OpenBSD

Frequently Asked Questions

OpenBSD FAQ

Older Versions of OpenBSD

Browse the OpenBSD website

This FAQ is specifically designed for the **2.5** release of **OpenBSD**, but in some cases will work on earlier versions. Information on previous versions is still available and you should always check http://www.openbsd.org/errata.html for important updates. The FAQ only follows release versions of OpenBSD and will not have information pertaining to any -current options until it has been released on CD. This is so we do not have any confusion on to which versions are being used or documented.

Information for previous versions of OpenBSD.

- OpenBSD 2.4

This FAQ will take you through most critical steps to setting up your own OpenBSD system. The addressed questions will range from newbie to advanced users. Hopefully you will find this FAQ useful. Downloadable version of the FAQ is available in a Text version.

- Text Version

Any questions can be directed to: faq@openbsd.org

The Current FAQ maintainers are:
Eric Jackson
Wim Vandeputte
Chris Cappuccio

$OpenBSD: index.html,v 1.80 1999/10/07 03:47:10 ericj Exp $

OpenBSD

1.0 - Introduction to OpenBSD

Table of Contents

1.1 - What is OpenBSD?

The OpenBSD project produces a freely available, multi-platform 4.4BSD-based UNIX-like operating system. Our efforts place emphasis on portability, standardization, correctness, and security. OpenBSD supports binary emulation of most binaries from SVR4 (Solaris), FreeBSD, Linux, BSDI, SunOS, and HPUX.

1.2 - On what systems does OpenBSD run?

OpenBSD 2.5 will install and run on the following platforms:

- i386 - CD bootable
- sparc - CD bootable
- alpha - CD bootable
- hp300
- amiga
- mac68k
- powerpc
- pmax
- mvme88k
    - The ports for pmax and mvme88k are not included on the 2.5 CD. They can be downloaded

from any of the OpenBSD ftp sites

bootable means that OpenBSD will boot directly from the CD. The CD set will boot on several hardware platforms. See section 3.0 of this FAQ for details of obtaining OpenBSD on CD.

Previous releases of OpenBSD also had ports for:

- arc
- mvme68k

These were removed from the 2.4 release, but were included in 2.3 if you are interested. If you have the proper equipment, bring them up to date! You can see updated information about specific platforms at http://www.openbsd.org/plat.html

Yes, OpenBSD will run on your multiprocessor machine, but it will only use one processor. There currently is no support for SMP. Check 8.12

1.3 - Is OpenBSD really free?

OpenBSD is all free. The binaries are free. The source is free. All parts of OpenBSD have reasonable copyright terms permitting free redistribution. This includes the ability to REUSE most parts of the OpenBSD source tree, either for personal or commercial purposes. OpenBSD includes NO further restrictions other than those implied by the original BSD license. Software which is written under stricter licenses cannot be included in the regular distribution of OpenBSD. This is intended to safeguard the free use of OpenBSD. For example, OpenBSD can be freely used for personal use, for academic use, by government institutions, by non-profit making organizations and by commercial organizations.

For further reading on other popular licenses read: http://www.openbsd.org/policy.html.

The maintainers of OpenBSD support the project largely from their own pockets. This includes the time spent programming for the project, equipment used to support the many ports, the network resources used to distribute OpenBSD to you, and the time spent answering questions and investigating users' bug reports. The OpenBSD developers are not independently wealthy and even small contributions of time, equipment, and resources make a big difference.

1.4 - Why might I want to use OpenBSD?

New users frequently want to know whether OpenBSD is superior to some other free UNIX-like operating system. That question is largely un-answerable and is the subject of countless (and useless) religious debates. Do not, under any circumstances, ask such a question on an OpenBSD mailing list.

Below are some reasons why we think OpenBSD is a useful operating system. Whether OpenBSD is right for you is a question that only you can answer.

- OpenBSD runs on many different hardware platforms.
- OpenBSD is thought of by many security professionals to be the most secure UNIX-like operating

system as the result of a 10-member 1.5-year long comprehensive source code security audit.

- OpenBSD is a full-featured UNIX-like operating system available in source form at no charge.
- OpenBSD integrates cutting-edge security technology suitable for building firewalls and private network services in a distributed environment.
- OpenBSD benefits from strong on-going development in many areas, offering opportunities to work with emerging technologies with an international community of programmers and end-users.
- OpenBSD offers opportunities for ordinary people to participate in the development and testing of the product.

1.5 - How can I help support OpenBSD?

We are greatly indebted to the people and organizations that have contributed to the OpenBSD project. They are acknowledged by name here:

http://www.openbsd.org/donations.html

OpenBSD has a constant need for several types of support from the user community. If you find OpenBSD useful, you are strongly encouraged to find a way to contribute. If none of the suggestions below are right for you, feel free to propose an alternative by sending e-mail to donations@openbsd.org.

- Buy an OpenBSD CD. It includes the current full release of OpenBSD, and is bootable on many platforms. It also generates revenue to support the OpenBSD project, and reduces the strain on network resources used to deliver the distribution via the Internet. This inexpensive two-CD set includes full source. Remember, your friends need their own copy!
- Donate money. The project has a constant need for cash to pay for equipment, network connectivity, and expenses relating to CD publishing. Manufacturing CDs requires an up-front out-of-pocket investment for the OpenBSD developers, without guaranteed return. Send e-mail to donations@openbsd.org to find out how to contribute. Even small donations make a profound difference.
- Donate equipment and parts. The project has a constant need for general and specific hardware. Items such as IDE and SCSI disks, and various types of RAM are always welcome. For other types of hardware such as computer systems and motherboards, you should inquire as to current need. Write to donations@openbsd.org to arrange for shipment.
- Donate your time and skills. Programmers who enjoy writing operating systems are naturally always welcome, but there are literally dozens of other ways that people can be useful. Follow mailing lists and help answer new-user questions.
- Help maintain documentation by submitting new FAQ material (to faq@openbsd.org). Form a local user group and get your friends hooked on OpenBSD. Make a case to your employer for using OpenBSD at work. If you're a student, talk to your professors about using OpenBSD as a learning tool for Computer Science or Engineering courses. It's also worth mentioning one of the most important ways you should not try to "help" The OpenBSD project: do not waste your time engaging in operating system flame wars on Usenet newsgroups. It does not help the project to find new users and can cause substantial harm to important relationships that developers have with

other developers.

1.6 - Who maintains OpenBSD?

OpenBSD is maintained by a development team spread across many different countries. The project is coordinated by Theo de Raadt, located in Canada.

[To Section 2.0 - Other OpenBSD information resources]


www@openbsd.org

$OpenBSD: faq1.html,v 1.16 1999/09/24 01:46:13 ericj Exp $

OpenBSD

2.0 - Other OpenBSD information resources

Table of Contents

2.1 - Web Pages of Interest

The official website for the OpenBSD project is located at: http://www.OpenBSD.org

A lot of valuable information can be found here regarding all aspects of the OpenBSD project.

Additional information for laptop users can be found at:
http://www.monkey.org/openbsd-mobile/

2.2 - Mailing Lists

The OpenBSD project maintains several popular mailing lists which users should subscribe to and follow. To subscribe to a mailing list, send an e-mail message to majordomo@openbsd.org. That address is an automated subscription service. In the body of your message, on a single line, you should include a subscribe command for the list you wish to join. For example:

```
subscribe announce
```

The list processor will reply to you, asking for confirmation of your intent to join the list. The confirmation you send back to the list processor will be included in its reply to you. It will look something like this:

```
auth 90855167 subscribe announce you@example.com
```

Once you have confirmed your intent to join, you will be immediately added to the list, and the list processor will notify you that you were successfully added.

To unsubscribe from a list, you will again send an e-mail message to majordomo@openbsd.org. It might look like this:

```
unsubscribe announce
```

If you have any difficulties with the mailing list system, please first read the instructions. They can be obtained by sending an e-mail message to majordomo@openbsd.org with a message body of "help". These are the currently-available OpenBSD mailing lists:

- **announce** - Important announcements. This is a low-volume list.
- **tech** - General technical discussions
- **misc** - User questions and answers
- **bugs** - Bugs received via sendbug(1) and discussions about them.
- **source-changes** - Automated mailing of CVS source tree changes.
- **ports** - Automated mailing of CVS source tree changes.

- **advocacy** - Discussion of advocating OpenBSD.

Archives of the OpenBSD mailing lists can be found by visiting the mailing lists web page: http://www.openbsd.org/mail.html
You can also get mail archives from http://www.monkey.org/cgi-bin/wilma . This website contains searchable archives of the OpenBSD mailing lists.

Another mailing list that may be of interest is openbsd-mobile@monkey.org. This mailing list is a discussion of the use of OpenBSD in mobile computing.

To subscribe to this list use:

'echo subscribe | mail "openbsd-mobile-request@monkey.org"'

The archives for that can be found at: http://www.monkey.org/openbsd-mobile/archive/

2.3 - Manual Pages

OpenBSD comes with extensive documentation in the form of manual pages, as well as longer documents relating to specific applications. To access the manual pages and other documentation, be sure that you installed the man, misc, and text distributions.

Here is a list of some of the most useful manual pages for new users:

- **afterboot(8)** - things to check after the first complete boot
- **boot(8)** - system boot strapping procedures
- **passwd.conf(5)** - format of the password configuration file
- **adduser_proc(8)** - procedure for adding new users
- **adduser(8)** - command for adding new users
- **vipw(8)** - edit the pass word file
- **man(1)** - display the on-line manual pages
- **sendbug(1)** - send a problem report (PR) about OpenBSD to a central support site.
- **disklabel(8)** - Read and write disk pack label.
- **ifconfig(8)** - configure network interface parameters.
- **route(8)** - manually manipulate the routing tables.
- **netstat(1)** - show network status.
- **reboot, halt(8)** - Stopping and restarting the system.
- **shutdown(8)** - close down the system at a given time.
- **boot_config(8)** - how to change kernel configuration at boot

Also, If you are one of the people who didn't install the man*.tar.gz package, you can find all the OpenBSD man pages on the web at http://www.openbsd.org/cgi-bin/man.cgi.

In general, if you know the name of a command or a manual page, you can read it by executing `man command'. For example: `man vi' to read about the vi editor. If you don't know the name of the command, or if `man command' doesn't find the manual page, you can search the manual page database by executing `apropos something' or `man -k something' where something is a likely word that might appear in the title of the manual page you're looking for. For example:

```
bsd# apropos "time zone"
tzfile (5) - time zone information
```

```
zdump (8) - time zone dumper
zic (8) - time zone compiler
```

The parenthetical numbers indicate the section of the manual in which that page can be found. In some cases, you may find manual pages with identical names living in separate sections of the manual. For example, assume that you want to know the format of the configuration files for the cron daemon. Once you know the section of the manual for the page you want, you would execute `man n command' where n is the manual section number.

```
bsd# man -k cron
cron (8) - daemon to execute scheduled commands (Vixie Cron)
crontab (1) - maintain crontab files for individual users (V3)
crontab (5) - tables for driving cron
bsd# man 5 crontab
```

In addition to the UNIX manual pages, there is a typesettable document set (included in the misc distribution). It lives in the /usr/share/doc directory. If you also installed the text distribution, then you can format each document set with a `make' in the appropriate subdirectory. The psd subdirectory is the Programmer's Supplementary Documents distribution. The smm subdirectory is the System Manager's Manual. The usd subdirectory is the UNIX User's Supplementary Documents distribution. You can perform your `make' in the three distribution subdirectories, or you can select a specific section of a distribution and do a `make' in its subdirectory. Some of the subdirectories are empty. By default, formatting the documents will result in Postscript output, suitable for printing. The Postscript output can be quite large -- you should assume 250-300% increase in volume. If you do not have access to a Postscript printer or display, you may also format the documents for reading on a terminal display. In each Makefile you'll need to add the flag -Tascii to each instance of the groff commands (or execute it by hand). Some of the documents use the ms formatting macros, and some use the me macros. The Makefile in each document subdirectory (eg, /usr/share/doc/usd/04.csh/Makefile) will tell you which one to use. For example:

```
bsd# cd /usr/share/doc/usd/04.csh
bsd# groff -Tascii -ms tabs csh.1 csh.2 csh.3 csh.4 csh.a csh.g > csh.txt
bsd# more csh.txt
```

The UNIX manual pages are generally more current and trustworthy than the typesettable documents. The typesettable documents sometimes explain complicated applications in more detail than the manual pages do.

For many, having a hardcopy of the man page can be useful. Here are the guidelines to making a printable copy of a man page.

How do I display a man page source file? (i.e., one whose filename ends in a number, like tcpdump.8).

This is found throughout the src tree. The man pages are found in the tree unformatted, and many times thru the use of CVS, they will be updated. To view these pages simply :

```
# nroff -mdoc <file> | more
```

How do I get a plain man page with no formatting or control characters?

This is helpful to get the man page straight, with no non-printable characters.
Example:

```
# man <command> | col -b
```

How can I get a PostScript copy of a man page that's print-ready?

Note that must be the man page source file (probably a file that ends in a number; i.e., tcpdump.8). The PostScript versions of the man pages look very nice. They can be printed or viewed on-screen with a program like gv (GhostView). GhostView can be found in our Ports Tree.

```
# groff -mdoc -Tps <man_src_file> > outfile.ps
```

2.4 - Reporting Bugs

Before submitting any bug report, please read http://www.openbsd.org/report.html

Proper bug reporting is one of the most important responsibilities of end users. Very detailed information is required to diagnose most serious bugs. Developers frequently get bugs reports via e-mail such as these:

```
From: joeuser@example.com
To: bugs@openbsd.org
Subject: HELP!!!

I have a PC and it won't boot!!!!! It's a 486!!!!!
```

Hopefully most people understand why such reports get summarily deleted. All bug reports should contain detailed information. If Joe User had really wanted his bug investigated, his bug report would have looked something like this:

```
From: smartuser@example.com
To: bugs@openbsd.org
Subject: 2.5 panics on an i386

After installing OpenBSD 2.5 from the CD which I purchased via your outstanding
on-line ordering system, I find that the system halts before the kernel even
gets loaded. After booting with a bootdisk and escaping to a shell. This is the
dmesg output:

OpenBSD 2.5-current (OSHIBANA) #2: Fri Jun  4 11:11:41 EDT 1999
    shinobi@oshibana:/sys/arch/i386/compile/OSHIBANA
cpu0: F00F bug workaround installed
cpu0: Intel Pentium (P54C) ("GenuineIntel" 586-class) 120 MHz
cpu0: FPU,V86,DE,PSE,TSC,MSR,MCE,CX8
BIOS mem  = 654336 conventional, 15728640 extended
real mem  = 16384000
avail mem = 13725696
using 225 buffers containing 921600 bytes of memory
mainbus0 (root)
bios0 at mainbus0: AT/286+(63) BIOS, date 08/20/96
bios0: apminfo 0xf02dc00c diskinfo 0xf02dc034 cksumlen 1 memmap 0xf02dc0b0
apm0 at bios0: Power Management spec V1.1
apm0: battery life expectancy 96%
apm0: AC on, battery charge high, charging, estimated 1:28 minutes
pci0 at mainbus0 bus 0: configuration mode 1 (no bios)
"Toshiba (2nd ID) Host-PCI" rev 0x11 at pci0 dev 0 function 0 not configured
"Chips and Technologies 65550" rev 0x04 at pci0 dev 4 function 0 not configured
```

```
isa0 at mainbus0
isadma0 at isa0
wdc0 at isa0 port 0x1f0-0x1f7 irq 14
wd0 at wdc0 drive 0: <TOSHIBA MK2720FC>
wd0: 1296MB, 2633 cyl, 16 head, 63 sec, 512 bytes/sec, 2654280 sec total
wd0: using 16-sector 16-bit pio transfers, lba addressing (128KB cache)
sb0 at isa0 port 0x220-0x237 irq 7 drq 1: dsp v3.02
midi0 at sb0: <SB MIDI UART>
audio0 at sb0
opl0 at sb0: model OPL3
midi at opl0 not configured
pccom0 at isa0 port 0x3f8-0x3ff irq 4: ns16550a, 16 byte fifo
pccom1 at isa0 port 0x2f8-0x2ff irq 3: ns16550a, 16 byte fifo
pccom2 at isa0 port 0x3e8-0x3ef irq 5: ns16550a, 16 byte fifo
vt0 at isa0 port 0x60-0x6f irq 1: generic VGA, 80 col, color, 8 scr, mf2-kbd
pms0 at vt0 irq 12
fdc0 at isa0 port 0x3f0-0x3f5 irq 6 drq 2
fd0 at fdc0 drive 0: 1.44MB 80 cyl, 2 head, 18 sec
pcic0 at isa0 port 0x3e0-0x3e1 iomem 0xd0000-0xd3fff
pcic0: controller 0 (Intel 82365SL Revision 1) has sockets A and B
pcmcia0 at pcic0 controller 0 socket 0
pcmcia1 at pcic0 controller 0 socket 1
biomask 40c0 netmask 42c0 ttymask 56c2
root on wd0a
pctr: 586-class performance counters and user-level cycle counter enabled
rootdev=0x0 rrootdev=0x300 rawdev=0x302

Thank you!
```

If Joe User had a working OpenBSD system from which he wanted to submit a bug report, he would have used the sendbug(1) utility to submit his bug report to the GNATS problem tracking system. Obviously you can't use sendbug(1) when your system won't boot, but you should use it whenever possible. You will still need to include detailed information about what happened, the exact configuration of your system, and how to reproduce the problem. The sendbug(1) command requires that your system be able to deliver electronic mail successfully on the Internet.

If you have submitted a bug report and you want to check its current status without annoying anyone, the best ways are:

- Visit the GNATS tracking system: http://cvs.openbsd.org/cgi-bin/wwwgnats.pl.
- Look in the bugs@openbsd.org list archives: http://www.openbsd.org/mail.html.

[Back to Main Index] [To Section 1.0 - Introduction to OpenBSD] [To Section 3.0 - Obtaining OpenBSD]


www@openbsd.org

$OpenBSD: faq2.html,v 1.20 1999/10/07 03:47:10 ericj Exp $

OpenBSD

3.0 - Obtaining OpenBSD

Table of Contents

3.1 - Buying an OpenBSD CD

Purchasing an OpenBSD CD is generally the best way to get started. Visit the ordering page to purchase your copy: http://www.openbsd.org/orders.html.

There are many good reasons to own an OpenBSD CD:

- CD sales support on-going development of OpenBSD.
- Development of a multi-platform operating system requires constant investment in equipment.
- Your support in the form of a CD purchase has a real impact on future development.
- The CD contains binaries (and source) for all supported platforms.
- The CD is bootable on several platforms, and can be used to bootstrap a machine without a pre-existing installed operating system.
- The CD is useful for bootstrapping even if you choose to install a snapshot.
- Installing from CD is faster! Installing from CD preserves network connectivity resources.
- OpenBSD CD's always come with very nice stickers. You system isn't fully complete without these. You can only get these stickers by buying a CD set or donating hardware.

If you're installing a release version of OpenBSD, you should use a CD. Snapshot releases can only be installed over the network.

3.1.1 - Buying OpenBSD T-Shirts

Yes, OpenBSD now has t-shirts for your wearing enjoyment. You can view these at http://www.OpenBSD.org/tshirts.html. Enjoy :)

3.2 - Downloading via FTP or AFS

There are numerous international mirror sites offering FTP access to OpenBSD releases and snapshots. AFS access is also available. You should always use the site closest to you. Before you begin fetching a release or snapshot, you may wish to use ping and traceroute to determine which mirror site is nearest to you and whether that nearest mirror is performing adequately. Of course, your OpenBSD release CD is always closer than any mirror. Access information is here:

http://www.openbsd.org/ftp.html

3.3 - Obtaining Current Source Code

Source to OpenBSD is freely redistributable and available at no charge. Generally the best way to get started with a current source tree is to install the source from the most recent CD and then configure AnonCVS to update it regularly. Information about AnonCVS, including how to set it up, is available here:

http://www.openbsd.org/anoncvs.html.

or check The OpenBSD FAQ

If you don't have sufficient network bandwidth to support AnonCVS, or if your Internet access is via UUCP, you can still keep your source current by using CTM instead of AnonCVS. If that's your situation, then starting with a recent release CD is even more important. Information about CTM, including how to set it up, is available here:

http://www.openbsd.org/ctm.html.

Yet another alternative is to get the source code from the web. You can do that thru cvsweb at:

Http://www.openbsd.org/cgi-bin/cvsweb/

[Back to Main Index] [To Section 2.0 - Other OpenBSD information resources] [To Section 4.0 - Installation Guide]


www@openbsd.org

$OpenBSD: faq3.html,v 1.10 1999/09/24 01:46:14 ericj Exp $

OpenBSD

4.0 - Installation Guide

Table of Contents

4.1 - Installing from the OpenBSD CD

Installing from a CD is definitely the BEST choice, but first you must have purchased a CD. You can obtain a CD from http://www.openbsd.org/orders.html. If you are using either the i386, sparc or alpha platforms, you can boot directly off the CD. Here we will go into more elaborate instructions for substantial architectures.

i386

To startup an install for the *i386* architecture, it's best to try booting off of the CD. Most newer BIOS's allow this, and you might have to play with it for a minute. If that isn't an option the bootdisk is available on the CD also. The bootdisk is placed in CD1:2.5/i386/floppy25.fs. Ways to write this to a floppy vary on what platform you are currently on.

Unix Flavor

- **dd if=floppy25.fs of=/dev/fd0c bs=126b**

Windows/MS-DOS

- **rawrite** - Which is available in the tools directory on the CD.

WindowsNT

- **fdimage** or **ntrw.exe** - Both are available in the tools directory on the CD.

NOTE - Make sure you use a floppy with NO BAD BLOCKS or you will have problems.

Once you disk is made, simply boot off of the disk. To guide you through the rest of the install process use log25.txt, which is an install walkthrough guide.

SPARC

With the SPARC architecture you have the ability to boot from the CD. To do this, use CD1 and type either,

- **boot cdrom 25/sparc/bsd.rd**

- **b sd(0,6,0)2.5/sparc/bsd.rd**

depending on your ROM version. Then simply follow the install walkthrough guide. Alternatively you can boot using a bootdisk. Write the disk using the ways detailed above for the i386 architecture, and boot it using either,

- **boot fd()**
- **boot floppy**

again depending on your ROM version.

ALPHA

Your alpha must use SRM firmware, as opposed to ARC. With the alpha architechture you can install by booting from the CD using CD2. You can do this by using a command such as

boot -fi 2.5/alpha/bsd.rd dkaX

Use **show device** to find your CDROM drive identifier. If you are unable to boot from the CD, you can still use the bootdisk proviced at *CD2:2.5/alpha/floppy25.fs*, and write it using the ways detailed above for the i386 architecture. And again follow the install walkthrough guide.

4.2 - Installation via FTP

The ftp installation is almost the same as the normal installation. The same bootdisk will be used.

floppy25.fs

Just like the CD install you will set up your disk, then when you specify install media you choose (f)tp. You must take care to give the proper settings when you are asked to configure your network. Have available your IP address, netmask, nameserver addresses, and default route. If you are using DHCP, don't worry, the boot disk knows how to talk to your DHCP server to get all of this (see next section below). Once your network is configured correctly, you must choose the (f)tp installation option. You will be asked a few simple questions about proxies and passive or active transfers, and the you will be presented with a list of ftp servers. Select the one closest to you, and you are on your way.

Consult the install text for help.

4.3 - Installation via DHCP

Installation via DHCP is the same as ftp, except your system will use DHCP to configure your network. When you arrive at the network config option, it will ask for your IP address. Type "dhcp" (Without the quotes of course) and your IP address, netmask, gateway, and name servers will be setup from the values given by the DHCP server. Then all you have to do is follow the remainder of the directions from ftp install above and you will be set.

Consult the install text for help.

4.4 - What files are needed for Installation?

There are many packages containing the OpenBSD binaries, but which ones do you need to get your system up and running? Here is an overview of each package.

- **base25.tar.gz** - Has the BASE OpenBSD system *NEEDED*
- **etc25.tar.gz** - Has all the files in /etc *NEEDED*
- **comp25.tar.gz** - Has the compiler and its tools, libs. *STRONGLY SUGGESTED*
- **man25.tar.gz** - Holds man pages *STRONGLY SUGGESTED*
- **misc25.tar.gz** - Holds misc info, setup docs
- **game25.tar.gz** - Has the Games for OpenBSD *FUN*
- **xbase25.tar.gz** - Has the base install for X11
- **xfont25.tar.gz** - Holds X11's font server and fonts
- **xlink25.tar.gz** - Has the X servers link kit
- **xserv25.tar.gz** - Has X11's X servers
- **xshare25.tar.gz** - Has manpages, locale settings, includes, etc for X
- **bsd** - This is the Kernel. *NEEDED*

4.5 - How much space do I need for an OpenBSD installation?

The following are suggested sub-tree sizes for a full system install. The numbers include enough extra space to permit you to run a typical home system that is connected to the internet:

SYSTEM	/	/usr	/var	/usr/X11R6
alpha	56M	540M	27M	161M
amiga	45M	399M	24M	36M
arc	47M	268M	20M	60M
hp300	31M	234M	24M	47M
i386	35M	229M	24M	72M
mac68k	29M	232M	24M	36M
mvme68k	48M	448M	24M	- (no Xserver)
pmax	50M	355M	24M	60M
sparc	40M	259M	24M	49M
powerpc	39M	423M	26M	124M

When you are in the disklabel editor, you may choose to make your entire system have just an 'a' and 'b' partition. The 'a' partition you set up in disklabel will become your root partition, which should be the sum of all the 3 main values above (/, /usr, and /var) plus some space for /tmp. The 'b' partition you set up automatically becomes your system swap partition -- we recommend a minimum of 32MB but if you have disk to spare make it at least 64MB.

However, we recommend you use many separate partitions so that users cannot fill up your important partitions as easily, thus causing nasty denial of service problems. If you want to be extra cautious, you should make the following separate partitions: / swap /usr /var /tmp /var/tmp /home.

4.6 - Multibooting OpenBSD

OpenBSD & NT

To multiboot OpenBSD and NT, you can use NTloader, the bootloader that NT uses. To multi-boot with NT, you need a copy of your OpenBSD pbr. After running installboot, you can copy it something like this:

```
# dd if=/dev/rsd0a of=openbsd.pbr bs=512 count=1
```

Now boot NT and put openbsd.pbr in c:. Add a line like this to the end of c:\boot.ini:

```
c:\openbsd.pbr="OpenBSD"
```

When you reboot, you should be able to select OpenBSD from the NT loader menu. There is much more information available about NTloader at the NTLDR Hacking Guide.

OpenBSD & Windows or DOS

To boot OpenBSD along with Windows 3.1, Windows95, or DOS you must use a boot loader on the system that can handle OpenBSD, Windows, and DOS! Some bootloaders of choice are osbs20b8.zip or The Ranish Partition Manager. Both of these are able to boot OpenBSD partitions.

OpenBSD & Linux

Please refer to INSTALL.linux, which gives indepth instructions on getting OpenBSD working with Linux.

4.7 - Sending your dmesg to dmesg@openbsd.org after the install

Just to remind people, it's important for the OpenBSD developers to keep track of what hardware works, and what hardware doesn't work perfectly..

A quote from /usr/src/etc/root/root.mail

```
If you wish to ensure that OpenBSD runs better on your machines, please do us
a favor (after you have your mail system setup!) and type
        dmesg | mail dmesg@openbsd.org
so that we can see what kinds of configurations people are running.  We will
use this information to improve device driver support in future releases.
(We would be much happier if this information was for the supplied GENERIC
kernel; not for a custom compiled kernel).  The device driver information
we get from this helps us fix existing drivers.
```

Also check with section 14.7

Make sure you send email from an account that is able to also receive email so developers can contact you back if they have something they want you to test or change in order to get your setup working. It's not important at all to send the email from the same machine that is running OpenBSD, so if that machine is unable to receive email, just

```
dmesg | mail your-account@yourmail.dom
```

and then forward that message to

```
dmesg@openbsd.org
```

where `your-account@yourmail.dom` is your regular email account. (or transfer the dmesg output using ftp/scp/floppydisk/carrier-pigeon/...)

NOTE - Please send only GENERIC kernel dmesg's. Custom kernels that have device drivers removed are not helpful.

[Back to Main Index] [To Section 3.0 - Obtaining OpenBSD] [To Section 5.0 - Kernel configuration and Disk Setup]


www@openbsd.org

$OpenBSD: faq4.html,v 1.59 1999/10/01 19:38:09 ericj Exp $

OpenBSD

5.0 - Kernel configuration and Disk Setup

Table of Contents

5.1 - Why would I want to create my own custom kernel?

It is a joyous occasion when you as a user get to configure and build your own custom kernel. Why would you want to do this though?

- You can define which drivers your computer has support for.
- You can get rid of unneeded drivers in the GENERIC kernel
- Speed up boot times
- Remove default options or add options which may not have been enabled by default

Under most circumstances you will NOT need to compile your own kernel. The GENERIC kernel will usually be all that you need.

5.2 - Kernel Configuration Options

Kernel Configuration Options are options that you add to your kernel configuration that place certain features into your kernel. This allows you to have exactly the support you want, without having support for unneeded devices. There are multitudes of options that allow you to customize your kernel. Here we will go over only some of them, those that are most commonly used. Check the options(4) man page for a more complete list of options. You can also check the example configuration files that are available for your architecture.

Not all kernel options have been tested for compatibility with all other options. Don't put an option in your kernel unless you actually have a reason to do so! The one kernel configuration which gets the most testing is the GENERIC kernel. This is usually a combination of the options in /usr/src/sys/arch/<your arch>/conf/GENERIC and /usr/src/sys/conf/GENERIC.

- Alpha Kernel Conf Files
- i386 Kernel Configuration files
- mac68k Kernel Configuration files
- pmax Kernel Configuration Files
- sparc Kernel Configuration Files
- Other Arch's

Look closely at these files and you will notice a line like:
include "../../../conf/GENERIC"
This means that it *IS* referencing yet another configuration file. This file stores non arch-dependent options. So when creating your Kernel Config be sure to look through /sys/conf/GENERIC and see what you want. There ARE options in there that are NEEDED.

All of the options listed below should be placed in your kernel configuration file in the format of:
option OPTION
for example. To place option debug in the kernel, add a line like this:
option DEBUG
Options in the OpenBSD kernel are translated into compiler preprocessor options, therefore an option like DEBUG would have the source compiled with option -DDEBUG. Which is equivalent to doing a #define DEBUG throughout the kernel.

OpenBSD has a great many compatibility options which allow you to use binaries from other OS's. Not all are availibly on every architechture, so be sure to read the man pages for each option to see if your arch is supported.

- COMPAT_SVR4(8) - Compatibility with SVR4 binaries.
- COMPAT_BSDOS(8) - Compatibility with BSD/OS binaries.
- COMPAT_LINUX(8) - Compatibility with Linux binaries.
- COMPAT_SUNOS(8) - Compatibility with SunOS binaries.
- COMPAT_ULTRIX(8) - Compatibility with Ultrix binaries.
- COMPAT_FREEBSD(8) - Compatibility with FreeBSD binaries.
- COMPAT_HPUX - Compatibility with HP-UX binaries. Only available on some m68k arch's.
- COMPAT_IBCS2(8) - Compatibility to run ibcs2 binaries.
- COMPAT_OSF1 - Run Digital Unix binaries. Available only on Alpha platform.
- COMPAT_43 - Compatibility with 4.3BSD. Use of this is certainly discouraged, But it is needed for our Navigator port
- COMPAT_11 - Compatibility with NetBSD 1.1
- COMPAT_NOMID - Compatibility with a.out executables that lack a machine id.

It is always helpful to be able to debug problems with the kernel. But many choose not to put these options in their kernel because these options add considerable size to the kernel. They are however extremely helpful in a case where a bug might be present. This well help the developers discover the

source of your problems much quicker. Here is a list of popular debugging options that can be added to your kernel.

- DDB - This compiles in the in-kernel debugger. This isn't available on all platforms. So be sure to read before adding it.
- KGDB - Compiles a remote kernel debugger using gdb's `remote target` feature.
- makeoptions DEBUG="-g" - Makes bsd.gdb along with bsd. This is useful for debugging crash dumps with gdb. NOTE: This option also turns on **option DEBUG**
- DEBUG - Used to put various debugging options in the kernel, where the source has defined them.
- KTRACE - Adds hooks for the system call tracing facility. Which allows users to use ktrace(1).
- DIAGNOSTIC - Adds code to the kernel that does internal consistency checks.
- GPROF - adds code to the kernel for kernel profiling with kgmon(8).
- makeoptions PROF="-pg" - The -pg flag causes the kernel to be compiled with support for profiling. Option GPROF is required to use this option.

Filesystem Options.

- FFS - Berkeley Fast Filesystem **NOTE:** This option is required
- EXT2FS - Second Extended File System, This is needed for those of you who want to read Linux partitions.
- MFS - Memory File System that stores files in swapable memory. This is not quite stable as of now.
- NFS - Network File System, This is needed if you will be using NFS.
- CD9660 - This is iso9660 + rockridge filesystem. This is required to read from cd's
- MSDOSFS - Needed to read MS-DOS FAT filesystems. Also has support for Windows 95 long name + mixed case extensions.
- FDESC - Includes code for a file system which can be mounted on /dev/fd
- KERNFS - Includes code which permits the mounting of a special file system (normally mounted on /kern) in which files representing various kernel variables and parameters may be found.
- NULLFS - Code to have a loopback filesystem. mount_null(8) has more information
- PROCFS - Includes code for a special file system (conventionally mounted on /proc)
- PORTAL - Includes the (experimental) portal filesystem. This permits interesting tricks like opening TCP sockets by opening files in the file system.
- UMAPFS - Includes a loopback file system in which user and group ids may be remapped -- this can be useful when mounting alien file systems with different uids and gids than the local system (eg, remote NFS).
- UNION - Includes code for the union file system, which permits directories to be mounted on top of each other in such a way that both file systems remain visible. This code isn't quite stable yet.
- XFS - Add hooks for using a filesystem that is compatible with the AFS filesystem. Currently used by the Arla/AFS code.
- LFS - Log Structured Filesystem.

- FFS_SOFTUPDATES - Allows for the use of softupdates. To read up on softupdates more read the Softupdates FAQ section.
- NFSSERVER - Allow for the server-side NFS code to be included in the kernel.
- NFSCLIENT - Allow for the client-side NFS code to be included in the kernel.
- FIFO - Support for FIFO's.. RECOMMENDED.
- NVNODE=integer - Where integer is the size of the cache used by the name-to-inode translation routines, (a.k.a. the namei() cache, though called by many other names in the kernel source).
- EXT2FS_SYSTEM_FLAG - This option changes the behavior of the APPEND and IMMUTABLE flags for a file on an EXT2FS filesystem. Read options(4) for more details.
- QUOTA - Support for Filesystem Quota's. To read up on using quotas read FAQ 10

Misc. Options

- PCIVERBOSE - Make the boot process more verbose for PCI peripherals.
- EISAVERBOSE - Make the boot process more verbose for EISA peripherals.
- PCMCIAVERBOSE - Make the boot process more verbose for PCMCIA peripherals.
- APERTURE - Provide in-kernel support for VGA framebuffer mapping by user processes. Needed to run X.
- XSERVER - Support for the X server console driver. Needed for X.
- LKM - Support for Loadable Kernel Modules. Not available on all arch's. Read lkm(4) for more information.
- INSECURE - Hardwires the kernel security level to -1. Read init(8) for more information on kernel security levels.
- MACHINE_NONCONTIG - Changes part of the VM/pmap interface, to allow for non-contiguous memory.
- RAM_DISK_HOOKS - Allows for machine dependent functions to be called when the ramdisk driver is configured.
- RAM_DISK_IS_ROOT - Forces the ramdisk to be root.
- CCDNBUF=integer - Set number of component buffers used by CCD(4). Default is 8. For more on CCD read the CCD(4) man page, or the Performance Tuning FAQ section.
- KMEMSTATS - This makes malloc(9), the kernel memory allocator keep statistics on its use. If option DEBUG is used, this option is automatically turned off by config.
- BOOT_CONFIG - Adds support of the -c boot option.

Networking Options

Also check the Networking FAQ or the Networking Performance Tuning FAQ.

- GATEWAY - Enables IPFORWARDING and (on most ports) increases the size of NMBCLUSTERS.
- NMBCLUSTERS=integer - Controls the size mbuf cluster map.
- IPFORWARDING - Enables IP routing behavior. With this option enabled, the machine will forward IP datagrams between its interfaces that are destined for other machines.

- MROUTING - Includes support for IP multicast routers.
- INET - Includes support for the TCP/IP protocol stack. This option is REQUIRED
- MCLSHIFT=value - This option is the base-2 logarithm of the size of mbuf clusters. Read options(4) for more information on this option.
- NS - Include support for the Xerox XNS protocol stack. See ns(4)
- ISO,TPIP - Include support for the ubiquitous OSI protocol stack. See iso(4) for more information.
- EON - Include support for OSI tunneling over IP.
- CCITT,LLC,HDLC - Include support for the X.25 protocol stack.
- IPX, IPXIP - Include support for Internetwork Packet Exchange protocol commonly in use by Novell NetWare.
- NETATALK - Include kernel support for the AppleTalk family of protocols.
- TCP_COMPAT_42 - Use of this option is extremely discouraged, so it should not be enabled. TCP bug compatibility with 4.2BSD. In 4.2BSD, TCP sequence numbers were 32-bit signed values. Modern implementations of TCP use unsigned values.
- TCP_NEWRENO - Turns on NewReno fast recovery phase, which allows one lost segment to be recovered per round trip time.
- TCP_SACK - Turns on selective acknowledgements.
- TCP_FACK - Turns on forward acknowledgements allowing a more precise estimate of outstanding data during the fast recovery phase by using SACK information. This option can be used together with TCP_SACK.
- IPFILTER - This option enables the IP filtering on the packet level using the ipfilter package.
- IPFILTER_LOG - This option, in conjunction with IPFILTER, enables logging of IP packets using ip-filter.
- IPFILTER_DEFAULT_BLOCK - This option sets the default policy of ip-filter. If it is set, ip-filter will block packets by default.
- PPP_FILTER - This option turns on pcap(3) based filtering for ppp connections.
- PPP_BSDCOMP - PPP BSD compression.
- PPP_DEFLATE - Used in conjunction with PPP_BSDCOMP.
- IPSEC - This option enables IP security protocol support. See ipsec(4) for more details. This now implies option KEY, which gives support for PFKEYv2.
- ENCDEBUG - This option enables debugging information to be conditionally logged in case IPSEC encounters errors.
- TCP_SIGNATURE - TCP MD5 Signatures, for BGP routing sessions

PCVT Options (*only valid on i386 architecture*)

- PCVT_NSCREENS=integer - Number of pcvt screens. Defaults to 8.
- PCVT_VT220KEYB - If activated, a keyboard layout resembling a DEC VT200 (TM) is generated.
- PCVT_SCREENSAVER - Enables the builtin screensaver feature. Defaults to on.

- PCVT_PRETTYSCRNS - If enabled, a blinking-star screensaver is used. Default is off.
- PCVT_CTRL_ALT_DEL - If enabled, the key combination <Ctrl> <Alt> <Del> invokes a CPU reset. Default is off.
- PCVT_USEKBDSEC - Do NOT override a security lock for the keyboard. Default is on.
- PCVT_24LINESDEF - If enabled, the 25-line modi (VT emulation with 25 lines, and HP emulation with 28 lines) default to 24 lines only to provide a better compatibility to the original DEV VT220 (TM).
- PCVT_EMU_MOUSE - Emulate a three-button mouse via the keypad. Useful for notebooks when running XFree86. Defaults to off
- PCVT_META_ESC - If enabled, a sequence composed of <esc>, followed by the normal key code is emitted if a key is pressed with the <Alt> key modifier. If disabled, then normal key code with the value 0x80 added is sent. Defaults to off.

SCSI Subsystem Options

- SCSITERSE - Terser SCSI error messages. This omits the table for decoding ASC/ASCQ info, saving about 8 bytes or so.
- SCSIDEBUG - Prints extra debugging info for the SCSI subsystem to the console.

5.3 - Building your own kernel

Full instructions for creating your own custom kernel are in the afterboot(8) man page.

To compile your kernel from the cdrom you need to first have the source code available. You just need the kernel source to be able to compile the kernel, and this source code is available on the cd. Here is how to copy the sources from the cd. This example assumes that CD1 is mounted on /mnt.

```
#mkdir -p /usr/src/sys
#cd /mnt/sys
#tar cf - . | ( cd /usr/src/sys; tar xvf - )
```

Now to create your custom kernel it is easiest to start with the GENERIC kernel. This is located at */usr/src/sys/arch/${arch}/conf/GENERIC*, where ${arch} is your architecture. There are other sample configurations available in that directory as well. Here are two examples for compiling your kernel. The first example is compiling your kernel on a read-only source tree. The second on a writeable source tree.

```
#cd /somewhere
#cp /usr/src/sys/arch/$ARCH/conf/SOMEFILE .
#vi SOMEFILE (to make the changes you want)
#config -s /usr/src/sys -b . SOMEFILE
#make
```

To compile a kernel inside a writeable source tree do the following:

```
#cd /sys/arch/$ARCH/conf
```

```
#vi SOMEFILE (to make any changes you want)
#config SOMEFILE (read more about it here : config(8))
#cd ../compile/SOMEFILE
#make
```

Where $ARCH is the architecture you are using (e.g. i386). You can also do a **make depend** to make the dependencies for the next time you compile your kernel.

To move your kernel into place.

```
#cp /bsd /bsd.old
#cp /sys/arch/$ARCH/compile/SOMEFILE/bsd /bsd
```

To revert to your old kernel at boot you need to just

```
boot> bsd.old
```

and your old kernel will be loaded instead of /bsd.

Sometimes when you build a new kernel you will be required to install new bootblocks. To do so, read faq15.8 on OpenBSD's Bootloader. Which will give you an overview on using OpenBSD's Bootloader

5.4 - Boot Time Configuration

Sometimes when booting your system you might notice that the kernel finds your device but maybe at the wrong IRQ. And maybe you need to use this device right away. Well, without rebuilding the kernel you can use OpenBSD's boot time kernel configuration. This will only correct your problem for one time. If you reboot, you will have to repeat this procedure. So, this is only meant as a temporary fix, and you should correct the problem by fixing and recompiling your kernel. Your kernel does however need **option BOOT_CONFIG** in the kernel, which GENERIC does have.

Most of this document can be found in the man page boot_config(8)

To boot into the User Kernel Config, or UKC, at boot time us the -c option.

```
Boot> boot wd0a:/bsd -c
```

Or whichever kernel it is you want to boot. Doing this will bring up a UKC prompt. From here you can issue commands directly to the kernel specifying devices you want to change or disable or even enable.

Here is a list of common commands in the UKC.

`add **device** - Add a device through copying another`

`change **devno | device** - Modify one or more devices`

`disable **devno | device** - Disable one or more devices`

`enable **devno | device** - Enable one or more devices`

`find **devno | device** - Find one or more devices`

`help - Short summary of these commands`

`list - List ALL known devices`

`exit/quit - Continue Booting`

`show **[attr [val]]** - Show devices with an attribute and optional with a specified value`

Once you get your device configured, use quit or exit and continue booting. After doing so you should correct your Kernel configuration and Compile a new kernel. Refer to Building your own kernel for help.

[Back to Main Index] [To Section 4.0 - Installation Guide] [To Section 6.0 - Networking]


www@openbsd.org

$OpenBSD: faq5.html,v 1.32 1999/09/24 01:46:14 ericj Exp $

OpenBSD

6.0 - Networking

Table of Contents

6.0.1 - Before we go any further

For the bulk of this document, it helps if you have read and at least partially understand the Kernel Configuration and Setup section of the FAQ, and the ifconfig(8) and netstat(1) man pages.

If you are a network administrator, and you are setting up routing protocols, if you are using your OpenBSD box as a router, if you need to go in depth into IP networking, you really need to read Understanding IP addressing. This is an excellent document. Understanding IP addressing contains fundamental knowledge to build upon when working with IP networks!

If you are working with applications such as web servers, ftp servers, and mail servers, you may benefit greatly by reading the RFCs. Of course, you can't all of them. That's not easy to do (although the IETF did it to evaluate the RFCs for year 2000 issues!) Rather, pick some topics that you are interested in, or that you use in your work. Look them up, find out how they are intended to work. The RFCs define many (thousands) of standards for protocols on the internet and how they are supposed to work.

6.1 - Initial Network Setup

Interfaces

Here we assume you have all your network interfaces working, and a basic TCP/IP knowledge.

All your interfaces should be listed with:

```
# ifconfig -a

lo0: flags=8009<UP,LOOPBACK,MULTICAST>
        inet 127.0.0.1 netmask 0xff000000
lo1: flags=8008<LOOPBACK,MULTICAST>
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>
        media: Ethernet 100baseTX half-duplex
        inet 10.1.1.1 netmask 0xffffff00 broadcast 10.1.1.255
sl0: flags=c010<POINTOPOINT,LINK2,MULTICAST>
        inet 10.1.1.1 netmask 0xffffff00 broadcast 10.1.1.255
```

```
sl0: flags=c010<POINTOPOINT,LINK2,MULTICAST>
sl1: flags=c010<POINTOPOINT,LINK2,MULTICAST>
ppp0: flags=8010<POINTOPOINT,MULTICAST>
ppp1: flags=8010<POINTOPOINT,MULTICAST>
tun0: flags=10<POINTOPOINT>
tun1: flags=10<POINTOPOINT>
enc0: flags=8<LOOPBACK>
```

First there is lo0 the loopback interface. It normally should be assigned address of 127.0.0.1 no matter what the rest of your network looks like. The next important one is xl0 in this example (could be ne*X*, ep*X*, or one of many other names, depending on the chipset that your network card uses), which is a board itself.

In this example, xl0 it has UP and RUNNING flags on, along with le0. That means that they are working (for the most part, anyways.) Your interface could be down if you never configured it, and then it would look like the other interfaces. Or, your interface could be up but incorrectly configured (wrong IP address or wrong netmask) and thus won't work properly. The other interfaces in this example are not part of this section, as sl and ppp are for serial line comunication, tun is a pseudo-device for tunneling and enc a pseudo-device for encryption. They are covered later in this document.

If xl0 was uninitialized, you can assign it an address by creating an ascii file, /etc/hostname.xl0, containing a string like this:

```
inet 10.1.1.1 255.255.255.0 NONE
```

In the above example we show *inet 10.1.1.1*, where the second argument is our IP address. *255.255.255.0* is our netmask and *NONE* shows no media tags. (full-duplex, etc).

In /etc/hosts we then put a hostname for ip address *10.1.1.1*:

```
10.1.1.1        mona    mona.openbsd.org.ar
```

Now check that you have a file /etc/myname and a file /etc/mygate. If you don't have them, you need to create them. If your gateway is a machine named "wintermute" which is also in the /etc/hosts, use these commands:

```
# echo "mona" > /etc/myname
# echo "wintermute" > /etc/mygate
```

Ok, you are done for a standalone system, but to activate your configuration now (without rebooting) do:

```
# sh /etc/netstart
```

This will give you a few errors, but don't worry as all them are concerning 127.0.0.x (loopback)

Note: if you want to use the system as a gateway, later there's a "Firewall Setup" section, but you need to read all this first.

Now check your routes are ok (here we use -n to make it simpler to look at):

```
# netstat -rn
Routing tables

Internet:
Destination        Gateway            Flags     Refs     Use    Mtu
Interface
default            10.1.1.254         UGS          0        0      -  xl0
10.1.1/24          link#1             UC           0        0      -  xl0
10.1.1.1           127.0.0.1          UGHS         0        0      -  lo0
10.1.1.254         link#1             UHL          1        0      -  xl0
127/8              127.0.0.1          UGRS         0        0      -  lo0
127.0.0.1          127.0.0.1          UH           3       24      -  lo0
```

```
224/8              link#1             UCS         0        0      -  xl0

Encap:
Source address/netmask          Port  Destination address/netmask     Port Proto
SA(Address/SPI/Proto)
```

(The `default` gateway here is 10.1.1.254)

IP Aliasing

If you want to map more than one IP address on a single network interface, you need to assign an "alias" to it. The way to do it on OpenBSD is simply assigning the interface another IP with the "alias" option like this:

```
# ifconfig xl0 alias 10.1.1.2 netmask 255.255.255.255
```

Here ne2 is assigned another IP number (remember it already has 10.1.1.1). Note the netmask used! Don't use 255.255.255.0 in this kind of case (both numbers are on the same subnet), only ONE address should handle the subnet. If you do such kind of misconfiguration, the following error appears:

```
ifconfig:SIOCAIFADDR: File exists
```

Now let's change the configuration file for this, /etc/ifaliases. Add to it this line:

```
xl0   10.1.1.2 255.255.255.255
```

This means, simply, interface `xl0` should have an alias `10.1.1.2` with the netmask `255.255.255.255`.

DNS Client Setup

Let's assume your DNS servers are 125.2.3.4 and 125.2.3.5, and your machine name belongs to the domain yourdomain.com. You should have the following lines to your /etc/resolv.conf file:

```
domain yourdomain.com
nameserver 125.2.3.4
nameserver 125.2.3.5
lookup file bind
```

Gateway Setup

This is the basic information you need to set up your OpenBSD box as a gateway (also called a router.) If you are using OpenBSD as a router on the Internet, we suggest that you also read the IP Filter setup instructions below to block potentially malicious traffic. Also, due to the low availability of IPv4 addresses from network service providers and regional registries, you may want to look at Network Address Translation (next section of this document) for information on conserving your IP address space.

If you want your OpenBSD machine to act as a dedicated router, and you are building a custom kernel configuration for it, you can either compile your kernel with options IPFORWARDING or GATEWAY. (See section 11 for further information on tuning NMBCLUSTERS if you are doing this.) For the rest of us, OpenBSD now has a sysctl mechanism to turn on and off IP Forwarding at boot. If you want it to "forward" any packets between your interfaces, modify the /etc/sysctl.conf line that toggles the forwarding variable on bootup:

```
net.inet.ip.forwarding=1
```

To make the changes without booting:

```
# sysctl -w net.inet.ip.forwarding=1
```

```
net.inet.ip.forwarding: 0 -> 1
```

Now modify the routes on the other hosts on both sides. There are many possible uses of OpenBSD as a router, using software such as routed(8), gated, mrtd, and zebra. OpenBSD has support in the ports collection for both gated and mrtd. OpenBSD supports several T1, HSSI, ATM, FDDI, Ethernet, and serial (PPP/SLIP) interfaces.

6.2 - IP Filter

The IP Filter package was created to handle two tasks, dealing with packet level forwarding permissions ipf(8) and mapping hosts/subnets to a range of external addresses ipnat(8). The configuration files for these two services are /etc/ipf.rules and /etc/ipnat.rules.

You need to edit /etc/rc.conf to activate them at boot time. You also need to have net.inet.ip.forwarding=1 in your /etc/sysctl.conf (or your kernel needs to have IPFORWARDING or GATEWAY options turned on.) You also need a kernel compiled with option IPFILTER and IPFILTER_LOG (the GENERIC kernels do have these options).

If you have IP Filter compiled into your kernel, but you don't have it turned on in your /etc/rc.conf file, you can still activate it easily.

```
# ipf -Fa -f /etc/ipf.rules -E
# ipnat -CF -f /etc/ipnat.rules
```

The `-E` flag on ipf 'enables' IP Filter. `-Fa` clears out any rules that you may have in there. `-f /etc/ipf.rules` loads the rules from /etc/ipf.rules.

If you make changes to /etc/ipf.rules after ipf is loaded, you can reload your rules pretty easily!

```
# ipf -Fa -f /etc/ipf.rules
```

Same for ipnat...

```
# ipnat -CF -f /etc/ipnat.rules
```

This document will cover some basic ipf and ipnat configurations below. There are a lot of nice examples in /usr/share/ipf/ for ipnat and ipf. We recommend you choose the one closest to what you want, and modify it to fit your needs. You can find other IP Filter information at the IP Filter mailing list archive, the IP Filter web site, and finally the IP Filter HOWTO.

IPF

Modify rc.conf so it has IPFILTER=YES. The file /etc/ipf.rules has a simple yet powerful syntax. Here we deal with the most common ways of usage, for a more strict definition see ipf(5). Here is assumed xl0 as the external interface to internet, on this one uses to have more rules than internal interfaces.

Configurations usually start letting everything come and go, and then apply the necessary rules to block offending packets. So this is first:

```
pass out from any to any
pass in from any to any
```

Now let's block any incoming connection to port 82 tcp (eg. there's an internal network report agent using http running on several hosts):

```
block in on xl0 proto tcp from any to any port = 82
```

This rule means:

"Block all incoming packets on xl0 interface whose protocol is TCP no matter destination/origin using port 82"

If you want to log all rejected packets add "log" after "block in", or "log quick" if you don't want it to send a message to every console root is logged in.

Also, a typical rule is to block rpc portmap:

```
block in log on xl0 proto udp from any to any port = sunrpc
```

6.3 - IPNAT

Initial work done by Wayne Fergerstrom <wayne@methadonia.net>

6.3.1 Introduction

Purpose

The purpose of this document is to explain and give and overview of installing and configuring Network Address Translation ("NAT") on an OpenBSD machine. This document is meant for users with a beginning to intermediate level of knowledge in UNIX, TCP/IP, and networking technologies. The user is assumed to have already set up and configured an OpenBSD machine with two network cards (one connected to the Internet and the other to the local network).

Based on RFC 1631, ipnat provides an easy way to map internal networks to a single routeable ("real") internet address. This is very useful if you don't have officially assigned addresses for every host on your internal network. When you set up private/internal networks, you can take advantage of reserved address blocks (assigned in RFC 1918), such as:

10.0.0.0/8 (10.0.0.0 - 10.255.255.255)
172.16.0.0/12 (172.16.0.0 - 172.31.255.255)
192.168.0.0/16 (192.168.0.0 - 192.168.255.255)

Terminology

The conventions used in this document are fairly straightforward. For documentation purposes I will review some of the terms and format for which this document adheres to.

"NAT"

This describes the function of "Network Address Translation." The process of NAT is described later in this document.

"ipnat"

This is short for "IP Network Address Translation." In-and-of itself, it can be used interchangeably with NAT. However, in this document the term "ipnat" will be used solely for command-line only use.

"IPF"

This is short for "IP Filter." IP Filter is a portable packet filtering software that is included as part of OpenBSD. IP Filter must be enabled before you can turn on ipnat. This is easy, just edit /etc/rc.conf and change ipf=NO to ipf=YES. That only changes it for the boot up sequence, you also need to do 'ipf -E' to turn on ipf while you are booted. Of course, this is described further, below.

Configuration

This is how the computers are setup concerning this document. Your setup will vary from this, but the purpose of the document is to give you an overview so you can conform this information to your setup.

Computer Operating System: OpenBSD v2.5 i386

NICs:

NetGear 10/100MB **pn0**
Connected to the EXTERNAL LAN (or WAN)
IP Address: 24.5.0.5
Netmask: 255.255.255.0

NetGear 10/100MB **pn1**
Connected to the INTERNAL LAN
IP Address: 192.168.1.1
Netmask: 255.255.255.0

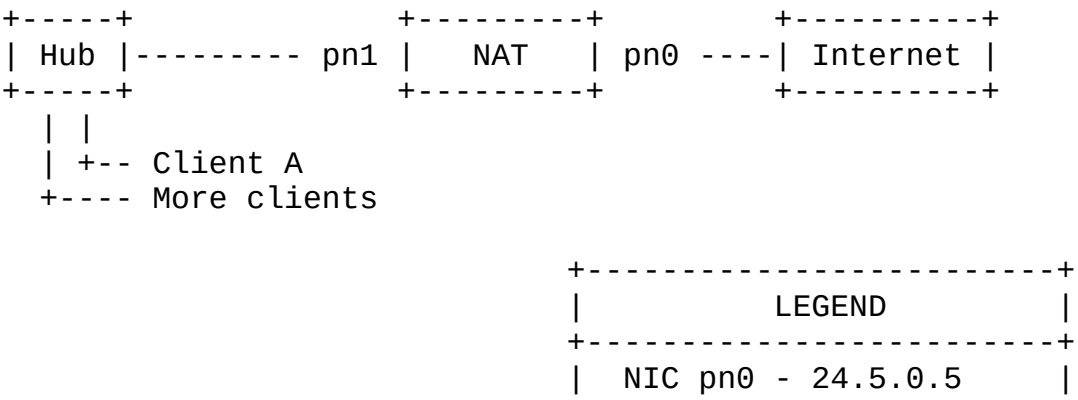
External, Internet-routeable IP (provided by ISP, in this example, a cable modem prodiver)

IP Address: 24.5.0.5
Netmask: 255.255.255.0
Gateway: 24.5.0.1

Local Area Network

In this example, machines on the LAN use the IP addressing scheme 192.168.1.xxx (where xxx is a unique number). There are a variety of different operating systems on the internal LAN including Windows 98, Windows NT, OpenBSD and Linux. Each machine is connected to a hub that is designated for internal use. For this document and its examples the client on the LAN will assume IP address 192.168.1.40

Diagram of Configuration



```
|  NIC pn1 - 192.168.1.1  |
| Client A - 192.168.1.40 |
+-------------------------+
```

6.3.2 Network Address Translation

Introduction to NAT

As more and more businesses and users get on the Internet, each one must have an IP address. Public IP addresses are becoming harder and harder to get. The solution for a lot of people has been Network Address Translation (or "NAT"). NAT is a very simple, yet powerful way to get your LAN connected to the Internet without having to purchase or lease IP addresses for each machine. NAT is also known as "IP Masquerading" if you're a Linux user.

When NAT is up and running correctly, it allows users on the internal LAN to access the Internet through a different IP address (the one you set up with your provider). Each machine on the LAN uses the one IP address (transparently) of the one machine that is set up to use the ISP assigned IP address.

The way NAT works is amazingly simple. When a client on the LAN wants to connect to a machine on the Internet, it sends out a TCP packet with a request to connect. Inside the TCP packet header is the client's IP address (i.e. 192.168.1.40) and the requested host's IP address (i.e. 123.45.67.89). The machine running NAT intercepts this TCP packet and changes the client's IP address from 192.168.1.40 to the IP address of the Internet-connected machine (i.e. 24.5.0.5). This effectively tricks the host machine into thinking the actual connection is from the NAT machine, not the actual client's machine. The host then sends back responses to the NAT machine like it was the one connecting. When the NAT machine receives the responses it quickly translates the destination IP address back from itself to the client's machine and sends the packet to the client. The client didn't have any idea of what happened and spoofed Internet connectivity is totally transparent.

The example below shows NAT a little more clearly:

```
Client ----------------- pn1 [ NAT ] pn0 ---------- Internet Host
192.168.1.40 --- 192.168.1.1 [ NAT ] 24.12.34.56 --- 123.45.67.89

OUTGOING TCP Packet                      OUTGOING TCP Packet
From: 192.168.1.40  >>=== NAT ===>>      From: 24.12.34.56
To: 123.45.67.89                         To: 123.45.67.89

INCOMING TCP Packet                      INCOMING TCP Packet
From: 123.45.67.89  <<=== NAT ===<<      From: 123.45.67.89
To:   192.168.1.40                       To: 24.12.34.56
```

Why to use NAT

When presented with a cable modem in my new apartment I was also presented with another minor problem. How to get Internet access to my roommates, when the cable modem resides in my room? There were a few options I could implement ranging from obtaining extra IP addresses, to setting up a proxy server, to setting up NAT. (Don't let the cable modem example fool you. NAT is powerful enough to masquerade a large network with hundreds or even thousands of computers!)

There are many reasons why I wanted to set up NAT. The number one reason is for saving money. There are two roommates in my house (each with their own PC) and myself with 3 computers. My ISP only allows for three IP addresses per household. This means that there weren't enough IPs to allow every machine internet access.

By using NAT each machine will have a unique (internal) IP address but share the one IP address given to me by my ISP. The cost goes down.

Setup

In order to enable NAT on your OpenBSD machine you will need to turn on IPF and NAT. This is easily accomplished by editing the files listed below (make the changes to the file so it looks like the options below):

/etc/rc.conf (this file used to start services at boot time)

ipfilter=YES
ipnat=YES

/etc/sysctl.conf

net.inet.ip.forwarding=1

After these changes are made, the machine is now ready to for the configuration of NAT.

Configuration

The first step is to configure the IPF rules file (*/etc/ipf.rules*). For the purposes of this document we will allow traffic to pass through this firewall option without any interference. The file should look like this:

```
pass in from any to any
pass out from any to any
```

Again for more information you can read FAQ 6.2

The NAT configuration file (*/etc/ipnat.rules*) has a very simple syntax. For the configuration set forth above, the file should contain the following entry:

```
map pn0 192.168.1.0/24 -> 24.5.0.5/32 portmap tcp/udp 10000:60000
map pn0 192.168.1.0/24 -> 24.5.0.5/32
```

Here is an explaination for the above lines.

"map"

This is the command you are giving ipnat. It is telling ipnat that this entry is an entry to change IP addresses between the LAN and the Internet.

"pn0"

This is the network interface that is connected to the Internet.

"192.168.1.0/24"

the IP address and netmask (the netmask is in CIDR format). Combined they state "any IP address of value 192.168.1.1 through 192.168.1.254" should be mapped. If you would prefer not to use CIDR notation you can substitute "/24" for "/255.255.255.0".

"24.5.0.5/32"

This IP address and netmask state the IP address that the LAN IP addresses will be mapped to. /32 means one single IP address. You can also map to a /24, or 256 IP addresses (or a /27, or whatever number of bits you'd like)!! This is useful if you have several thousand client machines behind your NAT.... (Of course, this is only useful if that /24 is being routed to your OpenBSD box!)

"portmap tcp/udp 10000:60000"

This maps all tcp/udp packets to ports in the range of 10000 to 60000.

The second line has almost the same entry except for the last portion. This tells ipnat to map anything else (not tcp/udp, those packets are already matched by the first line) to whatever port it requests (used for ICMP, and other protcols). Once this is in the file, all that's needed is to run the IPF daemon.

Running

Executing NAT is a very simple process also. Once the configuration is complete, there are two ways to enable NAT. The first (and best way if possible-to test the setup stage) is to reboot your OpenBSD machine. This is accomplished with the command "*reboot*"

If you would like to run ipnat from the command line, use the following commands:

```
# ipf -Fa -f /etc/ipf.rules -E
# ipnat -CF -f /etc/ipnat.rules
```

The first line is to enable IPF (remember that NAT piggy-backs on IPF therefore IPF must be initialized and running before NAT can be loaded). The options on the command line "-Fa" clear out any existing entries already in effect. "-f /etc/ipf.rules" tells ipf where the rules file can be found. "-E" is the switch to enable the IPF daemon.

The second command line is to enable NAT. "-CF" clears and flushes all existing entries in the NAT table. "-f /etc/ipnat.rules" tells NAT where the NAT rules file is at. NAT is now running. It's as simple as that.

Note: in order to reload the NAT settings (in case you edit the file but don't want to reboot) just execute the 2nd command over again. The settings will be flushed and reloaded.

6.3.3 Nat Knowledge Base

Checking NAT Status

To find out how NAT is doing or make sure the settings have taken effect, you use the "-l" option. This option will list all the settings and current sessions that ipnat is running:

```
# ipnat -l
map pn0 192.168.1.0/24 -> 24.5.0.5/32 portmap tcp/udp 10000:60000
map pn0 192.168.1.0/24 -> 24.5.0.5/32

List of active sessions:
MAP 192.168.1.40  2473  <- -> 24.5.0.5  13463 [129.128.5.191 80]
```

The purpose of the first two lines is to confirm the settings that were entered in /etc/ipnat.rules earlier. The line(s) below will show you a list of the current NAT controlled connections.

"MAP 192.168.1.40 2473"

This tells you the IP address of the machine on the LAN that is using NAT. The port number used to make the connection is displayed afterwards.

"<- ->"

This shows that NAT is handling the flow of traffic in both directions.

"24.5.0.5 13463"

This denotes that the connection is going to the Internet via IP address 24.5.0.5 and using port 13463.

"129.128.5.191 80"

The IP address and the port being connected to are listed last.

Limitations of NAT (in FTP)

There are a few limitations of NAT. One is with FTP. When a user connects to a remote FTP server and requests information or file, the FTP server will make a connection to the client and transfer the info. This is done on a random free port. This is a problem for users attempting to gain access to FTP servers from within the LAN. When the FTP server sends its information it sends it to the external NIC at a random port. The NAT machine will receive this, but because it has no mappings for the unknown packet and doesn't have any mappings for that port, it will drop the packet and won't deliver it.

The solution to this is to place yourself in "passive mode" in your FTP client. This will tell the server that you want to connect to the server, and not what you just read. Then when you make that connection out NAT will correctly handle your connection.

IP Filter provides another solution for this situation, that is, an ftp proxy which is built-in to the NAT code. To activate it, put something like this before your other NAT mappings.

```
map pn0 192.168.1.0/24 -> 24.5.0.5/32 proxy port ftp ftp/tcp
```

With this in place, the kernel will watch your FTP connections for the "PORT" command coming from the ftp client, and it will replace the IP address and port with it's own outside IP address, and a port of its own choosing. Then it will open up that port and tunnel the data to the port your ftp client asked for. Obviously, this is slightly more resource intensive. But, unless your NAT/IP Filter box is reaching critical mass, you should be fine.

Redirecting Traffic

At times you may need to redirect incoming or outgoing traffic for a certain protocol or port. A good example of this is if there were a server residing inside the LAN running a web server. Incoming connections to your valid Internet IP will find that unless your NAT box is running a web server, no connection can be made. For this purpose we use the NAT 'rdr' directive in the rules file to instruct where to redirect (or route) a particular connection to.

For our example, lets say a web server resides on the LAN with IP address of 192.168.1.80. The NAT rules file needs a new directive to handle this. Add a line similar to the following one to your ipnat.conf:

```
rdr pn0 24.5.0.5/32 port 80 -> 192.168.1.80 port 80
```

The reason for each line is this:

"rdr"

This is the command you are giving ipnat. It is telling ipnat that this entry is an entry to redirect a connection.

"pn0"

This is the network interface that is connected to the Internet.

"24.5.0.5/32"

This means an incoming connection to this IP address (only on pn0, as above)

"port 80"

This is the port (80) that should be redirected. The number "80" didn't have to be used. You can use "port www" also to specify a redirection of port 80. If you would like to use a name instead of a number, the service name and corresponding port, must exist in the file /etc/services.

"192.168.1.80"

The IP address and netmask of the LAN machine which the packets are redirected to. The netmask is always "/32" (and therefore not needed to be specified) so the packets can be redirected to a particular machine.

When the addition is complete reload the NAT rules, and the redirection will start immediately.

NAT versus Proxy

The difference between NAT and an application-based proxy is that the proxy software acts as a middle-man between the Internet and the machines connected on the LAN. This is fine, however each application you want to run on your machine and connect to the Internet through the proxy server MUST be proxy-aware (be able to use a proxy server). Not all applications are able to do this (especially games). Furthermore, there simply are not proxy server applications for all of the Internet services out there. NAT transparently maps your internal network so that it may connect to the Internet. The only security advantage to using a proxy software over NAT is that the proxy software may have been made security aware, and can filter based on content, to keep your Windows machine from getting a macro virus, it can protect against buffer overflows

to your client software, and more. To maintain these filters is often a high-maintenance job.

6.3.4 Links and X-References

OpenBSD files:

- /etc/ipnat.rules - NAT rules file
- /etc/rc.conf - need to edit to start up ipnat and ipf at boot time
- /etc/sysctl.conf - need to edit to enable IP forwarding
- /usr/share/ipf/nat.1 - samples of ipnat.rules

NAT Internet Links:

- http://www.openbsd.org/cgi-bin/man.cgi?query=ipnat&sektion=8
- http://coombs.anu.edu.au/ipfilter/
- http://www.geektools.com/rfc/rfc1631.txt

6.4 - DHCP

6.4.1 DHCP Client

To use the DHCP client dhclient(8) included with OpenBSD, edit /etc/hostname.xl0 (this is assuming your main ethernet interface is xl0. Yours might be ep0 or fxp0 or something else!) All you need to put in this hostname file is 'dhcp'

```
# echo dhcp >/etc/hostname.xl0
```

This will cause OpenBSD to automatically start the DHCP client on boot. OpenBSD will gather its IP address, default gateway, and DNS servers from the DHCP server.

If you want to start a dhcp client from the command line, make sure /etc/dhclient.conf exists, then try:

```
# dhclient fxp0
```

Where fxp0 is the interface that you want to receive dhcp on.

No matter how you start the dhclient, you can edit the /etc/dhclient.conf file to **not** update your DNS according to the dhcp server's idea of DNS by first uncommenting the 'requre' lines in it (they are examples of the default settings, but you need to uncomment them to override dhclient's defaults.)

```
request subnet-mask, broadcast-address, time-offset, routers,
      domain-name, domain-name-servers, host-name, lpr-servers, ntp-servers;
```

and then **remove** domain-name-servers. Of course, you may want to remove hostname, or other settings too.

6.4.2 DHCP Server

If you want to use OpenBSD as a DHCP server dhcpd(8), edit /etc/rc.conf. Set it up so that dhcpd_flags="-q" instead of dhcpd_flags=NO. Put the interfaces that you want dhcpd to *listen* on in /etc/dhcpd.interfaces.

```
# echo xl1 xl2 xl3 >/etc/dhcpd.interfaces
```

Then, edit /etc/dhcpd.conf. The options are pretty self explanatory.

```
        option  domain-name "xyz.mil";
        option  domain-name-servers 192.168.1.3, 192.168.1.5;
```

```
subnet 192.168.1.0 netmask 255.255.255.0 {
        option routers 192.168.1.1;

        range 192.168.1.32 192.168.1.127;
}
```

This will tell your dhcp clients that the domain to append to DNS requests is xyz.mil (so, if the user types in 'telnet joe' then it will send them to joe.xyz.mil). It will point them to DNS servers 192.168.1.3 and 192.168.1.5. For hosts that are on the same network as an ethernet interface on the OpenBSD machine, which is in the 192.168.1.0/24 range, it will assign them an IP address between 192.168.1.32 and 192.168.1.127. It will set their default gateway as 192.168.1.1.

If you want to start dhcpd from the command line, after editing /etc/dhcpd.conf, try:

```
# dhcpd -q fxp0
```

Where fxp0 is an interface that you want to start serving dhcp on. The -q flag makes dhcpd quiet, otherwise it is very noisy.

If you are serving DHCP to a Windows box, you may want to dhcpd to give the client a 'WINS' server address. To make this happen, just the following line to your /etc/dhcpd.conf:

```
option  netbios-name-servers    192.168.92.55;
```

(where 192.168.92.55 is the IP of your Windows or Samba server.) See dhcp-options(5) for more options that your DHCP clients may want.

6.5 - PPP

Point-to-Protocol is generally what is used to create a connection to your ISP via your modem. OpenBSD has 2 ways of doing this.

- pppd(8) - Which is the kernel ppp daemon.
- ppp(8) - Which is the userland ppp daemon.

The first one we will cover will be the userland PPP daemon. To start off you will need some simple information about your isp. Here is a list of helpful information that you will need.

- Your ISP's dialup number
- Your nameserver
- Your username and password.
- Your gateway

Some of these you can do without, but would be helpful in setting up your ppp. The userland PPP daemon uses the file /etc/ppp/ppp.conf as its configuration file. There are many helpful files in **/etc/ppp** that can have different setups for many different situations. You should take a browse though that directory.

Also, make sure, that if your not using a GENERIC kernel, that you have this line in your configuration file:

```
pseudo-device   tun             2
```

Initial Setup - for PPP(8)

Initial Setup for the userland PPP daemon consists of editing your **/etc/ppp/ppp.conf** file. This file doesn't exist by default, but there is a file **/etc/ppp/ppp.conf.sample** in which you can simply edit to create your own **ppp.conf** file. Here I will start with the simplist setup and probobly most used setup. Here is a quick **ppp.conf** file that will simply connect to your ISP and set your default routes and nameserver. With this file all the information you need is your ISP's phone number and your username and password.

```
default:
set log Phase Chat LCP IPCP CCP tun command
set device /dev/cua01
set speed 115200
set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 5 \"\" AT OK-AT-OK ATE1Q0
OK\\dATDT\\T TIMEOUT 40 CONNECT"
```

The section under the **default:** tag will get executed each time. Here we setup all our critical information. Here with "set log" we set our logging levels. This can be changed, refer to ppp(8) for more info on setting up logging levels. Our device gets set with "set device". This is the device that the modem is on. In this example the modem is on com port 2. Therefore com port 1 would be /dev/cua00. With "set speed" we set the speed of our dialup connection and with "set dial" we set our dialup parameters. With this we can change our timeout time, etc. This line should stay pretty much as it is though.

Now we can move on and setup our information specific to our ISP. We do this by adding another tag under our **default:** section. This tag can be called anything you want, easiest to just use the name of your ISP. Here I will use **myisp:** as our tag refering to our ISP. Here is a simple setup incorporating all we need to get ourselves connected.

```
myisp:
set phone 1234567
set login "ABORT NO\\sCARRIER TIMEOUT 5 ogin:--ogin: ppp word: ppp"
set timeout 120
set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.0 0.0.0.0
add default HISADDR
enable dns
```

Here we have setup essential info for that specific ISP. The first option "set phone" sets your ISP's dialup number. The "set login" sets our login options. Here we have the timeout set to 5, this means that we will abort our login attempt after 5 seconds if no carrier. Otherwise it will wait for "login:" to be sent and send in your username and password. In this example our Username = ppp and Password = ppp. These values will need to be changed. The line "set timeout" sets the timeout for the entire login process to 120 seconds. The "set ifaddr" line is a little tricky. Here is a more extensive explaination.

```
set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.0 0.0.0.0
```

In the above line, we have it set in the format of "**set ifaddr [myaddr[/nn] [hisaddr[/nn] [netmask [triggeraddr]]]]**". So the first IP specified is what we want as our IP. If you have a static IP address, you set it here. In our example we use /0 which says that no bits of this ip address need to match and the whole thing can be replaced. The second IP specified is what we expect as their IP. If you know this you can specify it. Again in our line we don't know what will be assigned, so we let them tell us. The third option is our netmask, here set to 255.255.255.0.If triggeraddr is specified, it is used in place of myaddr in the initial IPCP negotiation. However, only an address in the myaddr range will be accepted. This is useful when negotiating with some PPP implementations that will not assign an IP number unless their peer requests ``0.0.0.0".

The next option used "add default HISADDR" sets our default route to their IP. This is 'sticky', meaning that if their IP should change, our route will automatically be updated. With "enable dns" we are telling our ISP to authenticate our nameserver addresses. Do NOT do this if you are running an local DNS, as ppp will simply circumvent its use by entering some nameserver lines in /etc/resolv.conf.

Using PPP(8)

Now that we have our **ppp.conf** file setup we can start trying to make a connection to our ISP. I will detail some commonly used arguments with ppp.

- **ppp -auto myisp** - This will run ppp, configure your interfaces and connect to your isp and then go into the background.
- **ppp -ddial myisp** - This is similar to -auto, but if your connection is dropped it will try and reconnect.

By using **/usr/sbin/ppp** with no options will put you into interactive mode. From here you can interact directly with the modem, it is great for debugging problems in your **ppp.conf** file.

ppp(8) extra's

In some situations you might want commands executed as your connection is made or dropped. There are two files you can create for just these situations. **/etc/ppp/ppp.linkup** and **/etc/ppp/ppp.linkdown**. Sample configurations can be viewed here:

- ppp.linkup
- ppp.linkdown

6.6 - Tuning networking parameters

6.6.1 - How can I tweak the kernel so that there are a higher number of retries and longer timeouts for TCP sessions?

You would normally use this to allow for routing or connection problems. Of course, for it to be most effective, both sides of the connection need to use similar values.

To tweak this, use `sysctl` and increase the values of:

```
net.inet.tcp.keepinittime
net.inet.tcp.keepidle
net.inet.tcp.keepintvl
```

Using sysctl -a, you can see the current values of these (and many other) parameters. To change one, use `sysctl -w`, as in `sysctl -w net.inet.tcp.keepidle=28800`.

6.6.2 - How can I turn on directed broadcasts?

Normally, you don't want to do this. This allows someone to send traffic to the broadcast address(es) of your connected network(s) if you are using your OpenBSD box as a router.

There are some instances, in closed networks, where this may be useful, particularly when using older implementations of the NetBIOS protocol. This is another sysctl. `sysctl -w net.inet.ip.directed-broadcast=1` turns this on. Read about smurf attacks if you want to know why it is off by default.

6.6.3 - I don't want to the kernel to dynamically allocate a certain port

There is a sysctl for this also. From sysctl(8):

```
Set the list of reserved TCP ports that should not be allocated by the
kernel dynamically.  This can be used to keep daemons from stealing a
specific port that another program needs to function.  List elements may
be separated by commas and/or whitespace.

   sysctl -w net.inet.tcp.baddynamic=749,750,751,760,761,871

It is also possible to add or remove ports from the current list.

   sysctl -w net.inet.tcp.baddynamic=+748
   sysctl -w net.inet.tcp.baddynamic=-871
```

[Back to Main Index] [To Section 5.0 - Kernel configuration and Disk Setup] [To Section 7.0 - Keyboard controls]

---

www@openbsd.org

$OpenBSD: faq6.html,v 1.37 1999/10/07 19:14:08 chris Exp $

OpenBSD

7.0 - Keyboard Controls

Table of Contents

7.1 - How do I remap the keyboard?

By using **kcon(1)**, "kcon - keyboard control and remapping for the pcvt driver"

Example:

```
bsd# kcon -m gb
```

This will load the keycap file for Great Britain.

7.2 - Is there gpm or the like in OpenBSD?

No. But if you are willing to make a port. Please share. :)

[Back to Main Index] [To Section 6.0 - Networking] [To Section 8.0 - General Questions]


www@openbsd.org

$OpenBSD: faq7.html,v 1.7 1999/09/24 01:46:14 ericj Exp $

OpenBSD

8.0 - General Questions

Table of Contents

8.1 - What are these Kerberos warning when I first login?

When you first install your system you will most likely notice a warning message that is something like this:

Warning: no Kerberos tickets issued.

THIS WARNING IS COMPLETELY IRRELEVANT AND SHOULD ONLY BE REMOVED IF COMPLETELY NECESSARY

Well since you probably haven't set up Kerberos on your system, you wouldn't be getting a ticket. If you do have Kerberos running, you need to check into that. See faq10 Kerberos Setup FAQ. If you can't STAND that warning and never plan on using Kerberos here is how to get rid of it for good.

- Make sure you have the source code.
- Edit /usr/share/mk/bsd.own.mk and set KERBEROS to 'no', or create an /etc/mk.conf file that does this.
- cd /usr/src/usr.bin/login ; make clean ; make ; make install

8.2 - How do I change virtual terminals?

Simply type [ctrl] - [alt] - [One of the function keys] or just hit [f9]-[f12] which are just mapped to [ctrl] - [alt] functions. (i386)

Only the i386 arch has virtual terminal capabilities. You can also use virtual terminals when using X. For example, If you start X on term 1, and switch with [ctrl]-[alt]-[F2] to term 2, X will seem to disapear and will not come back by simply switching back to term 1. You must [ctrl]-[alt]-[F5] to get your X display back.

8.3 - I forgot my root password, what do I do now?

A few steps to recovery

1. Boot into single user mode. For i386 arch type boot -s at the boot prompt.
2. mount the drives.

   bsd# **fsck -p / && mount -u /**
3. If /usr is not the same partition that / is (and it shouldn't be) then you will need to mount it, also

   bsd# **fsck -p /usr && mount /usr**
4. run **passwd**
5. boot into mulituser mode.. and *remember* your password!

8.4 - X won't start, I get lots of error messages

If you have X completely set up and you are using an XF86Config that you know works then the problem most likely lies in the machdep.allowaperture. You also need to make sure that both:

```
option XSERVER
option APERTURE
```

are in your kernel configuration. [BOTH these are in the GENERIC kernel]

Then you need to edit /etc/sysctl.conf and set machdep.allowaperture=1. This will allow X to access the aperture driver. This would be set up if the question during install about whether or not you would be running X was answered correctly. OpenBSD requires for all X servers that the aperture driver be set, because it controls access to the I/O ports on video boards.

For other X problems on the i386, consult the XFree86 FAQ at http://www.xfree86.org/FAQ/.

8.5 - What is CVS? and How do I use it?

CVS is what the OpenBSD project uses to control changes to the source code. CVS stands for Concurrent Versions System. You can read more about CVS at http://www.cyclic.com/. CVS can be used by the end user to keep up to date with source changes, and changes in the ports tree. CVS makes it extremely simple to download the source via one of the many CVS mirrors for the project.

How to initially setup your CVS environment

There are a few ways to initially set up your CVS environment. To start off, you will need an initial CVS checkout of the sources. If you bought the CD, you're in luck, because it holds the CVS checkout for that release. You can extract it from your CD by doing one of the following:

1) copy the tree off it, (assuming the CD is mounted on /mnt):

```
# mkdir /usr/src
# cd /mnt; cp -Rp CVS Makefile bin distrib etc games gnu \
include kerberosIV lib libexec lkm regress sbin share \
sys usr.bin usr.sbin /usr/src
```

2) Or, alternatively, use a union mount with the CD below a writable directory. However, be aware that the union filesystem code is not flawless.

```
# mkdir /usr/src
# mount -t union -o -b /mnt /usr/src
```

after this, /usr/src will be a nice checkout area where all cvs(1) commands will work OK.

If you don't have an OpenBSD CD, you will have to retrieve the sources from on of the OpenBSD AnonCVS servers. These servers are listed on http://www.openbsd.org/anoncvs.html Once you have chosen a server you need to choose which module you are going

to retrieve. There are three main modules available for checkout from the CVS tree. These are:

- *src* - The src module has the complete source code for OpenBSD. This includes userland and kernel sources.
- *ports* - The ports module holds all you need to have the complete OpenBSD ports tree. To read more on the OpenBSD ports tree, read Section 8.6 of the OpenBSD FAQ.
- *www* - The www module contains all OpenBSD web pages, including this FAQ.

Now that you have decided which module that you wish to retrieve, there is one more step left before you can retrieve it. You must decide which method to use. CVS by default retrieves files using rsh(1), but some AnonCVS servers don't allow for this so in most cases it's best to use ssh. For those of you behind a firewall there are also the options of pserver and some AnonCVS servers run ssh on port 2022. Be sure to check http://www.openbsd.org/anoncvs.html for which servers support what protocols. Next I will show how to do a simple source checkout. Here I will be using an AnonCVS server located in the U.S., but remember that if you are outside of the U.S you need to use a server that is located nearby. There are many AnonCVS servers located throughout the world, so choose one nearest you. I will also be using ssh to retrieve the files.

```
ericj@oshibana:~> export CVS_RSH=/usr/local/bin/ssh
ericj@oshibana:~> echo $CVS_RSH
/usr/local/bin/ssh
ericj@oshibana:~> export CVSROOT=anoncvs@anoncvs.usa.openbsd.org:/cvs
ericj@oshibana:~> cvs get src
Warning: Remote host denied X11 forwarding, perhaps xauth program could not be run on
the server side.
cvs checkout: in directory src:
cvs checkout: cannot open CVS/Entries for reading: No such file or directory
cvs server: Updating src
U src/Makefile
[snip]
```

Notice here also that I set the CVSROOT environmental variable. This is the variable that tells cvs(1) which AnonCVS server to use. This can also be specified using the *-d* option. For example:

```
ericj@oshibana:~>cvs -d anoncvs@anoncvs.usa.openbsd.org:/cvs get src
```

These commands should be run in */usr*, which will then create the directories of */usr/src*, */usr/ports*, and */usr/www*. Depending, of course, on which module you checkout. You can download these modules to anywhere, but if you wanted to do work with them (ie make build), it is expected that they be at the place above.

Keeping your CVS tree up-to-date

Once you have your initial tree setup, keeping it up-to-date is the easy part. You can update your tree at any time you choose, some AnonCVS servers update more often then others, so again check http://www.openbsd.org/anoncvs.html. In this example I will be updating my www module from anoncvs.usa.openbsd.org. Notice the -q option that I use, this makes the output not so verbose coming from the server.

```
ericj@oshibana:~> echo $CVSROOT
anoncvs@anoncvs.usa.openbsd.org:/cvs
ericj@oshibana:~> cvs -q up -PAd www
Warning: Remote host denied X11 forwarding, perhaps xauth program could not be run on
the server side.
? www/faq/Makefile
? www/faq/txtdump.sh
U www/want.html
M www/faq/faq8.html
ericj@oshibana:~>
```

Other cvs options

For some, bandwidth and time are serious problems when updating repositories such as these. So CVS has a -z[1-9] option which uses gzip to compress the data. To use it, do -z[compression-level], for instance, -z3 for a compression level of 3.

8.6 - What is the ports tree?

The ports tree is a set of Makefiles that download, patch, configure and install userland programs so you can run them in OpenBSD environment without having to do all that by hand. You can get the ports tree from any of the OpenBSD ftp servers in /pub/OpenBSD/2.5/ports.tar.gz or try the latest updates in /pub/OpenBSD/snapshots/ports.tar.gz.

If you want to run one of the latest ports collections, you'll need to update your /usr/share/mk/ directory. CVS up your /usr/src/ directory and do a:

```
# cd /usr/src/share/mk/
# make install
```

first.

A snapshot of the ports tree is also created daily and can be downloaded from any of the OpenBSD ftp servers as **/pub/OpenBSD/snapshots/ports.tar.gz**.

In 2.5-current this information is now stored in the ports tree. In **ports/infrastructure/**. So, once you are running the latest snapshot, with the infrastructure dir, you no longer need to update bsd.port.mk per instructions above.

Ports are set up to be EXTREMELY easy to make and install. Here is an example install for someone wanting to install the X11 program xfig. You'll notice the dependencies are automatically detected and completed:

First you need to cd to the dir of the program you want. If you are searching for a program, you can either update your locate database, or use the search function talked about below. Once you are in the dir of the program you want, you can just type make install. For example.

```
fenetyllin:/usr/ports/graphics/xfig# make install
===>  Extracting for xfig-3.2.2
===>  xfig-3.2.2 depends on shared library: jpeg.62. - /usr/local/lib/libjpeg.so.62.0
found
===>  xfig-3.2.2 depends on shared library: Xaw3d.6. - not found
===>  Verifying install for Xaw3d.6. in /usr/ports/x11/Xaw3d
>> Xaw3d-1.3.tar.gz doesn't seem to exist on this system.
>> Attempting to fetch from ftp://crl.dec.com/pub/X11/contrib/widgets/Xaw3d/R6.1/.
Connected to crl.dec.com.
220 crl.dec.com FTP server (Digital UNIX Version 5.60) ready.
331 Guest login ok, send ident as password.
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
200 Type set to I.
250 CWD command successful.
250 CWD command successful.
Retrieving pub/X11/contrib/widgets/Xaw3d/R6.1/Xaw3d-1.3.tar.gz
local: Xaw3d-1.3.tar.gz remote: Xaw3d-1.3.tar.gz
227 Entering Passive Mode (192,58,206,2,5,14)
150 Opening BINARY mode data connection for Xaw3d-1.3.tar.gz (0.0.0.0,0) (290277
bytes).
100% |***********************************************|   283 KB    00:00 ETA
226 Transfer complete.
290277 bytes received in 101.09 seconds (2.80 KB/s)
221 Goodbye.
===>  Extracting for Xaw3d-1.3
/bin/mkdir -p /usr/ports/x11/Xaw3d/work/xc/lib/Xaw3d/X11/Xaw3d
```

```
cd /usr/ports/x11/Xaw3d/work/xc/lib/Xaw3d/X11/Xaw3d; ln -sf ../../*.h .
===>  Patching for Xaw3d-1.3
===>  Configuring for Xaw3d-1.3
mv -f Makefile Makefile.bak
imake -DUseInstalled -I/usr/X11R6/lib/X11/config
make Makefiles
[snip]
```

You can see a list of both ports and packages by using the pkg_info command.

```
bsd# /usr/sbin/pkg_info
zsh-3.0.5           The Z shell.
screen-3.7.4        A multi-screen window manager.
ssh-1.2.21          Secure shell client and server (remote login program).
emacs-20.2          GNU editing macros.
lynx-2.7.1ac-0.107  An alphanumeric display oriented World-Wide Web Client.
tcsh-6.07.02        An extended C-shell with many useful features.
bash-2.01           The GNU Borne Again Shell.
zip-2.2             Create/update ZIP files compatabile with pkzip.
mm-2.7              Implementation of MIME, the Multipurpose Internet Mail Exten
ircii-2.8.2-epic3.004 An enhanced version of ircII, the Internet Relay Chat client
ispell-3.1.20       An interactive spelling checker.
tin-1.3.970930      TIN newsreader (termcap based)
procmail-3.11p7     A local mail delivery agent.
strobe-1.03         Fast scatter/gather TCP port scanner
lsof-4.15           Lists information about open files.
xntp3-5.92          Network Time Protocol Implementation.
ncftp-2.4.3
 nmh-0.27            The New MH mail handling program
bzip2-0.1p12        A block-sorting file compressor
```

If you are looking for a specific program in the ports tree, you can try

```
fenetyllin:/usr/ports/# make search key="xfig"
Port:   xfig-3.2.2
Path:   /home/ports/graphics/xfig
Info:   A drawing program for X11
Maint:  angelos@openbsd.org
Index:  graphics x11
B-deps: Xaw3d-1.3 jpeg-6b
R-deps: Xaw3d-1.3 ghostscript-5.10 gv-3.5.8 jpeg-6b png-1.0.2 transfig-3.2.1
Archs:  any

Port:   transfig-3.2.1
Path:   /home/ports/print/transfig
Info:   Tools to convert Xfigs .fig files.
Maint:  angelos@openbsd.org
Index:  print
B-deps: jpeg-6b
R-deps: ghostscript-5.10 jpeg-6b png-1.0.2
Archs:  any
[snip]
```

More information about the ports can be found in the ports man page

Our ports tree is constantly being expanded, and if you would like to help please see: http://www.openbsd.org/ports.html

8.7 - What are packages?

Packages are the precompiled binaries of some of the most used programs. They are ready for use on an OpenBSD system. Again, like the ports, packages are very easy to maintain and update. Packages are constantly being added so be sure to check each release for additional packages.

Here is a list of tools used in managing packages.

- pkg_add (1) - a utility for installing software package distributions
- pkg_create (1) - a utility for creating software package distributions
- pkg_delete (1) - a utility for deleting previously installed software package distributions
- pkg_info (1) - a utility for displaying information on software packages

Here is a sample user installing ispell via a package:

bsd# **pkg_add ftp://ftp.openbsd.org/pub/OpenBSD/2.5/packages/**${arch}**/ispell-3.1.20.tgz**

where ${arch} is your architecture

Again, like with ports you can use the pkg_info(1) program to view what is already installed on your system.

To delete packages you will want to use the pkg_delete(1) command. It is used the same way as pkg_add(1).

bsd# **/usr/sbin/pkg_delete ispell-3.1.20**

and it is gone.

Packages can be found on any of the OpenBSD ftp mirror sites, or they can be found on your OpenBSD CDs.

8.8 - Is there any way to use my floppy drive if it's not attached during boot?

Sure. You need to add "flags 0x20" at the end of the fd* entry and recompile your kernel. The line should be read:

```
fd*     at fdc? drive ? flags 0x20
```

After that you would be able to use the floppy drive all the time. It doesn't matter if you plugged it later after boot.

8.9 - Boot time Options - Using the OpenBSD bootloader

When booting your OpenBSD system, you have probably noticed the boot prompt.

boot>

For most people, you won't have to do anything here. It will automatically boot if no commands are given. But sometimes problems arise, or special functions are needed. That's where these options will come in handy. To start off, you should read through the boot(8) man page. Here we will go over the most common used commands for the bootloader.

To start off, if no commands are issued, the bootloader will automatically try to boot **/bsd**. If that fails it will try **/obsd**, and so on till it finds a bootable kernel. You can specify this by hand by typing:

boot> **boot wd0a:/bsd**

or

boot> **b /bsd**

This will work if device wd0a is configured as your root device.

Here is a brief list of options you can use with the OpenBSD kernel.

- **-a** : This will allow you to specify an alternate root device after booting the kernel.
- **-c** : This allows you to enter the boot time configuration. Check the Boot Time Config section of the faq.
- **-s** : This is the option to boot into single user mode.

- **-d** : This option is used to dump the kernel into ddb. Keep in mind that you must have DDB built into the kernel.
- **-b** : Shortcut for RB_HALT.

These are entered in the format of: **boot [image [-abcds]]**

For further reading you can read boot_i386(8) man page

8.10 - S/Key

S/Key is a ``one-time password'' scheme. This allows for one-time passwords for use on un-secured channels. This can come very handy for those who don't have the ability to use ssh or any other encrypted channels. OpenBSD's S/Key implementation can use a variety of algorithms as the one-way hash. Here is the list of algorithms available:

- md4
- md5
- sha1
- rmd160.

Setting up S/Key - The first steps

To start off the file */etc/skeykeys* must exist. If this file is not in existence, have the super-user create it. This can be done simply by doing:

```
# touch /etc/skeykeys
```

Once that file is in existence, you can initalize your S/Key. To do this you will have to use skeyinit(1). With skeyinit(1), you will first be prompted for your password to the system. This is the same password that you used to log into the system. Running skeyinit(1) over an insecure channel is completely not recommended, so this should be done over a secure channel (such as ssh) or the console. Once you have authorized yourself with your system password you will be asked for yet another password. This password is the *secret password*, and is **NOT** your system password. The secret password is not limited to 8 characters like system passwords, actually it must be at least 10 characters. A few word phrases are suggested. Here is an example user being added.

```
oshibana:ericj> skeyinit ericj
[Adding ericj]
Reminder - Only use this method if you are directly connected
           or have an encrypted channel.  If you are using telnet
           or rlogin, exit with no password and use skeyinit -s.
Enter secret password:
Again secret password:

ID ericj skey is otp-md5 99 oshi45820
Next login password: HAUL BUS JAKE DING HOT HOG
```

One line of particular importance in here is *ID ericj skey is otp-md5 99 oshi45820*. This gives a lot of information to the user. Here is a breakdown of the sections and their importance.

- *otp-md5* - This shows which one-way was used to create your One-Time Password (otp).
- *99* - This is your sequence number. This is a number from 100 down to 1. Once it reaches one, another secret password must be created.
- *oshi45820* - This is your key.

But of more immediate importance is your password. Your password consists of 6 small words, combined together this is your password, spaces and all.

Actually using S/Key to login.

By now your skey has been initialized, and you have your password. You're ready to login. Here is an example session using s/key to login.

```
oshibana:ericj> ftp localhost
Connected to localhost.
220 oshibana.shin.ms FTP server (Version 6.4/OpenBSD) ready.
Name (localhost:ericj): ericj
331 Password [ otp-md5 96 oshi45820 ] for ericj required.
Password:
230- OpenBSD 2.5-current (OSHIBANA) #8: Tue Jun 22 19:20:16 EDT 1999
230-
230- Welcome to OpenBSD: The proactively secure Unix-like operating system.
230-
230- Please use the sendbug(1) utility to report bugs in the system.
230- Before reporting a bug, please try to reproduce it with the latest
230- version of the code.  With bug reports, please try to ensure that
230- enough information to reproduce the problem is enclosed, and if a
230- known fix for it exists, include that as well.
230-
230 User ericj logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.
```

Some of you might have noticed that my sequence number has changed. *otp-md5 96 oshi45820*. This is because by now I have used s/key to login several times. But how do you get your password after you've logged in once? Well to do this, you'll need to know what sequence number you're using and your key. As you're probably thinking, how can you remember which sequence number you're on? Well this is simple, use skeyinfo(1), and it will tell you what to use. For example here, I need to generate another password for a login that I might have to make in the future. (remember I'm doing this from a secure channel).

```
oshibana:ericj> skeyinfo
95 oshi45820
```

From this I can create the password for my next login. To do so, I'll use skey(1). I can use exactly that output from above to create my password.

```
oshibana:ericj> skey 95 oshi45820
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password:
NOOK CHUB HOYT SAC DOLE FUME
```

I'm sure many of you won't always have a secure connect to create these passwords, and creating them over an insecure connection isn't feasible, so how can you create multiple passwords at one time? Well you can supply skey(1) with a number of how many passwords you want created. This can then be printed out and taken with you wherever you go.

```
oshibana:ericj> skey -n 5 95 oshi45820
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password:
91: SHIM SET LEST HANS SMUG BOOT
92: SUE ARTY YAW SEED KURD BAND
93: JOEY SOOT PHI KYLE CURT REEK
94: WIRE BOGY MESS JUDE RUNT ADD
95: NOOK CHUB HOYT SAC DOLE FUME
```

Notice here though, that the bottom password should be the first used, because we are counting down from 100.

Using S/Key with telnet(1) and rlogin(1)

Using S/Key with telnet(1) and rlogin(1) is done in pretty much the same fashion as with ftp, only your first password must be "s/key". Example:

```
ericj@oshibana> telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

OpenBSD/i386 (oshibana) (ttyp2)

login: ericj
Password:              <------------ "s/key" entered.
otp-md5 98 oshi45821
Response: SCAN OLGA BING PUB REEL COCA
Last login: Thu Oct  7 12:21:48 on ttyp1 from 156.63.248.77
Warning: no Kerberos tickets issued.
OpenBSD 2.5-current (OSHIBANA) #4: Thu Sep  2 23:36:16 EDT 1999

Welcome to OpenBSD: The proactively secure Unix-like operating system.

Please use the sendbug(1) utility to report bugs in the system.
Before reporting a bug, please try to reproduce it with the latest
version of the code.  With bug reports, please try to ensure that
enough information to reproduce the problem is enclosed, and if a
known fix for it exists, include that as well.

You have mail.
ericj@oshibana>
```

Controlling S/Key

For more control over S/Key there is the */etc/skey.access* file. (This does not exist by default, so it must be created.) This file can restrict S/Key in three primary ways.

- permit internet - Which will allow certain IPs to connect and use s/key.
- permit user - Which will allow users by login name access to s/key use.
- permit port - Which is really only useful for assigned ports, (ie dial-up)

If I wanted to allow one single user (ericj), from a certain IP (10.1.1.5), I would create a file like so.

```
# cat /etc/skey.access
permit internet 10.1.1.5
permit user ericj
```

8.11 - Why is my Macintosh losing so much time?

This is caused by a hardware bug. OpenBSD uses clock interrupts to keep track of the current time, but these interrupts have the lowest priority in Apple's architecture. So, under heavy load, (such as disk or network activity) clock interrupts will be lost and the Unix clock will not advance as it should.

MacOS gets around the time problem by always reading the hardware clock. OpenBSD only reads the hardware clock at boot time and thereafter ignores it. You may notice that, at shutdown, the kernel is not confident enough to write the Unix time back into the hardware clock because this time loss problem is well known.

The best solution is to run xntpd (found in the ports collection) and just deal with the occasional lossage. Sometimes the lossage is so bad that even xntpd is afraid to skip the time. In this case, add the -g option to ntpd in /etc/rc.securelevel to force tracking.

See also: http://www.macbsd.com/macbsd/macbsd-docs/faq/faq-3.html#ss3.17

8.12 - Will OpenBSD run on multiprocessor machines?

The answer is in ../i386.html:

```
OpenBSD does not currently support multiple processors, but will run using
one processor on a multi-processor system board.
```

The reason is quite simple: there are just not enough developers who have access to SMP machines. If you want to donate SMP hardware, please refer to ../donations.html to have support in the future.

[Back to Main Index] [To Section 7.0 - Keyboard controls] [To Section 9.0 - Tips for linux users]

---

www@openbsd.org

$OpenBSD: faq8.html,v 1.34 1999/10/07 21:11:34 ericj Exp $

OpenBSD

9.0 - Migrating from Linux

Table of Contents

9.1 - Simple tips for Linux (and other free Unix-like OS) users

There are several differences between OpenBSD and Linux. These differences include but are not limited to, bootup procedure, network interface usage and disk management. Most differences are well documented, but involve searching manpages. This document tries to be an index of those differences.

- OpenBSD has a ports tree. This is to accomodate the fact that at this point not many applications are native to the OpenBSD environment. This is both an attempt to get applications to work on OpenBSD for end-users and to get more applications made with OpenBSD in mind. Eventually this ports tree will be used to make a nice set of binary packages.
- OpenBSD uses CVS for source changes. With Linux, source code is disseminated through separate distributions. OpenBSD has pioneered anonymous CVS, which allows anyone to extract the full source tree for any version of OpenBSD (from 2.0 to current, and all revisions of all files in between) at any time! There is also a very convenient and easy to use web interface to CVS.
- OpenBSD periodically releases snapshots for various architectures and makes a stable, official CD release every 6 months.
- OpenBSD contains STRONG CRYPTO, which USA based OS's can't contain. (See http://www.openbsd.org/crypto.html) OpenBSD has also gone through heavy security auditing and many security features have already been implemented into the source tree. (IPSEC, KERBEROS).
- OpenBSD's kernel is /bsd.
- The names of hard disks are usually /dev/wd and /dev/sd (ATA/SCSI)
- /sbin/ifconfig with no arguments in Linux gives the state of all the interfaces. Under OpenBSD you need the -a flag.
- /sbin/route with no arguments in Linux gives the state of all the active routes. Under OpenBSD you need the "show" parameter, or do a netstat -r (nice).
- OpenBSD comes with Darren Reed's IP Filter package, not ipfw. This means that:
  - IP-Masquerading is done through ipnat. (ipnat(1))
  - ipfwadm is done through ipf (ipf(1), ipf(5))
  - You should look at section 6 for detailed configuration assistance and information.

- Interface address is stored in /etc/hostname.<interfacename>. It can be a name instead of an IP address.
- The machine name is in /etc/myname
- The default gateway is in /etc/mygate
- The network interface aliases are in /etc/ifaliases
- OpenBSD's default shell is /bin/sh, which is the Korn shell. Shells such as bash and tcsh can be added as packages or installed from the ports tree.
- Password management changes a lot. The main files are different. (passwd(1), passwd(5))
- Devices are named by driver, not by type. So for example, there are no eth* devices. It would be ne0 for an ne2000 ethernet card, and xl0 for a 3Com Etherlink XL and Fast Etherlink XL ethernet device, etc.

9.2 - Dual booting Linux and OpenBSD

Yes! it is possible!

Read INSTALL.linux

9.3 - Converting your Linux (or other System 7-style) password file to BSD-style

First, figure out if your Linux password file is shadowed or not. If it is, grab John the Ripper and use the unshadow utility that comes with it to merge your `passwd` and `shadow` files into one System 7-style file.

Using your Linux password file, we'll call it `linux_passwd`, you need to add in ::0:0 between fields four and seven. Awk does this for you.

```
# cat linux_passwd | awk -F : '{printf("%s:%s:%s:%s::0:0:%s:%s:%s\n", $1,
$2,$3,$4,$5,$6,$7); }' > new_passwd
```

At this point, you want to edit the `new_passwd` file and remove the root and other system entries that are already present in your OpenBSD password file or aren't applicable with OpenBSD (all of them). Also, make sure there are no duplicate usernames or user IDs between `new_passwd` and your OpenBSD box's `/etc/passwd`. The easiest way to do this is to start with a fresh `/etc/passwd`.

```
# cat new_passwd >> /etc/master.passwd
# pwd_mkdb
```

The last step, `pwd_mkdb` is necessary to rebuild the /etc/spwd.db and /etc/pwd.db files. It also creates a System 7-style password file (minus encrypted passwords) at /etc/passwd for programs which use it. OpenBSD uses a stronger encryption for passwords, blowfish, which is very unlikely to be found on any system which uses full System 7-style password files. To switch over to this stronger encryption, simply have the users run 'passwd' and change their password. The new password they enter will be encrypted with your default setting (usually blowfish unless you've edited /etc/passwd.conf). Or, as root, you can run `passwd` *`username`*.

9.4 - Getting OpenBSD and Linux to interact

If you are migrating from Linux to OpenBSD, note that OpenBSD has COMPAT_LINUX enabled by default in the GENERIC kernel. To run any Linux binaries that are not statically linked (most of them), you need to follow the instructions on the compat_linux(8) manual page. A simple way to get most of the useful Linux libraries is to install linux_lib from your ports collection. To find out more about the Ports collection read FAQ 8.6. If you already have the ports tree installed use these commands to get linux libraries installed.

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

OpenBSD supports the EXT2FS file system. Use **disklabel *disk*** (where ***disk*** is the device name for your disk.) to see what OpenBSD thinks your Linux partition is (but **don't** use disklabel or fdisk to make any changes to it). For further information on using disklabel read FAQ 14.1.

[Back to Main Index] [To Section 8.0 - General Questions] [To Section 10.0 - System Administration]


www@openbsd.org

$OpenBSD: faq9.html,v 1.15 1999/10/07 06:20:39 chris Exp $

OpenBSD

10.0 - System Administration

Table of Contents

10.1 - Why does it say that I'm in the wrong group when I try to su root?

Existing users must be added to the "wheel" group by hand. This is done for security reasons, and you should be cautious with whom you give access to. On OpenBSD, users who are in the wheel group are allowed to use the su(1) userland program to become root. Users who are not in "wheel" cannot use su(1). Here is an example of a /etc/group entry to place the user **ericj** into the "wheel" group.

If you are adding a new user with adduser(8), you can put them in the wheel group by answering wheel at `Invite` *`user`* `into other groups:` This will add them to /etc/group, which will look something like this:

```
wheel:*:0:root,ericj
```

If you are looking for a way to allow users limited access to superuser privileges, without putting them in the "wheel" group, use sudo(8).

10.2 - How do I duplicate a filesystem?

To duplicate your filesystem use dump(8) and restore(8). For example. To duplicate everything under directory SRC to directory DST, do a:

```
# cd /SRC; dump 0f - . | (cd /DST; restore -rf - )
```

dump is designed to give you plenty of backup capabilities, and it may be an overkill if you just want to duplicate a part of a (or an entire) filesystem. The command tar(1) may be faster for this operation. The format looks very similar:

```
# cd /SRC; tar cf -  . | (cd /DST; tar xpf - )
```

10.3 - How do i start daemons with the system? (Overview of rc(8))

OpenBSD uses an rc(8) style startup. This uses a few key files for startup.

- /etc/rc - Main script. Should not be edited.
- /etc/rc.conf - Configuration file used by */etc/rc* to know what daemons should start with the system.
- /etc/netstart - Script used to initialize the network. Shouldn't be edited.
- /etc/rc.local - Script used for local administration. This is where new daemons or host specific information should be stored.
- /etc/rc.securelevel - Script which runs commands that must be run before the security level changes. See init(8)
- /etc/rc.shutdown - Script run on shutdown. Put anything you want done before shutdown in this file. See rc.shutdown(8)

How does rc(8) work?

The main files a system administrator should concentrate on are */etc/rc.conf*, */etc/rc.local* and */etc/rc.shutdown*. To get a look of how the rc(8) procedure works, here is a the flow:

After the kernel is booted. */etc/rc* is started.

- Filesystems checked. This will always be bypassed if the file /etc/fastboot exists. This is certainly not a good idea though.
- Configuration Variables are read in from */etc/rc.conf*
- Filesystems are mounted
- Clears out */tmp* and preserves any editor files
- Configures the network via */etc/netstart*
  - Configures your interfaces up.
  - Sets your hostname, domainname, etc.
- Starts system daemons
- Does various checks. (quota's, savecore, etc)
- Local daemons are run, ala */etc/rc.local*

Starting Daemons and Services that come with OpenBSD

Most daemons and services that come with OpenBSD by default can be started on boot by simply editing the */etc/rc.conf* configuration file. To start out take a look at the default /etc/rc.conf file. You'll see lines similar to this:

```
ftpd_flags=NO           # for non-inetd use: ftpd_flags="-D"
```

A line like this shows that ftpd is not starting up with the system. (At least not via rc(8), read the Anonymous FTP FAQ to read more about this.) In any case, each line also has a comment showing you the flags for **NORMAL** usage of that daemon or service. This doesn't mean that you must run that daemon or service with those flags. You can always use man(1) to see how you can have that daemon or service start up in any way you like. For example, Here is the default line pertaining to httpd(8).

```
httpd_flags=NO          # for normal use: "" (or "-DSSL" after reading ssl(8))
```

Here you can obviously see that starting up httpd normally no flags are necessary. So a line like: " **httpd_flags=**""" would be necessary. But to start httpd with ssl enabled. (Refer to the SSL FAQ or ssl(8)) You should start with a line like: "httpd_flags="-DSSL"".

Starting up local daemons and configuration

For other daemons that you might install with the system via ports or other ways, you will use the */etc/rc.local* file. For example, I've installed a daemon in which lie's at /usr/local/sbin/daemonx. I want this to start at boot time. I would put an entry into */etc/rc.local* like this:

```
if [ -x /usr/local/sbin/daemonx ]; then
            echo -n ' daemonx';       /usr/local/sbin/daemonx
fi
```

From now on, this daemon will be run at boot. You will be able to see any errors on boot, a normal boot with no errors would show a line like this:

```
Starting local daemons: daemonx.
```

rc.shutdown

/etc/rc.shutdown is a script that is run a shutdown. Anything you want done before the system shuts down should be added to this file. If you have apm, you can also set "powerdown=YES". Which will give you the equivlent of "shutdown -p".

10.4 - Why do users get relaying access denied when they are remotely sending mail through my OpenBSD system?

Try this:

```
# cat /etc/sendmail.cf | grep relay-domains
```

The output may look something like this:

```
FR-o /etc/mail/relay-domains
```

If this file doesn't exist, create it. You will need to enter the users who are sending mail remotely with the following syntax:

```
.domain.com    #Allow any .domain.com user to send mail through us
sub.domain.com #Allow sub.domain.com to send mail through us
```

Don't forget send a 'HangUP' signal to sendmail, (a signal which causes most daemons to re-read their configration file):

```
# kill -HUP `cat /var/run/sendmail.pid`
```

10.5 - I've set up POP, but users have trouble accessing mail thru POP. What can I do?

Most issues dealing with POP are problems with temporary files and lock files. If your pop server sends an error message such as:

```
-ERR Couldn't open temporary file, do you own it?
```

Try setting up your permissions as such:

```
permission in  /var
drwxrwxr-x   2 bin     mail     512 May 26 20:08 mail
```

```
permissions in  /var/mail
-rw-------   1 username   username        0 May 26 20:08 username
```

Another thing to check is that the user actually owns their own /var/mail file. Of course this should be the case (as in, /var/mail/joe should be owned by joe) but if it isn't set correctly it could be the problem!

Of course, making /var/mail writeable by group mail opens up some vague and obscure security problems. It is likely that you will never have problems with it. But it could (especially if you are a high profile site, ISP,...)! Try running cucipop or another POP daemon from the OpenBSD ports collection. Or, you could just have the wrong options selected for your pop daemon (like dot locking). Or, you may just need to change the directory that it locks in (although then the locking would only be valueable for the POP daemon.)

10.6 - Setting up a Secure HTTP server with SSL(8)

This will take you through all the steps it takes to set up your own Secure HTTP server. Since the OpenBSD 2.5 release, OpenBSD has shipped with SSLeay. Because of patented RSA algorigthms, these cannot be shipped with OpenBSD. libssl includes support for SSL version 2, SSL version 3, and TLS version 1. So for now, because of patent restrictions, OpenBSD must ship without a fully-functional libssl. But it is still completely useable. If you are able to use the RSA patented libraries, you can easily upgrade your libssl with a command like this.

pkg_add ftp://ftp.openbsd.org/pub/OpenBSD/${ver}**/packages/**${arch}**/libssl-1.1.tgz**

Where ${ver} is the OpenBSD version number, and ${arch} is your architecture.

The steps shown here are taken in part from the ssl(8) man page. Refer to it for further information. This FAQ entry only outlines how to create an RSA certificate for web servers, not a DSA server certificate. To find out how to do so, please refer to the ssl(8) man page.

Here is how you create your server key and certificate. These will go into **/etc/ssl/private/**. To use the RSA features, you must have upgraded your libssl. Now you can create your key. Using ssleay,

ssleay genrsa -out /etc/ssl/private/server.key 1024

This will create a key of 1024 bits. Next you need to create a **Certificate Signing Request** which is used to get a **CA** or **Certificate Authority** to sign your key. Here is the command to do so.

 # ssleay req -new -key /etc/ssl/private/server.key -out /etc/ssl/private/server.csr

You would now give /etc/ssl/private/server.csr to the **CA** to get it signed. Or you could also sign the certificate for yourself. Here is how you would do this.

 # ssleay x509 -req -days 365 -in /etc/ssl/private/server.csr -signkey /etc/ssl/private/server.key -out /etc/ssl/server.crt

This will sign your key, and it will be valid for up to 365 days. With this done you can now start the httpd server. To start httpd(8) with ssl you must start the daemon with the -DSSL flag. You can start this on boot in **/etc/rc.conf** by changing the httpd line to look like so:

```
httpd_flags="-DSSL"
```

This server will run on port 443.

10.7 - I edited /etc/passwd, but the changes didn't seem to take place. Why?

If you edit /etc/passwd, your changes will be lost. OpenBSD generates /etc/passwd dynamically with pwd_mkdb(8). The main password file in OpenBSD is /etc/master.passwd. According to pwd_mkdb(8),

```
FILES
     /etc/master.passwd  current password file
     /etc/passwd         a Version 7 format password file
     /etc/pwd.db         insecure password database file
```

```
/etc/pwd.db.tmp     temporary file
/etc/spwd.db        secure password database file
/etc/spwd.db.tmp    temporary file
```

In a traditional Unix password file, such as /etc/passwd, everything including the user's encrypted password is available to anyone on the system (and is a prime target for programs such as Crack.) 4.4BSD introduces the master.passwd file, which has an extended format (with additional options beyond what was provided by /etc/passwd) and is only readable by root. For faster access to data, the library calls which access this data normally read /etc/pwd.db and /etc/spwd.db.

OpenBSD does come with a tool with which you should edit your password file. It is called vipw(8). Vipw will use vi (or your favourite editor defined per $EDITOR) to edit /etc/master.passwd. After you are done editing, it will re-create /etc/passwd, /etc/pwd.db, and /etc/spwd.db as per your changes. Vipw also takes care of locking these files, so that if anyone else attempts to change them at the same time, they will be denied access.

10.8 - What is the best way to add and delete users?

The best way to add a user in OpenBSD is to use the adduser(8) script. You can configure this to work however you like by editing **/etc/adduser.conf**. You can add users by hand, but this is the recommended way to add users. adduser(8) allows for consistency checks on **/etc/passwd**, **/etc/group**, and shell databases. It will create the entries and HOME directories for you. It can even send a message to the user welcoming them. This can be changed to meet your needs. For further instructions on adding users read the adduser_proc(8) man page.

To delete users you should use the rmuser(8) utility. This will remove all existence of a user. It will remove any crontab(1) entries, their HOME dir (if it is owned by the user), and their mail. Of course it will also remove their **/etc/passwd** and **/etc/group** entries.

10.9 - How do I create an ftp-only account?

There are a few ways to do this, but a very common way to do such is to add /bin/false into **/etc/shells**. Then when you create the user set his shell to /bin/false, they will not be able log in interactively, but will be able to use ftp capabilities. adduser(8) will give them a home dir by default of /home/. If this is what you desire it doesn't need to be changed, however you can set this to whatever directory you wish.

10.10 - Setting up Quotas

Quotas are used to limit users space that they have available to them on your drives. It can be very helpful in situations where you have limited resources. Quotas can be set in two different ways.

- User Quotas
- Group Quotas

The first step to setting up quotas is to make sure that `option QUOTA` is in your Kernel Configuration. This option is in the GENERIC kernel. After this you need to mark in /etc/fstab the filesystems which will have quotas enabled. The keywords `userquota` and `groupquota` should be used to mark each fs that you will be using quotas on. By default the files `quota.user` and `quota.group` will be created at the root of that filesystem to hold the quota information. This default can be overwritten by doing, userquota=/var/quotas/quota.user. Or whatever file you want to use for holding quota information. Here is an example /etc/fstab that has one filesystem with userquotas enabled.

```
/dev/wd0a / ffs rw,userquota=/var/quotas/quota.user 1 1
```

Now it's time to set the user's quotas. To do so you use the utility edquota(8). A simple use is just **edquota <user>**. edquota(8) will use vi(1) to edit the quotas unless the environmental variable EDITOR is set to a different editor. For example:

edquota ericj

This will give you output similar to this:

```
Quotas for user ericj:
/: blocks in use: 62, limits (soft = 0, hard = 0)
```

```
        inodes in use: 25, limits (soft = 0, hard = 0)
```

To add limits, edit it to give results like this:

```
Quotas for user ericj:
/: blocks in use: 62, limits (soft = 1000, hard = 1050)
        inodes in use: 25, limits (soft = 0, hard = 0)
```

In this the softlimit is set to 1000 blocks and the hardlimit is set to 1050 blocks. A softlimit is a limit where the user is just warned when they cross it and have until their grace period is up to get their disk usage below their limit. Grace periods can be set by using the **-t** option on edquota(8). After the grace period is over the softlimit is handled as a hardlimit. This usually results in an allocation failure.

Now that the quotas are set, you need to turn the quotas on. To do this use quotaon(8). For example:

```
# quotaon -a
```

This will go thru /etc/fstab to turn on the filesystems with quota options. Now that quotas are up and running, you can view them by the quota(1). Using a command of **quota <user>** will give that users information. When called with no arguments the quota command will give your quota statistics. For example:

```
# quota ericj
```

Will result in output similar to this:

```
Disk quotas for user ericj (uid 1001):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
              /      62    1000    1050              27       0       0
```

By default quotas set in /etc/fstab will be started on boot. To turn them off use

```
# quotaoff -a
```

10.11 - How to Setup Kerberos Clients and Servers under OpenBSD

As a user/administrator of OpenBSD systems, you are fortunate that KerberosIV is an pre-installed component of the default system. Here is a guide to setting up both the Kerberos realm server, as well as a client.

An *EXTREMELY* important point to remember is that Kerberos clients and servers must have their system clocks synchronized. If there is more than a 5 minute time skew, you will receive wierd errors that do not immediately reveal themselves to be caused by time skew, such as:

```
kinit: Can't send request (send_to_kdc)
```

Another more accurate error is:

```
kauth: Time is out of bounds (krb_rd_req)
```

An easy way to synchronize system clocks is with xntpd, available in the ports tree at **/usr/ports/sysutils/xntpd/**.

This FAQ entry assumes you have prior knowledge of the Kerberos concepts. For a great, easy to understand, reference, see:

- The FreeBSD handbook
- Use the command **info kth-krb**
- Designing an Authentication System: a Dialogue in Four Scenes
- Papers and Documentation Describing KerberosIV

Or the book

- Network Security Private Communication in a Public World [Kaufman, Perlman, Speciner, 1995]

How to setup the Kerberos IV REALM and SERVER

We will be setting up the CIARASYSTEMS.COM realm, with avalanche.ciarasystems.com as the main server.

To start off, we will need to edit our configuration files. These files are located at `/etc/kerberosIV/`. The two files we are concerned about are *krb.realms* and *krb.conf*. Let's start off with *krb.conf*.

```
[root@avalanche kerberosIV] cat krb.conf
CIARASYSTEMS.COM
CIARASYSTEMS.COM avalanche.ciarasystems.com admin server
```

As you can see, this tells kerberos that the domain is CIARASYSTEMS.COM (or logical realm) and that within that domain, avalanche is the administration server. Next we will look at *krb.realms*. For more information on this refer to krb.conf.

```
[root@avalanche kerberosIV] cat krb.realms
avalanche.ciarasystems.com      CIARASYSTEMS.COM
.ciarasystems.com               CIARASYSTEMS.COM
```

krb.realms provides a translation from a hostname to the Kerberos realm name for the services provided by that host. Each line of the translation file is in one of the following forms (domain_name should be of the form .XXX.YYY). So in this example, avalanche is the hostname of a computer on the CIARASYSTEMS.COM realm. And .ciarasystems.com is the domain name on the realm CIARASYSTEMS.COM. Again, for further information read the krb.realms. man page.

Next we will run kdb_init(8) to create the initial Kerberos database.

```
[root@avalanche kerberosIV] kdb_init
Realm name [default  NO.DEFAULT.REALM ]: CIARASYSTEMS.COM
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.

Enter Kerberos master password: not shown
Verifying password -
Enter Kerberos master password:
```

Next we need to use kstash(8) which is used to save the Kerberos key distribution center (KDC) database master key in the master key cache file.

```
[root@avalanche kerberosIV] kstash
Enter Kerberos master password:

Current Kerberos master key version is 1.

Master key entered.  BEWARE!
Wrote master key to /etc/kerberosIV/master_key
```

This saves the encrypted master password in /etc/kerberosIV/master_key.

Next, we need two principals 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 Kerberospasswords and run commands like *rcp, rlogin* and *rsh*.

```
# 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: avalanche

<Not found>, Create [y] ? y

Principal: passwd, Instance: avalanche, kdc_key_ver: 1
New Password:                <----- Use 'RANDOM' as password
Verifying password -
New Password:

Random password [y] ? y

Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 1999-12-31 ] ? 2001-12-31
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.


Principal name: rcmd
Instance: avalanche

, Create [y] ? y
Principal: rcmd, Instance: avalanche, kdc_key_ver: 1
New Password:                  <----- Use 'RANDOM' as password
Verifying password -
New Password:

Random password [y] ? y

Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 1999-12-31 ] ? 2001-12-31
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name:                <----- Hit <ENTER> to end
```

A srvtab file is the service key file. These must be extracted from the Kerberos key distribution center database in order for services to authenticate using Kerberos. For each hostname specified on the command line, ext_srvtab(8) creates the service key file hostname-new-srvtab, containing all the entries in the database with an instance field of hostname.

```
[root@avalanche kerberosIV] ext_srvtab avalanche

Enter Kerberos master password:

Current Kerberos master key version is 1.

Master key entered.  BEWARE!
```

```
Generating 'avalanche-new-srvtab'....

[root@avalanche kerberosIV] mv avalanche-new-srvtab srvtab
[root@avalanche kerberosIV] chmod 600 srvtab
```

Now we can add users to our database.

```
[root@avalanche kerberosIV] 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: jeremie
Instance:

, Create [y] ? y

Principal: jeremie, 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
                               or you can add more entries.
```

So now all the Kerberos particulars are setup. All that is left is to enable boot-time loading of the Kerberos server and to enable Kerberized-daemons.

In /etc/rc.conf, set:

```
kerberos_server=YES
```

In /etc/inetd.conf, uncomment:

```
telnet       stream  tcp     nowait  root    /usr/libexec/telnetd    telnetd -k
klogin       stream  tcp     nowait  root    /usr/libexec/rlogind    rlogind -k
kshell       stream  tcp     nowait  root    /usr/libexec/rshd       rshd -k
kauth        stream  tcp     nowait  root    /usr/libexec/kauthd     kauthd
```

Then, either reboot, or:

```
[root@avalanche /] kill -HUP `cat /var/run/inetd.pid`
[root@avalanche /] /usr/libexec/kerberos >> /var/log/kerberos.log &
[root@avalanche /] /usr/libexec/kadmind -n >> /var/log/kadmind.log &
```

Note: this is a rather simple server setup. Usually, redundant servers are setup (as slave servers) so that if one server goes down, all the services that depend on Kerberos don't go down. We can also add 'su' privileges to a specific principal, see the FreeBSD Handbook.

How to kerberize your client workstation

We will be setting the workstation named *gatekeeper* to be in the CIARASYSTEMS.COM realm, with avalanche.ciarasystems.com as the main server.

To start off, we need to setup our *krb.conf* and *krb.realms* like the above machine. This is so *gatekeeper* will know what server is the KDC and what domain it is on. Again here are the file contents.

```
[root@gatekeeper kerberosIV] cat krb.conf
CIARASYSTEMS.COM
CIARASYSTEMS.COM avalanche.ciarasystems.com admin server

[root@gatekeeper kerberosIV] cat krb.realms
avalanche.ciarasystems.com      CIARASYSTEMS.COM
.ciarasystems.com               CIARASYSTEMS.COM
```

Now that is set up, we need to initialize kerberos. To obtain a ticket you use kinit(1).

```
xyz:jeremie% kinit
The OpenBSD Project (gatekeeper)
Kerberos Initialization
Kerberos name: jeremie
Password:
```

Now we have identified we can list our tickets with klist(1).

```
xyz:jeremie$ klist
Ticket file:    /tmp/tkt1000
Principal:      jeremie@CIARASYSTEMS.COM

  Issued           Expires          Principal
Jun 28 01:03:25  Jun 28 11:03:25  krbtgt.CIARASYSTEMS.COM@CIARASYSTEMS.COM
```

Looks like we are set now. All that's left to do is test it. Here we will test it with rlogin(1) and telnet(1).

```
xyz:jeremie% telnet avalanche
Trying 192.168.0.38...
Connected to avalanche.
Escape character is '^]'.
[ Trying mutual KERBEROS4 ... ]
[ Kerberos V4 accepts you ]
[ Kerberos V4 challenge successful ]
Last login: Sun Jun 27 22:52:25 on ttyp1 from gatekeeper
Warning: no Kerberos tickets issued.
OpenBSD 2.5 (AVALANCHE) #5: Tue Apr  6 01:18:16 EDT 1999
```

and

```
xyz:jeremie% rlogin avalanche
Last login: Sun Jun 27 22:53:39 on ttyp1 from gatekeeper
Warning: no Kerberos tickets issued.
OpenBSD 2.5 (AVALANCHE) #5: Tue Apr  6 01:18:16 EDT 1999
```

We can tell that it is indeed using Kerberos to authenticate the rlogin session. To get rid of any tickets issued, you would use kdestroy(1). For example:

```
xyz:jeremie% kdestroy
Tickets destroyed.
```

```
xyz:jeremie% rlogin avalanche
krcmd: No ticket file (tf_util)
rlogin: warning, using standard rlogin: can't provide Kerberos auth data.
avalanche: Connection refused
```

Do not worry about 'Warning: no Kerberos tickets issued.' This is because we're only doing kerberos authentication, not ticket passing. If you want ticket passing, use Dug Song's krb4 SSH patches. Stock KerberosIV doesn't have support for tgt passing, either - only the AFS kaserver's implementation of krb4, since the regular KerberosIV kdc checks client IP address listed in the ticket.

10.12 - Setting up Anonymous FTP Services.

Anonymous FTP allows users without accounts to access files on your computer via the File Transfer Protocol. This will give an overview of setting up the anonymous FTP server, and its logging, etc.

Adding the FTP account

To start off, you need to have an account on your system of "ftp". This account shouldn't have a useable password, and here we will set the login directory to /home/ftp. When using anonymous ftp, the ftp daemon will chroot itself to /home/ftp. To read up more on that, read the ftp(8) and chroot(2) man pages. Here is an example of adding the *ftp* user. I will do this using adduser(8). We also need to add /bin/false to our */etc/shells*, this is the "shell" that we will be giving to the ftp user. This won't allow them to login, even though we will give them an empty password. To do this you can simply *echo /bin/false >> /etc/shells*. Also if you wish for that shell to show up during the adduser questions, you need to modify */etc/adduser.conf*.

```
oshibana# adduser
Use option ``-silent'' if you don't want see all warnings & questions.

Reading /etc/shells
Check /etc/master.passwd
Check /etc/group

Ok, let's go.
Don't worry about mistakes. I will give you the chance later to correct any input.
Enter username [a-z0-9_]: ftp
Enter full name []: anonymous ftp
Enter shell csh false ksh nologin sh tcsh zsh [sh]: false
Uid [1002]:
Login group ftp [ftp]:
Login group is ``ftp''. Invite ftp into other groups: guest no
[no]: no
Enter password []:
Use an empty password? (y/n) [y]: y

Name:     ftp
Password: ****
Fullname: anonymous ftp
Uid:      1002
Gid:      1002 (ftp)
Groups:   ftp
HOME:     /home/ftp
Shell:    /usr/bin/false
OK? (y/n) [y]: y
Added user ``ftp''
Copy files from /usr/share/skel to /home/ftp
Add another user? (y/n) [y]: n
Goodbye!
```

Directory Setup

Along with the user, this created the directory */home/ftp*. This is what we want, but there are some changes that we will have to make to get it ready for anonymous ftp. Again these changes are explained in the ftp(8) man page.

- */home/ftp* - This is the main directory. It should be owned by root and have permissions of 555.
- */home/ftp/bin* - This directory will hold binaries that can be executed from within the chrooted ftpd environment. This directory should be mode 511. Anonymous FTP only needs ls(1) to operate correctly. So you need to copy /bin/ls to /home/ftp/bin. *~ftp/bin/ls* needs to be mode 111, which is executable only.
- */home/ftp/etc* - Directory to hold user information. You should copy /etc/pwd.db and /etc/group to this directory. This directory should be mode 511, and the two files should be mode 444. These are used to give owner names as opposed to numbers. The password fields in pwd.db will not be used.
- */home/ftp/pub* - This is the directory to place files in which you wish to share. This directory should also be mode 555.

Note that all these directories should be owned by "root". Here is a listing of what the directories should look like after their creation.

```
oshibana# pwd
/home
oshibana# ls -laR ftp
total 5
dr-xr-xr-x  5 root  ftp    512 Jul  6 11:33 .
drwxr-xr-x  7 root  wheel  512 Jul  6 10:58 ..
dr-x--x--x  2 root  ftp    512 Jul  6 11:33 bin
dr-x--x--x  2 root  ftp    512 Jul  6 11:34 etc
dr-xr-xr-x  2 root  ftp    512 Jul  6 11:33 pub

ftp/bin:
total 178
dr-x--x--x  2 root  ftp     512 Jul  6 11:33 .
dr-xr-xr-x  5 root  ftp     512 Jul  6 11:33 ..
---x--x--x  1 root  ftp  167936 Jul  6 11:33 ls

ftp/etc:
total 43
dr-x--x--x  2 root  ftp    512 Jul  6 11:34 .
dr-xr-xr-x  5 root  ftp    512 Jul  6 11:33 ..
-r--r--r--  1 root  ftp    316 Jul  6 11:34 group
-r--r--r--  1 root  ftp  40960 Jul  6 11:34 pwd.db

ftp/pub:
total 2
dr-xr-xr-x  2 root  ftp  512 Jul  6 11:33 .
dr-xr-xr-x  5 root  ftp  512 Jul  6 11:33 ..
```

Starting up the server and logging

With ftpd you can choose to either run it from inetd or the rc scripts can kick it off. These examples will show our daemon being started from inetd.conf. First we must become familiar with some of the options to ftpd. The default line from */etc/inetd.conf* is:

```
ftp             stream  tcp     nowait  root    /usr/libexec/ftpd       ftpd -US
```

Here ftpd is invoked with *-US*. This will log anonymous connections to */var/log/ftpd* and concurrent sessions to */var/run/utmp*. That will allow for these sessions to be seen via who(1). For some, you might want to run only an anonymous server, and disallow ftp for users. To do so you should invoke ftpd with the *-A* option. Here is a line that starts ftpd up for anonymous connections only. It also uses *-ll* which logs each connection to syslog, along with the get, retrieve, etc, ftp commands.

```
ftp             stream  tcp     nowait  root    /usr/libexec/tcpd       ftpd -llUSA
```

Note - For people using HIGH traffic ftp servers, you might want to not invoke ftpd from inetd.conf. The best option is to comment the ftpd line from inetd.conf and start ftpd from rc.conf along with the *-D* option. This will start ftpd as a daemon, and has much less overhead as starting it from inetd. Here is an example line to start it from rc.conf.

```
ftpd_flags="-DllUSA"            # for non-inetd use: ftpd_flags="-D"
```

This of course only works if you have ftpd taken out of */etc/inetd.conf*.

Other relevant files

- */etc/ftpwelcome* - This holds the Welcome message for people once they have connected to your ftp server.
- */etc/motd* - This holds the message for people once they have successfully logged into your ftp server.
- *.message* - This file can be placed in any directory. It will be shown once a user enters that directory.

[Back to Main Index] [To Section 9.0 - Migrating from Linux] [To Section 11.0 - Performance Tuning]


www@openbsd.org

$OpenBSD: faq10.html,v 1.31 1999/10/07 20:53:58 ericj Exp $

OpenBSD

11.0 - Performance Tuning

Table of Contents

11.1 - Networking

If you run a busy server, gateway or firewall, you should make sure to prevent memory starvation to various parts of the kernel described below.

The options(4) man page talks about the options presented.

An option you may need to change for a busy server, gateway or firewall is NMBCLUSTERS. This controls the size of the kernel mbuf cluster map. On your computer, if you get messages like "mb_map full", you need to increase this value. If traffic on a networking interface stops for no apparent reason, this may also be a sign that you need to increase this value. A reasonable value on the i386 port with most 100Mbps ethernet interfaces (no matter how many the machine has) is 8192.

option NMBCLUSTERS=8192

11.2 - Disk I/O

Disk I/O speed is a significant factor in the overall speed of your computer. It becomes increasingly important when your computer is hosting a multi-user environment (users of all kinds, from those who log-in interactively to those who see you as a file-server or a web-server.) Data storage constantly needs attention, especially when your partitions run out of space and when your disks fail. OpenBSD has several options to increase the speed of your disk operations and provide fault tolerance.

CCD

The first option is the use of ccd(4), the Concatenated Disk Driver. This allows you to join several partitions into one virtual disk (and thus, you can make several disks look like one disk). This concept is similar to that of LVM (logical volume management), which is found in many commercial Unix flavors.

If you are running GENERIC, ccd is already enabled. If not, you may need to add it to your kernel configuration. To start the setup of ccd, you need to add support for it in your kernel. A line such as:

```
pseudo-device   ccd     4       # concatenated disk devices
```

The above example gives you up to 4 ccd devices (virtual disks). Now you need to figure out what partitions on your real disks that you want to dedicate to ccd. Use disklabel to mark these partitions as type 'ccd'. On some architectures, disklabel may not allow you to do this. In this case, mark them as 'ffs'.

If you are using ccd to gain performance by striping, note that you will not get optimum performance unless you use the same model of disks with the same disklabel settings.

Edit /etc/ccd.conf to look something like this: (for more information on configuring ccd, look at ccdconfig(8))

```
# Configuration file for concatenated disk devices
#
# ccd   ileave  flags   component devices
ccd0   16      none    /dev/sd2e /dev/sd3e
```

To make your changes take effect, run

```
# ccdconfig -C
```

As long as /etc/ccd.conf exists, ccd will automatically configure itself upon boot. Now, you have a new disk, ccd0, a combination of /dev/sd2e and /dev/sd3e. Just use disklabel on it like you normally would to make the partition or partitions you want to use. Again, don't use the 'c' partition as an actual partition that you put stuff on. Make sure your useable partitions are at least one cylinder off from the beginning of the disk.

RAID

Another solution is raid(4) which will have you use raidctl(8) to control your raid devices. OpenBSD's RAID is based upon Greg Oster's NetBSD port of the CMU RAIDframe software. OpenBSD has support for RAID levels of 0, 1, 4, and 5.

With raid, as with ccd, support must be in the KERNEL. Unlike ccd, support for RAID is not found in GENERIC, it must be compiled into your kernel (RAID support adds some 500K to the size of an i386 kernel!)

```
pseudo-device   raid   4       # RAIDframe disk device
```

Setting up RAID on some operating systems is confusing and painful to say the least. Not so with RAIDframe. Read the raid(4) and raidctl(8) man pages to get full details. There are many options and possible configurations, a detailed explanation is beyond the scope of this document.

Filesystem Buffer

For fileservers with memory to spare, you can increase BUFCACHEPERCENT. That is, what percentage of your RAM should you use as a file system buffer. This option may change when the Unified Buffer Cache is completed and is part of OpenBSD. In the mean time, to increase BUFCACHEPERCENT, you should add a line to your kernel configuration like this:

option BUFCACHEPERCENT=30

Of course you can make it as low as 5 percent (the default) or as high as 50 percent (or more.)

Soft updates

Another tool that can be used to speed up your system is softupdates. One of the slowest operations in the traditional BSD file system is updating metainfo (which happens, among other times, when you create or delete files and directories.) Softupdates attempts to update metainfo in RAM instead of writing to the hard disk each and every single metainfo update. Another effect of this is that the metainfo on disk should always be complete, although not always up to date. So, a system crash should not require fsck upon boot up, but simply a background version of fsck that makes changes to the metainfo in RAM (a la softupdates). This means rebooting a server is much faster, because you don't have to wait for fsck! (OpenBSD does not have this feature yet.) You can read more about softupdates in the softupdates FAQ entry. If you use softupdates, you will most certainly want to take advantage of the Tuning kmem section below.

11.3 - Tuning kmem

If you are running 2.5 (not applicable for -current or 2.6), and if you start using the performance tuning measures above, you may start running out of kernel memory. If you start getting panics like "out of space in kmem_map" then you need to try

option NKMEMCLUSTERS=8192

Note that 8192 is valid for the i386 architecture, but may be too little or too much for others. Look at /usr/include/machine/param.h to see more information.

This change is not necessary for anyone running -current on i386 as this has already been cranked up from a commit on September 20th.

You may also want to increase the number of static kernel maps and entries. The default value for these options is architecture dependent and is specified in /sys/vm/vm_map.h. If you are using soft updates, the following values should keep you going!

option MAX_KMAP=120
option MAX_KMAPENT=6000

These changes may be unnecessary for anyone running current..

11.4 - Hardware choices

(Note- this section is heavily centered around the i386, or PC, architecture. That is to say... other architectures don't give you quite as many choices!)

The performance of your applications depends heavily on your OS and the facilities it provides. This may be part of the reason that you are using OpenBSD. The performance of your applications also depends heavily on your hardware. For many folks, the Price/Performance ratio of a brand new PC with a Intel Pentium III or AMD Athlon processor is much better then the Price/Performance ratio of a Sun UltraSparc 60! And, the price of OpenBSD can't be beat.

If you are shopping for a new PC, whether you are buying it piece by piece or completely pre-built, you want to make sure first that you are buying reliable parts. In the PC world, this is not easy. **Bad or otherwise unreliable or mismatched parts can make OpenBSD run poorly and crash often**. The best advice we can give is to be careful, buy brands and parts that have been reviewed by an authority you trust. Sometimes, when you skimp on the price of a PC, you lose in quality!

There are certain things that will help bring out the maximum performance of your hardware:

- Use multiple disks.

  Instead of buying one 20GB disk, buy multiple 4GB or 9GB disks. While this may cost more, distributing the load over multiple spindles will decrease the amount of time necessary to access data on the disks. And, with more spindles, you will get more reliability and faster data access with RAID.
- Use SCSI where you can.

  If you are building a server, and you need more then 10GB of disk space, the SCSI architecture is the best choice. IDE limits you to two disks per controller. Wide SCSI limits you to 15 per controller! While SCSI costs more, the flexibility and performance can justify these costs. IDE was not designed for use in a multi-user, multi-tasking environment.
- Use SDRAM.

  This option applies mainly to PCs. Most other architectures don't give you a choice of what kind of RAM you can use. Several PCs still do. Avoid SIMMs, as there are not many companies manufacturing them anymore, and in some cases they cost more then equivalent SDRAM!
- Use ECC or parity RAM.

  Parity adds some functionality to see if the data in RAM has been corrupted. ECC extends this functionality and attempts to correct some bit corruption errors on the fly. This option applies mainly to PCs. Most other architectures simply require parity or ECC capable RAM. Several non-PC computers won't even boot with non-parity RAM. If you aren't using ECC/parity RAM, you may get data corruption and other abnormalities. Several manufacturers of "cheap PC RAM" don't even make an ECC variety! This will help you avoid them! PC manufacturers often sell several product lines, divided around "servers" and "workstations." The servers will incorporate ECC RAM into their architecture. Unix workstation manufacturers have been using parity (and now ECC) for several years in all of their product lines.
- Avoid ISA devices.

  While most folks avoid ISA devices, because they are generally hard to configure and out of date, there are still plenty in existence. If you are using the ISA bus for your disk or network controllers, (or even worse, for both) remember that the ISA bus itself can be a performance bottleneck. If you need speed, look to PCI. Of course, there are still several ISA bus cards that work just fine. Unfortunately, most of these are sound cards and serial port cards.

11.5 - Why aren't we using async mounts?

Question: "I simply do "mount -u -o async /" which makes one package I use (which insists on touching a few hundred things from time to time) useable. Why is async mounting frowned upon and not on by default (as it is in some other unixen) ? Surely it is much simpler and therefore a safer way of improving performance in some applications ?"

Answer: "Async mounts is indeed faster then sync mounts, but they are also less safe. What happens in case of a power failure? Or a hardware problem? The quest for speed should not sacrifice the reliabilty and the stability of the system. Check the manpage for mount(8)."

```
async   All I/O to the file system should be done asynchronously.
        This is a dangerous flag to set since it does not guaran-
        tee to keep a consistent file system structure on the
        disk.  You should not use this flag unless you are pre-
```

```
pared to recreate the file system should your system
crash.  The most common use of this flag is to speed up
restore(8) where it can give a factor of two speed in-
crease.
```

On the other hand, when you are dealing with temp data that you can recreate from scratch after a crash, you could gain speed by using a separate partition, used for that data only, mounted async. If you don't mind risking the loss of all the data in the partition when something goes wrong...

[to Section 12.0 - For Advanced Users] [To Section 10.0 - System Administration]


www@openbsd.org

$OpenBSD: faq11.html,v 1.3 1999/09/29 22:57:42 ericj Exp $

OpenBSD

12.0 - For Advanced Users

Table of Contents

12.1 - Using OpenBSD's new PCI-IDE code

If you have a current source tree that was last updated after July 25th, you will notice that OpenBSD has PCI-IDE code enabled by default, a big win on performance for anyone using an IDE controller! The PCI-IDE code introduces new PCI and ISA IDE code, and both PCI and ISA versions have DMA (or Direct Memory Access) capability. DMA allows the IDE controller to transfer data right from disk to your memory instead of running it through the CPU. This lowers the overall utilization of your CPU, keeping disk access fast and clean. (Note that OpenBSD has implemented the pciide code from NetBSD.)

```
Date: Mon, 19 Jul 1999 18:07:40 -0700 (PDT)
From: Constantine Sapuntzakis <csapuntz@stanford.edu>
To: tech@openbsd.org
Subject: New ATA/IDE stuff in OpenBSD

This weekend, I started integrating the IDE code from NetBSD-current to
OpenBSD-current. It should be considered a work-in-progress.

I'm sending out this e-mail for three reasons: to tell you about the new
stuff, to ask for help in porting this across all the platforms with IDE,
and to ask for help in testing this new feature.

The code from NetBSD supports DMA, UltraDMA, PCI IDE, better error
recovery, etc. It's also faster too. The code has been extensively tested
in NetBSD but not in OpenBSD. The usual warnings apply: the new drivers
might cause data loss.

The new code also has a different way of handling ATAPI devices (like IDE
CD ROMs, IDE tape drives, etc.). ATAPI devices talk the SCSI packet
protocol and the new ATAPI layer acts like a SCSI adapter. As such, your
ATAPI CD will now appear as SCSI CD (cd*), your ATAPI tape drive as a
SCSI tape drive (st*), your ATAPI zip drive like a SCSI drive (sd*), etc.
(note: some older ZIP drives are not ATAPI devices)

Many ATAPI devices are NOT entirely SCSI-I and SCSI-II compliant. As
such, our SCSI device drivers may not work with your CD/tape drive/ZIP
drive. Integrating ATAPI device support into our SCSI devices is a high
priority so this shouldn't stay broken for long. However, in the interim,
you might get SCSI errors to the console if you try to use the devices.
```

```
Under no condition should your machine hang or panic. If your machine
does hang/panic, please send me mail, describing the steps leading up to
the hang and providing a dmesg dump.

For the best hard disk performance, you should not have an ATAPI device
on the same chain as a hard disk.

Currently, the ATA stuff is only integrated into the i386 port. I am
willing to do the integration work on other platforms, but I need an
account on such a machine to do compiles and your help in testing the
resulting kernel.

To enable this feature on i386, you are going to have to edit some files.
Comments directing what edits to do appear in the following files:

1) arch/i386/conf/files.i386 (don't forget to uncomment pciide_machdep.c)
2) conf/files
3) dev/isa/files.isa
4) dev/pci/files.pci

In addition, look at

5) arch/i386/conf/NEWATA for how to configure the new stuff

If you boot the new stuff, I'd appreciate it if you sent me a dmesg dump.
Once you've got a new kernel, if anything goes wrong, please send me
mail.

Feel free to provide positive feedback too. ;)

Thanks,
-Costa
```

With the PCI IDE code, your chipset may not be known. If so, you will get a message like:

```
pciide0: bus-master DMA support present, used without full driver support
```

If you get this message, you can try and force DMA mode by using 'flags 0x0001' on your pciide entry in your kernel config file. That would look something like this:

```
pciide* at pci ? dev ? function ? flags 0x0001
```

After doing this, the pciide code will try to use DMA mode regardless of whether or not it actually knows how to do so with your chipset. If this works, and your system makes it through fsck and startup, it is likely that this will work for good. If this does not work, and the system hangs or panics after booting, then you can't use DMA mode (yet, until support is added for your chipset). Note that if you find the documentation for your PCI-IDE controller's chipset, this is a good start to fully supporting your chipset within the PCI-IDE code. You can look on the manufacturer's website or call them. If your PCI-IDE controller is part of your motherboard, figure out who manufactures the chipset and pursue their resources!

Note that you will know that DMA support has been enabled if you see this message:

```
pciide0:0:0: using DMA data transfers
```

This means that pciide0, channel 0, drive 0 is using DMA data transfers.

Some notes:

DMA is not supported on wdc* unless a DMA channel (drq) is specified. I'm not sure what the "standard" drqs are for hard disk controllers. To enable

```
wdc0    at isa? port 0x1f0 irq 14 flags 0x00
```

Non-ultra DMA does not necessarily lead to higher bandwidth vs PIO. However, it decreases the CPU load significantly.

[Back to Main Index] [To Section 11.0 - Performance Tuning] [To Section 13.0 - IPsec]


www@openbsd.org

$OpenBSD: faq12.html,v 1.21 1999/10/01 19:38:09 ericj Exp $

OpenBSD

13.0 - Using IPsec in OpenBSD

Table of Contents

IPSec is partially documented in the vpn(8) man page. There are various files in /usr/share/ipsec/ directory which can also assist you. The manual pages for ipsec(4), ipsecadm(8), photurisd(8), startkey(1), isakmpd(8) and isakmpd.conf(5) are also available!

You should also check Codetalker which has an FAQ for setting up an IPSec VPN with OpenBSD. It explains some basic IPSec concepts and how to apply them to a VPN setup.

Section Under Work

[Back to Main Index] [To Section 12.0 - For Advanced Users] [To Section 14.0 - Using Disks in OpenBSD]

www@openbsd.org

$OpenBSD: faq13.html,v 1.5 1999/09/24 01:46:14 ericj Exp $

OpenBSD

14.0 - Working with disks in OpenBSD

Table of Contents

14.1 - Using the OpenBSD disklabel

Be sure to check the disklabel(8) man page!

Disklabel is used to identify disks and/or partitions. The main idea behind them is abstraction of disk hardware, since there exists several partitioning tables. A **disklabel** is a universal partitioning table. These labels tell the system everything about the disk, but most of it is irrelevant. The important parts of the label are both the part that tells the disk geometry and also the part that actually defines what partitions exist on your disk. This label resides on the beginning of the disk, this is to help bootstrap code use them (since most ROM based booting reads the first sectors of the first disk). Disklabel can, on some systems, be used to install bootstrap code.

As an additional gain, using disklabel helps overcome architecture limitations on disk partitioning. For example, on i386, you can only have 4 primary partitions. (Partitions that other operating systems, such as Windows NT or DOS can see.) With **disklabel**, you use one of these 'primary' partitions to store *all* of your OpenBSD partitions (eg. 'swap', '/', '/usr' and '/var'). And you still have 3 more partitions available for other OSs!

You can think of them as 'subpartitions' for installation purposes. But they are a necessity for kernel disk access based on geometry. And remember to use 'b' for the swap label, and 'a' for root filesystem (default names in kernel). (Note also that in other FAQ documents we usually refer to these OpenBSD 'subpartitions' as partitions. Most architectures do not have two layers of partitions like the i386 does.)

To just view the current disklabel for your disk just do:

```
# disklabel <disk-device name>
```

This will give output similar to this:

```
# using MBR partition 3: type A6 off 64 (0x40) size 16777152 (0xffffc0)
# /dev/rwd0c:
type: ESDI
disk:
label: TOSHIBA MK2720FC
flags:
```

```
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 2633
total sectors: 2654064
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # milliseconds
track-to-track seek: 0  # milliseconds
drivedata: 0

16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  a:  2071440    65583    4.2BSD     1024  8192    16   # (Cyl.   65*- 2120)
  b:    65520       63      swap                        # (Cyl.    0*- 65)
  c:  2654064        0    unused        0     0         # (Cyl.    0 - 2632)
  j:   512001  2137023    4.2BSD     1024  8192    16   # (Cyl. 2120*- 2627*)
```

In this example, there are 4 labels - A, B, C, J. A is one main partition, B is swap, and J is another partition. C in all cases represents the full disk. This is helpful in showing what values in size, cyl, etc you have to work with. But with this command you cannot change any values, you can just view.

To be able to edit your label's interactively you would use the **-E** flag.
Example:

```
# disklabel -E wd0
# using MBR partition 3: type A6 off 64 (0x40) size 16777152 (0xffffc0)

Treating sectors 64-16777216 as the OpenBSD portion of the disk.
You can use the 'b' command to change this.

Initial label editor (enter '?' for help at any prompt)
> ?
Available commands:
        p [unit]  - print label.
        M         - show entire OpenBSD man page for disklabel.
        e         - edit drive parameters.
        a [part]  - add new partition.
        b         - set OpenBSD disk boundaries.
        c [part]  - change partition size.
        d [part]  - delete partition.
        m [part]  - modify existing partition.
        r         - recalculate free space.
        u         - undo last change.
        s [path]  - save label to file.
        w         - write label to disk.
        q         - quit and save changes.
        x         - exit without saving changes.
        ? [cmnd]  - this message or command specific help.
Numeric parameters may use suffixes to indicate units:
        'b' for bytes, 'c' for cylinders, 'k' for kilobytes, 'm' for megabytes,
        'g' for gigabytes or no suffix for sectors (usually 512 bytes).
```

```
        Non-sector units will be rounded to the nearest cylinder.
Entering '?' at most prompts will give you (simple) context sensitive help.
> p
device: /dev/rwd0c
type: ESDI
disk:
label: TOSHIBA MK2720FC
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 2633
total sectors: 2654064
free sectors: 14193711
rpm: 3600

16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  a:  2071440    65583    4.2BSD     1024  8192    16   # (Cyl.   65*- 2120)
  b:    65520       63      swap                        # (Cyl.    0*- 65)
  c:  2654064        0    unused        0     0         # (Cyl.    0 - 2632)
  j:   512001  2137023    4.2BSD     1024  8192    16   # (Cyl. 2120*- 2627*)
> q
No changes.
```

In this example we get into disklabel's interactive mode. From here we can issue the command "?" asking for help. This gives us many commands we can use to add, delete, edit existing labels.

There is a new feature on disklabel, to help installation. The '-f' flag writes a 'fstab' like file, if given a mount point for a label. This is only used at installation.

To see an example install involving disklabel check log25.txt

14.2 - Using fdisk

First be sure to check the fdisk main page. fdisk(8)

Fdisk is a program to help with the maintenance of your partitions. This program is used at install time to set up your OpenBSD partition (this partition can contain several labels, each with filesystems/swap/etc.). It can divide space on your drives and set one active. This program will usually be used in Single User Mode (boot -s). Fdisk also sets the MBR on your various hard disks.

For installation purposes, most times you'll only need **ONE** Openbsd partition, and then using disklabel to put a swap and a filesystem on it.

To just view your partition table using fdisk just use:

```
# fdisk fd0
```

Which will give an output similar to this:

```
        Disk: fd0       geometry: 80/2/18 [2880 sectors]
        Offset: 0       Signatures: 0xAA55,0x0
                 Starting        Ending
         #: id  cyl  hd sec -  cyl  hd sec [     start -       size]
```

```
-------------------------------------------------------------------------
        *0: A6    0   0   1 -   79   1  18 [         0 -       2880] OpenBSD
         1: 00    0   0   0 -    0   0   0 [         0 -          0] unused
         2: A7    0   0   2 -   79   1  18 [         1 -       2879] NEXTSTEP
         3: 00    0   0   0 -    0   0   0 [         0 -          0] unused
```

In this example we are viewing the fdisk output of floppy drive. We can see the OpenBSD partition (A6) and its size. The * tells us that the OpenBSD partition is a bootable partition.

In the previous example we just viewed our information. What if we want to edit our partition table? Well, to do so we must use the **-e** flag. This will bring up a command line prompt to interact with fdisk.

```
# fdisk -e wd0
Enter 'help' for information
fdisk: 1> help
        help            Command help list
        manual          Show entire OpenBSD man page for fdisk
        reinit          Re-initialize loaded MBR (to defaults)
        disk            Edit current drive stats
        edit            Edit given table entry
        flag            Flag given table entry as bootable
        update          Update machine code in loaded MBR
        select          Select extended partition table entry MBR
        print           Print loaded MBR partition table
        write           Write loaded MBR to disk
        exit            Exit edit of current MBR, without saving changes
        quit            Quit edit of current MBR, saving current changes
        abort           Abort program without saving current changes
fdisk: 1>
```

It is perfectly safe in fdisk to go in and explore, just make sure to answer **N** to saving the changes and *DON'T* use the **write** command.

Here is an overview of the commands you can use when you choose the **-e** flag.

- **help** Display a list of commands that fdisk understands in the interactive edit mode.
- **reinit** Initialize the currently selected, in-memory copy of the boot block.
- **disk** Display the current drive geometry that fdisk has probed. You are given a chance to edit it if you wish.
- **edit** Edit a given table entry in the memory copy of the current boot block. You may edit either in BIOS geometry mode, or in sector offsets and sizes.
- **flag** Make the given partition table entry bootable. Only one entry can be marked bootable. If you wish to boot from an extended partition, you will need to mark the partition table entry for the extended partition as bootable.
- **update** Update the machine code in the memory copy of the currently selected boot block.
- **select** Select and load into memory the boot block pointed to by the extended partition table entry in the current boot block.
- **print** Print the currently selected in-memory copy of the boot block and its MBR table to the terminal.
- **write** Write the in-memory copy of the boot block to disk. You will be asked to confirm this operation.
- **exit** Exit the current level of fdisk, either returning to the previously selected in-memory copy of a boot block, or exiting the program if there is none.
- **quit** Exit the current level of fdisk, either returning to the previously selected in-memory copy of a boot block, or exiting the program if there is none. Unlike exit it does write the modified block out.
- **abort** Quit program without saving current changes.

14.3 - Adding extra disks in OpenBSD

Well once you get your disk installed **PROPERLY** you need to use fdisk(8) (*i386 only*) and disklabel(8) to set up your disk in OpenBSD.

For i386 folks, start with fdisk. Other architectures can ignore this.

```
# fdisk -i sd2
```

This will initialize the disk's "real" partition table for exclusive use by OpenBSD. Next you need to create a disklabel for it. This will seem confusing.

```
# disklabel -e sd2

(screen goes blank, your $EDITOR comes up)
type: SCSI
...bla...
sectors/track: 63
total sectors: 6185088
...bla...
16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  c:  6185088        0    unused        0     0         # (Cyl.    0 - 6135)
  d:  1405080       63    4.2BSD     1024  8192    16   # (Cyl.    0*- 1393*)
  e:  4779945  1405143    4.2BSD     1024  8192    16   # (Cyl. 1393*- 6135)
```

First, ignore the 'c' partition, it's always there and is for programs like disklabel to function! For normal operations, fsize should always be 1024, bsize should always be 8192, and cpg should always be 16. Fstype is 4.2BSD. Total sectors is the total size of the disk. Say this is a 3 gigabyte disk. Three gigabytes in disk manufacturer terms is 3000 megabytes. So divide 6185088/3000 (use bc(1)). You get 2061. So to make up partition sizes for a, d, e, f, g, ... just multiply X*2061 to get X megabytes of space on that partition. The offset for your first new partition should be the same as the "sectors/track" reported earlier in disklabel's output. For us it is 63. The offset for each partition afterwards should be a combination of the size of each partition and the offset of each partition (Except the C partition, since it has no play into this equation.)

Or, if you just want one partition on the disk, say you will use the whole thing for web storage or a home directory or something, just take the total size of the disk and subtract the sectors per track from it. 6185088-63 = 6185025. Your partition is

```
  d:  6185025       63    4.2BSD     1024  8192    16
```

If all this seems needlessly complex, you can just use disklabel -E to get the same partitioning mode that you got on your install disk! There, you can just use "96M" to specify "96 megabytes". (Or, if you have a disk big enough, 96G for 96 gigs!) Unfortunately, the -E mode uses the BIOS disk geometry, not the real disk geometry, and often times the two are not the same. To get around this limitation, type 'g d' for 'geometry disk'. (Other options are 'g b' for 'geometry bios' and 'g u' for geometry user, or simply, what the label said before disklabel made any changes.)

That was a lot. But you are not finished. Finally, you need to create the filesystem on that disk using newfs(8).

```
bsd# newfs wd1a
```

Or whatever your disk was named as per OpenBSD's disk numbering scheme. (Look at the output from dmesg(1) to see what your disk was named by OpenBSD.)

Now figure out where you are going to mount this new partition you just created. Say you want to put it on /u. First, make the directory /u. Then, mount it.

```
mount /dev/wd1a /u
```

Finally, add it to /etc/fstab

```
/dev/wd1a /u ffs rw 1 1
```

What if you need to migrate an existing directory like /usr/local? You should mount the new drive in /mnt and use cpio -pdum to copy /usr/local to the /mnt directory. Edit the /etc/fstab file to show that the /usr/local partition is now /dev/wd1a (your freshly formatted partition.) Example:

/dev/wd1a /usr/local ffs rw 1 1

Reboot into single user mode..**boot -s** Move the existing /usr/local to /usr/local-backup (or delete it if you feel lucky) and create an empty directory /usr/local. Then reboot the system, and whala!! the files are there!

14.4 - How to swap to a file

(Note: if you are looking to swap to a file because you are getting "virtual memory exhausted" errors, you should try raising the per-process limits first with csh's unlimit(1), or sh's ulimit(1).)

With OpenBSD 2.5 came the introduction of swapctl(8), which made dealing with swap devices much easier. Swapping to a file doesn't require a custom built kernel, although that can still be done, this faq will show you how to add swap space both ways.

Swapping to a file.

Swapping to a file is easiest and quickest way to get extra swap area's setup. This is not for users who are currently using Softupdates. (Which isn't enabled by default). To start out, you can see how much swap you are currently have and how much you are using with the swapctl(8) utility. You can do this by using the command:

```
ericj@oshibana> swapctl -l
Device      512-blocks     Used    Avail Capacity  Priority
swap_device      65520        8    65512     0%    0
```

This shows the devices currently being used for swapping and their current statistics. In the above example there is only one device named "swap_device". This is the predefined area on disk that is used for swapping. (Shows up as partition b when viewing disklabels) As you can also see in the above example, that device isn't getting much use at the moment. But for the purposes of this document, we will act as if an extra 32M is needed.

The first step to setting up a file as a swap device is to create the file. It's best to do this with the dd(1) utility. Here is an example of creating the file */var/swap* that is 32M large.

```
ericj@oshibana> sudo dd if=/dev/zero of=/var/swap bs=1k count=32768
32768+0 records in
32768+0 records out
33554432 bytes transferred in 20 secs (1677721 bytes/sec)
```

Once this has been done, we can turn on swapping to that device. Use the following command to turn on swapping to this device

```
ericj@oshibana> sudo swapctl -a /var/swap
```

Now we need to check to see if it has been correctly added to the list of our swap devices.

```
ericj@oshibana> swapctl -l
Device      512-blocks     Used    Avail Capacity  Priority
swap_device      65520        8    65512     0%    0
/var/swap        65536        0    65536     0%    0
Total           131056        8   131048     0%
```

Now that the file is setup and swapping is being done, you need to add a line to your */etc/fstab* file so that this file is configured on the next boot time also. If this line is not added, your won't have this swap device configured.

```
ericj@oshibana> cat /etc/fstab
/dev/wd0a / ffs rw 1 1
/var/swap /var/swap sw 0 0
```

Swapping via a vnode device

This is a more permenant solution to adding more swap space. To swap to a file permanently, first make a kernel with vnd0c as swap. If you have wd0a as root filesystem, wd0b is the previous swap, use this line in the kernel configuration file (refer to compiling a new kernel if in doubt):

```
config          bsd     root on wd0a swap on wd0b and vnd0c dumps on wd0b
```

After this is done, the file which will be used for swapping needs to be created. You should do this by using the same command as in the above examples.

```
ericj@oshibana> sudo dd if=/dev/zero of=/var/swap bs=1k count=32768
32768+0 records in
32768+0 records out
33554432 bytes transferred in 20 secs (1677721 bytes/sec)
```

Now your file is in place, you need to add the file to you */etc/fstab*. Here is a sample line to boot with this device as swap on boot.

```
ericj@oshibana> cat /etc/fstab
/dev/wd0a / ffs rw 1 1
/dev/vnd0c none swap sw 0 0
```

At this point your computer needs to be rebooted so that the kernel changes can take place. Once this has been done it's time to configure the device as swap. To do this you will use vnconfig(1).

```
ericj@oshibana> sudo vnconfig -c -v vnd0 /var/swap
vnd0: 33554432 bytes on /var/swap
```

Now for the last step, turning on swapping to that device. We will do this just like in the above examples, using swapctl(8). Then we will check to see if it was correctly added to our list of swap devices.

```
ericj@oshibana> sudo swapctl -a /dev/vnd0c
ericj@oshibana> swapctl -l
Device      512-blocks     Used    Avail Capacity  Priority
swap_device      65520        8    65512     0%    0
/dev/vnd0c       65536        0    65536     0%    0
Total           131056        8   131048     0%
```

14.5 - Soft-updates

Softupdates are experimental and are NOT recomended for regular users.

Over the last few years Kirk McKusick has been working on something called "soft updates". This is based on an idea proposed by Greg Ganger and Yale Patt that imposing a partial ordering on the buffer cache opertions would permit the requirement for synchronous writing of directory entries to be removed from the FFS code. Thus, a large performance increase of disk writing performance.

To enable Softupdates your kernel must have option

option FFS_SOFTUPDATES

then you need to boot into single-user mode:

```
boot>boot -s
[snip]
bsd# tunefs -s enable <raw device>
bsd# reboot -n
```

GOOD LUCK. Please direct all problem reports with as MUCH information as possible (kernel coredumps, tracebacks, etc) to Costa. Do not waste his time with any other questions or incomplete problem reports.

14.6 - When I boot after installation of OpenBSD/i386, it says "partition 3 id 0".

This means that your MBR (Master Boot Record) was not installed properly or that your BIOS is not compatible with your current MBR. To solve this first try to reinstall the OpenBSD boot blocks. To do so use installboot(8). For example, the most common usage is:

- Boot from the OpenBSD bootdisk
  - # **fsck /dev/rwd0a && mount /dev/wd0a /mnt**
  - # **cp /usr/mdec/boot /mnt/boot**
  - # **/usr/mdec/installboot -v /mnt/boot /usr/mdec/biosboot wd0**
- Here substitute your disk devices for rwd0a & wd0a. This will install the boot blocks again.
- Then Reboot.

If this doesn't work you still have a few more options. So, luck hasn't run out yet. The first is to use a bootloader such as OS-BS. The OpenBSD cd has the os-bs bootloader included in the tools directory. If you didn't purchase a CD-ROM, you can download os-bs from any of the OpenBSD ftp sites.

pub/OpenBSD/2.5/tools/osbs135.exe

and take some time to look through the web pages at http://www.prz.tu-berlin.de/~wolf/os-bs.html

There are also other commercial bootloaders or lilo that you can use for multi-booting.

Here is a simple outline of how to get lilo onto your system.

- Boot a DOS floppy and do a fdisk /MBR. Make sure you do that on the drive you'll be booting from.
- Boot from a linux disk and install LILO & chain it to your OpenBSD boot block.

For further instructions read INSTALL.linux

14.7 - How do I get a dmesg from a boot floppy?

RAMDISK images (boot floppies) do not ship with the *dmesg* utility. They do, however, have the `/kern` filesystem mounted. To copy the dmesg information to a file, do a:

```
# cat /kern/msgbuf >mydmesg
```

Boot disks also have 'more' to page through the output:

```
# more /kern/msgbuf
```

Also check with section 4.7

14.8 - Installing Bootblocks - i386 specific

Older versions of MS-DOS can only deal with disk geometries of 1024 cylinders or less. Since virtually all modern disks have more than 1024 cylinders, most SCSI BIOS chips (which come on the SCSI controller card) and IDE BIOS (which is part of the rest of the PC BIOS) have an option (sometimes the default) to "translate" the real disk geometry into something that fits within MS-DOS' ability. However, not all BIOS chips "translate" the geometry in the same way. If you change your BIOS (either with a new motherboard or a new SCSI controller), and the new one uses a different "translated" geometry, you will be unable to load the second stage boot loader (and thus unable to load the kernel). (This is because the first stage boot loader contains a list of the blocks that comprise /boot in terms of the original "translated" geometry.) If you are using IDE disks, and you make changes to your BIOS settings, you can (unknowingly) change its translation also (most IDE BIOS offer 3 different translations.) To fix your boot block so that you can boot normally, just put a boot floppy in your hard drive, and at the boot prompt, type "b hd0a:/bsd" to force it to boot from the first hard disk (and not the floppy). Your machine should come up normally. You now need to update the first stage boot loader to see the new geometry (and re-write the boot block accordingly).
Our example will assume your boot disk is sd0 (but for IDE it would be wd0, etc..):

```
# cd /usr/mdec; ./installboot /boot biosboot sd0
```

If installboot complains that it is unable to read the BIOS geometry, at the boot> prompt you may issue the "machine diskinfo" (or "ma di" for short) command to print the information you need. Feed the "heads" and "secs" values to installboot's -h and -s flags, respectively, so that the modified installboot command is the following:

```
# cd /usr/mdec; ./installboot -h <heads> -s <secs> /boot biosboot sd0
```

If a newer version of bootblocks are required, you will need to compile these yourself. To do so simply.

```
# cd /sys/arch/i386/stand/
# make && make install
# cd /usr/mdec; cp ./boot /boot
# ./installboot /boot biosboot sd0 (or whatever device your hard disk is)
```

[Back to Main Index] [To Section 13.0 - IPsec]

---

www@openbsd.org

$OpenBSD: faq14.html,v 1.4 1999/10/08 15:50:52 ericj Exp $

OpenBSD

11-2.4 - OpenBSD 2.4 Specific Information

Table of Contents

11.1 - In 2.4 why do I get UserDir error when running httpd?

It seems as though userdir support was left out of 2.4. Make sure to pick up the patch at:

http://www.openbsd.org/errata.html

11.2 - Installing at a partition 4GB or higher

The 2.4 cd's shipped with a problem in installboot. A patch can be obtained here http://www.openbsd.org/errata.html. This is fixed in -current.

To place the partition boot record, installboot used a 32 bit multiplication for the sectors times the sectorsize which is larger than 2^32 -1 (i.e. multiplication overflow) when the partition starts at 4 GB or higher. This means the pbr is installed in the real partition address modulo 4GB. It's still possible to install at >= 4GB with some effort:

You have 2 Options for doing so with OpenBSD 2.4. If you have Linux or any other unix like OS, you should back up the sectors first before trying anything.

- Suppose the partition starts at cylinder 611, and the drive has 255 heads, 63 sectors: offset for the obsd partition is 611*255*63= 9815715 sectors. multiply this by 512 and you get: 5025646080. 5025646080 mod 2^32 = 730678784

  (You can just subtract 2^32 on a hand calculator...) dividing by 512 gives sector 1427107.

  Ok, suppose were now in linux (where cylinder numbers are 1 higher btw..) and that the obsd partition is DOS partition 4:

  dd if=/dev/hda of=backup.txt skip=1427107 bs=512 count=1

  will backup the data that installboot will nuke ..

Now install OpenBSD. Return to Linux.

Now we're going to copy the partition boot record to the right place.

dd if=/dev/hda of=pbr.txt skip=1427107 bs=512 count=1
dd if=/pbr.txt of=/dev/hda4 bs=512 count=1

(adding obsd to lilo should now work)

Now write back the nuked sector:

dd if=/backup.txt of=/dev/hda seek=1427107 bs=512 count=1

Caveat: make sure that whatever gets nuked by installboot isn't essential to running dd (and starting the other OS). Backing up the affected partition is recommended.

- Use a correct version of installboot (compiled for you by another OBSD'er ..):

  put it on another partition, e.g. a DOS partition. Suppose wd0j is the DOS partition (see disklabel wd0, this is in case of IDE disks), and installboot is named installb there.

  After booting from floppy/CD, press ^C to get into the shell:
  - **mkdir dos**
  - **mount /dev/wd0j dos**
  - **cp dos/tmp/installb /usr/mdec/installboot**

Now enter 'install' and the installation procedure should use the new installboot.

---

www@openbsd.org
$OpenBSD: faq11-2.4.html,v 1.4 1999/09/24 17:50:41 ericj Exp $