)` with the `-p` option. The units for the `-p` option are in dollars, though, not hundredths of cents. For example,

```
# pac -p1.50
```

makes each page cost one dollar and fifty cents. You can really rake in the profits by using this option.

Finally, running `pac -s` will save the summary information in a summary accounting file, which is named the same as the printer's accounting file, but with `_sum` appended to the name. It then truncates the accounting file. When you run `pac(8)` again, it rereads the summary file to get starting totals, then adds information from the regular accounting file.

## How Can You Count Pages Printed?

In order to perform even remotely accurate accounting, you need to be able to determine how much paper a job uses. This is the essential problem of printer accounting.

For plain text jobs, the problem's not that hard to solve: you count how many lines are in a job and compare it to how many lines per page your printer supports. Do not forget to take into account backspaces in the file which overprint lines, or long logical lines that wrap onto one or more additional physical lines.

The text filter `lpf` (introduced in `lpf: a Text Filter`) takes into account these things when it does accounting. If you are writing a text filter which needs to do accounting, you might want to examine `lpf`'s source code.

How do you handle other file formats, though?

Well, for DVI-to-LaserJet or DVI-to-PostScript conversion, you can have your filter parse the diagnostic output of `dvi2lj` or `dvi2ps` and look to see how many pages were converted. You might be able to do similar things with other file formats and conversion programs.

But these methods suffer from the fact that the printer may not actually print all those pages. For example, it could jam, run out of toner, or explode—and the user would still get charged.

So, what can you do?

There is only one *sure* way to do *accurate* accounting. Get a printer that can tell you how much paper it uses, and attach it via a serial line or a network connection. Nearly all PostScript printers support this notion. Other makes and models do as well (networked Imagen laser printers, for example). Modify the filters for these printers to get the page usage after they print each job and have them log accounting information based on that value *only*. There is no line counting nor error-prone file examination required.

Of course, you can always be generous and make all printouts free.

## Alternatives to the Standard Spooler

If you have been reading straight through this manual, by now you have learned just about everything there is to know about the LPD spooling system that comes with FreeBSD. You can probably appreciate many of its shortcomings, which naturally leads to the question: “What other spooling systems are out there (and work with FreeBSD)?”

Unfortunately, I have located only *two* alternatives—and they are almost identical to each other! They are:

### PLP, the Portable Line Printer Spooler System

PLP was based on software developed by Patrick Powell and then maintained by an Internet-wide group of developers. The main site for the software is at <ftp://ftp.iona.ie/pub/plp>. There is also a web page (<http://www.iona.ie:8000/www/hyplan/jmason/plp.html>).

It is quite similar to the BSD LPD spooler, but boasts a host of features, including:

- Better network support, including built-in support for networked printers, NIS-maintained printcaps, and NFS-mounted spooling directories
- Sophisticated queue management, allowing multiple printers on a queue, transfer of jobs between queues, and queue redirection
- Remote printer control functions
- Prioritization of jobs
- Expansive security and access options

### LPRng

LPRng, which purportedly means “LPR: the Next Generation” is a complete rewrite of PLP. Patrick Powell and Justin Mason (the principal maintainer of PLP) collaborated to make LPRng. The main site for LPRng is <ftp://dickory.sdsu.edu/pub/LPRng>.

## Acknowledgments

I would like to thank the following people who have assisted in the development of this document:

Daniel Eischen <deischen@iworks.interworks.org>

For providing a plethora of HP filter programs for perusal.

Jake Hamby <jehamby@lightside.com>

For the Ghostscript-to-HP filter.

My wife, Mary Kelly <urquhart@argyre.colorado.edu>

For allowing me to spend more time with FreeBSD than with her.

## Chapter 8. Disks

*Contributed by David O'Brien <obrien@FreeBSD.ORG> 26 April 1998*

Lets say we want to add a new SCSI disk to a machine that currently only has a single drive. First turn off the computer and install the drive in the computer following the instructions of the computer, controller, and drive manufacturer. Due the wide variations of procedures to do this, the details are beyond the scope of this document.

Login as user `root`. After you've installed the drive, inspect `/var/run/dmesg.boot` to ensure the new disk was found. Continuing with our example, the newly added drive will be `sd1` and we want to mount it on `/1`. (if you are adding an IDE drive substitute `wd` for `sd`)

Because FreeBSD runs on IBM-PC compatible computers, it must take into account the PC BIOS partitions. These are different from the traditional BSD partitions. A PC disk has up to four BIOS partition entries. If the disk is going to be truly dedicated to FreeBSD, you can use the *dedicated* mode. Otherwise, FreeBSD will have to live with in one of the PC BIOS partitions. FreeBSD calls the PC BIOS partitions, *slices* so as not to confuse them with traditional BSD partitions. You may also use slices on a disk that is dedicated to FreeBSD, but used in a computer that also has another operating system installed. This is to not confuse the `fdisk` utility of the other operating system.

In the slice case the drive will be added as `/dev/sd1s1e`. This is read as: SCSI disk, unit number 1 (second SCSI disk), slice 1 (PC BIOS partition 1), and e BSD partition. In the dedicated case, the drive will be added simply as `/dev/sd1e`.

## Using sysinstall

You may use `/stand/sysinstall` to partition and label a new disk using its easy to use menus. Either login as user `root` or use the `su` command. Run `/stand/sysinstall` and enter the `Configure` menu. With in the `FreeBSD Configuration Menu`, scroll down and select the `Partition` item. Next you should be presented with a list of hard drives installed in your system. If you do not see `sd1` listed, you need to recheck your physical installation and `dmesg` output in the file `/var/run/dmesg.boot`.

Select `sd1` to enter the `FDISK Partition Editor`. Choose `A` to use the entire disk for FreeBSD. When asked if you want to "remain cooperative with any future possible operating systems", answer `YES`. Write the changes to the disk using `w`. Now exit the `FDISK` editor using `q`. Next you will be asked about the Master Boot Record. Since you are adding a disk to an already running system, choose `None`.

Next enter the `Disk Label Editor`. This is where you will create the traditional BSD partitions. A disk can have up to eight partitions, labeled a-h. A few of the partition labels have special uses. The `a` partition is used for the root partition (`/`). Thus only your system disk (e.g, the disk you boot from) should have an `a` partition. The `b` partition is used for swap partitions, and you may have many disks

with swap partitions. The `c` partition addresses the entire disk in dedicated mode, or the entire FreeBSD slice in slice mode. The other partitions are for general use.

Sysinstall's Label editor favors the `e` partition for non-root, non-swap partitions. With in the Label editor, create a single file system using `C`. When prompted if this will be a FS (file system) or swap, choose `FS` and give a mount point (e.g, `/mnt`). When adding a disk in post-install mode, Sysinstall will not create entries in `/etc/fstab` for you, so the mount point you specify isn't important.

You are now ready to write the new label to the disk and create a file system on it. Do this by hitting `w`. Ignore any errors from Sysinstall that it could not mount the new partition. Exit the Label Editor and Sysinstall completely.

The last step is to edit `/etc/fstab` to add an entry for your new disk.

## Using command line utilities

### \* Using Slices

#### Dedicated

If you will not be sharing the new drive with another operating system, you may use the `dedicated` mode. Remember this mode can confuse Microsoft operating systems; however, no damage will be done by them. IBM's OS/2 however, will "appropriate" any partition it finds which it doesn't understand.

```
# dd if=/dev/zero of=/dev/rsd1 bs=1k count=1
# disklabel -Brw sd1 auto
# disklabel -e sd1 # create the 'e' partition
# newfs -d0 /dev/rsd1e
# mkdir -p /1
# vi /etc/fstab # add an entry for /dev/sdle
# mount /1
```

An alternate method is:

```
# dd if=/dev/zero of=/dev/rsd1 count=2
# disklabel /dev/rsd1 | disklabel -BrR sd1 /dev/stdin
# newfs /dev/rsd1e
# mkdir -p /1
# vi /etc/fstab # add an entry for /dev/sdle
# mount /1
```

**\* Non-traditional Drives**

**\* Zip Drives**

**\* Jazz Drives**

**\* Sequest Drives**

## Chapter 9. Backups

Issues of hardware compatibility are among the most troublesome in the computer industry today and FreeBSD is by no means immune to trouble. In this respect, FreeBSD's advantage of being able to run on inexpensive commodity PC hardware is also its liability when it comes to support for the amazing variety of components on the market. While it would be impossible to provide an exhaustive listing of hardware that FreeBSD supports, this section serves as a catalog of the device drivers included with FreeBSD and the hardware each driver supports. Where possible and appropriate, notes about specific products are included. You may also want to refer to the kernel configuration file section in this handbook for a list of supported devices.

As FreeBSD is a volunteer project without a funded testing department, we depend on you, the user, for much of the information contained in this catalog. If you have direct experience of hardware that does or does not work with FreeBSD, please let us know by sending e-mail to the FreeBSD documentation project mailing list <freebsd-doc@FreeBSD.ORG>. Questions about supported hardware should be directed to the FreeBSD general questions mailing list <freebsd-questions@FreeBSD.ORG> (see Mailing Lists for more information). When submitting information or asking a question, please remember to specify exactly what version of FreeBSD you are using and include as many details of your hardware as possible.

### \* What about backups to floppies?

## Tape Media

The major tape media are the 4mm, 8mm, QIC, mini-cartridge and DLT.

### 4mm (DDS: Digital Data Storage)

4mm tapes are replacing QIC as the workstation backup media of choice. This trend accelerated greatly when Conner purchased Archive, a leading manufacturer of QIC drives, and then stopped production of QIC drives. 4mm drives are small and quiet but do not have the reputation for reliability that is enjoyed by 8mm drives. The cartridges are less expensive and smaller (3 x 2 x 0.5 inches, 76 x 51 x 12 mm) than 8mm cartridges. 4mm, like 8mm, has comparatively short head life for the same reason, both use helical scan.

Data throughput on these drives starts ~150kB/s, peaking at ~500kB/s. Data capacity starts at 1.3 GB and ends at 2.0 GB. Hardware compression, available with most of these drives, approximately doubles the

capacity. Multi-drive tape library units can have 6 drives in a single cabinet with automatic tape changing. Library capacities reach 240 GB.

4mm drives, like 8mm drives, use helical-scan. All the benefits and drawbacks of helical-scan apply to both 4mm and 8mm drives.

Tapes should be retired from use after 2,000 passes or 100 full backups.

## 8mm (Exabyte)

8mm tapes are the most common SCSI tape drives; they are the best choice of exchanging tapes. Nearly every site has an exabyte 2 GB 8mm tape drive. 8mm drives are reliable, convenient and quiet. Cartridges are inexpensive and small (4.8 x 3.3 x 0.6 inches; 122 x 84 x 15 mm). One downside of 8mm tape is relatively short head and tape life due to the high rate of relative motion of the tape across the heads.

Data thrupt ranges from ~250kB/s to ~500kB/s. Data sizes start at 300 MB and go up to 7 GB. Hardware compression, available with most of these drives, approximately doubles the capacity. These drives are available as single units or multi-drive tape libraries with 6 drives and 120 tapes in a single cabinet. Tapes are changed automatically by the unit. Library capacities reach 840+ GB.

Data is recorded onto the tape using helical-scan, the heads are positioned at an angle to the media (approximately 6 degrees). The tape wraps around 270 degrees of the spool that holds the heads. The spool spins while the tape slides over the spool. The result is a high density of data and closely packed tracks that angle across the tape from one edge to the other.

## QIC

QIC-150 tapes and drives are, perhaps, the most common tape drive and media around. QIC tape drives are the least expensive "serious" backup drives. The downside is the cost of media. QIC tapes are expensive compared to 8mm or 4mm tapes, up to 5 times the price per GB data storage. But, if your needs can be satisfied with a half-dozen tapes, QIC may be the correct choice. QIC is the *most* common tape drive. Every site has a QIC drive of some density or another. Therein lies the rub, QIC has a large number of densities on physically similar (sometimes identical) tapes. QIC drives are not quiet. These drives audibly seek before they begin to record data and are clearly audible whenever reading, writing or seeking. QIC tapes measure (6 x 4 x 0.7 inches; 15.2 x 10.2 x 1.7 mm). Mini-cartridges, which also use 1/4" wide tape are discussed separately. Tape libraries and changers are not available.

Data thrupt ranges from ~150kB/s to ~500kB/s. Data capacity ranges from 40 MB to 15 GB. Hardware compression is available on many of the newer QIC drives. QIC drives are less frequently installed; they are being supplanted by DAT drives.

Data is recorded onto the tape in tracks. The tracks run along the long axis of the tape media from one end to the other. The number of tracks, and therefore the width of a track, varies with the tape's capacity. Most if not all newer drives provide backward-compatibility at least for reading (but often also for writing). QIC has a good reputation regarding the safety of the data (the mechanics are simpler and more robust than for helical scan drives).

Tapes should be retired from use after 5,000 backups.

## \* Mini-Cartridge

## DLT

DLT has the fastest data transfer rate of all the drive types listed here. The 1/2" (12.5mm) tape is contained in a single spool cartridge (4 x 4 x 1 inches; 100 x 100 x 25 mm). The cartridge has a swinging gate along one entire side of the cartridge. The drive mechanism opens this gate to extract the tape leader. The tape leader has an oval hole in it which the drive uses to "hook" the tape. The take-up spool is located inside the tape drive. All the other tape cartridges listed here (9 track tapes are the only exception) have both the supply and take-up spools located inside the tape cartridge itself.

Data thrupt is approximately 1.5MB/s, three times the thrupt of 4mm, 8mm, or QIC tape drives. Data capacities range from 10GB to 20GB for a single drive. Drives are available in both multi-tape changers and multi-tape, multi-drive tape libraries containing from 5 to 900 tapes over 1 to 20 drives, providing from 50GB to 9TB of storage.

Data is recorded onto the tape in tracks parallel to the direction of travel (just like QIC tapes). Two tracks are written at once. Read/write head lifetimes are relatively long; once the tape stops moving, there is no relative motion between the heads and the tape.

## Using a new tape for the first time

The first time that you try to read or write a new, completely blank tape, the operation will fail. The console messages should be similar to:

```
st0(ncr1:4:0): NOT READY asc:4,1
st0(ncr1:4:0): Logical unit is in process of becoming ready
```

The tape does not contain an Identifier Block (block number 0). All QIC tape drives since the adoption of QIC-525 standard write an Identifier Block to the tape. There are two solutions:

`mt fsf 1` causes the tape drive to write an Identifier Block to the tape.

Use the front panel button to eject the tape.

Re-insert the tape and dump(8) data to the tape.

dump(8) will report `DUMP: End of tape detected` and the console will show: `HARDWARE FAILURE`  
`info:280 asc:80,96`

rewind the tape using: `mt rewind`

Subsequent tape operations are successful.

## Backup Programs

The three major programs are dump(8), tar(1), and cpio(1).

### Dump and Restore

dump(8) and restore(8) are the traditional Unix backup programs. They operate on the drive as a collection of disk blocks, below the abstractions of files, links and directories that are created by the filesystems. dump(8) backs up devices, entire filesystems, not parts of a filesystem and not directory trees that span more than one filesystem, using either soft links ln(1) or mounting one filesystem onto another. dump(8) does not write files and directories to tape, but rather writes the data blocks that are the building blocks of files and directories. dump(8) has quirks that remain from its early days in Version 6 of ATT Unix (circa 1975). The default parameters are suitable for 9-track tapes (6250 bpi), not the high-density media available today (up to 62,182 ftpi). These defaults must be overridden on the command line to utilize the capacity of current tape drives.

rdump(8) and rrestore(8) backup data across the network to a tape drive attached to another computer. Both programs rely upon rcmd(3) and ruserok(3) to access the remote tape drive. Therefore, the user performing the backup must have rhosts access to the remote computer. The arguments to rdump(8) and rrestore(8) must suitable to use on the remote computer. (e.g. When rdump'ing from a FreeBSD computer to an Exabyte tape drive connected to a Sun called komodo, use: `/sbin/rdump 0dsbfu 54000 13000 126 komodo:/dev/nrst8 /dev/rsd0a 2>&1`) Beware: there are security implications to allowing rhosts commands. Evaluate your situation carefully.

### Tar

tar(1) also dates back to Version 6 of ATT Unix (circa 1975). tar(1) operates in cooperation with the filesystem; tar(1) writes files and directories to tape. tar(1) does not support the full range of options that are available from cpio(1), but tar(1) does not require the unusual command pipeline that cpio(1) uses.

Most versions of `tar(1)` do not support backups across the network. The GNU version of `tar(1)`, which FreeBSD utilizes, supports remote devices using the same syntax as `rdump(8)`. To `tar(1)` to an Exabyte tape drive connected to a Sun called `komodo`, use: `/usr/bin/tar cf komodo:/dev/nrst8 . 2>&1`. For versions without remote device support, you can use a pipeline and `rsh(1)` to send the data to a remote tape drive. (XXX add an example command)

## Cpio

`cpio(1)` is the original Unix file interchange tape program for magnetic media. `cpio(1)` has options (among many others) to perform byte-swapping, write a number of different archives format, and pipe the data to other programs. This last feature makes `cpio(1)` an excellent choice for installation media. `cpio(1)` does not know how to walk the directory tree and a list of files must be provided thru `STDIN`.

`cpio(1)` does not support backups across the network. You can use a pipeline and `rsh(1)` to send the data to a remote tape drive. (XXX add an example command)

## Pax

`pax(1)` is IEEE/POSIX's answer to `tar(1)` and `cpio(1)`. Over the years the various versions of `tar(1)` and `cpio(1)` have gotten slightly incompatible. So rather than fight it out to fully standardize them, POSIX created a new archive utility. `pax(1)` attempts to read and write many of the various `cpio(1)` and `tar(1)` formats, plus new formats of its own. Its command set more resembles `cpio(1)` than `tar(1)`.

## Amanda

Amanda ([../ports/misc.html#amanda-2.4.0](#)) (Advanced Maryland Network Disk Archiver) is a client/server backup system, rather than a single program. An Amanda server will backup to a single tape drive any number of computers that have Amanda clients and network communications with the Amanda server. A common problem at locations with a number of large disks is the length of time required to backup to data directly to tape exceeds the amount of time available for the task. Amanda solves this problem. Amanda can use a "holding disk" to backup several filesystems at the same time. Amanda creates "archive sets": a group of tapes used over a period of time to create full backups of all the filesystems listed in Amanda's configuration file. The "archive set" also contains nightly incremental (or differential) backups of all the filesystems. Restoring a damaged filesystem requires the most recent full backup and the incremental backups.

The configuration file provides fine control backups and the network traffic that Amanda generates. Amanda will use any of the above backup programs to write the data to tape. Amanda is available as either a port or a package, it is not installed by default.

## Do nothing

“Do nothing” is not a computer program, but it is the most widely used backup strategy. There are no initial costs. There is no backup schedule to follow. Just say no. If something happens to your data, grin and bear it!

If your time and your data is worth little to nothing, then “Do nothing” is the most suitable backup program for your computer. But beware, Unix is a useful tool, you may find that within six months you have a collection of files that are valuable to you.

“Do nothing” is the correct backup method for `/usr/obj` and other directory trees that can be exactly recreated by your computer. An example is the files that comprise these handbook pages—they have been generated from SGML input files. Creating backups of these HTML files is not necessary. The SGML source files are backed up regularly.

## Which Backup Program is Best?

`dump(8)` *Period*. Elizabeth D. Zwicky torture tested all the backup programs discussed here. The clear choice for preserving all your data and all the peculiarities of Unix filesystems is `dump(8)`. Elizabeth created filesystems containing a large variety of unusual conditions (and some not so unusual ones) and tested each program by do a backup and restore of that filesystems. The peculiarities included: files with holes, files with holes and a block of nulls, files with funny characters in their names, unreadable and unwritable files, devices, files that change size during the backup, files that are created/deleted during the backup and more. She presented the results at LISA V in Oct. 1991. See torture-testing Backup and Archive Programs ([http://reality.sgi.com/zwicky\\_neu/testdump.doc.html](http://reality.sgi.com/zwicky_neu/testdump.doc.html)).

## Emergency Restore Procedure

### Before the Disaster

There are only four steps that you need to perform in preparation for any disaster that may occur.

First, print the disklabel from each of your disks (e.g. `disklabel sd0 | lpr`), your filesystem table (`/etc/fstab`) and all boot messages, two copies of each.

Second, determine that the boot and fixit floppies (`boot.flp` and `fixit.flp`) have all your devices. The easiest way to check is to reboot your machine with the boot floppy in the floppy drive and check the boot messages. If all your devices are listed and functional, skip on to step three.

Otherwise, you have to create two custom bootable floppies which has a kernel that can mount your all of your disks and access your tape drive. These floppies must contain: `fdisk(8)`, `disklabel(8)`, `newfs(8)`,

mount(8), and whichever backup program you use. These programs must be statically linked. If you use dump(8), the floppy must contain restore(8).

Third, create backup tapes regularly. Any changes that you make after your last backup may be irretrievably lost. Write-protect the backup tapes.

Fourth, test the floppies (either `boot.flp` and `fixit.flp` or the two custom bootable floppies you made in step two.) and backup tapes. Make notes of the procedure. Store these notes with the bootable floppy, the printouts and the backup tapes. You will be so distraught when restoring that the notes may prevent you from destroying your backup tapes (How? In place of `tar xvf /dev/rst0`, you might accidentally type `tar cvf /dev/rst0` and over-write your backup tape).

For an added measure of security, make bootable floppies and two backup tapes each time. Store one of each at a remote location. A remote location is NOT the basement of the same office building. A number of firms in the World Trade Center learned this lesson the hard way. A remote location should be physically separated from your computers and disk drives by a significant distance.

An example script for creating a bootable floppy:

```
#!/bin/sh
#
# create a restore floppy
#
# format the floppy
#
PATH=/bin:/sbin:/usr/sbin:/usr/bin

fdformat -q fd0
if [ $? -ne 0 ]
then
    echo "Bad floppy, please use a new one"
    exit 1
fi

# place boot blocks on the floppy
#
disklabel -w -B -b /usr/mdec/fdboot -s /usr/mdec/bootfd /dev/rfd0c fd1440

#
# newfs the one and only partition
#
newfs -t 2 -u 18 -l 1 -c 40 -i 5120 -m 5 -o space /dev/rfd0a

#
# mount the new floppy
#
```

```

mount /dev/fd0a /mnt

#
# create required directories
#
mkdir /mnt/dev
mkdir /mnt/bin
mkdir /mnt/sbin
mkdir /mnt/etc
mkdir /mnt/root
mkdir /mnt/mnt # for the root partition
mkdir /mnt/tmp
mkdir /mnt/var

#
# populate the directories
#
if [ ! -x /sys/compile/MINI/kernel ]
then
  cat < EOM
The MINI kernel does not exist, please create one.
Here is an example config file:
#
# MINI - A kernel to get FreeBSD on onto a disk.
#
machine "i386"
cpu "I486_CPU"
ident MINI
maxusers 5

options INET # needed for _tcp _icmpstat _ipstat
#           _udpstat _tcpstat _udb
options FFS #Berkeley Fast File System
options FAT_CURSOR #block cursor in syscons or pccons
options SCSI_DELAY=15 #Be pessimistic about Joe SCSI device
options NCONS=2 #1 virtual consoles
options USERCONFIG #Allow user configuration with -c XXX

config kernel root on sd0 swap on sd0 and sd1 dumps on sd0

controller isa0
controller pci0

controller fdc0 at isa? port "IO_FD1" bio irq 6 drq 2 vector fdintr
disk fd0 at fdc0 drive 0

```

```

controller ncr0

controller scbus0

device sc0 at isa? port "IO_KBD" tty irq 1 vector scintr
device npx0 at isa? port "IO_NPX" irq 13 vector npxintr

device sd0
device sd1
device sd2

device st0

pseudo-device loop # required by INET
pseudo-device gzip # Exec gzipped a.out's
EOM
  exit 1
fi

cp -f /sys/compile/MINI/kernel /mnt

gzip -c -best /sbin/init > /mnt/sbin/init
gzip -c -best /sbin/fsck > /mnt/sbin/fsck
gzip -c -best /sbin/mount > /mnt/sbin/mount
gzip -c -best /sbin/halt > /mnt/sbin/halt
gzip -c -best /sbin/restore > /mnt/sbin/restore

gzip -c -best /bin/sh > /mnt/bin/sh
gzip -c -best /bin/sync > /mnt/bin/sync

cp /root/.profile /mnt/root

cp -f /dev/MAKEDEV /mnt/dev
chmod 755 /mnt/dev/MAKEDEV

chmod 500 /mnt/sbin/init
chmod 555 /mnt/sbin/fsck /mnt/sbin/mount /mnt/sbin/halt
chmod 555 /mnt/bin/sh /mnt/bin/sync
chmod 6555 /mnt/sbin/restore

#
# create the devices nodes
#
cd /mnt/dev

```

```

./MAKEDEV std
./MAKEDEV sd0
./MAKEDEV sd1
./MAKEDEV sd2
./MAKEDEV st0
./MAKEDEV pty0
cd /

#
# create minimum filesystem table
#
cat > /mnt/etc/fstab «EOM
/dev/fd0a / ufs rw 1 1
EOM

#
# create minimum passwd file
#
cat > /mnt/etc/passwd «EOM
root:*:0:0:Charlie &:/root:/bin/sh
EOM

cat > /mnt/etc/master.passwd «EOM
root::0:0::0:0:Charlie &:/root:/bin/sh
EOM

chmod 600 /mnt/etc/master.passwd
chmod 644 /mnt/etc/passwd
/usr/sbin/pwd_mkdb -d/mnt/etc /mnt/etc/master.passwd

#
# umount the floppy and inform the user
#
/sbin/umount /mnt

```

## After the Disaster

The key question is: did your hardware survive? You have been doing regular backups so there is no need to worry about the software.

If the hardware has been damaged. First, replace those parts that have been damaged.

If your hardware is okay, check your floppies. If you are using a custom boot floppy, boot single-user (type `-s` at the `boot:` prompt). Skip the following paragraph.

If you are using the `boot.flp` and `fixit.flp` floppies, keep reading. Insert the `boot.flp` floppy in the first floppy drive and boot the computer. The original install menu will be displayed on the screen. Select the `Fixit-Repair` mode with `CDROM` or `floppy.` option. Insert the `fixit.flp` when prompted. `restore` and the other programs that you need are located in `/mnt2/stand`.

Recover each filesystem separately.

Try to `mount(8)` (e.g. `mount /dev/sd0a /mnt`) the root partition of your first disk. If the `disklabel` was damaged, use `disklabel(8)` to re-partition and label the disk to match the label that you printed and saved. Use `newfs(8)` to re-create the filesystems. Re-mount the root partition of the floppy read-write (`mount -u -o rw /mnt`). Use your backup program and backup tapes to recover the data for this filesystem (e.g. `restore vrf /dev/st0`). Unmount the filesystem (e.g. `umount /mnt`) Repeat for each filesystem that was damaged.

Once your system is running, backup your data onto new tapes. Whatever caused the crash or data loss may strike again. An another hour spent now, may save you from further distress later.

**\* I did not prepare for the Disaster, What Now?**

# Chapter 10. Disk Quotas

*Contributed by Mike Pritchard <mpp@FreeBSD.ORG>. 26 February 1996*

Quotas are an optional feature of the operating system that allow you to limit the amount of disk space and/or the number of files a user, or members of a group, may allocate on a per-file system basis. This is used most often on timesharing systems where it is desirable to limit the amount of resources any one user or group of users may allocate. This will prevent one user from consuming all of the available disk space.

## Configuring Your System to Enable Disk Quotas

Before attempting to use disk quotas it is necessary to make sure that quotas are configured in your kernel. This is done by adding the following line to your kernel configuration file:

```
options QUOTA
```

The stock `GENERIC` kernel does not have this enabled by default, so you will have to configure, build and install a custom kernel in order to use disk quotas. Please refer to the [Configuring the FreeBSD Kernel](#) section for more information on kernel configuration.

Next you will need to enable disk quotas in `/etc/sysconfig`. This is done by changing the line:

```
quotas=NO
```

to:

```
quotas=YES
```

If you are running FreeBSD 2.2.2 or later, the configuration file will be `/etc/rc.conf` instead and the variable name changed to:

```
check_quotas=YES
```

Finally you will need to edit `/etc/fstab` to enable disk quotas on a per-file system basis. This is where you can either enable user or group quotas or both for all of your file systems.

To enable per-user quotas on a file system, add the `userquota` option to the options field in the `/etc/fstab` entry for the file system you want to enable quotas on. For example:

```
/dev/sd1s2g /home ufs rw,userquota 1 2
```

Similarly, to enable group quotas, use the `groupquota` option instead of the `userquota` keyword. To enable both user and group quotas, change the entry as follows:

```
/dev/sd1s2g    /home    ufs rw,userquota,groupquota 1 2
```

By default the quota files are stored in the root directory of the file system with the names `quota.user` and `quota.group` for user and group quotas respectively. See `man fstab` for more information. Even though that man page says that you can specify an alternate location for the quota files, this is not recommended since all of the various quota utilities do not seem to handle this properly.

At this point you should reboot your system with your new kernel. `/etc/rc` will automatically run the appropriate commands to create the initial quota files for all of the quotas you enabled in `/etc/fstab`, so there is no need to manually create any zero length quota files.

In the normal course of operations you should not be required to run the `quotacheck`, `quotaon`, or `quotaoff` commands manually. However, you may want to read their man pages just to be familiar with their operation.

## Setting Quota Limits

Once you have configured your system to enable quotas, verify that they really are enabled. An easy way to do this is to run

```
# quota -v
```

You should see a one line summary of disk usage and current quota limits for each file system that quotas are enabled on.

You are now ready to start assigning quota limits with the `edquota` command.

You have several options on how to enforce limits on the amount of disk space a user or group may allocate, and how many files they may create. You may limit allocations based on disk space (block quotas) or number of files (inode quotas) or a combination of both. Each of these limits are further broken down into two categories: hard and soft limits.

A hard limit may not be exceeded. Once a user reaches their hard limit they may not make any further allocations on the file system in question. For example, if the user has a hard limit of 500 blocks on a file system and is currently using 490 blocks, the user can only allocate an additional 10 blocks. Attempting to allocate an additional 11 blocks will fail.

Soft limits on the other hand can be exceeded for a limited amount of time. This period of time is known as the grace period, which is one week by default. If a user stays over his or her soft limit longer than their grace period, the soft limit will turn into a hard limit and no further allocations will be allowed. When the user drops back below the soft limit, the grace period will be reset.

The following is an example of what you might see when you run then `edquota` command. When the `edquota` command is invoked, you are placed into the editor specified by the `EDITOR` environment variable, or in the `vi` editor if the `EDITOR` variable is not set, to allow you to edit the quota limits.

```
# edquota -u test
```

```
Quotas for user test:
```

```
/usr: blocks in use: 65, limits (soft = 50, hard = 75)
      inodes in use: 7, limits (soft = 50, hard = 60)
/usr/var: blocks in use: 0, limits (soft = 50, hard = 75)
          inodes in use: 0, limits (soft = 50, hard = 60)
```

You will normally see two lines for each file system that has quotas enabled. One line for the block limits, and one line for inode limits. Simply change the value you want updated to modify the quota limit. For example, to raise this users block limit from a soft limit of 50 and a hard limit of 75 to a soft limit of 500 and a hard limit of 600, change:

```
/usr: blocks in use: 65, limits (soft = 50, hard =
75)
```

to:

```
/usr: blocks in use: 65,
limits (soft = 500, hard = 600)
```

The new quota limits will be in place when you exit the editor.

Sometimes it is desirable to set quota limits on a range of uids. This can be done by use of the `-p` option on the `edquota` command. First, assign the desired quota limit to a user, and then run `edquota -p protouser startuid-enduid`. For example, if user `test` has the desired quota limits, the following command can be used to duplicate those quota limits for uids 10,000 through 19,999:

```
# edquota -p test 10000-19999
```

The ability to specify uid ranges was added to the system after 2.1 was released. If you need this feature on a 2.1 system, you will need to obtain a newer copy of `edquota`.

See `man edquota` for more detailed information.

## Checking Quota Limits and Disk Usage

You can use either the `quota` or the `repquota` commands to check quota limits and disk usage. The `quota` command can be used to check individual user and group quotas and disk usage. Only the

super-user may examine quotas and usage for other users, or for groups that they are not a member of. The `repquota` command can be used to get a summary of all quotas and disk usage for file systems with quotas enabled.

The following is some sample output from the `quota -v` command for a user that has quota limits on two file systems.

Disk quotas for user test (uid 1002):

```
Filesys-
tem blocks  quota  limit  grace  files  quota  limit  grace
      /usr    65*   50    75    5days  7     50    60
      /usr/var  0     50    75                0     50    60
```

On the `/usr` file system in the above example this user is currently 15 blocks over their soft limit of 50 blocks and has 5 days of their grace period left. Note the asterisk `*` which indicates that the user is currently over their quota limit.

Normally file systems that the user is not using any disk space on will not show up in the output from the `quota` command, even if they have a quota limit assigned for that file system. The `-v` option will display those file systems, such as the `/usr/var` file system in the above example.

## \* Quotas over NFS

This section is still under development.

# Chapter 11. The X Window System

Pending the completion of this section, please refer to documentation supplied by the The XFree86 Project, Inc (<http://www.xfree86.org/>).

## Chapter 12. PC Hardware compatibility

Issues of hardware compatibility are among the most troublesome in the computer industry today and FreeBSD is by no means immune to trouble. In this respect, FreeBSD's advantage of being able to run on inexpensive commodity PC hardware is also its liability when it comes to support for the amazing variety of components on the market. While it would be impossible to provide an exhaustive listing of hardware that FreeBSD supports, this section serves as a catalog of the device drivers included with FreeBSD and the hardware each driver supports. Where possible and appropriate, notes about specific products are included. You may also want to refer to the kernel configuration file section in this handbook for a list of supported devices.

As FreeBSD is a volunteer project without a funded testing department, we depend on you, the user, for much of the information contained in this catalog. If you have direct experience of hardware that does or does not work with FreeBSD, please let us know by sending e-mail to the FreeBSD documentation project mailing list <freebsd-doc@FreeBSD.ORG>. Questions about supported hardware should be directed to the FreeBSD general questions mailing list <freebsd-questions@FreeBSD.ORG> (see Mailing Lists for more information). When submitting information or asking a question, please remember to specify exactly what version of FreeBSD you are using and include as many details of your hardware as possible.

### Resources on the Internet

The following links have proven useful in selecting hardware. Though some of what you see won't necessarily be specific (or even applicable) to FreeBSD, most of the hardware information out there is OS independent. Please check with the FreeBSD hardware guide to make sure that your chosen configuration is supported before making any purchases.

- The Pentium Systems Hardware Performance Guide (<http://www.tomshardware.com/>)

### Sample Configurations

The following list of sample hardware configurations by no means constitutes an endorsement of a given hardware vendor or product by *The FreeBSD Project*. This information is provided only as a public service and merely catalogs some of the experiences that various individuals have had with different hardware combinations. Your mileage may vary. Slippery when wet. Beware of dog.

## Jordan's Picks

I have had fairly good luck building workstation and server configurations with the following components. I can't guarantee that you will too, nor that any of the companies here will remain "best buys" forever. I will try, when I can, to keep this list up-to-date but cannot obviously guarantee that it will be at any given time.

### Motherboards

For Pentium Pro (P6) systems, I'm quite fond of the Tyan (<http://www.tyan.com/html/products.html>) S1668 dual-processor motherboard as well as the Intel PR440FX motherboard with on-board SCSI WIDE and 100/10MB Intel Etherexpress NIC. You can build a dandy little single or dual processor system (which is supported in FreeBSD 3.0) for very little cost now that the Pentium Pro 180/256K chips have fallen so greatly in price, but no telling how much longer this will last.

For the Pentium II, I'm rather partial to the ASUS (<http://www.asus.com.tw>) P2I97-S (<http://www.asus.com.tw/Products/Motherboard/Pentiumpro/P2I97-s/index.html>) motherboard with the on-board Adaptec SCSI WIDE controller.

For Pentium machines, the ASUS P55T2P4 (<http://www.asus.com.tw/Products/Motherboard/Pentium/P55tp4/index.html>) motherboard appears to be a good choice for mid-to-high range Pentium server and workstation systems.

Those wishing to build more fault-tolerant systems should also be sure to use Parity memory or, for truly 24/7 applications, ECC memory.

**Note:** ECC memory does involve a slight performance trade-off (which may or may not be noticeable depending on your application) but buys you significantly increased fault-tolerance to memory errors.

### Disk Controllers

This one is a bit trickier, and while I used to recommend the Buslogic (<http://www.buslogic.com>) controllers unilaterally for everything from ISA to PCI, now I tend to lean towards the Adaptec (<http://www.adaptec.com>) 1542CF for ISA, Buslogic Bt747c for EISA and Adaptec 2940UW for PCI.

The NCR/Symbios cards for PCI have also worked well for me, though you need to make sure that your motherboard supports the BIOS-less model if you're using one of those (if your card has nothing which looks even vaguely like a ROM chip on it, you've probably got one which expects its BIOS to be on your motherboard).

If you should find that you need more than one SCSI controller in a PCI machine, you may wish to consider conserving your scarce PCI bus resources by buying the Adaptec 3940 card, which puts two

SCSI controllers (and internal busses) in a single slot.

**Note:** There are two types of 3940 on the market—the older model with AIC 7880 chips on it, and the newer one with AIC 7895 chips. The newer model requires CAM (<http://www.freebsd.org/pub/FreeBSD/development/cam/>) support which is not yet part of FreeBSD—you have to add it, or install from one of the CAM binary snapshot releases.

## Disk drives

In this particular game of Russian roulette, I'll make few specific recommendations except to say "SCSI over IDE whenever you can afford it." Even in small desktop configurations, SCSI often makes more sense since it allows you to easily migrate drives from server to desktop as falling drive prices make it economical to do so. If you have more than one machine to administer then think of it not simply as storage, think of it as a food chain! For a serious server configuration, there's not even any argument—use SCSI equipment and good cables.

## CDROM drives

My SCSI preferences extend to SCSI CDROM drives as well, and while the Toshiba (<http://www.toshiba.com>) drives have always been favourites of mine (in whatever speed is hot that week), I'm still fond of my good old Plextor (<http://www.plextor.com>) PX-12CS drive. It's only a 12 speed, but it's offered excellent performance and reliability.

Generally speaking, most SCSI CDROM drives I've seen have been of pretty solid construction and you probably won't go wrong with an HP or NEC SCSI CDROM drive either. SCSI CDROM prices also appear to have dropped considerably in the last few months and are now quite competitive with IDE CDROMs while remaining a technically superior solution. I now see no reason whatsoever to settle for an IDE CDROM drive if given a choice between the two.

## CD Recordable (WORM) drives

At the time of this writing, FreeBSD supports 3 types of CDR drives (though I believe they all ultimately come from Phillips anyway): The Phillips CDD 522 (Acts like a Plasmon), the PLASMON RF4100 and the HP 6020i. I myself use the HP 6020i for burning CDROMs (in 2.2 and later releases—it does not work with earlier releases of the SCSI code) and it works very well. See `/usr/share/examples/worm` (`file:/usr/share/examples/worm`) on your 2.2 system for example scripts used to create ISO9660 filesystem images (with RockRidge extensions) and burn them onto an HP6020i CDR.

## Tape drives

I've had pretty good luck with both 8mm drives (<http://www.Exabyte.COM:80/Products/8mm/8505XL/Rfeatures.html>) from Exabyte (<http://www.exabyte.com>) and 4mm (DAT) ([http://www-dmo.external.hp.com:80/tape/\\_cpb0001.htm](http://www-dmo.external.hp.com:80/tape/_cpb0001.htm)) drives from HP (<http://www.hp.com>).

For backup purposes, I'd have to give the higher recommendation to the Exabyte due to the more robust nature (and higher storage capacity) of 8mm tape.

## Video Cards

If you can also afford to buy a commercial X server for US\$99 from Xi Graphics, Inc. (formerly X Inside, Inc) (<http://www.xig.com/>) then I can heartily recommend the Matrox (<http://www.matrox.com/>) Millennium II (<http://www.matrox.com/mgaweb/brochure.htm>) card. Note that support for this card is also excellent with the XFree86 (<http://www.xfree86.org/>) server, which is now at version 3.3.2.

You also certainly can't go wrong with one of Number 9's (<http://www.nine.com/>) cards — their S3 Vision 868 and 968 based cards (the 9FX series) also being quite fast and very well supported by XFree86's S3 server. You can also pick up their Revolution 3D cards very cheaply these days, especially if you require a lot of video memory.

## Monitors

I have had very good luck with the Sony Multiscan 17seII monitors (<http://cons3.sel.sony.com/SEL/ccpg/display/ms17se2.html>), as have I with the Viewsonic offering in the same (Trinitron) tube. For larger than 17", all I can recommend at the time of this writing is to not spend any less than U.S. \$2,000 for a 21" monitor or \$1,700 for a 20" monitor if that's what you really need. There are good monitors available in the  $\geq 20$ " range and there are also cheap monitors in the  $\geq 20$ " range. Unfortunately, very few are both cheap and good!

## Networking

I can recommend the Intel EtherExpress Pro/100B card first and foremost, followed by the SMC (<http://www.smc.com/>) Ultra 16 controller for any ISA application and the SMC EtherPower or Compex ENET32 cards for slightly cheaper PCI based networking. In general, any PCI NIC based around DEC's DC21041 Ethernet controller chip, such as the Zynx ZX342 or DEC DE435, will generally work quite well and can frequently be found in 2-port and 4-port version (useful for firewalls and routers), though the Pro/100MB card has the edge when it comes to providing the best performance with the lower overhead.

If what you're looking for is the cheapest possible solution then almost any NE2000 clone will do a fine job for very little cost.

## Serial

If you're looking for high-speed serial networking solutions, then Digi International (<http://www.dgii.com/>) makes the SYNC/570 (<http://www.dgii.com/prodprofiles/profiles-prices/digiprofiles/digispecs/sync570.html>) series, with drivers now in FreeBSD-current. Emerging Technologies (<http://www.etinc.com>) also manufactures a board with T1/E1 capabilities, using software they provide. I have no direct experience using either product, however.

Multiport card options are somewhat more numerous, though it has to be said that FreeBSD's support for Cyclades (<http://www.cyclades.com/>)'s products is probably the tightest, primarily as a result of that company's commitment to making sure that we are adequately supplied with evaluation boards and technical specs. I've heard that the Cyclom-16Ye offers the best price/performance, though I've not checked the prices lately. Other multiport cards I've heard good things about are the BOCA and AST cards, and Stallion Technologies (<http://www.stallion.com/>) apparently offers an unofficial driver for their cards at this (<ftp://ftp.stallion.com/drivers/unsupported/freebsd/stalbsd-0.0.4.tar.gz>) location.

## Audio

I currently use a Creative Labs (<http://www.creaf.com/>) AWE32 though just about anything from Creative Labs will generally work these days. This is not to say that other types of sound cards don't also work, simply that I have little experience with them (I was a former GUS fan, but Gravis's soundcard situation has been dire for some time).

## Video

For video capture, there are two good choices — any card based on the Brooktree BT848 chip, such as the Hauppauge or WinTV boards, will work very nicely with FreeBSD. Another board which works for me is the Matrox (<http://www.matrox.com/>) Meteor (<http://www.matrox.com/imgweb/meteor.htm>) card. FreeBSD also supports the older video spigot card from Creative Labs, but those are getting somewhat difficult to find. Note that the Meteor frame grabber card *will not work* with motherboards based on the 440FX chipset! See the motherboard reference section for details. In such cases, it's better to go with a BT848 based board.

## Core/Processing

### Motherboards, busses, and chipsets

\* ISA

\* EISA

\* VLB

#### PCI

*Contributed by David O'Brien <obrien@FreeBSD.ORG> from postings by Rodney Grimes <rgrimes@FreeBSD.ORG>. 25 April 1995.*

*Continuing updates by Jordan K. Hubbard <jkh@FreeBSD.ORG>. Last update on 26 August 1996.*

Of the Intel PCI chip sets, the following list describes various types of known-brokenness and the degree of breakage, listed from worst to best.

Mercury:

Cache coherency problems, especially if there are ISA bus masters behind the ISA to PCI bridge chip. Hardware flaw, only known work around is to turn the cache off.

Saturn-I (*ie, 82424ZX at rev 0, 1 or 2*):

Write back cache coherency problems. Hardware flaw, only known work around is to set the external cache to write-through mode. Upgrade to Saturn-II.

Saturn-II (*ie, 82424ZX at rev 3 or 4*):

Works fine, but many MB manufactures leave out the external dirty bit SRAM needed for write back operation. Work arounds are either run it in write through mode, or get the dirty bit SRAM installed. (I have these for the ASUS PCI/I-486SP3G rev 1.6 and later boards).

Neptune:

Can not run more than 2 bus master devices. Admitted Intel design flaw. Workarounds include do not run more than 2 bus masters, special hardware design to replace the PCI bus arbiter (appears on Intel Altair board and several other Intel server group MB's). And of course Intel's official answer, move to the Triton chip set, we "fixed it there".

Triton (*ie, 430FX*):

No known cache coherency or bus master problems, chip set does not implement parity checking. Workaround for parity issue. Use Triton-II based motherboards if you have the choice.

Triton-II (*ie, 430HX*):

All reports on motherboards using this chipset have been favorable so far. No known problems.

Orion:

Early versions of this chipset suffered from a PCI write-posting bug which can cause noticeable performance degradation in applications where large amounts of PCI bus traffic is involved. B0 stepping or later revisions of the chipset fixed this problem.

440FX (<http://developer.intel.com/design/pcisets/desktop.htm#440FX>):

This Pentium Pro (<http://www.intel.com/procs/ppro/index.htm>) support chipset seems to work well, and does not suffer from any of the early Orion chipset problems. It also supports a wider variety of memory, including ECC and parity. The only known problem with it is that the Matrox Meteor frame grabber card doesn't like it.

## CPUs/FPUs

*Contributed by Satoshi Asami <asami@FreeBSD.ORG>. 26 December 1997.*

### **P6 class (Pentium Pro/Pentium II)**

Both the Pentium Pro and Pentium II work fine with FreeBSD. In fact, our main ftp site [ftp.freebsd.org](ftp://ftp.freebsd.org) (<ftp://ftp.freebsd.org/>) (also known as "[ftp.cdrom.com](ftp://ftp.cdrom.com)", world's largest ftp site) runs FreeBSD on a Pentium Pro. Configurations details (<ftp://ftp.cdrom.com/archive-info/wcarchive.txt>) are available for interested parties.

## Pentium class

The Intel Pentium (P54C), Pentium MMX (P55C), AMD K6 and Cyrix/IBM 6x86MX processors are all reported to work with FreeBSD. I will not go into details of which processor is faster than what, there are zillions of web sites on the Internet that tells you one way or another. :)

**Note:** Various CPUs have different voltage/cooling requirements. Make sure your motherboard can supply the exact voltage needed by the CPU. For instance, many recent MMX chips require split voltage (e.g., 2.9V core, 3.3V I/O). Also, some AMD and Cyrix/IBM chips run hotter than Intel chips. In that case, make sure you have good heatsink/fans (you can get the list of certified parts from their web pages).

## Clock speeds

*Contributed by Rodney Grimes <rgrimes@FreeBSD.ORG>. 1 October 1996.*

*Updated by Satoshi Asami <asami@FreeBSD.ORG>. 27 December 1997.*

Pentium class machines use different clock speeds for the various parts of the system. These being the speed of the CPU, external memory bus, and the PCI bus. It is not always true that a “faster” processor will make a system faster than a “slower” one, due to the various clock speeds used. Below is a table showing the differences:

Rated CPU MHz	External Clock and Memory Bus MHz	External to Internal Clock Multiplier	PCI Bus Clock MHz
60	60	1.0	30
66	66	1.0	33
75	50	1.5	25
90	60	1.5	30
100	50	2	25
100	66	1.5	33
120	60	2	30
133	66	2	33
150	60	2.5	30 (Intel, AMD)
150	75	2	37.5 (Cyrix/IBM 6x86MX)
166	66	2.5	33
180	60	3	30
200	66	3	33

233

66

3.5

33

**Note:** 66MHz may actually be 66.667MHz, but don't assume so.

The Pentium 100 can be run at either 50MHz external clock with a multiplier of 2 or at 66MHz and a multiplier of 1.5.

As can be seen the best parts to be using are the 100, 133, 166, 200 and 233, with the exception that at a multiplier of 3 or more the CPU starves for memory.

### **The AMD K6 Bug**

In 1997, there have been reports of the AMD K6 seg faulting during heavy compilation. That problem has been fixed in 3Q '97. According to reports, K6 chips with date mark "9733" or larger (i.e., manufactured in the 33rd week of '97 or later) do not have this bug.

### **\* 486 class**

### **\* 386 class**

### **286 class**

Sorry, FreeBSD does not run on 80286 machines. It is nearly impossible to run today's large full-featured UNIXes on such hardware.

## **\* Memory**

The minimum amount of memory you must have to install FreeBSD is 5 MB. Once your system is up and running you can build a custom kernel that will use less memory. If you use the `boot4.flp` you can get away with having only 4 MB.

\* BIOS

## Input/Output Devices

\* Video cards

\* Sound cards

## Serial ports and multiport cards

### The UART: What it is and how it works

*Copyright © 1996 Frank Durda IV <uhc1em@FreeBSD.ORG>, 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

### **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 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.

### **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.

### **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 asserting or deasserting one more more of these extra signals. The function of each of these additional signals is defined in the EIA RS232-C standard.

### **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.

<b>DB25 RS232-C Pin</b>	<b>DB9 IBM PC Pin</b>	<b>EIA Circuit Symbol</b>	<b>CCITT Circuit Symbol</b>	<b>Common Name</b>	<b>Signal Source</b>	<b>Description</b>
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

### 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.

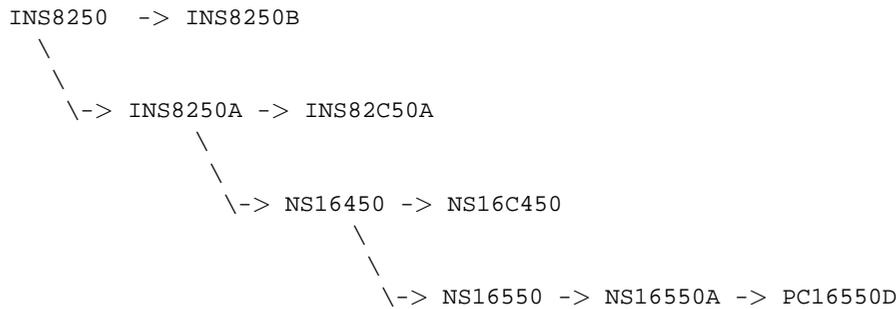
Modern high speed modems (2400, 9600, 14,400, and 19,200bps) in reality still operate at or below 2400 baud, or more accurately, 2400 Symbols per second. High speed modem are able to encode more bits of data into each Symbol using a technique called Constellation Stuffing, which is why the effective 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.

## 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 NS $nnnnnrqp$  part numbers are now of the format PC $nnnnnrqp$ .

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  $\Gamma$  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.

### **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 MSD utility as the primary tests for compatibility with the NS16550A. This over-simplistic 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
```

### 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.

I/O Port	Access Allowed	Description
+0x00	write (DLAB==0)	Transmit Holding Register (THR). Information written to this port are treated as data words and will be transmitted by the UART.
+0x01	read (DLAB==0)	Receive Buffer Register (RBR). Any data words received by the UART form the serial link are accessed by the host by reading this port.

+0x00	write/read (DLAB==1)	Divisor Latch LSB (DLL) This value will be divided from the master input clock (in the IBM PC, the master clock is 1.8432MHz) and the resulting clock will determine the baud rate of the UART. This register holds bits 0 thru 7 of the divisor.
+0x01	write/read (DLAB==1)	Divisor Latch MSB (DLH) This value will be divided from the master input clock (in the IBM PC, the master clock is 1.8432MHz) and the resulting clock will determine the baud rate of the UART. This register holds bits 8 thru 15 of the divisor.
+0x01	write/read (DLAB==0)	ENTRYTBL not supported.
+0x02	write	ENTRYTBL not supported.
+0x02	read	ENTRYTBL not supported.
+0x03	write/read	ENTRYTBL not supported.
+0x04	write/read	ENTRYTBL not supported.
+0x05	write/read	ENTRYTBL not supported.
+0x06	write/read	ENTRYTBL not supported.
+0x07	write/read	Scratch Register (SCR). This register performs no function in the UART. Any value can be written by the host to this location and read by the host later on.

### 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.

## 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 `sio(4)` manual page for detailed technical documentation.

### Digi International (DigiBoard) PC/8

*Contributed by Andrew Webster <[awebster@pubnix.net](mailto:awebster@pubnix.net)>. 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_MULTIPORT` or it will not work very well!

```
device      sio4      at isa? port 0x100 tty flags 0xb05
device      sio5      at isa? port 0x108 tty flags 0xb05
device      sio6      at isa? port 0x110 tty flags 0xb05
device      sio7      at isa? port 0x118 tty flags 0xb05
```

```

device      sio8    at isa? port 0x120 tty flags 0xb05
device      sio9    at isa? port 0x128 tty flags 0xb05
device      sio10   at isa? port 0x130 tty flags 0xb05
device      sio11   at isa? port 0x138 tty flags 0xb05 irq 9 vec-
tor siointr

```

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.

## Boca 16

*Contributed by Don Whiteside <whiteside@acm.org>. 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 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.

1. Add the line

```
options COM_MULTIPORT
```

to the config file.

2. Where the current `device sion` lines are, you will need to add 16 more devices. Only the last device includes the interrupt vector for the board. (See the `sio(4)` manual page for detail as to why.) 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 tty flags 0x1005
device sio2 at isa? port 0x108 tty flags 0x1005
device sio3 at isa? port 0x110 tty flags 0x1005

```

```

device sio4 at isa? port 0x118 tty flags 0x1005
...
device sio15 at isa? port 0x170 tty flags 0x1005
device sio16 at isa? port 0x178 tty flags 0x1005 irq 3 vector siointr

```

The flags entry *must* be changed from this example unless you are using the exact same sio assignments. Flags are set according to 0xMY<sub>Y</sub> 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.

3. 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

```

```

siol3: type 16550A (multiport)
siol4 at 0x168-0x16f flags 0x1005 on isa
siol4: type 16550A (multiport)
siol5 at 0x170-0x177 flags 0x1005 on isa
siol5: type 16550A (multiport)
siol6 at 0x178-0x17f irq 3 flags 0x1005 on isa
siol6: type 16550A (multiport master)

```

If the messages go by too fast to see,

```
# dmesg | more
```

will show you the boot messages.

4. Next, appropriate entries in `/dev` for the devices must be made using the `/dev/MAKEDEV` script. After becoming root:

```

# cd /dev
# ./MAKEDEV ttyl
# ./MAKEDEV cua1
(everything in between)
# ./MAKEDEV ttyg
# ./MAKEDEV cuag

```

If you do not want or need callout devices for some reason, you can dispense with making the `cua*` devices.

5. 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)

```
# echo at > ttyd*
```

for each device you have made. You *should* see the RX lights flash for each working port.

## Configuring the `cy` driver

*Contributed by Alex Nash <alex@freebsd.org>. 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:

1. Add the `cy` device to your kernel configuration (note that your `irq` and `iomem` settings may differ).

```
device cy0 at isa? tty irq 10 iomem 0xd4000 iosiz 0x2000 vector cyintr
```
2. Rebuild and install the new kernel.
3. Make the device nodes by typing (the following example assumes an 8-port board):

```
# cd /dev
# for i in 0 1 2 3 4 5 6 7;do ./MAKEDEV cuac$i ttyc$i;done
```

4. 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
```

5. Reboot with the new kernel.

## \* Parallel ports

## \* Modems

## \* Network cards

## \* Keyboards

## \* Mice

## \* Other

## Storage Devices

### Using ESDI hard disks

*Copyright © 1995, Wilko Bulte <wilko@yedi.iaf.nl>. 24 September 1995.*

ESDI is an acronym that means Enhanced Small Device Interface. It is loosely based on the good old ST506/412 interface originally devised by Seagate Technology, the makers of the first affordable 5.25" winchester disk.

The acronym says Enhanced, and rightly so. In the first place the speed of the interface is higher, 10 or 15 Mbits/second instead of the 5 Mbits/second of ST412 interfaced drives. Secondly some higher level commands are added, making the ESDI interface somewhat 'smarter' to the operating system driver writers. It is by no means as smart as SCSI by the way. ESDI is standardized by ANSI.

Capacities of the drives are boosted by putting more sectors on each track. Typical is 35 sectors per track, high capacity drives I have seen were up to 54 sectors/track.

Although ESDI has been largely obsoleted by IDE and SCSI interfaces, the availability of free or cheap surplus drives makes them ideal for low (or now) budget systems.

### Concepts of ESDI

#### Physical connections

The ESDI interface uses two cables connected to each drive. One cable is a 34 pin flat cable edge connector that carries the command and status signals from the controller to the drive and vice-versa. The command cable is daisy chained between all the drives. So, it forms a bus onto which all drives are connected.

The second cable is a 20 pin flat cable edge connector that carries the data to and from the drive. This cable is radially connected, so each drive has its own direct connection to the controller.

To the best of my knowledge PC ESDI controllers are limited to using a maximum of 2 drives per controller. This is compatibility feature(?) left over from the WD1003 standard that reserves only a single bit for device addressing.

#### Device addressing

On each command cable a maximum of 7 devices and 1 controller can be present. To enable the controller to uniquely identify which drive it addresses, each ESDI device is equipped with jumpers or switches to select the devices address.

On PC type controllers the first drive is set to address 0, the second disk to address 1. *Always make sure* you set each disk to an unique address! So, on a PC with its two drives/controller maximum the first drive is drive 0, the second is drive 1.

### Termination

The daisy chained command cable (the 34 pin cable remember?) needs to be terminated at the last drive on the chain. For this purpose ESDI drives come with a termination resistor network that can be removed or disabled by a jumper when it is not used.

So, one and *only* one drive, the one at the farthest end of the command cable has its terminator installed/enabled. The controller automatically terminates the other end of the cable. Please note that this implies that the controller must be at one end of the cable and *not* in the middle.

### Using ESDI disks with FreeBSD

Why is ESDI such a pain to get working in the first place?

People who tried ESDI disks with FreeBSD are known to have developed a profound sense of frustration. A combination of factors works against you to produce effects that are hard to understand when you have never seen them before.

This has also led to the popular legend ESDI and FreeBSD is a plain NO-GO. The following sections try to list all the pitfalls and solutions.

### ESDI speed variants

As briefly mentioned before, ESDI comes in two speed flavors. The older drives and controllers use a 10 Mbits/second data transfer rate. Newer stuff uses 15 Mbits/second.

It is not hard to imagine that 15 Mbits/second drive cause problems on controllers laid out for 10 Mbits/second. As always, consult your controller *and* drive documentation to see if things match.

### Stay on track

Mainstream ESDI drives use 34 to 36 sectors per track. Most (older) controllers cannot handle more than this number of sectors. Newer, higher capacity, drives use higher numbers of sectors per track. For instance, I own a 670 Mb drive that has 54 sectors per track.

In my case, the controller could not handle this number of sectors. It proved to work well except that it only used 35 sectors on each track. This meant losing a lot of disk space.

Once again, check the documentation of your hardware for more info. Going out-of-spec like in the example might or might not work. Give it a try or get another more capable controller.

### Hard or soft sectoring

Most ESDI drives allow hard or soft sectoring to be selected using a jumper. Hard sectoring means that the drive will produce a sector pulse on the start of each new sector. The controller uses this pulse to tell when it should start to write or read.

Hard sectoring allows a selection of sector size (normally 256, 512 or 1024 bytes per formatted sector). FreeBSD uses 512 byte sectors. The number of sectors per track also varies while still using the same number of bytes per formatted sector. The number of *unformatted* bytes per sector varies, dependent on your controller it needs more or less overhead bytes to work correctly. Pushing more sectors on a track of course gives you more usable space, but might give problems if your controller needs more bytes than the drive offers.

In case of soft sectoring, the controller itself determines where to start/stop reading or writing. For ESDI hard sectoring is the default (at least on everything I came across). I never felt the urge to try soft sectoring.

In general, experiment with sector settings before you install FreeBSD because you need to re-run the low-level format after each change.

### Low level formatting

ESDI drives need to be low level formatted before they are usable. A reformat is needed whenever you fiddle with the number of sectors/track jumpers or the physical orientation of the drive (horizontal, vertical). So, first think, then format. The format time must not be underestimated, for big disks it can take hours.

After a low level format, a surface scan is done to find and flag bad sectors. Most disks have a manufacturer bad block list listed on a piece of paper or adhesive sticker. In addition, on most disks the list is also written onto the disk. Please use the manufacturer's list. It is much easier to remap a defect now than after FreeBSD is installed.

Stay away from low-level formatters that mark all sectors of a track as bad as soon as they find one bad sector. Not only does this waste space, it also and more importantly causes you grief with bad144 (see the section on bad144).

### Translations

Translations, although not exclusively a ESDI-only problem, might give you real trouble. Translations come in multiple flavors. Most of them have in common that they attempt to work around the limitations

posed upon disk geometries by the original IBM PC/AT design (thanks IBM!).

First of all there is the (in)famous 1024 cylinder limit. For a system to be able to boot, the stuff (whatever operating system) must be in the first 1024 cylinders of a disk. Only 10 bits are available to encode the cylinder number. For the number of sectors the limit is 64 (0-63). When you combine the 1024 cylinder limit with the 16 head limit (also a design feature) you max out at fairly limited disk sizes.

To work around this problem, the manufacturers of ESDI PC controllers added a BIOS prom extension on their boards. This BIOS extension handles disk I/O for booting (and for some operating systems *all* disk I/O) by using translation. For instance, a big drive might be presented to the system as having 32 heads and 64 sectors/track. The result is that the number of cylinders is reduced to something below 1024 and is therefore usable by the system without problems. It is noteworthy to know that FreeBSD does not use the BIOS after its kernel has started. More on this later.

A second reason for translations is the fact that most older system BIOSes could only handle drives with 17 sectors per track (the old ST412 standard). Newer system BIOSes usually have a user-defined drive type (in most cases this is drive type 47).

**Warning:** Whatever you do to translations after reading this document, keep in mind that if you have multiple operating systems on the same disk, all must use the same translation

While on the subject of translations, I have seen one controller type (but there are probably more like this) offer the option to logically split a drive in multiple partitions as a BIOS option. I had select 1 drive == 1 partition because this controller wrote this info onto the disk. On power-up it read the info and presented itself to the system based on the info from the disk.

### Spare sectoring

Most ESDI controllers offer the possibility to remap bad sectors. During/after the low-level format of the disk bad sectors are marked as such, and a replacement sector is put in place (logically of course) of the bad one.

In most cases the remapping is done by using N-1 sectors on each track for actual data storage, and sector N itself is the spare sector. N is the total number of sectors physically available on the track. The idea behind this is that the operating system sees a 'perfect' disk without bad sectors. In the case of FreeBSD this concept is not usable.

The problem is that the translation from *bad* to *good* is performed by the BIOS of the ESDI controller. FreeBSD, being a true 32 bit operating system, does not use the BIOS after it has been booted. Instead, it has device drivers that talk directly to the hardware.

*So: don't use spare sectoring, bad block remapping or whatever it may be called by the controller manufacturer when you want to use the disk for FreeBSD.*

## Bad block handling

The preceding section leaves us with a problem. The controller's bad block handling is not usable and still FreeBSD's filesystems assume perfect media without any flaws. To solve this problem, FreeBSD use the `bad144` tool. Bad144 (named after a Digital Equipment standard for bad block handling) scans a FreeBSD slice for bad blocks. Having found these bad blocks, it writes a table with the offending block numbers to the end of the FreeBSD slice.

When the disk is in operation, the disk accesses are checked against the table read from the disk. Whenever a block number is requested that is in the `bad144` list, a replacement block (also from the end of the FreeBSD slice) is used. In this way, the `bad144` replacement scheme presents 'perfect' media to the FreeBSD filesystems.

There are a number of potential pitfalls associated with the use of `bad144`. First of all, the slice cannot have more than 126 bad sectors. If your drive has a high number of bad sectors, you might need to divide it into multiple FreeBSD slices each containing less than 126 bad sectors. Stay away from low-level format programs that mark *every* sector of a track as bad when they find a flaw on the track. As you can imagine, the 126 limit is quickly reached when the low-level format is done this way.

Second, if the slice contains the root filesystem, the slice should be within the 1024 cylinder BIOS limit. During the boot process the `bad144` list is read using the BIOS and this only succeeds when the list is within the 1024 cylinder limit.

**Note:** The restriction is not that only the root *filesystem* must be within the 1024 cylinder limit, but rather the entire *slice* that contains the root filesystem.

## Kernel configuration

ESDI disks are handled by the same `wddriver` as IDE and ST412 MFM disks. The `wd` driver should work for all WD1003 compatible interfaces.

Most hardware is jumperable for one of two different I/O address ranges and IRQ lines. This allows you to have two `wd` type controllers in one system.

When your hardware allows non-standard strappings, you can use these with FreeBSD as long as you enter the correct info into the kernel config file. An example from the kernel config file (they live in `/sys/i386/conf` BTW).

```
# First WD compatible controller
controller      wdc0      at isa? port "IO_WD1" bio irq 14 vector wdintr
disk            wd0        at wdc0 drive 0
disk            wd1        at wdc0 drive 1
# Second WD compatible controller
controller      wdc1      at isa? port "IO_WD2" bio irq 15 vector wdintr
```

```
disk          wd2      at wdcl drive 0
disk          wd3      at wdcl drive 1
```

## Particulars on ESDI hardware

### Adaptec 2320 controllers

I successfully installed FreeBSD onto a ESDI disk controlled by a ACB-2320. No other operating system was present on the disk.

To do so I low level formatted the disk using `NEFMT.EXE` (ftpable from `www.adaptec.com`) and answered NO to the question whether the disk should be formatted with a spare sector on each track. The BIOS on the ACD-2320 was disabled. I used the `free configurable` option in the system BIOS to allow the BIOS to boot it.

Before using `NEFMT.EXE` I tried to format the disk using the ACB-2320 BIOS builtin formatter. This proved to be a show stopper, because it did not give me an option to disable spare sectoring. With spare sectoring enabled the FreeBSD installation process broke down on the `bad144` run.

Please check carefully which ACB-232 $xy$  variant you have. The  $x$  is either 0 or 2, indicating a controller without or with a floppy controller on board.

The  $y$  is more interesting. It can either be a blank, a A-8 or a D. A blank indicates a plain 10 Mbits/second controller. An A-8 indicates a 15 Mbits/second controller capable of handling 52 sectors/track. A D means a 15 Mbits/second controller that can also handle drives with > 36 sectors/track (also 52 ?).

All variations should be capable of using 1:1 interleaving. Use 1:1, FreeBSD is fast enough to handle it.

### Western Digital WD1007 controllers

I successfully installed FreeBSD onto a ESDI disk controlled by a WD1007 controller. To be precise, it was a WD1007-WA2. Other variations of the WD1007 do exist.

To get it to work, I had to disable the sector translation and the WD1007's onboard BIOS. This implied I could not use the low-level formatter built into this BIOS. Instead, I grabbed `WDFMT.EXE` from `www.wdc.com` Running this formatted my drive just fine.

### Ultrastor U14F controllers

According to multiple reports from the net, Ultrastor ESDI boards work OK with FreeBSD. I lack any further info on particular settings.

## Further reading

If you intend to do some serious ESDI hacking, you might want to have the official standard at hand:

The latest ANSI X3T10 committee document is: Enhanced Small Device Interface (ESDI) [X3.170-1990/X3.170a-1991] [X3T10/792D Rev 11]

On Usenet the newsgroup comp.periphs (news:comp.periphs) is a noteworthy place to look for more info.

The World Wide Web (WWW) also proves to be a very handy info source: For info on Adaptec ESDI controllers see <http://www.adaptec.com/>. For info on Western Digital controllers see <http://www.wdc.com/>.

## Thanks to...

Andrew Gordon for sending me an Adaptec 2320 controller and ESDI disk for testing.

## What is SCSI?

*Copyright © 1995, Wilko Bulte <wilko@yedi.iaf.nl>. July 6, 1996.*

SCSI is an acronym for Small Computer Systems Interface. It is an ANSI standard that has become one of the leading I/O buses in the computer industry. The foundation of the SCSI standard was laid by Shugart Associates (the same guys that gave the world the first mini floppy disks) when they introduced the SASI bus (Shugart Associates Standard Interface).

After some time an industry effort was started to come to a more strict standard allowing devices from different vendors to work together. This effort was recognized in the ANSI SCSI-1 standard. The SCSI-1 standard (approx 1985) is rapidly becoming obsolete. The current standard is SCSI-2 (see Further reading), with SCSI-3 on the drawing boards.

In addition to a physical interconnection standard, SCSI defines a logical (command set) standard to which disk devices must adhere. This standard is called the Common Command Set (CCS) and was developed more or less in parallel with ANSI SCSI-1. SCSI-2 includes the (revised) CCS as part of the standard itself. The commands are dependent on the type of device at hand. It does not make much sense of course to define a Write command for a scanner.

The SCSI bus is a parallel bus, which comes in a number of variants. The oldest and most used is an 8 bit wide bus, with single-ended signals, carried on 50 wires. (If you do not know what single-ended means, do not worry, that is what this document is all about.) Modern designs also use 16 bit wide buses, with differential signals. This allows transfer speeds of 20Mbytes/second, on cables lengths of up to 25 meters. SCSI-2 allows a maximum bus width of 32 bits, using an additional cable. Quickly emerging are Ultra SCSI (also called Fast-20) and Ultra2 (also called Fast-40). Fast-20 is 20 million transfers per

second (20 Mbytes/sec on a 8 bit bus), Fast-40 is 40 million transfers per second (40 Mbytes/sec on a 8 bit bus). Most hard drives sold today are single-ended Ultra SCSI (8 or 16 bits).

Of course the SCSI bus not only has data lines, but also a number of control signals. A very elaborate protocol is part of the standard to allow multiple devices to share the bus in an efficient manner. In SCSI-2, the data is always checked using a separate parity line. In pre-SCSI-2 designs parity was optional.

In SCSI-3 even faster bus types are introduced, along with a serial SCSI busses that reduces the cabling overhead and allows a higher maximum bus length. You might see names like SSA and Fiberchannel in this context. None of the serial buses are currently in widespread use (especially not in the typical FreeBSD environment). For this reason the serial bus types are not discussed any further.

As you could have guessed from the description above, SCSI devices are intelligent. They have to be to adhere to the SCSI standard (which is over 2 inches thick BTW). So, for a hard disk drive for instance you do not specify a head/cylinder/sector to address a particular block, but simply the number of the block you want. Elaborate caching schemes, automatic bad block replacement etc are all made possible by this 'intelligent device' approach.

On a SCSI bus, each possible pair of devices can communicate. Whether their function allows this is another matter, but the standard does not restrict it. To avoid signal contention, the 2 devices have to arbitrate for the bus before using it.

The philosophy of SCSI is to have a standard that allows older-standard devices to work with newer-standard ones. So, an old SCSI-1 device should normally work on a SCSI-2 bus. I say Normally, because it is not absolutely sure that the implementation of an old device follows the (old) standard closely enough to be acceptable on a new bus. Modern devices are usually more well-behaved, because the standardization has become more strict and is better adhered to by the device manufacturers.

Generally speaking, the chances of getting a working set of devices on a single bus is better when all the devices are SCSI-2 or newer. This implies that you do not have to dump all your old stuff when you get that shiny 2GB disk: I own a system on which a pre-SCSI-1 disk, a SCSI-2 QIC tape unit, a SCSI-1 helical scan tape unit and 2 SCSI-1 disks work together quite happily. From a performance standpoint you might want to separate your older and newer (=faster) devices however.

## Components of SCSI

As said before, SCSI devices are smart. The idea is to put the knowledge about intimate hardware details onto the SCSI device itself. In this way, the host system does not have to worry about things like how many heads are hard disks has, or how many tracks there are on a specific tape device. If you are curious, the standard specifies commands with which you can query your devices on their hardware particulars. FreeBSD uses this capability during boot to check out what devices are connected and whether they need any special treatment.

The advantage of intelligent devices is obvious: the device drivers on the host can be made in a much more generic fashion, there is no longer a need to change (and qualify!) drivers for every odd new device that is introduced.

For cabling and connectors there is a golden rule: get good stuff. With bus speeds going up all the time you will save yourself a lot of grief by using good material.

So, gold plated connectors, shielded cabling, sturdy connector hoods with strain reliefs etc are the way to go. Second golden rule: do not use cables longer than necessary. I once spent 3 days hunting down a problem with a flaky machine only to discover that shortening the SCSI bus by 1 meter solved the problem. And the original bus length was well within the SCSI specification.

## SCSI bus types

From an electrical point of view, there are two incompatible bus types: single-ended and differential. This means that there are two different main groups of SCSI devices and controllers, which cannot be mixed on the same bus. It is possible however to use special converter hardware to transform a single-ended bus into a differential one (and vice versa). The differences between the bus types are explained in the next sections.

In lots of SCSI related documentation there is a sort of jargon in use to abbreviate the different bus types. A small list:

- FWD: Fast Wide Differential
- FND: Fast Narrow Differential
- SE: Single Ended
- FN: Fast Narrow
- etc.

With a minor amount of imagination one can usually imagine what is meant.

Wide is a bit ambiguous, it can indicate 16 or 32 bit buses. As far as I know, the 32 bit variant is not (yet) in use, so wide normally means 16 bit.

Fast means that the timing on the bus is somewhat different, so that on a narrow (8 bit) bus 10 Mbytes/sec are possible instead of 5 Mbytes/sec for 'slow' SCSI. As discussed before, bus speeds of 20 and 40 million transfers/second are also emerging (Fast-20 == Ultra SCSI and Fast-40 == Ultra2 SCSI).

**Note:** The data lines > 8 are only used for data transfers and device addressing. The transfers of commands and status messages etc are only performed on the lowest 8 data lines. The standard allows narrow devices to operate on a wide bus. The usable bus width is negotiated between the devices. You have to watch your device addressing closely when mixing wide and narrow.

### Single ended buses

A single-ended SCSI bus uses signals that are either 5 Volts or 0 Volts (indeed, TTL levels) and are relative to a COMMON ground reference. A single ended 8 bit SCSI bus has approximately 25 ground lines, who are all tied to a single 'rail' on all devices. A standard single ended bus has a maximum length of 6 meters. If the same bus is used with fast-SCSI devices, the maximum length allowed drops to 3 meters. Fast-SCSI means that instead of 5Mbytes/sec the bus allows 10Mbytes/sec transfers.

Fast-20 (Ultra SCSI) and Fast-40 allow for 20 and 40 million transfers/second respectively. So, F20 is 20 Mbytes/second on a 8 bit bus, 40 Mbytes/second on a 16 bit bus etc. For F20 the max bus length is 1.5 meters, for F40 it becomes 0.75 meters. Be aware that F20 is pushing the limits quite a bit, so you will quickly find out if your SCSI bus is electrically sound.

**Note:** If some devices on your bus use 'fast' to communicate your bus must adhere to the length restrictions for fast buses!

It is obvious that with the newer fast-SCSI devices the bus length can become a real bottleneck. This is why the differential SCSI bus was introduced in the SCSI-2 standard.

For connector pinning and connector types please refer to the SCSI-2 standard (see Further reading) itself, connectors etc are listed there in painstaking detail.

Beware of devices using non-standard cabling. For instance Apple uses a 25pin D-type connector (like the one on serial ports and parallel printers). Considering that the official SCSI bus needs 50 pins you can imagine the use of this connector needs some 'creative cabling'. The reduction of the number of ground wires they used is a bad idea, you better stick to 50 pins cabling in accordance with the SCSI standard. For Fast-20 and 40 do not even think about buses like this.

### Differential buses

A differential SCSI bus has a maximum length of 25 meters. Quite a difference from the 3 meters for a single-ended fast-SCSI bus. The idea behind differential signals is that each bus signal has its own return wire. So, each signal is carried on a (preferably twisted) pair of wires. The voltage difference between these two wires determines whether the signal is asserted or de-asserted. To a certain extent the voltage difference between ground and the signal wire pair is not relevant (do not try 10 kVolts though).

It is beyond the scope of this document to explain why this differential idea is so much better. Just accept that electrically seen the use of differential signals gives a much better noise margin. You will normally find differential buses in use for inter-cabinet connections. Because of the lower cost single ended is mostly used for shorter buses like inside cabinets.

There is nothing that stops you from using differential stuff with FreeBSD, as long as you use a controller that has device driver support in FreeBSD. As an example, Adaptec marketed the AHA1740 as

a single ended board, whereas the AHA1744 was differential. The software interface to the host is identical for both.

## Terminators

Terminators in SCSI terminology are resistor networks that are used to get a correct impedance matching. Impedance matching is important to get clean signals on the bus, without reflections or ringing. If you once made a long distance telephone call on a bad line you probably know what reflections are. With 20Mbytes/sec traveling over your SCSI bus, you do not want signals echoing back.

Terminators come in various incarnations, with more or less sophisticated designs. Of course, there are internal and external variants. Many SCSI devices come with a number of sockets in which a number of resistor networks can (must be!) installed. If you remove terminators from a device, carefully store them. You will need them when you ever decide to reconfigure your SCSI bus. There is enough variation in even these simple tiny things to make finding the exact replacement a frustrating business. There are also SCSI devices that have a single jumper to enable or disable a built-in terminator. There are special terminators you can stick onto a flat cable bus. Others look like external connectors, or a connector hood without a cable. So, lots of choice as you can see.

There is much debate going on if and when you should switch from simple resistor (passive) terminators to active terminators. Active terminators contain slightly more elaborate circuit to give cleaner bus signals. The general consensus seems to be that the usefulness of active termination increases when you have long buses and/or fast devices. If you ever have problems with your SCSI buses you might consider trying an active terminator. Try to borrow one first, they reputedly are quite expensive.

Please keep in mind that terminators for differential and single-ended buses are not identical. You should *not mix* the two variants.

OK, and now where should you install your terminators? This is by far the most misunderstood part of SCSI. And it is by far the simplest. The rule is: *every single line on the SCSI bus has 2 (two) terminators, one at each end of the bus.* So, two and not one or three or whatever. Do yourself a favor and stick to this rule. It will save you endless grief, because wrong termination has the potential to introduce highly mysterious bugs. (Note the “potential” here; the nastiest part is that it may or may not work.)

A common pitfall is to have an internal (flat) cable in a machine and also an external cable attached to the controller. It seems almost everybody forgets to remove the terminators from the controller. The terminator must now be on the last external device, and not on the controller! In general, every reconfiguration of a SCSI bus must pay attention to this.

**Note:** Termination is to be done on a per-line basis. This means if you have both narrow and wide buses connected to the same host adapter, you need to enable termination on the higher 8 bits of the bus on the adapter (as well as the last devices on each bus, of course).

What I did myself is remove all terminators from my SCSI devices and controllers. I own a couple of external terminators, for both the Centronics-type external cabling and for the internal flat cable connectors. This makes reconfiguration much easier.

On modern devices, sometimes integrated terminators are used. These things are special purpose integrated circuits that can be dis/en-abled with a control pin. It is not necessary to physically remove them from a device. You may find them on newer host adapters, sometimes they are software configurable, using some sort of setup tool. Some will even auto-detect the cables attached to the connectors and automatically set up the termination as necessary. At any rate, consult your documentation!

### **Terminator power**

The terminators discussed in the previous chapter need power to operate properly. On the SCSI bus, a line is dedicated to this purpose. So, simple huh?

Not so. Each device can provide its own terminator power to the terminator sockets it has on-device. But if you have external terminators, or when the device supplying the terminator power to the SCSI bus line is switched off you are in trouble.

The idea is that initiators (these are devices that initiate actions on the bus, a discussion follows) must supply terminator power. All SCSI devices are allowed (but not required) to supply terminator power.

To allow for un-powered devices on a bus, the terminator power must be supplied to the bus via a diode. This prevents the backflow of current to un-powered devices.

To prevent all kinds of nastiness, the terminator power is usually fused. As you can imagine, fuses might blow. This can, but does not have to, lead to a non functional bus. If multiple devices supply terminator power, a single blown fuse will not put you out of business. A single supplier with a blown fuse certainly will. Clever external terminators sometimes have a LED indication that shows whether terminator power is present.

In newer designs auto-restoring fuses that 'reset' themselves after some time are sometimes used.

### **Device addressing**

Because the SCSI bus is, eh, a bus there must be a way to distinguish or address the different devices connected to it.

This is done by means of the SCSI or target ID. Each device has a unique target ID. You can select the ID to which a device must respond using a set of jumpers, or a dip switch, or something similar. Some SCSI host adapters let you change the target ID from the boot menu. (Yet some others will not let you change the ID from 7.) Consult the documentation of your device for more information.

Beware of multiple devices configured to use the same ID. Chaos normally reigns in this case. A pitfall is that one of the devices sharing the same ID sometimes even manages to answer to I/O requests!

For an 8 bit bus, a maximum of 8 targets is possible. The maximum is 8 because the selection is done bitwise using the 8 data lines on the bus. For wide buses this increases to the number of data lines (usually 16).

**Note:** A narrow SCSI device can not communicate with a SCSI device with a target ID larger than 7. This means it is generally not a good idea to move your SCSI host adapter's target ID to something higher than 7 (or your CD-ROM will stop working).

The higher the SCSI target ID, the higher the priority the devices has. When it comes to arbitration between devices that want to use the bus at the same time, the device that has the highest SCSI ID will win. This also means that the SCSI host adapter usually uses target ID 7. Note however that the lower 8 IDs have higher priorities than the higher 8 IDs on a wide-SCSI bus. Thus, the order of target IDs is: [7 6 .. 1 0 15 14 .. 9 8] on a wide-SCSI system. (If you you are wondering why the lower 8 have higher priority, read the previous paragraph for a hint.)

For a further subdivision, the standard allows for Logical Units or LUNs for short. A single target ID may have multiple LUNs. For example, a tape device including a tape changer may have LUN 0 for the tape device itself, and LUN 1 for the tape changer. In this way, the host system can address each of the functional units of the tape changer as desired.

### Bus layout

SCSI buses are linear. So, not shaped like Y-junctions, star topologies, rings, cobwebs or whatever else people might want to invent. One of the most common mistakes is for people with wide-SCSI host adapters to connect devices on all three connectors (external connector, internal wide connector, internal narrow connector). Don't do that. It may appear to work if you are really lucky, but I can almost guarantee that your system will stop functioning at the most unfortunate moment (this is also known as "Murphy's law").

You might notice that the terminator issue discussed earlier becomes rather hairy if your bus is not linear. Also, if you have more connectors than devices on your internal SCSI cable, make sure you attach devices on connectors on both ends instead of using the connectors in the middle and let one or both ends dangle. This will screw up the termination of the bus.

The electrical characteristics, its noise margins and ultimately the reliability of it all are tightly related to linear bus rule.

*Stick to the linear bus rule!*

## Using SCSI with FreeBSD

### About translations, BIOSes and magic...

As stated before, you should first make sure that you have a electrically sound bus.

When you want to use a SCSI disk on your PC as boot disk, you must aware of some quirks related to PC BIOSes. The PC BIOS in its first incarnation used a low level physical interface to the hard disk. So, you had to tell the BIOS (using a setup tool or a BIOS built-in setup) how your disk physically looked like. This involved stating number of heads, number of cylinders, number of sectors per track, obscure things like precompensation and reduced write current cylinder etc.

One might be inclined to think that since SCSI disks are smart you can forget about this. Alas, the arcane setup issue is still present today. The system BIOS needs to know how to access your SCSI disk with the head/cyl/sector method in order to load the FreeBSD kernel during boot.

The SCSI host adapter or SCSI controller you have put in your AT/EISA/PCI/whatever bus to connect your disk therefore has its own on-board BIOS. During system startup, the SCSI BIOS takes over the hard disk interface routines from the system BIOS. To fool the system BIOS, the system setup is normally set to No hard disk present. Obvious, isn't it?

The SCSI BIOS itself presents to the system a so called *translated* drive. This means that a fake drive table is constructed that allows the PC to boot the drive. This translation is often (but not always) done using a pseudo drive with 64 heads and 32 sectors per track. By varying the number of cylinders, the SCSI BIOS adapts to the actual drive size. It is useful to note that  $32 * 64 / 2 =$  the size of your drive in megabytes. The division by 2 is to get from disk blocks that are normally 512 bytes in size to Kbytes.

Right. All is well now?! No, it is not. The system BIOS has another quirk you might run into. The number of cylinders of a bootable hard disk cannot be greater than 1024. Using the translation above, this is a show-stopper for disks greater than 1 GB. With disk capacities going up all the time this is causing problems.

Fortunately, the solution is simple: just use another translation, e.g. with 128 heads instead of 32. In most cases new SCSI BIOS versions are available to upgrade older SCSI host adapters. Some newer adapters have an option, in the form of a jumper or software setup selection, to switch the translation the SCSI BIOS uses.

It is very important that *all* operating systems on the disk use the *same translation* to get the right idea about where to find the relevant partitions. So, when installing FreeBSD you must answer any questions about heads/cylinders etc using the translated values your host adapter uses.

Failing to observe the translation issue might lead to un-bootable systems or operating systems overwriting each others partitions. Using fdisk you should be able to see all partitions.

You might have heard some talk of “lying” devices? Older FreeBSD kernels used to report the geometry of SCSI disks when booting. An example from one of my systems:

```
aha0 targ 0 lun 0: <MICROP 1588-15MB1057404HSP4>  
sd0: 636MB (1303250 total sec), 1632 cyl, 15 head, 53 sec, bytes/sec 512
```

Newer kernels usually do not report this information. e.g.

```
(bt0:0:0): "SEAGATE ST41651 7574" type 0 fixed SCSI 2  
sd0(bt0:0:0): Direct-Access 1350MB (2766300 512 byte sectors)
```

Why has this changed?

This info is retrieved from the SCSI disk itself. Newer disks often use a technique called zone bit recording. The idea is that on the outer cylinders of the drive there is more space so more sectors per track can be put on them. This results in disks that have more tracks on outer cylinders than on the inner cylinders and, last but not least, have more capacity. You can imagine that the value reported by the drive when inquiring about the geometry now becomes suspect at best, and nearly always misleading. When asked for a geometry, it is nearly always better to supply the geometry used by the BIOS, or *if the BIOS is never going to know about this disk*, (e.g. it is not a booting disk) to supply a fictitious geometry that is convenient.

### SCSI subsystem design

FreeBSD uses a layered SCSI subsystem. For each different controller card a device driver is written. This driver knows all the intimate details about the hardware it controls. The driver has a interface to the upper layers of the SCSI subsystem through which it receives its commands and reports back any status.

On top of the card drivers there are a number of more generic drivers for a class of devices. More specific: a driver for tape devices (abbreviation: st), magnetic disks (sd), CD-ROMs (cd) etc. In case you are wondering where you can find this stuff, it all lives in `/sys/scsi`. See the man pages in section 4 for more details.

The multi level design allows a decoupling of low-level bit banging and more high level stuff. Adding support for another piece of hardware is a much more manageable problem.

### Kernel configuration

Dependent on your hardware, the kernel configuration file must contain one or more lines describing your host adapter(s). This includes I/O addresses, interrupts etc. Consult the man page for your adapter driver to get more info. Apart from that, check out `/sys/1386/conf/LINT` for an overview of a kernel config file. `LINT` contains every possible option you can dream of. It does *not* imply `LINT` will actually get you to a working kernel at all.

Although it is probably stating the obvious: the kernel config file should reflect your actual hardware setup. So, interrupts, I/O addresses etc must match the kernel config file. During system boot messages

will be displayed to indicate whether the configured hardware was actually found.

**Note:** Note that most of the EISA/PCI drivers (namely `ahb`, `ahc`, `ncr` and `amd` will automatically obtain the correct parameters from the host adapters themselves at boot time; thus, you just need to write, for instance, `controller ahc0`.

An example loosely based on the FreeBSD 2.2.5-Release kernel config file `LINT` with some added comments (between `[]`):

```
# SCSI host adapters: 'aha', 'ahb', 'ahc', 'bt', 'nca'
#
# aha: Adaptec 154x
# ahb: Adaptec 174x
# ahc: Adaptec 274x/284x/294x
# aic: Adaptec 152x and sound cards using the Adaptec AIC-6360 (slow!)
# amd: AMD 53c974 based SCSI cards (e.g., Tekram DC-390 and 390T)
# bt: Most Buslogic controllers
# nca: ProAudioSpectrum cards using the NCR 5380 or Trantor T130
# ncr: NCR/Symbios 53c810/815/825/875 etc based SCSI cards
# uha: UltraStore 14F and 34F
# sea: Seagate ST01/02 8 bit controller (slow!)
# wds: Western Digital WD7000 controller (no scatter/gather!).
#

[For an Adaptec AHA274x/284x/294x/394x etc controller]
controller ahc0

[For an NCR/Symbios 53c875 based controller]
controller ncr0

[For an Ultrastor adapter]
controller uha0 at isa? port "IO_UHA0" bio irq ? drq 5 vector uhaintr

# Map SCSI buses to specific SCSI adapters
controller scbus0 at ahc0
controller scbus2 at ncr0
controller scbus1 at uha0

# The actual SCSI devices
disk sd0 at scbus0 target 0 unit 0 [SCSI disk 0 is at scbus 0, LUN 0]
disk sd1 at scbus0 target 1 [implicit LUN 0 if omitted]
disk sd2 at scbus1 target 3 [SCSI disk on the uha0]
disk sd3 at scbus2 target 4 [SCSI disk on the ncr0]
tape st1 at scbus0 target 6 [SCSI tape at target 6]
```

```
device cd0 at scbus?           [the first ever CD-
ROM found, no wiring]
```

The example above tells the kernel to look for a ahc (Adaptec 274x) controller, then for an NCR/Symbios board, and so on. The lines following the controller specifications tell the kernel to configure specific devices but *only* attach them when they match the target ID and LUN specified on the corresponding bus.

Wired down devices get “first shot” at the unit numbers so the first non “wired down” device, is allocated the unit number one greater than the highest “wired down” unit number for that kind of device. So, if you had a SCSI tape at target ID 2 it would be configured as st2, as the tape at target ID 6 is wired down to unit number 1.

**Note:** Wired down devices need not be found to get their unit number. The unit number for a wired down device is reserved for that device, even if it is turned off at boot time. This allows the device to be turned on and brought on-line at a later time, without rebooting. Notice that a device’s unit number has *no* relationship with its target ID on the SCSI bus.

Below is another example of a kernel config file as used by FreeBSD version < 2.0.5. The difference with the first example is that devices are not “wired down”. “Wired down” means that you specify which SCSI target belongs to which device.

A kernel built to the config file below will attach the first SCSI disk it finds to sd0, the second disk to sd1 etc. If you ever removed or added a disk, all other devices of the same type (disk in this case) would ‘move around’. This implies you have to change `/etc/fstab` each time.

Although the old style still works, you are *strongly* recommended to use this new feature. It will save you a lot of grief whenever you shift your hardware around on the SCSI buses. So, when you re-use your old trusty config file after upgrading from a pre-FreeBSD2.0.5.R system check this out.

```
[driver for Adaptec 174x]
controller      ahb0 at isa? bio irq 11 vector ahbintr

[for Adaptec 154x]
controller      aha0      at isa? port "IO_AHA0" bio irq 11 drq 5 vec-
tor ahaintr

[for Seagate ST01/02]
controller      sea0      at isa? bio irq 5 iomem 0xc8000 iosiz 0x2000 vec-
tor seaintr

controller      scbus0

device          sd0       [support for 4 SCSI harddisks, sd0 up sd3]
device          st0      [support for 2 SCSI tapes]
```

```
[for the CD-ROM]
device          cd0          #Only need one of these, the code dynamically grows
```

Both examples support SCSI disks. If during boot more devices of a specific type (e.g. sd disks) are found than are configured in the booting kernel, the system will simply allocate more devices, incrementing the unit number starting at the last number “wired down”. If there are no “wired down” devices then counting starts at unit 0.

Use `man 4 scsi` to check for the latest info on the SCSI subsystem. For more detailed info on host adapter drivers use eg `man 4 ahc` for info on the Adaptec 294x driver.

### Tuning your SCSI kernel setup

Experience has shown that some devices are slow to respond to INQUIRY commands after a SCSI bus reset (which happens at boot time). An INQUIRY command is sent by the kernel on boot to see what kind of device (disk, tape, CD-ROM etc) is connected to a specific target ID. This process is called device probing by the way.

To work around the ‘slow response’ problem, FreeBSD allows a tunable delay time before the SCSI devices are probed following a SCSI bus reset. You can set this delay time in your kernel configuration file using a line like:

```
options          SCSI_DELAY=15          #Be pessimistic about Joe SCSI device
```

This line sets the delay time to 15 seconds. On my own system I had to use 3 seconds minimum to get my trusty old CD-ROM drive to be recognized. Start with a high value (say 30 seconds or so) when you have problems with device recognition. If this helps, tune it back until it just stays working.

### Rogue SCSI devices

Although the SCSI standard tries to be complete and concise, it is a complex standard and implementing things correctly is no easy task. Some vendors do a better job than others.

This is exactly where the “rogue” devices come into view. Rogues are devices that are recognized by the FreeBSD kernel as behaving slightly (...) non-standard. Rogue devices are reported by the kernel when booting. An example for two of my cartridge tape units:

```
Feb 25 21:03:34 yedi /kernel: ahb0 targ 5 lun 0: <TANDBERG TDC 3600 -
06:>
Feb 25 21:03:34 yedi /kernel: st0: Tandberg tdc3600 is a known rogue
```

```
Mar 29 21:16:37 yedi /kernel: aha0 targ 5 lun 0: <ARCHIVE VIPER 150 21247-005>
Mar 29 21:16:37 yedi /kernel: st1: Archive Viper 150 is a known rogue
```

For instance, there are devices that respond to all LUNs on a certain target ID, even if they are actually only one device. It is easy to see that the kernel might be fooled into believing that there are 8 LUNs at that particular target ID. The confusion this causes is left as an exercise to the reader.

The SCSI subsystem of FreeBSD recognizes devices with bad habits by looking at the INQUIRY response they send when probed. Because the INQUIRY response also includes the version number of the device firmware, it is even possible that for different firmware versions different workarounds are used. See e.g. `/sys/scsi/st.c` and `/sys/scsi/scsiconf.c` for more info on how this is done.

This scheme works fine, but keep in mind that it of course only works for devices that are known to be weird. If you are the first to connect your bogus Mumbletech SCSI CD-ROM you might be the one that has to define which workaround is needed.

After you got your Mumbletech working, please send the required workaround to the FreeBSD development team for inclusion in the next release of FreeBSD. Other Mumbletech owners will be grateful to you.

### Multiple LUN devices

In some cases you come across devices that use multiple logical units (LUNs) on a single SCSI ID. In most cases FreeBSD only probes devices for LUN 0. An example are so called bridge boards that connect 2 non-SCSI harddisks to a SCSI bus (e.g. an Emulex MD21 found in old Sun systems).

This means that any devices with LUNs  $\neq 0$  are not normally found during device probe on system boot. To work around this problem you must add an appropriate entry in `/sys/scsi/scsiconf.c` and rebuild your kernel.

Look for a struct that is initialized like below:

```
{
    T_DIRECT, T_FIXED, "MAXTOR", "XT-4170S", "B5A",
    "mx1", SC_ONE_LU
}
```

For you Mumbletech BRIDGE2000 that has more than one LUN, acts as a SCSI disk and has firmware revision 123 you would add something like:

```
{
    T_DIRECT, T_FIXED, "MUMBLETECH", "BRIDGE2000", "123",
    "sd", SC_MORE_LUS
}
```

The kernel on boot scans the inquiry data it receives against the table and acts accordingly. See the source for more info.

### **Tagged command queueing**

Modern SCSI devices, particularly magnetic disks, support what is called tagged command queuing (TCQ).

In a nutshell, TCQ allows the device to have multiple I/O requests outstanding at the same time. Because the device is intelligent, it can optimise its operations (like head positioning) based on its own request queue. On SCSI devices like RAID (Redundant Array of Independent Disks) arrays the TCQ function is indispensable to take advantage of the device's inherent parallelism.

Each I/O request is uniquely identified by a "tag" (hence the name tagged command queuing) and this tag is used by FreeBSD to see which I/O in the device drivers queue is reported as complete by the device.

It should be noted however that TCQ requires device driver support and that some devices implemented it "not quite right" in their firmware. This problem bit me once, and it leads to highly mysterious problems. In such cases, try to disable TCQ.

### **Busmaster host adapters**

Most, but not all, SCSI host adapters are bus mastering controllers. This means that they can do I/O on their own without putting load onto the host CPU for data movement.

This is of course an advantage for a multitasking operating system like FreeBSD. It must be noted however that there might be some rough edges.

For instance an Adaptec 1542 controller can be set to use different transfer speeds on the host bus (ISA or AT in this case). The controller is settable to different rates because not all motherboards can handle the higher speeds. Problems like hangups, bad data etc might be the result of using a higher data transfer rate than your motherboard can stomach.

The solution is of course obvious: switch to a lower data transfer rate and try if that works better.

In the case of a Adaptec 1542, there is an option that can be put into the kernel config file to allow dynamic determination of the right, read: fastest feasible, transfer rate. This option is disabled by default:

```
options          "TUNE_1542"          #dynamic tune of bus DMA speed
```

Check the man pages for the host adapter that you use. Or better still, use the ultimate documentation (read: driver source).

## Tracking down problems

The following list is an attempt to give a guideline for the most common SCSI problems and their solutions. It is by no means complete.

- Check for loose connectors and cables.
- Check and double check the location and number of your terminators.
- Check if your bus has at least one supplier of terminator power (especially with external terminators).
- Check if no double target IDs are used.
- Check if all devices to be used are powered up.
- Make a minimal bus config with as little devices as possible.
- If possible, configure your host adapter to use slow bus speeds.
- Disable tagged command queuing to make things as simple as possible (for a NCR hostadapter based system see `man ncrcontrol`)
- If you can compile a kernel, make one with the `SCSIDEBUG` option, and try accessing the device with debugging turned on for that device. If your device does not even probe at startup, you may have to define the address of the device that is failing, and the desired debug level in `/sys/scsi/scsidebug.h`. If it probes but just does not work, you can use the `scsi(8)` command to dynamically set a debug level to it in a running kernel (if `SCSIDEBUG` is defined). This will give you *copious* debugging output with which to confuse the gurus. See `man 4 scsi` for more exact information. Also look at `man 8 scsi`.

## Further reading

If you intend to do some serious SCSI hacking, you might want to have the official standard at hand:

Approved American National Standards can be purchased from ANSI at

13th Floor  
11 West 42nd Street  
New York  
NY 10036  
Sales Dept: (212) 642-4900

You can also buy many ANSI standards and most committee draft documents from Global Engineering Documents,

15 Inverness Way East  
Englewood  
CO, 80112-5704  
Phone: (800) 854-7179  
Outside USA and Canada: (303) 792-2181  
Fax: (303) 792- 2192

Many X3T10 draft documents are available electronically on the SCSI BBS (719-574-0424) and on the `ncrinfo.ncr.com` anonymous ftp site.

Latest X3T10 committee documents are:

- AT Attachment (ATA or IDE) [X3.221-1994] (*Approved*)
- ATA Extensions (ATA-2) [X3T10/948D Rev 2i]
- Enhanced Small Device Interface (ESDI) [X3.170-1990/X3.170a-1991] (*Approved*)
- Small Computer System Interface — 2 (SCSI-2) [X3.131-1994] (*Approved*)
- SCSI-2 Common Access Method Transport and SCSI Interface Module (CAM) [X3T10/792D Rev 11]

Other publications that might provide you with additional information are:

- “SCSI: Understanding the Small Computer System Interface”, written by NCR Corporation. Available from: Prentice Hall, Englewood Cliffs, NJ, 07632 Phone: (201) 767-5937 ISBN 0-13-796855-8
- “Basics of SCSI”, a SCSI tutorial written by Ancot Corporation Contact Ancot for availability information at: Phone: (415) 322-5322 Fax: (415) 322-0455
- “SCSI Interconnection Guide Book”, an AMP publication (dated 4/93, Catalog 65237) that lists the various SCSI connectors and suggests cabling schemes. Available from AMP at (800) 522-6752 or (717) 564-0100
- “Fast Track to SCSI”, A Product Guide written by Fujitsu. Available from: Prentice Hall, Englewood Cliffs, NJ, 07632 Phone: (201) 767-5937 ISBN 0-13-307000-X
- “The SCSI Bench Reference”, “The SCSI Encyclopedia”, and the “SCSI Tutor”, ENDL Publications, 14426 Black Walnut Court, Saratoga CA, 95070 Phone: (408) 867-6642
- “Zadian SCSI Navigator” (quick ref. book) and “Discover the Power of SCSI” (First book along with a one-hour video and tutorial book), Zadian Software, Suite 214, 1210 S. Bascom Ave., San Jose, CA 92128, (408) 293-0800

On Usenet the newsgroups comp.periph.scsi (news:comp.periph.scsi) and comp.periph (news:comp.periph) are noteworthy places to look for more info. You can also find the SCSI-Faq there, which is posted periodically.

Most major SCSI device and host adapter suppliers operate ftp sites and/or BBS systems. They may be valuable sources of information about the devices you own.

## \* Disk/tape controllers

- \* SCSI

- \* IDE

- \* Floppy

## Hard drives

### SCSI hard drives

*Contributed by Satoshi Asami <asami@FreeBSD.ORG>. 17 February 1998.*

As mentioned in the SCSI section, virtually all SCSI hard drives sold today are SCSI-2 compliant and thus will work fine as long as you connect them to a supported SCSI host adapter. Most problems people encounter are either due to badly designed cabling (cable too long, star topology, etc.), insufficient termination, or defective parts. Please refer to the SCSI section first if your SCSI hard drive is not working. However, there are a couple of things you may want to take into account before you purchase SCSI hard drives for your system.

### Rotational speed

Rotational speeds of SCSI drives sold today range from around 4,500RPM to 10,000RPM. Most of them are either 5,400RPM or 7,200RPM. Even though the 7,200RPM drives can generally transfer data faster, they run considerably hotter than their 5,400RPM counterparts. A large fraction of today's disk drive

malfunctions are heat-related. If you do not have very good cooling in your PC case, you may want to stick with 5,400RPM or slower drives.

Note that newer drives, with higher areal recording densities, can deliver much more bits per rotation than older ones. Today's top-of-line 5,400RPM drives can sustain a throughput comparable to 7,200RPM drives of one or two model generations ago. The number to find on the spec sheet for bandwidth is "internal data (or transfer) rate". It is usually in megabits/sec so divide it by 8 and you'll get the rough approximation of how much megabytes/sec you can get out of the drive.

(If you are a speed maniac and want a 10,000RPM drive for your cute little peecce, be my guest; however, those drives become extremely hot. Don't even think about it if you don't have a fan blowing air *directly* at the drive or a properly ventilated disk enclosure.)

Obviously, the latest 10,000RPM drives and 7,200RPM drives can deliver more data than the latest 5,400RPM drives, so if absolute bandwidth is the necessity for your applications, you have little choice but to get the faster drives. Also, if you need low latency, faster drives are better; not only do they usually have lower average seek times, but also the rotational delay is one place where slow-spinning drives can never beat a faster one. (The average rotational latency is half the time it takes to rotate the drive once; thus, it's 3 milliseconds for 10,000RPM drives, 4.2ms for 7,200RPM drives and 5.6ms for 5,400RPM drives.) Latency is seek time plus rotational delay. Make sure you understand whether you need low latency or more accesses per second, though; in the latter case (e.g., news servers), it may not be optimal to purchase one big fast drive. You can achieve similar or even better results by using the ccd (concatenated disk) driver to create a striped disk array out of multiple slower drives for comparable overall cost.

Make sure you have adequate air flow around the drive, especially if you are going to use a fast-spinning drive. You generally need at least 1/2" (1.25cm) of spacing above and below a drive. Understand how the air flows through your PC case. Most cases have the power supply suck the air out of the back. See where the air flows in, and put the drive where it will have the largest volume of cool air flowing around it. You may need to seal some unwanted holes or add a new fan for effective cooling.

Another consideration is noise. Many 7,200 or faster drives generate a high-pitched whine which is quite unpleasant to most people. That, plus the extra fans often required for cooling, may make 7,200 or faster drives unsuitable for some office and home environments.

### **Form factor**

Most SCSI drives sold today are of 3.5" form factor. They come in two different heights; 1.6" ("half-height") or 1" ("low-profile"). The half-height drive is the same height as a CD-ROM drive. However, don't forget the spacing rule mentioned in the previous section. If you have three standard 3.5" drive bays, you will not be able to put three half-height drives in there (without frying them, that is).

## Interface

The majority of SCSI hard drives sold today are Ultra or Ultra-wide SCSI. The maximum bandwidth of Ultra SCSI is 20MB/sec, and Ultra-wide SCSI is 40MB/sec. There is no difference in max cable length between Ultra and Ultra-wide; however, the more devices you have on the same bus, the sooner you will start having bus integrity problems. Unless you have a well-designed disk enclosure, it is not easy to make more than 5 or 6 Ultra SCSI drives work on a single bus.

On the other hand, if you need to connect many drives, going for Fast-wide SCSI may not be a bad idea. That will have the same max bandwidth as Ultra (narrow) SCSI, while electronically it's much easier to get it "right". My advice would be: if you want to connect many disks, get wide SCSI drives; they usually cost a little more but it may save you down the road. (Besides, if you can't afford the cost difference, you shouldn't be building a disk array.)

There are two variant of wide SCSI drives; 68-pin and 80-pin SCA (Single Connector Attach). The SCA drives don't have a separate 4-pin power connector, and also read the SCSI ID settings through the 80-pin connector. If you are really serious about building a large storage system, get SCA drives and a good SCA enclosure (dual power supply with at least one extra fan). They are more electronically sound than 68-pin counterparts because there is no "stub" of the SCSI bus inside the disk canister as in arrays built from 68-pin drives. They are easier to install too (you just need to screw the drive in the canister, instead of trying to squeeze in your fingers in a tight place to hook up all the little cables (like the SCSI ID and disk activity LED lines).

## \* IDE hard drives

## Tape drives

*Contributed by Jonathan M. Bresler <jmb@FreeBSD.ORG>. 2 July 1996.*

### General tape access commands

mt(1) provides generic access to the tape drives. Some of the more common commands are `rewind`, `erase`, and `status`. See the mt(1) manual page for a detailed description.

### Controller Interfaces

There are several different interfaces that support tape drives. The interfaces are SCSI, IDE, Floppy and

Parallel Port. A wide variety of tape drives are available for these interfaces. Controllers are discussed in Disk/tape controllers.

## **SCSI drives**

The st(4) driver provides support for 8mm (Exabyte), 4mm (DAT: Digital Audio Tape), QIC (Quarter-Inch Cartridge), DLT (Digital Linear Tape), QIC Minicartridge and 9-track (remember the big reels that you see spinning in Hollywood computer rooms) tape drives. See the st(4) manual page for a detailed description.

The drives listed below are currently being used by members of the FreeBSD community. They are not the only drives that will work with FreeBSD. They just happen to be the ones that we use.

### **4mm (DAT: Digital Audio Tape)**

Archive Python

HP C1533A

HP C1534A

HP 35450A

HP 35470A

HP 35480A

SDT-5000

Wangtek 6200

### **8mm (Exabyte)**

EXB-8200

EXB-8500

EXB-8505

### **QIC (Quarter-Inch Cartridge)**

Archive Ananconda 2750

Archive Viper 60

Archive Viper 150

Archive Viper 2525  
Tandberg TDC 3600  
Tandberg TDC 3620  
Tandberg TDC 4222  
Wangtek 5525ES

### **DLT (Digital Linear Tape)**

Digital TZ87

### **Mini-Cartridge**

Conner CTMS 3200  
Exabyte 2501

### **Autoloaders/Changers**

Hewlett-Packard HP C1553A Autoloading DDS2

### **\* IDE drives**

### **Floppy drives**

Conner 420R

### **\* Parallel port drives**

### **Detailed Information**

#### **Archive Anaconda 2750**

The boot message identifier for this drive is ARCHIVE ANCDA 2750 28077 -003 type 1

```
removable SCSI 2
```

This is a QIC tape drive.

Native capacity is 1.35GB when using QIC-1350 tapes. This drive will read and write QIC-150 (DC6150), QIC-250 (DC6250), and QIC-525 (DC6525) tapes as well.

Data transfer rate is 350kB/s using dump(8). Rates of 530kB/s have been reported when using Amanda. Production of this drive has been discontinued.

The SCSI bus connector on this tape drive is reversed from that on most other SCSI devices. Make sure that you have enough SCSI cable to twist the cable one-half turn before and after the Archive Anaconda tape drive, or turn your other SCSI devices upside-down.

Two kernel code changes are required to use this drive. This drive will not work as delivered.

If you have a SCSI-2 controller, short jumper 6. Otherwise, the drive behaves as a SCSI-1 device. When operating as a SCSI-1 device, this drive, “locks” the SCSI bus during some tape operations, including: fsf, rewind, and rewoffl.

If you are using the NCR SCSI controllers, patch the file `/usr/src/sys/pci/ncr.c` (as shown below). Build and install a new kernel.

```
*** 4831,4835 ****
        };

!           if (np->latetime>4) {
                /*
                **           Although we tried to wake it up,
-- 4831,4836 ---
        };

!           if (np->latetime>1200) {
                /*
                **           Although we tried to wake it up,
```

Reported by: Jonathan M. Bresler <jmb@FreeBSD.ORG>

### Archive Python

The boot message identifier for this drive is ARCHIVE Python 28454-XXX4ASB type 1 removable SCSI 2 density code 0x8c, 512-byte blocks

This is a DDS-1 tape drive.

Native capacity is 2.5GB on 90m tapes.

Data transfer rate is XXX.

This drive was repackaged by Sun Microsystems as model 411.

Reported by: Bob Bishop <rb@gid.co.uk>

### **Archive Viper 60**

The boot message identifier for this drive is ARCHIVE VIPER 60 21116 -007 type 1 removable SCSI 1

This is a QIC tape drive.

Native capacity is 60MB.

Data transfer rate is XXX.

Production of this drive has been discontinued.

Reported by: Philippe Regnauld <regnauld@hsc.fr>

### **Archive Viper 150**

The boot message identifier for this drive is ARCHIVE VIPER 150 21531 -004 Archive Viper 150 is a known rogue type 1 removable SCSI 1. A multitude of firmware revisions exist for this drive. Your drive may report different numbers (e.g 21247 -005).

This is a QIC tape drive.

Native capacity is 150/250MB. Both 150MB (DC6150) and 250MB (DC6250) tapes have the recording format. The 250MB tapes are approximately 67% longer than the 150MB tapes. This drive can read 120MB tapes as well. It can not write 120MB tapes.

Data transfer rate is 100kB/s

This drive reads and writes DC6150 (150MB) and DC6250 (250MB) tapes.

This drives quirks are known and pre-compiled into the scsi tape device driver (st(4)).

Under FreeBSD 2.2-current, use `mt blocksize 512` to set the blocksize. (The particular drive had firmware revision 21247 -005. Other firmware revisions may behave differently) Previous versions of FreeBSD did not have this problem.

Production of this drive has been discontinued.

Reported by: Pedro A M Vazquez <vazquez@IQM.Unicamp.BR>

Mike Smith <msmith@atrad.adelaide.edu.au>

### **Archive Viper 2525**

The boot message identifier for this drive is ARCHIVE VIPER 2525 25462 -011 type 1 removable SCSI 1

This is a QIC tape drive.

Native capacity is 525MB.

Data transfer rate is 180kB/s at 90 inches/sec.

The drive reads QIC-525, QIC-150, QIC-120 and QIC-24 tapes. Writes QIC-525, QIC-150, and QIC-120.

Firmware revisions prior to 25462 -011 are bug ridden and will not function properly.

Production of this drive has been discontinued.

### **Conner 420R**

The boot message identifier for this drive is Conner tape.

This is a floppy controller, minicartridge tape drive.

Native capacity is XXXX

Data transfer rate is XXX

The drive uses QIC-80 tape cartridges.

Reported by: Mark Hannon <mark@seeware.DIALix.oz.au>

### **Conner CTMS 3200**

The boot message identifier for this drive is CONNER CTMS 3200 7.00 type 1 removable SCSI 2.

This is a minicartridge tape drive.

Native capacity is XXXX

Data transfer rate is XXX

The drive uses QIC-3080 tape cartridges.

Reported by: Thomas S. Traylor <tst@titan.cs.mci.com>

### **DEC TZ87 (<http://www.digital.com/info/Customer-Update/931206004.txt.html>)**

The boot message identifier for this drive is DEC TZ87 (C) DEC 9206 type 1 removable SCSI 2 density code 0x19

This is a DLT tape drive.

Native capacity is 10GB.

This drive supports hardware data compression.

Data transfer rate is 1.2MB/s.

This drive is identical to the Quantum DLT2000. The drive firmware can be set to emulate several well-known drives, including an Exabyte 8mm drive.

Reported by: Wilko Bulte <wilko@yedi.iaf.nl>

### **Exabyte EXB-2501**

**(<http://www.Exabyte.COM:80/Products/Minicartridge/2501/Rfeatures.html>)**

The boot message identifier for this drive is EXABYTE EXB-2501

This is a mini-cartridge tape drive.

Native capacity is 1GB when using MC3000XL minicartridges.

Data transfer rate is XXX

This drive can read and write DC2300 (550MB), DC2750 (750MB), MC3000 (750MB), and MC3000XL (1GB) minicartridges.

WARNING: This drive does not meet the SCSI-2 specifications. The drive locks up completely in response to a SCSI MODE\_SELECT command unless there is a formatted tape in the drive. Before using this drive, set the tape blocksize with

```
# mt -f /dev/st0ctl.0 blocksize 1024
```

Before using a minicartridge for the first time, the minicartridge must be formatted. FreeBSD 2.1.0-RELEASE and earlier:

```
# /sbin/scsi -f /dev/rst0.ct1 -s 600 -c "4 0 0 0 0 0"
```

(Alternatively, fetch a copy of the `scsiformat` shell script from FreeBSD 2.1.5/2.2.) FreeBSD 2.1.5 and later:

```
# /sbin/scsiformat -q -w /dev/rst0.ct1
```

Right now, this drive cannot really be recommended for FreeBSD.

Reported by: Bob Beaulieu <ez@eztravel.com>

### **Exabyte EXB-8200**

The boot message identifier for this drive is EXABYTE EXB-8200 252X type 1 removable SCSI 1

This is an 8mm tape drive.

Native capacity is 2.3GB.

Data transfer rate is 270kB/s.

This drive is fairly slow in responding to the SCSI bus during boot. A custom kernel may be required (set SCSI\_DELAY to 10 seconds).

There are a large number of firmware configurations for this drive, some have been customized to a particular vendor's hardware. The firmware can be changed via EPROM replacement.

Production of this drive has been discontinued.

Reported by: Mike Smith <msmith@atrad.adelaide.edu.au>

### **Exabyte EXB-8500**

The boot message identifier for this drive is EXABYTE EXB-8500-85Qanx0 0415 type 1 removable SCSI 2

This is an 8mm tape drive.

Native capacity is 5GB.

Data transfer rate is 300kB/s.

Reported by: Greg Lehey <grog@lemis.de>

### **Exabyte EXB-8505 (<http://www.Exabyte.COM:80/Products/8mm/8505XL/Rfeatures.html>)**

The boot message identifier for this drive is EXABYTE EXB-85058SQANXR1 05B0 type 1 removable SCSI 2

This is an 8mm tape drive which supports compression, and is upward compatible with the EXB-5200 and EXB-8500.

Native capacity is 5GB.

The drive supports hardware data compression.

Data transfer rate is 300kB/s.

Reported by: Glen Foster <gfoster@gfoster.com>

**Hewlett-Packard HP C1533A**

The boot message identifier for this drive is `HP C1533A 9503 type 1 removable SCSI 2`.

This is a DDS-2 tape drive. DDS-2 means hardware data compression and narrower tracks for increased data capacity.

Native capacity is 4GB when using 120m tapes. This drive supports hardware data compression.

Data transfer rate is 510kB/s.

This drive is used in Hewlett-Packard's SureStore 6000eU and 6000i tape drives and C1533A DDS-2 DAT drive.

The drive has a block of 8 dip switches. The proper settings for FreeBSD are: 1 ON; 2 ON; 3 OFF; 4 ON; 5 ON; 6 ON; 7 ON; 8 ON.

switch 1	switch 2	Result
On	On	Compression enabled at power-on, with host control
On	Off	Compression enabled at power-on, no host control
Off	On	Compression disabled at power-on, with host control
Off	Off	Compression disabled at power-on, no host control

Switch 3 controls MRS (Media Recognition System). MRS tapes have stripes on the transparent leader. These identify the tape as DDS (Digital Data Storage) grade media. Tapes that do not have the stripes will be treated as write-protected. Switch 3 OFF enables MRS. Switch 3 ON disables MRS.

See HP SureStore Tape Products ([http://www.hp.com/tape/c\\_intro.html](http://www.hp.com/tape/c_intro.html)) and Hewlett-Packard Disk and Tape Technical Information ([http://www.impediment.com/hp/hp\\_technical.html](http://www.impediment.com/hp/hp_technical.html)) for more information on configuring this drive.

*Warning:* Quality control on these drives varies greatly. One FreeBSD core-team member has returned 2 of these drives. Neither lasted more than 5 months.

Reported by: Stefan Esser <[se@FreeBSD.ORG](mailto:se@FreeBSD.ORG)>

**Hewlett-Packard HP 1534A**

The boot message identifier for this drive is `HP HP35470A T503 type 1 removable SCSI 2 Sequential-Access density code 0x13, variable blocks`.

This is a DDS-1 tape drive. DDS-1 is the original DAT tape format.

Native capacity is 2GB when using 90m tapes.

Data transfer rate is 183kB/s.

The same mechanism is used in Hewlett-Packard's SureStore 2000i (<http://www.dmo.hp.com/tape/sst2000.htm>) tape drive, C35470A DDS format DAT drive, C1534A DDS format DAT drive and HP C1536A DDS format DAT drive.

The HP C1534A DDS format DAT drive has two indicator lights, one green and one amber. The green one indicates tape action: slow flash during load, steady when loaded, fast flash during read/write operations. The amber one indicates warnings: slow flash when cleaning is required or tape is nearing the end of its useful life, steady indicates an hard fault. (factory service required?)

Reported by Gary Crutcher <[gcrutchr@nightflight.com](mailto:gcrutchr@nightflight.com)>

### **Hewlett-Packard HP C1553A Autoloading DDS2**

The boot message identifier for this drive is "".

This is a DDS-2 tape drive with a tape changer. DDS-2 means hardware data compression and narrower tracks for increased data capacity.

Native capacity is 24GB when using 120m tapes. This drive supports hardware data compression.

Data transfer rate is 510kB/s (native).

This drive is used in Hewlett-Packard's SureStore 12000e (<http://www.dmo.hp.com/tape/sst12000.htm>) tape drive.

The drive has two selectors on the rear panel. The selector closer to the fan is SCSI id. The other selector should be set to 7.

There are four internal switches. These should be set: 1 ON; 2 ON; 3 ON; 4 OFF.

At present the kernel drivers do not automatically change tapes at the end of a volume. This shell script can be used to change tapes:

```
#!/bin/sh
PATH="/sbin:/usr/sbin:/bin:/usr/bin"; export PATH

usage()
{
    echo "Usage: dds_changer [123456ne] raw-device-name"
    echo "1..6 = Select cartridge"
    echo "next cartridge"
    echo "eject magazine"
    exit 2
}
```

```
}  
  
if [ $# -ne 2 ] ; then  
    usage  
fi  
  
cdb3=0  
cdb4=0  
cdb5=0  
  
case $1 in  
    [123456])  
        cdb3=$1  
        cdb4=1  
        ;;  
    n)  
        ;;  
    e)  
        cdb5=0x80  
        ;;  
    ?)  
        usage  
        ;;  
esac  
  
scsi -f $2 -s 100 -c "1b 0 0 $cdb3 $cdb4 $cdb5"
```

### **Hewlett-Packard HP 35450A**

The boot message identifier for this drive is HP HP35450A -A C620 type 1 removable SCSI 2 Sequential-Access density code 0x13

This is a DDS-1 tape drive. DDS-1 is the original DAT tape format.

Native capacity is 1.2GB.

Data transfer rate is 160kB/s.

Reported by: mark thompson <mark.a.thompson@pobox.com>

### **Hewlett-Packard HP 35470A**

The boot message identifier for this drive is HP HP35470A 9 09 type 1 removable SCSI 2

This is a DDS-1 tape drive. DDS-1 is the original DAT tape format.

Native capacity is 2GB when using 90m tapes.

Data transfer rate is 183kB/s.

The same mechanism is used in Hewlett-Packard's SureStore 2000i (<http://www.dmo.hp.com/tape/sst2000.htm>) tape drive, C35470A DDS format DAT drive, C1534A DDS format DAT drive, and HP C1536A DDS format DAT drive.

*Warning:* Quality control on these drives varies greatly. One FreeBSD core-team member has returned 5 of these drives. None lasted more than 9 months.

Reported by: David Dawes <[dawes@rf900.physics.usyd.edu.au](mailto:dawes@rf900.physics.usyd.edu.au)> (9 09)

### **Hewlett-Packard HP 35480A**

The boot message identifier for this drive is `HP HP35480A 1009 type 1 removable SCSI 2 Sequential-Access density code 0x13`.

This is a DDS-DC tape drive. DDS-DC is DDS-1 with hardware data compression. DDS-1 is the original DAT tape format.

Native capacity is 2GB when using 90m tapes. It cannot handle 120m tapes. This drive supports hardware data compression. Please refer to the section on HP C1533A for the proper switch settings.

Data transfer rate is 183kB/s.

This drive is used in Hewlett-Packard's SureStore 5000eU (<http://www.dmo.hp.com/tape/sst5000.htm>) and 5000i (<http://www.dmo.hp.com/tape/sst5000.htm>) tape drives and C35480A DDS format DAT drive..

This drive will occasionally hang during a tape eject operation (`mt offline`). Pressing the front panel button will eject the tape and bring the tape drive back to life.

**WARNING:** HP 35480-03110 only. On at least two occasions this tape drive when used with FreeBSD 2.1.0, an IBM Server 320 and an 2940W SCSI controller resulted in all SCSI disk partitions being lost. The problem has not be analyzed or resolved at this time.

### **Sony SDT-5000 (<http://www.sel.sony.com/SEL/ccpg/storage/tape/t5000.html>)**

There are at least two significantly different models: one is a DDS-1 and the other DDS-2. The DDS-1 version is `SDT-5000 3.02`. The DDS-2 version is `SONY SDT-5000 327M`. The DDS-2 version has a 1MB cache. This cache is able to keep the tape streaming in almost any circumstances.

The boot message identifier for this drive is `SONY SDT-5000 3.02 type 1 removable SCSI 2 Sequential-Access density code 0x13`

Native capacity is 4GB when using 120m tapes. This drive supports hardware data compression.

Data transfer rate is depends upon the model or the drive. The rate is 630kB/s for the SONY SDT-5000 327M while compressing the data. For the SONY SDT-5000 3.02, the data transfer rate is 225kB/s.

In order to get this drive to stream, set the blocksize to 512 bytes (`mt blocksize 512`) reported by Kenneth Merry [ken@ulc199.residence.gatech.edu](mailto:ken@ulc199.residence.gatech.edu)

SONY SDT-5000 327M information reported by Charles Henrich [henrich@msu.edu](mailto:henrich@msu.edu)

Reported by: Jean-Marc Zucconi <[jmz@FreeBSD.ORG](mailto:jmz@FreeBSD.ORG)>

### **Tandberg TDC 3600**

The boot message identifier for this drive is `TANDBERG TDC 3600 =08: type 1 removable SCSI 2`

This is a QIC tape drive.

Native capacity is 150/250MB.

This drive has quirks which are known and work around code is present in the scsi tape device driver (`st(4)`). Upgrading the firmware to XXX version will fix the quirks and provide SCSI 2 capabilities.

Data transfer rate is 80kB/s.

IBM and Emerald units will not work. Replacing the firmware EPROM of these units will solve the problem.

Reported by: Michael Smith <[msmith@atrad.adelaide.edu.au](mailto:msmith@atrad.adelaide.edu.au)>

### **Tandberg TDC 3620**

This is very similar to the Tandberg TDC 3600 drive.

Reported by: Jörg Wunsch <[joerg@FreeBSD.ORG](mailto:joerg@FreeBSD.ORG)>

### **Tandberg TDC 4222**

The boot message identifier for this drive is `TANDBERG TDC 4222 =07 type 1 removable SCSI 2`

This is a QIC tape drive.

Native capacity is 2.5GB. The drive will read all cartridges from the 60 MB (DC600A) upwards, and write 150 MB (DC6150) upwards. Hardware compression is optionally supported for the 2.5 GB cartridges.

This drives quirks are known and pre-compiled into the scsi tape device driver (st(4)) beginning with FreeBSD 2.2-current. For previous versions of FreeBSD, use `mt` to read one block from the tape, rewind the tape, and then execute the backup program (`mt fsr 1; mt rewind; dump ...`)

Data transfer rate is 600kB/s (vendor claim with compression), 350 KB/s can even be reached in start/stop mode. The rate decreases for smaller cartridges.

Reported by: Jörg Wunsch <joerg@FreeBSD.ORG>

### **Wangtek 5525ES**

The boot message identifier for this drive is `WANGTEK 5525ES SCSI REV7 3R1 type 1 removable SCSI 1 density code 0x11, 1024-byte blocks`

This is a QIC tape drive.

Native capacity is 525MB.

Data transfer rate is 180kB/s.

The drive reads 60, 120, 150, and 525MB tapes. The drive will not write 60MB (DC600 cartridge) tapes. In order to overwrite 120 and 150 tapes reliably, first erase (`mt erase`) the tape. 120 and 150 tapes used a wider track (fewer tracks per tape) than 525MB tapes. The “extra” width of the previous tracks is not overwritten, as a result the new data lies in a band surrounded on both sides by the previous data unless the tape have been erased.

This drives quirks are known and pre-compiled into the scsi tape device driver (st(4)).

Other firmware revisions that are known to work are: M75D

Reported by: Marc van Kempen <marc@bowtie.nl> REV73R1 Andrew Gordon <Andrew.Gordon@net-tel.co.uk> M75D

### **Wangtek 6200**

The boot message identifier for this drive is `WANGTEK 6200-HS 4B18 type 1 removable SCSI 2 Sequential-Access density code 0x13`

This is a DDS-1 tape drive.

Native capacity is 2GB using 90m tapes.

Data transfer rate is 150kB/s.

Reported by: Tony Kimball <alk@Think.COM>

**\* Problem drives**

**CD-ROM drives**

*Contributed by David O'Brien <obrien@FreeBSD.ORG>. 23 November 1997.*

As mentioned in Jordan's Picks Generally speaking those in *The FreeBSD Project* prefer SCSI CDROM drives over IDE CDROM drives. However not all SCSI CDROM drives are equal. Some feel the quality of some SCSI CDROM drives have been deteriorating to that of IDE CDROM drives. Toshiba used to be the favored stand-by, but many on the SCSI mailing list have found displeasure with the 12x speed XM-5701TA as its volume (when playing audio CDROMs) is not controllable by the various audio player software.

Another area where SCSI CDROM manufacturers are cutting corners is adherence to the SCSI specification. Many SCSI CDROMs will respond to multiple LUNs for its target address. Known violators include the 6x Teac CD-56S 1.0D.

**\* Other**

**\* Other**

**\* PCMCIA**

# Chapter 13. Localization

## Russian Language (KOI8-R encoding)

*Contributed by Andrey A. Chernov <ache@FreeBSD.ORG> 1 May 1997.*

See more info about KOI8-R encoding at KOI8-R References (Russian Net Character Set) (<http://www.nagual.pp.ru/~ache/koi8.html>).

### Console Setup

1. Add following line to your kernel configuration file:

```
options          "SC_MOUSE_CHAR=0x03"
```

to move character codes used for mouse cursor off KOI8-R pseudographics range.

2. Russian console entry in `/etc/rc.conf` should look like:

```
keymap=ru.koi8-r
keychange="61 ^[[K"
scrnmap=koi8-r2cp866
font8x16=cp866b-8x16
font8x14=cp866-8x14
font8x8=cp866-8x8
```

**Note:** `^[[` means that real ESC character must be entered into `/etc/rc.conf`, not just `^[[` string.

This tuning means KOI8-R keyboard with Alternative screen font mapped to KOI8-R encoding to preserve pseudographics, `Gray Delete` key remapped to match `termcap(5)` entry for FreeBSD console.

RUS/LAT switch will be `CapsLock`. Old `CapsLock` function still available via `Shift+CapsLock`. `CapsLock` LED will indicate RUS mode, not `CapsLock` mode.

3. For each `ttyv?` entry in `/etc/ttys` change terminal type from `cons25` to `cons25r`, i.e. each entry should look like:

```
ttyv0 "/usr/libexec/getty Pc"          cons25r on  secure
```

## Locale Setup

There is two environment variables for locale setup:

- LANG for POSIX setlocale(3) family functions;
- MM\_CHARSET for applications MIME chararter set.

The best way is using `/etc/login.conf` russian user's login class in `passwd(5)` entry login class position. See `login.conf(5)` for details.

### Login Class Method

First of all check your `/etc/login.conf` have russian login class, this entry may looks like:

```
russian:Russian Users Accounts:\
      :charset=KOI8-R:\
      :lang=ru_RU.KOI8-R:\
      :tc=default:
```

### How to do it with vipw(8)

If you use `vipw(8)` for adding new users, `/etc/master.passwd` entry should looks like:

```
user:password:1111:11:russian:0:0:User Name:/home/user:/bin/csh
```

### How to do it with adduser(8)

If you use `adduser(8)` for adding new users:

- Set

```
defaultclass = russian
```

in `/etc/adduser.conf` (you must enter default class for all non-Russian users in this case);

- Alternative variant will be answering `russian` each time when you see

```
Enter login class: default []:
```

prompt from `adduser(8)`;

- Another variant: call

```
# adduser -class russian
```

for each Russian user you want to add.

**How to do it with pw(8)**

If you use pw(8) for adding new users, call it in this form:

```
# pw useradd user_name -L russian
```

**Shell Startup Files Method**

If you don't want to use login class method for some reasons, just set this two environment variables in the following shell startup files:

- /etc/profile:
 

```
LANG=ru_RU.KOI8-R; export LANG
MM_CHARSET=KOI8-R; export MM_CHARSET
```
- /etc/csh.login:
 

```
setenv LANG ru_RU.KOI8-R
setenv MM_CHARSET KOI8-R
```

Alternatively you can add this instructions to

- /usr/share/skel/dot.profile:
 

(similar to /etc/profile above);
- /usr/share/skel/dot.login:
 

(similar to /etc/csh.login above).

**Printer Setup**

Since most printers with Russian characters comes with hardware code page CP866, special output filter needed for KOI8-R -> CP866 conversion. Such filter installed by default as /usr/libexec/lpr/ru/koi2alt. So, Russian printer /etc/printcap entry should looks like:

```
lp|Russian local line printer:\
    :sh:of=/usr/libexec/lpr/ru/koi2alt:\
    :lp=/dev/lpt0:sd=/var/spool/output/lpd:lf=/var/log/lpd-errs:
```

See `printcap(5)` for detailed description.

## MSDOS FS and Russian file names

Look at following example `fstab(5)` entry to enable support for Russian file names in MSDOS FS:

```
/dev/sd0s1      /dos/c    msdos    rw,-W=koi2dos,-L=ru_RU.KOI8-R 0 0
```

See `mount.msdos(8)` for detailed description of `-w` and `-L` options.

## X Window Setup

Step by step instructions:

1. Do non-X locale setup first as described.

**Note:** Russian KOI8-R locale may not work with old XFree86 releases (lower than 3.3). XFree86 port from `/usr/ports/x11/XFree86` already have most recent XFree86 version, so it will work, if you install XFree86 from this port. XFree86 version shipped with the latest FreeBSD distribution should work too (check XFree86 version number not less than 3.3 first).

2. Go to `/usr/ports/russian/X.language` directory and say

```
# make all install
```

there. This port install latest version of KOI8-R fonts. XFree86 3.3 already have some KOI8-R fonts, but this ones scaled better.

Check find "Files" section in your `/etc/XF86Config`, following lines must be before any other `FontPath` entries:

```
FontPath  "/usr/X11R6/lib/X11/fonts/cyrillic/misc"
FontPath  "/usr/X11R6/lib/X11/fonts/cyrillic/75dpi"
FontPath  "/usr/X11R6/lib/X11/fonts/cyrillic/100dpi"
```

If you use high resolution video mode, swap 75 dpi and 100 dpi lines.

3. To activate Russian keyboard add

```
XkbKeymap  "xfree86(ru)"
```

line into "Keyboard" section in your `/etc/XF86Config`, also make sure that `XkbDisable` is turned off (commented out) there.

RUS/LAT switch will be `CapsLock`. Old `CapsLock` function still available via `Shift+CapsLock` (in LAT mode only).

**Note:** Russian XKB keyboard may not work with old XFree86 versions, see locale note for more info. Russian XKB keyboard may not work with non-localized applications too, minimally localized application should call `XtSetLanguageProc (NULL, NULL, NULL)`; function early in the program.

## German Language (ISO 8859-1)

Slaven Rezic <eserte@cs.tu-berlin.de> wrote a tutorial how to use umlauts on a FreeBSD machine. The tutorial is written in German and available at <http://www.de.freebsd.org/de/umlaute/>.

## **III. Network Communications**

# Chapter 14. Serial Communications

## Serial Basics

*Assembled from FAQ.*

This section should give you some general information about serial ports. If you do not find what you want here, check into the Terminal and Dialup sections of the handbook.

The `ttYdX` (or `cuaaX`) device is the regular device you will want to open for your applications. When a process opens the device, it will have a default set of terminal I/O settings. You can see these settings with the command

```
# stty -a -f /dev/ttyd1
```

When you change the settings to this device, the settings are in effect until the device is closed. When it is reopened, it goes back to the default set. To make changes to the default set, you can open and adjust the settings of the “initial state” device. For example, to turn on CLOCAL mode, 8 bits, and XON/XOFF flow control by default for `ttYd5`, do:

```
# stty -f /dev/ttyid5 clocal cs8 ixon ixoff
```

A good place to do this is in `/etc/rc.serial`. Now, an application will have these settings by default when it opens `ttYd5`. It can still change these settings to its liking, though.

You can also prevent certain settings from being changed by an application by making adjustments to the “lock state” device. For example, to lock the speed of `ttYd5` to 57600 bps, do

```
# stty -f /dev/ttyld5 57600
```

Now, an application that opens `ttYd5` and tries to change the speed of the port will be stuck with 57600 bps.

Naturally, you should make the initial state and lock state devices writable only by `root`. The `MAKEDEV` script does *not* do this when it creates the device entries.

## Terminals

*Contributed by Sean Kelly <kelly@fs1.noaa.gov> 28 July 1996*

Terminals provide a convenient and low-cost way to access the power of your FreeBSD system when you are not at the computer's console or on a connected network. This section describes how to use terminals with FreeBSD.

## Uses and Types of Terminals

The original Unix systems did not have consoles. Instead, people logged in and ran programs through terminals that were connected to the computer's serial ports. It is quite similar to using a modem and some terminal software to dial into a remote system to do text-only work.

Today's PCs have consoles capable of high quality graphics, but the ability to establish a login session on a serial port still exists in nearly every Unix-style operating system today; FreeBSD is no exception. By using a terminal attached to a unused serial port, you can log in and run any text program that you would normally run on the console or in an `xterm` window in the X Window System.

For the business user, you can attach many terminals to a FreeBSD system and place them on your employees' desktops. For a home user, a spare computer such as an older IBM PC or a Macintosh can be a terminal wired into a more powerful computer running FreeBSD. You can turn what might otherwise be a single-user computer into a powerful multiple user system.

For FreeBSD, there are three kinds of terminals:

- Dumb terminals
- PCs acting as terminals
- X terminals

The remaining subsections describe each kind.

### Dumb Terminals

Dumb terminals are specialized pieces of hardware that let you connect to computers over serial lines. They are called "dumb" because they have only enough computational power to display, send, and receive text. You cannot run any programs on them. It is the computer to which you connect them that has all the power to run text editors, compilers, email, games, and so forth.

There are hundreds of kinds of dumb terminals made by many manufacturers, including Digital Equipment Corporation's VT-100 and Wyse's WY-75. Just about any kind will work with FreeBSD. Some high-end terminals can even display graphics, but only certain software packages can take advantage of these advanced features.

Dumb terminals are popular in work environments where workers do not need access to graphic applications such as those provided by the X Window System.

## PCs Acting As Terminals

If a dumb terminal has just enough ability to display, send, and receive text, then certainly any spare personal computer can be a dumb terminal. All you need is the proper cable and some *terminal emulation* software to run on the computer.

Such a configuration is popular in homes. For example, if your spouse is busy working on your FreeBSD system's console, you can do some text-only work at the same time from a less powerful personal computer hooked up as a terminal to the FreeBSD system.

## X Terminals

X terminals are the most sophisticated kind of terminal available. Instead of connecting to a serial port, they usually connect to a network like Ethernet. Instead of being relegated to text-only applications, they can display any X application.

We introduce X terminals just for the sake of completeness. However, this chapter does *not* cover setup, configuration, or use of X terminals.

## Cables and Ports

To connect a terminal to your FreeBSD system, you need the right kind of cable and a serial port to which to connect it. This section tells you what to do. If you are already familiar with your terminal and the cable it requires, skip to Configuration.

### Cables

Because terminals use serial ports, you need to use serial—also known as RS-232C—cables to connect the terminal to the FreeBSD system.

There are a couple of kinds of serial cables. Which one you'll use depends on the terminal you want to connect:

- If you are connecting a personal computer to act as a terminal, use a null-modem cable. A null-modem cable connects two computers or terminals together.
- If you have an actual terminal, your best source of information on what cable to use is the documentation that accompanied the terminal. If you do not have the documentation, then try a null-modem cable. If that does not work, then try a standard cable.

Also, the serial port on *both* the terminal and your FreeBSD system must have connectors that will fit the cable you are using.

### Null-modem cables

A null-modem cable passes some signals straight through, like “signal ground,” but switches other signals. For example, the “send data” pin on one end goes to the “receive data” pin on the other end.

If you like making your own cables, here is a table showing a recommended way to construct a null-modem cable for use with terminals. This table shows the RS-232C signal names and the pin numbers on a DB-25 connector.

Signal	Pin #		Pin #	Signal
TxD	2	connects to	3	RxD
RxD	3	connects to	2	TxD
DTR	20	connects to	6	DSR
DSR	6	connects to	20	DTR
SG	7	connects to	7	SG
DCD	8	connects to	4	RTS
RTS	4		5	CTS
CTS	5	connects to	8	DCD

**Note:** For DCD to RTS, connect pins 4 to 5 internally in the connector hood, and then to pin 8 in the remote hood.

### Standard RS-232C Cables

A standard serial cable passes all the RS-232C signals straight-through. That is, the “send data” pin on one end of the cable goes to the “send data” pin on the other end. This is the type of cable to connect a modem to your FreeBSD system, and the type of cable needed for some terminals.

### Ports

Serial ports are the devices through which data is transferred between the FreeBSD host computer and the terminal. This section describes the kinds of ports that exist and how they are addressed in FreeBSD.

## Kinds of Ports

Several kinds of serial ports exist. Before you purchase or construct a cable, you need to make sure it will fit the ports on your terminal and on the FreeBSD system.

Most terminals will have DB25 ports. Personal computers, including PCs running FreeBSD, will have DB25 or DB9 ports. If you have a multiport serial card for your PC, you may have RJ-12 or RJ-45 ports.

See the documentation that accompanied the hardware for specifications on the kind of port in use. A visual inspection of the port often works, too.

## Port Names

In FreeBSD, you access each serial port through an entry in the `/dev` directory. There are two different kinds of entries:

- Callin ports are named `/dev/ttydX` where `X` is the port number, starting from zero. Generally, you use the callin port for terminals. Callin ports require that the serial line assert the data carrier detect (DCD) signal to work.
- Callout ports are named `/dev/cuaaX`. You usually do not use the callout port for terminals, just for modems. You may use the callout port if the serial cable or the terminal does not support the carrier detect signal.

See the `sio(4)` manual page for more information.

If you have connected a terminal to the first serial port (COM1 in DOS parlance), then you want to use `/dev/ttyd0` to refer to the terminal. If it is on the second serial port (also known as COM2), it is `/dev/ttyd1`, and so forth.

Note that you may have to configure your kernel to support each serial port, especially if you have a multiport serial card. See *Configuring the FreeBSD Kernel* for more information.

## Configuration

This section describes what you need to configure on your FreeBSD system to enable a login session on a terminal. It assumes you have already configured your kernel to support the serial port to which the terminal is connected—and that you have connected it.

In a nutshell, you need to tell the `init` process, which is responsible for process control and initialization, to start a `getty` process, which is responsible for reading a login name and starting the `login` program.

To do so, you have to edit the `/etc/ttys` file. First, use the `su` command to become root. Then, make the following changes to `/etc/ttys`:

1. Add an line to `/etc/ttys` for the entry in the `/dev` directory for the serial port if it is not already there.
2. Specify that `/usr/libexec/getty` be run on the port, and specify the appropriate `getty` type from the `/etc/gettytab` file.
3. Specify the default terminal type.
4. Set the port to “on.”
5. Specify whether the port should be “secure.”
6. Force `init` to reread the `/etc/ttys` file.

As an optional step, you may wish to create a custom `getty` type for use in step 2 by making an entry in `/etc/gettytab`. This document does not explain how to do so; you are encouraged to see the `gettytab(5)` and the `getty(8)` manual pages for more information.

The remaining sections detail how to do these steps. We will use a running example throughout these sections to illustrate what we need to do. In our example, we will connect two terminals to the system: a Wyse-50 and a old 286 IBM PC running Procomm terminal software emulating a VT-100 terminal. We connect the Wyse to the second serial port and the 286 to the sixth serial port (a port on a multiport serial card).

For more information on the `/etc/ttys` file, see the `ttys(5)` manual page.

## Adding an Entry to `/etc/ttys`

First, you need to add an entry to the `/etc/ttys` file, unless one is already there.

The `/etc/ttys` file lists all of the ports on your FreeBSD system where you want to allow logins. For example, the first virtual console `ttv0` has an entry in this file. You can log in on the console using this entry. This file contains entries for the other virtual consoles, serial ports, and pseudo-ttys. For a hardwired terminal, just list the serial port's `/dev` entry without the `/dev` part.

When you installed your FreeBSD system, the `/etc/ttys` file included entries for the first four serial ports: `ttvd0` through `ttvd3`. If you are attaching a terminal on one of those ports, you do not need to add an entry.

In our example, we attached a Wyse-50 to the second serial port, `ttvd1`, which is already in the file. We need to add an entry for the 286 PC connected to the sixth serial port. Here is an excerpt of the `/etc/ttys` file after we add the new entry:

```
ttvd1  "/usr/libexec/getty std.9600"  unknown off secure
```

ttyd5

## Specifying the *getty* Type

Next, we need to specify what program will be run to handle the logins on a terminal. For FreeBSD, the standard program to do that is `/usr/libexec/getty`. It is what provides the `login:` prompt.

The program `getty` takes one (optional) parameter on its command line, the *getty* type. A *getty* type tells about characteristics on the terminal line, like bps rate and parity. The `getty` program reads these characteristics from the file `/etc/gettytab`.

The file `/etc/gettytab` contains lots of entries for terminal lines both old and new. In almost all cases, the entries that start with the text `std` will work for hardwired terminals. These entries ignore parity. There is a `std` entry for each bps rate from 110 to 115200. Of course, you can add your own entries to this file. The manual page `gettytab(5)` provides more information.

When setting the *getty* type in the `/etc/ttys` file, make sure that the communications settings on the terminal match.

For our example, the Wyse-50 uses no parity and connects at 38400 bps. The 286 PC uses no parity and connects at 19200 bps. Here is the `/etc/ttys` file so far (showing just the two terminals in which we are interested):

```
ttyd1  "/usr/libexec/getty std.38400"  unknown off secure
ttyd5  "/usr/libexec/getty std.19200"
```

Note that the second field—where we specify what program to run—appears in quotes. This is important, otherwise the type argument to `getty` might be interpreted as the next field.

## Specifying the Default Terminal Type

The third field in the `/etc/ttys` file lists the default terminal type for the port. For dialup ports, you typically put `unknown` or `dialup` in this field because users may dial up with practically any kind of terminal or software. For hardwired terminals, the terminal type does not change, so you can put a real terminal type in this field.

Users will usually use the `tset` program in their `.login` or `.profile` files to check the terminal type and prompt for one if necessary. By setting a terminal type in the `/etc/ttys` file, users can forego such prompting.

To find out what terminal types FreeBSD supports, see the file `/usr/share/misc/termcap`. It lists about 600 terminal types. You can add more if you wish. See the `termcap(5)` manual page for information.

In our example, the Wyse-50 is a Wyse-50 type of terminal (although it can emulate others, we will leave it in Wyse-50 mode). The 286 PC is running Procomm which will be set to emulate a VT-100. Here are the pertinent yet unfinished entries from the `/etc/ttys` file:

```
ttyd1  "/usr/libexec/getty std.38400"  wy50  off secure
ttyd5  "/usr/libexec/getty std.19200"  vt100
```

## Enabling the Port

The next field in `/etc/ttys`, the fourth field, tells whether to enable the port. Putting `on` here will have the `init` process start the program in the second field, `getty`, which will prompt for a login. If you put `off` in the fourth field, there will be no `getty`, and hence no logins on the port.

So, naturally, you want an `on` in this field. Here again is the `/etc/ttys` file. We have turned each port `on`.

```
ttyd1  "/usr/libexec/getty std.38400"  wy50  on secure
ttyd5  "/usr/libexec/getty std.19200"  vt100 on
```

## Specifying Secure Ports

We have arrived at the last field (well, almost: there is an optional window specifier, but we will ignore that). The last field tells whether the port is secure.

What does “secure” mean?

It means that the root account (or any account with a user ID of 0) may login on the port. Insecure ports do not allow root to login.

How do you use secure and insecure ports?

By marking a port as insecure, the terminal to which it is connected will not allow root to login. People who know the root password to your FreeBSD system will first have to login using a regular user account. To gain superuser privileges, they will then have to use the `su` command.

Because of this, you will have two records to help track down possible compromises of root privileges: both the `login` and the `su` command make records in the system log (and logins are also recorded in the `wtmp` file).

By marking a port as secure, the terminal will allow root in. People who know the root password will just login as root. You will not have the potentially useful `login` and `su` command records.

Which should you use?

Just use “insecure.” Use “insecure” *even* for terminals *not* in public user areas or behind locked doors. It is quite easy to login and use `su` if you need superuser privileges.

Here finally are the completed entries in the `/etc/ttys` file, with comments added to describe where the terminals are:

```
ttyd1  "/usr/libexec/getty std.38400"  wy50  on insecure # Kitchen
ttyd5  "/usr/libexec/getty std.19200"  vt100 on insecure # Guest bathroom
```

### Force `init` to Reread `/etc/ttys`

When you boot FreeBSD, the first process, `init`, will read the `/etc/ttys` file and start the programs listed for each enabled port to prompt for logins.

After you edit `/etc/ttys`, you do not want to have to reboot your system to get `init` to see the changes. So, `init` will reread `/etc/ttys` if it receives a `SIGHUP` (hangup) signal.

So, after you have saved your changes to `/etc/ttys`, send `SIGHUP` to `init` by typing:

```
# kill -HUP 1
```

(The `init` process *always* has process ID 1.)

If everything is set up correctly, all cables are in place, and the terminals are powered up, you should see login prompts. Your terminals are ready for their first logins!

## Debugging your connection

Even with the most meticulous attention to detail, something could still go wrong while setting up a terminal. Here is a list of symptoms and some suggested fixes.

No login prompt appears

Make sure the terminal is plugged in and powered up. If it is a personal computer acting as a terminal, make sure it is running terminal emulation software on the correct serial port.

Make sure the cable is connected firmly to both the terminal and the FreeBSD computer. Make sure it is the right kind of cable.

Make sure the terminal and FreeBSD agree on the bps rate and parity settings. If you have a video display terminal, make sure the contrast and brightness controls are turned up. If it is a printing terminal, make sure paper and ink are in good supply.

Make sure that a `getty` process is running and serving the terminal. Type

```
#
ps -axww|grep getty
```

to get a list of running `getty` processes. You should see an entry for the terminal. For example, the display

```
22189  dl  Is+   0:00.03 /usr/libexec/getty std.38400 ttyd1
```

shows that a `getty` is running on the second serial port `ttyd1` and is using the `std.38400` entry in `/etc/gettytab`.

If no `getty` process is running, make sure you have enabled the port in `/etc/ttys`. Make sure you have run `kill -HUP 1`.

Garbage appears instead of a login prompt

Make sure the terminal and FreeBSD agree on the bps rate and parity settings. Check the `getty` processes to make sure the correct `getty` type is in use. If not, edit `/etc/ttys` and run `kill -HUP 1`.

Characters appear doubled; the password appears when typed

Switch the terminal (or the terminal emulation software) from “half duplex” or “local echo” to “full duplex.”

## Dialin Service

*Contributed by Guy Helmer <ghelmer@cs.iastate.edu>.*

This document provides suggestions for configuring a FreeBSD system to handle dialup modems. This document is written based on the author’s experience with FreeBSD versions 1.0, 1.1, and 1.1.5.1 (and experience with dialup modems on other UNIX-like operating systems); however, this document may not answer all of your questions or provide examples specific enough to your environment. The author cannot be responsible if you damage your system or lose data due to attempting to follow the suggestions here.

## Prerequisites

To begin with, the author assumes you have some basic knowledge of FreeBSD. You need to have FreeBSD installed, know how to edit files in a UNIX-like environment, and how to look up manual

pages on the system. As discussed below, you will need certain versions of FreeBSD, and knowledge of some terminology & modem and cabling.

## FreeBSD Version

First, it is assumed that you are using FreeBSD version 1.1 or higher (including versions 2.x). FreeBSD version 1.0 included two different serial drivers, which complicates the situation. Also, the serial device driver (`sio`) has improved in every release of FreeBSD, so more recent versions of FreeBSD are assumed to have better and more efficient drivers than earlier versions.

## Terminology

A quick rundown of terminology:

bps

Bits per Second — the rate at which data is transmitted

DTE

Data Terminal Equipment — for example, your computer

DCE

Data Communications Equipment — your modem

RS-232

EIA standard for serial communications via hardware

If you need more information about these terms and data communications in general, the author remembers reading that *The RS-232 Bible* (anybody have an ISBN?) is a good reference.

When talking about communications data rates, the author does not use the term “baud”. Baud refers to the number of electrical state transitions that may be made in a period of time, while “bps” (bits per second) is the “correct” term to use (at least it does not seem to bother the curmudgeons quite a much).

## External vs. Internal Modems

External modems seem to be more convenient for dialup, because external modems often can be semi-permanently configured via parameters stored in non-volatile RAM and they usually provide

lighted indicators that display the state of important RS-232 signals. Blinking lights impress visitors, but lights are also very useful to see whether a modem is operating properly.

Internal modems usually lack non-volatile RAM, so their configuration may be limited only to setting DIP switches. If your internal modem has any signal indicator lights, it is probably difficult to view the lights when the system's cover is in place.

## Modems and Cables

A background knowledge of these items is assumed

- You know how to connect your modem to your computer so that the two can communicate (unless you have an internal modem, which does not need such a cable)
- You are familiar with your modem's command set, or know where to look up needed commands
- You know how to configure your modem (probably via a terminal communications program) so you can set the non-volatile RAM parameters

The first, connecting your modem, is usually simple — most straight-through serial cables work without any problems. You need to have a cable with appropriate connectors (DB-25 or DB-9, male or female) on each end, and the cable must be a DCE-to-DTE cable with these signals wired:

- Transmitted Data (SD)
- Received Data (RD)
- Request to Send (RTS)
- Clear to Send (CTS)
- Data Set Ready (DSR)
- Data Terminal Ready (DTR)
- Carrier Detect (CD)
- Signal Ground (SG)

FreeBSD needs the RTS and CTS signals for flow-control at speeds above 2400bps, the CD signal to detect when a call has been answered or the line has been hung up, and the DTR signal to reset the modem after a session is complete. Some cables are wired without all of the needed signals, so if you have problems, such as a login session not going away when the line hangs up, you may have a problem with your cable.

The second prerequisite depends on the modem(s) you use. If you do not know your modem's command set by heart, you will need to have the modem's reference book or user's guide handy. Sample

commands for USR Sportster 14,400 external modems will be given, which you may be able to use as a reference for your own modem's commands.

Lastly, you will need to know how to setup your modem so that it will work well with FreeBSD. Like other UNIX-like operating systems, FreeBSD uses the hardware signals to find out when a call has been answered or a line has been hung up and to hangup and reset the modem after a call. FreeBSD avoids sending commands to the modem or watching for status reports from the modem. If you are familiar with connecting modems to PC-based bulletin board systems, this may seem awkward.

## Serial Interface Considerations

FreeBSD supports NS8250-, NS16450-, NS16550-, and NS16550A-based EIA RS-232C (CCITT V.24) communications interfaces. The 8250 and 16450 devices have single-character buffers. The 16550 device provides a 16-character buffer, which allows for better system performance. (Bugs in plain 16550's prevent the use of the 16-character buffer, so use 16550A's if possible). Because single-character-buffer devices require more work by the operating system than the 16-character-buffer devices, 16550A-based serial interface cards are much preferred. If the system has many active serial ports or will have a heavy load, 16550A-based cards are better for low-error-rate communications.

## Quick Overview

Here is the process that FreeBSD follows to accept dialup logins. A `getty` process, spawned by `init`, patiently waits to open the assigned serial port (`/dev/ttyd0`, for our example). The command `ps ax` might show this:

```
4850 ?? I      0:00.09 /usr/libexec/getty V19200 ttyd0
```

When a user dials the modem's line and the modems connect, the CD line is asserted by the modem. The kernel notices that carrier has been detected and completes `getty`'s open of the port. `getty` sends a `login:` prompt at the specified initial line speed. `getty` watches to see if legitimate characters are received, and, in a typical configuration, if it finds junk (probably due to the modem's connection speed being different than `getty`'s speed), `getty` tries adjusting the line speeds until it receives reasonable characters.

We hope `getty` finds the correct speed and the user sees a `login:` prompt. After the user enters his/her login name, `getty` executes `/usr/bin/login`, which completes the login by asking for the user's password and then starting the user's shell.

Let's dive into the configuration...

## Kernel Configuration

FreeBSD kernels typically come prepared to search for four serial ports, known in the PC-DOS world as COM1:, COM2:, COM3:, and COM4:. FreeBSD can presently also handle “dumb” multiport serial interface cards, such as the Boca Board 1008 and 2016 (please see the manual page `sio(4)` for kernel configuration information if you have a multiport serial card). The default kernel only looks for the standard COM ports, though.

To see if your kernel recognizes any of your serial ports, watch for messages while the kernel is booting, or use the `/sbin/dmesg` command to replay the kernel’s boot messages. In particular, look for messages that start with the characters `sio`. Hint: to view just the messages that have the word `sio`, use the command:

```
# /sbin/dmesg | grep 'sio'
```

For example, on a system with four serial ports, these are the serial-port specific kernel boot messages:

```
sio0 at 0x3f8-0x3ff irq 4 on isa
sio0: type 16550A
sio1 at 0x2f8-0x2ff irq 3 on isa
sio1: type 16550A
sio2 at 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
sio3 at 0x2e8-0x2ef irq 9 on isa
sio3: type 16550A
```

If your kernel does not recognize all of your serial ports, you will probably need to configure a custom FreeBSD kernel for your system.

Please see the BSD System Manager’s Manual chapter on “Building Berkeley Kernels with Config” [the source for which is in `/usr/src/share/doc/smm`] and “FreeBSD Configuration Options” [in `/sys/conf/options` and in `/sys/arch/conf/options.arch`, with `arch` for example being `i386`] for more information on configuring and building kernels. You may have to unpack the kernel source distribution if have not installed the system sources already (`srcdist/srcsys.??` in FreeBSD 1.1, `srcdist/sys.??` in FreeBSD 1.1.5.1, or the entire source distribution in FreeBSD 2.0) to be able to configure and build kernels.

Create a kernel configuration file for your system (if you have not already) by `cd`ing to `/sys/i386/conf`. Then, if you are creating a new custom configuration file, copy the file `GENERICAH` (or `GENERICBT`, if you have a BusTek SCSI controller on FreeBSD 1.x) to `YOURSYS`, where `YOURSYS` is the name of your system, but in upper-case letters. Edit the file, and change the device lines:

```
device sio0 at isa? port "IO_COM1" tty irq 4 vector siointr
device sio1 at isa? port "IO_COM2" tty irq 3 vector siointr
device sio2 at isa? port "IO_COM3" tty irq 5 vector siointr
```

```
device sio3      at isa? port "IO_COM4" tty irq 9 vector siointr
```

You can comment-out or completely remove lines for devices you do not have. If you have a multiport serial board, such as the Boca Board BB2016, please see the `sio(4)` man page for complete information on how to write configuration lines for multiport boards. Be careful if you are using a configuration file that was previously used for a different version of FreeBSD because the device flags have changed between versions.

**Note:** `port "IO_COM1"` is a substitution for `port 0x3f8`, `IO_COM2` is `0x2f8`, `IO_COM3` is `0x3e8`, and `IO_COM4` is `0x2e8`, which are fairly common port addresses for their respective serial ports; interrupts 4, 3, 5, and 9 are fairly common interrupt request lines. Also note that regular serial ports *cannot* share interrupts on ISA-bus PCs (multiport boards have on-board electronics that allow all the 16550A's on the board to share one or two interrupt request lines).

When you are finished adjusting the kernel configuration file, use the program `config` as documented in “Building Berkeley Kernels with Config” and the `config(8)` manual page to prepare a kernel building directory, then build, install, and test the new kernel.

## Device Special Files

Most devices in the kernel are accessed through “device special files”, which are located in the `/dev` directory. The `sio` devices are accessed through the `/dev/ttyd?` (dial-in) and `/dev/cua0?` (call-out) devices. On FreeBSD version 1.1.5 and higher, there are also initialization devices (`/dev/ttyid?` and `/dev/cuai0?`) and locking devices (`/dev/ttyld?` and `/dev/cual0?`). The initialization devices are used to initialize communications port parameters each time a port is opened, such as `crtsets` for modems which use CTS/RTS signaling for flow control. The locking devices are used to lock flags on ports to prevent users or programs changing certain parameters; see the manual pages `termios(4)`, `sio(4)`, and `stty(1)` for information on the terminal settings, locking & initializing devices, and setting terminal options, respectively.

### Making Device Special Files

A shell script called `MAKEDEV` in the `/dev` directory manages the device special files. (The manual page for `MAKEDEV(8)` on FreeBSD 1.1.5 is fairly bogus in its discussion of COM ports, so ignore it.) To use `MAKEDEV` to make dialup device special files for COM1: (port 0), `cd` to `/dev` and issue the command `MAKEDEV ttyd0`. Likewise, to make dialup device special files for COM2: (port 1), use `MAKEDEV ttyd1`.

`MAKEDEV` not only creates the `/dev/ttyd?` device special files, but also creates the `/dev/cua0?` (and all of the initializing and locking special files under FreeBSD 1.1.5 and up) and removes the hardwired

terminal special file `/dev/tty0?`, if it exists.

After making new device special files, be sure to check the permissions on the files (especially the `/dev/cua*` files) to make sure that only users who should have access to those device special files can read & write on them — you probably do not want to allow your average user to use your modems to dialout. The default permissions on the `/dev/cua*` files should be sufficient:

```
crw-rw---  1 uucp    dialer    28, 129 Feb 15 14:38 /dev/cua01
crw-rw---  1 uucp    dialer    28, 161 Feb 15 14:38 /dev/cuai01
crw-rw---  1 uucp    dialer    28, 193 Feb 15 14:38 /dev/cual01
```

These permissions allow the user `uucp` and users in the group `dialer` to use the call-out devices.

## Configuration Files

There are three system configuration files in the `/etc` directory that you will probably need to edit to allow dialup access to your FreeBSD system. The first, `/etc/gettytab`, contains configuration information for the `/usr/libexec/getty` daemon. Second, `/etc/ttys` holds information that tells `/sbin/init` what `tty` devices should have `getty` processes running on them. Lastly, you can place port initialization commands in the `/etc/rc.serial` script if you have FreeBSD 1.1.5.1 or higher; otherwise, you can initialize ports in the `/etc/rc.local` script.

There are two schools of thought regarding dialup modems on UNIX. One group likes to configure their modems and system so that no matter at what speed a remote user dials in, the local computer-to-modem RS-232 interface runs at a locked speed. The benefit of this configuration is that the remote user always sees a system login prompt immediately. The downside is that the system does not know what a user's true data rate is, so full-screen programs like Emacs will not adjust their screen-painting methods to make their response better for slower connections.

The other school configures their modems' RS-232 interface to vary its speed based on the remote user's connection speed. For example, V.32bis (14.4 Kbps) connections to the modem might make the modem run its RS-232 interface at 19.2 Kbps, while 2400 bps connections make the modem's RS-232 interface run at 2400 bps. Because `getty` does not understand any particular modem's connection speed reporting, `getty` gives a `login:` message at an initial speed and watches the characters that come back in response. If the user sees junk, it is assumed that they know they should press the `<Enter>` key until they see a recognizable prompt. If the data rates do not match, `getty` sees anything the user types as "junk", tries going to the next speed and gives the `login:` prompt again. This procedure can continue ad nauseum, but normally only takes a keystroke or two before the user sees a good prompt. Obviously, this login sequence does not look as clean as the former "locked-speed" method, but a user on a low-speed connection should receive better interactive response from full-screen programs.

The author will try to give balanced configuration information, but is biased towards having the modem's data rate follow the connection rate.

### **/etc/gettytab**

`/etc/gettytab` is a `termcap(5)`-style file of configuration information for `getty(8)`. Please see the `gettytab(5)` manual page for complete information on the format of the file and the list of capabilities.

### **Locked-Speed Config**

If you are locking your modem's data communications rate at a particular speed, you probably will not need to make any changes to `/etc/gettytab`.

### **Matching-Speed Config**

You will need to setup an entry in `/etc/gettytab` to give `getty` information about the speeds you wish to use for your modem. If you have a 2400 bps modem, you can probably use the existing `D2400` entry. This entry already exists in the FreeBSD 1.1.5.1 `gettytab` file, so you do not need to add it unless it is missing under your version of FreeBSD:

```
#
# Fast dialup terminals, 2400/1200/300 rotary (can start either way)
#
D2400|d2400|Fast-Dial-2400:\
      :nx=D1200:tc=2400-baud:
3|D1200|Fast-Dial-1200:\
      :nx=D300:tc=1200-baud:
5|D300|Fast-Dial-300:\
      :nx=D2400:tc=300-baud:
```

If you have a higher speed modem, you will probably need to add an entry in `/etc/gettytab`; here is an entry you could use for a 14.4 Kbps modem with a top interface speed of 19.2 Kbps:

```
#
# Additions for a V.32bis Modem
#
um|V300|High Speed Modem at 300,8-bit:\
      :nx=V19200:tc=std.300:
un|V1200|High Speed Modem at 1200,8-bit:\
      :nx=V300:tc=std.1200:
uo|V2400|High Speed Modem at 2400,8-bit:\
      :nx=V1200:tc=std.2400:
up|V9600|High Speed Modem at 9600,8-bit:\
```

```

        :nx=V2400:tc=std.9600:
uq|V19200|High Speed Modem at 19200,8-bit:\
        :nx=V9600:tc=std.19200:

```

On FreeBSD 1.1.5 and later, this will result in 8-bit, no parity connections. Under FreeBSD 1.1, add `:np:` parameters to the `std.xxx` entries at the top of the file for 8 bits, no parity; otherwise, the default is 7 bits, even parity.

The example above starts the communications rate at 19.2 Kbps (for a V.32bis connection), then cycles through 9600 bps (for V.32), 2400 bps, 1200 bps, 300 bps, and back to 19.2 Kbps. Communications rate cycling is implemented with the `nx=` (“next table”) capability. Each of the lines uses a `tc=` (“table continuation”) entry to pick up the rest of the “standard” settings for a particular data rate.

If you have a 28.8 Kbps modem and/or you want to take advantage of compression on a 14.4 Kbps modem, you need to use a higher communications rate than 19.2 Kbps. Here is an example of a `gettytab` entry starting a 57.6 Kbps:

```

#
# Additions for a V.32bis or V.34 Modem
# Starting at 57.6 Kbps
#
vm|VH300|Very High Speed Modem at 300,8-bit:\
        :nx=VH57600:tc=std.300:
vn|VH1200|Very High Speed Modem at 1200,8-bit:\
        :nx=VH300:tc=std.1200:
vo|VH2400|Very High Speed Modem at 2400,8-bit:\
        :nx=VH1200:tc=std.2400:
vp|VH9600|Very High Speed Modem at 9600,8-bit:\
        :nx=VH2400:tc=std.9600:
vq|VH57600|Very High Speed Modem at 57600,8-bit:\
        :nx=VH9600:tc=std.57600:

```

If you have a slow CPU or a heavily loaded system and you do not have 16550A-based serial ports, you may receive `sio` “silo” errors at 57.6 Kbps.

### **`/etc/ttys`**

`/etc/ttys` is the list of `ttys` for `init` to monitor. `/etc/ttys` also provides security information to `login` (user `root` may only login on `ttys` marked `secure`). See the manual page for `ttys(5)` for more information.

You will need to either modify existing lines in `/etc/ttys` or add new lines to make `init` run `getty` processes automatically on your new dialup ports. The general format of the line will be the same, whether you are using a locked-speed or matching-speed configuration:

```
ttyd0  "/usr/libexec/getty xxx"  dialup on
```

The first item in the above line is the device special file for this entry — `ttyd0` means `/dev/ttyd0` is the file that this `getty` will be watching. The second item, `"/usr/libexec/getty xxx"` (`xxx` will be replaced by the initial `gettytab` capability) is the process `init` will run on the device. The third item, `dialup`, is the default terminal type. The fourth parameter, `on`, indicates to `init` that the line is operational. There can be a fifth parameter, `secure`, but it should only be used for terminals which are physically secure (such as the system console).

The default terminal type (`dialup` in the example above) may depend on local preferences. `dialup` is the traditional default terminal type on dialup lines so that users may customize their login scripts to notice when the terminal is `dialup` and automatically adjust their terminal type. However, the author finds it easier at his site to specify `vt102` as the default terminal type, since the users just use VT102 emulation on their remote systems.

After you have made changes to `/etc/ttys`, you may send the `init` process a HUP signal to re-read the file. You can use the command

```
# kill -1
  1
```

to send the signal. If this is your first time setting up the system, though, you may want to wait until your modem(s) are properly configured and connected before signaling `init`.

### Locked-Speed Config

For a locked-speed configuration, your `ttys` entry needs to have a fixed-speed entry provided to `getty`. For a modem whose port speed is locked at 19.2 Kbps, the `ttys` entry might look like this:

```
ttyd0  "/usr/libexec/getty std.19200"  dialup on
```

If your modem is locked at a different data rate, substitute the appropriate name for the `std.speed` entry for `std.19200` from `/etc/gettytab` for your modem's data rate.

### Matching-Speed Config

In a matching-speed configuration, your `ttys` entry needs to reference the appropriate beginning "auto-baud" (sic) entry in `/etc/gettytab`. For example, if you added the above suggested entry for a

matching-speed modem that starts at 19.2 Kbps (the `gettytab` entry containing the `V19200` starting point), your `ttys` entry might look like this:

```
ttyd0  "/usr/libexec/getty V19200"  dialup on
```

### **`/etc/rc.serial` Or `/etc/rc.local`**

High-speed modems, like V.32, V.32bis, and V.34 modems, need to use hardware (RTS/CTS) flow control. You can add `stty` commands to `/etc/rc.serial` on FreeBSD 1.1.5.1 and up, or `/etc/rc.local` on FreeBSD 1.1, to set the hardware flow control flag in the FreeBSD kernel for the modem ports.

For example, on a sample FreeBSD 1.1.5.1 system, `/etc/rc.serial` reads:

```
#!/bin/sh
#
# Serial port initial configuration

stty -f /dev/ttyid1 crtscts
stty -f /dev/cuai01 crtscts
```

This sets the `termios` flag `crtscts` on serial port #1's (COM2:) dialin and dialout initialization devices.

On an old FreeBSD 1.1 system, these entries were added to `/etc/rc.local` to set the `crtscts` flag on the devices:

```
# Set serial ports to use RTS/CTS flow control
stty -f /dev/ttyd0 crtscts
stty -f /dev/ttyd1 crtscts
stty -f /dev/ttyd2 crtscts
stty -f /dev/ttyd3 crtscts
```

Since there is no initialization device special file on FreeBSD 1.1, one has to just set the flags on the sole device special file and hope the flags are not cleared by a miscreant.

## **Modem Settings**

If you have a modem whose parameters may be permanently set in non-volatile RAM, you will need to use a terminal program (such as `Telx` under PC-DOS or `tip` under FreeBSD) to set the parameters.

Connect to the modem using the same communications speed as the initial speed `getty` will use and configure the modem's non-volatile RAM to match these requirements:

- CD asserted when connected
- DTR asserted for operation; dropping DTR hangs up line & resets modem
- CTS transmitted data flow control
- Disable XON/XOFF flow control
- RTS received data flow control
- Quiet mode (no result codes)
- No command echo

Please read the documentation for your modem to find out what commands and/or DIP switch settings you need to give it.

For example, to set the above parameters on a USRobotics Sportster 14,400 external modem, one could give these commands to the modem:

```
ATZ
AT&C1&D2&H1&I0&R2&W
```

You might also want to take this opportunity to adjust other settings in the modem, such as whether it will use V.42bis and/or MNP5 compression.

The USR Sportster 14,400 external modem also has some DIP switches that need to be set; for other modems, perhaps you can use these settings as an example:

- Switch 1: UP — DTR Normal
- Switch 2: Do not care (Verbal Result Codes/Numeric Result Codes)
- Switch 3: UP — Suppress Result Codes
- Switch 4: DOWN — No echo, offline commands
- Switch 5: UP — Auto Answer
- Switch 6: UP — Carrier Detect Normal
- Switch 7: UP — Load NVRAM Defaults
- Switch 8: Do not care (Smart Mode/Dumb Mode)

Result codes should be disabled/suppressed for dialup modems to avoid problems that can occur if `getty` mistakenly gives a `login:` prompt to a modem that is in command mode and the modem echoes

the command or returns a result code. I have heard this sequence can result in an extended, silly conversation between `getty` and the modem.

### Locked-speed Config

For a locked-speed configuration, you will need to configure the modem to maintain a constant modem-to-computer data rate independent of the communications rate. On a USR Sportster 14,400 external modem, these commands will lock the modem-to-computer data rate at the speed used to issue the commands:

```
ATZ
AT&B1&W
```

### Matching-speed Config

For a variable-speed configuration, you will need to configure your modem to adjust its serial port data rate to match the incoming call rate. On a USR Sportster 14,400 external modem, these commands will lock the modem's error-corrected data rate to the speed used to issue the commands, but allow the serial port rate to vary for non-error-corrected connections:

```
ATZ
AT&B2&W
```

### Checking the Modem's Configuration

Most high-speed modems provide commands to view the modem's current operating parameters in a somewhat human-readable fashion. On the USR Sportster 14,400 external modems, the command `ATI5` displays the settings that are stored in the non-volatile RAM. To see the true operating parameters of the modem (as influenced by the USR's DIP switch settings), use the commands `ATZ` and then `ATI4`.

If you have a different brand of modem, check your modem's manual to see how to double-check your modem's configuration parameters.

## Troubleshooting

Here are a few steps you can follow to check out the dialup modem on your system.

## Checking out the FreeBSD system

Hook up your modem to your FreeBSD system, boot the system, and, if your modem has status indication lights, watch to see whether the modem's DTR indicator lights when the `login:` prompt appears on the system's console — if it lights up, that should mean that FreeBSD has started a `getty` process on the appropriate communications port and is waiting for the modem to accept a call.

If the DTR indicator doesn't light, login to the FreeBSD system through the console and issue a `ps ax` to see if FreeBSD is trying to run a `getty` process on the correct port. You should see a lines like this among the processes displayed:

```
114 ?? I      0:00.10 /usr/libexec/getty V19200 ttyd0
115 ?? I      0:00.10 /usr/libexec/getty V19200 ttyd1
```

If you see something different, like this:

```
114 d0 I      0:00.10 /usr/libexec/getty V19200 ttyd0
```

and the modem has not accepted a call yet, this means that `getty` has completed its open on the communications port. This could indicate a problem with the cabling or a mis-configured modem, because `getty` should not be able to open the communications port until CD (carrier detect) has been asserted by the modem.

If you do not see any `getty` processes waiting to open the desired `ttyd?` port, double-check your entries in `/etc/ttys` to see if there are any mistakes there. Also, check the log file `/var/log/messages` to see if there are any log messages from `init` or `getty` regarding any problems. If there are any messages, triple-check the configuration files `/etc/ttys` and `/etc/gettytab`, as well as the appropriate device special files `/dev/ttyd?`, for any mistakes, missing entries, or missing device special files.

## Try Dialing In

Try dialing into the system; be sure to use 8 bits, no parity, 1 stop bit on the remote system. If you do not get a prompt right away, or get garbage, try pressing `<Enter>` about once per second. If you still do not see a `login:` prompt after a while, try sending a `BREAK`. If you are using a high-speed modem to do the dialing, try dialing again after locking the dialing modem's interface speed (via `AT&B1` on a USR Sportster, for example).

If you still cannot get a `login:` prompt, check `/etc/gettytab` again and double-check that

- The initial capability name specified in `/etc/ttys` for the line matches a name of a capability in `/etc/gettytab`
- Each `nx=` entry matches another `gettytab` capability name

- Each `tc=` entry matches another `gettytab` capability name

If you dial but the modem on the FreeBSD system will not answer, make sure that the modem is configured to answer the phone when DTR is asserted. If the modem seems to be configured correctly, verify that the DTR line is asserted by checking the modem's indicator lights (if it has any).

If you have gone over everything several times and it still does not work, take a break and come back to it later. If it still does not work, perhaps you can send an electronic mail message to the FreeBSD general questions mailing list <[freebsd-questions@FreeBSD.ORG](mailto:freebsd-questions@FreeBSD.ORG)> describing your modem and your problem, and the good folks on the list will try to help.

## Acknowledgments

Thanks to these people for comments and advice:

Sean Kelly <[kelly@fsl.noaa.gov](mailto:kelly@fsl.noaa.gov)>  
for a number of good suggestions

## Dialout Service

*Information integrated from FAQ.*

The following are tips to getting your host to be able to connect over the modem to another computer. This is appropriate for establishing a terminal session with a remote host.

This is useful to log onto a BBS.

This kind of connection can be extremely helpful to get a file on the Internet if you have problems with PPP. If you need to ftp something and PPP is broken, use the terminal session to ftp it. Then use `zmodem` to transfer it to your machine.

## Why cannot I run `tip` or `cu`?

On your system, the programs `tip` and `cu` are probably executable only by `uucp` and group `dialer`. You can use the group `dialer` to control who has access to your modem or remote systems. Just add yourself to group `dialer`.

Alternatively, you can let everyone on your system run `tip` and `cu` by typing:

```
# chmod 4511 /usr/bin/tip
```

You do not have to run this command for `cu`, since `cu` is just a hard link to `tip`.

## My stock Hayes modem is not supported, what can I do?

Actually, the man page for `tip` is out of date. There is a generic Hayes dialer already built in. Just use `at=hayes` in your `/etc/remote` file.

The Hayes driver is not smart enough to recognize some of the advanced features of newer modems—messages like `BUSY`, `NO DIALTONE`, or `CONNECT 115200` will just confuse it. You should turn those messages off when you use `tip` (using `ATX0&W`).

Also, the dial timeout for `tip` is 60 seconds. Your modem should use something less, or else `tip` will think there is a communication problem. Try `ATS7=45&W`.

Actually, as shipped `tip` does not yet support it fully. The solution is to edit the file `tipconf.h` in the directory `/usr/src/usr.bin/tip/tip`. Obviously you need the source distribution to do this.

Edit the line `#define HAYES 0` to `#define HAYES 1`. Then make and make `install`. Everything works nicely after that.

## How am I expected to enter these AT commands?

Make what is called a “direct” entry in your `/etc/remote` file. For example, if your modem is hooked up to the first serial port, `/dev/cuaa0`, then put in the following line:

```
cuaa0:dv=/dev/cuaa0:br#19200:pa=none
```

Use the highest bps rate your modem supports in the `br` capability. Then, type `tip cuaa0` and you will be connected to your modem.

If there is no `/dev/cuaa0` on your system, do this:

```
# cd /dev
# MAKEDEV cuaa0
```

Or use `cu` as root with the following command:

```
# cu -lline -sspeed
```

*line* is the serial port (e.g. `/dev/cuaa0`) and *speed* is the speed (e.g. `57600`). When you are done entering the AT commands hit `~.` to exit.

## The @ sign for the pn capability does not work!

The @ sign in the phone number capability tells tip to look in `/etc/phones` for a phone number. But the @ sign is also a special character in capability files like `/etc/remote`. Escape it with a backslash:

```
pn=\<@
```

## How can I dial a phone number on the command line?

Put what is called a “generic” entry in your `/etc/remote` file. For example:

```
tip115200|Dial any phone number at 115200 bps:\
      :dv=/dev/cuaa0:br#115200:at=hayes:pa=none:du:
tip57600|Dial any phone number at 57600 bps:\
      :dv=/dev/cuaa0:br#57600:at=hayes:pa=none:du:
```

Then you can things like:

```
# tip -115200 5551234
```

If you prefer `cu` over `tip`, use a generic `cu` entry:

```
cu115200|Use cu to dial any number at 115200bps:\
      :dv=/dev/cuaa1:br#57600:at=hayes:pa=none:du:
```

and type:

```
# cu 5551234 -s 115200
```

## Do I have to type in the bps rate every time I do that?

Put in an entry for `tip1200` or `cu1200`, but go ahead and use whatever bps rate is appropriate with the `br` capability. `tip` thinks a good default is 1200 bps which is why it looks for a `tip1200` entry. You do not have to use 1200 bps, though.

## I access a number of hosts through a terminal server.

Rather than waiting until you are connected and typing `CONNECT <host>` each time, use `tip`'s `cm` capability. For example, these entries in `/etc/remote`:

```

pain|pain.deep13.com|Forrester's machine:\
      :cm=CONNECT pain\n:tc=deep13:
muffin|muffin.deep13.com|Frank's machine:\
      :cm=CONNECT muffin\n:tc=deep13:
deep13:Gizmonics Institute terminal server:\
      :dv=/dev/cua02:br#38400:at=hayes:du:pa=none:pn=5551234:

```

will let you type `tip pain` or `tip muffin` to connect to the hosts `pain` or `muffin`; and `tip deep13` to get to the terminal server.

## Can tip try more than one line for each site?

This is often a problem where a university has several modem lines and several thousand students trying to use them...

Make an entry for your university in `/etc/remote` and use `@` for the `pn` capability:

```

big-university:\
      :pn=\@:tc=dialout
dialout:\
      :dv=/dev/cuaa3:br#9600:at=courier:du:pa=none:

```

Then, list the phone numbers for the university in `/etc/phones`:

```

big-university 5551111
big-university 5551112
big-university 5551113
big-university 5551114

```

`tip` will try each one in the listed order, then give up. If you want to keep retrying, run `tip` in a while loop.

## Why do I have to hit CTRL+P twice to send CTRL+P once?

CTRL+P is the default “force” character, used to tell `tip` that the next character is literal data. You can set the force character to any other character with the `~s` escape, which means “set a variable.”

Type `~sforce=single-char` followed by a newline. *single-char* is any single character. If you leave out *single-char*, then the force character is the nul character, which you can get by typing CTRL+2 or CTRL+SPACE. A pretty good value for *single-char* is SHIFT+CTRL+6, which I have seen only used on some terminal servers.

You can have the force character be whatever you want by specifying the following in your `$HOME/.tiprc` file:

```
force=<single-char>
```

## Suddenly everything I type is in UPPER CASE??

You must have pressed CTRL+A, `tip`'s “raise character,” specially designed for people with broken caps-lock keys. Use `~s` as above and set the variable `raisechar` to something reasonable. In fact, you can set it to the same as the force character, if you never expect to use either of these features.

Here is a sample `.tiprc` file perfect for Emacs users who need to type CTRL+2 and CTRL+A a lot:

```
force=^^
raisechar=^^
```

The ^^ is SHIFT+CTRL+6.

## How can I do file transfers with `tip`?

If you are talking to another UNIX system, you can send and receive files with `~p` (put) and `~t` (take). These commands run `cat` and `echo` on the remote system to accept and send files. The syntax is:

```
~p local-file [remote-file]
```

```
~t remote-file [local-file]
```

There is no error checking, so you probably should use another protocol, like `zmodem`.

## How can I run `zmodem` with `tip`?

To receive files, start the sending program on the remote end. Then, type `~C rz` to begin receiving them locally.

To send files, start the receiving program on the remote end. Then, type `~C sz files` to send them to the remote system.

# Chapter 15. PPP and SLIP

If your connection to the Internet is through a modem, or you wish to provide other people with dialup connections to the Internet using FreeBSD, you have the option of using PPP or SLIP. Furthermore, two varieties of PPP are provided: *user* (sometimes referred to as *ijppp*) and *kernel*. The procedures for configuring both types of PPP, and for setting up SLIP are described in this chapter.

## Setting up User PPP

User PPP was introduced to FreeBSD in release 2.0.5 as an addition to the existing kernel implementation of PPP. So, what is different about this new PPP that warrants its addition? To quote from the manual page:

This is a user process PPP software package. Normally, PPP is implemented as a part of the kernel (e.g. as managed by `pppd`) and it is thus somewhat hard to debug and/or modify its behavior. However, in this implementation PPP is done as a user process with the help of the tunnel device driver (`tun`).

In essence, this means that rather than running a PPP daemon, the `ppp` program can be run as and when desired. No PPP interface needs to be compiled into the kernel, as the program can use the generic tunnel device to get data into and out of the kernel.

From here on out, user `ppp` will be referred to simply as `ppp` unless a distinction needs to be made between it and any other PPP client/server software such as `pppd`. Unless otherwise stated, all commands in this section should be executed as root.

There are a large number of enhancements in version 2 of `ppp`. You can discover what version you have by running `ppp` with no arguments and typing `show version` at the prompt. It is a simple matter to upgrade to the latest version of `ppp` (under any version of FreeBSD) by downloading the latest archive via [www.Awfulhak.org](http://www.Awfulhak.org) (<http://www.Awfulhak.org/ppp.html>).

## Before you start

This document assumes you are in roughly this position:

You have an account with an Internet Service Provider (ISP) which lets you use PPP. Further, you have a modem (or other device) connected and configured correctly which allows you to connect to your ISP.

You are going to need the following information to hand:

- Your ISP's phone number(s).

- Your login name and password. This can be either a regular unix style login/password pair, or a PPP PAP or CHAP login/password pair.
- The IP addresses of one or more nameservers. Normally, you will be given two IP numbers. You *must* have this information for **PPP** version 1.x unless you run your own nameserver. From version 2 onwards, **PPP** supports nameserver address negotiation. If your ISP supports this, then using the command `enable dns` in your config file will tell **PPP** to set the nameservers for you.

The following information may have been supplied by your ISP, but is not strictly necessary:

- The IP address of your ISP's gateway. The gateway is the machine to which you will connect and will be set up as your *default route*. If your ISP hasn't given you this number, we can make one up and your ISP's PPP server will tell us the correct value when we connect.

This IP number is referred to as `HISADDR` by `ppp`.

- Your ISP's netmask. If your ISP hasn't given you this information, you can safely use a netmask of `255.255.255.0`.

If your ISP allocates you a static IP address and hostname then you can enter this information. Otherwise, we simply let the peer assign whatever IP number it sees fit.

If you do not have any of the required information, contact your ISP and make sure they provide it to you.

## Building a ppp ready kernel

As the description states, `ppp` uses the kernel `tun` device. It is necessary to make sure that your kernel has support for this device compiled in.

To check this, go to your kernel compile directory (`/sys/i386/conf` or `/sys/pc98/conf`) and examine your kernel configuration file. It needs to have the line

```
pseudo-device tun 1
```

in it somewhere. The stock `GENERIC` kernel has this as standard, so if you have not installed a custom kernel or you do not have a `/sys` directory, you do not have to change anything.

If your kernel configuration file does not have this line in it, or you need to configure more than one `tun` device (for example, if you are setting up a server and could have 16 dialup `ppp` connections at any one time then you will need to use `16` instead of `1`), then you should add the line, re-compile, re-install and boot the new kernel. Please refer to the Configuring the FreeBSD Kernel section for more information on kernel configuration.

You can check how many tunnel devices your current kernel has by typing the following:

```
# ifconfig -a
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
      inet 200.10.100.1 -> 203.10.100.24 netmask 0xffffffff
tun1: flags=8050<POINTOPOINT,RUNNING,MULTICAST> mtu 576
tun2: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
      inet 203.10.100.1 -> 203.10.100.20 netmask 0xffffffff
tun3: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
```

This case shows four tunnel devices, two of which are currently configured and being used. It should be noted that the `RUNNING` flag above indicates that the interface has been used at some point—it is not an error if your interface does not show up as `RUNNING`.

If you have a kernel without the `tun` device, and you can not rebuild it for some reason, all is not lost. You should be able to dynamically load the code. Refer to the appropriate `modload(8)` and `lkm(4)` pages for further details.

You may also wish to take this opportunity to configure a firewall. Details can be found in the `Firewalls` section.

## Check the tun device

Most users will only require one `tun` device (`/dev/tun0`). If you have used more (i.e., a number other than 1 in the `pseudo-device` line in the kernel configuration file) then alter all references to `tun0` below to reflect whichever device number you are using.

The easiest way to make sure that the `tun0` device is configured correctly is to re-make it. To do this, execute the following commands:

```
# cd /dev
# ./MAKEDEV tun0
```

If you require 16 tunnel devices in your kernel, you will need to create more than just `tun0`:

```
# cd /dev
# ./MAKEDEV tun15
```

Also, to confirm that the kernel is configured correctly, the following command should give the indicated output:

```
# ifconfig tun0
tun0: flags=8050<POINTOPOINT,RUNNING,MULTICAST> mtu 1500
```

The `RUNNING` flag may not yet be set, in which case you will see:

```
# ifconfig tun0
tun0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
```

## Name Resolution Configuration

The resolver is the part of the system that turns IP addresses into hostnames and vice versa. It can be configured to look for maps that describe IP to hostname mappings in one of two places. The first is a file called `/etc/hosts` (man 5 `hosts`). The second is the Internet Domain Name Service (DNS), a distributed data base, the discussion of which is beyond the scope of this document.

This section describes briefly how to configure your resolver.

The resolver is a set of system calls that do the name mappings, but you have to tell them where to find their information. You do this by first editing the file `/etc/host.conf`. Do *not* call this file `/etc/hosts.conf` (note the extra `s`) as the results can be confusing.

### Edit the `/etc/host.conf` file

This file should contain the following two lines (in this order):

```
hosts
bind
```

These instructs the resolver to first look in the file `/etc/hosts`, and then to consult the DNS if the name was not found.

### Edit the `/etc/hosts(5)` file

This file should contain the IP addresses and names of machines on your network. At a bare minimum it should contain entries for the machine which will be running ppp. Assuming that your machine is called `foo.bar.com` with the IP address `10.0.0.1`, `/etc/hosts` should contain:

```
127.0.0.1    localhost
10.0.0.1    foo.bar.com  foo
```

The first line defines the alias `localhost` as a synonym for the current machine. Regardless of your own IP address, the IP address for this line should always be `127.0.0.1`. The second line maps the name `foo.bar.com` (and the shorthand `foo`) to the IP address `10.0.0.1`.

If your provider allocates you a static IP address and name, then use these in place of the `10.0.0.1` entry.

## Edit the `/etc/resolv.conf` file

`/etc/resolv.conf` tells the resolver how to behave. If you are running your own DNS, you may leave this file empty. Normally, you will need to enter the following line(s):

```
nameserver x.x.x.x
nameserver y.y.y.y
domain bar.com
```

The `x.x.x.x` and `y.y.y.y` addresses are those given to you by your ISP. Add as many `nameserver` lines as your ISP provides. The `domain` line defaults to your hostname's domain, and is probably unnecessary. Refer to the `resolv.conf` manual page for details of other possible entries in this file.

If you are running PPP version 2 or greater, the `enable dns` command will tell PPP to request that your ISP confirms the `nameserver` values. If your ISP supplies different addresses (or if there are no `nameserver` lines in `/etc/resolv.conf`), PPP will rewrite the file with the ISP-supplied values.

## PPP Configuration

Both user `ppp` and `pppd` (the kernel level implementation of PPP) use configuration files located in the `/etc/ppp` directory. The sample configuration files provided are a good reference for user `ppp`, so don't delete them.

Configuring `ppp` requires that you edit a number of files, depending on your requirements. What you put in them depends to some extent on whether your ISP allocates IP addresses statically (i.e., you get given one IP address, and always use that one) or dynamically (i.e., your IP address can be different for each PPP session).

### PPP and Static IP addresses

You will need to create a configuration file called `/etc/ppp/ppp.conf`. It should look similar to the example below.

**Note:** Lines that end in a `:` start in the first column, all other lines should be indented as shown using spaces or tabs.

```
1  default:
2      set device /dev/cuaa0
3      set speed 115200
4      set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 5 \\\" ATE1Q0 OK-AT-
OK \\dATDT\\TTIMEOUT 40 CONNECT"
```

```
5 provider:
6   set phone "(0123) 456 7890"
7   set login "TIMEOUT 10 \"\" \"\" gin:-gin: foo word: bar col: ppp"
8   set timeout 300
9   set ifaddr x.x.x.x y.y.y.y 255.255.255.0 0.0.0.0
10  add default HISADDR
11  enable dns
```

Do not include the line numbers, they are just for reference in this discussion.

Line 1:

Identifies the default entry. Commands in this entry are executed automatically when ppp is run.

Line 2:

Identifies the device to which the modem is connected. COM1: is /dev/cuaa0 and COM2: is /dev/cuaa1.

Line 3:

Sets the speed you want to connect at. If 115200 doesn't work (it should with any reasonably new modem), try 38400 instead.

Line 4:

The dial string. User PPP uses an expect-send syntax similar to the chat(8) program. Refer to the manual page for information on the features of this language.

Line 5:

Identifies an entry for a provider called "provider".

Line 6:

Sets the phone number for this provider. Multiple phone numbers may be specified using the : or | character as a separator. The difference between these separators is described in the ppp manual page. To summarize, if you want to rotate through the numbers, use the :. If you want to always attempt to dial the first number first and only use the other numbers if the first number fails, use the |. Always quote the entire set of phone numbers as shown.

## Line 7:

The login string is of the same chat-like syntax as the dial string. In this example, the string works for a service whose login session looks like this:

```
J. Random Provider
login: foo
password: bar
protocol: ppp
```

You will need to alter this script to suit your own needs. When you write this script for the first time, you should enable “chat” logging to ensure that the conversation is going as expected.

If you’re using PAP or CHAP, there will be no login at this point, so your login string can be left blank. See PAP and CHAP authentication for further details.

## Line 8:

Sets the default timeout (in seconds) for the connection. Here, the connection will be closed automatically after 300 seconds of inactivity. If you never want to timeout, set this value to zero.

## Line 9:

Sets the interface addresses. The string `x.x.x.x` should be replaced by the IP address that your provider has allocated to you. The string `y.y.y.y` should be replaced by the IP address that your ISP indicated for their gateway (the machine to which you connect). If your ISP hasn’t given you a gateway address, use `10.0.0.2/0`. If you need to use a “guessed” address, make sure that you create an entry in `/etc/ppp/ppp.linkup` as per the instructions for PPP and Dynamic IP addresses. If this line is omitted, `ppp` cannot run in `-auto` or `-dynamic` mode.

## Line 10:

Adds a default route to your ISP’s gateway. The special word `HISADDR` is replaced with the gateway address specified on line 9. It is important that this line appears after line 9, otherwise `HISADDR` will not yet be initialized.

## Line 11:

This line tells PPP to ask your ISP to confirm that your nameserver addresses are correct. If your ISP supports this facility, PPP can then update `/etc/resolv.conf` with the correct nameserver entries.

It is not necessary to add an entry to `ppp.linkup` when you have a static IP address as your routing table entries are already correct before you connect. You may however wish to create an entry to invoke programs after connection. This is explained later with the `sendmail` example.

Example configuration files can be found in the `/etc/ppp` directory.

## PPP and Dynamic IP addresses

If your service provider does not assign static IP numbers, `ppp` can be configured to negotiate the local and remote addresses. This is done by “guessing” an IP number and allowing `ppp` to set it up correctly using the IP Configuration Protocol (IPCP) after connecting. The `ppp.conf` configuration is the same as PPP and Static IP addresses, with the following change:

```
9      set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.0
```

Again, do not include the line numbers, they are just for reference in this discussion. Indentation of at least one space is required.

Line 9:

The number after the `/` character is the number of bits of the address that `ppp` will insist on. You may wish to use IP numbers more appropriate to your circumstances, but the above example will always work.

The last argument (`0.0.0.0`) tells PPP to negotiate using address `0.0.0.0` rather than `10.0.0.1`. Do not use `0.0.0.0` as the first argument to `set ifaddr` as it prevents PPP from setting up an initial route in `-auto` mode.

If you are running version 1.x of PPP, you will also need to create an entry in `/etc/ppp/ppp.linkup`. `ppp.linkup` is used after a connection has been established. At this point, `ppp` will know what IP addresses should *really* be used. The following entry will delete the existing bogus routes, and create correct ones:

```
1      provider:
2          delete ALL
3          add 0 0 HISADDR
```

Line 1:

On establishing a connection, `ppp` will look for an entry in `ppp.linkup` according to the following rules: First, try to match the same label as we used in `ppp.conf`. If that fails, look for an entry for the IP number of our gateway. This entry is a four-octet IP style label. If we still haven't found an entry, look for the `MYADDR` entry.

Line 2:

This line tells `ppp` to delete all existing routes for the acquired `tun` interface (except the direct route entry).

Line 3:

This line tells `ppp` to add a default route that points to `HISADDR`. `HISADDR` will be replaced with the IP number of the gateway as negotiated in the IPCP.

See the `pmdemand` entry in the files `/etc/ppp/ppp.conf.sample` and `/etc/ppp/ppp.linkup.sample` for a detailed example.

Version 2 of PPP introduces “sticky routes”. Any `add` or `delete` lines that contain `MYADDR` or `HISADDR` will be remembered, and any time the actual values of `MYADDR` or `HISADDR` change, the routes will be re-applied. This removes the necessity of repeating these lines in `ppp.linkup`.

## Receiving incoming calls with `ppp`

This section describes setting up `ppp` in a server role.

When you configure `ppp` to receive incoming calls on a machine connected to a LAN, you must decide if you wish to forward packets to the LAN. If you do, you should allocate the peer an IP number from your LAN’s subnet, and use the command

```
enable proxy
```

in your `ppp.conf` file. You should also confirm that the `/etc/rc.conf` file (this file used to be called `/etc/sysconfig`) contains the following:

```
gateway=YES
```

## Which `getty`?

Configuring FreeBSD for Dialup Services provides a good description on enabling dialup services using `getty`.

An alternative to `getty` is `mgetty` (<http://www.leo.org/~doering/mgetty/index.html>), a smarter version of `getty` designed with dialup lines in mind.

The advantages of using `mgetty` is that it actively *talks* to modems, meaning if port is turned off in `/etc/ttys` then your modem won’t answer the phone.

Later versions of `mgetty` (from 0.99beta onwards) also support the automatic detection of PPP streams, allowing your clients script-less access to your server.

Refer to Mgetty and AutoPPP for more information on mgetty.

### PPP permissions

ppp must normally be run as user id 0. If however you wish to allow ppp to run in server mode as a normal user by executing ppp as described below, that user must be given permission to run ppp by adding them to the network group in /etc/group.

You will also need to give them access to one or more sections of the configuration file using the allow command:

```
allow users fred mary
```

If this command is used in the default section, it gives the specified users access to everything.

### Setting up a PPP shell for dynamic-IP users

Create a file called /etc/ppp/ppp-shell containing the following:

```
#!/bin/sh
IDENT=`echo $0 | sed -e 's/^\.*-\(.*\)$/\1/'`
CALLEDAS="$IDENT"
TTY=`tty`

if [ x$IDENT = xdialup ]; then
    IDENT=`basename $TTY`
fi

echo "PPP for $CALLEDAS on $TTY"
echo "Starting PPP for $IDENT"

exec /usr/sbin/ppp -direct $IDENT
```

This script should be executable. Now make a symbolic link called ppp-dialup to this script using the following commands:

```
# ln -s ppp-shell /etc/ppp/ppp-dialup
```

You should use this script as the *shell* for all your dialup ppp users. This is an example from /etc/passwd for a dialup PPP user with username pchilds. (remember don't directly edit the password file, use vipw)

```
pchilds:*:1011:300:Peter Childs PPP:/home/ppp:/etc/ppp/ppp-dialup
```

Create a `/home/ppp` directory that is world readable containing the following 0 byte files

```
-r-r-r-  1 root    wheel          0 May 27 02:23 .hushlogin
-r-r-r-  1 root    wheel          0 May 27 02:22 .rhosts
```

which prevents `/etc/motd` from being displayed.

### Setting up a PPP shell for static-IP users

Create the `ppp-shell` file as above and for each account with statically assigned IPs create a symbolic link to `ppp-shell`.

For example, if you have three dialup customers `fred`, `sam`, and `mary`, that you route class C networks for, you would type the following:

```
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-fred
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-sam
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-mary
```

Each of these users dialup accounts should have their shell set to the symbolic link created above. (ie. `mary`'s shell should be `/etc/ppp/ppp-mary`).

### Setting up `ppp.conf` for dynamic-IP users

The `/etc/ppp/ppp.conf` file should contain something along the lines of

```
default:
    set debug phase lcp chat
    set timeout 0

ttyd0:
    set ifaddr 203.14.100.1 203.14.100.20 255.255.255.255
    enable proxy

ttyd1:
    set ifaddr 203.14.100.1 203.14.100.21 255.255.255.255
    enable proxy
```

**Note:** The indenting is important.

The `default:` section is loaded for each session. For each dialup line enabled in `/etc/ttys` create an entry similar to the one for `ttyd0:` above. Each line should get a unique IP address from your pool of IP addresses for dynamic users.

**Setting up `ppp.conf` for static-IP users**

Along with the contents of the sample `/etc/ppp/ppp.conf` above you should add a section for each of the statically assigned dialup users. We will continue with our `fred`, `sam`, and `mary` example.

```
fred:
    set ifaddr 203.14.100.1 203.14.101.1 255.255.255.255

sam:
    set ifaddr 203.14.100.1 203.14.102.1 255.255.255.255

mary:
    set ifaddr 203.14.100.1 203.14.103.1 255.255.255.255
```

The file `/etc/ppp/ppp.linkup` should also contain routing information for each static IP user if required. The line below would add a route for the `203.14.101.0` class C via the client's ppp link.

```
fred:
    add 203.14.101.0 netmask 255.255.255.0 HISADDR

sam:
    add 203.14.102.0 netmask 255.255.255.0 HISADDR

mary:
    add 203.14.103.0 netmask 255.255.255.0 HISADDR
```

**More on `mgetty`, AutoPPP, and MS extensions***mgetty and AutoPPP*

Configuring and compiling `mgetty` with the `AUTO_PPP` option enabled allows `mgetty` to detect the LCP phase of PPP connections and automatically spawn off a ppp shell. However, since the default login/password sequence does not occur it is necessary to authenticate users using either PAP or CHAP.

This section assumes the user has successfully configured, compiled, and installed a version of `mgetty` with the `AUTO_PPP` option (v0.99beta or later)

Make sure your `/usr/local/etc/mgetty+sendfax/login.config` file has the following in it:

```
/AutoPPP/ - - /etc/ppp/ppp-pap-dialup
```

This will tell `mgetty` to run the `ppp-pap-dialup` script for detected PPP connections.

Create a file called `/etc/ppp/ppp-pap-dialup` containing the following (the file should be executable):

```
#!/bin/sh
exec /usr/sbin/ppp -direct pap$IDENT
```

For each dialup line enabled in `/etc/ttys` create a corresponding entry in `/etc/ppp/ppp.conf`. This will happily co-exist with the definitions we created above.

```
pap:
    enable pap
    set ifaddr 203.14.100.1 203.14.100.20-203.14.100.40
    enable proxy
```

Each user logging in with this method will need to have a username/password in `/etc/ppp/ppp.secret` file, or alternatively add the

```
enable passwdauth
```

option to authenticate users via pap from the `/etc/passwd` file.

If you wish to assign some users a static IP number, you can specify the number as the third argument in `/etc/ppp/ppp.secret`. See `/etc/ppp/ppp.secret.sample` for examples.

### *MS extentions*

It is possible to configure PPP to supply DNS and NetBIOS nameserver addresses on demand.

To enable these extensions with PPP version 1.x, the following lines might be added to the relevant section of `/etc/ppp/ppp.conf`.

```
enable msex
set ns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

And for PPP version 2 and above:

```
accept dns
set dns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

This will tell the clients the primary and secondary name server addresses, and a netbios nameserver host.

In version 2 and above, if the `set dns` line is omitted, PPP will use the values found in `/etc/resolv.conf`.

## PAP and CHAP authentication

Some ISPs set their system up so that the authentication part of your connection is done using either of the PAP or CHAP authentication mechanisms. If this is the case, your ISP will not give a `login:` prompt when you connect, but will start talking PPP immediately.

PAP is less secure than CHAP, but security is not normally an issue here as passwords, although being sent as plain text with PAP, are being transmitted down a serial line only. There's not much room for crackers to "eavesdrop".

Referring back to the PPP and Static IP addresses or PPP and Dynamic IP addresses sections, the following alterations must be made:

```
7      set login
...
12     set authname MyUserName
13     set authkey MyPassword
```

As always, do not include the line numbers, they are just for reference in this discussion. Indentation of at least one space is required.

Line 7:

Your ISP will not normally require that you log into the server if you're using PAP or CHAP. You must therefore disable your "set login" string.

Line 12:

This line specifies your PAP/CHAP user name. You will need to insert the correct value for *MyUserName*.

Line 13:

This line specifies your PAP/CHAP password. You will need to insert the correct value for *MyPassword*. You may want to add an additional line

```
15     accept PAP
```

or

```
15     accept CHAP
```

to make it obvious that this is the intention, but PAP and CHAP are both accepted by default.

## Changing your `ppp` configuration on the fly

It is possible to talk to the `ppp` program while it is running in the background, but only if a suitable diagnostic port has been set up. To do this, add the following line to your configuration:

```
set server /var/run/ppp-tun%d DiagnosticPassword 0177
```

This will tell PPP to listen to the specified unix-domain socket, asking clients for the specified password before allowing access. The `%d` in the name is replaced with the tun device number that is in use.

Once a socket has been set up, the `pppctl(8)` program may be used in scripts that wish to manipulate the running program.

## Final system configuration

You now have `ppp` configured, but there are a few more things to do before it is ready to work. They all involve editing the `/etc/rc.conf` file (was `/etc/sysconfig`).

Working from the top down in this file, make sure the `hostname=` line is set, e.g.:

```
hostname=foo.bar.com
```

If your ISP has supplied you with a static IP address and name, it's probably best that you use this name as your host name.

Look for the `network_interfaces` variable. If you want to configure your system to dial your ISP on demand, make sure the `tun0` device is added to the list, otherwise remove it.

```
network_interfaces="lo0 tun0" ifconfig_tun0=
```

**Note:** The `ifconfig_tun0` variable should be empty, and a file called `/etc/start_if.tun0` should be created. This file should contain the line

```
ppp -auto mysystem
```

This script is executed at network configuration time, starting your `ppp` daemon in automatic mode. If you have a LAN for which this machine is a gateway, you may also wish to use the `-alias` switch. Refer to the manual page for further details.

Set the `router` program to `NO` with the line

```
router_enable=NO          (/etc/rc.conf)
router=NO                 (/etc/sysconfig)
```

It is important that the `routed` daemon is not started (it's started by default) as `routed` tends to delete the default routing table entries created by `ppp`.

It is probably worth your while ensuring that the `sendmail_flags` line does not include the `-q` option, otherwise `sendmail` will attempt to do a network lookup every now and then, possibly causing your machine to dial out. You may try:

```
sendmail_flags="-bd"
```

The upshot of this is that you must force `sendmail` to re-examine the mail queue whenever the `ppp` link is up by typing:

```
# /usr/sbin/sendmail -q
```

You may wish to use the `!bg` command in `ppp.linkup` to do this automatically:

```
1 provider:
2 delete ALL
3 add 0 0 HISADDR
4 !bg sendmail -bd -q30m
```

If you don't like this, it is possible to set up a "dfilter" to block SMTP traffic. Refer to the sample files for further details.

All that is left is to reboot the machine.

After rebooting, you can now either type

```
# ppp
```

and then `dial provider` to start the PPP session, or, if you want `ppp` to establish sessions automatically when there is outbound traffic (and you haven't created the `start_if.tun0` script), type

```
# ppp -auto provider
```

## Summary

To recap, the following steps are necessary when setting up `ppp` for the first time:

Client side:

1. Ensure that the `tun` device is built into your kernel.
2. Ensure that the `tunX` device file is available in the `/dev` directory.

3. Create an entry in `/etc/ppp/ppp.conf`. The `pmdemand` example should suffice for most ISPs.
4. If you have a dynamic IP address, create an entry in `/etc/ppp/ppp.linkup`.
5. Update your `/etc/rc.conf` (or `sysconfig`) file.
6. Create a `start_if.tun0` script if you require demand dialing.

Server side:

1. Ensure that the `tun` device is built into your kernel.
2. Ensure that the `tunX` device file is available in the `/dev` directory.
3. Create an entry in `/etc/passwd` (using the `vipw(8)` program).
4. Create a profile in this users home directory that runs `ppp -direct direct-server` or similar.
5. Create an entry in `/etc/ppp/ppp.conf`. The `direct-server` example should suffice.
6. Create an entry in `/etc/ppp/ppp.linkup`.
7. Update your `/etc/rc.conf` (or `sysconfig`) file.

## Acknowledgments

This section of the handbook was last updated on Monday Aug 10, 1998 by Brian Somers  
<brian@FreeBSD.ORG>

Thanks to the following for their input, comments & suggestions:

Nik Clayton <nik@FreeBSD.ORG>

Dirk-Willem van Gulik <Dirk.vanGulik@jrc.it>

Peter Childs <pjchilds@imforei.apana.org.au>

## Setting up Kernel PPP

*Contributed by Gennady B. Sorokopud <gena@NetVision.net.il>.*

Before you start setting up PPP on your machine make sure that `pppd` is located in `/usr/sbin` and directory `/etc/ppp` exists.

`pppd` can work in two modes:

1. as a “client”, i.e. you want to connect your machine to outside world via PPP serial connection or modem line.
2. as a “server”, i.e. your machine is located on the network and used to connect other computers using PPP.

In both cases you will need to set up an options file (`/etc/ppp/options` or `~/.ppprc` if you have more than one user on your machine that uses PPP).

You also will need some modem/serial software (preferably `kermit`) so you can dial and establish connection with remote host.

## Working as a PPP client

I used the following `/etc/ppp/options` to connect to CISCO terminal server PPP line.

```

rtscts          # enable hardware flow control
modem           # modem control line
noipdefault     # remote PPP server must supply your IP address.
                # if the remote host doesn't send your IP during IPCP
                # negotiation , remove this option
passive        # wait for LCP packets
domain ppp.foo.com      # put your domain name here

:<remote_ip>    # put the IP of remote PPP host here
                # it will be used to route packets via PPP link
                # if you didn't specified the noipdefault option
                # change this line to <local_ip>:<remote_ip>

defaultroute    # put this if you want that PPP server will be your
                # default router

```

To connect:

1. Dial to the remote host using `kermit` (or other modem program) enter your user name and password (or whatever is needed to enable PPP on the remote host)
2. Exit `kermit` (without hanging up the line).
3. enter:

```
# /usr/src/usr.sbin/pppd.new/pppd /dev/tty01 19200
```

Use the appropriate speed and device name.

Now your computer is connected with PPP. If the connection fails for some reasons you can add the debug option to the `/etc/ppp/options` file and check messages on the console to track the problem

Following `/etc/ppp/pppup` script will make all 3 stages automatically:

```
#!/bin/sh
ps ax |grep pppd |grep -v grep
pid=`ps ax |grep pppd |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill ${pid}
fi
ps ax |grep kermit |grep -v grep
pid=`ps ax |grep kermit |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermit, PID=' ${pid}
    kill -9 ${pid}
fi

ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.dial
pppd /dev/tty01 19200
```

`/etc/ppp/kermit.dial` is kermit script that dials and makes all necessary authorization on the remote host. (Example of such script is attached to the end of this document)

Use the following `/etc/ppp/pppdown` script to disconnect the PPP line:

```
#!/bin/sh
pid=`ps ax |grep pppd |grep -v grep|awk '{print $1;}'`
if [ X${pid} != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill -TERM ${pid}
fi

ps ax |grep kermit |grep -v grep
pid=`ps ax |grep kermit |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermit, PID=' ${pid}
    kill -9 ${pid}
fi

/sbin/ifconfig ppp0 down
/sbin/ifconfig ppp0 delete
```

```
kermit -y /etc/ppp/kermit.hup
/etc/ppp/ppptest
```

Check if PPP is still running (/usr/etc/ppp/ppptest):

```
#!/bin/sh
pid=`ps ax| grep pppd |grep -v grep|awk '{print $1;}'`
if [ X${pid} != "X" ] ; then
    echo 'pppd running: PID=' ${pid-NONE}
else
    echo 'No pppd running.'
fi
set -x
netstat -n -I ppp0
ifconfig ppp0
```

Hangs up modem line (/etc/ppp/kermit.hup):

```
set line /dev/tty01 ; put your modem device here
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
echo \13
exit
```

Here is an alternate method using chat instead of kermit.

*Contributed by Robert Huff <rhuff@cybercom.net>.*

The following two files are sufficient to accomplish a pppd connection.

/etc/ppp/options:

/dev/cuaa1 115200

```

crtsets # enable hardware flow control
modem # modem control line
connect "/usr/bin/chat -f /etc/ppp/login.chat.script"
noipdefault # remote PPP server must supply your IP address.
    # if the remote host doesn't send your IP during
    # IPCP negotiation, remove this option
passive # wait for LCP packets
domain <your.domain> # put your domain name here

: # put the IP of remote PPP host here
    # it will be used to route packets via PPP link
    # if you didn't specified the noipdefault option
    # change this line to <local_ip>:<remote_ip>

defaultroute # put this if you want that PPP server will be
    # your default router

```

/etc/ppp/login.chat.script:

(This should actually go into a single line.)

```

ABORT BUSY ABORT 'NO CARRIER' "" AT OK ATDT<phone.number>
CONNECT "" TIMEOUT 10 ogin:-\r-ogin: <login-id>
TIMEOUT 5 sword: <password>

```

Once these are installed and modified correctly, all you need to do is

```
# pppd
```

This sample based primarily on information provided by: Trev Roydhouse  
<Trev.Roydhouse@f401.n711.z3.fidonet.org> and used by permission.

## Working as a PPP server

/etc/ppp/options:

```

crtsets # Hardware flow control
netmask 255.255.255.0 # netmask ( not required )
192.114.208.20:192.114.208.165 # ip's of local and remote hosts
    # local ip must be different from one
    # you assigned to the ethernet ( or other )
    # interface on your machine.
    # remote IP is ip address that will be
    # assigned to the remote machine

```

```

domain ppp.foo.com           # your domain
passive                      # wait for LCP
modem                        # modem line

```

Following `/etc/ppp/pppserv` script will enable ppp server on your machine:

```

#!/bin/sh
ps ax |grep pppd |grep -v grep
pid=`ps ax |grep pppd |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill ${pid}
fi
ps ax |grep kermit |grep -v grep
pid=`ps ax |grep kermit |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermit, PID=' ${pid}
    kill -9 ${pid}
fi

# reset ppp interface
ifconfig ppp0 down
ifconfig ppp0 delete

# enable autoanswer mode
kermit -y /etc/ppp/kermit.ans

# run ppp
pppd /dev/tty01 19200

```

Use this `/etc/ppp/pppservdown` script to stop ppp server:

```

#!/bin/sh
ps ax |grep pppd |grep -v grep
pid=`ps ax |grep pppd |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill ${pid}
fi
ps ax |grep kermit |grep -v grep
pid=`ps ax |grep kermit |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermit, PID=' ${pid}
    kill -9 ${pid}
fi

```

```

ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.noans

```

Following kermit script will enable/disable autoanswer mode on your modem (/etc/ppp/kermit.ans):

```

set line /dev/tty01
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
inp 5 OK
echo \13
out ATS0=1\13 ; change this to out ATS0=0\13 if you want to disable
                ; autoanswer mod

inp 5 OK
echo \13
exit

```

This /etc/ppp/kermit.dial script is used for dialing and authorizing on remote host. You will need to customize it for your needs. Put your login and password in this script, also you will need to change input statement depending on responses from your modem and remote host.

```

;
; put the com line attached to the modem here:
;
set line /dev/tty01
;
; put the modem speed here:
;
set speed 19200
set file type binary ; full 8 bit file xfer

```

```

set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none
set modem hayes
set dial hangup off
set carrier auto           ; Then SET CARRIER if necessary,
set dial display on       ; Then SET DIAL if necessary,
set input echo on
set input timeout proceed
set input case ignore
def \%x 0                  ; login prompt counter
goto slhup

:slcmd                    ; put the modem in command mode
echo Put the modem in command mode.
clear                     ; Clear unread characters from input buffer
pause 1
output +++                ; hayes escape sequence
input 1 OK\13\10         ; wait for OK
if success goto slhup
output \13
pause 1
output at\13
input 1 OK\13\10
if fail goto slcmd       ; if modem doesn't answer OK, try again

:slhup                    ; hang up the phone
clear                     ; Clear unread characters from input buffer
pause 1
echo Hanging up the phone.
output ath0\13           ; hayes command for on hook
input 2 OK\13\10
if fail goto slcmd       ; if no OK answer, put modem in command mode

:sldial                   ; dial the number
pause 1
echo Dialing.
output atdt9,550311\13\10 ; put phone number here
assign \%x 0             ; zero the time counter

```

```

:look
clear                ; Clear unread characters from input buffer
increment \%x        ; Count the seconds
input 1 {CONNECT }
if success goto sllogin
reinput 1 {NO CARRIER\13\10}
if success goto sldial
reinput 1 {NO DIALTONE\13\10}
if success goto slnodial
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 60 goto look
else goto slhup

:sllogin              ; login
assign \%x 0         ; zero the time counter
pause 1
echo Looking for login prompt.

:slloop
increment \%x        ; Count the seconds
clear                ; Clear unread characters from input buffer
output \13
;
; put your expected login prompt here:
;
input 1 {Username: }
if success goto sluid
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 10 goto slloop ; try 10 times to get a login prompt
else goto slhup        ; hang up and start again if 10 failures

:sluid
;
; put your userid here:
;
output ppp-login\13
input 1 {Password: }
;
; put your password here:

```

```

;
output ppp-password\13
input 1 {Entering SLIP mode.}
echo
quit

:slnodial
echo \7No dialtone. Check the telephone line!\7
exit 1

; local variables:
; mode: csh
; comment-start: ";"
; comment-start-skip: ";"
; end:

```

## Setting up a SLIP Client

*Contributed by Satoshi Asami <asami@FreeBSD.ORG> 8 Aug 1995.*

The following is one way to set up a FreeBSD machine for SLIP on a static host network. For dynamic hostname assignments (i.e., your address changes each time you dial up), you probably need to do something much fancier.

First, determine which serial port your modem is connected to. I have a symbolic link to `/dev/modem` from `/dev/cuaa1`, and only use the modem name in my configuration files. It can become quite cumbersome when you need to fix a bunch of files in `/etc` and `.kernrc`'s all over the system!

**Note:** `/dev/cuaa0` is COM1, `cuaa1` is COM2, etc.

Make sure you have

```
pseudo-device    sl      1
```

in your kernel's config file. It is included in the `GENERIC` kernel, so this will not be a problem unless you deleted it.

## Things you have to do only once

1. Add your home machine, the gateway and nameservers to your `/etc/hosts` file. Mine looks like this:

```
127.0.0.1          localhost loghost
136.152.64.181    silvia.HIP.Berkeley.EDU silvia.HIP silvia
136.152.64.1      inr-3.Berkeley.EDU inr-3 slip-gateway
128.32.136.9      ns1.Berkeley.edu ns1
128.32.136.12     ns2.Berkeley.edu ns2
```

By the way, `silvia` is the name of the car that I had when I was back in Japan (it is called 270SX here in U.S.).

2. Make sure you have `hosts` before `bind` in your `/etc/host.conf`. Otherwise, funny things may happen.
3. Edit the file `/etc/rc.conf`. Note that you should edit the file `/etc/sysconfig` instead if you are running FreeBSD previous to version 2.2.2.

1. Set your hostname by editing the line that says:

```
hostname=myname.my.domain
```

You should give it your full Internet hostname.

2. Add `sl0` to the list of network interfaces by changing the line that says:

```
network_interfaces="lo0"
```

to:

```
network_interfaces="lo0 sl0"
```

3. Set the startup flags of `sl0` by adding a line:

```
ifconfig_sl0="inet ${hostname} slip-gateway netmask 0xffffffff00 up"
```

4. Designate the default router by changing the line:

```
defaultrouter=NO
```

to:

```
defaultrouter=slip-gateway
```

4. Make a file `/etc/resolv.conf` which contains:

```
domain HIP.Berkeley.EDU
nameserver 128.32.136.9
nameserver 128.32.136.12
```

As you can see, these set up the nameserver hosts. Of course, the actual domain names and addresses depend on your environment.

5. Set the password for root and toor (and any other accounts that does not have a password). Use `passwd`, do not edit the `/etc/passwd` or `/etc/master.passwd` files!
6. Reboot your machine and make sure it comes up with the correct hostname.

## Making a SLIP connection

1. Dial up, type `slip` at the prompt, enter your machine name and password. The things you need to enter depends on your environment. I use `kermit`, with a script like this:

```
# kermit setup
set modem hayes
set line /dev/modem
set speed 115200
set parity none
set flow rts/cts
set terminal bytesize 8
set file type binary
# The next macro will dial up and login
define slip dial 643-9600, input 10 =>, if failure stop, -
output slip\x0d, input 10 Username:, if failure stop, -
output silvia\x0d, input 10 Password:, if failure stop, -
output ***\x0d, echo \x0aCONNECTED\x0a
```

(of course, you have to change the hostname and password to fit yours). Then you can just type `slip` from the `kermit` prompt to get connected.

**Note:** Leaving your password in plain text anywhere in the filesystem is generally a BAD idea. Do it at your own risk. I am just too lazy.

2. Leave the `kermit` there (you can suspend it by `z`) and as root, type:

```
# slattach -h -c -s 115200 /dev/modem
```

If you are able to `ping` hosts on the other side of the router, you are connected! If it does not work, you might want to try `-a` instead of `-c` as an argument to `slattach`.

## How to shutdown the connection

Type

```
# kill -INT `cat /var/run/slattach.modem.pid`
```

(as root) to kill slattach. Then go back to kermit (fg if you suspended it) and exit from it (q).

The slattach man page says you have to use `ifconfig s10 down` to mark the interface down, but this does not seem to make any difference for me. (`ifconfig s10` reports the same thing.)

Some times, your modem might refuse to drop the carrier (mine often does). In that case, simply start kermit and quit it again. It usually goes out on the second try.

## Troubleshooting

If it does not work, feel free to ask me. The things that people tripped over so far:

- Not using `-c` or `-a` in slattach (I have no idea why this can be fatal, but adding this flag solved the problem for at least one person)
- Using `s10` instead of `s10` (might be hard to see the difference on some fonts).
- Try `ifconfig s10` to see your interface status. I get:

```
# ifconfig s10
s10: flags=10<POINTOPOINT>
    inet 136.152.64.181 -> 136.152.64.1 netmask ffffffff00
```

- Also, `netstat -r` will give the routing table, in case you get the "no route to host" messages from ping. Mine looks like:

```
# netstat -r
Routing tables
Destination      Gate-
way              Flags    Refs     Use  IfaceMTU   Rtt    Netmasks:

(root node)
(root node)

Route Tree for Protocol Family inet:
(root node) =>
default          inr-3.Berkeley.EDU UG          8   224515  s10 -      -
localhost.Berkel localhost.Berkeley UH          5    42127  lo0 -      -
0.438
inr-3.Berkeley.E silvia.HIP.Berkele UH          1         0  s10 -      -
```

```

silvia.HIP.Berke localhost.Berkeley UGH          34 47641234  1o0 -
0.438
(root node)

```

(this is after transferring a bunch of files, your numbers should be smaller).

## Setting up a SLIP Server

*Contributed by Guy Helmer <ghelmer@cs.iastate.edu>. v1.0, 15 May 1995.*

This document provides suggestions for setting up SLIP Server services on a FreeBSD system, which typically means configuring your system to automatically startup connections upon login for remote SLIP clients. The author has written this document based on his experience; however, as your system and needs may be different, this document may not answer all of your questions, and the author cannot be responsible if you damage your system or lose data due to attempting to follow the suggestions here.

This guide was originally written for SLIP Server services on a FreeBSD 1.x system. It has been modified to reflect changes in the pathnames and the removal of the SLIP interface compression flags in early versions of FreeBSD 2.X, which appear to be the only major changes between FreeBSD versions. If you do encounter mistakes in this document, please email the author with enough information to help correct the problem.

## Prerequisites

This document is very technical in nature, so background knowledge is required. It is assumed that you are familiar with the TCP/IP network protocol, and in particular, network and node addressing, network address masks, subnetting, routing, and routing protocols, such as RIP. Configuring SLIP services on a dial-up server requires a knowledge of these concepts, and if you are not familiar with them, please read a copy of either Craig Hunt's *TCP/IP Network Administration* published by O'Reilly & Associates, Inc. (ISBN Number 0-937175-82-X), or Douglas Comer's books on the TCP/IP protocol.

It is further assumed that you have already setup your modem(s) and configured the appropriate system files to allow logins through your modems. If you have not prepared your system for this yet, please see the tutorial for configuring dialup services; if you have a World-Wide Web browser available, browse the list of tutorials at <http://www.freebsd.org/>; otherwise, check the place where you found this document for a document named `dialup.txt` or something similar. You may also want to check the manual pages for `sio(4)` for information on the serial port device driver and `ttys(5)`, `gettytab(5)`, `getty(8)`, & `init(8)` for information relevant to configuring the system to accept logins on modems, and perhaps `stty(1)` for information on setting serial port parameters (such as `local` for directly-connected serial interfaces).

## Quick Overview

In its typical configuration, using FreeBSD as a SLIP server works as follows: a SLIP user dials up your FreeBSD SLIP Server system and logs in with a special SLIP login ID that uses `/usr/sbin/sliplogin` as the special user's shell. The `sliplogin` program browses the file `/etc/sliphome/slip.hosts` to find a matching line for the special user, and if it finds a match, connects the serial line to an available SLIP interface and then runs the shell script `/etc/sliphome/slip.login` to configure the SLIP interface.

### An Example of a SLIP Server Login

For example, if a SLIP user ID were `Shelmerg`, `Shelmerg`'s entry in `/etc/master.passwd` would look something like this (except it would be all on one line):

```
Shelmerg:password:1964:89::0:0:Guy Helmer -
SLIP:/usr/users/Shelmerg:/usr/sbin/sliplogin
```

When `Shelmerg` logs in, `sliplogin` will search `/etc/sliphome/slip.hosts` for a line that had a matching user ID; for example, there may be a line in `/etc/sliphome/slip.hosts` that reads:

```
Shelmerg          dc-slip sl-helmer          0xfffffc00  autocomp
```

`sliplogin` will find that matching line, hook the serial line into the next available SLIP interface, and then execute `/etc/sliphome/slip.login` like this:

```
/etc/sliphome/slip.login 0 19200 Shelmerg dc-slip sl-
helmer 0xfffffc00 autocomp
```

If all goes well, `/etc/sliphome/slip.login` will issue an `ifconfig` for the SLIP interface to which `sliplogin` attached itself (slip interface 0, in the above example, which was the first parameter in the list given to `slip.login`) to set the local IP address (`dc-slip`), remote IP address (`sl-helmer`), network mask for the SLIP interface (`0xfffffc00`), and any additional flags (`autocomp`). If something goes wrong, `sliplogin` usually logs good informational messages via the daemon `syslog` facility, which usually goes into `/var/log/messages` (see the manual pages for `syslogd(8)` and `syslog.conf(5)`, and perhaps check `/etc/syslog.conf` to see to which files `syslogd` is logging).

OK, enough of the examples — let us dive into setting up the system.

## Kernel Configuration

FreeBSD's default kernels usually come with two SLIP interfaces defined (`s10` and `s11`); you can use

`netstat -i` to see whether these interfaces are defined in your kernel.

Sample output from `netstat -i`:

Name	Mtu	Network	Address	Ipkts	Ierrs	Opkts	Oerrs	Coll
ed0	1500	<Link>	0.0.c0.2c.5f.4a	291311	0	174209	0	133
ed0	1500	138.247.224	ivory	291311	0	174209	0	133
lo0	65535	<Link>		79	0	79	0	0
lo0	65535	loop	localhost	79	0	79	0	0
s10*	296	<Link>		0	0	0	0	0
s11*	296	<Link>		0	0	0	0	0

The `s10` and `s11` interfaces shown in `netstat -i`'s output indicate that there are two SLIP interfaces built into the kernel. (The asterisks after the `s10` and `s11` indicate that the interfaces are “down”.)

However, FreeBSD's default kernels do not come configured to forward packets (ie, your FreeBSD machine will not act as a router) due to Internet RFC requirements for Internet hosts (see RFC's 1009 [Requirements for Internet Gateways], 1122 [Requirements for Internet Hosts — Communication Layers], and perhaps 1127 [A Perspective on the Host Requirements RFCs]), so if you want your FreeBSD SLIP Server to act as a router, you will have to edit the `/etc/rc.conf` file (called `/etc/sysconfig` in FreeBSD releases prior to 2.2.2) and change the setting of the `gateway` variable to `YES`. If you have an older system which predates even the `/etc/sysconfig` file, then add the following command:

```
sysctl -w net.inet.ip.forwarding = 1
```

to your `/etc/rc.local` file.

You will then need to reboot for the new settings to take effect.

You will notice that near the end of the default kernel configuration file (`/sys/i386/conf/GENERIC`) is a line that reads:

```
pseudo-device sl 2
```

This is the line that defines the number of SLIP devices available in the kernel; the number at the end of the line is the maximum number of SLIP connections that may be operating simultaneously.

Please refer to *Configuring the FreeBSD Kernel* for help in reconfiguring your kernel.

## Sliplogin Configuration

As mentioned earlier, there are three files in the `/etc/sliphome` directory that are part of the configuration for `/usr/sbin/sliplogin` (see `sliplogin(8)` for the actual manual page for `sliplogin`): `slip.hosts`, which defines the SLIP users & their associated IP addresses; `slip.login`, which

usually just configures the SLIP interface; and (optionally) `slip.logout`, which undoes `slip.login`'s effects when the serial connection is terminated.

## `slip.hosts` Configuration

`/etc/sliphome/slip.hosts` contains lines which have at least four items, separated by whitespace:

- SLIP user's login ID
- Local address (local to the SLIP server) of the SLIP link
- Remote address of the SLIP link
- Network mask

The local and remote addresses may be host names (resolved to IP addresses by `/etc/hosts` or by the domain name service, depending on your specifications in `/etc/host.conf`), and I believe the network mask may be a name that can be resolved by a lookup into `/etc/networks`. On a sample system, `/etc/sliphome/slip.hosts` looks like this:

```
#
# login local-addr      remote-addr      mask              opt1    opt2
#                               (normal,compress,noicmp)
#
Shelmerg dc-slip        sl-helmerg       0xfffffc00       autocomp
```

At the end of the line is one or more of the options.

- `normal` — no header compression
- `compress` — compress headers
- `autocomp` — compress headers if the remote end allows it
- `noicmp` — disable ICMP packets (so any “ping” packets will be dropped instead of using up your bandwidth)

Note that `sliplogin` under early releases of FreeBSD 2 ignored the options that FreeBSD 1.x recognized, so the options `normal`, `compress`, `autocomp`, and `noicmp` had no effect until support was added in FreeBSD 2.2 (unless your `slip.login` script included code to make use of the flags).

Your choice of local and remote addresses for your SLIP links depends on whether you are going to dedicate a TCP/IP subnet or if you are going to use “proxy ARP” on your SLIP server (it is not “true” proxy ARP, but that is the terminology used in this document to describe it). If you are not sure which method to select or how to assign IP addresses, please refer to the TCP/IP books referenced in the `slips-prereqs` section and/or consult your IP network manager.

If you are going to use a separate subnet for your SLIP clients, you will need to allocate the subnet number out of your assigned IP network number and assign each of your SLIP client's IP numbers out of that subnet. Then, you will probably either need to configure a static route to the SLIP subnet via your SLIP server on your nearest IP router, or install `gated` on your FreeBSD SLIP server and configure it to talk the appropriate routing protocols to your other routers to inform them about your SLIP server's route to the SLIP subnet.

Otherwise, if you will use the "proxy ARP" method, you will need to assign your SLIP client's IP addresses out of your SLIP server's Ethernet subnet, and you will also need to adjust your `/etc/sliphome/slip.login` and `/etc/sliphome/slip.logout` scripts to use `arp(8)` to manage the proxy-ARP entries in the SLIP server's ARP table.

### `slip.login` Configuration

The typical `/etc/sliphome/slip.login` file looks like this:

```
#!/bin/sh -
#
#      @(#)slip.login  5.1 (Berkeley) 7/1/90
#
# generic login file for a slip line.  sliplogin invokes this with
# the parameters:
#      1      2      3      4      5      6      7-n
#  slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
```

This `slip.login` file merely `ifconfig`'s the appropriate SLIP interface with the local and remote addresses and network mask of the SLIP interface.

If you have decided to use the "proxy ARP" method (instead of using a separate subnet for your SLIP clients), your `/etc/sliphome/slip.login` file will need to look something like this:

```
#!/bin/sh -
#
#      @(#)slip.login  5.1 (Berkeley) 7/1/90
#
# generic login file for a slip line.  sliplogin invokes this with
# the parameters:
#      1      2      3      4      5      6      7-n
#  slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
```

```
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
# Answer ARP requests for the SLIP client with our Ethernet addr
/usr/sbin/arp -s $5 00:11:22:33:44:55 pub
```

The additional line in this `slip.login`, `arp -s $5 00:11:22:33:44:55 pub`, creates an ARP entry in the SLIP server's ARP table. This ARP entry causes the SLIP server to respond with the SLIP server's Ethernet MAC address whenever another IP node on the Ethernet asks to speak to the SLIP client's IP address.

When using the example above, be sure to replace the Ethernet MAC address (`00:11:22:33:44:55`) with the MAC address of your system's Ethernet card, or your "proxy ARP" will definitely not work! You can discover your SLIP server's Ethernet MAC address by looking at the results of running `netstat -i`; the second line of the output should look something like:

```
ed0  1500  <Link>0.2.c1.28.5f.4a          191923 0  129457  0  116
```

This indicates that this particular system's Ethernet MAC address is `00:02:c1:28:5f:4a` — the periods in the Ethernet MAC address given by `netstat -i` must be changed to colons and leading zeros should be added to each single-digit hexadecimal number to convert the address into the form that `arp(8)` desires; see the manual page on `arp(8)` for complete information on usage.

**Note:** When you create `/etc/sliphome/slip.login` and `/etc/sliphome/slip.logout`, the "execute" bit (ie, `chmod 755 /etc/sliphome/slip.login /etc/sliphome/slip.logout`) must be set, or `sliplogin` will be unable to execute it.

### `slip.logout` Configuration

`/etc/sliphome/slip.logout` is not strictly needed (unless you are implementing "proxy ARP"), but if you decide to create it, this is an example of a basic `slip.logout` script:

```
#!/bin/sh -
#
#      slip.logout

#
# logout file for a slip line.  sliplogin invokes this with
# the parameters:
#      1      2      3      4      5      6      7-n
#  slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 down
```

If you are using “proxy ARP”, you will want to have `/etc/sliphome/slip.logout` remove the ARP entry for the SLIP client:

```
#!/bin/sh -
#
#      @(#)slip.logout

#
# logout file for a slip line.  sliplogin invokes this with
# the parameters:
#      1      2      3      4      5      6      7-n
#  slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 down
# Quit answering ARP requests for the SLIP client
/usr/sbin/arp -d $5
```

The `arp -d $5` removes the ARP entry that the “proxy ARP” `slip.login` added when the SLIP client logged in.

It bears repeating: make sure `/etc/sliphome/slip.logout` has the execute bit set for after you create it (ie, `chmod 755 /etc/sliphome/slip.logout`).

## Routing Considerations

If you are not using the “proxy ARP” method for routing packets between your SLIP clients and the rest of your network (and perhaps the Internet), you will probably either have to add static routes to your closest default router(s) to route your SLIP client subnet via your SLIP server, or you will probably need to install and configure `gated` on your FreeBSD SLIP server so that it will tell your routers via appropriate routing protocols about your SLIP subnet.

### Static Routes

Adding static routes to your nearest default routers can be troublesome (or impossible, if you do not have authority to do so...). If you have a multiple-router network in your organization, some routers, such as Cisco and Proteon, may not only need to be configured with the static route to the SLIP subnet, but also need to be told which static routes to tell other routers about, so some expertise and troubleshooting/tweaking may be necessary to get static-route-based routing to work.

## Running `gated`

An alternative to the headaches of static routes is to install `gated` on your FreeBSD SLIP server and configure it to use the appropriate routing protocols (RIP/OSPF/BGP/EGP) to tell other routers about your SLIP subnet. You can use `gated` from the ports collection or retrieve and build it yourself from the GateD anonymous ftp site (<ftp://ftp.gated.merit.edu/research.and.development/gated/>); I believe the current version as of this writing is `gated-R3_5Alpha_8.tar.Z`, which includes support for FreeBSD “out-of-the-box”. Complete information and documentation on `gated` is available on the Web starting at the Merit GateD Consortium (<http://www.gated.merit.edu/>). Compile and install it, and then write a `/etc/gated.conf` file to configure your `gated`; here is a sample, similar to what the author used on a FreeBSD SLIP server:

```
#
# gated configuration file for dc.dsu.edu; for gated version 3.5alpha5
# Only broadcast RIP information for xxx.xxx.yy out the ed Ethernet interface
#
#
# tracing options
#
traceoptions "/var/tmp/gated.output" replace size 100k files 2 general ;

rip yes {
    interface sl noripout noripin ;
    interface ed ripin ripout version 1 ;
    traceoptions route ;
} ;

#
# Turn on a bunch of tracing info for the interface to the kernel:
kernel {
    traceoptions remnants request routes info interface ;
} ;

#
# Propagate the route to xxx.xxx.yy out the Ethernet interface via RIP
#

export proto rip interface ed {
    proto direct {
        xxx.xxx.yy mask 255.255.252.0 metric 1; # SLIP connections
    } ;
} ;

#
```

```
# Accept routes from RIP via ed Ethernet interfaces

import proto rip interface ed {
    all ;
} ;
```

The above sample `gated.conf` file broadcasts routing information regarding the SLIP subnet `xxx.xxx.yy` via RIP onto the Ethernet; if you are using a different Ethernet driver than the `ed` driver, you will need to change the references to the `ed` interface appropriately. This sample file also sets up tracing to `/var/tmp/gated.output` for debugging `gated`'s activity; you can certainly turn off the tracing options if `gated` works OK for you. You will need to change the `xxx.xxx.yy`'s into the network address of your own SLIP subnet (be sure to change the net mask in the `proto direct` clause as well).

When you get `gated` built and installed and create a configuration file for it, you will need to run `gated` in place of `routed` on your FreeBSD system; change the `routed/gated` startup parameters in `/etc/netstart` as appropriate for your system. Please see the manual page for `gated` for information on `gated`'s command-line parameters.

## Acknowledgments

Thanks to these people for comments and advice regarding this tutorial:

Wilko Bulte <wilko@yedi.iaf.nl>

Piero Serini

<Piero@Strider.Inet.IT>

# Chapter 16. Advanced Networking

## Gateways and Routes

*Contributed by Coranth Gryphon <gryphon@healer.com>. 6 October 1995.*

For one machine to be able to find another, there must be a mechanism in place to describe how to get from one to the other. This is called Routing. A “route” is a defined pair of addresses: a “destination” and a “gateway”. The pair indicates that if you are trying to get to this *destination*, send along through this *gateway*. There are three types of destinations: individual hosts, subnets, and “default”. The “default route” is used if none of the other routes apply. We will talk a little bit more about default routes later on. There are also three types of gateways: individual hosts, interfaces (also called “links”), and ethernet hardware addresses.

### An example

To illustrate different aspects of routing, we will use the following example which is the output of the command `netstat -r`:

Destination	Gateway	Flags	Refs	Use	Netif	Expire
default	outside-gw	UGSc	37	418	ppp0	
localhost	localhost	UH	0	181	lo0	
test0	0:e0:b5:36:cf:4f	UHLW	5	63288	ed0	77
10.20.30.255	link#1	UHLW	1	2421		
foobar.com	link#1	UC	0	0		
host1	0:e0:a8:37:8:1e	UHLW	3	4601	lo0	
host2	0:e0:a8:37:8:1e	UHLW	0	5	lo0 =>	
host2.foobar.com	link#1	UC	0	0		
224	link#1	UC	0	0		

The first two lines specify the default route (which we will cover in the next section) and the `localhost` route.

The interface (`Netif` column) that it specifies to use for `localhost` is `lo0`, also known as the loopback device. This says to keep all traffic for this destination internal, rather than sending it out over the LAN, since it will only end up back where it started anyway.

The next thing that stands out are the `0:e0:...` addresses. These are ethernet hardware addresses. FreeBSD will automatically identify any hosts (`test0` in the example) on the local ethernet and add a route for that host, directly to it over the ethernet interface, `ed0`. There is also a timeout (`Expire`

column) associated with this type of route, which is used if we fail to hear from the host in a specific amount of time. In this case the route will be automatically deleted. These hosts are identified using a mechanism known as RIP (Routing Information Protocol), which figures out routes to local hosts based upon a shortest path determination.

FreeBSD will also add subnet routes for the local subnet (10.20.30.255 is the broadcast address for the subnet 10.20.30, and `foobar.com` is the domain name associated with that subnet). The designation `link#1` refers to the first ethernet card in the machine. You will notice no additional interface is specified for those.

Both of these groups (local network hosts and local subnets) have their routes automatically configured by a daemon called `routed`. If this is not run, then only routes which are statically defined (ie. entered explicitly) will exist.

The `host1` line refers to our host, which it knows by ethernet address. Since we are the sending host, FreeBSD knows to use the loopback interface (100) rather than sending it out over the ethernet interface.

The two `host2` lines are an example of what happens when we use an `ifconfig` alias (see the section of ethernet for reasons why we would do this). The `=>` symbol after the 100 interface says that not only are we using the loopback (since this is address also refers to the local host), but specifically it is an alias. Such routes only show up on the host that supports the alias; all other hosts on the local network will simply have a `link#1` line for such.

The final line (destination subnet 224) deals with MultiCasting, which will be covered in a another section.

The other column that we should talk about are the `Flags`. Each route has different attributes that are described in the column. Below is a short table of some of these flags and their meanings:

U	Up: The route is active.
H	Host: The route destination is a single host.
G	Gateway: Send anything for this destination on to this remote system, which will figure out from there where to send it.
S	Static: This route was configured manually, not automatically generated by the system.
C	Clone: Generates a new route based upon this route for machines we connect to. This type of route is normally used for local networks.
W	WasCloned: Indicated a route that was auto-configured based upon a local area network (Clone) route.

L Link: Route involves references to ethernet hardware.

## Default routes

When the local system needs to make a connection to remote host, it checks the routing table to determine if a known path exists. If the remote host falls into a subnet that we know how to reach (Cloned routes), then the system checks to see if it can connect along that interface.

If all known paths fail, the system has one last option: the “default” route. This route is a special type of gateway route (usually the only one present in the system), and is always marked with a *c* in the flags field. For hosts on a local area network, this gateway is set to whatever machine has a direct connection to the outside world (whether via PPP link, or your hardware device attached to a dedicated data line).

If you are configuring the default route for a machine which itself is functioning as the gateway to the outside world, then the default route will be the gateway machine at your Internet Service Provider’s (ISP) site.

Let us look at an example of default routes. This is a common configuration:

```
[Local2] <-ether-> [Local1] <-PPP-> [ISP-Serv] <-ether-> [T1-GW]
```

The hosts `Local1` and `Local2` are at your site, with the former being your PPP connection to your ISP’s Terminal Server. Your ISP has a local network at their site, which has, among other things, the server where you connect and a hardware device (T1-GW) attached to the ISP’s Internet feed.

The default routes for each of your machines will be:

host	default gateway	interface
Local2	Local1	ethernet
Local1	T1-GW	PPP

A common question is “Why (or how) would we set the T1-GW to be the default gateway for Local1, rather than the ISP server it is connected to?”.

Remember, since the PPP interface is using an address on the ISP’s local network for your side of the connection, routes for any other machines on the ISP’s local network will be automatically generated. Hence, you will already know how to reach the T1-GW machine, so there is no need for the intermediate step of sending traffic to the ISP server.

As a final note, it is common to use the address `. . . 1` as the gateway address for your local network. So

(using the same example), if your local class-C address space was 10.20.30 and your ISP was using 10.9.9 then the default routes would be:

```
Local2 (10.20.30.2)      -> Local1 (10.20.30.1)
Local1 (10.20.30.1, 10.9.9.30) -> T1-GW (10.9.9.1)
```

## Dual homed hosts

There is one other type of configuration that we should cover, and that is a host that sits on two different networks. Technically, any machine functioning as a gateway (in the example above, using a PPP connection) counts as a dual-homed host. But the term is really only used to refer to a machine that sits on two local-area networks.

In one case, the machine has two ethernet cards, each having an address on the separate subnets. Alternately, the machine may only have one ethernet card, and be using `ifconfig` aliasing. The former is used if two physically separate ethernet networks are in use, the latter if there is one physical network segment, but two logically separate subnets.

Either way, routing tables are set up so that each subnet knows that this machine is the defined gateway (inbound route) to the other subnet. This configuration, with the machine acting as a Bridge between the two subnets, is often used when we need to implement packet filtering or firewall security in either or both directions.

## Routing propagation

We have already talked about how we define our routes to the outside world, but not about how the outside world finds us.

We already know that routing tables can be set up so that all traffic for a particular address space (in our examples, a class-C subnet) can be sent to a particular host on that network, which will forward the packets inbound.

When you get an address space assigned to your site, your service provider will set up their routing tables so that all traffic for your subnet will be sent down your PPP link to your site. But how do sites across the country know to send to your ISP?

There is a system (much like the distributed DNS information) that keeps track of all assigned address-spaces, and defines their point of connection to the Internet Backbone. The “Backbone” are the main trunk lines that carry Internet traffic across the country, and around the world. Each backbone

machine has a copy of a master set of tables, which direct traffic for a particular network to a specific backbone carrier, and from there down the chain of service providers until it reaches your network.

It is the task of your service provider to advertise to the backbone sites that they are the point of connection (and thus the path inward) for your site. This is known as route propagation.

## Troubleshooting

Sometimes, there is a problem with routing propagation, and some sites are unable to connect to you. Perhaps the most useful command for trying to figure out where a routing is breaking down is the `tracert(8)` command. It is equally useful if you cannot seem to make a connection to a remote machine (i.e. `ping(8)` fails).

The `tracert(8)` command is run with the name of the remote host you are trying to connect to. It will show the gateway hosts along the path of the attempt, eventually either reaching the target host, or terminating because of a lack of connection.

For more information, see the manual page for `tracert(8)`.

## NFS

*Contributed by John Lind <john@starfire.MN.ORG>.*

Certain Ethernet adapters for ISA PC systems have limitations which can lead to serious network problems, particularly with NFS. This difficulty is not specific to FreeBSD, but FreeBSD systems are affected by it.

The problem nearly always occurs when (FreeBSD) PC systems are networked with high-performance workstations, such as those made by Silicon Graphics, Inc., and Sun Microsystems, Inc. The NFS mount will work fine, and some operations may succeed, but suddenly the server will seem to become unresponsive to the client, even though requests to and from other systems continue to be processed. This happens to the client system, whether the client is the FreeBSD system or the workstation. On many systems, there is no way to shut down the client gracefully once this problem has manifested itself. The only solution is often to reset the client, because the NFS situation cannot be resolved.

Though the “correct” solution is to get a higher performance and capacity Ethernet adapter for the FreeBSD system, there is a simple workaround that will allow satisfactory operation. If the FreeBSD system is the *server*, include the option `-w=1024` on the mount from the client. If the FreeBSD system is the *client*, then mount the NFS file system with the option `-r=1024`. These options may be specified using the fourth field of the `fstab` entry on the client for automatic mounts, or by using the `-o` parameter of the mount command for manual mounts.

It should be noted that there is a different problem, sometimes mistaken for this one, when the NFS servers and clients are on different networks. If that is the case, make *certain* that your routers are routing the necessary UDP information, or you will not get anywhere, no matter what else you are doing.

In the following examples, `fastws` is the host (interface) name of a high-performance workstation, and `freebox` is the host (interface) name of a FreeBSD system with a lower-performance Ethernet adapter. Also, `/sharedfs` will be the exported NFS filesystem (see `man exports`), and `/project` will be the mount point on the client for the exported file system. In all cases, note that additional options, such as `hard` or `soft` and `bg` may be desirable in your application.

Examples for the FreeBSD system (`freebox`) as the client: in `/etc/fstab` on `freebox`:

```
fastws:/sharedfs /project nfs rw,-r=1024 0 0
```

As a manual mount command on `freebox`:

```
# mount -t nfs -o -r=1024 fastws:/sharedfs /project
```

Examples for the FreeBSD system as the server: in `/etc/fstab` on `fastws`:

```
freebox:/sharedfs /project nfs rw,-w=1024 0 0
```

As a manual mount command on `fastws`:

```
# mount -t nfs -o -w=1024 freebox:/sharedfs /project
```

Nearly any 16-bit Ethernet adapter will allow operation without the above restrictions on the read or write size.

For anyone who cares, here is what happens when the failure occurs, which also explains why it is unrecoverable. NFS typically works with a “block” size of 8k (though it may do fragments of smaller sizes). Since the maximum Ethernet packet is around 1500 bytes, the NFS “block” gets split into multiple Ethernet packets, even though it is still a single unit to the upper-level code, and must be received, assembled, and *acknowledged* as a unit. The high-performance workstations can pump out the packets which comprise the NFS unit one right after the other, just as close together as the standard allows. On the smaller, lower capacity cards, the later packets overrun the earlier packets of the same unit before they can be transferred to the host and the unit as a whole cannot be reconstructed or acknowledged. As a result, the workstation will time out and try again, but it will try again with the entire 8K unit, and the process will be repeated, ad infinitum.

By keeping the unit size below the Ethernet packet size limitation, we ensure that any complete Ethernet packet received can be acknowledged individually, avoiding the deadlock situation.

Overruns may still occur when a high-performance workstation is slamming data out to a PC system, but with the better cards, such overruns are not guaranteed on NFS “units”. When an overrun occurs, the

units affected will be retransmitted, and there will be a fair chance that they will be received, assembled, and acknowledged.

## Diskless Operation

*Contributed by Martin Renters <martin@FreeBSD.ORG>.*

`netboot.com/netboot.rom` allow you to boot your FreeBSD machine over the network and run FreeBSD without having a disk on your client. Under 2.0 it is now possible to have local swap. Swapping over NFS is also still supported.

Supported Ethernet cards include: Western Digital/SMC 8003, 8013, 8216 and compatibles; NE1000/NE2000 and compatibles (requires recompile)

## Setup Instructions

1. Find a machine that will be your server. This machine will require enough disk space to hold the FreeBSD 2.0 binaries and have bootp, tftp and NFS services available. Tested machines:
  - HP9000/8xx running HP-UX 9.04 or later (pre 9.04 doesn't work)
  - Sun/Solaris 2.3. (you may need to get bootp)
2. Set up a bootp server to provide the client with IP, gateway, netmask.

```
diskless:\
    :ht=ether:\
    :ha=0000c01f848a:\
    :sm=255.255.255.0:\
    :hn:\
    :ds=192.1.2.3:\
    :ip=192.1.2.4:\
    :gw=192.1.2.5:\
    :vm=rfc1048:
```

3. Set up a TFTP server (on same machine as bootp server) to provide booting information to client. The name of this file is `cfg.X.X.X.X` (or `/tftpboot/cfg.X.X.X.X`, it will try both) where `X.X.X.X` is the IP address of the client. The contents of this file can be any valid netboot commands. Under 2.0, netboot has the following commands:

<code>help</code>	<code>print help list</code>
<code>ip X.X.X.X</code>	<code>print/set client's IP address</code>

server <i>X.X.X.X</i>	print/set bootp/tftp server address
netmask <i>X.X.X.X</i>	print/set netmask
hostname <i>name</i>	print/set hostname
kernel <i>name</i>	print/set kernel name
rootfs <i>ip:/fs</i>	print/set root filesystem
swapfs <i>ip:/fs</i>	print/set swap filesystem
swapsize <i>size</i>	set diskless swapsize in Kbytes
diskboot	boot from disk
autoboot	continue boot process
trans on off	turn transceiver on off
flags bcdhsv	set boot flags

A typical completely diskless cfg file might contain:

```
rootfs 192.1.2.3:/rootfs/myclient
swapfs 192.1.2.3:/swapfs
swapsize 20000
hostname myclient.mydomain
```

A cfg file for a machine with local swap might contain:

```
rootfs 192.1.2.3:/rootfs/myclient
hostname myclient.mydomain
```

4. Ensure that your NFS server has exported the root (and swap if applicable) filesystems to your client, and that the client has root access to these filesystems. A typical `/etc/exports` file on FreeBSD might look like:

```
/rootfs/myclient -maproot=0:0 myclient.mydomain
/swapfs -maproot=0:0 myclient.mydomain
```

And on HP-UX:

```
/rootfs/myclient -root=myclient.mydomain
/swapfs -root=myclient.mydomain
```

5. If you are swapping over NFS (completely diskless configuration) create a swap file for your client using `dd`. If your `swapfs` command has the arguments `/swapfs` and the size 20000 as in the example above, the swapfile for myclient will be called `/swapfs/swap.X.X.X.X` where `X.X.X.X` is the client's IP addr, eg:

```
# dd if=/dev/zero of=/swapfs/swap.192.1.2.4 bs=1k count=20000
```

Also, the client's swap space might contain sensitive information once swapping starts, so make sure to restrict read and write access to this file to prevent unauthorized access:

```
# chmod 0600 /swapfs/swap.192.1.2.4
```

6. Unpack the root filesystem in the directory the client will use for its root filesystem (`/rootfs/myclient` in the example above).
  - On HP-UX systems: The server should be running HP-UX 9.04 or later for HP9000/800 series machines. Prior versions do not allow the creation of device files over NFS.
  - When extracting `/dev` in `/rootfs/myclient`, beware that some systems (HPUX) will not create device files that FreeBSD is happy with. You may have to go to single user mode on the first bootup (press control-c during the bootup phase), `cd /dev` and do a `sh ./MAKEDEV all` from the client to fix this.
7. Run `netboot.com` on the client or make an EPROM from the `netboot.rom` file

## Using Shared `/` and `/usr` filesystems

At present there isn't an officially sanctioned way of doing this, although I have been using a shared `/usr` filesystem and individual `/` filesystems for each client. If anyone has any suggestions on how to do this cleanly, please let me and/or the FreeBSD core team <freebsd-core@FreeBSD.ORG> know.

## Compiling netboot for specific setups

Netboot can be compiled to support NE1000/2000 cards by changing the configuration in `/sys/i386/boot/netboot/Makefile`. See the comments at the top of this file.

## ISDN

*Last modified by Bill Lloyd <wlloyd@mpd.ca>.*

A good resource for information on ISDN technology and hardware is Dan Kegel's ISDN Page (<http://alumni.caltech.edu/~dank/isdn/>).

A quick simple roadmap to ISDN follows:

- If you live in Europe I suggest you investigate the ISDN card section.
- If you are planning to use ISDN primarily to connect to the Internet with an Internet Provider on a dialup non-dedicated basis, I suggest you look into Terminal Adapters. This will give you the most

flexibility, with the fewest problems, if you change providers.

- If you are connecting two lans together, or connecting to the Internet with a dedicated ISDN connection, I suggest you consider the stand alone router/bridge option.

Cost is a significant factor in determining what solution you will choose. The following options are listed from least expensive to most expensive.

## ISDN Cards

*Contributed by Hellmuth Michaelis <hm@FreeBSD.ORG>.*

This section is really only relevant to ISDN users in countries where the DSS1/Q.931 ISDN standard is supported.

Some growing number of PC ISDN cards are supported under FreeBSD 2.2.x and up by the isdn4bsd driver package. It is still under development but the reports show that it is successfully used all over Europe.

The latest isdn4bsd version is available from <ftp://isdn4bsd@ftp.consol.de/pub/>, the main isdn4bsd ftp site (you have to log in as user `isdn4bsd`, give your mail address as the password and change to the `pub` directory. Anonymous ftp as user `ftp` or `anonymous` will *not* give the desired result).

Isdn4bsd allows you to connect to other ISDN routers using either IP over raw HDLC or by using synchronous PPP. A telephone answering machine application is also available.

Many ISDN PC cards are supported, mostly the ones with a Siemens ISDN chipset (ISAC/HSCX), support for other chipsets (from Motorola, Cologne Chip Designs) is currently under development. For an up-to-date list of supported cards, please have a look at the README (<ftp://isdn4bsd@ftp.consol.de/pub/README>) file.

In case you are interested in adding support for a different ISDN protocol, a currently unsupported ISDN PC card or otherwise enhancing isdn4bsd, please get in touch with [hm@kts.org](mailto:hm@kts.org).

A majordomo maintained mailing list is available. To join the list, send mail to [majordomo@FreeBSD.ORG](mailto:majordomo@FreeBSD.ORG) and specify:

```
subscribe freebsd-isdn
```

in the body of your message.

## ISDN Terminal Adapters

Terminal adapters(TA), are to ISDN what modems are to regular phone lines.

Most TA's use the standard Hayes modem AT command set, and can be used as a drop in replacement for a modem.

A TA will operate basically the same as a modem except connection and throughput speeds will be much faster than your old modem. You will need to configure PPP exactly the same as for a modem setup. Make sure you set your serial speed as high as possible.

The main advantage of using a TA to connect to an Internet Provider is that you can do Dynamic PPP. As IP address space becomes more and more scarce, most providers are not willing to provide you with a static IP anymore. Most standalone routers are not able to accommodate dynamic IP allocation.

TA's completely rely on the PPP daemon that you are running for their features and stability of connection. This allows you to upgrade easily from using a modem to ISDN on a FreeBSD machine, if you already have PPP setup. However, at the same time any problems you experienced with the PPP program and are going to persist.

If you want maximum stability, use the kernel PPP option, not the user-land iijPPP.

The following TA's are known to work with FreeBSD.

- Motorola BitSurfer and Bitsurfer Pro
- Adtran

Most other TA's will probably work as well, TA vendors try to make sure their product can accept most of the standard modem AT command set.

The real problem with external TA's is like modems you need a good serial card in your computer.

You should read the serial ports section in the handbook for a detailed understanding of serial devices, and the differences between asynchronous and synchronous serial ports.

A TA running off a standard PC serial port (asynchronous) limits you to 115.2Kbs, even though you have a 128Kbs connection. To fully utilize the 128Kbs that ISDN is capable of, you must move the TA to a synchronous serial card.

Do not be fooled into buying an internal TA and thinking you have avoided the synchronous/asynchronous issue. Internal TA's simply have a standard PC serial port chip built into them. All this will do, is save you having to buy another serial cable, and find another empty electrical socket.

A synchronous card with a TA is at least as fast as a standalone router, and with a simple 386 FreeBSD box driving it, probably more flexible.

The choice of sync/TA vs standalone router is largely a religious issue. There has been some discussion of this in the mailing lists. I suggest you search the archives (<http://www.freebsd.org/search.html>) for the complete discussion.

## Standalone ISDN Bridges/Routers

ISDN bridges or routers are not at all specific to FreeBSD or any other operating system. For a more complete description of routing and bridging technology, please refer to a Networking reference book.

In the context of this page, I will use router and bridge interchangeably.

As the cost of low end ISDN routers/bridges comes down, it will likely become a more and more popular choice. An ISDN router is a small box that plugs directly into your local Ethernet network(or card), and manages its own connection to the other bridge/router. It has all the software to do PPP and other protocols built in.

A router will allow you much faster throughput than a standard TA, since it will be using a full synchronous ISDN connection.

The main problem with ISDN routers and bridges is that interoperability between manufacturers can still be a problem. If you are planning to connect to an Internet provider, I recommend that you discuss your needs with them.

If you are planning to connect two lan segments together, ie: home lan to the office lan, this is the simplest lowest maintenance solution. Since you are buying the equipment for both sides of the connection you can be assured that the link will work.

For example to connect a home computer or branch office network to a head office network the following setup could be used.

### Example 16-1. Branch office or Home network

Network is 10 Base T Ethernet. Connect router to network cable with AUI/10BT transceiver, if necessary.

```
--Sun workstation
|
--FreeBSD box
|
--Windows 95 (Do not admit to owning it)
|
Standalone router
|
ISDN BRI line
```

If your home/branch office is only one computer you can use a twisted pair crossover cable to connect to the standalone router directly.

**Example 16-2. Head office or other lan**

Network is Twisted Pair Ethernet.

```

-----Novell Server
| H |
|  --Sun
|  |
| U --FreeBSD
|  |
|  --Windows 95
| B |
|___--Standalone router
      |
      ISDN BRI line

```

One large advantage of most routers/bridges is that they allow you to have 2 *separate independent* PPP connections to 2 separate sites at the *same* time. This is not supported on most TA's, except for specific(expensive) models that have two serial ports. Do not confuse this with channel bonding, MPP etc.

This can be very useful feature, for example if you have an dedicated internet ISDN connection at your office and would like to tap into it, but don't want to get another ISDN line at work. A router at the office location can manage a dedicated B channel connection (64Kbs) to the internet, as well as a use the other B channel for a separate data connection. The second B channel can be used for dialin, dialout or dynamically bond(MPP etc.) with the first B channel for more bandwidth.

An Ethernet bridge will also allow you to transmit more than just IP traffic, you can also send IPX/SPX or whatever other protocols you use.

# Chapter 17. Electronic Mail

*Contributed by Bill Lloyd <wlloyd@mpd.ca>.*

Electronic Mail configuration is the subject of many System Administration books. If you plan on doing anything beyond setting up one mailhost for your network, you need industrial strength help.

Some parts of E-Mail configuration are controlled in the Domain Name System (DNS). If you are going to run your own own DNS server check out `/etc/namedb` and `man -k named` for more information.

## Basic Information

These are the major programs involved in an E-Mail exchange. A “mailhost” is a server that is responsible for delivering and receiving all email for your host, and possibly your network.

### User program

This is a program like **elm**, **pine**, **mail**, or something more sophisticated like a WWW browser. This program will simply pass off all e-mail transactions to the local “mailhost”, either by calling `sendmail` or delivering it over TCP.

### Mailhost Server Daemon

Usually this program is `sendmail` or `smail` running in the background. Turn it off or change the command line options in `/etc/rc.conf` (or, prior to FreeBSD 2.2.2, `/etc/sysconfig`). It is best to leave it on, unless you have a specific reason to want it off. Example: You are building a Firewall.

You should be aware that `sendmail` is a potential weak link in a secure site. Some versions of `sendmail` have known security problems.

`sendmail` does two jobs. It looks after delivering and receiving mail.

If `sendmail` needs to deliver mail off your site it will look up in the DNS to determine the actual host that will receive mail for the destination.

If it is acting as a delivery agent `sendmail` will take the message from the local queue and deliver it across the Internet to another `sendmail` on the receivers computer.

## DNS — Name Service

The Domain Name System and its daemon `named`, contain the database mapping hostname to IP address, and hostname to mailhost. The IP address is specified in an A record. The MX record specifies the mailhost that will receive mail for you. If you do not have a MX record mail for your hostname, the mail will be delivered to your host directly.

Unless you are running your own DNS server, you will not be able to change any information in the DNS yourself. If you are using an Internet Provider, speak to them.

## POP Servers

This program gets the mail from your mailbox and gives it to your browser. If you want to run a POP server on your computer, you will need to do 2 things.

1. Get pop software from the Ports collection (`../ports/mail.html`) that can be found in `/usr/ports` or `packages` collection. This handbook section has a complete reference on the Ports system.
2. Modify `/etc/inetd.conf` to load the POP server.

The pop program will have instructions with it. Read them.

## Configuration

### Basic

As your FreeBSD system comes “out of the box”<sup>[TM]</sup>, you should be able to send E-mail to external hosts as long as you have `/etc/resolv.conf` setup or are running a name server. If you want to have mail for your host delivered to your specific host, there are two methods:

- Run a name server (`man -k named`) and have your own domain `smallminingco.com`
- Get mail delivered to the current DNS name for your host. Ie: `dorm6.ahouse.school.edu`

No matter what option you choose, to have mail delivered directly to your host, you must be a full Internet host. You must have a permanent IP address. IE: NO dynamic PPP. If you are behind a firewall, the firewall must be passing on smtp traffic to you. From `/etc/services`:

```
smtp      25/tcp mail      #Simple Mail Transfer
```

If you want to receive mail at your host itself, you must make sure that the DNS MX entry points to your host address, or there is no MX entry for your DNS name.

Try this:

```
# hostname
newbsdbox.freebsd.org
# host newbsdbox.freebsd.org
newbsdbox.freebsd.org has address 204.216.27.xx
```

If that is all that comes out for your machine, mail directory to <root@newbsdbox.freebsd.org> will work no problems.

If instead, you have this:

```
# host newbsdbox.freebsd.org
newbsdbox.FreeBSD.org has address 204.216.27.xx
newbsdbox.FreeBSD.org mail is handled (pri=10) by freefall.FreeBSD.org
```

All mail sent to your host directly will end up on `freefall`, under the same username.

This information is setup in your domain name server. This should be the same host that is listed as your primary nameserver in `/etc/resolv.conf`

The DNS record that carries mail routing information is the Mail eXchange entry. If no MX entry exists, mail will be delivered directly to the host by way of the Address record.

The MX entry for `freefall.freebsd.org` at one time.

```
freefall          MX    30    mail.crl.net
freefall          MX    40    agora.rdrop.com
freefall          HINFO Pentium    FreeBSD
freefall          MX    10    freefall.FreeBSD.org
freefall          MX    20    who.cdrom.com
freefall          A     204.216.27.xx
freefall          CNAME www.FreeBSD.org
```

`freefall` has many MX entries. The lowest MX number gets the mail in the end. The others will queue mail temporarily, if `freefall` is busy or down.

Alternate MX sites should have separate connections to the Internet, to be most useful. An Internet Provider or other friendly site can provide this service.

`dig`, `nslookup`, and `host` are your friends.

## Mail for your Domain (Network).

To setup up a network mailhost, you need to direct the mail from arriving at all the workstations. In other words, you want to hijack all mail for `*.smallminingco.com` and divert it to one machine, your “mailhost”.

The network users on their workstations will most likely pick up their mail over POP or telnet.

A user account with the *same username* should exist on both machines. Please use `adduser` to do this as required. If you set the `shell` to `/nonexistent` the user will not be allowed to login.

The mailhost that you will be using must be designated the Mail eXchange for each workstation. This must be arranged in DNS (ie BIND, named). Please refer to a Networking book for in-depth information.

You basically need to add these lines in your DNS server.

```
pc24.smallminingco.com A xxx.xxx.xxx.xxx ; Workstation ip
                        MX 10 smtp.smallminingco.com ; Your mailhost
```

You cannot do this yourself unless you are running a DNS server. If you do not want to run a DNS server, get somebody else like your Internet Provider to do it.

This will redirect mail for the workstation to the Mail eXchange host. It does not matter what machine the A record points to, the mail will be sent to the MX host.

This feature is used to implement Virtual E-Mail Hosting.

### Example

I have a customer with domain `foo.bar` and I want all mail for `foo.bar` to be sent to my machine `smtp.smalliap.com`. You must make an entry in your DNS server like:

```
foo.bar                MX 10 smtp.smalliap.com ; your mailhost
```

The A record is not needed if you only want E-Mail for the domain. IE: Don't expect ping `foo.bar` to work unless an Address record for `foo.bar` exists as well.

On the mailhost that actually accepts mail for final delivery to a mailbox, `sendmail` must be told what hosts it will be accepting mail for.

Add `pc24.smallminingco.com` to `/etc/sendmail.cw` (if you are using `FEATURE(use_cw_file)`), or add a `Cw myhost.smalliap.com` line to `/etc/sendmail.cf`

If you plan on doing anything serious with `sendmail` you should install the `sendmail` source. The source has plenty of documentation with it. You will find information on getting `sendmail` source from the UUCP information.

## Setting up UUCP.

*Stolen from the FAQ.*

The sendmail configuration that ships with FreeBSD is suited for sites that connect directly to the Internet. Sites that wish to exchange their mail via UUCP must install another `sendmail` configuration file.

Tweaking `/etc/sendmail.cf` manually is considered something for purists. Sendmail version 8 comes with a new approach of generating config files via some `m4` preprocessing, where the actual hand-crafted configuration is on a higher abstraction level. You should use the configuration files under `/usr/src/usr.sbin/sendmail/cf`.

If you did not install your system with full sources, the `sendmail` config stuff has been broken out into a separate source distribution tarball just for you. Assuming you have your CD-ROM mounted, do:

```
# cd /usr/src
# tar -xvzf /cdrom/dists/src/ssmailcf.aa
```

Do not panic, this is only a few hundred kilobytes in size. The file `README` in the `cf` directory can serve as a basic introduction to `m4` configuration.

For UUCP delivery, you are best advised to use the *mailertable* feature. This constitutes a database that `sendmail` can use to base its routing decision upon.

First, you have to create your `.mc` file. The directory `/usr/src/usr.sbin/sendmail/cf/cf` is the home of these files. Look around, there are already a few examples. Assuming you have named your file `foo.mc`, all you need to do in order to convert it into a valid `sendmail.cf` is:

```
# cd /usr/src/usr.sbin/sendmail/cf/cf
# make foo.cf
```

If you don't have a `/usr/obj` hierarchy, then:

```
# cp foo.cf /etc/sendmail.cf
```

Otherwise:

```
# cp /usr/obj/`pwd`/foo.cf /etc/sendmail.cf
```

A typical `.mc` file might look like:

```
include(`../m4/cf.m4')
VERSIONID('Your version number')
OSTYPE(bsd4.4)

FEATURE(nodns)
```

```

FEATURE(nocanonify)
FEATURE(mailertable)

define('UUCP_RELAY', your.uucp.relay)
define('UUCP_MAX_SIZE', 200000)

MAILER(local)
MAILER(smtp)
MAILER(uucp)

Cw    your.alias.host.name
Cw    youruucpnodename.UUCP

```

The `nodns` and `nocanonify` features will prevent any usage of the DNS during mail delivery. The `UUCP_RELAY` clause is needed for bizarre reasons, do not ask. Simply put an Internet hostname there that is able to handle `.UUCP` pseudo-domain addresses; most likely, you will enter the mail relay of your ISP there.

Once you have this, you need this file called `/etc/mailertable`. A typical example of this gender again:

```

#
# makemap hash /etc/mailertable.db < /etc/mailertable
#
horus.interface-business.de    uucp-dom:horus
.interface-business.de        uucp-dom:if-bus
interface-business.de          uucp-dom:if-bus
.heep.sax.de                   smtp8:%1 horus.UUCP
uucp-dom:horus                 if-bus.UUCP
uucp-dom:if-bus .              uucp-dom:sax

```

As you can see, this is part of a real-life file. The first three lines handle special cases where domain-addressed mail should not be sent out to the default route, but instead to some UUCP neighbor in order to “shortcut” the delivery path. The next line handles mail to the local Ethernet domain that can be delivered using SMTP. Finally, the UUCP neighbors are mentioned in the `.UUCP` pseudo-domain notation, to allow for a `uucp-neighbor!recipient` override of the default rules. The last line is always a single dot, matching everything else, with UUCP delivery to a UUCP neighbor that serves as your universal mail gateway to the world. All of the node names behind the `uucp-dom:` keyword must be valid UUCP neighbors, as you can verify using the command `uuname`.

As a reminder that this file needs to be converted into a DBM database file before being usable, the command line to accomplish this is best placed as a comment at the top of the `mailertable`. You always have to execute this command each time you change your `mailertable`.

Final hint: if you are uncertain whether some particular mail routing would work, remember the `-bt` option to `sendmail`. It starts `sendmail` in “address test mode”; simply enter 0, followed by the address you wish to test for the mail routing. The last line tells you the used internal mail agent, the destination host this agent will be called with, and the (possibly translated) address. Leave this mode by typing Control-D.

```
% sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 0 foo@interface-business.de
rewrite: ruleset 0 input: foo @ interface-business . de
...
rewrite: ruleset 0 returns: $# uucp-dom $@ if-bus $: foo < @ interface-
business . de
```

## FAQ

*Migration from FAQ.*

### Why do I have to use the FQDN for hosts on my site?

You will probably find that the host is actually in a different domain; for example, if you are in `foo.bar.edu` and you wish to reach a host called `mumble` in the `bar.edu` domain, you will have to refer to it by the fully-qualified domain name, `mumble.bar.edu`, instead of just `mumble`.

Traditionally, this was allowed by BSD BIND resolvers. However the current version of **BIND** that ships with FreeBSD no longer provides default abbreviations for non-fully qualified domain names other than the domain you are in. So an unqualified host `mumble` must either be found as `mumble.foo.bar.edu`, or it will be searched for in the root domain.

This is different from the previous behavior, where the search continued across `mumble.bar.edu`, and `mumble.edu`. Have a look at RFC 1535 for why this was considered bad practice, or even a security hole.

As a good workaround, you can place the line

```
search foo.bar.edu bar.edu
```

instead of the previous

```
domain foo.bar.edu
```

into your `/etc/resolv.conf`. However, make sure that the search order does not go beyond the “boundary between local and public administration”, as RFC 1535 calls it.

## Sendmail says mail loops back to myself

This is answered in the sendmail FAQ as follows:

```
* I am getting "Local configuration error" messages, such as:
```

```
553 relay.domain.net config error: mail loops back to myself
554 <user@domain.net>... Local configuration error
```

```
How can I solve this problem?
```

```
You have asked mail to the domain (e.g., domain.net) to be
forwarded to a specific host (in this case, relay.domain.net)
by using an MX record, but the relay machine does not recognize
itself as domain.net. Add domain.net to /etc/sendmail.cw
(if you are using FEATURE(use_cw_file)) or add "Cw domain.net"
to /etc/sendmail.cf.
```

The sendmail FAQ is in `/usr/src/usr.sbin/sendmail` and is recommended reading if you want to do any “tweaking” of your mail setup.

## How can I do E-Mail with a dialup PPP host?

You want to connect a FreeBSD box on a lan, to the Internet. The FreeBSD box will be a mail gateway for the lan. The PPP connection is non-dedicated.

There are at least two way to do this.

The other is to use UUCP.

The key is to get a Internet site to provide secondary MX services for your domain. For example:

```
bigco.com.          MX 10 bigco.com.
                   MX      20      smalliap.com.
```

Only one host should be specified as the final recipient ( add `Cw bigco.com` in `/etc/sendmail.cf` on `bigco.com`).

When the senders `sendmail` is trying to deliver the mail it will try to connect to you over the modem link. It will most likely time out because you are not online. `sendmail` will automatically deliver it to

the secondary MX site, ie your Internet provider. The secondary MX site will try every (sendmail\_flags = "-bd -q15m" in /etc/rc.conf ) 15 minutes to connect to your host to deliver the mail to the primary MX site.

You might want to use something like this as a login script.

```
#!/bin/sh
# Put me in /usr/local/bin/pppbigco
( sleep 60 ; /usr/sbin/sendmail -q ) &
/usr/sbin/ppp -direct pppbigco
```

If you are going to create a separate login script for a user you could use `sendmail -qRbigco.com` instead in the script above. This will force all mail in your queue for `bigco.com` to be processed immediately.

A further refinement of the situation is as follows.

Message stolen from the `freebsd-isp` mailing list.

```
> we provide the secondary mx for a customer. The customer connects to
> our services several times a day automatically to get the mails to
> his primary mx (We do not call his site when a mail for his domains
> arrived). Our sendmail sends the mailqueue every 30 minutes. At the
> moment he has to stay 30 minutes online to be sure that all mail is
> gone to the primary mx.
>
> Is there a command that would initiate sendmail to send all the mails
> now? The user has not root-privileges on our machine of course.
```

In the 'privacy flags' section of `sendmail.cf`, there is a definition `Oppoaway,restrictqrun`

Remove `restrictqrun` to allow non-root users to start the queue processing. You might also like to rearrange the MXs. We are the 1st MX for our customers like this, and we have defined:

```
# If we are the best MX for a host, try directly instead of generating
# local config error.
OwTrue
```

That way a remote site will deliver straight to you, without trying the customer connection. You then send to your customer. Only works for "hosts", so you need to get your customer to name their mail machine "customer.com" as well as "hostname.customer.com" in the DNS. Just put an A record in the DNS for "customer.com".

## **IV. Advanced topics**

# Chapter 18. The Cutting Edge: FreeBSD-current and FreeBSD-stable

FreeBSD is under constant development between releases. For people who want to be on the cutting edge, there are several easy mechanisms for keeping your system in sync with the latest developments. Be warned: the cutting edge is not for everyone! This chapter will help you decide if you want to track the development system, or stick with one of the released versions.

## Staying Current with FreeBSD

*Contributed by Jordan K. Hubbard <jkh@FreeBSD.ORG>.*

### What is FreeBSD-current?

FreeBSD-current is, quite literally, nothing more than a daily snapshot of the working sources for FreeBSD. These include work in progress, experimental changes and transitional mechanisms that may or may not be present in the next official release of the software. While many of us compile almost daily from FreeBSD-current sources, there are periods of time when the sources are literally un-compilable. These problems are generally resolved as expeditiously as possible, but whether or not FreeBSD-current sources bring disaster or greatly desired functionality can literally be a matter of which part of any given 24 hour period you grabbed them in!

### Who needs FreeBSD-current?

FreeBSD-current is made generally available for 3 primary interest groups:

1. Members of the FreeBSD group who are actively working on some part of the source tree and for whom keeping “current” is an absolute requirement.
2. Members of the FreeBSD group who are active testers, willing to spend time working through problems in order to ensure that FreeBSD-current remains as sane as possible. These are also people who wish to make topical suggestions on changes and the general direction of FreeBSD.
3. Peripheral members of the FreeBSD (or some other) group who merely wish to keep an eye on things and use the current sources for reference purposes (e.g. for *reading*, not running). These people also make the occasional comment or contribute code.

## What is FreeBSD-current *not*?

1. A fast-track to getting pre-release bits because you heard there is some cool new feature in there and you want to be the first on your block to have it.
2. A quick way of getting bug fixes.
3. In any way “officially supported” by us. We do our best to help people genuinely in one of the 3 “legitimate” FreeBSD-current categories, but we simply *do not have the time* to provide tech support for it. This is not because we are mean and nasty people who do not like helping people out (we would not even be doing FreeBSD if we were), it is literally because we cannot answer 400 messages a day *and* actually work on FreeBSD! I am sure that, if given the choice between having us answer lots of questions or continuing to improve FreeBSD, most of you would vote for us improving it.

## Using FreeBSD-current

1. Join the FreeBSD-current mailing list <freebsd-current@FreeBSD.ORG> and the FreeBSD CVS commit message mailing list <cvс-all@FreeBSD.ORG> . This is not just a good idea, it is *essential*. If you are not on the *FreeBSD-current* mailing list, you will not see the comments that people are making about the current state of the system and thus will probably end up stumbling over a lot of problems that others have already found and solved. Even more importantly, you will miss out on important bulletins which may be critical to your system’s continued health.

The <cvс-all> mailing list will allow you to see the commit log entry for each change as it is made along with any pertinent information on possible side-effects.

To join these lists, send mail to <majordomo@FreeBSD.ORG> and specify:

```
subscribe freebsd-current
subscribe cvs-all
```

in the body of your message. Optionally, you can also say `help` and Majordomo will send you full help on how to subscribe and unsubscribe to the various other mailing lists we support.

2. Grab the sources from `ftp.FreeBSD.ORG`. You can do this in three ways:
  - a. Use the **CTM** facility. Unless you have a good TCP/IP connection at a flat rate, this is the way to do it.
  - b. Use the `cvsup` program with this supfile (`ftp://ftp.FreeBSD.org/pub/FreeBSD/FreeBSD-current/src/share/examples/cvsup/standard-supfile`). This is the second most recommended method, since it allows you to grab the entire collection once and then only what has changed

from then on. Many people run `cvsup` from cron and keep their sources up-to-date automatically. For a fairly easy interface to this, simply type:

```
# pkg_add -
f ftp://ftp.FreeBSD.org/pub/FreeBSD/development/CVSup/cvsupit.tgz
```

- c. Use `ftp`. The source tree for FreeBSD-current is always “exported” on:  
`ftp://ftp.FreeBSD.ORG/pub/FreeBSD/FreeBSD-current`. We also use `wu-ftpd` which allows compressed/tar’d grabbing of whole trees. e.g. you see:

```
usr.bin/lex
```

You can do:

```
ftp> cd usr.bin
ftp> get lex.tar.Z
```

and it will get the whole directory for you as a compressed tar file.

3. Essentially, if you need rapid on-demand access to the source and communications bandwidth is not a consideration, use `cvsup` or `ftp`. Otherwise, use **CTM**.

If you are grabbing the sources to run, and not just look at, then grab *all* of current, not just selected portions. The reason for this is that various parts of the source depend on updates elsewhere, and trying to compile just a subset is almost guaranteed to get you into trouble.

Before compiling current, read the Makefile in `/usr/src` carefully. You should at least run a `make world` the first time through as part of the upgrading process. Reading the FreeBSD-current mailing list `<freebsd-current@FreeBSD.ORG>` will keep you up-to-date on other bootstrapping procedures that sometimes become necessary as we move towards the next release.

4. Be active! If you are running FreeBSD-current, we want to know what you have to say about it, especially if you have suggestions for enhancements or bug fixes. Suggestions with accompanying code are received most enthusiastically!

## Staying Stable with FreeBSD

*Contributed by Jordan K. Hubbard <jkh@FreeBSD.ORG>.*

### What is FreeBSD-stable?

FreeBSD-stable is our development branch for a more low-key and conservative set of changes intended for our next mainstream release. Changes of an experimental or untested nature do not go into this

branch (see FreeBSD-current).

## Who needs FreeBSD-stable?

If you are a commercial user or someone who puts maximum stability of their FreeBSD system before all other concerns, you should consider tracking *stable*. This is especially true if you have installed the most recent release (3.1-RELEASE (<ftp://ftp.FreeBSD.org/pub/FreeBSD/3.1-RELEASE>) at the time of this writing) since the *stable* branch is effectively a bug-fix stream relative to the previous release.

**Warning:** The *stable* tree endeavors, above all, to be fully compilable and stable at all times, but we do occasionally make mistakes (these are still active sources with quickly-transmitted updates, after all). We also do our best to thoroughly test fixes in *current* before bringing them into *stable*, but sometimes our tests fail to catch every case. If something breaks for you in *stable*, please let us know *immediately!* (see next section).

## Using FreeBSD-stable

1. Join the FreeBSD-stable mailing list <[freebsd-stable@FreeBSD.ORG](mailto:freebsd-stable@FreeBSD.ORG)>. This will keep you informed of build-dependencies that may appear in *stable* or any other issues requiring special attention. Developers will also make announcements in this mailing list when they are contemplating some controversial fix or update, giving the users a chance to respond if they have any issues to raise concerning the proposed change.

The <[cvs-all](mailto:cvs-all)> mailing list will allow you to see the commit log entry for each change as it is made along with any pertinent information on possible side-effects.

To join these lists, send mail to <[majordomo@FreeBSD.ORG](mailto:majordomo@FreeBSD.ORG)> and specify:

```
subscribe freebsd-stable
subscribe cvs-all
```

in the body of your message. Optionally, you can also say `help` and Majordomo will send you full help on how to subscribe and unsubscribe to the various other mailing lists we support.

2. If you are installing a new system and want it to be as stable as possible, you can simply grab the latest dated branch snapshot from <ftp://releng3.FreeBSD.org/pub/FreeBSD/> and install it like any other release.

If you are already running a previous release of 2.2 and wish to upgrade via sources then you can easily do so from [ftp.FreeBSD.ORG](ftp://FreeBSD.ORG). This can be done in one of three ways:

- a. Use the **CTM** facility. Unless you have a good TCP/IP connection at a flat rate, this is the way to do it.
- b. Use the `cvsup` program with this supfile (`ftp://ftp.FreeBSD.org/pub/FreeBSD/FreeBSD-current/src/share/examples/cvsup/stable-supfile`). This is the second most recommended method, since it allows you to grab the entire collection once and then only what has changed from then on. Many people run `cvsup` from cron to keep their sources up-to-date automatically. For a fairly easy interface to this, simply type;

```
# pkg_add -
f ftp://ftp.FreeBSD.org/pub/FreeBSD/development/CVSup/cvsupit.tgz
```

- c. Use `ftp`. The source tree for FreeBSD-stable is always “exported” on:  
`ftp://ftp.FreeBSD.ORG/pub/FreeBSD/FreeBSD-stable`

We also use `wu-ftpd` which allows compressed/tar'd grabbing of whole trees. e.g. you see:

```
usr.bin/lex
```

You can do:

```
ftp> cd usr.bin
ftp> get lex.tar.Z
```

and it will get the whole directory for you as a compressed tar file.

3. Essentially, if you need rapid on-demand access to the source and communications bandwidth is not a consideration, use `cvsup` or `ftp`. Otherwise, use **CTM**.
4. Before compiling stable, read the Makefile in `/usr/src` carefully. You should at least run a make world the first time through as part of the upgrading process. Reading the FreeBSD-stable mailing list `<freebsd-stable@FreeBSD.ORG>` will keep you up-to-date on other bootstrapping procedures that sometimes become necessary as we move towards the next release.

## Synchronizing Source Trees over the Internet

*Contributed by Jordan K. Hubbard <jkh@FreeBSD.ORG>.*

There are various ways of using an Internet (or email) connection to stay up-to-date with any given area of the FreeBSD project sources, or all areas, depending on what interests you. The primary services we offer are Anonymous CVS, CVSup, and CTM.

**Anonymous CVS** and **CVSup** use the *pull* model of updating sources. In the case of **CVSup** the user (or a cron script) invokes the `cvsup` program, and it interacts with a `cvsupd` server somewhere to bring your files up to date. The updates you receive are up-to-the-minute and you get them when, and only

when, you want them. You can easily restrict your updates to the specific files or directories that are of interest to you. Updates are generated on the fly by the server, according to what you have and what you want to have. **Anonymous CVS** is quite a bit more simplistic than **CVSup** in that it's just an extension to **CVS** which allows it to pull changes directly from a remote CVS repository. **CVSup** can do this far more efficiently, but **Anonymous CVS** is easier to use.

**CTM**, on the other hand, does not interactively compare the sources you have with those on the master archive or otherwise pull them across.. Instead, a script which identifies changes in files since its previous run is executed several times a day on the master CTM machine, any detected changes being compressed, stamped with a sequence-number and encoded for transmission over email (in printable ASCII only). Once received, these "CTM deltas" can then be handed to the `ctm.rmail(1)` utility which will automatically decode, verify and apply the changes to the user's copy of the sources. This process is far more efficient than **CVSup**, and places less strain on our server resources since it is a *push* rather than a *pull* model.

There are other trade-offs, of course. If you inadvertently wipe out portions of your archive, **CVSup** will detect and rebuild the damaged portions for you. **CTM** won't do this, and if you wipe some portion of your source tree out (and don't have it backed up) then you will have to start from scratch (from the most recent CVS "base delta") and rebuild it all with **CTM** or, with `anoncvs`, simply delete the bad bits and resync.

For more information on **Anonymous CVS**, **CTM**, and **CVSup**, please see one of the following sections:

## Anonymous CVS

*Contributed by Jordan K. Hubbard <jkh@FreeBSD.ORG>*

### Introduction

Anonymous CVS (or, as it is otherwise known, `anoncvs`) is a feature provided by the CVS utilities bundled with FreeBSD for synchronizing with a remote CVS repository. Among other things, it allows users of FreeBSD to perform, with no special privileges, read-only CVS operations against one of the FreeBSD project's official `anoncvs` servers. To use it, one simply sets the `CVSROOT` environment variable to point at the appropriate `anoncvs` server and then uses the `cvs(1)` command to access it like any local repository.

While it can also be said that the **CVSup** and `anoncvs` services both perform essentially the same function, there are various trade-offs which can influence the user's choice of synchronization methods. In a nutshell, **CVSup** is much more efficient in its usage of network resources and is by far the most technically sophisticated of the two, but at a price. To use **CVSup**, a special client must first be installed and configured before any bits can be grabbed, and then only in the fairly large chunks which **CVSup** calls *collections*.

**Anoncv**s, by contrast, can be used to examine anything from an individual file to a specific program (like `ls` or `grep`) by referencing the CVS module name. Of course, **anoncv**s is also only good for read-only operations on the CVS repository, so if it's your intention to support local development in one repository shared with the FreeBSD project bits then **CVSup** is really your only option.

## Using Anonymous CVS

Configuring `cvs(1)` to use an Anonymous CVS repository is a simple matter of setting the `CVSROOT` environment variable to point to one of the FreeBSD project's *anoncv*s servers. At the time of this writing, the following servers are available:

- *USA*: `anoncv@anoncv.FreeBSD.org:/cvs`

Since CVS allows one to “check out” virtually any version of the FreeBSD sources that ever existed (or, in some cases, will exist :), you need to be familiar with the revision (`-r`) flag to `cvs(1)` and what some of the permissible values for it in the FreeBSD Project repository are.

There are two kinds of tags, revision tags and branch tags. A revision tag refers to a specific revision. Its meaning stays the same from day to day. A branch tag, on the other hand, refers to the latest revision on a given line of development, at any given time. Because a branch tag does not refer to a specific revision, it may mean something different tomorrow than it means today.

Here are the branch tags that users might be interested in:

### HEAD

Symbolic name for the main line, or FreeBSD-current. Also the default when no revision is specified.

### RELENG\_3

The line of development for FreeBSD-3.x, also known as FreeBSD-stable. Not valid for the ports collection.

### RELENG\_2\_2

The line of development for FreeBSD-2.2.x, also known as 2.2-stable. Not valid for the ports collection.

### RELENG\_2\_1\_0

The line of development for FreeBSD-2.1.x - this branch is largely obsolete. Not valid for the ports collection.

Here are the revision tags that users might be interested in:

RELENG\_2\_2\_6\_RELEASE

FreeBSD-2.2.6. Not valid for the ports collection.

RELENG\_2\_2\_5\_RELEASE

FreeBSD-2.2.5. Not valid for the ports collection.

RELENG\_2\_2\_2\_RELEASE

FreeBSD-2.2.2. Not valid for the ports collection.

RELENG\_2\_2\_1\_RELEASE

FreeBSD-2.2.1. Not valid for the ports collection.

RELENG\_2\_2\_0\_RELEASE

FreeBSD-2.2.0. Not valid for the ports collection.

RELENG\_2\_1\_7\_RELEASE

FreeBSD-2.1.7. Not valid for the ports collection.

RELENG\_2\_1\_6\_1\_RELEASE

FreeBSD-2.1.6.1. Not valid for the ports collection.

RELENG\_2\_1\_6\_RELEASE

FreeBSD-2.1.6. Not valid for the ports collection.

RELENG\_2\_1\_5\_RELEASE

FreeBSD-2.1.5. Not valid for the ports collection.

RELENG\_2\_1\_0\_RELEASE

FreeBSD-2.1.0. Not valid for the ports collection.

When you specify a branch tag, you normally receive the latest versions of the files on that line of development. If you wish to receive some past version, you can do so by specifying a date with the `-D` `date` flag. See the `cvs(1)` man page for more details.

## Examples

While it really is recommended that you read the manual page for `cvs(1)` thoroughly before doing anything, here are some quick examples which essentially show how to use Anonymous CVS:

### Example 18-1. Checking out something from -current (`ls(1)`) and deleting it again:

```
% setenv CVSROOT anoncvs@anoncvs.freebsd.org:/cvs
% cvs co ls
% cvs release -d ls
```

### Example 18-2. Checking out the version of `ls(1)` in the 2.2-stable branch:

```
% setenv CVSROOT anoncvs@anoncvs.freebsd.org:/cvs
% cvs co -rRELENG_2_2 ls
% cvs release -d ls
```

### Example 18-3. Creating a list of changes (as `unidiffs`) to `ls(1)` between FreeBSD 2.2.2 and FreeBSD 2.2.6:

```
% setenv CVSROOT anoncvs@anoncvs.freebsd.org:/cvs
% cvs rdiff -u -rRELENG_2_2_2_RELEASE -rRELENG_2_2_6_RELEASE ls
```

### Example 18-4. Finding out what other module names can be used:

```
% setenv CVSROOT anoncvs@anoncvs.freebsd.org:/cvs
% cvs co modules
% more modules/modules
% cvs release -d modules
```

## Other Resources

The following additional resources may be helpful in learning CVS:

- CVS Tutorial (<http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/cvs/>) from Cal Poly.
- Cyclic Software (<http://www.cyclic.com>), commercial maintainers of CVS.
- CVSWeb (<http://www.FreeBSD.org/cgi/cvsweb.cgi>) is the FreeBSD Project web interface for CVS.

## CTM

*Contributed by Poul-Henning Kamp <phk@FreeBSD.ORG>. Updated 19-October-1997.*

**CTM** is a method for keeping a remote directory tree in sync with a central one. It has been developed for usage with FreeBSD's source trees, though other people may find it useful for other purposes as time goes by. Little, if any, documentation currently exists at this time on the process of creating deltas, so talk to Poul-Henning Kamp <phk@FreeBSD.ORG> for more information should you wish to use **CTM** for other things.

### Why should I use CTM?

**CTM** will give you a local copy of the FreeBSD source trees. There are a number of “flavors” of the tree available. Whether you wish to track the entire cvs tree or just one of the branches, **CTM** can provide you the information. If you are an active developer on FreeBSD, but have lousy or non-existent TCP/IP connectivity, or simply wish to have the changes automatically sent to you, **CTM** was made for you. You will need to obtain up to three deltas per day for the most active branches. However, you should consider having them sent by automatic email. The sizes of the updates are always kept as small as possible. This is typically less than 5K, with an occasional (one in ten) being 10-50K and every now and then a biggie of 100K+ or more coming around.

You will also need to make yourself aware of the various caveats related to working directly from the development sources rather than a pre-packaged release. This is particularly true if you choose the “current” sources. It is recommended that you read *Staying current with FreeBSD*.

### What do I need to use CTM?

You will need two things: The **CTM** program and the initial deltas to feed it (to get up to “current” levels).

The **CTM** program has been part of FreeBSD ever since version 2.0 was released, and lives in `/usr/src/usr.sbin/CTM` if you have a copy of the source online.

If you are running a pre-2.0 version of FreeBSD, you can fetch the current **CTM** sources directly from:

`ftp://ftp.FreeBSD.ORG/pub/FreeBSD/FreeBSD-current/src/usr.sbin/ctm`

The “deltas” you feed **CTM** can be had two ways, FTP or e-mail. If you have general FTP access to the Internet then the following FTP sites support access to **CTM**:

`ftp://ftp.FreeBSD.ORG/pub/FreeBSD/CTM`

or see section mirrors.

FTP the relevant directory and fetch the `README` file, starting from there.

If you may wish to get your deltas via email:

Send email to <majordomo@FreeBSD.ORG> to subscribe to one of the **CTM** distribution lists.

“ctm-cvs-cur” supports the entire cvs tree. “ctm-src-cur” supports the head of the development branch.

“ctm-src-2\_2” supports the 2.2 release branch, etc. (If you do not know how to subscribe yourself using majordomo, send a message first containing the word `help` — it will send you back usage instructions.)

When you begin receiving your **CTM** updates in the mail, you may use the `ctm_rmail` program to unpack and apply them. You can actually use the `ctm_rmail` program directly from a entry in `/etc/aliases` if you want to have the process run in a fully automated fashion. Check the `ctm_rmail` man page for more details.

**Note:** No matter what method you use to get the **CTM** deltas, you should subscribe to the <ctm-announce@FreeBSD.ORG> mailing list. In the future, this will be the only place where announcements concerning the operations of the **CTM** system will be posted. Send an email to <majordomo@FreeBSD.ORG> with a single line of `subscribe ctm-announce` to get added to the list.

## Starting off with CTM for the first time

Before you can start using **CTM** deltas, you will need to get a to a starting point for the deltas produced subsequently to it.

First you should determine what you already have. Everyone can start from an “empty” directory. You must use an initial “Empty” delta to start off your **CTM** supported tree. At some point it is intended that one of these “started” deltas be distributed on the CD for your convenience. This does not currently happen however.

However, since the trees are many tens of megabytes, you should prefer to start from something already at hand. If you have a RELEASE CD, you can copy or extract an initial source from it. This will save a significant transfer of data.

You can recognize these “starter” deltas by the `x` appended to the number (`src-cur.3210XEmpty.gz` for instance). The designation following the `x` corresponds to the origin of your initial “seed”. `Empty` is an empty directory. As a rule a base transition from `Empty` is produced every 100 deltas. By the way, they are large! 25 to 30 Megabytes of `gzip`'ed data is common for the `XEmpty` deltas.

Once you've picked a base delta to start from, you will also need all deltas with higher numbers following it.

## Using CTM in your daily life

To apply the deltas, simply say:

```
# cd /where/ever/you/want/the/stuff
# ctm -v -v /where/you/store/your/deltas/src-xxx.*
```

**CTM** understands deltas which have been put through `gzip`, so you do not need to gunzip them first, this saves disk space.

Unless it feels very secure about the entire process, **CTM** will not touch your tree. To verify a delta you can also use the `-c` flag and **CTM** will not actually touch your tree; it will merely verify the integrity of the delta and see if it would apply cleanly to your current tree.

There are other options to **CTM** as well, see the manual pages or look in the sources for more information.

I would also be very happy if somebody could help with the “user interface” portions, as I have realized that I cannot make up my mind on what options should do what, how and when...

That’s really all there is to it. Every time you get a new delta, just run it through **CTM** to keep your sources up to date.

Do not remove the deltas if they are hard to download again. You just might want to keep them around in case something bad happens. Even if you only have floppy disks, consider using `fdwrite` to make a copy.

## Keeping your local changes

As a developer one would like to experiment with and change files in the source tree. **CTM** supports local modifications in a limited way: before checking for the presence of a file `foo`, it first looks for `foo.ctm`. If this file exists, **CTM** will operate on it instead of `foo`.

This behaviour gives us a simple way to maintain local changes: simply copy the files you plan to modify to the corresponding file names with a `.ctm` suffix. Then you can freely hack the code, while **CTM** keeps the `.ctm` file up-to-date.

## Other interesting CTM options

### Finding out exactly what would be touched by an update

You can determine the list of changes that **CTM** will make on your source repository using the `-l` option to **CTM**.

This is useful if you would like to keep logs of the changes, pre- or post- process the modified files in any manner, or just are feeling a tad paranoid :-).

### Making backups before updating

Sometimes you may want to backup all the files that would be changed by a **CTM** update.

Specifying the `-B backup-file` option causes **CTM** to backup all files that would be touched by a given **CTM** delta to `backup-file`.

### Restricting the files touched by an update

Sometimes you would be interested in restricting the scope of a given **CTM** update, or may be interested in extracting just a few files from a sequence of deltas.

You can control the list of files that **CTM** would operate on by specifying filtering regular expressions using the `-e` and `-x` options.

For example, to extract an up-to-date copy of `lib/libc/Makefile` from your collection of saved **CTM** deltas, run the commands:

```
# cd /where/ever/you/want/to/extract/it/  
# ctm -e '^lib/libc/Makefile' ~ctm/src-xxx.*
```

For every file specified in a **CTM** delta, the `-e` and `-x` options are applied in the order given on the command line. The file is processed by **CTM** only if it is marked as eligible after all the `-e` and `-x` options are applied to it.

### Future plans for CTM

Tons of them:

- Use some kind of authentication into the **CTM** system, so as to allow detection of spoofed **CTM** updates.
- Clean up the options to **CTM**, they became confusing and counter intuitive.

The bad news is that I am very busy, so any help in doing this will be most welcome. And do not forget to tell me what you want also...

### Miscellaneous stuff

All the “DES infected” (e.g. export controlled) source is not included. You will get the “international” version only. If sufficient interest appears, we will set up a `sec-cur` sequence too. There is a sequence

of deltas for the `ports` collection too, but interest has not been all that high yet. Tell me if you want an email list for that too and we will consider setting it up.

## Thanks!

Bruce Evans <bde@FreeBSD.ORG>

for his pointed pen and invaluable comments.

Søren Schmidt <sos@FreeBSD.ORG>

for patience.

Stephen McKay

wrote `ctm_[rs]mail`, much appreciated.

Jordan K. Hubbard <jkh@FreeBSD.ORG>

for being so stubborn that I had to make it better.

All the users

I hope you like it...

## CVSup

*Contributed by John Polstra <jdp@FreeBSD.ORG>.*

### Introduction

**CVSup** is a software package for distributing and updating source trees from a master CVS repository on a remote server host. The FreeBSD sources are maintained in a CVS repository on a central development machine in California. With **CVSup**, FreeBSD users can easily keep their own source trees up to date.

**CVSup** uses the so-called *pull* model of updating. Under the pull model, each client asks the server for updates, if and when they are wanted. The server waits passively for update requests from its clients. Thus all updates are instigated by the client. The server never sends unsolicited updates. Users must

either run the **CVSup** client manually to get an update, or they must set up a `cron` job to run it automatically on a regular basis.

The term **CVSup**, capitalized just so, refers to the entire software package. Its main components are the client `cvsup` which runs on each user's machine, and the server `cvsupd` which runs at each of the FreeBSD mirror sites.

As you read the FreeBSD documentation and mailing lists, you may see references to **sup**. **Sup** was the predecessor of **CVSup**, and it served a similar purpose. **CVSup** is in used in much the same way as `sup` and, in fact, uses configuration files which are backward-compatible with `sup`'s. **Sup** is no longer used in the FreeBSD project, because **CVSup** is both faster and more flexible.

## Installation

The easiest way to install **CVSup** if you are running FreeBSD 2.2 or later is to use either the port (<ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/ports-current/net/cvsup.tar>) from the FreeBSD ports collection or the corresponding binary package (<ftp://ftp.FreeBSD.org/pub/FreeBSD/packages-current/net/cvsup-16.0.tgz>), depending on whether you prefer to roll your own or not.

If you are running FreeBSD-2.1.6 or 2.1.7, you unfortunately cannot use the binary package versions due to the fact that they require a version of the C library that does not yet exist in FreeBSD-2.1.{6,7}. You can easily use the port (<ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/ports-current/net/cvsup.tar>), however, just as with FreeBSD 2.2. Simply unpack the tar file, `cd` to the `cvsup` subdirectory and type `make install`.

Because **CVSup** is written in Modula-3 (<http://www.research.digital.com/SRC/modula-3/html/home.html>), both the package and the port require that the Modula-3 runtime libraries be installed. These are available as the `lang/modula-3-lib` (<ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/ports-current/lang/modula-3-lib.tar>) port and the `lang/modula-3-lib-3.6` (<ftp://ftp.FreeBSD.org/pub/FreeBSD/packages-current/lang/modula-3-lib-3.6.tgz>) package. If you follow the same directions as for `cvsup`, these libraries will be compiled and/or installed automatically when you install the **CVSup** port or package.

The Modula-3 libraries are rather large, and fetching and compiling them is not an instantaneous process. For that reason, a third option is provided. You can get *statically linked* FreeBSD executables for **CVSup** from either the USA distribution site:

- <ftp://ftp.FreeBSD.org/pub/FreeBSD/development/CVSup/cvsup-bin-16.0.tar.gz> (client including GUI).
- <ftp://ftp.FreeBSD.org/pub/FreeBSD/development/CVSup/cvsup.nogui-bin-16.0.tar.gz> (client without GUI).

- <ftp://ftp.FreeBSD.org/pub/FreeBSD/development/CVSup/cvsupd-bin-16.0.tar.gz> (server).  
as well as from the many FreeBSD FTP mirror sites around the world.
- <ftp://ftp.cs.tu-berlin.de/pub/FreeBSD/development/CVSup/cvsup-bin-15.3.tar.gz> (client including GUI).
- <ftp://ftp.cs.tu-berlin.de/pub/FreeBSD/development/CVSup/cvsup.nogui-bin-15.3.tar.gz> (client without GUI).
- <ftp://ftp.cs.tu-berlin.de/pub/FreeBSD/development/CVSup/cvsupd-bin-15.3.tar.gz> (server).

Most users will need only the client. These executables are entirely self-contained, and they will run on any version of FreeBSD from FreeBSD-2.1.0 to FreeBSD-current.

In summary, your options for installing CVSup are:

- FreeBSD-2.2 or later: static binary, port, or package
- FreeBSD-2.1.6, 2.1.7: static binary or port
- FreeBSD-2.1.5 or earlier: static binary

## Configuration

CVSup's operation is controlled by a configuration file called the `supfile`. Beginning with FreeBSD-2.2, there are some sample `supfiles` in the directory `/usr/share/examples/cvsup` (file:`/usr/share/examples/cvsup`). These examples are also available from <ftp://ftp.FreeBSD.org/pub/FreeBSD/FreeBSD-current/src/share/examples/cvsup/> if you are on a pre-2.2 system.

The information in a `supfile` answers the following questions for `cvsup`:

- Which files do you want to receive?
- Which versions of them do you want?
- Where do you want to get them from?
- Where do you want to put them on your own machine?
- Where do you want to put your status files?

In the following sections, we will construct a typical `supfile` by answering each of these questions in turn. First, we describe the overall structure of a `supfile`.

A `supfile` is a text file. Comments begin with `#` and extend to the end of the line. Lines that are blank and lines that contain only comments are ignored.

Each remaining line describes a set of files that the user wishes to receive. The line begins with the name of a “collection”, a logical grouping of files defined by the server. The name of the collection tells the server which files you want. After the collection name come zero or more fields, separated by white space. These fields answer the questions listed above. There are two types of fields: flag fields and value fields. A flag field consists of a keyword standing alone, e.g., `delete` or `compress`. A value field also begins with a keyword, but the keyword is followed without intervening white space by `=` and a second word. For example, `release=cvs` is a value field.

A `supfile` typically specifies more than one collection to receive. One way to structure a `supfile` is to specify all of the relevant fields explicitly for each collection. However, that tends to make the `supfile` lines quite long, and it is inconvenient because most fields are the same for all of the collections in a `supfile`. **CVSup** provides a defaulting mechanism to avoid these problems. Lines beginning with the special pseudo-collection name `*default` can be used to set flags and values which will be used as defaults for the subsequent collections in the `supfile`. A default value can be overridden for an individual collection, by specifying a different value with the collection itself. Defaults can also be changed or augmented in mid-`supfile` by additional `*default` lines.

With this background, we will now proceed to construct a `supfile` for receiving and updating the main source tree of FreeBSD-current.

- Which files do you want to receive?

The files available via **CVSup** are organized into named groups called “collections”. The collections that are available are described here. In this example, we wish to receive the entire main source tree for the FreeBSD system. There is a single large collection `src-all` which will give us all of that, except the export-controlled cryptography support. Let us assume for this example that we are in the USA or Canada. Then we can get the cryptography code with one additional collection, `cvs-crypto`. As a first step toward constructing our `supfile`, we simply list these collections, one per line:

```
src-all
cvs-crypto
```

- Which version(s) of them do you want?

With **CVSup**, you can receive virtually any version of the sources that ever existed. That is possible because the `cvsupd` server works directly from the CVS repository, which contains all of the versions. You specify which one of them you want using the `tag=` and `date=` value fields.

**Warning:** Be very careful to specify any `tag=` fields correctly. Some tags are valid only for certain collections of files. If you specify an incorrect or misspelled tag, CVSup will delete files which you probably do not want deleted. In particular, use *only* `tag=.` for the `ports-*` collections.

The `tag=` field names a symbolic tag in the repository. There are two kinds of tags, revision tags and branch tags. A revision tag refers to a specific revision. Its meaning stays the same from day to day. A

branch tag, on the other hand, refers to the latest revision on a given line of development, at any given time. Because a branch tag does not refer to a specific revision, it may mean something different tomorrow than it means today.

Here are the branch tags that users might be interested in:

tag=.

The main line of development, also known as FreeBSD-current.

**Note:** The . is not punctuation; it is the name of the tag. Valid for all collections.

RELENG\_3

The line of development for FreeBSD-3.x, also known as FreeBSD-stable. Not valid for the ports collection.

RELENG\_2\_2

The line of development for FreeBSD-2.2.x, also known as 2.2-stable. Not valid for the ports collection.

tag=RELENG\_2\_1\_0

The line of development for FreeBSD-2.1.x - this branch is largely obsolete. Not valid for the ports-\* collections.

Here are the revision tags that users might be interested in:

tag=RELENG\_3\_0\_0\_RELEASE

FreeBSD-3.0. Not valid for the ports-\* collections.

tag=RELENG\_2\_2\_8\_RELEASE

FreeBSD-2.2.8. Not valid for the ports-\* collections.

tag=RELENG\_2\_2\_7\_RELEASE

FreeBSD-2.2.7. Not valid for the ports-\* collections.

tag=RELENG\_2\_2\_6\_RELEASE

FreeBSD-2.2.6. Not valid for the ports-\* collections.

tag=RELENG\_2\_2\_5\_RELEASE

FreeBSD-2.2.5. Not valid for the ports-\* collections.

tag=RELENG\_2\_2\_2\_RELEASE

FreeBSD-2.2.2. Not valid for the ports-\* collections.

tag=RELENG\_2\_2\_1\_RELEASE

FreeBSD-2.2.1. Not valid for the ports-\* collections.

tag=RELENG\_2\_2\_0\_RELEASE

FreeBSD-2.2.0. Not valid for the ports-\* collections.

tag=RELENG\_2\_1\_7\_RELEASE

FreeBSD-2.1.7. Not valid for the ports-\* collections.

tag=RELENG\_2\_1\_6\_1\_RELEASE

FreeBSD-2.1.6.1. Not valid for the ports-\* collections.

tag=RELENG\_2\_1\_6\_RELEASE

FreeBSD-2.1.6. Not valid for the ports-\* collections.

tag=RELENG\_2\_1\_5\_RELEASE

FreeBSD-2.1.5. Not valid for the ports-\* collections.

tag=RELENG\_2\_1\_0\_RELEASE

FreeBSD-2.1.0. Not valid for the ports-\* collections.

**Warning:** Be very careful to type the tag name exactly as shown. **CVSup** cannot distinguish between valid and invalid tags. If you misspell the tag, **CVSup** will behave as though you had specified a valid tag which happens to refer to no files at all. It will delete your existing sources in that case.

When you specify a branch tag, you normally receive the latest versions of the files on that line of development. If you wish to receive some past version, you can do so by specifying a date with the `date=` value field. The `cvsup(1)` manual page explains how to do that.

For our example, we wish to receive FreeBSD-current. We add this line at the beginning of our `supfile`:

```
*default tag=.
```

There is an important special case that comes into play if you specify neither a `tag=` field nor a `date=` field. In that case, you receive the actual RCS files directly from the server's CVS repository, rather than receiving a particular version. Developers generally prefer this mode of operation. By maintaining a copy of the repository itself on their systems, they gain the ability to browse the revision histories and examine past versions of files. This gain is achieved at a large cost in terms of disk space, however.

- Where do you want to get them from?

We use the `host=` field to tell `cvsup` where to obtain its updates. Any of the CVSup mirror sites will do, though you should try to select one that is close to you in cyberspace. In this example we will use a fictional FreeBSD distribution site, `cvsup666.FreeBSD.org`:

```
*default host=cvsup666.FreeBSD.org
```

You will need to change the `host` to one that actually exists before running CVSup. On any particular run of `cvsup`, you can override the `host` setting on the command line, with `-h hostname`.

- Where do you want to put them on your own machine?

The `prefix=` field tells `cvsup` where to put the files it receives. In this example, we will put the source files directly into our main source tree, `/usr/src`. The `src` directory is already implicit in the collections we have chosen to receive, so this is the correct specification:

```
*default prefix=/usr
```

- Where should `cvsup` maintain its status files?

The `cvsup` client maintains certain status files in what is called the “base” directory. These files help **CVSup** to work more efficiently, by keeping track of which updates you have already received. We will use the standard base directory, `/usr/local/etc/cvsup`:

```
*default base=/usr/local/etc/cvsup
```

This setting is used by default if it is not specified in the `supfile`, so we actually do not need the above line.

If your base directory does not already exist, now would be a good time to create it. The `cvsup` client will refuse to run if the base directory does not exist.

- Miscellaneous `supfile` settings:

There is one more line of boiler plate that normally needs to be present in the `supfile`:

```
*default release=cvs delete use-rel-suffix compress
```

`release=cvs` indicates that the server should get its information out of the main FreeBSD CVS repository. This is virtually always the case, but there are other possibilities which are beyond the scope of this discussion.

`delete` gives **CVSup** permission to delete files. You should always specify this, so that **CVSup** can keep your source tree fully up to date. **CVSup** is careful to delete only those files for which it is responsible. Any extra files you happen to have will be left strictly alone.

`use-rel-suffix` is ... arcane. If you really want to know about it, see the `cvsup(1)` manual page. Otherwise, just specify it and do not worry about it.

`compress` enables the use of `gzip`-style compression on the communication channel. If your network link is T1 speed or faster, you probably should not use compression. Otherwise, it helps substantially.

- Putting it all together:

Here is the entire `supfile` for our example:

```
*default tag=.
*default host=cvsup666.FreeBSD.org
*default prefix=/usr
*default base=/usr/local/etc/cvsup
*default release=cvs delete use-rel-suffix compress

src-all
cvs-crypto
```

## Running CVSup

You are now ready to try an update. The command line for doing this is quite simple:

```
# cvsup supfile
```

where *supfile* is of course the name of the `supfile` you have just created. Assuming you are running under X11, `cvsup` will display a GUI window with some buttons to do the usual things. Press the “go” button, and watch it run.

Since you are updating your actual `/usr/src` tree in this example, you will need to run the program as `root` so that `cvsup` has the permissions it needs to update your files. Having just created your configuration file, and having never used this program before, that might understandably make you

nervous. There is an easy way to do a trial run without touching your precious files. Just create an empty directory somewhere convenient, and name it as an extra argument on the command line:

```
# mkdir /var/tmp/dest
# cvsup supfile /var/tmp/dest
```

The directory you specify will be used as the destination directory for all file updates. **CVSup** will examine your usual files in `/usr/src`, but it will not modify or delete any of them. Any file updates will instead land in `/var/tmp/dest/usr/src`. **CVSup** will also leave its base directory status files untouched when run this way. The new versions of those files will be written into the specified directory. As long as you have read access to `/usr/src`, you do not even need to be root to perform this kind of trial run.

If you are not running X11 or if you just do not like GUIs, you should add a couple of options to the command line when you run `cvsup`:

```
# cvsup -g -L 2 supfile
```

The `-g` tells `cvsup` not to use its GUI. This is automatic if you are not running X11, but otherwise you have to specify it.

The `-L 2` tells `cvsup` to print out the details of all the file updates it is doing. There are three levels of verbosity, from `-L 0` to `-L 2`. The default is 0, which means total silence except for error messages.

There are plenty of other options available. For a brief list of them, type `cvsup -h`. For more detailed descriptions, see the manual page.

Once you are satisfied with the way updates are working, you can arrange for regular runs of `cvsup` using `cron(8)`. Obviously, you should not let `cvsup` use its GUI when running it from `cron`.

## CVSup File Collections

The file collections available via **CVSup** are organized hierarchically. There are a few large collections, and they are divided into smaller sub-collections. Receiving a large collection is equivalent to receiving each of its sub-collections. The hierarchical relationships among collections are reflected by the use of indentation in the list below.

The most commonly used collections are `src-all`, `cvs-crypto`, and `ports-all`. The other collections are used only by small groups of people for specialized purposes, and some mirror sites may not carry all of them.

```
cvs-all release=cvs
```

The main FreeBSD CVS repository, excluding the export-restricted cryptography code.

`distrib release=cvs`

Files related to the distribution and mirroring of FreeBSD.

`doc-all release=cvs`

Sources for the FreeBSD handbook and other documentation.

`ports-all release=cvs`

The FreeBSD ports collection.

`ports-archivers release=cvs`

Archiving tools.

`ports-astro release=cvs`

Astronomical ports.

`ports-audio release=cvs`

Sound support.

`ports-base release=cvs`

Miscellaneous files at the top of `/usr/ports`.

`ports-benchmarks release=cvs`

Benchmarks.

`ports-biology release=cvs`

Biology.

`ports-cad release=cvs`

Computer aided design tools.

`ports-chinese release=cvs`

Chinese language support.

ports-comms release=cvs

Communication software.

ports-converters release=cvs

character code converters.

ports-databases release=cvs

Databases.

ports-deskutils release=cvs

Things that used to be on the desktop before computers were invented.

ports-devel release=cvs

Development utilities.

ports-editors release=cvs

Editors.

ports-emulators release=cvs

Emulators for other operating systems.

ports-games release=cvs

Games.

ports-german release=cvs

German language support.

ports-graphics release=cvs

Graphics utilities.

ports-japanese release=cvs

Japanese language support.

ports-korean release=cvs

Korean language support.

ports-lang release=cvs

Programming languages.

ports-mail release=cvs

Mail software.

ports-math release=cvs

Numerical computation software.

ports-mbone release=cvs

MBone applications.

ports-misc release=cvs

Miscellaneous utilities.

ports-net release=cvs

Networking software.

ports-news release=cvs

USENET news software.

ports-plan9 release=cvs

Various programs from Plan9.

ports-print release=cvs

Printing software.

ports-russian release=cvs

Russian language support.

ports-security release=cvs

Security utilities.

ports-shells release=cvs

Command line shells.

ports-sysutils release=cvs

System utilities.

ports-textproc release=cvs

text processing utilities (does not include desktop publishing).

ports-vietnamese release=cvs

Vietnamese language support.

ports-www release=cvs

Software related to the World Wide Web.

ports-x11 release=cvs

Ports to support the X window system.

ports-x11-clocks release=cvs

X11 clocks.

ports-x11-fm release=cvs

X11 file managers.

ports-x11-fonts release=cvs

X11 fonts and font utilities.

ports-x11-toolkits release=cvs

X11 toolkits.

ports-x11-wm

X11 window managers.

src-all release=cvs

The main FreeBSD sources, excluding the export-restricted cryptography code.

src-base release=cvs

Miscellaneous files at the top of `/usr/src`.

src-bin release=cvs

User utilities that may be needed in single-user mode (`/usr/src/bin`).

src-contrib release=cvs

Utilities and libraries from outside the FreeBSD project, used relatively unmodified (`/usr/src/contrib`).

src-etc release=cvs

System configuration files (`/usr/src/etc`).

src-games release=cvs

Games (`/usr/src/games`).

src-gnu release=cvs

Utilities covered by the GNU Public License (`/usr/src/gnu`).

src-include release=cvs

Header files (`/usr/src/include`).

src-kerberosIV release=cvs

KerberosIV security package (`/usr/src/kerberosIV`).

src-lib release=cvs

Libraries (`/usr/src/lib`).

`src-libexec release=cvs`

System programs normally executed by other programs (`/usr/src/libexec`).

`src-release release=cvs`

Files required to produce a FreeBSD release (`/usr/src/release`).

`src-sbin release=cvs`

System utilities for single-user mode (`/usr/src/sbin`).

`src-share release=cvs`

Files that can be shared across multiple systems (`/usr/src/share`).

`src-sys release=cvs`

The kernel (`/usr/src/sys`).

`src-tools release=cvs`

Various tools for the maintenance of FreeBSD (`/usr/src/tools`).

`src-usrbin release=cvs`

User utilities (`/usr/src/usr.bin`).

`src-usrsbin release=cvs`

System utilities (`/usr/src/usr.sbin`).

`www release=cvs`

The sources for the World Wide Web data.

`cvs-crypto release=cvs`

The export-restricted cryptography code.

`src-crypto release=cvs`

Export-restricted utilities and libraries from outside the FreeBSD project, used relatively unmodified (`/usr/src/crypto`).

```
src-eBones release=cvsvs
    Kerberos and DES (/usr/src/eBones).
```

```
src-secure release=cvsvs
    DES (/usr/src/secure).
```

```
distrib release=self
    The CVSup server's own configuration files. Used by CVSup mirror sites.
```

```
gnats release=current
    The GNATS bug-tracking database.
```

```
mail-archive release=current
    FreeBSD mailing list archive.
```

```
www release=current
    The installed World Wide Web data. Used by WWW mirror sites.
```

### For more information

For the CVSup FAQ and other information about CVSup, see The CVSup Home Page (<http://www.polstra.com/projects/freeware/CVSup/>).

Most FreeBSD-related discussion of **CVSup** takes place on the FreeBSD technical discussions mailing list <[freebsd-hackers@FreeBSD.ORG](mailto:freebsd-hackers@FreeBSD.ORG)>. New versions of the software are announced there, as well as on the FreeBSD announcements mailing list <[freebsd-announce@FreeBSD.ORG](mailto:freebsd-announce@FreeBSD.ORG)>.

Questions and bug reports should be addressed to the author of the program at <[cvsup-bugs@polstra.com](mailto:cvsup-bugs@polstra.com)>.

## Using `make world` to rebuild your system

*Contributed by Nik Clayton <[nik@FreeBSD.ORG](mailto:nik@FreeBSD.ORG)>.*

Once you have synchronised your local source tree against a particular version of FreeBSD (`stable`, `current` and so on) you must then use the source tree to rebuild the system.

Currently, the best source of information on how to do that is a tutorial available from <http://www.nothing-going-on.demon.co.uk/FreeBSD/make-world/make-world.html>.

A successor to this tutorial will be integrated into the handbook.

# Chapter 19. Contributing to FreeBSD

*Contributed by Jordan K. Hubbard <jkh@FreeBSD.ORG>.*

So you want to contribute something to FreeBSD? That is great! We can always use the help, and FreeBSD is one of those systems that *relies* on the contributions of its user base in order to survive. Your contributions are not only appreciated, they are vital to FreeBSD's continued growth!

Contrary to what some people might also have you believe, you do not need to be a hot-shot programmer or a close personal friend of the FreeBSD core team in order to have your contributions accepted. The FreeBSD Project's development is done by a large and growing number of international contributors whose ages and areas of technical expertise vary greatly, and there is always more work to be done than there are people available to do it.

Since the FreeBSD project is responsible for an entire operating system environment (and its installation) rather than just a kernel or a few scattered utilities, our TODO list also spans a very wide range of tasks, from documentation, beta testing and presentation to highly specialized types of kernel development. No matter what your skill level, there is almost certainly something you can do to help the project!

Commercial entities engaged in FreeBSD-related enterprises are also encouraged to contact us. Need a special extension to make your product work? You will find us receptive to your requests, given that they are not too outlandish. 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 a lot of existing assumptions about how software is developed, sold, and maintained throughout its life cycle, and we urge you to at least give it a second look.

## What Is Needed

The following list of tasks and sub-projects represents something of an amalgam of the various core team TODO lists and user requests we have collected over the last couple of months. Where possible, tasks have been ranked by degree of urgency. If you are interested in working on one of the tasks you see here, send mail to the coordinator listed by clicking on their names. If no coordinator has been appointed, maybe you would like to volunteer?

## High priority tasks

The following tasks are considered to be urgent, usually because they represent something that is badly broken or sorely needed:

1. 3-stage boot issues. Overall coordination: FreeBSD technical discussions mailing list  
<freebsd-hackers@FreeBSD.ORG>
  - Do WinNT compatible drive tagging so that the 3rd stage can provide an accurate mapping of BIOS geometries for disks.
2. Filesystem problems. Overall coordination: FreeBSD filesystem project mailing list  
<freebsd-fs@FreeBSD.ORG>
  - Fix the MSDOS file system.
  - Clean up and document the nullfs filesystem code. Coordinator: Eivind Eklund  
<eivind@FreeBSD.ORG>
  - Fix the union file system. Coordinator: David Greenman <dg@FreeBSD.ORG>
3. Implement Int13 vm86 disk driver. Coordinator: FreeBSD technical discussions mailing list  
<freebsd-hackers@FreeBSD.ORG>
4. New bus architecture. Coordinator: New Bus Architecture mailing list  
<new-bus-arch@bostonradio.org>
  - Port existing ISA drivers to new architecture.
  - Move all interrupt-management code to appropriate parts of the bus drivers.
  - Port PCI subsystem to new architecture. Coordinator: Doug Rabson <dfr@FreeBSD.ORG>
  - Figure out the right way to handle removable devices and then use that as a substrate on which PC-Card and CardBus support can be implemented.
  - Resolve the probe/attach priority issue once and for all.
  - Move any remaining buses over to the new architecture.
5. Kernel issues. Overall coordination: FreeBSD technical discussions mailing list  
<freebsd-hackers@FreeBSD.ORG>
6. Add more pro-active security infrastructure. Overall coordination: FreeBSD security mailing list  
<freebsd-security@FreeBSD.ORG>
  - Build something like Tripwire(TM) into the kernel, with a remote and local part. There are a number of cryptographic issues to getting this right; contact the coordinator for details.  
Coordinator: Eivind Eklund <eivind@FreeBSD.ORG>
  - Make the entire kernel use `suser()` instead of comparing to 0. It is presently using about half of each. Coordinator: Eivind Eklund <eivind@FreeBSD.ORG>

- Split `securelevels` into different parts, to allow an administrator to throw away those privileges he can throw away. Setting the overall `securelevel` needs to have the same effect as now, obviously. Coordinator: Eivind Eklund <eivind@FreeBSD.ORG>
- Make it possible to upload a list of “allowed program” to BPF, and then block BPF from accepting other programs. This would allow BPF to be used e.g. for DHCP, without allowing an attacker to start snooping the local network.
- Update the security checker script. We should at least grab all the checks from the other BSD derivatives, and add checks that a system with `securelevel` increased also have reasonable flags on the relevant parts. Coordinator: Eivind Eklund <eivind@FreeBSD.ORG>
- Add authorization infrastructure to the kernel, to allow different authorization policies. Part of this could be done by modifying `suser()`. Coordinatory: Eivind Eklund <eivind@FreeBSD.ORG>
- Add code to teh NFS layer so that you cannot `chdir("..")` out of an NFS partition. E.g., `/usr` is a UFS partition with `/usr/src` NFS exported. Now it is possible to use the NFS filehandle for `/usr/src` to get access to `/usr`.

## Medium priority tasks

The following tasks need to be done, but not with any particular urgency:

1. Full KLD based driver support/Configuration Manager.
  - Write a configuration manager (in the 3rd stage boot?) that probes your hardware in a sane manner, keeps only the KLDs required for your hardware, etc.
2. PCMCIA/PCCARD. Coordinators: Michael Smith <msmith@FreeBSD.ORG> and Poul-Henning Kamp <phk@FreeBSD.ORG>
  - Documentation!
  - Reliable operation of the `pcic` driver (needs testing).
  - Recognizer and handler for `sio.c` (mostly done).
  - Recognizer and handler for `ed.c` (mostly done).
  - Recognizer and handler for `ep.c` (mostly done).
  - User-mode recognizer and handler (partially done).
3. Advanced Power Management. Coordinators: Michael Smith <msmith@FreeBSD.ORG> and Poul-Henning Kamp <phk@FreeBSD.ORG>

- APM sub-driver (mostly done).
- IDE/ATA disk sub-driver (partially done).
- syscons/pcvt sub-driver.
- Integration with the PCMCIA/PCCARD drivers (suspend/resume).

## Low priority tasks

The following tasks are purely cosmetic or represent such an investment of work that it is not likely that anyone will get them done anytime soon:

The first N items are from Terry Lambert <terry@lambert.org>

1. NetWare Server (protected mode ODI driver) loader and subservices to allow the use of ODI card drivers supplied with network cards. The same thing for NDIS drivers and NetWare SCSI drivers.
2. An "upgrade system" option that works on Linux boxes instead of just previous rev FreeBSD boxes.
3. Symmetric Multiprocessing with kernel preemption (requires kernel preemption).
4. A concerted effort at support for portable computers. This is somewhat handled by changing PCMCIA bridging rules and power management event handling. But there are things like detecting internal vs. external display and picking a different screen resolution based on that fact, not spinning down the disk if the machine is in dock, and allowing dock-based cards to disappear without affecting the machines ability to boot (same issue for PCMCIA).

## Smaller tasks

Most of the tasks listed in the previous sections 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", or people without programming skills.

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 and install the latest release from it and report any failures in the process.
2. Read the <freebsd-bugs> mailing list. 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. 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 (SGML is not difficult to learn,

but there is no objection to ASCII submissions).

4. Help translate FreeBSD documentation into your native language (if not already available) — just send an email to FreeBSD documentation project mailing list <freebsd-doc@FreeBSD.ORG> asking if anyone is working on it. Note that you are not committing yourself to translating every single FreeBSD document by doing this — in fact, the documentation most in need of translation is the installation instructions.
5. Read the freebsd-questions mailing list and the comp.unix.bsd.freebsd.misc (news:comp.unix.bsd.freebsd.misc) newsgroup 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.
6. If you know of any bugfixes 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.
7. Move contributed software to `src/contrib` in the source tree.
8. Make sure code in `src/contrib` is up to date.
9. Look for year 2000 bugs (and fix any you find!)
10. Build the source tree (or just part of it) with extra warnings enabled and clean up the warnings.
11. Fix warnings for ports which do deprecated things like using `gets()` or including `malloc.h`.
12. If you have contributed any ports, send your patches back to the original author (this will make your life easier when they bring out the next version)
13. Suggest further tasks for this list!

## How to Contribute

Contributions to the system generally fall into one or more of the following 6 categories:

### Bug reports and general commentary

An idea or suggestion of *general* technical interest should be mailed to the FreeBSD technical discussions mailing list <freebsd-hackers@FreeBSD.ORG>. Likewise, people with an interest in such things (and a tolerance for a *high* volume of mail!) may subscribe to the hackers mailing list by sending mail to <majordomo@FreeBSD.ORG>. See mailing lists 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 `send-pr(1)` program or its WEB-based equivalent (<http://www.freebsd.org/send-pr.html>). Try to fill-in each field of the bug report. Unless they exceed 65KB, include any patches directly in the report. Consider compressing them and using `uuencode(1)` if they exceed 20KB. Upload very large submissions to <ftp.freebsd.org/pub/FreeBSD/incoming/> (<ftp://ftp.FreeBSD.ORG/pub/FreeBSD/incoming/>).

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 by sending mail to `<bug-followup@FreeBSD.ORG>`. Use the number as the message subject, e.g. "Re: kern/3377". Additional information for any bug report should be submitted this way.

If you do not receive confirmation in a timely fashion (3 days to a week, depending on your email connection) or are, for some reason, unable to use the `send-pr(1)` command, then you may ask someone to file it for you by sending mail to the FreeBSD problem reports mailing list `<freebsd-bugs@FreeBSD.ORG>`.

## Changes to the documentation

Changes to the documentation are overseen by the FreeBSD documentation project mailing list `<freebsd-doc@FreeBSD.ORG>`. Send submissions and changes (even small ones are welcome!) using `send-pr` as described in Bug Reports and General Commentary.

## Changes to existing source code

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 the core 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 *Staying current with FreeBSD* 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 FreeBSD announcements mailing list `<freebsd-announce@FreeBSD.ORG>` and the FreeBSD-current mailing list `<freebsd-current@FreeBSD.ORG>` 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 `diff(1)` command, with the "context diff" form being preferred. For example:

```
% diff -c oldfile newfile
```

or

```
% diff -c -r olddir newdir
```

would generate such a set of context diffs for the given source file or directory hierarchy. See the man page for `diff(1)` for more details.

Once you have a set of diffs (which you may test with the `patch(1)` command), you should submit them for inclusion with FreeBSD. Use the `send-pr(1)` program as described in Bug Reports and General Commentary. *Do not* just send the diffs to the FreeBSD technical discussions mailing list `<freebsd-hackers@FreeBSD.ORG>` 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.

If you feel it appropriate (e.g. you have added, deleted, or renamed files), bundle your changes into a `tar` file and run the `uuencode(1)` program on it. Shar archives are also welcome.

If your change is of a potentially sensitive nature, e.g. you are unsure of copyright issues governing its further distribution or you are simply not ready to release it without a tighter review first, then you should send it to FreeBSD core team `<freebsd-core@FreeBSD.ORG>` directly rather than submitting it with `send-pr(1)`. The core mailing list 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 9 intro` and `man 9 style` for some information on coding style. We would appreciate it if you were at least aware of this information before submitting code.

## New code or major value-added packages

In the rare 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 `uuencode`'d tar files or upload them to our ftp site `ftp://ftp.FreeBSD.ORG/pub/FreeBSD/incoming`.

When working with large amounts of code, the touchy subject of copyrights also invariably comes up. Acceptable copyrights for code included in FreeBSD are:

1. The BSD copyright. This copyright is most preferred due to its “no strings attached” nature and general attractiveness to commercial enterprises. Far from discouraging such commercial use, the FreeBSD Project actively encourages such participation by commercial interests who might eventually be inclined to invest something of their own into FreeBSD.
2. The GNU Public License, or “GPL”. This license is not quite as popular with us due to the amount of extra effort demanded of anyone using the code for commercial purposes, but given the sheer

quantity of GPL'd code we currently require (compiler, assembler, text formatter, etc) it would be silly to refuse additional contributions under this license. Code under the GPL also goes into a different part of the tree, that being `/sys/gnu` or `/usr/src/gnu`, and is therefore easily identifiable to anyone for whom the GPL presents a problem.

Contributions coming under any other type of copyright must be carefully reviewed before their inclusion into FreeBSD will be considered. Contributions for which particularly restrictive commercial copyrights apply are generally rejected, though the authors are always encouraged to make such changes available through their own channels.

To place a "BSD-style" copyright on your work, include the following text at the very beginning of every source code file you wish to protect, replacing the text between the `%%` with the appropriate information.

```
Copyright (c) %%proper_years_here%%
    %%your_name_here%%, %%your_state%% %%your_zip%%. 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 as the first lines of this file unmodified.
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 %%your_name_here%% "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 %%your_name_here%% 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 convenience, a copy of this text can be found in  
`/usr/share/examples/etc/bsd-style-copyright`.

## Money, Hardware or Internet access

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

While the FreeBSD Project is not a 501(c)(3) (charitable) corporation and hence cannot offer special tax incentives for any donations made, any such donations will be gratefully accepted on behalf of the project by FreeBSD, Inc.

FreeBSD, Inc. was founded in early 1995 by Jordan K. Hubbard <jkh@FreeBSD.ORG> and David Greenman <dg@FreeBSD.ORG> with the goal of furthering the aims of the FreeBSD Project and giving it a minimal corporate presence. Any and all funds donated (as well as any profits that may eventually be realized by FreeBSD, Inc.) will be used exclusively to further the project's goals.

Please make any checks payable to FreeBSD, Inc., sent in care of the following address:

FreeBSD, Inc.  
c/o Jordan Hubbard  
4041 Pike Lane, Suite F  
Concord  
CA, 94520

(currently using the Walnut Creek CDRom address until a PO box can be opened)

Wire transfers may also be sent directly to:

Bank Of America  
Concord Main Office  
P.O. Box 37176  
San Francisco  
CA, 94137-5176

Routing #: 121-000-358  
Account #: 01411-07441 (FreeBSD, Inc.)

Any correspondence related to donations should be sent to Jordan Hubbard <jkh@FreeBSD.org>, either via email or to the FreeBSD, Inc. postal address given above.

If you do not wish to be listed in our donors section, please specify this when making your donation.  
Thanks!

## Donating hardware

Donations of hardware in any of the 3 following categories are also gladly accepted by the FreeBSD Project:

- General purpose hardware such as disk drives, memory or complete systems should be sent to the FreeBSD, Inc. address listed in the *donating funds* section.
- Hardware for which ongoing compliance testing is desired. We are currently trying to put together a testing lab of all components that FreeBSD supports so that proper regression testing can be done with each new release. We are still lacking many important pieces (network cards, motherboards, etc) and if you would like to make such a donation, please contact David Greenman <dg@FreeBSD.ORG> for information on which items are still required.
- Hardware currently unsupported by FreeBSD for which you would like to see such support added. Please contact the FreeBSD core team <freebsd-core@FreeBSD.ORG> before sending such items as we will need to find a developer willing to take on the task before we can accept delivery of new hardware.

## Donating Internet access

We can always use new mirror sites for FTP, WWW or cvsup. If you would like to be such a mirror, please contact the FreeBSD project administrators <admin@FreeBSD.ORG> for more information.

## Donors Gallery

The FreeBSD Project is indebted to the following donors and would like to publically thank them here!

- *Contributors to the central server project:*

The following individuals and businesses made it possible for the FreeBSD Project to build a new central server machine to eventually replace `freefall.freebsd.org` by donating the following items:

- Ade Barkah <mbarkah@freebsd.org> and his employer, Hemisphere Online (<http://www.hemi.com>), donated a *Pentium Pro (P6) 200Mhz CPU*
- ASA Computers (<http://www.asacomputers.com>) donated a *Tyan 1662 motherboard*.

- Joe McGuckin <joe@via.net> of ViaNet Communications (<http://www.via.net>) donated a *Kingston ethernet controller*.
- Jack O'Neill <jack@diamond.xtalwind.net> donated an *NCR 53C875 SCSI controller card*.
- Ulf Zimmermann <ulf@Alameda.net> of Alameda Networks (<http://www.Alameda.net>) donated *128MB of memory, a 4 Gb disk drive and the case*.
- *Direct funding:*

The following individuals and businesses have generously contributed direct funding to the project:

  - Annelise Anderson <ANDRSN@HOOVER.STANFORD.EDU>
  - Matt Dillon <dillon@best.net>
  - Epilogue Technology Corporation (<http://www.epilogue.com/>)
  - Sean Eric Fagan
  - Don Scott Wilde
  - Gianmarco Giovannelli <gmarco@masternet.it>
  - Josef C. Grosch <joeg@truenorth.org>
  - Robert T. Morris
  - Chuck Robey <chuckr@freebsd.org>
  - Kenneth P. Stox <ken@stox.sa.interact.com> of Imaginary Landscape, LLC. (<http://www.imagescape.com>)
  - Dmitry S. Kohmanyuk <dk@dog.farm.org>
  - Laser5 (<http://www.cdrom.co.jp/>) of Japan (a portion of the profits from sales of their various FreeBSD CD-ROMs).
  - Fuki Shuppan Publishing Co. (<http://www.mmjp.or.jp/fuki/>) donated a portion of their profits from *Hajimete no FreeBSD* (FreeBSD, Getting started) to the FreeBSD and XFree86 projects.
  - ASCII Corp. (<http://www.ascii.co.jp/>) donated a portion of their profits from several FreeBSD-related books to the FreeBSD project.
  - Yokogawa Electric Corp (<http://www.yokogawa.co.jp/>) has generously donated significant funding to the FreeBSD project.
  - BuffNET (<http://www.buffnet.net/>)
  - Pacific Solutions (<http://www.pacificsolutions.com/>)
- *Hardware contributors:*

The following individuals and businesses have generously contributed hardware for testing and device driver development/support:

- Walnut Creek CDROM for providing the Pentium P5-90 and 486/DX2-66 EISA/VL systems that are being used for our development work, to say nothing of the network access and other donations of hardware resources.
- TRW Financial Systems, Inc. provided 130 PCs, three 68 GB file servers, twelve Ethernets, two routers and an ATM switch for debugging the diskless code.
- Dermot McDonnell donated the Toshiba XM3401B CDROM drive currently used in freefall.
- Chuck Robey <chuckr@glue.umd.edu> contributed his floppy tape streamer for experimental work.
- Larry Altneu <larry@ALR.COM>, and Wilko Bulte <wilko@yedi.iaf.nl>, provided Wangtek and Archive QIC-02 tape drives in order to improve the wt driver.
- Ernst Winter <ewinter@lobo.muc.de> contributed a 2.88 MB floppy drive to the project. This will hopefully increase the pressure for rewriting the floppy disk driver. ;-)
- Tekram Technologies (<http://www.tekram.com>) sent one each of their DC-390, DC-390U and DC-390F FAST and ULTRA SCSI host adapter cards for regression testing of the NCR and AMD drivers with their cards. They are also to be applauded for making driver sources for free operating systems available from their FTP server <ftp://ftp.tekram.com/scsi/FreeBSD>.
- <Larry M. Augustin> contributed not only a Symbios Sym8751S SCSI card, but also a set of data books, including one about the forthcoming Sym53c895 chip with Ultra-2 and LVD support, and the latest programming manual with information on how to safely use the advanced features of the latest Symbios SCSI chips. Thanks a lot!
- Christoph Kukulies <kuku@freebsd.org> donated an FX120 12 speed Mitsumi CDROM drive for IDE CDROM driver development.
- *Special contributors:*
  - Walnut Creek CDROM (<http://www.cdrom.com>) has donated almost more than we can say (see the history document for more details). In particular, we would like to thank them for the original hardware used for `freefall.FreeBSD.ORG`, our primary development machine, and for `thud.FreeBSD.ORG`, a testing and build box. We are also indebted to them for funding various contributors over the years and providing us with unrestricted use of their T1 connection to the Internet.
  - The interface business GmbH, Dresden (<http://www.interface-business.de>) has been patiently supporting Jörg Wunsch <joerg@FreeBSD.ORG> who has often preferred FreeBSD work over

paywork, and used to fall back to their (quite expensive) EUnet Internet connection whenever his private connection became too slow or flakey to work with it...

- Berkeley Software Design, Inc. (<http://www.bsdi.com>) has contributed their DOS emulator code to the remaining BSD world, which is used in the *dosemu* command.

## Core Team Alumnus

The following people were members of the FreeBSD core team during the period indicated. We thank them for their past efforts in the service of the FreeBSD project.

*In rough chronological order:*

- Guido van Rooij (1995 - 1999)
- John Dyson (1993 - 1998)
- Nate Williams (1992 - 1996)
- Rod Grimes (1992 - 1995)
- Andreas Schulz (1992 - 1995)
- Geoff Rehmert (1993 - 1995)
- Paul Richards (1992 - 1995)
- Scott Mace (1993 - 1994)
- Andrew Moore (1993 - 1994)
- Christoph Robitschko (1993 - 1994)
- J. T. Conklin (1992 - 1993)

## Derived Software Contributors

This software was originally derived from William F. Jolitz's 386BSD release 0.1, though almost none of the original 386BSD specific code remains. This software has been essentially re-implemented from the 4.4BSD-Lite release provided by the Computer Science Research Group (CSRG) at the University of California, Berkeley and associated academic contributors.

There are also portions of NetBSD and OpenBSD that have been integrated into FreeBSD as well, and we would therefore like to thank all the contributors to NetBSD and OpenBSD for their work.

## Additional FreeBSD Contributors

(in alphabetical order by first name):

- ABURAYA Ryushirou <rewsirow@ff.iiij4u.or.jp>
- AMAGAI Yoshiji <amagai@nue.org>
- Aaron Bornstein <aaronb@j51.com>
- Aaron Smith <aaron@tau.veritas.com>
- Achim Patzner <ap@noses.com>
- Ada T Lim <ada@bsd.org>
- Adam Baran <badam@mw.mil.pl>
- Adam Glass <glass@postgres.berkeley.edu>
- Adam McDougall <mcdouga9@egr.msu.edu>
- Adrian Colley <aecolley@ois.ie>
- Adrian Hall <adrian@ibmpcug.co.uk>
- Adrian Mariano <adrian@cam.cornell.edu>
- Adrian Steinmann <ast@marabu.ch>
- Adrian T. Filipi-Martin <atf3r@agate.cs.virginia.edu>
- Ajit Thyagarajan <unknown>
- Akio Morita <amorita@meadow.scphys.kyoto-u.ac.jp>
- Akira SAWADA <unknown>
- Akira Watanabe <akira@myaw.ei.meisei-u.ac.jp>
- Akito Fujita <fujita@zoo.ncl.omron.co.jp>
- Alain Kalker <A.C.P.M.Kalker@student.utwente.nl>
- Alan Bawden <alan@curry.epilogue.com>
- Alan Cox <alc@cs.rice.edu>
- Alec Wolman <wolman@cs.washington.edu>
- Aled Morris <aledm@routers.co.uk>
- Alex <garbanzo@hooked.net>
- Alex D. Chen <dhchen@Canvas.dorm7.nccu.edu.tw>

- Alex G. Bulushev <bag@demos.su>
- Alex Le Heux <alexlh@funk.org>
- Alexander B. Povolotsky <tarkhil@mgt.msk.ru>
- Alexander Leidinger <netchild@wurzelauisix.CS.Uni-SB.DE>
- Alexandre Snarskii <snar@paranoia.ru>
- Alistair G. Crooks <agc@uts.amdahl.com>
- Allan Saddi <asaddi@philosophysw.com>
- Allen Campbell <allenc@verinet.com>
- Amakawa Shuhei <amakawa@hoh.t.u-tokyo.ac.jp>
- Amancio Hasty <hasty@star-gate.com>
- Amir Farah <amir@control.com>
- Amy Baron <amee@beer.org>
- Anatoly A. Orehovskiy <tolik@mpeks.tomsk.su>
- Anatoly Vorobey <mellon@pobox.com>
- Anders Nordby <nickerne@nome.no>
- Anders Thulin <Anders.X.Thulin@telia.se>
- Andras Olah <olah@cs.utwente.nl>
- Andre Albsmeier <Andre.Albsmeier@mchp.siemens.de>
- Andre Oppermann <andre@pipeline.ch>
- Andreas Haakh <ah@alman.robin.de>
- Andreas Kohout <shanee@rabbit.augusta.de>
- Andreas Lohr <andreas@marvin.ROBIN.de>
- Andreas Schulz <unknown>
- Andreas Wetzel <mickey@deadline.snafu.de>
- Andreas Wrede <andreas@planix.com>
- Andres Vega Garcia <unknown>
- Andrew Atrens <atreand@statcan.ca>
- Andrew Gillham <gillham@andrews.edu>
- Andrew Gordon <andrew.gordon@net-tel.co.uk>

- Andrew Herbert <andrew@werple.apana.org.au>
- Andrew J. Korty <ajk@purdue.edu>
- Andrew L. Moore <alm@mclink.com>
- Andrew McRae <amcrae@cisco.com>
- Andrew Stevenson <andrew@ugh.net.au>
- Andrew Timonin <tim@pool1.convey.ru>
- Andrew V. Stesin <stesin@elvisti.kiev.ua>
- Andrew Webster <awebster@dataradio.com>
- Andrey Zakhvatov <andy@icc.surw.chel.su>
- Andy Farkas <andyf@speednet.com.au>
- Andy Valencia <ajv@csd.mot.com>
- Andy Whitcroft <andy@sarc.city.ac.uk>
- Angelo Turetta <ATuretta@stylo.it>
- Anthony C. Chavez <magus@xmission.com>
- Anthony Yee-Hang Chan <yeehang@netcom.com>
- Anton Berezin <tobez@plab.ku.dk>
- Antti Kaipila <anttik@iki.fi>
- Are Bryne <are.bryne@communique.no>
- Ari Suutari <ari@suutari.iki.fi>
- Arjan de Vet <devet@IAEhv.nl>
- Arne Henrik Juul <arnej@Lise.Unit.NO>
- Assar Westerlund <assar@sics.se>
- Atsushi Furuta <furuta@sra.co.jp>
- Atsushi Murai <amurai@spec.co.jp>
- Bakul Shah <bvs@bitblocks.com>
- Barry Bierbauch <pivrnc@vszbr.cz>
- Barry Lustig <barry@ictv.com>
- Ben Hutchinson <benhutch@xfiles.org.uk>
- Ben Jackson <unknown>

- Ben Smithurst <ben@scientia.demon.co.uk>
- Ben Walter <bwalter@itachi.swcp.com>
- Benjamin Lewis <bhlewis@gte.net>
- Bernd Rosauer <br@schiele-ct.de>
- Bill Kish <kish@osf.org>
- Bill Trost <trost@cloud.rain.com>
- Blaz Zupan <blaz@amis.net>
- Bob Van Valzah <Bob@whitebarn.com>
- Bob Willcox <bob@luke.pmr.com>
- Boris Staeblow <balu@dva.in-berlin.de>
- Boyd R. Faulkner <faulkner@asgard.bga.com>
- Brad Karp <karp@eecs.harvard.edu>
- Bradley Dunn <bradley@dunn.org>
- Brandon Gillespie <brandon@roguetrader.com>
- Bill Lloyd <wlloyd@mpd.ca>
- Bob Wilcox <bob@obiwan.uucp>
- Boyd Faulkner <faulkner@mpd.tandem.com>
- Brent J. Nordquist <bjn@visi.com>
- Brett Lymn <blymn@mulga.awadi.com.AU>
- Brett Taylor <brett@peloton.physics.montana.edu>
- Brian Campbell <brianc@pobox.com>
- Brian Clapper <bmc@willscreek.com>
- Brian Cully <shmit@kublai.com>
- Brian F. Feldman <green@unixhelp.org>
- Brian Handy <handy@lambic.space.lockheed.com>
- Brian Litzinger <brian@MediaCity.com>
- Brian McGovern <bmcgover@cisco.com>
- Brian Moore <ziff@houdini.eecs.umich.edu>
- Brian R. Haug <haug@conterra.com>

- Brian Tao <taob@risc.org>
- Brion Moss <brion@queeg.com>
- Bruce A. Mah <bmah@ca.sandia.gov>
- Bruce Albrecht <bruce@zuhause.mn.org>
- Bruce Gingery <bgingery@gtcs.com>
- Bruce J. Keeler <lloodvrij@gridpoint.com>
- Bruce Murphy <packrat@iinet.net.au>
- Bruce Walter <walter@fortean.com>
- Carey Jones <mcj@acquiesce.org>
- Carl Fongheiser <cmf@netins.net>
- Carl Mascott <cmascott@world.std.com>
- Casper <casper@acc.am>
- Castor Fu <castor@geocast.com>
- Cejka Rudolf <cejkar@dcse.fee.vutbr.cz>
- Chain Lee <chain@110.net>
- Charles Hannum <mycroft@ai.mit.edu>
- Charles Henrich <henrich@msu.edu>
- Charles Mott <cmott@srv.net>
- Charles Owens <owensc@enc.edu>
- Chet Ramey <chet@odin.INS.CWRU.Edu>
- Chia-liang Kao <clkao@CirX.ORG>
- Chiharu Shibata <chi@bd.mbn.or.jp>
- Chip Norkus <unknown>
- Choi Jun Ho <junker@jazz.snu.ac.kr>
- Chris Csanady <cc@tarsier.ca.sandia.gov>
- Chris Dabrowski <chris@vader.org>
- Chris Dillon <cdillon@wolves.k12.mo.us>
- Chris Piazza <cpiazza@home.net>
- Chris Shenton <cshenton@angst.it.hq.nasa.gov>

- Chris Stenton <jacs@gnome.co.uk>
- Chris Timmons <skynyrd@opus.cts.cwu.edu>
- Chris Torek <torek@ee.lbl.gov>
- Christian Gusenbauer <cg@fimp01.fim.uni-linz.ac.at>
- Christian Haury <Christian.Haury@sagem.fr>
- Christian Weisgerber <naddy@bigeye.rhein-neckar.de>
- Christoph P. Kukulies <kuku@FreeBSD.org>
- Christoph Robitschko <chmr@edvz.tu-graz.ac.at>
- Christoph Weber-Fahr <wefa@callcenter.systemhaus.net>
- Christopher G. Demetriou <cgd@postgres.berkeley.edu>
- Christopher T. Johnson <cjohnson@neunacht.netgsi.com>
- Chrisy Luke <chrisy@flix.net>
- Chuck Hein <chein@cisco.com>
- Clive Lin <clive@CiRX.ORG>
- Colman Reilly <careilly@tcd.ie>
- Conrad Sabatier <conrads@neosoft.com>
- Coranth Gryphon <gryphon@healer.com>
- Cornelis van der Laan <nils@guru.ims.uni-stuttgart.de>
- Cove Schneider <cove@brazil.nbn.com>
- Craig Leres <leres@ee.lbl.gov>
- Craig Loomis <unknown>
- Craig Metz <cmetz@inner.net>
- Craig Spannring <cts@internetcds.com>
- Craig Struble <cstruble@vt.edu>
- Cristian Ferretti <cfs@riemann.mat.puc.cl>
- Curt Mayer <curt@toad.com>
- Cy Schubert <cschuber@uumail.gov.bc.ca>
- DI. Christian Gusenbauer <cg@scotty.edvz.uni-linz.ac.at>
- Dai Ishijima <ishijima@tri.pref.osaka.jp>

- Damian Hamill <damian@cablenet.net>
- Dan Cross <tenser@spitfire.ecsel.psu.edu>
- Dan Lukes <dan@obluda.cz>
- Dan Nelson <dnelson@emsphone.com>
- Dan Walters <hannibal@cyberstation.net>
- Daniel Baker <dbaker@crash.ops.neosoft.com>
- Daniel M. Eischen <deischen@iworks.InterWorks.org>
- Daniel O'Connor <doconnor@gsoft.com.au>
- Daniel Poirot <poirot@aio.jsc.nasa.gov>
- Daniel Rock <rock@cs.uni-sb.de>
- Danny Egen <unknown>
- Danny J. Zerkel <dzerkel@phofarm.com>
- Darren Reed <avalon@coombs.anu.edu.au>
- Dave Adkins <adkin003@tc.umn.edu>
- Dave Andersen <angio@aros.net>
- Dave Blizzard <dblizzard@sprynet.com>
- Dave Bodenshtab <imdave@synet.net>
- Dave Burgess <burgess@hrd769.brooks.af.mil>
- Dave Chapeskie <dchapes@ddm.on.ca>
- Dave Cornejo <dave@dogwood.com>
- Dave Edmondson <davided@sco.com>
- Dave Glowacki <dglo@ssec.wisc.edu>
- Dave Marquardt <marquard@austin.ibm.com>
- Dave Tweten <tweten@FreeBSD.org>
- David A. Adkins <adkin003@tc.umn.edu>
- David A. Bader <dbader@umiacs.umd.edu>
- David Borman <dab@bsd.i.com>
- David Dawes <dawes@XFree86.org>
- David Filo <filo@yahoo.com>

- David Holland <d holland@eecs.harvard.edu>
- David Holloway <daveh@gwythaint.tamis.com>
- David Horwitt <dhorwitt@ucsd.edu>
- David Hovemeyer <daveho@info.com>
- David Jones <dej@qpoint.torfree.net>
- David Kelly <dkelly@tomcat1.tbe.com>
- David Kulp <dkulp@neomorphic.com>
- David L. Nugent <davidn@blaze.net.au>
- David Leonard <d@scry.dstc.edu.au>
- David Malone <dwmalone@maths.tcd.ie>
- David Muir Sharnoff <muir@idiom.com>
- David S. Miller <davem@jenolan.rutgers.edu>
- David Wolfskill <dhw@whistle.com>
- Dean Gaudet <dgaudet@arctic.org>
- Dean Huxley <dean@fsa.ca>
- Denis Fortin <unknown>
- Dennis Glatting <dennis.glatting@software-munitions.com>
- Denton Gentry <denny1@home.com>
- Derek Inksetter <derek@saidev.com>
- Dima Sivachenko <dima@Chg.RU>
- Dirk Keunecke <dk@panda.rhein-main.de>
- Dirk Nehrling <nerle@pdv.de>
- Dmitry Khrustalev <dima@xyzyzy.machaon.ru>
- Dmitry Kohmanyuk <dk@farm.org>
- Dom Mitchell <dom@myrddin.demon.co.uk>
- Don Croyle <croyle@gelemna.ft-wayne.in.us>
- Don Whiteside <whiteside@acm.org>
- Don Morrison <dmorrisn@u.washington.edu>
- Don Yuniskis <dgy@rtd.com>

- Donald Maddox <dmaddox@conterra.com>
- Doug Barton <studded@dal.net>
- Douglas Ambrisko <ambrisko@whistle.com>
- Douglas Carmichael <dcarmich@mcs.com>
- Douglas Crosher <dte@scrooge.ee.swin.oz.au>
- Drew Derbyshire <ahd@kew.com>
- Duncan Barclay <dmlb@ragnet.demon.co.uk>
- Dustin Sallings <dustin@spy.net>
- Eckart "Isegrim" Hofmann <Isegrim@Wunder-Nett.org>
- Ed Gold <vegold01@starbase.spd.louisville.edu>
- Ed Hudson <elh@p5.spnet.com>
- Edward Wang <edward@edcom.com>
- Edwin Groothuis <edwin@nwm.wan.philips.com>
- Eiji-usagi-MATSUMOTO <usagi@clave.gr.jp>
- ELISA Font Project
- Elmar Bartel <bartel@informatik.tu-muenchen.de>
- Eric A. Griff <eagriff@global2000.net>
- Eric Blood <eblood@cs.unr.edu>
- Eric J. Haug <ejh@slustl.slu.edu>
- Eric J. Schwertfeger <eric@cybernut.com>
- Eric L. Hernes <erich@lodgenet.com>
- Eric P. Scott <eps@sirius.com>
- Eric Sprinkle <eric@ennovatenetworks.com>
- Erich Stefan Boleyn <erich@uruk.org>
- Erik E. Rantapaa <rantapaa@math.umn.edu>
- Erik H. Moe <ehm@cris.com>
- Ernst Winter <ewinter@lobo.muc.de>
- Eugene M. Kim <astralblue@usa.net>
- Eugene Radchenko <genie@qsar.chem.msu.su>

- Evan Champion <evanc@synapse.net>
- Faried Nawaz <fn@Hungry.COM>
- Flemming Jacobsen <fj@tfs.com>
- Fong-Ching Liaw <fong@juniper.net>
- Francis M J Hsieh <mjshieh@life.nthu.edu.tw>
- Frank Bartels <knarf@camelot.de>
- Frank Chen Hsiung Chan <frankch@waru.life.nthu.edu.tw>
- Frank Durda IV <uhclem@nemesis.lonestar.org>
- Frank MacLachlan <fpm@n2.net>
- Frank Nobis <fn@Radio-do.de>
- Frank Volf <volf@oasis.IAEhv.nl>
- Frank ten Wolde <franky@pinewood.nl>
- Frank van der Linden <frank@fwi.uva.nl>
- Fred Cawthorne <fcawth@jjarray.umn.edu>
- Fred Gilham <gilham@csl.sri.com>
- Fred Templin <templin@erg.sri.com>
- Frederick Earl Gray <fgray@rice.edu>
- FUJIMOTO Kensaku <fujimoto@oscar.elec.waseda.ac.jp>
- FUJISHIMA Satsuki <k5@respo.or.jp>
- FURUSAWA Kazuhisa <furusawa@com.cs.osakafu-u.ac.jp>
- Gabor Kincses <gabor@acm.org>
- Gabor Zahemszky <zgabor@CoDe.hu>
- Garance A Drosehn <gad@eclipse.its.rpi.edu>
- Gareth McCaughan <gjm11@dpms.cam.ac.uk>
- Gary A. Browning <gab10@griffcd.amdahl.com>
- Gary Howland <gary@hotlava.com>
- Gary J. <garyj@rks32.pcs.dec.com>
- Gary Kline <kline@thought.org>
- Gaspar Chilingarov <nightmar@lemming.acc.am>

- Gea-Suan Lin <gsl@tpts4.seed.net.tw>
- Geoff Rehmet <csgr@alpha.ru.ac.za>
- Georg Wagner <georg.wagner@ubs.com>
- Gerard Roudier <groudier@club-internet.fr>
- Gianmarco Giovannelli <gmarco@giovannelli.it>
- Gil Kloepfer Jr. <gil@limbic.ssd1.com>
- Gilad Rom <rom\_glsa@ein-hashofet.co.il>
- Ginga Kawaguti <ginga@amalthea.phys.s.u-tokyo.ac.jp>
- Giles Lean <giles@nemeton.com.au>
- Glen Foster <gfoster@gfoster.com>
- Glenn Johnson <gljohns@bellsouth.net>
- Godmar Back <gback@facility.cs.utah.edu>
- Goran Hammarback <goran@astro.uu.se>
- Gord Matzigkeit <gord@enci.ucalgary.ca>
- Graham Wheeler <gram@cdsec.com>
- Greg A. Woods <woods@zeus.leitch.com>
- Greg Ansley <gja@ansley.com>
- Greg Troxel <gdt@ir.bbn.com>
- Greg Ungerer <gerg@stallion.oz.au>
- Gregory Bond <gnb@itga.com.au>
- Gregory D. Moncreaff <moncrg@bt340707.res.ray.com>
- Guy Harris <guy@netapp.com>
- Guy Helmer <ghelmer@cs.iastate.edu>
- HAMADA Naoki <hamada@astec.co.jp>
- HONDA Yasuhiro <honda@kashio.info.mie-u.ac.jp>
- HOSOBUCHI Noriyuki <hoso@buchi.tama.or.jp>
- Hannu Savolainen <hannu@voxware.pp.fi>
- Hans Huebner <hans@artcom.de>
- Hans Petter Bieker <zerium@webindex.no>

- Hans Zuidam <hans@brandinnovators.com>
- Harlan Stenn <Harlan.Stenn@pfcs.com>
- Harold Barker <hbarker@dsms.com>
- Havard Eidnes <Havard.Eidnes@runit.sintef.no>
- Heikki Suonsivu <hsu@cs.hut.fi>
- Heiko W. Rupp <unknown>
- Helmut F. Wirth <hfwirth@ping.at>
- Henrik Vestergaard Draboel <hvd@terry.ping.dk>
- Herb Peyerl <hpeyerl@NetBSD.org>
- Hideaki Ohmon <ohmon@tom.sfc.keio.ac.jp>
- Hidekazu Kuroki <hidekazu@cs.titech.ac.jp>
- Hideki Yamamoto <hyama@acm.org>
- Hidetoshi Shimokawa <simokawa@sat.t.u-tokyo.ac.jp>
- Hideyuki Suzuki <hideyuki@sat.t.u-tokyo.ac.jp>
- Hirayama Issei <iss@mail.wbs.ne.jp>
- Hiroaki Sakai <sakai@miya.ee.kagu.sut.ac.jp>
- Hiroharu Tamaru <tamaru@ap.t.u-tokyo.ac.jp>
- Hironori Ikura <hikura@kaisei.org>
- Hiroshi Nishikawa <nis@pluto.dti.ne.jp>
- Hiroya Tsubakimoto <unknown>
- Hiroyuki NAKAJI <nakaji@zeisei3.dpri.kyoto-u.ac.jp>
- Holger Veit <Holger.Veit@gmd.de>
- Holm Tiffe <holm@geophysik.tu-freiberg.de>
- Horance Chou <horance@freedom.ie.cycu.edu.tw>
- Horihiro Kumagaio <kuma@jp.freebsd.org>
- Horikawa Kazuo <k-horik@mail.yk.rim.or.jp>
- Hr.Ladavac <lada@ws2301.gud.siemens.co.at>
- Hubert Feyrer <hubertf@NetBSD.ORG>
- Hugh F. Mahon <hugh@nsmdserv.cnd.hp.com>

- Hugh Mahon <h\_mahon@fc.hp.com>
- Hung-Chi Chu <hcchu@r350.ee.ntu.edu.tw>
- IMAI Takeshi <take-i@ceres.dti.ne.jp>
- IMAMURA Tomoaki <tomoak-i@is.aist-nara.ac.jp>
- Ian Dowse <iedowse@maths.tcd.ie>
- Ian Holland <ianh@tortuga.com.au>
- Ian Struble <ian@broken.net>
- Ian Vaudrey <i.vaudrey@bigfoot.com>
- Igor Khasilev <igor@jabber.paco.odessa.ua>
- Igor Roshchin <str@giganda.komkon.org>
- Igor Sviridov <siac@ua.net>
- Igor Vinokurov <igor@zynaps.ru>
- Ikuo Nakagawa <ikuo@isl.intec.co.jp>
- Ilya V. Komarov <mur@lynx.ru>
- Issei Suzuki <issei@jp.FreeBSD.org>
- Itsuro Saito <saito@miv.t.u-tokyo.ac.jp>
- J. Bryant <jbryant@argus.flash.net>
- J. David Lowe <lowe@saturn5.com>
- J. Han <hjh@best.com>
- J. Hawk <jhawk@MIT.EDU>
- J.T. Conklin <jtc@cygnus.com>
- J.T. Jang <keith@email.gcn.net.tw>
- Jack <jack@zeus.xtalwind.net>
- Jacob Bohn Lorensen <jacob@jblhome.ping.mk>
- Jagane D Sundar <jagane@netcom.com>
- Jake Hamby <jehamby@lightside.com>
- James Clark <jjc@jclark.com>
- James D. Stewart <jds@c4system.com>
- James Jegers <jimj@miller.cs.uwm.edu>

- James Raynard <fhackers@jraynard.demon.co.uk>
- James T. Liu <jtliu@phlebas.rockefeller.edu>
- James da Silva <jds@cs.umd.edu>
- Jan Conard <charly@fachschaften.tu-muenchen.de>
- Jan Koum <jkb@FreeBSD.org>
- Janick Taillandier <Janick.Taillandier@ratp.fr>
- Janusz Kokot <janek@gaja.ipan.lublin.pl>
- Jarle Greipsland <jarle@idt.unit.no>
- Jason Garman <init@risen.org>
- Jason Thorpe <thorpej@NetBSD.org>
- Jason Wright <jason@OpenBSD.org>
- Jason Young <doogie@forbidden-donut.anet-stl.com>
- Javier Martin Rueda <jmrueda@diatel.upm.es>
- Jay Fenlason <hack@datacube.com>
- Jaye Mathisen <mrcpu@cdsnet.net>
- Jeff Bartig <jeffb@doit.wisc.edu>
- Jeff Forys <jeff@forys.cranbury.nj.us>
- Jeff Kletsky <Jeff@Wagsky.com>
- Jeffrey Evans <evans@scnc.k12.mi.us>
- Jeffrey Wheat <jeff@cetlink.net>
- Jens Schweikhardt <schweikh@ito.uni-stuttgart.de>
- Jeremy Allison <jallison@whistle.com>
- Jeremy Chatfield <jdc@xinside.com>
- Jeremy Lea <reg@shale.csir.co.za>
- Jeremy Prior <unknown>
- Jeroen Ruigrok/Asmodai <asmodai@wxs.nl>
- Jesse Rosenstock <jmr@ugcs.caltech.edu>
- Jian-Da Li <jdli@csie.nctu.edu.tw>
- Jim Babb <babb@FreeBSD.org>

- Jim Binkley <jrb@cs.pdx.edu>
- Jim Carroll <jim@carroll.com>
- Jim Flowers <jflowers@ezo.net>
- Jim Leppek <jleppek@harris.com>
- Jim Lowe <james@cs.uwm.edu>
- Jim Mattson <jmattson@sonic.net>
- Jim Mercer <jim@komodo.reptiles.org>
- Jim Mock <jim@phrantic.phear.net>
- Jim Wilson <wilson@moriamoria.cygnum.com>
- Jimbo Bahooli <griffin@blackhole.iceworld.org>
- Jin Guojun <jin@george.lbl.gov>
- Joachim Kuebart <unknown>
- Joao Carlos Mendes Luis <jonny@jonny.eng.br>
- Jochen Pohl <jpo.drs@sni.de>
- Joe "Marcus" Clarke <marcus@miami.edu>
- Joe Abley <jabley@clear.co.nz>
- Joe Jih-Shian Lu <jslu@dns.ntu.edu.tw>
- Joe Orthoefer <j\_orthoefer@tia.net>
- Joe Traister <traister@mojozone.org>
- Joel Faedi <Joel.Faedi@esial.u-nancy.fr>
- Joel Ray Holveck <joelh@gnu.org>
- Joel Sutton <sutton@aardvark.apana.org.au>
- Johan Granlund <johan@granlund.nu>
- Johan Karlsson <k@numeri.campus.luth.se>
- Johan Larsson <johan@moon.campus.luth.se>
- Johann Tonsing <jtonsing@mikom.csir.co.za>
- Johannes Helander <unknown>
- Johannes Stille <unknown>
- John Baldwin <jobaldwi@vt.edu>

- John Beckett <jbeckett@southern.edu>
- John Beukema <jbeukema@hk.super.net>
- John Brezak <unknown>
- John Capo <jc@irbs.com>
- John F. Woods <jfw@jfwhome.funhouse.com>
- John Goerzen <jgoerzen@alexanderwohl.complete.org>
- John Hay <jhay@mikom.csir.co.za>
- John Heidemann <johnh@isi.edu>
- John Hood <cgull@owl.org>
- John Kohl <unknown>
- John Lind <john@starfire.mn.org>
- John Mackin <john@physiol.su.oz.au>
- John P <johnp@lodgenet.com>
- John Perry <perry@vishnu.alias.net>
- John Preisler <john@vapornet.com>
- John Rochester <jr@cs.mun.ca>
- John Sadler <john\_sadler@alum.mit.edu>
- John Saunders <john@pacer.nlc.net.au>
- John W. DeBoskey <jwd@unx.sas.com>
- John Wehle <john@feith.com>
- John Woods <jfw@eddie.mit.edu>
- Jon Morgan <morgan@terminus.trailblazer.com>
- Jonathan H N Chin <jc254@newton.cam.ac.uk>
- Jonathan Hanna <jh@pc-21490.bc.rogers.wave.ca>
- Jorge Goncalves <j@bug.fe.up.pt>
- Jorge M. Goncalves <ee96199@tom.fe.up.pt>
- Jos Backus <jbackus@plex.nl>
- Jose M. Alcaide <jose@we.lc.ehu.es>
- Josef Grosch <jgrosch@superior.mooseriver.com>

- Josef Karthausser <joe@uk.freebsd.org>
- Joseph Stein <joes@seaport.net>
- Josh Gilliam <josh@quick.net>
- Josh Tiefenbach <josh@ican.net>
- Juergen Lock <nox@jelal.hb.north.de>
- Juha Inkari <inkari@cc.hut.fi>
- Jukka A. Ukkonen <jua@iki.fi>
- Julian Assange <proff@suburbia.net>
- Julian Coleman <j.d.coleman@ncl.ac.uk>
- Julian H. Stacey <jhs@freebsd.org>
- Julian Jenkins <kaveman@magna.com.au>
- Junichi Satoh <junichi@jp.freebsd.org>
- Junji SAKAI <sakai@jp.freebsd.org>
- Junya WATANABE <junya-w@remus.dti.ne.jp>
- K.Higashino <a00303@cc.hc.keio.ac.jp>
- KUNISHIMA Takeo <kunishi@c.oka-pu.ac.jp>
- Kai Vorma <vode@snakemail.hut.fi>
- Kaleb S. Keithley <kaleb@ics.com>
- Kaneda Hiloshi <vanitas@ma3.seikyuu.ne.jp>
- Kapil Chowksey <kchowksey@hss.hns.com>
- Karl Denninger <karl@mcs.com>
- Karl Dietz <Karl.Dietz@triplan.com>
- Karl Lehenbauer <karl@NeoSoft.com>
- Kato Takenori <kato@eclogite.eps.nagoya-u.ac.jp>
- Kauzo Horikawa <h-horik@yk.rim.or.jp>
- Kawanobe Koh <kawanobe@st.rim.or.jp>
- Kazuhiko Kiriyaama <kiri@kiri.toba-cmt.ac.jp>
- Kazuo Horikawa <horikawa@jp.FreeBSD.org>
- Kees Jan Koster <kjkl@ukc.ac.uk>

- Keith Bostic <bostic@bostic.com>
- Keith E. Walker <unknown>
- Keith Moore <unknown>
- Keith Sklower <unknown>
- Ken Hornstein <unknown>
- Ken Key <key@cs.utk.edu>
- Ken Mayer <kmayer@freegate.com>
- Kenji Saito <marukun@mx2.nisiq.net>
- Kenji Tomita <tommyk@da2.so-net.or.jp>
- Kenneth Furge <kenneth.furge@us.endress.com>
- Kenneth Monville <desmo@bandwidth.org>
- Kenneth R. Westerback <krw@tcn.net>
- Kenneth Stailey <kstailey@gnu.ai.mit.edu>
- Kent Talarico <kent@shipwreck.tsoft.net>
- Kent Vander Velden <graphix@iastate.edu>
- Kentaro Inagaki <JBD01226@niftyserve.ne.jp>
- Kevin Bracey <kbracey@art.acorn.co.uk>
- Kevin Day <toasty@dragondata.com>
- Kevin Lahey <kml@nas.nasa.gov>
- Kevin Street <street@iname.com>
- Kevin Van Maren <vanmaren@fast.cs.utah.edu>
- Kiroh HARADA <kiroh@kh.rim.or.jp>
- Klaus Klein <kleink@layla.inka.de>
- Klaus-J. Wolf <Yanestra@t-online.de>
- Koichi Sato <copan@ppp.fastnet.or.jp>
- Kostya Lukin <lukin@okbmei.msk.su>
- Kouichi Hirabayashi <kh@mogami-wire.co.jp>
- Kurt D. Zeilenga <Kurt@Boolean.NET>
- Kurt Olsen <kurto@tiny.mcs.usu.edu>

- L. Jonas Olsson <ljo@ljo-slip.DIALIN.CWRU.Edu>
- Lars Köller <Lars.Koeller@Uni-Bielefeld.DE>
- Larry Altneu <larry@ALR.COM>
- Laurence Lopez <lopez@mv.mv.com>
- Lee Cremeans <lcremean@tidalwave.net>
- Liang Tai-hwa <avatar@www.mmlab.cse.yzu.edu.tw>
- Lon Willett <lon%softt.uucp@math.utah.edu>
- Louis A. Mamakos <louie@TransSys.COM>
- Louis Mamakos <loieue@TransSys.com>
- Lucas James <Lucas.James@ldjpc.apana.org.au>
- Lyndon Nerenberg <lyndon@orthanc.com>
- M.C. Wong <unknown>
- MANTANI Nobutaka <nobutaka@nobutaka.com>
- MIHIRA Sanpei Yoshio <sanpei@sanpei.org>
- MITA Yoshio <mita@jp.FreeBSD.ORG>
- MITSUNAGA Noriaki <mitchy@er.ams.eng.osaka-u.ac.jp>
- MOROHOSHI Akihiko <moro@race.u-tokyo.ac.jp>
- Magnus Enbom <dot@tinto.campus.luth.se>
- Mahesh Neelakanta <mahesh@gcomm.com>
- Makoto MATSUSHITA <matusita@jp.freebsd.org>
- Makoto WATANABE <watanabe@zlab.phys.nagoya-u.ac.jp>
- Malte Lance <malte.lance@gmx.net>
- Manu Iyengar <iyengar@grunthos.pscwa.pscs.com>
- Marc Frajola <marc@dev.com>
- Marc Ramirez <mrami@mramirez.sy.yale.edu>
- Marc Slemko <marcs@znep.com>
- Marc van Kempen <wmbfmk@urc.tue.nl>
- Marcel Moolenaar <marcel@scc.nl>
- Mario Sergio Fujikawa Ferreira <lioux@gns.com.br>

- Mark Andrews <unknown>
- Mark Cammidge <mark@gmtunx.ee.uct.ac.za>
- Mark Diekhans <markd@grizzly.com>
- Mark Huizer <xaa@stack.nl>
- Mark J. Taylor <mtaylor@cybernet.com>
- Mark Krentel <krentel@rice.edu>
- Mark Mayo <markm@vmunix.com>
- Mark Thompson <thompson@tgsoft.com>
- Mark Tinguely <tinguely@plains.nodak.edu>
- Mark Treacy <unknown>
- Mark Valentine <mark@linus.demon.co.uk>
- Martin Birgmeier
- Martin Ibert <mib@ppe.bb-data.de>
- Martin Kammerhofer <dada@sbox.tu-graz.ac.at>
- Martin Renters <martin@tdc.on.ca>
- Martti Kuparinen <erakupa@kk.etx.ericsson.se>
- Masachika ISHIZUKA <ishizuka@isis.min.ntt.jp>
- Mas.TAKEMURA <unknown>
- Masafumi NAKANE <max@wide.ad.jp>
- Masahiro Sekiguchi <seki@sysrap.cs.fujitsu.co.jp>
- Masanobu Saitoh <msaitoh@spa.is.uec.ac.jp>
- Masanori Kanaoka <kana@saijo.mke.mei.co.jp>
- Masanori Kiriake <seiken@ncs.co.jp>
- Masatoshi TAMURA <tamrin@shinzan.kuee.kyoto-u.ac.jp>
- Mats Lofkvist <mal@algonet.se>
- Matt Bartley <mbartley@lear35.cytex.com>
- Matt Thomas <matt@3am-software.com>
- Matt White <mwhite+@CMU.EDU>
- Matthew C. Mead <mmead@Glock.COM>

- Matthew Cashdollar <mattc@rfcnet.com>
- Matthew Flatt <mflatt@cs.rice.edu>
- Matthew Fuller <fullermd@futuresouth.com>
- Matthew N. Dodd <winter@jurai.net>
- Matthew Stein <matt@bdd.net>
- Matthias Pfaller <leo@dachau.marco.de>
- Matthias Scheler <tron@netbsd.org>
- Mattias Gronlund <Mattias.Gronlund@sa.erisoft.se>
- Mattias Pantzare <pantzer@ludd.luth.se>
- Maurice Castro <maurice@planet.serc.rmit.edu.au>
- Max Euston <meuston@jmrodgers.com>
- Max Khon <fjoe@husky.iclub.nsu.ru>
- Maxim Bolotin <max@rsu.ru>
- Micha Class <michael\_class@hpbbse.bbn.hp.com>
- Michael Butler <imb@scgt.oz.au>
- Michael Butschky <butsch@computi.erols.com>
- Michael Clay <mclay@weareb.org>
- Michael Elbel <me@FreeBSD.ORG>
- Michael Galassi <nerd@percival.rain.com>
- Michael Hancock <michaelh@cet.co.jp>
- Michael Hohmuth <hohmuth@inf.tu-dresden.de>
- Michael Perlman <canuck@caam.rice.edu>
- Michael Petry <petry@netwolf.NetMasters.com>
- Michael Reifenberger <root@totum.plaut.de>
- Michael Searle <searle@longacre.demon.co.uk>
- Michal Listos <mcl@Amnesiac.123.org>
- Michio Karl Jinbo <karl@marcer.nagaokaut.ac.jp>
- Miguel Angel Sagreras <msagre@cactus.fi.uba.ar>
- Mihoko Tanaka <m\_tonaka@pa.yokogawa.co.jp>

- Mika Nystrom <mika@cs.caltech.edu>
- Mikael Hybsch <micke@dynas.se>
- Mikael Karpberg <karpen@ocean.campus.luth.se>
- Mike Del <repenting@hotmail.com>
- Mike Durian <durian@plutotech.com>
- Mike Durkin <mdurkin@tsoft.sf-bay.org>
- Mike E. Matsnev <mike@azog.cs.msu.su>
- Mike Evans <mevans@candle.com>
- Mike Grupenhoff <kashmir@umiacs.umd.edu>
- Mike Hibler <mike@marker.cs.utah.edu>
- Mike Karels <unknown>
- Mike McGaughey <mmcg@cs.monash.edu.au>
- Mike Meyer <mwm@shiva.the-park.com>
- Mike Mitchell <mitchell@ref.tfs.com>
- Mike Murphy <mrm@alpharel.com>
- Mike Peck <mike@binghamton.edu>
- Mike Spengler <mks@msc.edu>
- Mikhail A. Sokolov <mishania@demos.su>
- Mikhail Teterin <mi@aldan.ziplink.net>
- Ming-I Hseh <PA@FreeBSD.ee.Ntu.edu.TW>
- Mitsuru IWASAKI <iwasaki@pc.jaring.my>
- Monte Mitzelfelt <monte@gonefishing.org>
- Morgan Davis <root@io.cts.com>
- Mostyn Lewis <mostyn@mrl.com>
- Motoyuki Kasahara <m-kasahr@sra.co.jp>
- Motoyuki Konno <motoyuki@snipe.rim.or.jp>
- Munechika Sumikawa <sumikawa@kame.net>
- Murray Stokely <murray@cdrom.com>
- N.G.Smith <ngs@sesame.hensa.ac.uk>

- NAGAO Tadaaki <nagao@cs.titech.ac.jp>
- NAKAJI Hiroyuki <nakaji@zeisei.dpri.kyoto-u.ac.jp>
- NAKAMURA Kazushi <nkazushi@highway.or.jp>
- NAKAMURA Motonori <motonori@econ.kyoto-u.ac.jp>
- NIIMI Satoshi <sa2c@and.or.jp>
- NOKUBI Hirotaka <h-nokubi@yyy.or.jp>
- Nadav Eiron <nadav@barcode.co.il>
- Nanbor Wang <nw1@cs.wustl.edu>
- Naofumi Honda <honda@Kururu.math.sci.hokudai.ac.jp>
- Naoki Hamada <nao@tom-yam.or.jp>
- Narvi <narvi@haldjas.folklore.ee>
- Nathan Dorfman <nathan@rtfm.net>
- Neal Fachan <kneel@ishiboo.com>
- Neil Blakey-Milner <nbm@rucus.ru.ac.za>
- Niall Smart <rotel@indigo.ie>
- Nick Barnes <Nick.Barnes@pobox.com>
- Nick Handel <nhandel@NeoSoft.com>
- Nick Hilliard <nick@foobar.org>
- Nick Sayer <nsayer@quack.kfu.com>
- Nick Williams <njw@cs.city.ac.uk>
- Nickolay N. Dudorov <nnd@itfs.nsk.su>
- Niklas Hallqvist <niklas@filippa.appli.se>
- Nisha Talagala <nisha@cs.berkeley.edu>
- No Name <ZW6T-KND@j.asahi-net.or.jp>
- No Name <adrian@virginia.edu>
- No Name <alex@elvisti.kiev.ua>
- No Name <anto@netscape.net>
- No Name <bobson@egg.ics.nitech.ac.jp>
- No Name <bovynf@awe.be>

- No Name <burg@is.ge.com>
- No Name <chris@gnome.co.uk>
- No Name <colsen@usa.net>
- No Name <coredump@nervosa.com>
- No Name <dannyman@arh0300.urh.uiuc.edu>
- No Name <davids@SECFNET.COM>
- No Name <derek@free.org>
- No Name <devet@adv.IAEhv.nl>
- No Name <djv@bedford.net>
- No Name <dvv@sprint.net>
- No Name <enami@ba2.so-net.or.jp>
- No Name <flash@eru.tubank.msk.su>
- No Name <flash@hway.ru>
- No Name <fn@pain.csr.vt.edu>
- No Name <gclarkii@netport.neosoft.com>
- No Name <gordon@sheaky.lonestar.org>
- No Name <graaf@iae.nl>
- No Name <greg@greg.rim.or.jp>
- No Name <grossman@cygnus.com>
- No Name <gusw@fub46.zedat.fu-berlin.de>
- No Name <hfir@math.rochester.edu>
- No Name <hnokubi@yyy.or.jp>
- No Name <iaint@css.tuu.utas.edu.au>
- No Name <invis@visi.com>
- No Name <ishisone@sra.co.jp>
- No Name <iverson@lionheart.com>
- No Name <jpt@magic.net>
- No Name <junker@jazz.snu.ac.kr>
- No Name <k-sugyou@ccs.mt.nec.co.jp>

- No Name <kenji@reseau.toyonaka.osaka.jp>
- No Name <kfurge@worldnet.att.net>
- No Name <lh@aus.org>
- No Name <lhecking@nmrc.ucc.ie>
- No Name <mrgreen@mame.mu.oz.au>
- No Name <nakagawa@jp.freebsd.org>
- No Name <ohki@gssm.otsuka.tsukuba.ac.jp>
- No Name <owaki@st.rim.or.jp>
- No Name <pechter@shell.monmouth.com>
- No Name <pete@pelican.pelican.com>
- No Name <pritic003@maroon.tc.umn.edu>
- No Name <risner@studio.com>
- No Name <roman@rpd.univ.kiev.ua>
- No Name <root@ns2.redline.ru>
- No Name <root@uglabgw.ug.cs.sunysb.edu>
- No Name <stephen.ma@jtec.com.au>
- No Name <sumii@is.s.u-tokyo.ac.jp>
- No Name <takas-su@is.aist-nara.ac.jp>
- No Name <tamone@eig.unige.ch>
- No Name <tjevans@raleigh.ibm.com>
- No Name <tony-o@iiij.ad.jp amurai@spec.co.jp>
- No Name <torii@tcd.hitachi.co.jp>
- No Name <uenami@imasy.or.jp>
- No Name <uhlar@netlab.sk>
- No Name <vode@hut.fi>
- No Name <wlloyd@mpd.ca>
- No Name <wlr@furball.wellsfargo.com>
- No Name <wmbfmk@urc.tue.nl>
- No Name <yamagata@nwgpc.kek.jp>

- No Name <ziggy@ryan.org>
- Nobuhiro Yasutomi <nobu@psrc.isac.co.jp>
- Nobuyuki Koganemaru <kogane@koganemaru.co.jp>
- Norio Suzuki <nosuzuki@e-mail.ne.jp>
- Noritaka Ishizumi <graphite@jp.FreeBSD.ORG>
- Noriyuki Soda <soda@sra.co.jp>
- Olaf Wagner <wagner@luthien.in-berlin.de>
- Oleg Sharoiko <os@rsu.ru>
- Oliver Breuninger <ob@seicom.NET>
- Oliver Friedrichs <oliver@secnet.com>
- Oliver Fromme <oliver.fromme@heim3.tu-clausthal.de>
- Oliver Laumann <net@informatik.uni-bremen.de>
- Oliver Oberdorf <oly@world.std.com>
- Olof Johansson <offe@ludd.luth.se>
- Osokin Sergey aka oZZ <ozz@freebsd.org.ru>
- Pace Willisson <pace@blitz.com>
- Paco Rosich <rosich@modico.eleinf.uv.es>
- Palle Girgensohn <girgen@partitur.se>
- Parag Patel <parag@cgt.com>
- Pascal Pederiva <pascal@zuo.dec.com>
- Pasvorn Boonmark <boonmark@juniper.net>
- Patrick Gardella <patrick@creativegroup.com>
- Patrick Hausen <unknown>
- Paul Antonov <apg@demos.su>
- Paul F. Werkowski <unknown>
- Paul Fox <pgf@foxharp.boston.ma.us>
- Paul Koch <koch@thehub.com.au>
- Paul Kranenburg <pk@NetBSD.org>
- Paul Mackerras <paulus@cs.anu.edu.au>

- Paul Popelka <paulp@uts.amdahl.com>
- Paul S. LaFollette, Jr. <unknown>
- Paul Saab <paul@mu.org>
- Paul Sandys <myj@nyct.net>
- Paul T. Root <proot@horton.iaces.com>
- Paul Vixie <paul@vix.com>
- Paulo Menezes <paulo@isr.uc.pt>
- Paulo Menezes <pm@dee.uc.pt>
- Pedro A M Vazquez <vazquez@IQM.Unicamp.BR>
- Pedro Giffuni <giffunip@asme.org>
- Pete Bentley <pete@demon.net>
- Peter Childs <pjchilds@imforei.apana.org.au>
- Peter Cornelius <pc@inr.fzk.de>
- Peter Haight <peterh@prognnet.com>
- Peter Jeremy <perer.jeremy@alcatel.com.au>
- Peter M. Chen <pmchen@eecs.umich.edu>
- Peter Much <peter@citylink.dinoex.sub.org>
- Peter Olsson <unknown>
- Peter Philipp <pjp@bsd-daemon.net>
- Peter Stubbs <PETERS@staidan.qld.edu.au>
- Phil Maker <pjm@cs.ntu.edu.au>
- Phil Sutherland <philsuth@mycroft.dialix.oz.au>
- Phil Taylor <phil@zipmail.co.uk>
- Philip Musumeci <philip@mit.edu.au>
- Pierre Y. Dampure <pierre.dampure@k2c.co.uk>
- Pius Fischer <pius@ienet.com>
- Pomegranate <daver@flag.blackened.net>
- Powerdog Industries <kevin.ruddy@powerdog.com>
- R. Kym Horsell

- Rajesh Vaidheeswaran <rv@fore.com>
- Ralf Friedl <friedl@informatik.uni-kl.de>
- Randal S. Masutani <randal@comtest.com>
- Randall Hopper <rh@ct.picker.com>
- Randall W. Dean <rwd@osf.org>
- Randy Bush <rbush@bainbridge.verio.net>
- Reinier Bezuidenhout <rbezuide@mikom.csir.co.za>
- Remy Card <Remy.Card@masi.ibp.fr>
- Ricardas Cepas <rch@richard.eu.org>
- Richard Henderson <richard@atheist.tamu.edu>
- Richard Hwang <rhwang@bigpanda.com>
- Richard J Kuhns <rjk@watson.graue1.com>
- Richard M. Neswold <rneswold@drmemory.fnal.gov>
- Richard Seaman, Jr. <dick@tar.com>
- Richard Stallman <rms@gnu.ai.mit.edu>
- Richard Straka <straka@user1.inficad.com>
- Richard Tobin <richard@cogsci.ed.ac.uk>
- Richard Wackerbarth <rkw@Dataplex.NET>
- Richard Winkel <rich@math.missouri.edu>
- Richard Wiwatowski <rjwiwat@adelaide.on.net>
- Rick Macklem <rick@snowwhite.cis.uoguelph.ca>
- Rick Macklin <unknown>
- Rob Austein <sra@epilogue.com>
- Rob Mallory <rmallory@qualcomm.com>
- Rob Snow <rsnow@txdirect.net>
- Robert Crowe <bob@speakez.com>
- Robert D. Thrush <rd@phoenix.aii.com>
- Robert Eckardt <roberte@MEP.Ruhr-Uni-Bochum.de>
- Robert Sanders <rsanders@mindspring.com>

- Robert Sexton <robert@kudra.com>
- Robert Shady <rls@id.net>
- Robert Swindells <swindellsr@genrad.co.uk>
- Robert Watson <robert@cyrus.watson.org>
- Robert Withrow <witr@rwwa.com>
- Robert Yoder <unknown>
- Robin Carey <robin@mailgate.dtc.rankxerox.co.uk>
- Roger Hardiman <roger@cs.strath.ac.uk>
- Roland Jesse <jesse@cs.uni-magdeburg.de>
- Ron Bickers <rbickers@intercenter.net>
- Ron Lenk <rlenk@widget.xmission.com>
- Ronald Kuehn <kuehn@rz.tu-clausthal.de>
- Rudolf Cejka <unknown>
- Ruslan Belkin <rus@home2.UA.net>
- Ruslan Ermilov <ru@ucb.crimea.ua>
- Ruslan Shevchenko <rssh@cam.grad.kiev.ua>
- Russell L. Carter <rcarter@pinyon.org>
- Russell Vincent <rv@groa.uct.ac.za>
- Ryan Younce <ryany@pobox.com>
- SANETO Takanori <sanewo@strg.sony.co.jp>
- SAWADA Mizuki <miz@qb3.so-net.ne.jp>
- SUGIMURA Takashi <sugimura@jp.FreeBSD.ORG>
- SURANYI Peter <suranyip@jks.is.tsukuba.ac.jp>
- Sakari Jalovaara <sja@tekla.fi>
- Sam Hartman <hartmans@mit.edu>
- Samuel Lam <skl@ScalableNetwork.com>
- Sander Vesik <sander@haldjas.folklore.ee>
- Sandro Sigala <ssigala@globalnet.it>
- Sascha Blank <blank@fox.uni-trier.de>

- Sascha Wildner <swildner@channelz.GUN.de>
- Satoh Junichi <junichi@astec.co.jp>
- Satoshi Taoka <taoka@infonets.hiroshima-u.ac.jp>
- Scot Elliott <scot@poptart.org>
- Scot W. Hetzel <hetzels@westbend.net>
- Scott A. Kenney <saken@rmta.ml.org>
- Scott Blachowicz <scott.blachowicz@seaslug.org>
- Scott Burris <scott@pita.cns.ucla.edu>
- Scott Hazen Mueller <scott@zorch.sf-bay.org>
- Scott Michel <scottm@cs.ucla.edu>
- Scott Reynolds <scott@clmgt.marquette.mi.us>
- Sebastian Strollo <seb@erix.ericsson.se>
- Seigou TANIMURA <tanimura@naklab.dnj.ynu.ac.jp>
- Serge A. Babkin <babkin@hq.icb.chel.su>
- Serge V. Vakulenko <vak@zebub.msk.su>
- Sergei Chechetkin <csl@whale.sunbay.crimea.ua>
- Sergei S. Laskavy <laskavy@pc759.cs.msu.su>
- Sergey Gershtein <sg@mplik.ru>
- Sergey Potapov <sp@alkor.ru>
- Sergey Shkonda <serg@bcs.zp.ua>
- Sergey V.Dorokhov <svd@kbtelecom.nalnet.ru>
- Sergio Lenzi <lenzi@bsi.com.br>
- Shaun Courtney <shaun@emma.eng.uct.ac.za>
- Shawn M. Carey <smcarey@mailbox.syr.edu>
- Sheldon Hearn <axl@iafrica.com>
- Shigio Yamaguchi <shigio@wafu.netgate.net>
- Shunsuke Akiyama <akiyama@jp.freebsd.org>
- Simon <simon@masi.ibp.fr>
- Simon Burge <simonb@telstra.com.au>

- Simon J Gerraty <sjg@melb.bull.oz.au>
- Simon Marlow <simonm@dcs.gla.ac.uk>
- Simon Shapiro <shimon@simon-shapiro.org>
- Sin'ichiro MIYATANI <siu@phaseone.co.jp>
- Slaven Rezic <eserte@cs.tu-berlin.de>
- Soochon Radee <slr@mitre.org>
- Soren Dayton <csgiving@midway.uchicago.edu>
- Soren Dossing <sauber@netcom.com>
- Soren S. Jorvang <soren@dt.dk>
- Stefan Bethke <stb@hanse.de>
- Stefan Eggers <seggers@semyam.dinoco.de>
- Stefan Moeding <s.moeding@ndh.net>
- Stefan Petri <unknown>
- Stefan 'Sec' Zehl <sec@42.org>
- Steinar Haug <sthaug@nethelp.no>
- Stephane E. Potvin <sepotvin@videotron.ca>
- Stephane Legrand <stephane@lituus.fr>
- Stephen Clawson <sclawson@marker.cs.utah.edu>
- Stephen F. Combs <combsf@saalem.ge.com>
- Stephen Farrell <stephen@farrell.org>
- Stephen Hocking <sysseh@devetir.qld.gov.au>
- Stephen J. Roznowski <sjr@home.net>
- Stephen McKay <syssgm@devetir.qld.gov.au>
- Stephen Melvin <melvin@zytek.com>
- Steve Bauer <sbauer@rock.sdsmt.edu>
- Steve Deering <unknown>
- Steve Gerakines <steve2@genesis.tiac.net>
- Steve Gericke <steveg@control.com>
- Steve Piette <steve@simon.chi.il.US>

- Steve Schwarz <schwarz@alpharel.com>
- Steven G. Kargl <kargl@troutmask.apl.washington.edu>
- Steven H. Samorodin <samorodi@NUXI.com>
- Steven McCanne <mccanne@cs.berkeley.edu>
- Steven Plite <splite@purdue.edu>
- Steven Wallace <unknown>
- Stuart Henderson <stuart@internationalschool.co.uk>
- Sue Blake <sue@welearn.com.au>
- Sugiura Shiro <ssugiura@duo.co.jp>
- Sujal Patel <smpatel@wam.umd.edu>
- Sune Stjerneby <stjerneby@usa.net>
- Suzuki Yoshiaki <zensyo@ann.tama.kawasaki.jp>
- Tadashi Kumano <kumano@str1.nhk.or.jp>
- Taguchi Takeshi <taguchi@tohoku.iij.ad.jp>
- Takahashi Yoshihiro <nyan@dd.catv.ne.jp>
- Takahiro Yugawa <yugawa@orleans.rim.or.jp>
- Takanori Watanabe <takawata@shidahara1.planet.sci.kobe-u.ac.jp>
- Takashi Mega <mega@minz.org>
- Takashi Uozu <j1594016@ed.kagu.sut.ac.jp>
- Takayuki Ariga <a00821@cc.hc.keio.ac.jp>
- Takeru NAIKI <naiki@bfd.es.hokudai.ac.jp>
- Takeshi Amaike <amaike@iri.co.jp>
- Takeshi MUTOH <mutoh@info.nara-k.ac.jp>
- Takeshi Ohashi <ohashi@mickey.ai.kyutech.ac.jp>
- Takeshi WATANABE <watanabe@crayon.earth.s.kobe-u.ac.jp>
- Takuya SHIOZAKI <tshiozak@makino.ise.chuo-u.ac.jp>
- Tatoku Ogaito <tacha@tera.fukui-med.ac.jp>
- Tatsumi HOSOKAWA <hosokawa@jp.FreeBSD.org>
- Ted Buswell <tbuswell@mediaone.net>

- Ted Faber <faber@isi.edu>
- Ted Lemon <unknown>
- Terry Lambert <terry@lambert.org>
- Terry Lee <terry@uivlsi.csl.uiuc.edu>
- Tetsuya Furukawa <tetsuya@secom-sis.co.jp>
- Theo de Raadt <deraadt@OpenBSD.org>
- Thomas <thomas@mathematik.uni-Bremen.de>
- Thomas D. Dean <tomdean@ix.netcom.com>
- Thomas David Rivers <rivers@dignus.com>
- Thomas G. McWilliams <tgm@netcom.com>
- Thomas Gellekum <thomas@ghpc8.ihf.rwth-aachen.de>
- Thomas Graichen <graichen@omega.physik.fu-berlin.de>
- Thomas König <Thomas.Koenig@ciw.uni-karlsruhe.de>
- Thomas Ptacek <unknown>
- Thomas Stromberg <tstrombe@rtci.com>
- Thomas Valentino Crimi <tcrimi+@andrew.cmu.edu>
- Thomas Wintergerst <thomas@lemur.nord.de>
- Þórður Ívarsson <totii@est.is>
- Tim Kientzle <kientzle@netcom.com>
- Tim Singletary <tsingle@sunland.gsfc.nasa.gov>
- Tim Wilkinson <tim@sarc.city.ac.uk>
- Timo J. Rinne <tri@iki.fi>
- Todd Miller <millert@openbsd.org>
- Tom <root@majestix.cmr.no>
- Tom <tom@sdf.com>
- Tom Gray - DCA <dcasba@rain.org>
- Tom Hukins <tom@eborcom.com>
- Tom Jobbins <tom@tom.tj>
- Tom Pusateri <pusateri@juniper.net>

- Tom Rush <tarush@mindspring.com>
- Tom Samplonius <tom@misery.sdf.com>
- Tomohiko Kurahashi <kura@melchior.q.t.u-tokyo.ac.jp>
- Tony Kimball <alk@Think.COM>
- Tony Li <tli@jnx.com>
- Tony Lynn <wing@cc.nsysu.edu.tw>
- Torbjorn Granlund <tege@matematik.su.se>
- Toshihiko ARAI <toshi@tenchi.ne.jp>
- Toshihiko SHIMOKAWA <toshi@tea.forus.or.jp>
- Toshihiro Kanda <candy@kgc.co.jp>
- Toshiomi Moriki <Toshiomi.Moriki@mal.seikyuu.ne.jp>
- Trefor S. <trefor@flevel.co.uk>
- Trevor Blackwell <tlb@viaweb.com>
- URATA Shuichiro <s-urata@nmit.tmg.nec.co.jp>
- Ugo Paternostro <paterno@dsi.unifi.it>
- Ulf Kieber <kieber@sax.de>
- Ulli Linzen <ulli@perceval.camelot.de>
- Ustimenko Semen <semen@iclub.nsu.ru>
- Uwe Arndt <arndt@mailhost.uni-koblenz.de>
- Vadim Chekan <vadim@gc.lviv.ua>
- Vadim Kolontsov <vadim@tversu.ac.ru>
- Vadim Mikhailov <mvp@braz.ru>
- Van Jacobson <van@ee.lbl.gov>
- Vasily V. Grechishnikov <bazilio@ns1.ied-vorstu.ac.ru>
- Vasim Valejev <vasim@uddias.diaspro.com>
- Vernon J. Schryver <vjs@mica.denver.sgi.com>
- Vic Abell <abe@cc.purdue.edu>
- Ville Eerola <ve@sci.fi>
- Vincent Poy <vince@venus.gaiainet.net>

- Vincenzo Capuano <VCAPUANO@vmprofs.esoc.esa.de>
- Virgil Champlin <champlin@pa.dec.com>
- Vladimir A. Jakovenko <vovik@ntu-kpi.kiev.ua>
- Vladimir Kushnir <kushn@mail.kar.net>
- Vsevolod Lobko <seva@alex-ua.com>
- W. Gerald Hicks <>wghicks@bellsouth.net>
- W. Richard Stevens <rstevens@noao.edu>
- Walt Howard <howard@ee.utah.edu>
- Warren Toomey <wkt@csadfa.cs.adfa.oz.au>
- Wayne Scott <wscott@ichips.intel.com>
- Werner Griessl <werner@btplda.phy.uni-bayreuth.de>
- Wes Santee <wsantee@wsantee.oz.net>
- Wietse Venema <wietse@wzv.win.tue.nl>
- Wilfredo Sanchez <wsanchez@apple.com>
- Wiljo Heinen <wiljo@freeside.ki.open.de>
- Wilko Bulte <wilko@yedi.iaf.nl>
- Willem Jan Withagen <wjw@surf.IAE.nl>
- William Jolitz <withheld>
- William Liao <william@tale.net>
- Wojtek Pilorz <wpilorz@celebris.bdk.lublin.pl>
- Wolfgang Helbig <helbig@ba-stuttgart.de>
- Wolfgang Solfrank <ws@tools.de>
- Wolfgang Stanglmeier <wolf@FreeBSD.org>
- Wu Ching-hong <woju@FreeBSD.ee.Ntu.edu.TW>
- Yarema <yds@ingress.com>
- Yaroslav Terletsy <ts@polynet.lviv.ua>
- Yen-Shuo Su <yssu@CCCA.NCTU.edu.tw>
- Ying-Chieh Liao <ijliao@csie.NCTU.edu.tw>
- Yixin Jin <yjin@rain.cs.ucla.edu>

- Yoshiaki Uchikawa <yoshiaki@kt.rim.or.jp>
- Yoshihiko OHTA <yohta@bres.tsukuba.ac.jp>
- Yoshihisa NAKAGAWA <y-nakaga@ccs.mt.nec.co.jp>
- Yoshikazu Goto <gotoh@ae.anritsu.co.jp>
- Yoshimasa Ohnishi <ohnishi@isc.kyutech.ac.jp>
- Yoshishige Arai <ryo2@on.rim.or.jp>
- Yuichi MATSUTAKA <matutaka@osa.att.ne.jp>
- Yujiro MIYATA <miyata@bioele.nuee.nagoya-u.ac.jp>
- Yukihiro Nakai <nacai@iname.com>
- Yusuke Nawano <azuki@azkey.org>
- Yuval Yarom <yval@cs.huji.ac.il>
- Yves Fonk <yves@cpcoup5.tn.tudelft.nl>
- Yves Fonk <yves@dutncp8.tn.tudelft.nl>
- Zach Heilig <zach@gaffaneys.com>
- Zahemshky Gabor <zgabor@code.hu>
- Zhong Ming-Xun <zmx@mail.CDPA.nsysu.edu.tw>
- arci <vega@sophia.inria.fr>
- der Mouse <mouse@Collatz.McRCIM.McGill.EDU>
- frf <frf@xocolatl.com>
- Ege Rekk <aagero@aage.priv.no>

## 386BSD Patch Kit Patch Contributors

(in alphabetical order by first name):

- Adam Glass <glass@postgres.berkeley.edu>
- Adrian Hall <adrian@ibmpcug.co.uk>
- Andrey A. Chernov <ache@astral.msk.su>
- Andrew Herbert <andrew@werple.apana.org.au>
- Andrew Moore <alm@netcom.com>

- Andy Valencia <ajv@csd.mot.com> <jtk@netcom.com>
- Arne Henrik Juul <arnej@Lise.Unit.NO>
- Bakul Shah <bvs@bitblocks.com>
- Barry Lustig <barry@ictv.com>
- Bob Wilcox <bob@obiwan.uucp>
- Branko Lankester
- Brett Lymn <blymn@mulga.awadi.com.AU>
- Charles Hannum <mycroft@ai.mit.edu>
- Chris G. Demetriou <cgd@postgres.berkeley.edu>
- Chris Torek <torek@ee.lbl.gov>
- Christoph Robitschko <chmr@edvz.tu-graz.ac.at>
- Daniel Poirot <poirot@aio.jsc.nasa.gov>
- Dave Burgess <burgess@hrd769.brooks.af.mil>
- Dave Rivers <rivers@ponds.uucp>
- David Dawes <dawes@physics.su.OZ.AU>
- David Greenman <dg@Root.COM>
- Eric J. Haug <ejh@slustl.slu.edu>
- Felix Gaetgens <felix@escape.vsse.in-berlin.de>
- Frank Maclachlan <fpm@crash.cts.com>
- Gary A. Browning <gab10@griffcd.amdahl.com>
- Gary Howland <gary@hotlava.com>
- Geoff Rehmet <csgr@alpha.ru.ac.za>
- Goran Hammarback <goran@astro.uu.se>
- Guido van Rooij <guido@gvr.org>
- Guy Harris <guy@auspex.com>
- Havard Eidnes <Havard.Eidnes@runit.sintef.no>
- Herb Peyerl <hpeyerl@novatel.cuc.ab.ca>
- Holger Veit <Holger.Veit@gmd.de>
- Ishii Masahiro, R. Kym Horsell

- J.T. Conklin <jtc@cygnus.com>
- Jagane D Sundar <jagane@netcom.com>
- James Clark <jjc@jclark.com>
- James Jegers <jimj@miller.cs.uwm.edu>
- James W. Dolter
- James da Silva <jds@cs.umd.edu> et al
- Jay Fenlason <hack@datacube.com>
- Jim Wilson <wilson@moria.cygnus.com>
- Jörg Lohse <lohse@tech7.informatik.uni-hamburg.de>
- Jörg Wunsch <joerg\_wunsch@uriah.heep.sax.de>
- John Dyson
- John Woods <jfw@eddie.mit.edu>
- Jordan K. Hubbard <jkh@whisker.hubbard.ie>
- Julian Elischer <julian@dialix.oz.au>
- Julian Stacey <jhs@freebsd.org>
- Karl Dietz <Karl.Dietz@triplan.com>
- Karl Lehenbauer <karl@NeoSoft.com> <karl@one.neosoft.com>
- Keith Bostic <bostic@toe.CS.Berkeley.EDU>
- Ken Hughes
- Kent Talarico <kent@shipwreck.tsoft.net>
- Kevin Lahey <kml%rokkaku.UUCP@mathcs.emory.edu> <kml@mosquito.cis.ufl.edu>
- Marc Frajola <marc@dev.com>
- Mark Tinguely <tinguely@plains.nodak.edu> <tinguely@hookie.cs.ndsu.NoDak.edu>
- Martin Renters <martin@tdc.on.ca>
- Michael Clay <mclay@weareb.org>
- Michael Galassi <nerd@percival.rain.com>
- Mike Durkin <mdurkin@tsoft.sf-bay.org>
- Naoki Hamada <nao@tom-yam.or.jp>
- Nate Williams <nate@bsd.coe.montana.edu>

- Nick Handel <nhandel@NeoSoft.com> <nick@madhouse.neosoft.com>
- Pace Willisson <pace@blitz.com>
- Paul Kranenburg <pk@cs.few.eur.nl>
- Paul Mackerras <paulus@cs.anu.edu.au>
- Paul Popelka <paulp@uts.amdahl.com>
- Peter da Silva <peter@NeoSoft.com>
- Phil Sutherland <philsuth@mycroft.dialix.oz.au>
- Poul-Henning Kamp <phk@FreeBSD.ORG>
- Ralf Friedl <friedl@informatik.uni-kl.de>
- Rick Macklem <root@snowwhite.cis.uoguelph.ca>
- Robert D. Thrush <rd@phoenix.aii.com>
- Rodney W. Grimes <rgrimes@cdrom.com>
- Sascha Wildner <swildner@channelz.GUN.de>
- Scott Burris <scott@pita.cns.ucla.edu>
- Scott Reynolds <scott@clmgt.marquette.mi.us>
- Sean Eric Fagan <sef@kithrup.com>
- Simon J Gerraty <sjg@melb.bull.oz.au> <sjg@zen.void.oz.au>
- Stephen McKay <syssgm@devetir.qld.gov.au>
- Terry Lambert <terry@icarus.weber.edu>
- Terry Lee <terry@uivlsi.csl.uiuc.edu>
- Tor Egge <Tor.Egge@idi.ntnu.no>
- Warren Toomey <wkt@csadfa.cs.adfa.oz.au>
- Wiljo Heinen <wiljo@freeside.ki.open.de>
- William Jolitz <withheld>
- Wolfgang Solfrank <ws@tools.de>
- Wolfgang Stanglmeier <wolf@dentaro.GUN.de>
- Yuval Yarom <yval@cs.huji.ac.il>

# Chapter 20. Source Tree Guidelines and Policies

*Contributed by Poul-Henning Kamp <phk@FreeBSD.ORG>.*

This chapter documents various guidelines and policies in force for the FreeBSD source tree.

## MAINTAINER on Makefiles

June 1996.

If a particular portion of the FreeBSD distribution is being maintained by a person or group of persons, they can communicate this fact to the world by adding a

```
MAINTAINER= email-addresses
```

line to the `Makefiles` covering this portion of the source tree.

The semantics of this are as follows:

The maintainer owns and is responsible for that code. This means that he is responsible for fixing bugs and answer problem reports pertaining to that piece of the code, and in the case of contributed software, for tracking new versions, as appropriate.

Changes to directories which have a maintainer defined shall be sent to the maintainer for review before being committed. Only if the maintainer does not respond for an unacceptable period of time, to several emails, will it be acceptable to commit changes without review by the maintainer. However, it is suggested that you try and have the changes reviewed by someone else if at all possible.

It is of course not acceptable to add a person or group as maintainer unless they agree to assume this duty. On the other hand it doesn't have to be a committer and it can easily be a group of people.

## Contributed Software

*Contributed by Poul-Henning Kamp <phk@FreeBSD.ORG> and David O'Brien <obrien@FreeBSD.ORG>.*

June 1996.

Some parts of the FreeBSD distribution consist of software that is actively being maintained outside the FreeBSD project. For historical reasons, we call this *contributed* software. Some examples are perl, gcc

and patch.

Over the last couple of years, various methods have been used in dealing with this type of software and all have some number of advantages and drawbacks. No clear winner has emerged.

Since this is the case, after some debate one of these methods has been selected as the “official” method and will be required for future imports of software of this kind. Furthermore, it is strongly suggested that existing contributed software converge on this model over time, as it has significant advantages over the old method, including the ability to easily obtain diffs relative to the “official” versions of the source by everyone (even without cvs access). This will make it significantly easier to return changes to the primary developers of the contributed software.

Ultimately, however, it comes down to the people actually doing the work. If using this model is particularly unsuited to the package being dealt with, exceptions to these rules may be granted only with the approval of the core team and with the general consensus of the other developers. The ability to maintain the package in the future will be a key issue in the decisions.

**Note:** Because of some unfortunate design limitations with the RCS file format and CVS's use of vendor branches, minor, trivial and/or cosmetic changes are *strongly discouraged* on files that are still tracking the vendor branch. “Spelling fixes” are explicitly included here under the “cosmetic” category and are to be avoided for files with revision 1.1.x.x. The repository bloat impact from a single character change can be rather dramatic.

The **Tcl** embedded programming language will be used as example of how this model works:

`src/contrib/tcl` contains the source as distributed by the maintainers of this package. Parts that are entirely not applicable for FreeBSD can be removed. In the case of Tcl, the `mac`, `win` and `compat` subdirectories were eliminated before the import

`src/lib/libtcl` contains only a “bmake style” Makefile that uses the standard `bsd.lib.mk` makefile rules to produce the library and install the documentation.

`src/usr.bin/tclsh` contains only a bmake style Makefile which will produce and install the `tclsh` program and its associated man-pages using the standard `bsd.prog.mk` rules.

`src/tools/tools/tcl_bmake` contains a couple of shell-scripts that can be of help when the tcl software needs updating. These are not part of the built or installed software.

The important thing here is that the `src/contrib/tcl` directory is created according to the rules: It is supposed to contain the sources as distributed (on a proper CVS vendor-branch and without RCS keyword expansion) with as few FreeBSD-specific changes as possible. The ‘easy-import’ tool on freefall will assist in doing the import, but if there are any doubts on how to go about it, it is imperative that you ask first and not blunder ahead and hope it “works out”. CVS is not forgiving of import accidents and a fair amount of effort is required to back out major mistakes.

Because of the previously mentioned design limitations with CVS's vendor branches, it is required that "official" patches from the vendor be applied to the original distributed sources and the result re-imported onto the vendor branch again. Official patches should never be patched into the FreeBSD checked out version and "committed", as this destroys the vendor branch coherency and makes importing future versions rather difficult as there will be conflicts.

Since many packages contain files that are meant for compatibility with other architectures and environments that FreeBSD, it is permissible to remove parts of the distribution tree that are of no interest to FreeBSD in order to save space. Files containing copyright notices and release-note kind of information applicable to the remaining files shall *not* be removed.

If it seems easier, the `bmake Makefiles` can be produced from the dist tree automatically by some utility, something which would hopefully make it even easier to upgrade to a new version. If this is done, be sure to check in such utilities (as necessary) in the `src/tools` directory along with the port itself so that it is available to future maintainers.

In the `src/contrib/tcl` level directory, a file called `FREEBSD-upgrade` should be added and it should state things like:

- Which files have been left out
- Where the original distribution was obtained from and/or the official master site.
- Where to send patches back to the original authors
- Perhaps an overview of the FreeBSD-specific changes that have been made.

However, please do not import `FREEBSD-upgrade` with the contributed source. Rather you should `cvs add FREEBSD-upgrade ; cvs ci` after the initial import. Example wording from `src/contrib/cpio` is below:

This directory contains virgin sources of the original distribution files on a "vendor" branch. Do not, under any circumstances, attempt to upgrade the files in this directory via patches and a `cvs commit`. New versions or official-patch versions must be imported. Please remember to import with `"-ko"` to prevent CVS from corrupting any vendor RCS Ids.

For the import of GNU `cpio 2.4.2`, the following files were removed:

```
INSTALL          cpio.info       mkdir.c
Makefile.in     cpio.texi      mkinstalldirs
```

To upgrade to a newer version of `cpio`, when it is available:

1. Unpack the new version into an empty directory.  
[Do not make ANY changes to the files.]

2. Remove the files listed above and any others that don't apply to FreeBSD.
3. Use the command:

```
cvsvimport -ko -m 'Virgin import of GNU cpio v<version>' \  
src/contrib/cpio GNU cpio_<version>
```

For example, to do the import of version 2.4.2, I typed:

```
cvsvimport -ko -m 'Virgin import of GNU v2.4.2' \  
src/contrib/cpio GNU cpio_2_4_2
```
4. Follow the instructions printed out in step 3 to resolve any conflicts between local FreeBSD changes and the newer version.

Do not, under any circumstances, deviate from this procedure.

To make local changes to cpio, simply patch and commit to the main branch (aka HEAD). Never make local changes on the GNU branch.

All local changes should be submitted to "cpio@gnu.ai.mit.edu" for inclusion in the next vendor release.

obrien@freebsd.org - 30 March 1997

## Shared Libraries

*Contributed by Satoshi Asami <asami@FreeBSD.ORG>, Peter Wemm <peter@FreeBSD.ORG>, and David O'Brien <obrien@FreeBSD.ORG> 9 December 1996.*

If you are adding shared library support to a port or other piece of software that doesn't have one, the version numbers should follow these rules. Generally, the resulting numbers will have nothing to do with the release version of the software.

The three principles of shared library building are:

- Start from 1.0
- If there is a change that is backwards compatible, bump minor number
- If there is an incompatible change, bump major number

For instance, added functions and bugfixes result in the minor version number being bumped, while deleted functions, changed function call syntax etc. will force the major version number to change.

Stick to version numbers of the form major.minor (*x.y*). Our dynamic linker does not handle version numbers of the form *x.y.z* well. Any version number after the *y* (ie. the third digit) is totally ignored when comparing shared lib version numbers to decide which library to link with. Given two shared libraries that differ only in the “micro” revision, `ld.so` will link with the higher one. Ie: if you link with `libfoo.so.3.3.3`, the linker only records `3.3` in the headers, and will link with anything starting with `libfoo.so.3.(anything >= 3).(highest available)`.

**Note:** `ld.so` will always use the highest “minor” revision. Ie: it will use `libc.so.2.2` in preference to `libc.so.2.0`, even if the program was initially linked with `libc.so.2.0`.

For non-port libraries, it is also our policy to change the shared library version number only once between releases. When you make a change to a system library that requires the version number to be bumped, check the `Makefile`'s commit logs. It is the responsibility of the committer to ensure that the first such change since the release will result in the shared library version number in the `Makefile` to be updated, and any subsequent changes will not.

# Chapter 21. Adding New Kernel Configuration Options

*Contributed by Jörg Wunsch <joerg@FreeBSD.ORG>*

**Note:** You should be familiar with the section about kernel configuration before reading here.

## What's a *Kernel Option*, Anyway?

The use of kernel options is basically described in the kernel configuration section. There's also an explanation of "historic" and "new-style" options. The ultimate goal is to eventually turn all the supported options in the kernel into new-style ones, so for people who correctly did a `make depend` in their kernel compile directory after running `config(8)`, the build process will automatically pick up modified options, and only recompile those files where it is necessary. Wiping out the old compile directory on each run of `config(8)` as it is still done now can then be eliminated again.

Basically, a kernel option is nothing else than the definition of a C preprocessor macro for the kernel compilation process. To make the build truly optional, the corresponding part of the kernel source (or kernel `.h` file) must be written with the option concept in mind, i.e. the default must have been made overridable by the `config` option. This is usually done with something like:

```
#ifndef THIS_OPTION
#define THIS_OPTION (some_default_value)
#endif /* THIS_OPTION */
```

This way, an administrator mentioning another value for the option in his `config` file will take the default out of effect, and replace it with his new value. Clearly, the new value will be substituted into the source code during the preprocessor run, so it must be a valid C expression in whatever context the default value would have been used.

It is also possible to create value-less options that simply enable or disable a particular piece of code by embracing it in

```
#ifdef THAT_OPTION

[your code here]

#endif
```

Simply mentioning `THAT_OPTION` in the config file (with or without any value) will then turn on the corresponding piece of code.

People familiar with the C language will immediately recognize that everything could be counted as a “config option” where there is at least a single `#ifdef` referencing it... However, it’s unlikely that many people would put

```
options notyet,notdef
```

in their config file, and then wonder why the kernel compilation falls over. :-)

Clearly, using arbitrary names for the options makes it very hard to track their usage throughout the kernel source tree. That is the rationale behind the *new-style* option scheme, where each option goes into a separate `.h` file in the kernel compile directory, which is by convention named `opt_foo.h`. This way, the usual Makefile dependencies could be applied, and `make` can determine what needs to be recompiled once an option has been changed.

The old-style option mechanism still has one advantage for local options or maybe experimental options that have a short anticipated lifetime: since it is easy to add a new `#ifdef` to the kernel source, this has already made it a kernel config option. In this case, the administrator using such an option is responsible himself for knowing about its implications (and maybe manually forcing the recompilation of parts of his kernel). Once the transition of all supported options has been done, `config(8)` will warn whenever an unsupported option appears in the config file, but it will nevertheless include it into the kernel Makefile.

## Now What Do I Have to Do for it?

First, edit `sys/conf/options` (or `sys/i386/conf/options.<arch>`, e. g. `sys/i386/conf/options.i386`), and select an `opt_foo.h` file where your new option would best go into.

If there is already something that comes close to the purpose of the new option, pick this. For example, options modifying the overall behaviour of the SCSI subsystem can go into `opt_scsi.h`. By default, simply mentioning an option in the appropriate option file, say `FOO`, implies its value will go into the corresponding file `opt_foo.h`. This can be overridden on the right-hand side of a rule by specifying another filename.

If there is no `opt_foo.h` already available for the intended new option, invent a new name. Make it meaningful, and comment the new section in the `options[.<arch>]` file. `config(8)` will automatically pick up the change, and create that file next time it is run. Most options should go in a header file by themselves..

Packing too many options into a single `opt_foo.h` will cause too many kernel files to be rebuilt when one of the options has been changed in the config file.

Finally, find out which kernel files depend on the new option. Unless you have just invented your option, and it does not exist anywhere yet,

```
% find /usr/src/sys -name  
  type f | xargs fgrep NEW_OPTION
```

is your friend in finding them. Go and edit all those files, and add

```
#include "opt_foo.h"
```

*on top*, before all the `#include <xxx.h>` stuff. This sequence is most important as the options could override defaults from the regular include files, if the defaults are of the form

```
#ifndef NEW_OPTION #define NEW_OPTION (something)  
#endif
```

in the regular header.

Adding an option that overrides something in a system header file (i.e., a file sitting in `/usr/include/sys/`) is almost always a mistake. `opt_foo.h` cannot be included into those files since it would break the headers more seriously, but if it is not included, then places that include it may get an inconsistent value for the option. Yes, there are precedents for this right now, but that does not make them more correct.

# Chapter 22. Kernel Debugging

*Contributed by Paul Richards <paul@FreeBSD.ORG> and Jörg Wunsch <joerg@FreeBSD.ORG>*

## Debugging a Kernel Crash Dump with `kgdb`

Here are some instructions for getting kernel debugging working on a crash dump. They assume that you have enough swap space for a crash dump. If you have multiple swap partitions and the first one is too small to hold the dump, you can configure your kernel to use an alternate dump device (in the `config kernel` line), or you can specify an alternate using the `dumpon(8)` command. The best way to use `dumpon(8)` is to set the `dumpdev` variable in `/etc/rc.conf`. Typically you want to specify one of the swap devices specified in `/etc/fstab`. Dumps to non-swap devices, tapes for example, are currently not supported. Config your kernel using `config -g`. See Kernel Configuration for details on configuring the FreeBSD kernel.

Use the `dumpon(8)` command to tell the kernel where to dump to (note that this will have to be done after configuring the partition in question as swap space via `swapon(8)`). This is normally arranged via `/etc/rc.conf` and `/etc/rc`. Alternatively, you can hard-code the dump device via the `dump` clause in the `config` line of your kernel config file. This is deprecated and should be used only if you want a crash dump from a kernel that crashes during booting.

**Note:** In the following, the term `kgdb` refers to `gdb` run in “kernel debug mode”. This can be accomplished by either starting the `gdb` with the option `-k`, or by linking and starting it under the name `kgdb`. This is not being done by default, however, and the idea is basically deprecated since the GNU folks do not like their tools to behave differently when called by another name. This feature may well be discontinued in further releases.

When the kernel has been built make a copy of it, say `kernel.debug`, and then run `strip -d` on the original. Install the original as normal. You may also install the unstripped kernel, but symbol table lookup time for some programs will drastically increase, and since the whole kernel is loaded entirely at boot time and cannot be swapped out later, several megabytes of physical memory will be wasted.

If you are testing a new kernel, for example by typing the new kernel’s name at the boot prompt, but need to boot a different one in order to get your system up and running again, boot it only into single user state using the `-s` flag at the boot prompt, and then perform the following steps:

```
# fsck -p
# mount -a -t ufs          # so your file system for /var/crash is writable
# savecore -N /kernel.panicked /var/crash
# exit                    # ...to multi-user
```

This instructs `savecore(8)` to use another kernel for symbol name extraction. It would otherwise default to the currently running kernel and most likely not do anything at all since the crash dump and the kernel symbols differ.

Now, after a crash dump, go to `/sys/compile/WHATEVER` and run `kgdb`. From `kgdb` do:

```
symbol-file kernel.debug
exec-file /var/crash/kernel.0
core-file /var/crash/vmcore.0
```

and voila, you can debug the crash dump using the kernel sources just like you can for any other program.

Here is a script log of a `kgdb` session illustrating the procedure. Long lines have been folded to improve readability, and the lines are numbered for reference. Despite this, it is a real-world error trace taken during the development of the `pcvt` console driver.

```
1:Script started on Fri Dec 30 23:15:22 1994
2:# cd /sys/compile/URIAH
3:# kgdb kernel /var/crash/vmcore.1
4:Reading symbol data from /usr/src/sys/compile/URIAH/kernel...done.
5:IdlePTD 1f3000
6:panic: because you said to!
7:current pcb at 1e3f70
8:Reading in symbols for ../../i386/i386/machdep.c...done.
9:(kgdb) where
10:#0 boot (arghowto=256) (../../i386/i386/machdep.c line 767)
11:#1 0xf0115159 in panic ()
12:#2 0xf01955bd in diediedie () (../../i386/i386/machdep.c line 698)
13:#3 0xf010185e in db_fncall ()
14:#4 0xf0101586 in db_command (-266509132, -266509516, -267381073)
15:#5 0xf0101711 in db_command_loop ()
16:#6 0xf01040a0 in db_trap ()
17:#7 0xf0192976 in kdb_trap (12, 0, -272630436, -266743723)
18:#8 0xf019d2eb in trap_fatal (...)
19:#9 0xf019ce60 in trap_pfault (...)
20:#10 0xf019cb2f in trap (...)
21:#11 0xf01932a1 in exception:calltrap ()
22:#12 0xf0191503 in cnopen (...)
23:#13 0xf0132c34 in spec_open ()
24:#14 0xf012d014 in vn_open ()
25:#15 0xf012a183 in open ()
26:#16 0xf019d4eb in syscall (...)
27:(kgdb) up 10
28:Reading in symbols for ../../i386/i386/trap.c...done.
29:#10 0xf019cb2f in trap (frame={tf_es = -260440048, tf_ds = 16, tf_\
30:edi = 3072, tf_esi = -266445372, tf_ebp = -272630356, tf_esp = -27\
```

```

31:2630396, tf_ebx = -266427884, tf_edx = 12, tf_ecx = -266427884, tf\
32:_eax = 64772224, tf_trapno = 12, tf_err = -272695296, tf_eip = -26\
33:6672343, tf_cs = -266469368, tf_eflags = 66066, tf_esp = 3072, tf_\
34:ss = -266427884}) (../../i386/i386/trap.c line 283)
35:283                                (void) trap_pfault(&frame, FALSE);
36:(kgdb) frame frame->tf_ebp frame->tf_eip
37:Reading in symbols for ../../i386/isa/pcvt/pcvt_drv.c...done.
38:#0  0xf01ae729 in pcopen (dev=3072, flag=3, mode=8192, p=(struct p\
39:roc *) 0xf07c0c00) (../../i386/isa/pcvt/pcvt_drv.c line 403)
40:403                                return ((*linesw[tp->t_line].l_open)(dev, tp));
41:(kgdb) list
42:398
43:399                                tp->t_state |= TS_CARR_ON;
44:400                                tp->t_cflag |= CLOCAL; /* cannot be a modem (-) */
45:401
46:402                                #if PCVT_NETBSD || (PCVT_FREEBSD >= 200)
47:403                                    return ((*linesw[tp->t_line].l_open)(dev, tp));
48:404                                #else
49:405                                    return ((*linesw[tp->t_line].l_open)(dev, tp, flag));
50:406                                #endif /* PCVT_NETBSD || (PCVT_FREEBSD >= 200) */
51:407                                }
52:(kgdb) print tp
53:Reading in symbols for ../../i386/i386/cons.c...done.
54:$1 = (struct tty *) 0x1bae
55:(kgdb) print tp->t_line
56:$2 = 1767990816
57:(kgdb) up
58:#1  0xf0191503 in cnopen (dev=0x00000000, flag=3, mode=8192, p=(st\
59:ruct proc *) 0xf07c0c00) (../../i386/i386/cons.c line 126)
60:    return ((*cdevsw[major(dev)].d_open)(dev, flag, mode, p));
61:(kgdb) up
62:#2  0xf0132c34 in spec_open ()
63:(kgdb) up
64:#3  0xf012d014 in vn_open ()
65:(kgdb) up
66:#4  0xf012a183 in open ()
67:(kgdb) up
68:#5  0xf019d4eb in syscall (frame={tf_es = 39, tf_ds = 39, tf_edi = \
69: 2158592, tf_esi = 0, tf_ebp = -272638436, tf_esp = -272629788, tf\
70:_ebx = 7086, tf_edx = 1, tf_ecx = 0, tf_eax = 5, tf_trapno = 582, \
71:tf_err = 582, tf_eip = 75749, tf_cs = 31, tf_eflags = 582, tf_esp \
72:= -272638456, tf_ss = 39}) (../../i386/i386/trap.c line 673)
73:673                                error = (*callp->sy_call)(p, args, rval);
74:(kgdb) up
75:Initial frame selected; you cannot go up.

```

```

76:(kgdb) quit
77:# exit
78:exit
79:
80:Script done on Fri Dec 30 23:18:04 1994

```

Comments to the above script:

line 6:

This is a dump taken from within DDB (see below), hence the panic comment “because you said to!”, and a rather long stack trace; the initial reason for going into DDB has been a page fault trap though.

line 20:

This is the location of function `trap()` in the stack trace.

line 36:

Force usage of a new stack frame; this is no longer necessary now. The stack frames are supposed to point to the right locations now, even in case of a trap. (I do not have a new core dump handy `<g>`, my kernel has not panicked for a rather long time.) From looking at the code in source line 403, there is a high probability that either the pointer access for “tp” was messed up, or the array access was out of bounds.

line 52:

The pointer looks suspicious, but happens to be a valid address.

line 56:

However, it obviously points to garbage, so we have found our error! (For those unfamiliar with that particular piece of code: `tp->t_line` refers to the line discipline of the console device here, which must be a rather small integer number.)

## Debugging a crash dump with DDD

Examining a kernel crash dump with a graphical debugger like `ddd` is also possible. Add the `-k` option to the `ddd` command line you would use normally. For example;

```
# ddd -k /var/crash/kernel.0 /var/crash/vmcore.0
```

You should then be able to go about looking at the crash dump using `ddd`'d graphical interface.

## Post-mortem Analysis of a Dump

What do you do if a kernel dumped core but you did not expect it, and it is therefore not compiled using `config -g`? Not everything is lost here. Do not panic!

Of course, you still need to enable crash dumps. See above on the options you have to specify in order to do this.

Go to your kernel compile directory, and edit the line containing `COPTFLAGS?=-O`. Add the `-g` option there (but *do not* change anything on the level of optimization). If you do already know roughly the probable location of the failing piece of code (e.g., the `pcvt` driver in the example above), remove all the object files for this code. Rebuild the kernel. Due to the time stamp change on the Makefile, there will be some other object files rebuild, for example `trap.o`. With a bit of luck, the added `-g` option will not change anything for the generated code, so you will finally get a new kernel with similar code to the faulting one but some debugging symbols. You should at least verify the old and new sizes with the `size(1)` command. If there is a mismatch, you probably need to give up here.

Go and examine the dump as described above. The debugging symbols might be incomplete for some places, as can be seen in the stack trace in the example above where some functions are displayed without line numbers and argument lists. If you need more debugging symbols, remove the appropriate object files and repeat the `kgdb` session until you know enough.

All this is not guaranteed to work, but it will do it fine in most cases.

## On-line Kernel Debugging Using DDB

While `kgdb` as an offline debugger provides a very high level of user interface, there are some things it cannot do. The most important ones being breakpointing and single-stepping kernel code.

If you need to do low-level debugging on your kernel, there is an on-line debugger available called DDB. It allows to setting breakpoints, single-stepping kernel functions, examining and changing kernel variables, etc. However, it cannot access kernel source files, and only has access to the global and static symbols, not to the full debug information like `kgdb`.

To configure your kernel to include DDB, add the option line

```
options DDB
```

to your config file, and rebuild. (See Kernel Configuration for details on configuring the FreeBSD kernel.

**Note:** Note that if you have an older version of the boot blocks, your debugger symbols might not be loaded at all. Update the boot blocks; the recent ones load the DDB symbols automatically.)

Once your DDB kernel is running, there are several ways to enter DDB. The first, and earliest way is to type the boot flag `-d` right at the boot prompt. The kernel will start up in debug mode and enter DDB prior to any device probing. Hence you can even debug the device probe/attach functions.

The second scenario is a hot-key on the keyboard, usually Ctrl-Alt-ESC. For syscons, this can be remapped; some of the distributed maps do this, so watch out. There is an option available for serial consoles that allows the use of a serial line BREAK on the console line to enter DDB (`options BREAK_TO_DEBUGGER` in the kernel config file). It is not the default since there are a lot of crappy serial adapters around that gratuitously generate a BREAK condition, for example when pulling the cable.

The third way is that any panic condition will branch to DDB if the kernel is configured to use it. For this reason, it is not wise to configure a kernel with DDB for a machine running unattended.

The DDB commands roughly resemble some `gdb` commands. The first thing you probably need to do is to set a breakpoint:

```
b function-name
b address
```

Numbers are taken hexadecimal by default, but to make them distinct from symbol names; hexadecimal numbers starting with the letters `a-f` need to be preceded with `0x` (this is optional for other numbers). Simple expressions are allowed, for example: `function-name + 0x103`.

To continue the operation of an interrupted kernel, simply type:

```
c
```

To get a stack trace, use:

```
trace
```

**Note:** Note that when entering DDB via a hot-key, the kernel is currently servicing an interrupt, so the stack trace might be not of much use for you.

If you want to remove a breakpoint, use

```
del
del address-expression
```

The first form will be accepted immediately after a breakpoint hit, and deletes the current breakpoint. The second form can remove any breakpoint, but you need to specify the exact address; this can be obtained from:

```
show b
```

To single-step the kernel, try:

```
s
```

This will step into functions, but you can make DDB trace them until the matching return statement is reached by:

```
n
```

**Note:** This is different from `gdb's next` statement; it is like `gdb's finish`.

To examine data from memory, use (for example):

```
x/wx 0xf0133fe0,40
x/hd db_syntab_space
x/bc termbuf,10
x/s stringbuf
```

for word/halfword/byte access, and hexadecimal/decimal/character/ string display. The number after the comma is the object count. To display the next 0x10 items, simply use:

```
x ,10
```

Similarly, use

```
x/ia foofunc,10
```

to disassemble the first 0x10 instructions of `foofunc`, and display them along with their offset from the beginning of `foofunc`.

To modify memory, use the write command:

```
w/b termbuf 0xa 0xb 0
w/w 0xf0010030 0 0
```

The command modifier (b/h/w) specifies the size of the data to be written, the first following expression is the address to write to and the remainder is interpreted as data to write to successive memory locations.

If you need to know the current registers, use:

```
show reg
```

Alternatively, you can display a single register value by e.g.

```
p $eax
```

and modify it by:

```
set $eax new-value
```

Should you need to call some kernel functions from DDB, simply say:

```
call func(arg1, arg2, ...)
```

The return value will be printed.

For a ps(1) style summary of all running processes, use:

```
ps
```

Now you have now examined why your kernel failed, and you wish to reboot. Remember that, depending on the severity of previous malfunctioning, not all parts of the kernel might still be working as expected. Perform one of the following actions to shut down and reboot your system:

```
call diediedie()
```

This will cause your kernel to dump core and reboot, so you can later analyze the core on a higher level with kgdb. This command usually must be followed by another `continue` statement. There is now an alias for this: `panic`.

```
call boot(0)
```

Which might be a good way to cleanly shut down the running system, `sync()` all disks, and finally reboot. As long as the disk and file system interfaces of the kernel are not damaged, this might be a good way for an almost clean shutdown.

```
call cpu_reset()
```

is the final way out of disaster and almost the same as hitting the Big Red Button.

If you need a short command summary, simply type:

```
help
```

However, it is highly recommended to have a printed copy of the ddb(4) manual page ready for a debugging session. Remember that it is hard to read the on-line manual while single-stepping the kernel.

## On-line Kernel Debugging Using Remote GDB

This feature has been supported since FreeBSD 2.2, and it's actually a very neat one.

GDB has already supported *remote debugging* for a long time. This is done using a very simple protocol along a serial line. Unlike the other methods described above, you will need two machines for doing this. One is the host providing the debugging environment, including all the sources, and a copy of the kernel binary with all the symbols in it, and the other one is the target machine that simply runs a similar copy of the very same kernel (but stripped of the debugging information).

You should configure the kernel in question with `config -g`, include DDB into the configuration, and compile it as usual. This gives a large blurb of a binary, due to the debugging information. Copy this kernel to the target machine, strip the debugging symbols off with `strip -x`, and boot it using the `-d` boot option. Connect the first serial line of the target machine to any serial line of the debugging host. Now, on the debugging machine, go to the compile directory of the target kernel, and start gdb:

```
% gdb -k kernel
GDB is free software and you are welcome to distribute copies of it
  under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB 4.16 (i386-unknown-freebsd),
Copyright 1996 Free Software Foundation, Inc...
(kgdb)
```

Initialize the remote debugging session (assuming the first serial port is being used) by:

```
(kgdb) target remote /dev/cuaa0
```

Now, on the target host (the one that entered DDB right before even starting the device probe), type:

```
Debugger("Boot flags requested debugger")
Stopped at Debugger+0x35: movb $0, edata+0x51bc
db> gdb
```

DDB will respond with:

```
Next trap will enter GDB remote protocol mode
```

Every time you type `gdb`, the mode will be toggled between remote GDB and local DDB. In order to force a next trap immediately, simply type `s` (step). Your hosting GDB will now gain control over the target kernel:

```
Remote debugging using /dev/cuaa0
Debugger (msg=0xf01b0383 "Boot flags requested debugger")
  at ../../i386/i386/db_interface.c:257
```

```
(kgdb)
```

You can use this session almost as any other GDB session, including full access to the source, running it in gud-mode inside an Emacs window (which gives you an automatic source code display in another Emacs window) etc.

Remote GDB can also be used to debug LKMs. First build the LKM with debugging symbols:

```
# cd /usr/src/lkm/linux
# make clean; make COPTS=-g
```

Then install this version of the module on the target machine, load it and use `modstat` to find out where it was loaded:

```
# linux
# modstat
Type      Id Off Loadaddr Size Info      Rev Module Name
EXEC      0  4 f5109000 001c f510f010  1 linux_mod
```

Take the load address of the module and add 0x20 (probably to account for the a.out header). This is the address that the module code was relocated to. Use the `add-symbol-file` command in GDB to tell the debugger about the module:

```
(kgdb) add-symbol-file /usr/src/lkm/linux/linux_mod.o 0xf5109020
add symbol table from file "/usr/src/lkm/linux/linux_mod.o" at
text_addr = 0xf5109020? (y or n) y
(kgdb)
```

You now have access to all the symbols in the LKM.

## Debugging a Console Driver

Since you need a console driver to run DDB on, things are more complicated if the console driver itself is failing. You might remember the use of a serial console (either with modified boot blocks, or by specifying `-h` at the `Boot:` prompt), and hook up a standard terminal onto your first serial port. DDB works on any configured console driver, of course also on a serial console.

# Chapter 23. Linux Emulation

*Contributed by Brian N. Handy <handy@sxt4.physics.montana.edu> and Rich Murphey <rich@FreeBSD.ORG>*

## How to Install the Linux Emulator

Linux emulation in FreeBSD has reached a point where it is possible to run a large fraction of Linux binaries in both a.out and ELF format. The linux emulation in the 2.1-STABLE branch is capable of running Linux DOOM and Mathematica; the version present in 3.1-RELEASE is vastly more capable and runs all these as well as Quake, Abuse, IDL, netrek for Linux and a whole host of other programs.

There are some Linux-specific operating system features that are not supported on FreeBSD. Linux binaries will not work on FreeBSD if they use the Linux `/proc` filesystem (which is different from the optional FreeBSD `/proc` filesystem) or i386-specific calls, such as enabling virtual 8086 mode.

Depending on which version of FreeBSD you are running, how you get Linux-emulation up will vary slightly:

## Installing Linux Emulation in 2.1-STABLE

The `GENERIC` kernel in 2.1-STABLE is not configured for linux compatibility so you must reconfigure your kernel for it. There are two ways to do this: 1. linking the emulator statically in the kernel itself and 2. configuring your kernel to dynamically load the linux loadable kernel module (LKM).

To enable the emulator, add the following to your configuration file (c.f. `/sys/i386/conf/LINT`):

```
options COMPAT_LINUX
```

If you want to run doom or other applications that need shared memory, also add the following.

```
options SYSVSHM
```

The linux system calls require 4.3BSD system call compatibility. So make sure you have the following.

```
options "COMPAT_43"
```

If you prefer to statically link the emulator in the kernel rather than use the loadable kernel module (LKM), then add

```
options LINUX
```

Then run config and install the new kernel as described in the kernel configuration section.

If you decide to use the LKM you must also install the loadable module. A mismatch of versions between the kernel and loadable module can cause the kernel to crash, so the safest thing to do is to reinstall the LKM when you install the kernel.

```
# cd /usr/src/lkm/linux
# make all install
```

Once you have installed the kernel and the LKM, you can invoke 'linux' as root to load the LKM.

```
# linux
Linux emulator installed
Module loaded as ID 0
```

To see whether the LKM is loaded, run modstat.

```
% modstat
Type      Id Off Loadaddr Size Info      Rev
Module Name EXEC      0   3 f0baf000 0018 f0bb4000   1 linux_emulator
```

You can cause the LKM to be loaded when the system boots in either of two ways. In FreeBSD 2.2.1-RELEASE and 2.1-STABLE enable it in `/etc/sysconfig`

```
linux=YES
```

by changing it from NO to YES. FreeBSD 2.1 RELEASE and earlier do not have such a line and on those you will need to edit `/etc/rc.local` to add the following line.

```
linux
```

## Installing Linux Emulation in 2.2.2-RELEASE and later

It is no longer necessary to specify options `LINUX` or options `COMPAT_LINUX`. Linux emulation is done with an LKM ("Loadable Kernel Module") so it can be installed on the fly without having to reboot. You will need the following things in your startup files, however:

1. In `/etc/rc.conf`, you need the following line:

```
linux_enable=YES
```

2. This, in turn, triggers the following action in `/etc/rc.i386`:

```
# Start the Linux binary emulation if requested.
```

```

if [ "X${linux_enable}" = X"YES" ]; then echo -n '
        linux';                linux > /dev/null 2>&1
fi

```

If you want to verify it is running, modstat will do that:

```

% modstat
Type      Id Off Loadaddr Size Info      Rev Module Name
EXEC      0  4 f09e6000 001c f09ec010  1 linux_mod

```

However, there have been reports that this fails on some 2.2-RELEASE and later systems. If for some reason you cannot load the linux LKM, then statically link the emulator in the kernel by adding

```
options LINUX
```

to your kernel config file. Then run config and install the new kernel as described in the kernel configuration section.

## Installing Linux Runtime Libraries

### Installing using the linux\_lib port

Most linux applications use shared libraries, so you are still not done until you install the shared libraries. It is possible to do this by hand, however, it is vastly simpler to just grab the linux\_lib port:

```

# cd /usr/ports/emulators/linux_lib
# make all install

```

and you should have a working linux emulator. Legend (and the mail archives :-)) seems to hold that Linux emulation works best with linux binaries linked against the ZMAGIC libraries; QMAGIC libraries (such as those used in Slackware V2.0) may tend to give the Linuxulator heartburn. Also, expect some programs to complain about incorrect minor versions of the system libraries. In general, however, this does not seem to be a problem.

### Installing libraries manually

If you do not have the “ports” distribution, you can install the libraries by hand instead. You will need the Linux shared libraries that the program depends on and the runtime linker. Also, you will need to create a “shadow root” directory, /compat/linux, for Linux libraries on your FreeBSD system. Any shared libraries opened by Linux programs run under FreeBSD will look in this tree first. So, if a Linux program loads, for example, /lib/libc.so, FreeBSD will first try to open /compat/linux/lib/libc.so,

and if that does not exist then it will try `/lib/libc.so`. Shared libraries should be installed in the shadow tree `/compat/linux/lib` rather than the paths that the Linux `ld.so` reports.

FreeBSD-2.2-RELEASE and later works slightly differently with respect to `/compat/linux`: all files, not just libraries, are searched for from the “shadow root” `/compat/linux`.

Generally, you will need to look for the shared libraries that Linux binaries depend on only the first few times that you install a Linux program on your FreeBSD system. After a while, you will have a sufficient set of Linux shared libraries on your system to be able to run newly imported Linux binaries without any extra work.

## How to install additional shared libraries

What if you install the `linux_lib` port and your application still complains about missing shared libraries? How do you know which shared libraries Linux binaries need, and where to get them?

Basically, there are 2 possibilities (when following these instructions: you will need to be root on your FreeBSD system to do the necessary installation steps).

If you have access to a Linux system, see what shared libraries the application needs, and copy them to your FreeBSD system. Example: you have just ftp'ed the Linux binary of Doom. Put it on the Linux system you have access to, and check which shared libraries it needs by running `ldd linuxxdoom`:

```
% ldd linuxxdoom
libXt.so.3 (DLL Jump 3.1) => /usr/X11/lib/libXt.so.3.1.0
libX11.so.3 (DLL Jump 3.1) => /usr/X11/lib/libX11.so.3.1.0
libc.so.4 (DLL Jump 4.5p126) => /lib/libc.so.4.6.29
```

You would need to get all the files from the last column, and put them under `/compat/linux`, with the names in the first column as symbolic links pointing to them. This means you eventually have these files on your FreeBSD system:

```
/compat/linux/usr/X11/lib/libXt.so.3.1.0
/compat/linux/usr/X11/lib/libXt.so.3 -> libXt.so.3.1.0
/compat/linux/usr/X11/lib/libX11.so.3.1.0
/compat/linux/usr/X11/lib/libX11.so.3 -> libX11.so.3.1.0
/compat/linux/lib/libc.so.4.6.29 /compat/linux/lib/libc.so.4 -
> libc.so.4.6.29
```

**Note:** Note that if you already have a Linux shared library with a matching major revision number to the first column of the `ldd` output, you will not need to copy the file named in the last column to your system, the one you already have should work. It is advisable to copy the shared library anyway if it is a newer version, though. You can remove the old one, as long as you make the symbolic link point to the new one. So, if you have these libraries on your system:

```
/compat/linux/lib/libc.so.4.6.27
/compat/linux/lib/libc.so.4 -> libc.so.4.6.27
```

and you find a new binary that claims to require a later version according to the output of `ldd`:

```
libc.so.4 (DLL Jump 4.5pl26) -> libc.so.4.6.29
```

If it is only one or two versions out of date in the in the trailing digit then do not worry about copying `/lib/libc.so.4.6.29` too, because the program should work fine with the slightly older version. However, if you like you can decide to replace the `libc.so` anyway, and that should leave you with:

```
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

**Note:** The symbolic link mechanism is *only* needed for Linux binaries. The FreeBSD runtime linker takes care of looking for matching major revision numbers itself and you do not need to worry about it.

## Configuring the `ld.so` — for FreeBSD 2.2-RELEASE and later

This section applies only to FreeBSD 2.2-RELEASE and later. Those running 2.1-STABLE should skip this section.

Finally, if you run FreeBSD 2.2-RELEASE you must make sure that you have the Linux runtime linker and its config files on your system. You should copy these files from the Linux system to their appropriate place on your FreeBSD system (to the `/compat/linux` tree):

```
/compat/linux/lib/ld.so
/compat/linux/etc/ld.so.config
```

If you do not have access to a Linux system, you should get the extra files you need from various ftp sites. Information on where to look for the various files is appended below. For now, let us assume you know where to get the files.

Retrieve the following files (all from the same ftp site to avoid any version mismatches), and install them under `/compat/linux` (i.e. `/foo/bar` is installed as `/compat/linux/foo/bar`):

```
/sbin/ldconfig
/usr/bin/ldd
/lib/libc.so.x.y.z
/lib/ld.so
```

`ldconfig` and `ldd` do not necessarily need to be under `/compat/linux`; you can install them elsewhere in the system too. Just make sure they do not conflict with their FreeBSD counterparts. A good idea would be to install them in `/usr/local/bin` as `ldconfig-linux` and `ldd-linux`.

Create the file `/compat/linux/etc/ld.so.conf`, containing the directories in which the Linux runtime linker should look for shared libs. It is a plain text file, containing a directory name on each line. `/lib` and `/usr/lib` are standard, you could add the following:

```
/usr/X11/lib
/usr/local/lib
```

When a linux binary opens a library such as `/lib/libc.so` the emulator maps the name to `/compat/linux/lib/libc.so` internally. All linux libraries should be installed under `/compat/linux` (e.g. `/compat/linux/lib/libc.so`, `/compat/linux/usr/X11/lib/libX11.so`, etc.) in order for the emulator to find them.

Those running FreeBSD 2.2-RELEASE should run the Linux `ldconfig` program.

```
# cd /compat/linux/lib
# /compat/linux/sbin/ldconfig
```

`ldconfig` is statically linked, so it does not need any shared libraries to run. It creates the file `/compat/linux/etc/ld.so.cache` which contains the names of all the shared libraries and should be rerun to recreate this file whenever you install additional shared libraries.

On 2.1-STABLE do not install `/compat/linux/etc/ld.so.cache` or run `ldconfig`; in 2.1-STABLE the syscalls are implemented differently and `ldconfig` is not needed or used.

You should now be set up for Linux binaries which only need a shared `libc`. You can test this by running the Linux `ldd` on itself. Supposing that you have it installed as `ldd-linux`, it should produce something like:

```
# ldd-linux `which ldd-linux`
libc.so.4 (DLL Jump 4.5p126) => /lib/libc.so.4.6.29
```

This being done, you are ready to install new Linux binaries. Whenever you install a new Linux program, you should check if it needs shared libraries, and if so, whether you have them installed in the `/compat/linux` tree. To do this, you run the Linux version `ldd` on the new program, and watch its output. `ldd` (see also the manual page for `ldd(1)`) will print a list of shared libraries that the program depends on, in the form *majorname (jumpversion) => fullname*.

If it prints `not found` instead of *fullname* it means that you need an extra library. The library needed is shown in *majorname* and will be of the form `libXXXX.so.N`. You will need to find a `libXXXX.so.N.mm` on a Linux ftp site, and install it on your system. The `XXXX` (name) and `N` (major

revision number) should match; the minor number(s) *mm* are less important, though it is advised to take the most recent version.

## Installing Linux ELF binaries

ELF binaries sometimes require an extra step of “branding”. If you attempt to run an unbranded ELF binary, you will get an error message like the following;

```
% ./my-linux-elf-binary
ELF binary type not known
Abort
```

To help the FreeBSD kernel distinguish between a FreeBSD ELF binary from a Linux binary, use the `brandelf(1)` utility.

```
% brandelf -t Linux my-linux-elf-binary
```

The GNU toolchain now places the appropriate branding information into ELF binaries automatically, so you should be needing to do this step increasingly rarely in future.

## Configuring the host name resolver

If DNS does not work or you get the messages

```
resolv+: "bind" is an invalid keyword
resolv+: "hosts" is an invalid keyword
```

then you need to configure a `/compat/linux/etc/host.conf` file containing:

```
order hosts, bind
multi on
```

where the order here specifies that `/etc/hosts` is searched first and DNS is searched second. When `/compat/linux/etc/host.conf` is not installed linux applications find FreeBSD's `/etc/host.conf` and complain about the incompatible FreeBSD syntax. You should remove `bind` if you have not configured a name-server using the `/etc/resolv.conf` file.

Lastly, those who run 2.1-STABLE need to set an the `RESOLV_HOST_CONF` environment variable so that applications will know how to search the host tables. If you run FreeBSD 2.2-RELEASE or later, you can skip this. For the `/bin/csh` shell use:

```
% setenv RESOLV_HOST_CONF /compat/linux/etc/host.conf
```

For /bin/sh use:

```
% RESOLV_HOST_CONF=/compat/linux/etc/host.conf; export RESOLV_HOST_CONF
```

## Finding the necessary files

**Note:** The information below is valid as of the time this document was written, but certain details such as names of ftp sites, directories and distribution names may have changed by the time you read this.

Linux is distributed by several groups that make their own set of binaries that they distribute. Each distribution has its own name, like “Slackware” or “Yggdrasil”. The distributions are available on a lot of ftp sites. Sometimes the files are unpacked, and you can get the individual files you need, but mostly they are stored in distribution sets, usually consisting of subdirectories with gzipped tar files in them. The primary ftp sites for the distributions are:

1. sunsite.unc.edu:/pub/Linux/distributions
2. tsx-11.mit.edu:/pub/linux/distributions

Some European mirrors:

1. ftp.luth.se:/pub/linux/distributions
2. ftp.demon.co.uk:/pub/unix/linux
3. src.doc.ic.ac.uk:/packages/linux/distributions

For simplicity, let us concentrate on Slackware here. This distribution consists of a number of subdirectories, containing separate packages. Normally, they are controlled by an install program, but you can retrieve files “by hand” too. First of all, you will need to look in the `contents` subdir of the distribution. You will find a lot of small text files here describing the contents of the separate packages. The fastest way to look something up is to retrieve all the files in the `contents` subdirectory, and `grep` through them for the file you need. Here is an example of a list of files that you might need, and in which contents-file you will find it by grepping through them:

Library	Package
<code>ld.so</code>	<code>ldso</code>
<code>ldconfig</code>	<code>ldso</code>

ldd	ldso
libc.so.4	shlibs
libX11.so.6.0	xf_lib
libXt.so.6.0	xf_lib
libX11.so.3	oldlibs
libXt.so.3	oldlibs

So, in this case, you will need the packages `ldso`, `shlibs`, `xf_lib` and `oldlibs`. In each of the contents-files for these packages, look for a line saying `PACKAGE LOCATION`, it will tell you on which “disk” the package is, in our case it will tell us in which subdirectory we need to look. For our example, we would find the following locations:

Package	Location
<code>ldso</code>	<code>diska2</code>
<code>shlibs</code>	<code>diska2</code>
<code>oldlibs</code>	<code>diskx6</code>
<code>xf_lib</code>	<code>diskx9</code>

The locations called “diskXX” refer to the `slakware/XX` subdirectories of the distribution, others may be found in the `contrib` subdirectory. In this case, we could now retrieve the packages we need by retrieving the following files (relative to the root of the Slackware distribution tree):

- `slakware/a2/ldso.tgz`
- `slakware/a2/shlibs.tgz`
- `slakware/x6/oldlibs.tgz`
- `slakware/x9/xf_lib.tgz`

Extract the files from these gzipped tarfiles in your `/compat/linux` directory (possibly omitting or afterwards removing files you do not need), and you are done.

*See also:* `ftp.freebsd.org:pub/FreeBSD/2.0.5-RELEASE/xperimnt/linux-emu/README` and `/usr/src/sys/i386/ibcs2/README.iBCS2`

## How to Install Mathematica on FreeBSD

*Contributed by Rich Murphey <rich@FreeBSD.ORG> and Chuck Robey <chuckr@glue.umd.edu>*

This document shows how to install the Linux binary distribution of Mathematica 2.2 on FreeBSD 2.1.

Mathematica supports Linux but not FreeBSD as it stands. So once you have configured your system for Linux compatibility you have most of what you need to run Mathematica.

For those who already have the student edition of Mathematica for DOS the cost of upgrading to the Linux version at the time this was written, March 1996, was \$45.00. It can be ordered directly from Wolfram at (217) 398-6500 and paid for by credit card.

## Unpacking the Mathematica distribution

The binaries are currently distributed by Wolfram on CDROM. The CDROM has about a dozen tar files, each of which is a binary distribution for one of the supported architectures. The one for Linux is named `LINUX.TAR`. You can, for example, unpack this into `/usr/local/Mathematica`:

```
# cd /usr/local
# mkdir Mathematica
# cd Mathematica
# tar -xvf /cdrom/LINUX.TAR
```

## Obtaining your Mathematica Password

Before you can run Mathematica you will have to obtain a password from Wolfram that corresponds to your “machine ID”.

Once you have installed the linux compatibility runtime libraries and unpacked the mathematica you can obtain the “machine ID” by running the program `mathinfo` in the `Install` directory.

```
# cd /usr/local/Mathematica/Install
# mathinfo
LINUX: 'ioctl' fd=5, typ=0x89(), num=0x27 not implemented
richc.isdn.bcm.tmc.edu 9845-03452-90255
```

So, for example, the “machine ID” of `richc` is `9845-03452-90255`. You can ignore the message about the `ioctl` that is not implemented. It will not prevent Mathematica from running in any way and you can safely ignore it, though you will see the message every time you run Mathematica.

When you register with Wolfram, either by email, phone or fax, you will give them the “machine ID” and they will respond with a corresponding password consisting of groups of numbers. You need to add them both along with the machine name and license number in your `mathpass` file.

You can do this by invoking:

```
# cd /usr/local/Mathematica/Install
# math.install
```

It will ask you to enter your license number and the Wolfram supplied password. If you get them mixed up or for some reason the `math.install` fails, that is OK; you can simply edit the file `mathpass` in this same directory to correct the info manually.

After getting past the password, `math.install` will ask you if you accept the install defaults provided, or if you want to use your own. If you are like us and distrust all install programs, you probably want to specify the actual directories. Beware. Although the `math.install` program asks you to specify directories, it will not create them for you, so you should perhaps have a second window open with another shell so that you can create them before you give them to the install program. Or, if it fails, you can create the directories and then restart the `math.install` program. The directories we chose to create beforehand and specify to `math.install` were:

```
/usr/local/Mathematica/bin           for binaries
/usr/local/Mathematica/man/man1      for man pages
/usr/local/Mathematica/lib/X11       for the XKeysymb file
```

You can also tell it to use `/tmp/math.record` for the system record file, where it puts logs of sessions. After this `math.install` will continue on to unpacking things and placing everything where it should go.

The Mathematica Notebook feature is included separately, as the X Front End, and you have to install it separately. To get the X Front End stuff correctly installed, `cd` into the `/usr/local/Mathematica/FrontEnd` directory and execute the `xfe.install` shell script. You will have to tell it where to put things, but you do not have to create any directories because it will use the same directories that had been created for `math.install`. When it finishes, there should be a new shell script in `/usr/local/Mathematica/bin` called `mathematica`.

Lastly, you need to modify each of the shell scripts that Mathematica has installed. At the beginning of every shell script in `/usr/local/Mathematica/bin` add the following line:

```
% XKEYSYMDB=/usr/local/Mathematica/lib/X11/XKeysymbDB; export XKEYSYMDB
```

This tells Mathematica where to find its own version of the key mapping file `XKeysymbDB`. Without this you will get pages of error messages about missing key mappings.

On 2.1-STABLE you need to add the following as well:

```
% RESOLV_HOST_CONF=/compat/linux/etc/host.conf; export RESOLV_HOST_CONF
```

This tells Mathematica to use the linux version of `host.conf`. This file has a different syntax from FreeBSD's `host.conf`, so you will get an error message about `/etc/host.conf` if you leave this out.

You might also want to modify your `/etc/manpath.config` file to read the new man directory, and you may need to edit your `~/.cshrc` file to add `/usr/local/Mathematica/bin` to your path.

That is about all it takes. With this you should be able to type `mathematica` and get a really slick looking Mathematica Notebook screen up. Mathematica has included the Motif user interfaces, but it is compiled in statically, so you do not need the Motif libraries. Good luck doing this yourself!

## Bugs

The Notebook front end is known to hang sometimes when reading notebook files with an error messages similar to:

```
File ../Untitled-1.mb appears to be broken for OMPR.257.0
```

We have not found the cause for this, but it only affects the Notebook's X Window front end, not the `mathematica` engine itself. So the command line interface invoked by `'math'` is unaffected by this bug.

## Acknowledgments

A well-deserved thanks should go to Søren Schmidt <sos@FreeBSD.ORG> and Peter Wemm <peter@FreeBSD.ORG> who made linux emulation what it is today, and Michael Smith who drove these two guys like dogs to get it to the point where it runs Linux binaries better than linux! :-)

# Chapter 24. FreeBSD Internals

## The FreeBSD Booting Process

*Contributed by Poul-Henning Kamp <phk@FreeBSD.ORG>. v1.1, April 26th.*

Booting FreeBSD is essentially a three step process: load the kernel, determine the root filesystem and initialize user-land things. This leads to some interesting possibilities shown below.

### Loading a kernel

We presently have three basic mechanisms for loading the kernel as described below: they all pass some information to the kernel to help the kernel decide what to do next.

#### Biosboot

Biosboot is our “bootblocks”. It consists of two files which will be installed in the first 8Kbytes of the floppy or hard-disk slice to be booted from.

Biosboot can load a kernel from a FreeBSD filesystem.

#### Dosboot

Dosboot was written by DI. Christian Gusenbauer, and is unfortunately at this time one of the few pieces of code that will not compile under FreeBSD itself because it is written for Microsoft compilers.

Dosboot will boot the kernel from a MS-DOS file or from a FreeBSD filesystem partition on the disk. It attempts to negotiate with the various and strange kinds of memory manglers that lurk in high memory on MS/DOS systems and usually wins them for its case.

#### Netboot

Netboot will try to find a supported Ethernet card, and use BOOTP, TFTP and NFS to find a kernel file to boot.

## Determine the root filesystem

Once the kernel is loaded and the boot-code jumps to it, the kernel will initialize itself, trying to determine what hardware is present and so on; it then needs to find a root filesystem.

Presently we support the following types of root filesystems:

### UFS

This is the most normal type of root filesystem. It can reside on a floppy or on hard disk.

### MSDOS

While this is technically possible, it is not particular useful because of the FAT filesystem's inability to deal with links, device nodes and other such "UNIXisms".

### MFS

This is actually a UFS filesystem which has been compiled into the kernel. That means that the kernel does not really need any hard disks, floppies or other hardware to function.

### CD9660

This is for using a CD-ROM as root filesystem.

### NFS

This is for using a fileserver as root filesystem, basically making it a diskless machine.

## Initialize user-land things

To get the user-land going, the kernel, when it has finished initialization, will create a process with `pid == 1` and execute a program on the root filesystem; this program is normally `/sbin/init`.

You can substitute any program for `/sbin/init`, as long as you keep in mind that:

there is no `stdin/out/err` unless you open it yourself. If you exit, the machine panics. Signal handling is special for `pid == 1`.

An example of this is the `/stand/sysinstall` program on the installation floppy.

## Interesting combinations

Boot a kernel with a MFS in it with a special `/sbin/init` which...

### A — Using DOS

- mounts your `C:` as `/C:`
- Attaches `C:/freebsd.fs` on `/dev/vn0`
- mounts `/dev/vn0` as `/rootfs`
- makes symlinks `/rootfs/bin -> /bin` `/rootfs/etc -> /etc` `/rootfs/sbin -> /sbin` (etc...)

Now you are running FreeBSD without repartitioning your hard disk...

### B — Using NFS

NFS mounts your `server:~you/FreeBSD` as `/nfs`, chroots to `/nfs` and executes `/sbin/init` there

Now you are running FreeBSD diskless, even though you do not control the NFS server...

### C — Start an X-server

Now you have an X-terminal, which is better than that dingy

X-under-windows-so-slow-you-can-see-what-it-does thing that your boss insist is better than forking out money on hardware.

### D — Using a tape

Takes a copy of `/dev/rwd0` and writes it to a remote tape station or fileserver.

Now you finally get that backup you should have made a year ago...

### E — Acts as a firewall/web-server/what do I know...

This is particularly interesting since you can boot from a write-protected floppy, but still write to your root filesystem...

## PC Memory Utilization

*Contributed by Jörg Wunsch <joerg@FreeBSD.ORG>. 16 Apr 1995.*

*A short description of how FreeBSD uses memory on the i386 platform*

The boot sector will be loaded at `0:0x7c00`, and relocates itself immediately to `0x7c0:0`. (This is nothing magic, just an adjustment for the `%cs` selector, done by an `ljmp`.)

It then loads the first 15 sectors at `0x10000` (segment `BOOTSEG` in the biosboot Makefile), and sets up the stack to work below `0x1fff0`. After this, it jumps to the entry of `boot2` within that code. I.e., it jumps over itself and the (dummy) partition table, and it is going to adjust the `%cs` selector—we are still in 16-bit mode there.

`boot2` asks for the boot file, and examines the `a.out` header. It masks the file entry point (usually `0xf0100000`) by `0x00ffffff`, and loads the file there. Hence the usual load point is 1 MB (`0x00100000`). During load, the boot code toggles back and forth between real and protected mode, to use the BIOS in real mode.

The boot code itself uses segment selectors `0x18` and `0x20` for `%cs` and `%ds/%es` in protected mode, and `0x28` to jump back into real mode. The kernel is finally started with `%cs 0x08` and `%ds/%es/%ss 0x10`, which refer to dummy descriptors covering the entire address space.

The kernel will be started at its load point. Since it has been linked for another (high) address, it will have to execute PIC until the page table and page directory stuff is setup properly, at which point paging will be enabled and the kernel will finally run at the address for which it was linked.

*Contributed by David Greenman <dg@FreeBSD.ORG>. 16 Apr 1995.*

The physical pages immediately following the kernel BSS contain `proc0`'s page directory, page tables, and upages. Some time later when the VM system is initialized, the physical memory between `0x1000-0x9ffff` and the physical memory after the kernel (`text+data+bss+proc0` stuff+other misc) is made available in the form of general VM pages and added to the global free page list.

## DMA: What it Is and How it Works

*Copyright © 1995,1997 Frank Durda IV <uhclem@FreeBSD.ORG>, All Rights Reserved. 10 December 1996. Last Update 8 October 1997.*

Direct Memory Access (DMA) is a method of allowing data to be moved from one location to another in a computer without intervention from the central processor (CPU).

The way that the DMA function is implemented varies between computer architectures, so this discussion will limit itself to the implementation and workings of the DMA subsystem on the IBM Personal Computer (PC), the IBM PC/AT and all of its successors and clones.

The PC DMA subsystem is based on the Intel 8237 DMA controller. The 8237 contains four DMA channels that can be programmed independently and any one of the channels may be active at any moment. These channels are numbered 0, 1, 2 and 3. Starting with the PC/AT, IBM added a second 8237 chip, and numbered those channels 4, 5, 6 and 7.

The original DMA controller (0, 1, 2 and 3) moves one byte in each transfer. The second DMA controller (4, 5, 6, and 7) moves 16-bits from two adjacent memory locations in each transfer, with the first byte always coming from an even-numbered address. The two controllers are identical components and the difference in transfer size is caused by the way the second controller is wired into the system.

The 8237 has two electrical signals for each channel, named DRQ and -DACK. There are additional signals with the names HRQ (Hold Request), HLDA (Hold Acknowledge), -EOP (End of Process), and the bus control signals -MEMR (Memory Read), -MEMW (Memory Write), -IOR (I/O Read), and -IOW (I/O Write).

The 8237 DMA is known as a “fly-by” DMA controller. This means that the data being moved from one location to another does not pass through the DMA chip and is not stored in the DMA chip. Subsequently, the DMA can only transfer data between an I/O port and a memory address, but not between two I/O ports or two memory locations.

**Note:** The 8237 does allow two channels to be connected together to allow memory-to-memory DMA operations in a non-“fly-by” mode, but nobody in the PC industry uses this scarce resource this way since it is faster to move data between memory locations using the CPU.

In the PC architecture, each DMA channel is normally activated only when the hardware that uses a given DMA channel requests a transfer by asserting the DRQ line for that channel.

## A Sample DMA transfer

Here is an example of the steps that occur to cause and perform a DMA transfer. In this example, the floppy disk controller (FDC) has just read a byte from a diskette and wants the DMA to place it in memory at location 0x00123456. The process begins by the FDC asserting the DRQ2 signal (the DRQ line for DMA channel 2) to alert the DMA controller.

The DMA controller will note that the DRQ2 signal is asserted. The DMA controller will then make sure that DMA channel 2 has been programmed and is unmasked (enabled). The DMA controller also makes sure that none of the other DMA channels are active or want to be active and have a higher priority. Once these checks are complete, the DMA asks the CPU to release the bus so that the DMA may use the bus. The DMA requests the bus by asserting the HRQ signal which goes to the CPU.

The CPU detects the HRQ signal, and will complete executing the current instruction. Once the processor has reached a state where it can release the bus, it will. Now all of the signals normally generated by the CPU (-MEMR, -MEMW, -IOR, -IOW and a few others) are placed in a tri-stated

condition (neither high or low) and then the CPU asserts the HLDA signal which tells the DMA controller that it is now in charge of the bus.

Depending on the processor, the CPU may be able to execute a few additional instructions now that it no longer has the bus, but the CPU will eventually have to wait when it reaches an instruction that must read something from memory that is not in the internal processor cache or pipeline.

Now that the DMA “is in charge”, the DMA activates its -MEMR, -MEMW, -IOR, -IOW output signals, and the address outputs from the DMA are set to 0x3456, which will be used to direct the byte that is about to be transferred to a specific memory location.

The DMA will then let the device that requested the DMA transfer know that the transfer is commencing. This is done by asserting the -DACK signal, or in the case of the floppy disk controller, -DACK2 is asserted.

The floppy disk controller is now responsible for placing the byte to be transferred on the bus Data lines. Unless the floppy controller needs more time to get the data byte on the bus (and if the peripheral does need more time it alerts the DMA via the READY signal), the DMA will wait one DMA clock, and then de-assert the -MEMW and -IOR signals so that the memory will latch and store the byte that was on the bus, and the FDC will know that the byte has been transferred.

Since the DMA cycle only transfers a single byte at a time, the FDC now drops the DRQ2 signal, so the DMA knows that it is no longer needed. The DMA will de-assert the -DACK2 signal, so that the FDC knows it must stop placing data on the bus.

The DMA will now check to see if any of the other DMA channels have any work to do. If none of the channels have their DRQ lines asserted, the DMA controller has completed its work and will now tri-state the -MEMR, -MEMW, -IOR, -IOW and address signals.

Finally, the DMA will de-assert the HRQ signal. The CPU sees this, and de-asserts the HOLDA signal. Now the CPU activates its -MEMR, -MEMW, -IOR, -IOW and address lines, and it resumes executing instructions and accessing main memory and the peripherals.

For a typical floppy disk sector, the above process is repeated 512 times, once for each byte. Each time a byte is transferred, the address register in the DMA is incremented and the counter in the DMA that shows how many bytes are to be transferred is decremented.

When the counter reaches zero, the DMA asserts the EOP signal, which indicates that the counter has reached zero and no more data will be transferred until the DMA controller is reprogrammed by the CPU. This event is also called the Terminal Count (TC). There is only one EOP signal, and since only one DMA channel can be active at any instant, the DMA channel that is currently active must be the DMA channel that just completed its task.

If a peripheral wants to generate an interrupt when the transfer of a buffer is complete, it can test for its -DACKn signal and the EOP signal both being asserted at the same time. When that happens, it means the DMA will not transfer any more information for that peripheral without intervention by the CPU. The peripheral can then assert one of the interrupt signals to get the processors’ attention. In the PC

architecture, the DMA chip itself is not capable of generating an interrupt. The peripheral and its associated hardware is responsible for generating any interrupt that occurs. Subsequently, it is possible to have a peripheral that uses DMA but does not use interrupts.

It is important to understand that although the CPU always releases the bus to the DMA when the DMA makes the request, this action is invisible to both applications and the operating systems, except for slight changes in the amount of time the processor takes to execute instructions when the DMA is active. Subsequently, the processor must poll the peripheral, poll the registers in the DMA chip, or receive an interrupt from the peripheral to know for certain when a DMA transfer has completed.

## DMA Page Registers and 16Meg address space limitations

You may have noticed earlier that instead of the DMA setting the address lines to 0x00123456 as we said earlier, the DMA only set 0x3456. The reason for this takes a bit of explaining.

When the original IBM PC was designed, IBM elected to use both DMA and interrupt controller chips that were designed for use with the 8085, an 8-bit processor with an address space of 16 bits (64K). Since the IBM PC supported more than 64K of memory, something had to be done to allow the DMA to read or write memory locations above the 64K mark. What IBM did to solve this problem was to add an external data latch for each DMA channel that holds the upper bits of the address to be read to or written from. Whenever a DMA channel is active, the contents of that latch are written to the address bus and kept there until the DMA operation for the channel ends. IBM called these latches “Page Registers”.

So for our example above, the DMA would put the 0x3456 part of the address on the bus, and the Page Register for DMA channel 2 would put 0x0012xxxx on the bus. Together, these two values form the complete address in memory that is to be accessed.

Because the Page Register latch is independent of the DMA chip, the area of memory to be read or written must not span a 64K physical boundary. For example, if the DMA accesses memory location 0xffff, after that transfer the DMA will then increment the address register and the DMA will access the next byte at location 0x0000, not 0x10000. The results of letting this happen are probably not intended.

**Note:** “Physical” 64K boundaries should not be confused with 8086-mode 64K “Segments”, which are created by mathematically adding a segment register with an offset register. Page Registers have no address overlap and are mathematically OR-ed together.

To further complicate matters, the external DMA address latches on the PC/AT hold only eight bits, so that gives us 8+16=24 bits, which means that the DMA can only point at memory locations between 0 and 16Meg. For newer computers that allow more than 16Meg of memory, the standard PC-compatible DMA cannot access memory locations above 16Meg.

To get around this restriction, operating systems will reserve a RAM buffer in an area below 16Meg that

also does not span a physical 64K boundary. Then the DMA will be programmed to transfer data from the peripheral and into that buffer. Once the DMA has moved the data into this buffer, the operating system will then copy the data from the buffer to the address where the data is really supposed to be stored.

When writing data from an address above 16Meg to a DMA-based peripheral, the data must be first copied from where it resides into a buffer located below 16Meg, and then the DMA can copy the data from the buffer to the hardware. In FreeBSD, these reserved buffers are called “Bounce Buffers”. In the MS-DOS world, they are sometimes called “Smart Buffers”.

**Note:** A new implementation of the 8237, called the 82374, allows 16 bits of page register to be specified, allows access to the entire 32 bit address space, without the use of bounce buffers.

## DMA Operational Modes and Settings

The 8237 DMA can be operated in several modes. The main ones are:

### Single

A single byte (or word) is transferred. The DMA must release and re-acquire the bus for each additional byte. This is commonly-used by devices that cannot transfer the entire block of data immediately. The peripheral will request the DMA each time it is ready for another transfer.

The standard PC-compatible floppy disk controller (NEC 765) only has a one-byte buffer, so it uses this mode.

### Block/Demand

Once the DMA acquires the system bus, an entire block of data is transferred, up to a maximum of 64K. If the peripheral needs additional time, it can assert the READY signal to suspend the transfer briefly. READY should not be used excessively, and for slow peripheral transfers, the Single Transfer Mode should be used instead.

The difference between Block and Demand is that once a Block transfer is started, it runs until the transfer count reaches zero. DRQ only needs to be asserted until -DACK is asserted. Demand Mode will transfer one more bytes until DRQ is de-asserted, at which point the DMA suspends the transfer and releases the bus back to the CPU. When DRQ is asserted later, the transfer resumes where it was suspended.

Older hard disk controllers used Demand Mode until CPU speeds increased to the point that it was more efficient to transfer the data using the CPU, particularly if the memory locations used in the

transfer were above the 16Meg mark.

### Cascade

This mechanism allows a DMA channel to request the bus, but then the attached peripheral device is responsible for placing the addressing information on the bus instead of the DMA. This is also used to implement a technique known as “Bus Mastering”.

When a DMA channel in Cascade Mode receives control of the bus, the DMA does not place addresses and I/O control signals on the bus like the DMA normally does when it is active. Instead, the DMA only asserts the -DACK signal for the active DMA channel.

At this point it is up to the peripheral connected to that DMA channel to provide address and bus control signals. The peripheral has complete control over the system bus, and can do reads and/or writes to any address below 16Meg. When the peripheral is finished with the bus, it de-asserts the DRQ line, and the DMA controller can then return control to the CPU or to some other DMA channel.

Cascade Mode can be used to chain multiple DMA controllers together, and this is exactly what DMA Channel 4 is used for in the PC architecture. When a peripheral requests the bus on DMA channels 0, 1, 2 or 3, the slave DMA controller asserts HLDREQ, but this wire is actually connected to DRQ4 on the primary DMA controller instead of to the CPU. The primary DMA controller, thinking it has work to do on Channel 4, requests the bus from the CPU using HLDREQ signal. Once the CPU grants the bus to the primary DMA controller, -DACK4 is asserted, and that wire is actually connected to the HLDA signal on the slave DMA controller. The slave DMA controller then transfers data for the DMA channel that requested it (0, 1, 2 or 3), or the slave DMA may grant the bus to a peripheral that wants to perform its own bus-mastering, such as a SCSI controller.

Because of this wiring arrangement, only DMA channels 0, 1, 2, 3, 5, 6 and 7 are usable with peripherals on PC/AT systems.

**Note:** DMA channel 0 was reserved for refresh operations in early IBM PC computers, but is generally available for use by peripherals in modern systems.

When a peripheral is performing Bus Mastering, it is important that the peripheral transmit data to or from memory constantly while it holds the system bus. If the peripheral cannot do this, it must release the bus frequently so that the system can perform refresh operations on main memory.

The Dynamic RAM used in all PCs for main memory must be accessed frequently to keep the bits stored in the components “charged”. Dynamic RAM essentially consists of millions of capacitors with each one holding one bit of data. These capacitors are charged with power to represent a 1 or drained to represent a 0. Because all capacitors leak, power must be added at regular intervals to keep the 1 values intact. The RAM chips actually handle the task of pumping power back into all of

the appropriate locations in RAM, but they must be told when to do it by the rest of the computer so that the refresh activity won't interfere with the computer wanting to access RAM normally. If the computer is unable to refresh memory, the contents of memory will become corrupted in just a few milliseconds.

Since memory read and write cycles “count” as refresh cycles (a dynamic RAM refresh cycle is actually an incomplete memory read cycle), as long as the peripheral controller continues reading or writing data to sequential memory locations, that action will refresh all of memory.

Bus-mastering is found in some SCSI host interfaces and other high-performance peripheral controllers.

#### Autoinitialize

This mode causes the DMA to perform Byte, Block or Demand transfers, but when the DMA transfer counter reaches zero, the counter and address are set back to where they were when the DMA channel was originally programmed. This means that as long as the peripheral requests transfers, they will be granted. It is up to the CPU to move new data into the fixed buffer ahead of where the DMA is about to transfer it when doing output operations, and read new data out of the buffer behind where the DMA is writing when doing input operations.

This technique is frequently used on audio devices that have small or no hardware “sample” buffers. There is additional CPU overhead to manage this “circular” buffer, but in some cases this may be the only way to eliminate the latency that occurs when the DMA counter reaches zero and the DMA stops transfers until it is reprogrammed.

## Programming the DMA

The DMA channel that is to be programmed should always be “masked” before loading any settings. This is because the hardware might unexpectedly assert the DRQ for that channel, and the DMA might respond, even though not all of the parameters have been loaded or updated.

Once masked, the host must specify the direction of the transfer (memory-to-I/O or I/O-to-memory), what mode of DMA operation is to be used for the transfer (Single, Block, Demand, Cascade, etc), and finally the address and length of the transfer are loaded. The length that is loaded is one less than the amount you expect the DMA to transfer. The LSB and MSB of the address and length are written to the same 8-bit I/O port, so another port must be written to first to guarantee that the DMA accepts the first byte as the LSB and the second byte as the MSB of the length and address.

Then, be sure to update the Page Register, which is external to the DMA and is accessed through a different set of I/O ports.

Once all the settings are ready, the DMA channel can be un-masked. That DMA channel is now considered to be “armed”, and will respond when the DRQ line for that channel is asserted.

Refer to a hardware data book for precise programming details for the 8237. You will also need to refer to the I/O port map for the PC system, which describes where the DMA and Page Register ports are located. A complete port map table is located below.

## DMA Port Map

All systems based on the IBM-PC and PC/AT have the DMA hardware located at the same I/O ports. The complete list is provided below. Ports assigned to DMA Controller #2 are undefined on non-AT designs.

### 0x00–0x1f DMA Controller #1 (Channels 0, 1, 2 and 3)

#### DMA Address and Count Registers

0x00	write	Channel 0 starting address
0x00	read	Channel 0 current address
0x01	write	Channel 0 starting word count
0x01	read	Channel 0 remaining word count
0x02	write	Channel 1 starting address
0x02	read	Channel 1 current address
0x03	write	Channel 1 starting word count
0x03	read	Channel 1 remaining word count
0x04	write	Channel 2 starting address
0x04	read	Channel 2 current address
0x05	write	Channel 2 starting word count
0x05	read	Channel 2 remaining word count
0x06	write	Channel 3 starting address
0x06	read	Channel 3 current address
0x07	write	Channel 3 starting word count
0x07	read	Channel 3 remaining word count

#### DMA Command Registers

0x08	write	Command Register
------	-------	------------------

0x08	read	Status Register
0x09	write	Request Register
0x09	read	-
0x0a	write	Single Mask Register Bit
0x0a	read	-
0x0b	write	Mode Register
0x0b	read	-
0x0c	write	Clear LSB/MSB Flip-Flop
0x0c	read	-
0x0d	write	Master Clear/Reset
0x0d	read	Temporary Register (not available on newer versions)
0x0e	write	Clear Mask Register
0x0e	read	-
0x0f	write	Write All Mask Register Bits
0x0f	read	Read All Mask Register Bits (only in Intel 82374)

### 0xc0–0xdf DMA Controller #2 (Channels 4, 5, 6 and 7)

#### DMA Address and Count Registers

0xc0	write	Channel 4 starting address
0xc0	read	Channel 4 current address
0xc2	write	Channel 4 starting word count
0xc2	read	Channel 4 remaining word count
0xc4	write	Channel 5 starting address
0xc4	read	Channel 5 current address
0xc6	write	Channel 5 starting word count
0xc6	read	Channel 5 remaining word count
0xc8	write	Channel 6 starting address
0xc8	read	Channel 6 current address
0xca	write	Channel 6 starting word count

0xca	read	Channel 6 remaining word count
0xcc	write	Channel 7 starting address
0xcc	read	Channel 7 current address
0xce	write	Channel 7 starting word count
0xce	read	Channel 7 remaining word count

**DMA Command Registers**

0xd0	write	Command Register
0xd0	read	Status Register
0xd2	write	Request Register
0xd2	read	-
0xd4	write	Single Mask Register Bit
0xd4	read	-
0xd6	write	Mode Register
0xd6	read	-
0xd8	write	Clear LSB/MSB Flip-Flop
0xd8	read	-
0xda	write	Master Clear/Reset
0xda	read	Temporary Register (not present in Intel 82374)
0xdc	write	Clear Mask Register
0xdc	read	-
0xde	write	Write All Mask Register Bits
0xdf	read	Read All Mask Register Bits (only in Intel 82374)

**0x80–0x9f DMA Page Registers**

0x87	r/w	Channel 0 Low byte (23-16) page Register
0x83	r/w	Channel 1 Low byte (23-16) page Register

0x81	r/w	Channel 2 Low byte (23-16) page Register
0x82	r/w	Channel 3 Low byte (23-16) page Register
0x8b	r/w	Channel 5 Low byte (23-16) page Register
0x89	r/w	Channel 6 Low byte (23-16) page Register
0x8a	r/w	Channel 7 Low byte (23-16) page Register
0x8f	r/w	Low byte page Refresh

### 0x400–0x4ff 82374 Enhanced DMA Registers

The Intel 82374 EISA System Component (ESC) was introduced in early 1996 and includes a DMA controller that provides a superset of 8237 functionality as well as other PC-compatible core peripheral components in a single package. This chip is targeted at both EISA and PCI platforms, and provides modern DMA features like scatter-gather, ring buffers as well as direct access by the system DMA to all 32 bits of address space.

If these features are used, code should also be included to provide similar functionality in the previous 16 years worth of PC-compatible computers. For compatibility reasons, some of the 82374 registers must be programmed *after* programming the traditional 8237 registers for each transfer. Writing to a traditional 8237 register forces the contents of some of the 82374 enhanced registers to zero to provide backward software compatibility.

0x401	r/w	Channel 0 High byte (bits 23-16) word count
0x403	r/w	Channel 1 High byte (bits 23-16) word count
0x405	r/w	Channel 2 High byte (bits 23-16) word count
0x407	r/w	Channel 3 High byte (bits 23-16) word count
0x4c6	r/w	Channel 5 High byte (bits 23-16) word count

0x4ca	r/w	Channel 6 High byte (bits 23-16) word count
0x4ce	r/w	Channel 7 High byte (bits 23-16) word count
0x487	r/w	Channel 0 High byte (bits 31-24) page Register
0x483	r/w	Channel 1 High byte (bits 31-24) page Register
0x481	r/w	Channel 2 High byte (bits 31-24) page Register
0x482	r/w	Channel 3 High byte (bits 31-24) page Register
0x48b	r/w	Channel 5 High byte (bits 31-24) page Register
0x489	r/w	Channel 6 High byte (bits 31-24) page Register
0x48a	r/w	Channel 6 High byte (bits 31-24) page Register
0x48f	r/w	High byte page Refresh
0x4e0	r/w	Channel 0 Stop Register (bits 7-2)
0x4e1	r/w	Channel 0 Stop Register (bits 15-8)
0x4e2	r/w	Channel 0 Stop Register (bits 23-16)
0x4e4	r/w	Channel 1 Stop Register (bits 7-2)
0x4e5	r/w	Channel 1 Stop Register (bits 15-8)
0x4e6	r/w	Channel 1 Stop Register (bits 23-16)
0x4e8	r/w	Channel 2 Stop Register (bits 7-2)
0x4e9	r/w	Channel 2 Stop Register (bits 15-8)

0x4ea	r/w	Channel 2 Stop Register (bits 23-16)
0x4ec	r/w	Channel 3 Stop Register (bits 7-2)
0x4ed	r/w	Channel 3 Stop Register (bits 15-8)
0x4ee	r/w	Channel 3 Stop Register (bits 23-16)
0x4f4	r/w	Channel 5 Stop Register (bits 7-2)
0x4f5	r/w	Channel 5 Stop Register (bits 15-8)
0x4f6	r/w	Channel 5 Stop Register (bits 23-16)
0x4f8	r/w	Channel 6 Stop Register (bits 7-2)
0x4f9	r/w	Channel 6 Stop Register (bits 15-8)
0x4fa	r/w	Channel 6 Stop Register (bits 23-16)
0x4fc	r/w	Channel 7 Stop Register (bits 7-2)
0x4fd	r/w	Channel 7 Stop Register (bits 15-8)
0x4fe	r/w	Channel 7 Stop Register (bits 23-16)
0x40a	write	Channels 0-3 Chaining Mode Register
0x40a	read	Channel Interrupt Status Register
0x4d4	write	Channels 4-7 Chaining Mode Register
0x4d4	read	Chaining Mode Status
0x40c	read	Chain Buffer Expiration Control Register
0x410	write	Channel 0 Scatter-Gather Command Register

0x411	write	Channel 1 Scatter-Gather Command Register
0x412	write	Channel 2 Scatter-Gather Command Register
0x413	write	Channel 3 Scatter-Gather Command Register
0x415	write	Channel 5 Scatter-Gather Command Register
0x416	write	Channel 6 Scatter-Gather Command Register
0x417	write	Channel 7 Scatter-Gather Command Register
0x418	read	Channel 0 Scatter-Gather Status Register
0x419	read	Channel 1 Scatter-Gather Status Register
0x41a	read	Channel 2 Scatter-Gather Status Register
0x41b	read	Channel 3 Scatter-Gather Status Register
0x41d	read	Channel 5 Scatter-Gather Status Register
0x41e	read	Channel 5 Scatter-Gather Status Register
0x41f	read	Channel 7 Scatter-Gather Status Register
0x420-0x423	r/w	Channel 0 Scatter-Gather Descriptor Table Pointer Register
0x424-0x427	r/w	Channel 1 Scatter-Gather Descriptor Table Pointer Register
0x428-0x42b	r/w	Channel 2 Scatter-Gather Descriptor Table Pointer Register
0x42c-0x42f	r/w	Channel 3 Scatter-Gather Descriptor Table Pointer Register
0x434-0x437	r/w	Channel 5 Scatter-Gather Descriptor Table Pointer Register

0x438-0x43b	r/w	Channel 6 Scatter-Gather Descriptor Table Pointer Register
0x43c-0x43f	r/w	Channel 7 Scatter-Gather Descriptor Table Pointer Register

## The FreeBSD VM System

*Contributed by Matthew Dillon <dillon@FreeBSD.ORG>. 6 Feb 1999*

### Management of physical memory—`vm_page_t`

Physical memory is managed on a page-by-page basis through the `vm_page_t` structure. Pages of physical memory are categorized through the placement of their respective `vm_page_t` structures on one of several paging queues.

A page can be in a wired, active, inactive, cache, or free state. Except for the wired state, the page is typically placed in a doubly link list queue representing the state that it is in. Wired pages are not placed on any queue.

FreeBSD implements a more involved paging queue for cached and free pages in order to implement page coloring. Each of these states involves multiple queues arranged according to the size of the processor's L1 and L2 caches. When a new page needs to be allocated, FreeBSD attempts to obtain one that is reasonably well aligned from the point of view of the L1 and L2 caches relative to the VM object the page is being allocated for.

Additionally, a page may be held with a reference count or locked with a busy count. The VM system also implements an “ultimate locked” state for a page using the `PG_BUSY` bit in the page's flags.

In general terms, each of the paging queues operates in a LRU fashion. A page is typically placed in a wired or active state initially. When wired, the page is usually associated with a page table somewhere. The VM system ages the page by scanning pages in a more active paging queue (LRU) in order to move them to a less-active paging queue. Pages that get moved into the cache are still associated with a VM object but are candidates for immediate reuse. Pages in the free queue are truly free. FreeBSD attempts to minimize the number of pages in the free queue, but a certain minimum number of truly free pages must be maintained in order to accommodate page allocation at interrupt time.

If a process attempts to access a page that does not exist in its page table but does exist in one of the paging queues (such as the inactive or cache queues), a relatively inexpensive page reactivation fault

occurs which causes the page to be reactivated. If the page does not exist in system memory at all, the process must block while the page is brought in from disk.

FreeBSD dynamically tunes its paging queues and attempts to maintain reasonable ratios of pages in the various queues as well as attempts to maintain a reasonable breakdown of clean vs dirty pages. The amount of rebalancing that occurs depends on the system's memory load. This rebalancing is implemented by the pageout daemon and involves laundering dirty pages (syncing them with their backing store), noticing when pages are activity referenced (resetting their position in the LRU queues or moving them between queues), migrating pages between queues when the queues are out of balance, and so forth. FreeBSD's VM system is willing to take a reasonable number of reactivation page faults to determine how active or how idle a page actually is. This leads to better decisions being made as to when to launder or swap-out a page.

## The unified buffer cache—`vm_object_t`

FreeBSD implements the idea of a generic “VM object”. VM objects can be associated with backing store of various types—unbacked, swap-backed, physical device-backed, or file-backed storage. Since the filesystem uses the same VM objects to manage in-core data relating to files, the result is a unified buffer cache.

VM objects can be *shadowed*. That is, they can be stacked on top of each other. For example, you might have a swap-backed VM object stacked on top of a file-backed VM object in order to implement a `MAP_PRIVATE` `mmap()`ing. This stacking is also used to implement various sharing properties, including, copy-on-write, for forked address spaces.

It should be noted that a `vm_page_t` can only be associated with one VM object at a time. The VM object shadowing implements the perceived sharing of the same page across multiple instances.

## Filesystem I/O—`struct buf`

vnode-backed VM objects, such as file-backed objects, generally need to maintain their own clean/dirty info independent from the VM system's idea of clean/dirty. For example, when the VM system decides to synchronize a physical page to its backing store, the VM system needs to mark the page clean before the page is actually written to its backing store. Additionally, filesystems need to be able to map portions of a file or file metadata into KVM in order to operate on it.

The entities used to manage this are known as filesystem buffers, `struct buf`'s, and also known as `bp`'s. When a filesystem needs to operate on a portion of a VM object, it typically maps part of the object into a `struct buf` and then maps the pages in the `struct buf` into KVM. In the same manner, disk I/O is typically issued by mapping portions of objects into buffer structures and then issuing the I/O on the buffer structures. The underlying `vm_page_t`'s are typically busied for the duration of the I/O. Filesystem

buffers also have their own notion of being busy, which is useful to filesystem driver code which would rather operate on filesystem buffers instead of hard VM pages.

FreeBSD reserves a limited amount of KVM to hold mappings from struct bufs, but it should be made clear that this KVM is used solely to hold mappings and does not limit the ability to cache data. Physical data caching is strictly a function of `vm_page_t`'s, not filesystem buffers. However, since filesystem buffers are used placeholder I/O, they do inherently limit the amount of concurrent I/O possible. As there are usually a few thousand filesystem buffers available, this is not usually a problem.

## Mapping Page Tables - `vm_map_t`, `vm_entry_t`

FreeBSD separates the physical page table topology from the VM system. All hard per-process page tables can be reconstructed on the fly and are usually considered throwaway. Special page tables such as those managing KVM are typically permanently preallocated. These page tables are not throwaway.

FreeBSD associates portions of `vm_objects` with address ranges in virtual memory through `vm_map_t` and `vm_entry_t` structures. Page tables are directly synthesized from the `vm_map_t/vm_entry_t/vm_object_t` hierarchy. Remember when I mentioned that physical pages are only directly associated with a `vm_object`. Well, that isn't quite true. `vm_page_t`'s are also linked into page tables that they are actively associated with. One `vm_page_t` can be linked into several *pmaps*, as page tables are called. However, the hierarchical association holds so all references to the same page in the same object reference the same `vm_page_t` and thus give us buffer cache unification across the board.

## KVM Memory Mapping

FreeBSD uses KVM to hold various kernel structures. The single largest entity held in KVM is the filesystem buffer cache. That is, mappings relating to `struct buf` entities.

Unlike Linux, FreeBSD does NOT map all of physical memory into KVM. This means that FreeBSD can handle memory configurations up to 4G on 32 bit platforms. In fact, if the mmu were capable of it, FreeBSD could theoretically handle memory configurations up to 8TB on a 32 bit platform. However, since most 32 bit platforms are only capable of mapping 4GB of ram, this is a moot point.

KVM is managed through several mechanisms. The main mechanism used to manage KVM is the *zone allocator*. The zone allocator takes a chunk of KVM and splits it up into constant-sized blocks of memory in order to allocate a specific type of structure. You can use `vmstat -m` to get an overview of current KVM utilization broken down by zone.

## Tuning the FreeBSD VM system

A concerted effort has been made to make the FreeBSD kernel dynamically tune itself. Typically you do not need to mess with anything beyond the `maxusers` and `NMBCLUSTERS` kernel config options. That is, kernel compilation options specified in (typically) `/usr/src/sys/i386/conf/CONFIG_FILE`. A description of all available kernel configuration options can be found in `/usr/src/sys/i386/conf/LINT`.

In a large system configuration you may wish to increase `maxusers`. Values typically range from 10 to 128. Note that raising `maxusers` too high can cause the system to overflow available KVM resulting in unpredictable operation. It is better to leave `maxusers` at some reasonable number of add other options, such as `NMBCLUSTERS`, to increase specific resources.

If your system is going to use the network heavily, you may want to increase `NMBCLUSTERS`. Typical values range from 1024 to 4096.

The `NBUF` parameter is also traditionally used to scale the system. This parameter determines the amount of KVA the system can use to map filesystem buffers for I/O. Note that this parameter has nothing whatsoever to do with the unified buffer cache! This parameter is dynamically tuned in 3.0-CURRENT and later kernels and should generally not be adjusted manually. We recommend that you *not* try to specify an `NBUF` parameter. Let the system pick it. Too small a value can result in extremely inefficient filesystem operation while too large a value can starve the page queues by causing too many pages to become wired down.

By default, FreeBSD kernels are not optimized. You can set debugging and optimization flags with the `makeoptions` directive in the kernel configuration. Note that you should not use `-g` unless you can accomodate the large (typically 7 MB+) kernels that result.

```
makeoptions    DEBUG="-g"
makeoptions    COPTFLAGS="-O2 -pipe"
```

`Sysctl` provides a way to tune kernel parameters at run-time. You typically do not need to mess with any of the `sysctl` variables, especially the VM related ones.

Run time VM and system tuning is relatively straightforward. First, use `softupdates` on your UFS/FFS filesystems whenever possible. `/usr/src/contrib/sys/softupdates/README` contains instructions (and restrictions) on how to configure it up.

Second, configure sufficient swap. You should have a swap partition configured on each physical disk, up to four, even on your “work” disks. You should have at least 2x the swap space as you have main memory, and possibly even more if you do not have a lot of memory. You should also size your swap partition based on the maximum memory configuration you ever intend to put on the machine so you do not have to repartition your disks later on. If you want to be able to accomodate a crash dump, your first swap partition must be at least as large as main memory and `/var/crash` must have sufficient free space to hold the dump.

NFS-based swap is perfectly acceptable on -4.x or later systems, but you must be aware that the NFS server will take the brunt of the paging load.

# V. Appendices

# Chapter 25. Obtaining FreeBSD

## CD-ROM Publishers

FreeBSD is available on CD-ROM from Walnut Creek CDROM:

Walnut Creek CDROM  
4041 Pike Lane, Suite F  
Concord  
CA, 94520  
USA  
Phone: +1 925 674-0783  
Fax: +1 925 674-0821  
Email: <info@cdrom.com>  
WWW: <http://www.cdrom.com/>

## FTP Sites

The official sources for FreeBSD are available via anonymous FTP from:

<ftp://ftp.FreeBSD.ORG/pub/FreeBSD>.

The FreeBSD mirror sites database (<http://www.itworks.com.au/~gavin/FBSDsites.php3>) is more accurate than the mirror listing in the handbook, as it gets its information from the DNS rather than relying on static lists of hosts.

Additionally, FreeBSD is available via anonymous FTP from the following mirror sites. If you choose to obtain FreeBSD via anonymous FTP, please try to use a site near you.

Argentina, Australia, Brazil, Canada, Czech Republic, Denmark, Estonia, Finland, France, Germany, Hong Kong, Ireland, Israel, Japan, Korea, Netherlands, Poland, Portugal, Russia, South Africa, Spain, Slovak Republic, Slovenia, Sweden, Taiwan, Thailand, UK, Ukraine, USA.

### Argentina

In case of problems, please contact the hostmaster <hostmaster@ar.FreeBSD.ORG> for this domain.

- <ftp://ftp.ar.FreeBSD.ORG/pub/FreeBSD>

### Australia

In case of problems, please contact the hostmaster <hostmaster@au.FreeBSD.ORG> for this domain.

- <ftp://ftp.au.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp2.au.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp3.au.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp4.au.FreeBSD.ORG/pub/FreeBSD>

### Brazil

In case of problems, please contact the hostmaster <hostmaster@br.FreeBSD.ORG> for this domain.

- <ftp://ftp.br.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp2.br.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp3.br.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp4.br.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp5.br.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp6.br.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp7.br.FreeBSD.ORG/pub/FreeBSD>

### Canada

In case of problems, please contact the hostmaster <hostmaster@ca.FreeBSD.ORG> for this domain.

- <ftp://ftp.ca.FreeBSD.ORG/pub/FreeBSD>

### Czech Republic

- <ftp://sunsite.mff.cuni.cz/OS/FreeBSD> Contact: <jj@sunsite.mff.cuni.cz>.

### Denmark

In case of problems, please contact the hostmaster <hostmaster@dk.FreeBSD.ORG> for this domain.

- <ftp://ftp.dk.freebsd.ORG/pub/FreeBSD>

### Estonia

In case of problems, please contact the hostmaster <hostmaster@ee.FreeBSD.ORG> for this domain.

- <ftp://ftp.ee.freebsd.ORG/pub/FreeBSD>

### Finland

In case of problems, please contact the hostmaster <hostmaster@fi.FreeBSD.ORG> for this domain.

- <ftp://ftp.fi.freebsd.ORG/pub/FreeBSD>

### France

In case of problems, please contact the hostmaster <hostmaster@fr.FreeBSD.ORG> for this domain.

- <ftp://ftp.fr.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp2.fr.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp3.fr.FreeBSD.ORG/pub/FreeBSD>

### Germany

In case of problems, please contact the hostmaster <hostmaster@de.FreeBSD.ORG> for this domain.

- <ftp://ftp.de.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp2.de.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp3.de.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp4.de.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp5.de.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp6.de.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp7.de.FreeBSD.ORG/pub/FreeBSD>

#### Hong Kong

- <ftp://ftp.hk.super.net/pub/FreeBSD> Contact: <ftp-admin@HK.Super.NET>.

#### Ireland

In case of problems, please contact the hostmaster <hostmaster@ie.FreeBSD.ORG> for this domain.

- <ftp://ftp.ie.FreeBSD.ORG/pub/FreeBSD>

#### Israel

In case of problems, please contact the hostmaster <hostmaster@il.FreeBSD.ORG> for this domain.

- <ftp://ftp.il.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp2.il.FreeBSD.ORG/pub/FreeBSD>

#### Japan

In case of problems, please contact the hostmaster <hostmaster@jp.FreeBSD.ORG> for this domain.

- <ftp://ftp.jp.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp2.jp.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp3.jp.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp4.jp.FreeBSD.ORG/pub/FreeBSD>

- <ftp://ftp5.jp.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp6.jp.FreeBSD.ORG/pub/FreeBSD>

#### Korea

In case of problems, please contact the hostmaster <[hostmaster@kr.FreeBSD.ORG](mailto:hostmaster@kr.FreeBSD.ORG)> for this domain.

- <ftp://ftp.kr.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp2.kr.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp3.kr.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp4.kr.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp5.kr.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp6.kr.FreeBSD.ORG/pub/FreeBSD>

#### Netherlands

In case of problems, please contact the hostmaster <[hostmaster@nl.FreeBSD.ORG](mailto:hostmaster@nl.FreeBSD.ORG)> for this domain.

- <ftp://ftp.nl.freebsd.ORG/pub/FreeBSD>

#### Poland

In case of problems, please contact the hostmaster <[hostmaster@pl.FreeBSD.ORG](mailto:hostmaster@pl.FreeBSD.ORG)> for this domain.

- <ftp://ftp.pl.freebsd.ORG/pub/FreeBSD>

#### Portugal

In case of problems, please contact the hostmaster <[hostmaster@pt.FreeBSD.ORG](mailto:hostmaster@pt.FreeBSD.ORG)> for this domain.

- <ftp://ftp.pt.freebsd.org/pub/FreeBSD>
- <ftp://ftp2.pt.freebsd.org/pub/FreeBSD>

#### Russia

In case of problems, please contact the hostmaster <hostmaster@ru.FreeBSD.ORG> for this domain.

- <ftp://ftp.ru.freebsd.org/pub/FreeBSD>
- <ftp://ftp2.ru.freebsd.org/pub/FreeBSD>
- <ftp://ftp3.ru.freebsd.org/pub/FreeBSD>
- <ftp://ftp4.ru.freebsd.org/pub/FreeBSD>

#### South Africa

In case of problems, please contact the hostmaster <hostmaster@za.FreeBSD.ORG> for this domain.

- <ftp://ftp.za.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp2.za.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp3.za.FreeBSD.ORG/pub/FreeBSD> (<ftp://ftp3.za.FreeBSD.ORG/FreeBSD>)

#### Slovak Republic

In case of problems, please contact the hostmaster <hostmaster@sk.FreeBSD.ORG> for this domain.

- <ftp://ftp.sk.freebsd.ORG/pub/FreeBSD>

#### Slovenia

In case of problems, please contact the hostmaster <hostmaster@si.FreeBSD.ORG> for this domain.

- <ftp://ftp.si.freebsd.ORG/pub/FreeBSD>

#### Spain

In case of problems, please contact the hostmaster <hostmaster@es.FreeBSD.ORG> for this domain.

- <ftp://ftp.es.freebsd.ORG/pub/FreeBSD>

#### Sweden

In case of problems, please contact the hostmaster <hostmaster@se.FreeBSD.ORG> for this domain.

- <ftp://ftp.se.freebsd.ORG/pub/FreeBSD>
- <ftp://ftp2.se.freebsd.ORG/pub/FreeBSD>
- <ftp://ftp3.se.freebsd.ORG/pub/FreeBSD>

#### Taiwan

In case of problems, please contact the hostmaster <hostmaster@tw.FreeBSD.ORG> for this domain.

- <ftp://ftp.tw.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp2.tw.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp3.tw.FreeBSD.ORG/pub/FreeBSD>

#### Thailand

- <ftp://ftp.nectec.or.th/pub/FreeBSD> Contact: <ftpadmin@ftp.nectec.or.th>.

#### Ukraine

- <ftp://ftp.ua.FreeBSD.ORG/pub/FreeBSD> Contact: <freebsd-mnt@lucky.net>.

#### UK

In case of problems, please contact the hostmaster <hostmaster@uk.FreeBSD.ORG> for this domain.

- <ftp://ftp.uk.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp2.uk.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp3.uk.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp4.uk.FreeBSD.ORG/pub/FreeBSD>

## USA

In case of problems, please contact the hostmaster <hostmaster@FreeBSD.ORG> for this domain.

- <ftp://ftp.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp2.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp3.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp4.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp5.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp6.FreeBSD.ORG/pub/FreeBSD>

The latest versions of export-restricted code for FreeBSD (2.0C or later) (eBones and secure) are being made available at the following locations. If you are outside the U.S. or Canada, please get secure (DES) and eBones (Kerberos) from one of the following foreign distribution sites:

## South Africa

Hostmaster <hostmaster@internat.FreeBSD.ORG> for this domain.

- <ftp://ftp.internat.FreeBSD.ORG/pub/FreeBSD>
- <ftp://ftp2.internat.FreeBSD.ORG/pub/FreeBSD>

## Brazil

Hostmaster <hostmaster@br.FreeBSD.ORG> for this domain.

- <ftp://ftp.br.FreeBSD.ORG/pub/FreeBSD>

## Finland

- <ftp://nic.funet.fi/pub/unix/FreeBSD/eurocrypt> Contact: <count@nic.funet.fi>.

## CTM Sites

CTM/FreeBSD is available via anonymous FTP from the following mirror sites. If you choose to obtain CTM via anonymous FTP, please try to use a site near you.

In case of problems, please contact Poul-Henning Kamp <[phk@FreeBSD.ORG](mailto:phk@FreeBSD.ORG)>.

California, Bay Area, official source

- <ftp://ftp.freebsd.org/pub/FreeBSD/development/CTM>

Germany, Trier

- <ftp://ftp.uni-trier.de/pub/unix/systems/BSD/FreeBSD/CTM>

South Africa, backup server for old deltas

- <ftp://ftp.internat.freebsd.org/pub/FreeBSD/CTM>

Taiwan/R.O.C, Chiayi

- <ftp://ctm.tw.freebsd.org/pub/FreeBSD/CTM>
- <ftp://ctm2.tw.freebsd.org/pub/FreeBSD/CTM>
- <ftp://ctm3.tw.freebsd.org/pub/freebsd/CTM>

If you did not find a mirror near to you or the mirror is incomplete, try FTP search (<http://ftpsearch.ntnu.no/>) at <http://ftpsearch.ntnu.no/ftpsearch> (<http://ftpsearch.ntnu.no/ftpsearch/>). FTP search is a great free archie server in Trondheim, Norway.

## CVSup Sites

CVSup servers for FreeBSD are running at the following sites:

Argentina

- [cvsup.ar.FreeBSD.ORG](http://cvsup.ar.FreeBSD.ORG) (maintainer <msagre@cactus.fi.uba.ar>)

Australia

- [cvsup.au.FreeBSD.ORG](http://cvsup.au.FreeBSD.ORG) (maintainer <dawes@physics.usyd.edu.au>)

Brazil

- [cvsup.br.FreeBSD.ORG](http://cvsup.br.FreeBSD.ORG) (maintainer <cvsup@cvsup.br.freebsd.org>)

Canada

- [cvsup.ca.FreeBSD.ORG](http://cvsup.ca.FreeBSD.ORG) (maintainer <dm@glbalserve.net>)

Denmark

- [cvsup.dk.FreeBSD.ORG](http://cvsup.dk.FreeBSD.ORG) (maintainer <jesper@skriver.dk>)

Estonia

- [cvsup.ee.FreeBSD.ORG](http://cvsup.ee.FreeBSD.ORG) (maintainer <taavi@uninet.ee>)

Finland

- [cvsup.fi.FreeBSD.ORG](http://cvsup.fi.FreeBSD.ORG) (maintainer <count@key.sms.fi>)

## Germany

- [cvsup.de.FreeBSD.ORG](http://cvsup.de.FreeBSD.ORG) (maintainer <wosch@freebsd.org>)
- [cvsup2.de.FreeBSD.ORG](http://cvsup2.de.FreeBSD.ORG) (maintainer <petzi@freebsd.org>)
- [cvsup3.de.FreeBSD.ORG](http://cvsup3.de.FreeBSD.ORG) (maintainer <ag@leo.org>)

## Iceland

- [cvsup.is.FreeBSD.ORG](http://cvsup.is.FreeBSD.ORG) (maintainer <adam@veda.is>)

## Japan

- [cvsup.jp.FreeBSD.ORG](http://cvsup.jp.FreeBSD.ORG) (maintainer <simokawa@sat.t.u-tokyo.ac.jp>)
- [cvsup2.jp.FreeBSD.ORG](http://cvsup2.jp.FreeBSD.ORG) (maintainer <max@FreeBSD.ORG>)
- [cvsup3.jp.FreeBSD.ORG](http://cvsup3.jp.FreeBSD.ORG) (maintainer <shige@cin.nihon-u.ac.jp>)
- [cvsup4.jp.FreeBSD.ORG](http://cvsup4.jp.FreeBSD.ORG) (maintainer <cvsup-admin@ftp.media.kyoto-u.ac.jp>)
- [cvsup5.jp.FreeBSD.ORG](http://cvsup5.jp.FreeBSD.ORG) (maintainer <cvsup@imasy.or.jp>)

## Netherlands

- [cvsup.nl.FreeBSD.ORG](http://cvsup.nl.FreeBSD.ORG) (maintainer <xaa@xaa.iae.nl>)

## Norway

- [cvsup.no.FreeBSD.ORG](http://cvsup.no.FreeBSD.ORG) (maintainer <Tor.Egge@idt.ntnu.no>)

## Poland

- [cvsup.pl.FreeBSD.ORG](http://cvsup.pl.FreeBSD.ORG) (maintainer <Mariusz@kam.pl>)

Russia

- `cvsup.ru.FreeBSD.ORG` (maintainer <mishania@demos.su>)
- `cvsup2.ru.FreeBSD.ORG` (maintainer <dv@dv.ru>)

Sweden

- `cvsup.se.FreeBSD.ORG` (maintainer <pantzer@ludd.luth.se>)

Slovak Republic

- `cvsup.sk.FreeBSD.ORG` (maintainer <tps@tps.sk>)
- `cvsup2.sk.FreeBSD.ORG` (maintainer <tps@tps.sk>)

South Africa

- `cvsup.za.FreeBSD.ORG` (maintainer <markm@FreeBSD.ORG>)
- `cvsup2.za.FreeBSD.ORG` (maintainer <markm@FreeBSD.ORG>)

Taiwan

- `cvsup.tw.FreeBSD.ORG` (maintainer <jdli@freebsd.csie.nctu.edu.tw>)

Ukraine

- `cvsup2.ua.FreeBSD.ORG` (maintainer <freebsd-mnt@lucky.net>)

United Kingdom

- `cvsup.uk.FreeBSD.ORG` (maintainer <joe@pavilion.net>)

## USA

- `cvsup1.FreeBSD.ORG` (maintainer <skynyrd@opus.cts.cwu.edu>), Washington state
- `cvsup2.FreeBSD.ORG` (maintainer <jdp@FreeBSD.ORG>), California
- `cvsup3.FreeBSD.ORG` (maintainer <wollman@FreeBSD.ORG>), Massachusetts
- `cvsup4.FreeBSD.ORG` (maintainer <shmit@rcn.com>), Virginia
- `cvsup5.FreeBSD.ORG` (maintainer <cvsup@adsu.bellsouth.com>), Georgia

The export-restricted code for FreeBSD (eBones and secure) is available via **CVSup** at the following international repository. Please use this site to get the export-restricted code, if you are outside the USA or Canada.

## South Africa

- `cvsup.internat.FreeBSD.ORG` (maintainer <markm@FreeBSD.ORG>)

The following **CVSup** site is especially designed for **CTM** users. Unlike the other CVSup mirrors, it is kept up-to-date by **CTM**. That means if you **CVSup** `cvs-all` with `release=cvs` from this site, you get a version of the repository (including the inevitable `.ctm_status` file) which is suitable for being updated using the **CTM** `cvs-cur` deltas. This allows users who track the entire `cvs-all` tree to go from **CVSup** to **CTM** without having to rebuild their repository from scratch using a fresh **CTM** base delta.

**Note:** This special feature only works for the `cvs-all` distribution with `cvs` as the release tag. CVSupping any other distribution and/or release will get you the specified distribution, but it will not be suitable for **CTM** updating.

**Note:** Because the current version of **CTM** does not preserve the timestamps of files, the timestamps at this mirror site are not the same as those at other mirror sites. Switching between this site and other sites is not recommended. It will work correctly, but will be somewhat inefficient.

Germany

- `ctm.FreeBSD.ORG` (maintainer `<blank@fox.uni-trier.de>`)

## AFS Sites

AFS servers for FreeBSD are running at the following sites;

Sweden

The path to the files are: `/afs/stacken.kth.se/ftp/pub/FreeBSD`

- `stacken.kth.se`, Stacken Computer Club, KTH, Sweden
- `130.237.234.3`, `milko.stacken.kth.se`
- `130.237.234.43`, `hot.stacken.kth.se`
- `130.237.234.44`, `dog.stacken.kth.se`

Maintainer `<ftp@stacken.kth.se>`

## Chapter 26. Bibliography

While the manual pages provide the definitive reference for individual pieces of the FreeBSD operating system, they are notorious for not illustrating how to put the pieces together to make the whole operating system run smoothly. For this, there is no substitute for a good book on UNIX system administration and a good users' manual.

### Books & Magazines Specific to FreeBSD

#### *International books & Magazines:*

- Using FreeBSD (<http://freebsd.csie.nctu.edu.tw/~jdli/book.html>) (in Chinese).
- FreeBSD for PC 98'ers (in Japanese), published by SHUWA System Co, LTD. ISBN 4-87966-468-5 C3055 P2900E.
- FreeBSD (in Japanese), published by CUTT. ISBN 4-906391-22-2 C3055 P2400E.
- Complete Introduction to FreeBSD ([http://www.shoeisha.co.jp/pc/index/shinkan/97\\_05\\_06.htm](http://www.shoeisha.co.jp/pc/index/shinkan/97_05_06.htm)) (in Japanese), published by Shoeisha Co., Ltd (<http://www.shoeisha.co.jp/>). ISBN 4-88135-473-6 P3600E.
- Personal UNIX Starter Kit FreeBSD (<http://www.ascii.co.jp/pb/book1/shinkan/detail/1322785.html>) (in Japanese), published by ASCII (<http://www.ascii.co.jp/>). ISBN 4-7561-1733-3 P3000E.
- FreeBSD Handbook (Japanese translation), published by ASCII (<http://www.ascii.co.jp/>). ISBN 4-7561-1580-2 P3800E.
- FreeBSD mit Methode (in German), published by Computer und Literatur Verlag/Vertrieb Hanser, 1998. ISBN 3-932311-31-0.
- FreeBSD Install and Utilization Manual (<http://www.pc.mycom.co.uk/FreeBSD/install-manual.html>) (in Japanese), published by Mainichi Communications Inc. (<http://www.pc.mycom.co.jp/>).

#### *English language books & Magazines:*

- The Complete FreeBSD (<http://www.cdrom.com/titles/freebsd/bsdbook2.htm>), published by Walnut Creek CDROM (<http://www.cdrom.com>).

## Users' Guides

- Computer Systems Research Group, UC Berkeley. *4.4BSD User's Reference Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-075-9
- Computer Systems Research Group, UC Berkeley. *4.4BSD User's Supplementary Documents*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-076-7
- *UNIX in a Nutshell*. O'Reilly & Associates, Inc., 1990. ISBN 093717520X
- Mui, Linda. *What You Need To Know When You Can't Find Your UNIX System Administrator*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-104-6
- Ohio State University (<http://www-wks.acs.ohio-state.edu/>) has written a UNIX Introductory Course ([http://www-wks.acs.ohio-state.edu/unix\\_course/unix.html](http://www-wks.acs.ohio-state.edu/unix_course/unix.html)) which is available online in HTML and postscript format.
- Jpman Project, Japan FreeBSD Users Group (<http://www.jp.FreeBSD.ORG/>). FreeBSD User's Reference Manual (<http://www.pc.mycom.co.jp/FreeBSD/urm.html>) (Japanese translation). Mainichi Communications Inc. (<http://www.pc.mycom.co.jp/>), 1998. ISBN4-8399-0088-4 P3800E.

## Administrators' Guides

- Albitz, Paul and Liu, Cricket. *DNS and BIND*, 2nd Ed. O'Reilly & Associates, Inc., 1997. ISBN 1-56592-236-0
- Computer Systems Research Group, UC Berkeley. *4.4BSD System Manager's Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-080-5
- Costales, Brian, et al. *Sendmail*, 2nd Ed. O'Reilly & Associates, Inc., 1997. ISBN 1-56592-222-0
- Frisch, Aileen. *Essential System Administration*, 2nd Ed. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-127-5
- Hunt, Craig. *TCP/IP Network Administration*. O'Reilly & Associates, Inc., 1992. ISBN 0-937175-82-X
- Nemeth, Evi. *UNIX System Administration Handbook*. 2nd Ed. Prentice Hall, 1995. ISBN 0131510517
- Stern, Hal *Managing NFS and NIS* O'Reilly & Associates, Inc., 1991. ISBN 0-937175-75-7

- Jpman Project, Japan FreeBSD Users Group (<http://www.jp.FreeBSD.ORG/>). FreeBSD System Administrator's Manual (<http://www.pc.mycom.co.jp/FreeBSD/sam.html>) (Japanese translation). Mainichi Communications Inc. (<http://www.pc.mycom.co.jp/>), 1998. ISBN4-8399-0109-0 P3300E.

## Programmers' Guides

- Asente, Paul. *X Window System Toolkit*. Digital Press. ISBN 1-55558-051-3
- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Reference Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-078-3
- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Supplementary Documents*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-079-1
- Harbison, Samuel P. and Steele, Guy L. Jr. *C: A Reference Manual*. 4rd ed. Prentice Hall, 1995. ISBN 0-13-326224-3
- Kernighan, Brian and Dennis M. Ritchie. *The C Programming Language*. PTR Prentice Hall, 1988. ISBN 0-13-110362-9
- Lehey, Greg. *Porting UNIX Software*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-126-7
- Plauger, P. J. *The Standard C Library*. Prentice Hall, 1992. ISBN 0-13-131509-9
- Stevens, W. Richard. *Advanced Programming in the UNIX Environment*. Reading, Mass. : Addison-Wesley, 1992 ISBN 0-201-56317-7
- Stevens, W. Richard. *UNIX Network Programming*. 2nd Ed, PTR Prentice Hall, 1998. ISBN 0-13-490012-X
- Wells, Bill. "Writing Serial Drivers for UNIX". *Dr. Dobb's Journal*. 19(15), December 1994. pp68-71, 97-99.

## Operating System Internals

- Andleigh, Prabhat K. *UNIX System Architecture*. Prentice-Hall, Inc., 1990. ISBN 0-13-949843-5
- Jolitz, William. "Porting UNIX to the 386". *Dr. Dobb's Journal*. January 1991-July 1992.
- Leffler, Samuel J., Marshall Kirk McKusick, Michael J Karels and John Quarterman *The Design and Implementation of the 4.3BSD UNIX Operating System*. Reading, Mass. : Addison-Wesley, 1989.

ISBN 0-201-06196-1

- Leffler, Samuel J., Marshall Kirk McKusick, *The Design and Implementation of the 4.3BSD UNIX Operating System: Answer Book*. Reading, Mass. : Addison-Wesley, 1991. ISBN 0-201-54629-9
- McKusick, Marshall Kirk, Keith Bostic, Michael J Karels, and John Quarterman. *The Design and Implementation of the 4.4BSD Operating System*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-54979-4
- Stevens, W. Richard. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63346-9
- Schimmel, Curt. *Unix Systems for Modern Architectures*. Reading, Mass. : Addison-Wesley, 1994. ISBN 0-201-63338-8
- Stevens, W. Richard. *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP and the UNIX Domain Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63495-3
- Vahalia, Uresh. *UNIX Internals – The New Frontiers*. Prentice Hall, 1996. ISBN 0-13-101908-2
- Wright, Gary R. and W. Richard Stevens. *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63354-X

## Security Reference

- Cheswick, William R. and Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63357-4
- Garfinkel, Simson and Gene Spafford. *Practical UNIX Security*. 2nd Ed. O'Reilly & Associates, Inc., 1996. ISBN 1-56592-148-8
- Garfinkel, Simson. *PGP Pretty Good Privacy* O'Reilly & Associates, Inc., 1995. ISBN 1-56592-098-8

## Hardware Reference

- Anderson, Don and Tom Shanley. *Pentium Processor System Architecture*. 2nd Ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40992-5
- Ferraro, Richard F. *Programmer's Guide to the EGA, VGA, and Super VGA Cards*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-62490-7

- Intel Corporation publishes documentation on their CPUs, chipsets and standards on their developer web site (<http://developer.intel.com/>), usually as PDF files.
- Shanley, Tom. *80486 System Architecture*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40994-1
- Shanley, Tom. *ISA System Architecture*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40996-8
- Shanley, Tom. *PCI System Architecture*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40993-3
- Van Gilluwe, Frank. *The Undocumented PC*. Reading, Mass: Addison-Wesley Pub. Co., 1994. ISBN 0-201-62277-7

## UNIX History

- Lion, John *Lion's Commentary on UNIX, 6th Ed. With Source Code*. ITP Media Group, 1996. ISBN 1573980137
- Raymond, Eric S. *The New Hacker's Dictionary, 3rd edition*. MIT Press, 1996. ISBN 0-262-68092-0. Also known as the Jargon File (<http://www.ccil.org/jargon/jargon.html>)
- Salus, Peter H. *A quarter century of UNIX*. Addison-Wesley Publishing Company, Inc., 1994. ISBN 0-201-54777-5
- Simon Garfinkel, Daniel Weise, Steven Strassmann. *The UNIX-HATERS Handbook*. IDG Books Worldwide, Inc., 1994. ISBN 1-56884-203-1
- Don Libes, Sandy Ressler *Life with UNIX* — special edition. Prentice-Hall, Inc., 1989. ISBN 0-13-536657-7
- *The BSD family tree*. 1997.  
<ftp://ftp.freebsd.org/pub/FreeBSD/FreeBSD-current/src/share/misc/bsd-family-tree> or local (file:/usr/share/misc/bsd-family-tree) on a FreeBSD-current machine.
- *The BSD Release Announcements collection*. 1997. <http://www.de.FreeBSD.ORG/de/ftp/releases/>
- *Networked Computer Science Technical Reports Library*. <http://www.ncstrl.org/>
- *Old BSD releases from the Computer Systems Research group (CSRG)*.  
<http://www.mckusick.com/csrg/>: The 4CD set covers all BSD versions from 1BSD to 4.4BSD and 4.4BSD-Lite2 (but not 2.11BSD, unfortunately). As well, the last disk holds the final sources plus the SCCS files.

## Magazines and Journals

- *The C/C++ Users Journal*. R&D Publications Inc. ISSN 1075-2838
- *Sys Admin — The Journal for UNIX System Administrators* Miller Freeman, Inc., ISSN 1061-2688

# Chapter 27. Resources on the Internet

*Contributed by Jordan K. Hubbard <jkh@FreeBSD.ORG>.*

The rapid pace of FreeBSD progress makes print media impractical as a means of following the latest developments. Electronic resources are the best, if not often the only, way stay informed of the latest advances. Since FreeBSD is a volunteer effort, the user community itself also generally serves as a “technical support department” of sorts, with electronic mail and USENET news being the most effective way of reaching that community.

The most important points of contact with the FreeBSD user community are outlined below. If you are aware of other resources not mentioned here, please send them to the FreeBSD documentation project mailing list <freebsd-doc@FreeBSD.ORG>so that they may also be included.

## Mailing lists

Though many of the FreeBSD development members read USENET, we cannot always guarantee that we will get to your questions in a timely fashion (or at all) if you post them only to one of the `comp.unix.bsd.freebsd.*` groups. By addressing your questions to the appropriate mailing list you will reach both us and a concentrated FreeBSD audience, invariably assuring a better (or at least faster) response.

The charters for the various lists are given at the bottom of this document. *Please read the charter before joining or sending mail to any list.* Most of our list subscribers now receive many hundreds of FreeBSD related messages every day, and by setting down charters and rules for proper use we are striving to keep the signal-to-noise ratio of the lists high. To do less would see the mailing lists ultimately fail as an effective communications medium for the project.

Archives are kept for all of the mailing lists and can be searched using the FreeBSD World Wide Web server (<http://www.FreeBSD.ORG/search.html>). The keyword searchable archive offers an excellent way of finding answers to frequently asked questions and should be consulted before posting a question.

## List summary

*General lists:* The following are general lists which anyone is free (and encouraged) to join:

List	Purpose
freebsd-advocacy	FreeBSD Evangelism
freebsd-announce	Important events and project milestones

freebsd-bugs	Bug reports
freebsd-chat	Non-technical items related to the FreeBSD community
freebsd-current	Discussion concerning the use of FreeBSD-current
freebsd-isp	Issues for Internet Service Providers using FreeBSD
freebsd-jobs	FreeBSD employment and consulting opportunities
freebsd-newbies	New FreeBSD users activities and discussions
freebsd-questions	User questions and technical support
freebsd-stable	Discussion concerning the use of FreeBSD-stable

*Technical lists:* The following lists are for technical discussion. You should read the charter for each list carefully before joining or sending mail to one as there are firm guidelines for their use and content.

<b>List</b>	<b>Purpose</b>
freebsd-afs	Porting AFS to FreeBSD
freebsd-alpha	Porting FreeBSD to the Alpha
freebsd-doc	Creating FreeBSD related documents
freebsd-database	Discussing database use and development under FreeBSD
freebsd-emulation	Emulation of other systems such as Linux/DOS/Windows
freebsd-fs	Filesystems
freebsd-hackers	General technical discussion
freebsd-hardware	General discussion of hardware for running FreeBSD
freebsd-isdn	ISDN developers
freebsd-java	Java developers and people porting JDKs to FreeBSD
freebsd-mobile	Discussions about mobile computing
freebsd-mozilla	Porting mozilla to FreeBSD
freebsd-net	Networking discussion and TCP/IP/source code
freebsd-platforms	Concerning ports to non-Intel architecture platforms

freebsd-ports	Discussion of the ports collection
freebsd-scsi	The SCSI subsystem
freebsd-security	Security issues
freebsd-small	Using FreeBSD in embedded applications
freebsd-smp	Design discussions for [A]Symmetric MultiProcessing
freebsd-sparc	Porting FreeBSD to Sparc systems
freebsd-tokenring	Support Token Ring in FreeBSD

*Limited lists:* The following lists require approval from <core@FreeBSD.ORG> to join, though anyone is free to send messages to them which fall within the scope of their charters. It is also a good idea establish a presence in the technical lists before asking to join one of these limited lists.

List	Purpose
freebsd-admin	Administrative issues
freebsd-arch	Architecture and design discussions
freebsd-core	FreeBSD core team
freebsd-hubs	People running mirror sites (infrastructural support)
freebsd-install	Installation development
freebsd-security-notifications	Security notifications
freebsd-user-groups	User group coordination

*CVS lists:* The following lists are for people interested in seeing the log messages for changes to various areas of the source tree. They are *Read-Only* lists and should not have mail sent to them.

List	Source area	Area Description (source for)
cvb-all	/usr/src	All changes to the tree (superset)

## How to subscribe

All mailing lists live on FreeBSD.ORG, so to post to a given list you simply mail to <listname@FreeBSD.ORG>. It will then be redistributed to mailing list members world-wide.

To subscribe to a list, send mail to <majordomo@FreeBSD.ORG> and include

```
subscribe <listname> [<optional address>]
```

in the body of your message. For example, to subscribe yourself to `freebsd-announce`, you'd do:

```
% mail majordomo@FreeBSD.ORG
subscribe freebsd-announce
^D
```

If you want to subscribe yourself under a different name, or submit a subscription request for a local mailing list (this is more efficient if you have several interested parties at one site, and highly appreciated by us!), you would do something like:

```
% mail majordomo@FreeBSD.ORG
subscribe freebsd-announce local-announce@somesite.com
^D
```

Finally, it is also possible to unsubscribe yourself from a list, get a list of other list members or see the list of mailing lists again by sending other types of control messages to `majordomo`. For a complete list of available commands, do this:

```
% mail majordomo@FreeBSD.ORG
help
^D
```

Again, we would like to request that you keep discussion in the technical mailing lists on a technical track. If you are only interested in the “high points” then it is suggested that you join `freebsd-announce`, which is intended only for infrequent traffic.

## List charters

All FreeBSD mailing lists have certain basic rules which must be adhered to by anyone using them. Failure to comply with these guidelines will result in two (2) written warnings from the FreeBSD Postmaster <[postmaster@freebsd.org](mailto:postmaster@freebsd.org)>, after which, on a third offense, the poster will be removed from all FreeBSD mailing lists and filtered from further posting to them. We regret that such rules and measures are necessary at all, but today's Internet is a pretty harsh environment, it would seem, and many fail to appreciate just how fragile some of its mechanisms are.

Rules of the road:

- The topic of any posting should adhere to the basic charter of the list it is posted to, e.g. if the list is about technical issues then your posting should contain technical discussion. Ongoing irrelevant chatter or flaming only detracts from the value of the mailing list for everyone on it and will not be

tolerated. For free-form discussion on no particular topic, the freebsd-chat <freebsd-chat@freebsd.org> mailing list is freely available and should be used instead.

- No posting should be made to more than 2 mailing lists, and only to 2 when a clear and obvious need to post to both lists exists. For most lists, there is already a great deal of subscriber overlap and except for the most esoteric mixes (say "-stable & -scsi"), there really is no reason to post to more than one list at a time. If a message is sent to you in such a way that multiple mailing lists appear on the Cc line then the cc line should also be trimmed before sending it out again. *You are still responsible for your own cross-postings, no matter who the originator might have been.*
- Personal attacks and profanity (in the context of an argument) are not allowed, and that includes users and developers alike. Gross breaches of netiquette, like excerpting or reposting private mail when permission to do so was not and would not be forthcoming, are frowned upon but not specifically enforced. *However*, there are also very few cases where such content would fit within the charter of a list and it would therefore probably rate a warning (or ban) on that basis alone.
- Advertising of non-FreeBSD related products or services is strictly prohibited and will result in an immediate ban if it is clear that the offender is advertising by spam.

*Individual list charters:*

#### FREEBSD-AFS

*Andrew File System*

This list is for discussion on porting and using AFS from CMU/Transarc

#### FREEBSD-ADMIN

*Administrative issues*

This list is purely for discussion of freebsd.org related issues and to report problems or abuse of project resources. It is a closed list, though anyone may report a problem (with our systems!) to it.

#### FREEBSD-ANNOUNCE

*Important events / milestones*

This is the mailing list for people interested only in occasional announcements of significant FreeBSD events. This includes announcements about snapshots and other releases. It contains announcements of new FreeBSD capabilities. It may contain calls for volunteers etc. This is a low volume, strictly moderated mailing list.

## FREEBSD-ARCH

### *Architecture and design discussions*

This is a moderated list for discussion of FreeBSD architecture. Messages will mostly be kept technical in nature, with (rare) exceptions for other messages the moderator deems need to reach all the subscribers of the list. Examples of suitable topics;

- How to re-vamp the build system to have several customized builds running at the same time.
- What needs to be fixed with VFS to make Heidemann layers work.
- How do we change the device driver interface to be able to use the same drivers cleanly on many buses and architectures?
- How do I write a network driver?

The moderator reserves the right to do minor editing (spell-checking, grammar correction, trimming) of messages that are posted to the list. The volume of the list will be kept low, which may involve having to delay topics until an active discussion has been resolved.

## FREEBSD-BUGS

### *Bug reports*

This is the mailing list for reporting bugs in FreeBSD. Whenever possible, bugs should be submitted using the `send-pr(1)` command or the WEB interface (<http://www.freebsd.org/send-pr.html>) to it.

## FREEBSD-CHAT

### *Non technical items related to the FreeBSD community*

This list contains the overflow from the other lists about non-technical, social information. It includes discussion about whether Jordan looks like a toon ferret or not, whether or not to type in capitals, who is drinking too much coffee, where the best beer is brewed, who is brewing beer in their basement, and so on. Occasional announcements of important events (such as upcoming parties, weddings, births, new jobs, etc) can be made to the technical lists, but the follow ups should be directed to this -chat list.

## FREEBSD-CORE

### *FreeBSD core team*

This is an internal mailing list for use by the core members. Messages can be sent to it when a serious FreeBSD-related matter requires arbitration or high-level scrutiny.

#### FREEBSD-CURRENT

*Discussions about the use of FreeBSD-current*

This is the mailing list for users of freebsd-current. It includes warnings about new features coming out in -current that will affect the users, and instructions on steps that must be taken to remain -current. Anyone running “current” must subscribe to this list. This is a technical mailing list for which strictly technical content is expected.

#### FREEBSD-CURRENT-DIGEST

*Discussions about the use of FreeBSD-current*

This is the digest version of the freebsd-current mailing list. The digest consists of all messages sent to freebsd-current bundled together and mailed out as a single message. The average digest size is about 40kB. This list is *Read-Only* and should not be posted to.

#### FREEBSD-DOC

*Documentation project*

This mailing list is for the discussion of issues and projects related to the creation of documentation for FreeBSD. The members of this mailing list are collectively referred to as “The FreeBSD Documentation Project”. It is an open list; feel free to join and contribute!

#### FREEBSD-FS

*Filesystems*

Discussions concerning FreeBSD filesystems. This is a technical mailing list for which strictly technical content is expected.

#### FREEBSD-ISDN

*ISDN Communications*

This is the mailing list for people discussing the development of ISDN support for FreeBSD.

#### FREEBSD-JAVA

*Java Development*

This is the mailing list for people discussing the development of significant Java applications for FreeBSD and the porting and maintenance of JDKs.

## FREEBSD-HACKERS

### *Technical discussions*

This is a forum for technical discussions related to FreeBSD. This is the primary technical mailing list. It is for individuals actively working on FreeBSD, to bring up problems or discuss alternative solutions. Individuals interested in following the technical discussion are also welcome. This is a technical mailing list for which strictly technical content is expected.

## FREEBSD-HACKERS-DIGEST

### *Technical discussions*

This is the digest version of the freebsd-hackers mailing list. The digest consists of all messages sent to freebsd-hackers bundled together and mailed out as a single message. The average digest size is about 40kB. This list is *Read-Only* and should not be posted to.

## FREEBSD-HARDWARE

### *General discussion of FreeBSD hardware*

General discussion about the types of hardware that FreeBSD runs on, various problems and suggestions concerning what to buy or avoid.

## FREEBSD-INSTALL

### *Installation discussion*

This mailing list is for discussing FreeBSD installation development for the future releases and is closed.

## FREEBSD-ISP

### *Issues for Internet Service Providers*

This mailing list is for discussing topics relevant to Internet Service Providers (ISPs) using FreeBSD. This is a technical mailing list for which strictly technical content is expected.

## FREEBSD-NEWBIES

### *Newbies activities discussion*

We cover any of the activities of newbies that are not already dealt with elsewhere, including: independent learning and problem solving techniques, finding and using resources and asking for help elsewhere, how to use mailing lists and which lists to use, general chat, making mistakes,

boasting, sharing ideas, stories, moral (but not technical) support, and taking an active part in the FreeBSD community. We take our problems and support questions to `freebsd-questions`, and use `freebsd-newbies` to meet others who are doing the same things that we do as newbies.

## FREEBSD-PLATFORMS

### *Porting to Non-Intel platforms*

Cross-platform `freebsd` issues, general discussion and proposals for non-Intel FreeBSD ports. This is a technical mailing list for which strictly technical content is expected.

## FREEBSD-PORTS

### *Discussion of “ports”*

Discussions concerning FreeBSD’s “ports collection” (`/usr/ports`), proposed ports, modifications to ports collection infrastructure and general coordination efforts. This is a technical mailing list for which strictly technical content is expected.

## FREEBSD-QUESTIONS

### *User questions*

This is the mailing list for questions about FreeBSD. You should not send “how to” questions to the technical lists unless you consider the question to be pretty technical.

## FREEBSD-QUESTIONS-DIGEST

### *User questions*

This is the digest version of the `freebsd-questions` mailing list. The digest consists of all messages sent to `freebsd-questions` bundled together and mailed out as a single message. The average digest size is about 40kB.

## FREEBSD-SCSI

### *SCSI subsystem*

This is the mailing list for people working on the `scsi` subsystem for FreeBSD. This is a technical mailing list for which strictly technical content is expected.

## FREEBSD-SECURITY

### *Security issues*

FreeBSD computer security issues (DES, Kerberos, known security holes and fixes, etc). This is a technical mailing list for which strictly technical content is expected.

#### FREEBSD-SECURITY-NOTIFICATIONS

*Security Notifications* Notifications of FreeBSD security problems and fixes. This is not a discussion list. The discussion list is FreeBSD-security.

#### FREEBSD-SMALL

This list discusses topics related to unusually small and embedded FreeBSD installations. This is a technical mailing list for which strictly technical content is expected.

#### FREEBSD-STABLE

*Discussions about the use of FreeBSD-stable*

This is the mailing list for users of freebsd-stable. It includes warnings about new features coming out in -stable that will affect the users, and instructions on steps that must be taken to remain -stable. Anyone running “stable” should subscribe to this list. This is a technical mailing list for which strictly technical content is expected.

#### FREEBSD-USER-GROUPS

*User Group Coordination List*

This is the mailing list for the coordinators from each of the local area Users Groups to discuss matters with each other and a designated individual from the Core Team. This mail list should be limited to meeting synopsis and coordination of projects that span User Groups. It is a closed list.

## Usenet newsgroups

In addition to two FreeBSD specific newsgroups, there are many others in which FreeBSD is discussed or are otherwise relevant to FreeBSD users. Keyword searchable archives ([http://minnie.cs.adfa.oz.au/BSD-info/bsdnews\\_search.html](http://minnie.cs.adfa.oz.au/BSD-info/bsdnews_search.html)) are available for some of these newsgroups from courtesy of Warren Toomey <[wkt@cs.adfa.oz.au](mailto:wkt@cs.adfa.oz.au)>.

## BSD specific newsgroups

- comp.unix.bsd.freebsd.announce (news:comp.unix.bsd.freebsd.announce)
- comp.unix.bsd.freebsd.misc (news:comp.unix.bsd.freebsd.misc)

## Other Unix newsgroups of interest

- comp.unix (news:comp.unix)
- comp.unix.questions (news:comp.unix.questions)
- comp.unix.admin (news:comp.unix.admin)
- comp.unix.programmer (news:comp.unix.programmer)
- comp.unix.shell (news:comp.unix.shell)
- comp.unix.user-friendly (news:comp.unix.user-friendly)
- comp.security.unix (news:comp.security.unix)
- comp.sources.unix (news:comp.sources.unix)
- comp.unix.advocacy (news:comp.unix.advocacy)
- comp.unix.misc (news:comp.unix.misc)
- comp.bugs.4bsd (news:comp.bugs.4bsd)
- comp.bugs.4bsd.ucb-fixes (news:comp.bugs.4bsd.ucb-fixes)
- comp.unix.bsd (news:comp.unix.bsd)

## X Window System

- comp.windows.x.i386unix (news:comp.windows.x.i386unix)
- comp.windows.x (news:comp.windows.x)
- comp.windows.x.apps (news:comp.windows.x.apps)
- comp.windows.x.announce (news:comp.windows.x.announce)
- comp.windows.x.intrinsics (news:comp.windows.x.intrinsics)
- comp.windows.x.motif (news:comp.windows.x.motif)

- [comp.windows.x.pex](mailto:comp.windows.x.pex) (news:comp.windows.x.pex)
- [comp.emulators.ms-windows.wine](mailto:comp.emulators.ms-windows.wine) (news:comp.emulators.ms-windows.wine)

## World Wide Web servers

- <http://www.FreeBSD.ORG/> — Central Server.
- <http://www.au.freebsd.org/FreeBSD/> — Australia/1.
- <http://www2.au.freebsd.org/FreeBSD/> — Australia/2.
- <http://www3.au.freebsd.org/FreeBSD/> — Australia/3.
- <http://www.br.freebsd.org/www.freebsd.org/> — Brazil/1.
- <http://www.br2.freebsd.org/www.freebsd.org/> — Brazil/2.
- <http://www3.br.freebsd.org/> — Brazil/3.
- <http://www.bg.freebsd.org/> — Bulgaria.
- <http://www.ca.freebsd.org/> — Canada/1.
- <http://freebsd.kawartha.com/> — Canada/2.
- <http://www.dk.freebsd.org/> — Denmark.
- <http://www.ee.freebsd.org/> — Estonia.
- <http://www.fi.freebsd.org/> — Finland.
- <http://www.fr.freebsd.org/> — France.
- <http://www.de.freebsd.org/> — Germany/1.
- <http://www1.de.freebsd.org/> — Germany/2.
- <http://www.de.freebsd.org/> (<http://www2.de.freebsd.org/>) — Germany/3.
- <http://www.hu.freebsd.org/> — Hungary.
- <http://www.is.freebsd.org/> — Iceland.
- <http://www.ie.freebsd.org/> — Ireland.
- <http://www.jp.freebsd.org/www.freebsd.org/> — Japan.
- <http://www.kr.freebsd.org/> — Korea.
- <http://www.lv.freebsd.org/> — Latvia.

- <http://rama.asiapac.net/freebsd/> — Malaysia.
- <http://www.nl.freebsd.org/> — Netherlands.
- <http://www.no.freebsd.org/> — Norway.
- <http://www.pt.freebsd.org/> — Portugal/1.
- <http://www2.pt.freebsd.org/> — Portugal/2.
- <http://www3.pt.freebsd.org/> — Portugal/3.
- <http://www.ro.freebsd.org/> — Romania.
- <http://www.ru.freebsd.org/> — Russia/1.
- <http://www2.ru.freebsd.org/> — Russia/2.
- <http://www3.ru.freebsd.org/> — Russia/3.
- <http://www4.ru.freebsd.org/> — Russia/4.
- <http://www.sk.freebsd.org/> — Slovak Republic.
- <http://www.si.freebsd.org/> — Slovenia.
- <http://www.es.freebsd.org/> — Spain.
- <http://www.za.freebsd.org/> — South Africa/1.
- <http://www2.za.freebsd.org/> — South Africa/2.
- <http://www.se.freebsd.org/www.freebsd.org/> — Sweden.
- <http://www.tr.freebsd.org/> — Turkey.
- <http://www.ua.freebsd.org/> — Ukraine/1.
- <http://www2.ua.freebsd.org/> — Ukraine/2.
- <http://www.uk.freebsd.org/> — United Kingdom.
- <http://freebsd.advansys.net/> — USA/Indiana.
- <http://www6.freebsd.org/> — USA/Oregon.
- <http://www2.freebsd.org/> (<http://www2freebsd.org/>) — USA/Texas.

# Chapter 28. FreeBSD Project Staff

The FreeBSD Project is managed and operated by the following groups of people:

## The FreeBSD Core Team

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.

(in alphabetical order by last name):

- Satoshi Asami <asami@FreeBSD.ORG>
- Jonathan M. Bresler <jmb@FreeBSD.ORG>
- Andrey A. Chernov <ache@FreeBSD.ORG>
- Bruce Evans <bde@FreeBSD.ORG>
- Justin T. Gibbs <gibbs@FreeBSD.ORG>
- David Greenman <dg@FreeBSD.ORG>
- Jordan K. Hubbard <jkh@FreeBSD.ORG>
- Poul-Henning Kamp <phk@FreeBSD.ORG>
- Rich Murphey <rich@FreeBSD.ORG>
- Gary Palmer <gpalmer@FreeBSD.ORG>
- John Polstra <jdp@FreeBSD.ORG>
- Søren Schmidt <sos@FreeBSD.ORG>
- Peter Wemm <peter@FreeBSD.ORG>
- Garrett Wollman <wollman@FreeBSD.ORG>
- Jörg Wunsch <joerg@FreeBSD.ORG>

## The FreeBSD Developers

These are the people who have commit privileges and do the engineering work on the FreeBSD source tree. All core team members are also developers.

- Ugen J.S.Antsilevich <ugen@FreeBSD.ORG>
- Ade Barkah <mbarkah@FreeBSD.ORG>
- Stefan Bethke <stb@FreeBSD.ORG>
- Pierre Beyssac <pb@fasterix.freenix.org>
- Andrzej Bialecki <abial@FreeBSD.ORG>
- John Birrell <jb@cimlogic.com.au>
- Torsten Blum <torstenb@FreeBSD.ORG>
- Donald Burr <dburr@FreeBSD.ORG>
- Philippe Charnier <charnier@FreeBSD.ORG>
- Luoqi Chen <luoqi@FreeBSD.ORG>
- Eric J. Chet <ejc@FreeBSD.ORG>
- Kenjiro Cho <kjc@FreeBSD.ORG>
- Gary Clark II <gclarkii@FreeBSD.ORG>
- Archie Cobbs <archie@FreeBSD.ORG>
- Martin Cracauer <cracauer@FreeBSD.ORG>
- Adam David <adam@FreeBSD.ORG>
- Matthew Dillon <dillon@FreeBSD.ORG>
- Peter Dufault <dufault@FreeBSD.ORG>
- Frank Durda IV <uhclem@FreeBSD.ORG>
- Tor Egge <tegge@FreeBSD.ORG>
- Eivind Eklund <eivind@FreeBSD.ORG>
- Julian Elischer <julian@FreeBSD.ORG>
- Ralf S. Engelschall <rse@FreeBSD.ORG>
- Stefan Esser <se@FreeBSD.ORG>
- Sean Eric Fagan <sef@FreeBSD.ORG>
- Bill Fenner <fenner@FreeBSD.ORG>
- John Fieber <jfieber@FreeBSD.ORG>
- James FitzGibbon <james@nexus.net>
- Marc G. Fournier <scrappy@FreeBSD.ORG>

- Lars Fredriksen <lars@FreeBSD.ORG>
- &.dirk;
- Shigeyuki Fukushima <shige@FreeBSD.ORG>
- Bill Fumerola <billf@FreeBSD.ORG>
- Andrew Gallatin <gallatin@FreeBSD.ORG>
- Thomas Gellekum <tg@FreeBSD.ORG>
- Brandon Gillespie <brandon@FreeBSD.ORG>
- Thomas Graichen <graichen@FreeBSD.ORG>
- Joe Greco <jgreco@FreeBSD.ORG>
- Rodney Grimes <rgrimes@FreeBSD.ORG>
- John-Mark Gurney <jmg@FreeBSD.ORG>
- Hiroyuki HANAI <hanai@FreeBSD.ORG>
- Peter Hawkins <thepish@FreeBSD.ORG>
- John Hay <jhay@FreeBSD.ORG>
- Wolfgang Helbig <helbig@FreeBSD.ORG>
- Guy Helmer <ghelmer@cs.iastate.edu>
- Eric L. Hernes <erich@FreeBSD.ORG>
- Nick Hibma <n\_hibma@FreeBSD.ORG>
- Seiichirou Hiraoka <flathill@FreeBSD.ORG>
- Howard F. Hu <foxfair@FreeBSD.ORG>
- Tatsumi Hosokawa <hosokawa@FreeBSD.ORG>
- Jeffrey Hsu <hsu@FreeBSD.ORG>
- Matthew Hunt <mph@FreeBSD.ORG>
- Jun-ichiro Itoh <itojun@itojun.org>
- Matthew Jacob <mjacob@FreeBSD.ORG>
- Gary Jennejohn <gj@FreeBSD.ORG>
- Nate Johnson <nsj@FreeBSD.ORG>
- L Jonas Olsson <ljo@FreeBSD.ORG>
- Takenori KATO <kato@FreeBSD.ORG>

- Andreas Klemm <andreas@FreeBSD.ORG>
- Motoyuki Konno <motoyuki@FreeBSD.ORG>
- Joseph Koshy <jkoshy@FreeBSD.ORG>
- Jun Kuriyama <kuriyama@FreeBSD.ORG>
- Greg Lehey <grog@FreeBSD.ORG>
- Jonathan Lemon <jlemon@FreeBSD.ORG>
- Don “Truck” Lewis <truckman@FreeBSD.ORG>
- Warner Losh <imp@FreeBSD.ORG>
- Scott Mace <smace@FreeBSD.ORG>
- Stephen McKay <mckay@FreeBSD.ORG>
- Kirk McKusick <mckusick@FreeBSD.ORG>
- Kenneth D. Merry <ken@FreeBSD.ORG>
- Hellmuth Michaelis <hm@FreeBSD.ORG>
- Ted Mittelstaedt <tedm@FreeBSD.ORG>
- Atsushi Murai <amurai@FreeBSD.ORG>
- Mark Murray <markm@FreeBSD.ORG>
- Masafumi NAKANE <max@FreeBSD.ORG>
- Alex Nash <alex@freebsd.org>
- Mark Newton <newton@FreeBSD.ORG>
- Robert Nordier <rnordier@FreeBSD.ORG>
- David Nugent <davidn@blaze.net.au>
- David O’Brien <obrien@FreeBSD.ORG>
- Daniel O’Callaghan <danny@FreeBSD.ORG>
- L Jonas Olsson <ljo@FreeBSD.ORG>
- Steve Passe <fsmp@FreeBSD.ORG>
- Sujal Patel <smpatel@FreeBSD.ORG>
- Bill Paul <wpaul@FreeBSD.ORG>
- Joshua Peck Macdonald <jmacd@FreeBSD.ORG>
- Wes Peters <wes@FreeBSD.ORG>

- Steve Price <steve@FreeBSD.ORG>
- Mike Pritchard <mpp@FreeBSD.ORG>
- Doug Rabson <dfr@FreeBSD.ORG>
- James Raynard <jraynard@freebsd.org>
- Darren Reed <darrenr@FreeBSD.ORG>
- Geoff Rehmet <csgr@FreeBSD.ORG>
- Martin Renters <martin@FreeBSD.ORG>
- Paul Richards <paul@FreeBSD.ORG>
- Ollivier Robert <roberto@FreeBSD.ORG>
- Chuck Robey <chuckr@FreeBSD.ORG>
- Guido van Rooij <guido@FreeBSD.ORG>
- Dima Ruban <dima@FreeBSD.ORG>
- Kenji SADA <sada@FreeBSD.ORG>
- Wolfram Schneider <wosch@FreeBSD.ORG>
- Andreas Schulz <ats@FreeBSD.ORG>
- Justin Seger <jseger@freebsd.org>
- Hidetoshi Shimokawa <simokawa@FreeBSD.ORG>
- Vanilla I. Shu <vanilla@FreeBSD.ORG>
- Michael Smith <msmith@FreeBSD.ORG>
- Dag-Erling C. Smørgrav <des@FreeBSD.ORG>
- Brian Somers <brian@FreeBSD.ORG>
- Mike Spengler <mks@FreeBSD.ORG>
- Gene Stark <stark@FreeBSD.ORG>
- Karl Strickland <karl@FreeBSD.ORG>
- Dmitrij Tejblum <dt@FreeBSD.ORG>
- Chris Timmons <cwt@FreeBSD.ORG>
- Paul Traina <pst@FreeBSD.ORG>
- Tim Vanderhoek <hoek@FreeBSD.ORG>
- Jacques Vidrine <nectar@FreeBSD.ORG>

- Steven Wallace <swallace@FreeBSD.ORG>
- Doug White <dwhite@FreeBSD.ORG>
- Nate Williams <nate@FreeBSD.ORG>
- Kazutaka YOKOTA <yokota@FreeBSD.ORG>
- Jean-Marc Zucconi <jmz@FreeBSD.ORG>
- Archie Cobbs <archie@FreeBSD.ORG>

## The FreeBSD Documentation Project

The FreeBSD Documentation Project (<http://www.freebsd.org/docproj.html>) is responsible for a number of different services, each service being run by an individual and his *deputies* (if any):

### Documentation Project Manager

Nik Clayton <nik@FreeBSD.ORG>

### Webmaster

Wolfram Schneider <wosch@FreeBSD.ORG>

### Handbook & FAQ Editor

FAQ Maintainer <faq@freebsd.org>

### News Editor

Nate Johnson <nsj@FreeBSD.ORG>

*Deputy:* John Cavanaugh <john@FreeBSD.ORG>

### FreeBSD Really-Quick NewsLetter Editor

Chris Coleman <chrisc@vmunix.com>

### Gallery Editor

Nate Johnson <nsj@FreeBSD.ORG>

*Deputy:* Charles A. Wimmer <cawimm@FreeBSD.ORG>

Commercial Editor

Ade Barkah <mbarkah@FreeBSD.ORG>

Web Changes Editor

Ade Barkah <mbarkah@FreeBSD.ORG>

Style Police & Art Director

Chris Watson <opsys@open-systems.net>

Database Engineer

Mark Mayo <mark@vmunix.com>

CGI Engineer

Stefan Bethke <stb@FreeBSD.ORG>

Bottle Washing

Nate Johnson <nsj@FreeBSD.ORG>

LinuxDoc to DocBook conversion

Nik Clayton <nik@FreeBSD.ORG>

## Who Is Responsible for What

Principal Architect

David Greenman <dg@FreeBSD.ORG>

Documentation Project Manager (<http://www.freebsd.org/docproj/docproj.html>)

Nik Clayton <nik@FreeBSD.ORG>

Internationalization

Andrey A. Chernov <ache@FreeBSD.ORG>

Networking

Garrett Wollman <wollman@FreeBSD.ORG>

Postmaster

Jonathan M. Bresler <jmb@FreeBSD.ORG>

Release Coordinator

Jordan K. Hubbard <jkh@FreeBSD.ORG>

Public Relations & Corporate Liaison

Jordan K. Hubbard <jkh@FreeBSD.ORG>

Security Officer (<http://www.freebsd.org/security/>)

Warner Losh <imp@FreeBSD.ORG>

>Source Repository Managers (<http://www.freebsd.org/support.html#cvs>)

Principal: Peter Wemm <peter@FreeBSD.ORG>

Assistant: John Polstra <jdp@FreeBSD.ORG>

International (Crypto): Mark Murray <markm@FreeBSD.ORG>

Ports Manager (<http://www.freebsd.org/ports/>)

Satoshi Asami <asami@FreeBSD.ORG>

XFree86 Project, Inc. Liaison

Rich Murphey <rich@FreeBSD.ORG>

Usenet Support

Jörg Wunsch <joerg@FreeBSD.ORG>

GNATS Administrator (<http://www.freebsd.org/support.html#gnats>)

Steve Price <steve@FreeBSD.ORG>

Webmaster (<http://www.freebsd.org/internal/>)

Wolfram Schneider <[wosch@FreeBSD.ORG](mailto:wosch@FreeBSD.ORG)>

## Chapter 29. PGP keys

In case you need to verify a signature or send encrypted email to one of the officers or core team members a number of keys are provided here for your convenience.

### Officers

#### FreeBSD Security Officer <security-officer@freebsd.org>

```
FreeBSD Security Officer <security-officer@freebsd.org>
Fingerprint = 41 08 4E BB DB 41 60 71 F9 E5 0E 98 73 AF 3F 11
```

```
---BEGIN PGP PUBLIC KEY BLOCK---
Version: 2.6.3i
```

```
mQCNAzF7MY4AAAEAK7qBgPuBejER5HQbQlsOldk3ZVWX1Rj54raz3IbuAUrDrQL
h3g57T9QY++f3Mot2Laf5lDJbsMfWrtwPrPwCCFRYQd6XH778a+l4ju5axyjrt/L
Ciw9RrOC+WapV3lIdLuqYge2QRC1LvKACIPNbIcgbnLeRGLovFUuHi5z0oilAAUR
tDdGcmVlQlNEIFNlY3VyaXR5IE9mZmljZXXIgpPHNlY3VyaXR5LW9mZmljZXXJAZnJl
ZWJzZC5vcmc+iQCVAwUMX6yrOJgpPLZnQjrAQHyowQA1Nv2AY8vJIrdp2ttV6RU
tZBYnI7gTO3sFC2bhIHScvfVU3JphfqWQ7AntXcd2yPjGcchUfc/EcLltslqW4y7
PMP4GHZp9vHog1NAsgLC9Y1P/lcOeuhZ0pDpZZ5zxTo6TQcCBjQA6KhibFP4TJq1
3olFfPBh3B/Tu3dqmEbSWpuJAJUDBRAxez3C9RVb+45ULV0BAak8A/9JIG/jrJaz
QbKom6wMw852C/Z0qBLJy7KdN30099zMjQYeC9PnlkZ0USjQ4TSpC8UerYv6IfhV
nNY6gyF2Hx4CbEFlonpfa1c4yxtXKtilkSN6wBy/ki3SmqtfDhPQ4Q3lp63cSe5A
3aoHcjvWuqPLpW4ba2uHVKGP3g7Sst6AOYkAlQMFEDF8mz0ff6kIA1j8vQEBmZcd
/REaUPDRx6qrlXRQlms6pfgNKEwnKmcUzQLCvKBnYYGmD5ydPLxCPsFnPcPthaUb
5zVgMTjfs2fkeiRrua4duGRgqN4xy7VRAsIQeMSITBOZeBZZf2oa9Ntidr5PumS
9uQ9bvdFwMpsmk2MaRG9BSoy5Wvy8VxROYUwpt8Cf2iQCVAwUMXsyqWtaZ42B
sqd5AQHKjAQAvolI30Nyu3IyTfNeCb/DvOe9t1On/o+VUDNjIE/PuBe1s2Y94a/P
BfcohpKC2kza3NiW6lLTp00OWQsuu0QAPc02vYOyzeZWY4y3Phnw60pWzLcFdemT
0GiYS5Xm1o9nAhPFciybn9jlq8UadIlIq0wbqWgdInBT8YI/l4f5sf6JAJUDBRAx
ezKXVS4eLnPSiKUBAc5OBACIXt1KqQC3B53qt7bNMV46m81fuw1PhKaJEI033mCD
ovzyEFFQeOyRXeu25Jg9Bq0Sn37ynISuchSmt2tUD5W0+p1MUGyTqnfqeJMUWBzO
v4Xhp6a8RtDdUMBOTtrol6iulGiRrCKxzVgEl4i+9Z0ZiE6BWLg5AetoF5n3mGk1
lw==
=ipyA
---END PGP PUBLIC KEY BLOCK---
```

**Warner Losh <imp@FreeBSD.ORG>**

```

Warner Losh <imp@village.org>
aka <imp@freebsd.org>
Fingerprint = D4 31 FD B9 F7 90 17 E8 37 C5 E7 7F CF A6 C1 B9
---BEGIN PGP PUBLIC KEY BLOCK---
Version: 2.6.2

mQCNazDzTiAAAAEEAK8D7KWEbVfUrmIqHEnAvphNIqHEbqqT8s+c5f5c2uHtlcH
V4mV2TlUaDSVBN4+/D70oHmZc4IgiQwMPCWRrSezg9z/MaKlWHaslc8YT6Xclq+o
EP/fAdKUrq49H0QQbkQk6Ks5wKW6v9AOvdmsS6ZJEcet6d9G4dxynu/2qPvHAAUR
tCBNLibXYXJuZuXIgTG9zaCA8aWlwQHZpbGxhZ2Uub3JnPokAlQMFEDM/SK1VLh4u
c9KIpQEBFPsD/ln0YuuUPvD4CismZ9bx9M84y5sxLolgfEfp9Ux196ZSeaPpkA0g
C9YX/IyIy5VHh3372SDWN5iVSDYPwtCmZziwIV2YxzPtZw0nUu82P/Fn8ynlCSWB
5povLZmgrWijTJdnUWI0ApVBUTQoiW5MyrNN51H3HLWXGoXMgQFZXKWyIQCVAwUQ
MzmhkfUVW/uOVCldAQG3+AP/TlHL/5EYF0ij0yQmNTztlcLt0ble3N3zn/wPFFWs
BfrQ+nsv1zw7cEgxLtktk73wBGM9jUIIdJu8phgLtl5a0m9UjBq5oxrJaNJr6UTxN
a+sFkapTLTlg84UFUO/+8qRB12v+hZr2WeXMYjHAFUT18mp3xwjW9DUV+2fW1Wag
YDKJAJUDBRAzOYK1s1pi6lmfMj0BARBbA/930CHswOF0Hir+4YYUslejDnZ2J3zn
icTZhl9uAfeQq++Xor1x476j67Z9fESxyHltUxCmwxsJluOJRwzjyEoMlyFrIN4C
de0C8g8BF+sRt7VLURLERvlBvFrVZueXSnXvmMoWFnqpSpt3EmN6TNaLe8Cm87a
k6EvQy0dpnkPKokAlQMFEDD9Lorccp7v9qjlYQEBRUD/3N4cCMWjzsIFp2Vh9y+
RzUrblyF84tJyA7Rr1p+A7dx7je3Zx5QMEXosWl1WGN5vC9YH2WZwv6sCU61gU
rSy9z8KH1BEHh+Z6fdRMrjd9byPf+n3cktT0NhS23oXB1ZhNZcB2KKhVPlNctMqO
3gTYx+Nl06xqjR+J2NnBYU8p =7fQV
---END PGP PUBLIC KEY BLOCK---

```

**Core Team members****Satoshi Asami <asami@FreeBSD.ORG>**

```

Satoshi Asami <asami@cs.berkeley.edu>
aka <asami@FreeBSD.ORG>
Fingerprint = EB 3C 68 9E FB 6C EB 3F DB 2E 0F 10 8F CE 79 CA

---BEGIN PGP PUBLIC KEY BLOCK---
Version: 2.6.2

mQCNazPVyoQAAAEAL7W+kipxB171Z4SVyyL9skaA7hG3eRsSOWk7lfvUBLtPog
f30KwrApoc/jwLf4+Qpdzv5DLEt/6Hd/c1skhJ+q1gMNHyz5ABmUxrTRRNvJMTrb

```

```

3fPU3oZj7sL/MyiFaT1zF8EaMP/iS2ZtcFsbY0qGeA8E/58uk4NA0SoeCNiJAAUR
tCVTYXRvc2hpIEFzYWlpIDxhc2FtaUBjcy5iZXJrZWxleS5lZHU+iQCVAwUQM/AT
+EqGN2HYnOMZAQF1lQP/esXb2FuTblyX5yoolIm8YnIk1SEgCGbyEbOMMBznVNDy
5g2TAD0ofLxPxy5Vodjg8rf+lFMVtO5amUH6aNcORXRncE83T10JmeM6JEp0T6jw
zOHKz8jRzygYLBaYGsNIJ4BGxa4LeaGxJpO1ZEvrLnkPH/YEXK5oQmq9/DlrtYOJ
AEUDBRAz42JT8ng6GBbVvu0BAU8nAYCsJ8PiJpRUGlrz6rxjX8hqM1v3vqFHLcG+
G52nVMBSy+RZBgzYIPwI5EZtWAKb22JAJUDBRAz4QBWdbtuOHa j97EBaaQPA/46
+NLUp+Wubl90JoonoXocwAg88tvAUVSzszPXj0lvypAiSi2AJKsmn+5PuQ+/IoQy
lywRsexiQ5GD7C72SZ1yw2WI9DWFaAi+qa4b8n9fcLYrnHpyCY+zxEpu4pam8FJ7H
JocEUZz5HRoKKOLHERzXDiuTkkm72b1g1mCqAQvnB4kAlQMFEDEPZ3gyDQNEqHgJY
iQEBFFUEALu2C0uo+1Z7C5+xshWRY5xNCzK2006bANVJ+CO2fih96KhwsMof3lw
fDso5HJSwgFd8WT/sR+Wwzz6BAE5UtgsQq5GcsdYQuGIlylCYUpDp5sgswNm+OA
bX5a+r4F/ZJqrqTlJ56Mer0VVsNfe5nIRsjd/rnFAFVfjcQtaQmjiQCVAwUQM9uV
mcdm8Q+/vPRJAQELHgP9GqNiMpLQlZig17fDnDJ73P0e5t/hRLEfehZD1mEI2TK7j
YeQbw078nZgyyuljZ7YsbstRIsWVCxobX5eH1kX+hIxuUqCAkCsWUY4abG89kHJr
XGQn6X1CX7xbZ+b6b9jLK+bJKFcLSfyqR3M2eCyscSiZYkWKQ5l3FYvbUzkeB6K0
IVNhdG9zaGkgQXNhbWkgPGFzYWlpQEZYZWVCU0QuT1JHPg==
=39SC
---END PGP PUBLIC KEY BLOCK---

```

## Jonathan M. Bresler <jmb@FreeBSD.ORG>

```

Jonathan M. Bresler <jmb@FreeBSD.org>
f16      Fingerprint16 = 31 57 41 56 06 C1 40 13 C5 1C E3 E5 DC 62 0E FB

```

```

---BEGIN PGP PUBLIC KEY BLOCK---
Version: PGPfreeware 5.0i for non-commercial use

```

```

mQCNAzG2GToAAAEFANI6+4SJAAGBpl53XcfEr1M9wZyBqC0tzpie7Zm4vhv3hO8s
o5BizSbcJheQimQiZAY4OnlrCpPxi jMFSaihshs/VMAz1qbisUYAMqwGEO/T4QIB
nWNo0Q/qOniLMxUrxS1RpeW5vbghErHBKUX9GVhxbiVfbwc4wAHbXdKX5jjdAAUR
tCVKb25hdGhhbiBNLiBCcmVzbGVyIDxqbWJARnJlZUJTRC5PUkc+iQCVAwUQNbtI
gAHbXdKX5jjdAQHamQP+OQr10QRknamIPmuHmFYJZ0jU9XPIvTTMuOiUYLcXlTdn
GyTUuzhbEywgtOldW2V5ia8platXThtqC68NsnN/xQfHA5xmFXVbayNKn8H5stDY
2s/4+CZ06mmJfqYmONF1RCbUk/M84rVT3Gn2tydsxFh4Pm321f4WREZWRiLqmw+J
AJUDBRA0DfF99RVb+45ULV0BACz0BACCydiSUG1VR0a5DBcHdtin2iZMPsJUUPRqJ
tWvP6VeI8OFpNWQ4LW6ETAvn35HxV2kCcQMyht1kMD+KEJz7r8Vb94TS7KtZnNvk
2D1XUx8Locj6xel5c/Lnzlnnp7Bp1Xbj2u/NzCaZQ0eYBdP/k7RLYBYHQqln5x7
BOuiRJNVU4kAlQMFEDEQLcShVLh4uc9KIPEBjv4D/3mDrD0MM9EYOVuyXik3UGVI
8quYNA9ErVcLdt10NjYc16VI2H0nYVgPRag3Wt7W8wlXShpokfC/vCnT7f5JgRf8
h2a1/MjQxtld+4/Js8k7GLa53oLon6YQYk32IEKexoLPwIRO4L2BHwa3GzHJJP2
aTR/Ep90/pLdaOu/oJDUiQCVAwUQMqyL0LNaYutZnzI9AQF25QP9GFxhBrz2tiWz
2+0gWbpcGNnyZbfsVjF6ojGDdmsjJMyWCGw49XR/vPKYIJY9EYo4t49GIa jRkISQ

```

```

NNiIz22fBAjT2uY9YlvnTJ9NJleMfHr4dybo7oEKYMWwi jQzGjqf2m8wf9OaaofE
KwBX6nxcRbKsxm/BVLKczGYl3Xt jkcuJAJUDBRA1o15TZWCprDT5+dUBATzXA/9h
/ZUuhoRKTWViaistGJfWi26FB/Km5nDQBr/Erw3XksQCMwTLyEugg6dahQ1u9Y5E
5tKPxbB69eF+7JXVHE/z3zizR6VL3sdRx74TPacPsdhZRjChEQc0htLLYAPk JrFP
VAzAlSlm7qd+MXf8fJovQs6xPtZJXukQukPNlhqZ94kAPwMFEDSH/kF4tXKgazlt
bxECfk4AoO+VaFVfguUkWX10pPSSfvPyPKqiAJ4xn8RSIelttmnqkkDMhLh00mKj
lLQuSm9uYXRoYW4gTS4gQnJlc2xlcia8Sm9uYXRoYW4uQnJlc2xlcckBVU2kubmV0
PokAlQMFEDXbdSkB213Sl+Y43QEBV/4D/RLJNTrtAqJlATxXWv9g8Cr3/YF0GTmx
5dIrJOpBup7eSSmiM/BL9Is4YMsoVbXCI/8TqA67TMICvq35PZU4wboQB8DqBAR+
gQ8578M7Ekw1OAF6JXY6AF2P8k7hMcVBcVOACELPT/NyPNByG5QRDoNm1sokJaWU
/21s4QSBZZ1b
=zbCw
---END PGP PUBLIC KEY BLOCK---

```

## Andrey A. Chernov <ache@FreeBSD.ORG>

```

Andrey A. Chernov <ache@FreeBSD.org>
    aka <ache@nagual.pp.ru>
Key fingerprint = 33 03 9F 48 33 7B 4A 15 63 48 88 0A C4 97 FD 49

---BEGIN PGP PUBLIC KEY BLOCK---
Version: 2.6.3ia

```

```

mQCNAiqUMGQAAAEAPGhcD6A2Buey5LYz0sphDLpVgOZc/bb9UHABA GKUAGXmafS
Dcb2HnsuYGGX/zrQXuCiwIGtXcZWB97APTkOhFsZnPindR5n/dde/mw9FnuhwqD
m+rKSL1HlN0z/Msa5y7g16760wHhSR6NoBSEg5wQAHIMMq7Q0uJgpPLZnQjrAAUT
tCVBbmRyZXkgQS4gQ2hlcm5vdiA8YWN0ZUBuYwd1YWwucHAucnU+iQCVAwUQM2Ez
u+JgpPLZnQjrAQEYugP8DPnS8ixJ5OeuYgPFQf5sy6l+LrB6hyaS+lgSUpahWjNY
cnaDmfda/q/BV5d4+y5r1Qe/pjnYG7/yQuAR3jhlXz8XDrqlBOnW9AtYjDt5rMfJ
aGFTGXAPGZ6k6zQZE0/Yurt8ia3qjvuzm3Fw4NjrHRx7ETHRvVJDvxA6Ggsvmr20
JEFuZHJleSBBLiBdaGVybm92IDxhY2hlQEZyZWVUCU0Qub3JnPokAlQMFEDR5uVbi
YKTy2Z0I6wEBLgED/2mn+hw4/3peLx0Sb9LNx//NfCCKVefSf2G9Qwhx6dvwbX7h
mFca97h7BQN4GubU1Z5Ffs6TeamSBrotBYGmOCwvJ6S9WigF9YHQIQ3B4LEjskAt
pcjU583y42zM11kkvEuQU2Gde61daIylJyOxsgpjSWpkxq50fgY2kLMfgl/ftCZB
bmRyZXkgQS4gQ2hlcm5vdiA8YWN0ZUBuYwd1YWwucnU+iQCVAwUQMcm5HeJgpPLZ
nQjrAQHwvQP9GdmAflgdcuayHEgNkc1lmacPH1lcwWjYjzA2YoecFMGV7iqKK8QY
rr1MjbgXf8DAG8Ubfm0QbI8Lj8iG3NgqIru0c72UuHGSn/APfGGG0AtPX5UK/k7B
gIOCa2po6NA5nrSp8tDsdEz/4gyea84RXl2prtTf5Jj07hflbRstGXX0MkFuZHJl
eSBBLiBdaGVybm92LCBCbGFjayBNYWdlIDxhY2hlQGFzdHJhbC5tc2suc3U+iQCV

```

```

AwUQMCsAo5/rGryoL8h3AQHq1QQAidyNFqA9hvrMmcjpY7csJVf1Gvj574Wj4GPa
o3pZeuQaMBmsWqaXLynWU/Aldb6kTz6+nRcQX50zFH0THSPfApwEW7yybSTI5apJ
mWT3qhKN2vmLNg2yNzhqLTzHLD1lH3i1pfQq8WevrNfjLUco5S/VueKTma/osnzC
Cw7fQzCJAJUDBRAwKvwoalpnjYgyp3kBARihBACoXr3qfG65hFCyKJISmjOvaoGr
anxUIkeDS0yQdTHzhQ+dwB1OhhK15E0NwrOMKajLMm90n6+Zdb5y/FIjppriu8dI
rlHrWZlewa88eEDM+Q/NxT1iYg+HaKDAE171jmLpSpCL0MiJt00i36L3ekVD7Hv8
vffOZHPShirIzJOZTYkAlQMFEDAau6zFLUdtDb+QbQEBOQX8D/AxwkYeFaYxZYMFO
DHIvSk23hAsjCmUA2UillFeWAusb+o8xRfPdc7TnosrIifJqbF5+fcHCG5VSTG1h
Bhd18YwUeabf/h9O2BsQX55yWRuB2x3diJlxI/VVdG+rxlMCmE4ZR1Tl9x+Mtun9
KqKvPb39V1kCBYQ3hlgNt/TJUY4riQCVawUQMBHmmyJR1tlmbQBRAQFQkwP/YC3a
hs3ZMMoriOlt3ZxGNUUPTF7rIER3j+c7mqGG46dEnDB5sUrzkacpoLX5sjltGR3b
vz9a4vmk1Av3KFNnvrrZZ3/BZFGpq3mCTiAC9zsyNYQ8L0AfGIUO5goCIjqwOTNQI
AOPnsJ5S+nMAkQB4YmmNlI6GTb3D18zfhPZ6uciJAJUCBRAwD0sl4uW74fteFRkB
AWsAA/9NYqBRBKbmltQDpyK4+jBAYjkXBjMARFXXJYtlnTgOHMpZqoVyW96xnaa5
MzxEiu7ZWm5oL10QDIp1krkBP2KcmvfSMMHb5aGCCQc2/P8NlfXAuHtNGzYiIOUA
Iwi8ih/S1liVfvngF9uV3d3koE7VsQ9OA4Qo0ZL2ggW+/gEaYIkaLQMFEDA0z6qx
/IyHe3rl4QEBivYD/jIr8Xqo/2I5gncghSeFR01n0vELFIvaF4cHofGzyzBpYsfa
+6pgFI1IM+LUF3kbUkAY/2uSf9U5ECcaMCTWCwVgJVO+oG075SHEM4buhrzutZiM
ldTyTaepaPpTyRMUux9ZMMYjs7sbqLIdleDwrJxUPhrBNvf/w2W2sYHSY8cdiQCV
AwUQMAzqgHcdkq6JcsfBAQGTxwQatgeLfi2rhSodllpDXUwz+SS6bejFTWGRsWFM
y9QnOcqryw7LyuFmWein4jasjY033JsODfWQPipVNA3UEnXVg9+n8AvNMP08JkRv
CnleNg0VaJy9J368uArio93agd2Yf/R5r+QEuPjIssV8hdcy/luEhSiXWf6bLMV
HEA0J+OJAJUDBRAwDui+4mCk8tmdCOsBAatBBACHB+qtW880seRCDZLjl/bT1b14
5po60U7u6a3PEBKY0NA72tWDQuRPF/Cn/0+VdFNxQUsgkrbwaJWooiOKQsvlOm3R
rsxKbn9uvEKLxExyKH3pxp76kvz/1EWwEeKvBK+84Pb11zpg3W7u2XDfi3VQPTi3
5SZMAHc6C0ct/mjN1YkAlQMFEDAMrPD7wj+NstMUOQEBJckD/ik4WsZzm2qOx9Fw
erGq7Zwchc+JqlYeN5PxpzqSf4AG7+7dFin+oe6X2FcIzgbYY+IfmgJIHEVjdHH5
+uAXyb6l4iKc89eQawO3t88pfHLJWbTzmnvgz2cMrxt94HRvgkHfvcpgEGgbyldq6
EB33OunazFcfZFRicXk1sfyLDvYE
=1ahV
---END PGP PUBLIC KEY BLOCK---

```

## Jordan K. Hubbard <jkh@FreeBSD.ORG>

```

Jordan K. Hubbard <jkh@FreeBSD.org>
Fingerprint = 3C F2 27 7E 4A 6C 09 0A 4B C9 47 CD 4F 4D 0B 20

---BEGIN PGP PUBLIC KEY BLOCK---
Version: 2.6.3ia

```

```

mQCNAzFjX0IAAAEEAML+nm9/kDNPP43ZUZGjYkm2QLtoC1Wxr8JulZXqk7qmhYcQ
jvX+fyoriJ6/7ZlnLe2oG5j9tZOnRLPvMaz0g9CpW6Dz3nkXrNPkmoFV9B8D94Mk
tyFeRjFqncCuqBj6D+H8FtBwEeeTecSh2tJ0bZTXnAMhxeOdvUVW/uOVC1dAAUR

```

```
tCNKb3JkYW4gSy4gSHViYmFyZCA8amtOQEzyZWVCU0Qub3JnPokBFQMFEDXCTXQM
j46yp4IfPQEBwO8IAIN0J09AXBf86dFUTFGcAMrEQqOF5IL+KGorAjzuYxERhKfD
ZV7jA+sCQqxkWfcVcE20kVyVYqzZIkio9a5zXP6Twa247JkPt54S1PmMDYHNlRIY
laXlNoji+4q3HP2DfHqXRT2859rYpm/fG/v6pWkos5voPKcZ2OFep9W+Ap88oqw+
5rx4VetZnJq1Epmis4INj6XqNqj85+MOOIYE+f445ohDM6B/Mxazd6cHFGGIR+az
VjZ6lCDMLjzhB5+FqfrDLyUmjqkMTR5z9DL+psUvPlCkYbQ11NEWtEmiIWjUcNjN
GCxGzv5bXk0XPu3ADwbPkFE2usW1cSM7AQFiwuyJAJUDBRAxe+Q9a1pnjYGyp3kB
AV7XA/oCSL/Cc2USpQ2ckwkGpyvIkYBPsZicabSNJAzm2hsU9Qa6WOPxD8olDdB
uJNiW/gznPC4NsQ0N8Zr4IqRX/TDvF04WhLmd8AN9SorVv2q0BKgU6fLuk979tJ
utrewH6PR2qBOjAaR0FJNk4pcYAHet+e7KaKy96YFvWkIyDvc4kAlQMFEDF8ldof
f6kIA1j8vQEBDH4D/0Zm0oNlpXrAE1EOFrmp43HURHbiJ8n0Gra1w9sbfo4PV+/H
U8ojTdwly6r0+prH7NODckgtIQNpqLuqM8PF2pPtUJj9HwTmSqfaT/LMztfPA6PQ
csyT7xxdXl0+4xTD1lavGSJfYsI8XCAy85cTs+PQwuyzugE/iykJ0lBnj/paiQCV
AwUQMxv1BvUVV/uOVC1dAQF2fQP/RfYC6RrpFTZHjo2qsUHSRk0vmsYfwG5NHP5y
oQBMsaQJESckN4n2J0gr4T75U4vS62afXgPLJP3lOHkU2Vc7xhAuBvsbGr5RP8c5
LvPOeUEyz6ZArp1KUHrtcm2iK1FB0mY4dOYphWyWMkDgYExabqlrAq7FKZftpq/C
BiMRuaw=
=C/Jw
---END PGP PUBLIC KEY BLOCK---
```

## Poul-Henning Kamp <phk@FreeBSD.ORG>

```
Poul-Henning Kamp <phk@FreeBSD.org>
Fingerprint = A3 F3 88 28 2F 9B 99 A2 49 F4 E2 FA 5A 78 8B 3E
```

```
---BEGIN PGP PUBLIC KEY BLOCK---
Version: 2.6.3ia
```

```
mQCNAzAdpMIAAAEEALHDgrFUwhZtb7PbXg3upELoDVEUPFRwnmpJH1rRqyROUGcI
ooVe7u+FQlIs5OsXK8ECs/5Wpe2UrZSzhVjwBYOND5H42YtI5UULZLRCo5bFftVA
K9Rpo5icfTsYihrzU2nmnycwFMk+jYXyT/ZDYWDP/BM9iLjj0x9/qQgDWPY9AAUR
tCNQb3VsLUhlbm5pbmCGS2FtcCA8cGhrQEzyZWVCU0Qub3JnPokAlQMFEDQ0aZl
u244dqP3sQEBu4ID/jXFFeJgs2MdTDNOZM/FbfDhI4qxAbYUsqS3+Ra16yd8Wd/A
jV+IHJE2NomFWl8UrUjCGinXiWzPgK1OfFjrS9OglwQLvAl0X84BA8MTP9BQR4w7
6I/RbksgUSrVCI08MJwlydjSPocWGBex1VjbxXzyuJk7H+TG+zuI5BuBcNIiQCV
AwUQMwYr2rNaYutZnzI9AQHiIQP/XxtBWFxAbRgVLEhRNpS07YdU+LsZGLLOZehN
9L4UnJFHQQPNOpMey2gF7Y95aBOW5/1xS5v1QpwmRfCnTWsm/gqdzK6rulf1r5A
y94LO5TAC6ucNu396Y4vo1TyD1StnRC466KlvmTQtAtFGgXlORWLL9URLzCRfd1h
D0yXd9aJAJUDBRAxf019a1pnjYGyp3kBAQqyA/4v64vP31lF0Sadn6ias761hkz/
SMDtuLzILmofSCC4o4KWMjiWJHs2Soo41QlZil+xMHZv32JKiwFlGtPHqL+EHYXy
Q4H3vmf9/1KF+0XCamtgI0wWUMziPSTJK8xXbRRmMDK/0F4TnVVaUhnmf+h5K7O6
XdmejDTa0X/NWcicmIkAlQMFEDF81ef1FVv7j1QtXQEBcnwD/0rolPpUtlkLmred
tsGTkNa7MFLegrYRvDDrHOWPZH152W2jPUncY+eArQJakeHiTDmJNpFagLZglhE0
```

```

bqJyca+UwCXX+6upAclWHEBMg2byiWMMqyPVEEnpUoHm1sIkgdNWlfQAmipRBFYh
2LyCgWvR8CbtwPYIFvUmGgB3MR87iQCVAwUMUSeXB9/qQgDWPY9AQGPkwP/WEDy
E12Gkvua9CotMAifot2vTwuvWwPnOpIEx0Ivey4aVbRLD90gGCJw8OGDEtqFPcNV
8aIiy3fYVKXGZZjvCKd7zRfhNmQn0eLDcymq2OX3aPrMc2rRlkt4Jx425ukRlgsO
qiQA9w91aWhY8dlw/EKzk8oJm52x4VgXaBACMjaJAJUDBRAxOUOg72G56RHVjtUB
AbL4A/9HOn5Qa0lq9tKI/HkSdc5fGQD/66VdCBAb292RbB7CS/EM07MdbcqRRYIa
0+0gWQ3OdsWPdCVgH5RIhp/WiC+UPkR1cY8N9Mg2kTwJfZZfnqN+BgWlgRMPN27C
OhYN18Q33N19CpBLrZWABF44jPeT0EvvTzP/5ZQ7T75EsYKYiYkAlQMFEDDmryQA
8tkJ67sbQQEBPdsEALCj6v10BuJLLJTLxmmrkqAZPVzt5QdeO3Eqa2tcPWcU0nqP
vHYMzZcZ7oFg58NZsWrhSQQDIB5e+K65Q/h6dC7W/aDskZd64jxtEznX2kt0/MOR
8OdsDis1K2f9KQftrAx81KmVwW4Tqtz17NWTDXt44fMOTibCwVq8v2DFkTjY
=JKbP
---END PGP PUBLIC KEY BLOCK---

```

## Rich Murphey <rich@FreeBSD.ORG>

```

Rich Murphey <rich@FreeBSD.org>
fingerprint = AF A0 60 C4 84 D6 0C 73 D1 EF C0 E9 9D 21 DB E4

---BEGIN PGP PUBLIC KEY BLOCK---
Version: 2.6.2

```

```

mQCNay97V+MAAAEEALiNM3FCwm3qrCe81E20UOS1NclOWfZHNAyOyjlAhHeINvol
FBF2Gd5Lbj0y8SLMno5yJ6P4F4r+x3jwHZrzAIwMs/lxDXRtB0VeVwnlj6a3Rezs
wbfaTeSvyh5JohEcKdoYiMG5wjATOWK/NAwIPthB1RzRjnEeer3HI3ZYNEOpAAUR
tCRSaWNoIE11cnBoZXkgPHJpY2hAbGfTcHJleS51dG1iLmVkdT6JAJUDBRAve15W
vccjdlg0Q06kBAZTZBACcNd/LiVnMFURPrO4pVRn1sVQeokVX7izeWQ7sie31Iy7g
Sb97WRLEYDi686osaGfsuKNA87Rm+q5F+jxeUV4w4szoqp60gGvCbD0KCB2hWraP
/2s2qdVAXhfcoTin/Qp1ZWvXxFF7imGA/IjYifB42VkaRYu6BwLEm3YAGfGcSw==
=QoiM
---END PGP PUBLIC KEY BLOCK---

```

## John Polstra <jdp@FreeBSD.ORG>

```

John D. Polstra <jdp@polstra.com>
Fingerprint = 54 3A 90 59 6B A4 9D 61 BF 1D 03 09 35 8D F6 0D

---BEGIN PGP PUBLIC KEY BLOCK---
Version: 2.6.2

```

```

mQCNazMELMEAAAEALizp6ZW9QifQgWoFmG3cXhzQ1+Gt+a4SladC/TdHdBvwlM/

```

```

I6Ok7TC0dKF8blW3VRgeHo4F3XhGn+n9MqIdboh4HJC5Iiy63m98sVLJSwyGO4oM
dkEGyyCLxqP6h/DU/tzNBdqFzetGtYvU4ftt3R00a506cr2CHcdm8Q+/vPRJAAUR
tCFKb2huIEQuIFBvbHN0cmEgPGpkcEBwb2xzdHJhLmNvbT6JAJUDBRAzBNBE9RVb
+45ULV0BAWgiA/0WWO3+c3qlptPCHJ3DFm6gG/qNKsY94agL/mHOr0fxMP5l2qKX
O6albWkvGoYq0EwoKGFfn0QeHiCl6jVi3CdBX+W7bObMcoi+foqZ6zluOWBC1Jdk
WQ5/DeqQGYXqbYjqO8voCSctAPge3XlMwVpMZTv24u+nYxtLkE0ZcwtY9IkAlQMF
EDMEt/DHZvEPv7z0SQEBXh8D/2egM5ckIRpGz9kcFTDC1gdWWtlgwCl1iI2p9gEhq
aufy+FUJlZS4GSQQLWB0BlrTmDC9HuyQ+KZqKFRbVZLyzkH7Wfs4zDmwQryLV5wkN
C4BRRBXZfWY8s4+zT2WQD1aPO+ZsgRauYlkJgTvXTPU2JCN62Nsd8R7bJS5tuHEm
7HGmiQCVaUwQMwSvHB9/qQGDWPY9AQFAhAQAgJlAlbKITrEoJ0+pLIsOV3eQ348m
SVHEBGikU3Xznjr8NzT9aYtq4TIzt8jplqP3QoVlkaLyPzF0NjvfZ+ffYp/sIaU
wPbEpgtmHnVWJAebMbNs/Adlw8GDvxet9IaCbMjGZnHmfnEqOBIx7VBDPHHoJxM
V31K/PIoYsHAY5w=
=cHFa
---END PGP PUBLIC KEY BLOCK---

```

## Guido van Rooij <guido@FreeBSD.ORG>

```

Guido van Rooij <guido@gvr.win.tue.nl>
Fingerprint = 16 79 09 F3 C0 E4 28 A7 32 62 FA F6 60 31 C0 ED

---BEGIN PGP PUBLIC KEY BLOCK---
Version: 2.6.2

```

```

mQCNAzGeO84AAAEAKKAY91Na//DXwlUusr9GVESslVwVP6DyH1wcZXhfN1fyZHq
SwmMCEdHYoojQds+VqDliiZQvv1RLByBgj622PDAPN4+Z49HjGs7YbZsUNuQqPPU
wRPPp6ty69x1hPKqlsQIB5MS4radpCM+4wbZbhxv714rP3RWUbnAYutZnzI9AAUR
tCZhdWlkbyB2YW4gUm9vaWogPgdlawRvQGd2ci53aW4udHVlLm5sPokAlQMFEDMG
Hcgff6kIA1j8vQEBbYgd/jm9xHuUuY+iXDKOzpCXBYACYEZDV913MjtyBAmaVqYo
Rh5HFimkGXe+rCo78Aau0hc57fFMTsJqnuWEqVt3GRq28hSK1FOZ7ni9/XibHcmN
rt2yugl3hYpCl1jo4nrDL1NxiBbamkGW/vFGcljS0jqXz6NDVbGx5Oo7HBBYxByz
iQCVaWUQMhmtVjt/x7zOdmsfAQFuVQQapsVUTigt5YwJQA9Nd5Z0+a/oVtZpyw5Z
OljLJP3vqJdMa6TidhfcatjHbFTve5xldmjFgMX/MQTD8zf/+Xccy/PX4+lnKNpP
eSflY4aK+E8KHmBGd6GzX6CIboyGYLS9e3kGnN06F2AQtaLyJFgQ71wRaGuyKmqG
FwTn7jiKb1aJAJUDBRAYeOLXpt3in6QQUSEBATwQA/9jqu0Nbk154+Pn+9mJX/YT
fYR2UqK/5FKCqgL5nt/Deg2re0zMD1f8F9Dj6vuAAxq8hnOkIHKlWo1mjkRkKzJi
mSPEWl3AuHJ31k948J8it4f8kq/o44usIA2KKVmlI63Q/rmNdfWCyiYQEVGcRbTm
GTdZIHycOgV5dOo4ebFqgYkAlQMFEDIE1nMEJn15jgpJ0QEBW6kEAKqN8XSgzTqf
CrxFXT07MlHhfdBKUTNUoboxCGCLNW05vf1A8F5fdE5i14LiwkldWIzPxWD+Sa3L
fNpCfCZTaCiyGcLyTzVfBHA18MBA00X6JiTpdcM22jLGUWbf/aJK3yz/nfbWntd/
LRHysIdVp29lP5BF+J9/Lzbb/9LxP1taiQCVaUQMgRXZ44CzbsJWQz9AQFf7gP/
Qa2FS5S6RYK3rYanWADVe/ikFV2lxuM1azlWbsmljXvKVWGe6cV693nS5lGGAjx
lbd2ADwXjlkNhv45HLWfM9PEve09Jjr6tMuxVt8N2pxiX+1PLUN9CtphTIU7Yfjn

```

```
s6ryZZfwGHSfIxNGi5ua2SoXhg0svaYnxHxXm0tH24iJAJUDBRAyAkpV8qaAEa3W
TBkBARfQBAC+S3kbulEAN3SI7/A+A/dt19DfZezT9C4SRBGsl2clQFMGIXmMQ/7v
71LXrKQ7U2zVbgNFU8smw5h2vBIL6f1PyexSmc3mz9JY4er8KeZpcf6H0rSkH1+i
d7TF0GvuTdnPFO8hc9En+GG6QHOqbkB4NRZ6cwtfwUMhk2FHXBnjf4kAlQMFEDH5
FFukUJAsCdPmTQEBe74EAMBSxDnbD9cuI5MfF/QeTNEG4BIVUZtAkDme4Eg7zvsP
d3DeJKCGEnjicWYrRTCGwaCWzMQk+/+MOMdkI6Oml+AIurJLoHceHS9jPlizdP7f
N2jkdeJSBsixunbQWtUElSgOQQ4iF5kqwBhxtOfEP/L9QsoydrMR1yB6WPD75H7V
iQCVAwUQMZ9YNGtaZ42Bsqd5AQH0PAQAhpVlAc3ZM/KOTywBSh8zWKVlSk3q/zGn
k7hJmFThnlhH1723+WmXE8aAPJi+VXOWJUFQgwELJ6R8jSU2qvk2mlVWYYSqRKvc
VRQMqT2wjss0GE1Ngg7tMrkRHT0il7E2xxIb8vMrIwmdkbTfYqBUhhGnsWPHZHq7
MoAl/b+rK7CJAJUDBRAxnvXh3IDyptUyfLkBAYTDA/4mEKlIP/EUX2Zmxgrd/JQB
hqcQlkTrBAaDonOqe/4oewMKR7yaMpztYhJs97i03Vu3fgoLhDspE55ooEeHj0r4
cOdiWfYDsjsFUYSNVhW4OSruMA3c29ynMqNHD7hpr3rcCPUi7J2RncocOcCjK2
BQb/9IAUNeK4C9gPxMEZLokAlQMFEDGeO86zWmLrWZ8yPQEBEEID/2fPEUrSX3Yk
j5TJPFZ9MNx01Eo7AHYjnJgEbNI4pYm6C3PnMlsYfCSQDHuXmRQHAOWSdwOLvCkN
F8eDaF3M6u0urgeVJ+KVUnTz2+LZoZs12XSZKcte0HxjBvPpWMTTrYyimGezH79C
mgDVjsHaYOx3EXF0nnDmtXurGioEmWlJ
=mSvM
---END PGP PUBLIC KEY BLOCK---
```

## Peter Wemm <peter@FreeBSD.ORG>

```
Peter Wemm <peter@FreeBSD.org>
  aka <peter@spinner.dialix.com>
  aka <peter@haywire.dialix.com>
  aka <peter@perth.dialix.oz.au>
Key fingerprint = 47 05 04 CA 4C EE F8 93 F6 DB 02 92 6D F5 58 8A

---BEGIN PGP PUBLIC KEY BLOCK---
```

```
Version: 2.6.3ia
```

```
mQCNAy9/FJwAAAEALxs9dE9tFd0Ru1TXdq30lKfEoe5uYKKuldHRBOacG2Wny6/
W3I1l57hOi2+xmQ5X/mHkapywxvy4cyLdt3li4GEKDvxpDvEzAYcy2n9dIup/eg2
kEhRBX9G5k/LKM4NQsRIieaIEGGgCZRm0lINqw495aZYrPpO4EqGN2HYnOMZAAUT
tCVQZXRlciBXZwltIDxwZXRLckBoYXl3aXJlLmRyYWxpeC5jb20+iQCVAwUQMwWT
cXW7bjh2o/exAQEFkQP+LIx5zKlYpluR24xGApMFNrnTjh+iDIWnxxb2M2Kb6x4G
9z6OmbUCoDTGrX9SSL2Usm2RD0BZfyv9D9QRWC2TSOPkPRqQgIycc1lvglolJUN
eixqsxlFeKLGEx9eRQCCbo3dQIUjC2yaOe484QamhsK1nL5xpoNWI1P9zIOPDiGJ
AJUDBRAxSRPqSoY3Ydic4xkBAWLA/9q1Fdnk4unpGQsG31Qbtr4AzaQD5m/JHI
4gRmSmbj6luJMGNG3fp006Gd/Z7uxyCJB8pTst2a8C/ljOYZxWT+5uSzkQXeMi5c
YcIlsZbUpkHtmqPW623hr1PB3ZLa1TicTbQW+NzJsxQlPc6XG9fGkT9WXQW3Xhet
AP+juVTAhLQlUGV0ZXIgv2VtbSA8cGV0ZXJAcGVydGguZGlhbG14Lm96LmF1PokA
lQMFEDGxFCFKhjdH2JzjGQEB6XkD/2HOfuFrnQUtdwFPuKgtEqNeSr64jQ3Maz8
```

```

xgEtBaw/ymlPbhbCk311UWQq4+izZE2xktHTFClJfaMnxVifboPyuiSF99KHiWnf
/Gspet0S7m/+RXIwZilqSqvAanxMiA7kKgFSCmchzas8TQcyyXHtn/gl9v0khJkb
/fv3R20btB5QZXRlciBXZW1tIDxwZXRlckBGcmVlQlNELm9yZz6JAJUDBRAXsRJd
SoY3Ydic4xkBAZJUA/4i/NWHz5LIH/R4IF/3V3LleFyMFr5EPFY0/4mcv2v+ju9g
broEM/xd4LlPrx1XqPeZ74JQ6K9mHR64RhKR7ZJJ9A+12yr5dVqihe911KyLKab9
4qZUHYi36WQu2VtLgnw/t8Jg44fQSzbBF5q9iTzcfNOYhRkSD3BdDrC3llyw07Ql
UGV0ZXIgv2VtbsA8cGV0ZXJAc3Bpbm5lci5kaWFsaXguY29tPokAlQMFEDGxEi1K
hjdH2JzjGQEBdA4EAKmNFlj8RF9HQsoI3UabnvYqAWN5wCwEB4u+Zf8zq6OHic23
TzoKlSPlmsdBE1dXXQGS6aidkLT+xOdeewNs7nfUICH/DBjSuklAOJzKliXPQW7E
kuKNwy4eq5bl+j3HB27i+WBXhn6OaNNQY674LGar41EGq44Wo5ATcIicig/z
=gv+h
---END PGP PUBLIC KEY BLOCK---

```

## Jörg Wunsch <joerg@FreeBSD.ORG>

```

Type Bits/KeyID      Date       User ID
pub 1024/76A3F7B1 1996/04/27 Joerg Wunsch <joerg_wunsch@uriah.heep.sax.de>
      Key finger-
print = DC 47 E6 E4 FF A6 E9 8F 93 21 E0 7D F9 12 D6 4E
      Joerg Wunsch <joerg_wunsch@interface-
business.de>
      Joerg Wunsch <j@uriah.heep.sax.de>
      Joerg Wunsch <j@interface-business.de>

```

```

---BEGIN PGP PUBLIC KEY BLOCK---
Version: 2.6.3ia

```

```

mQCNAzGCFeAAAAEEAKmRBU2Nvc7nZy1Ouid61HunA/5hF4091cXm71/KPaT7dskz
q5sFXvPJPpawwvqHPHfEbAK42ZaywyFp59L1GaYj87Pda+PlAYRJyY2DJ15/7JPe
ziq+7B8MdvbX6D526sdmcr+jXPbHznASjkx9DPmK+7TgFuJyXW7bjh2o/exAAUR
tC1Kb2VyZyBXdw5zY2ggPGpvZXJnX3d1bnNjaEB1cmlhaC5oZWVwLnNheC5kZT6J
AJUDBRA0FFkBs1pi6lmfMj0BAFDCA/ocfkjrhrvRwRcPSL8klJ1YDoUJdmw+v4nJc
pw3OpYXbwKOPLClsE7K3KCQscHel7auf9lnrekAwbrXv9Clp0TegYeAQNjw5vZ9f
L6UZ513fH8E2GGA7+kqgNWSlKxAnG5GdUvJ9viyrWm8dqWRGo+loDWlZ12L2OgAD
fp7jVZTI1okAlQMFEDQPrLoff6kIA1j8vQEB2XQEAK/+SsQPCT/X4RB/PBbxUr28
GpGJMn3AafAaA3p1Yw3nb4ONbqEw9tJtofAn4UeGraiWw8nHYR2DAzoAjr6OzuX3
TtUV+57BIzrTPHcNkb6h8fPuHU+dFzr+LNoPaGJsFeov6w+Ug6qS9wa5FGDAgaRo
LHSyBxcRVocBoEaS5S5EiQCVawUQM5BktWVgqaw0+fnVAQGKPPw+OiWho3Zm2GKp
lEjiZ5zx3y8upzb+r1Qutb08jr2Ewja04hLg0fCrt6Ad3DoVqxe4POghIpmHM404
tcW92THQil70CLzfCxtfUc6edzoP3krD1/Gwpm2hGrmYA9b/ez9+r2vKBbnUhpMc
glx5pflIzHU9R2XyQz9Xu7FI2baOSZqJAJUDBRAyCIWZdbtuOHa j97EBAVMzA/41
VIph36l+yO9WGKkEB+NYbYOz2W/kyi74kXLvLdTXcRYFaCSZORSsQKPGNMpZUoL
oAKxE25AoCgl5towqr/sCcu0A0MMvJddUvlQ2T+y1SpGmWchqoXCXN7FdGyxrZ5zz

```

```

xzLIvtcio6kaHd76XxyJpltcASupdD53nEtXnu8sRrQxSm9lcmcgV3Vuc2NoIDxq
b2VyZl93dW5zY2hAaW50ZXJmYWNLWJl1c2luZXNzLmRlPokAlQMFEDIHfr1u244
dqP3sQEBWoID/RhBm+qtW+hu2fqAj9d8CVgEKJugrxZIpXuCKFvO+bCgQtogt9EX
+TJh4s8UUDcFkyEiu8CT2C3RrrlgrvckfxvrTgzSsvtYyv1072X3GkVY+S1UMBMA
rd1lqNW23oT7Q558ajnsaL065XJ5m7HacgTTikiofYG8ils7TrsEq6PtCJk2Vy
ZyBXdw5zY2ggPGpAdXJpYWguaGVlcC5zYXguZGU+iQCVAwUQMas91D4gHQUlG9CZ
AQGYOwQAhPpiobK3d/fz+jWrbQgjkO+j39g1YGXB22+6iuEprFRs/ufKYtjljNT
NK3B4DWSkyIPawcu04Lotijp6jke2bsjFSSashGWcsJlpnwsv7EeFitT3oWTTTQQ
ItPbtNyLW6M6xB+jLgtaAvJqfOlzgo9BLfHuA2LY+WvbVW447SWJAJUDBRaxqWRs
dbtuOHaJ97EBAXDBA/49rzZB5akkTSbt/gNd380JgC+H8N5da25vV9dD3KoAvXfW
fw7OxIsxvQ/Ab+rJmukrrWxPdsC+1WU1+1rGa4PvJp/VJRDes2awGrn+i07/cQoS
IVziC27JpcbvtLvLVcBIiylyT/RvJ+87a3jPRHt3VFGcpFh4KykxxSNiyGygl4kA
lQMFEDGCUB31Fv7jlQtXQEB5KgD/iIJZe5lFkPr2B/Cr7BKMVBotl/JSu05NsHg
JZ3uK15w4mVtNPZcFi/dKbn+qRM6LKDFe/GF0HZD/ZD1FJt8yQjzF2w340B+F2GG
EOwnClqZdTEAqnIBzM/ECQQqH+6Bi8gpkFZrFgg5eON7ikqmusDn0lYStM/CBfgp
SbR8kDmFtCZKb2VyZyBXdw5zY2ggPGpAaW50ZXJmYWNLWJl1c2luZXNzLmRlPokA
lQMFEDHioSdlYKmsNPn51QEByz8D/10uMrwP7MdaXnptd1XNFhpaAPYTVAOcaKlY
OGI/LLR9PiU3FbqXO+7INhaxFjBxa0Tw/p4au5Lq1+Mx8ledHniJZNS8tz3I3goi
jIC3+jn2gnVAwnK5UZUTUVUn/JLVk/oSaIJNIMMDaw4J9xPVVkb+Fh1A+XqtPsVa
YESrNp0+iQCVAwUQMwXkzcdm8Q+/vPRJAQEA4QQAgNNX1HFgXrMetDb+w6yEGQDk
JCDAY9b6mA2HNeKLQAhsOzL4HwAl+iuQaCgo3lyFC+1Sf097OUTs74z5X1vCedqV
oFw9CxI3xuctt3pJCbbN68fl0lnq0WdYouWWG1FwLlh5PEy//VtwX9lqgsizlhzi
t+fX6BT4BgKi5baDhrWJAJUDBRAyCKved9eCJxX4hUkBAebMA/9mRPy6K6i7TX2R
jUKS12p5oYrXPk12Zsw4i juktslxzQhOCyMSCGK2UEC4UM9MXp1H1JZQxN/DcfnM
7VaUt+ve0wZ6DC9gBSHJ1hKVxHe5XTj26mIr4rcXNy2XEDMK9QsnBxIAZnBVTjSO
LdhqqSMP3ULLOpB1RL2RYrqi27IXr4kAlQMFEDGpbndlu244dqP3sQEBJnQD/RVS
Azgf4uorv3fpbosI0LE3LÜufAYGBSjNjnskeKyudZkNkI5zGGDwVneH/cSkkt4OR
oeqcTBxKeMaMuXPV130QahgNwWjfuTvl5OZ8orsQGGWIn5FhqYXsKkjEGxIOBOF
vvlVQ0UbcR0N2+5F6Mb5GqrXZpIesn7jFJpkQKPU
=97h7
---END PGP PUBLIC KEY BLOCK---

```

## Developers

### Wolfram Schneider <wosch@FreeBSD.ORG>

```

Type Bits/KeyID      Date      User ID
pub 1024/2B7181AD 1997/08/09 Wolfram Schneider <wosch@FreeBSD.org>
Fingerprint = CA 16 91 D9 75 33 F1 07 1B F0 B4 9F 3E 95 B6 09

```



```
vRHkM3JURUjIVZdAQNVxxBso8NJG5KayP0Q96Vw+3sEwFK49jt14RCJy4IkAlQMF
EDNzvb1sq+iWcxFJBQEBfZwD/R3KNFf9ype9Dea8j1YIeNZ1E3e03en1I8fMj6Em
S1/L1WfFzMnfFCxZs7JgPtkBuB3CqP8f+LodDt6PHPqNakmI9E6fiuGfJZ3jFZYA
TXa0XKuIoxIJNKhqkpbF8ixJZFTxFwAAwVYM3+sqr4qQ8FzVc5entxjyxPFNkwJw
RWV+iQCVAwUQM2aiBQ7tvOdmanQhAQE7LgQAiN6Hz+zd8bh0nO6VizbJxWFRHPbr
QWnJXGoMYyy88DyszAXC4zRshlyGUDQdHeP/1DFCXDEu78GfDCLaJ1bm25yVR7kL
xDZaEUQEbWqxfiwuzizAjkaxrW7dBbWILwWqrYF5TXClw+oUU/oIUW4t6t+GpAO1
8PLYhSMXVYErrAA=
=EdyZ
---END PGP PUBLIC KEY BLOCK---
```

