Langue: [fr]

[cs] [en] [de] [es]
[id] [ja] [pl] [pt]
[pt]
[zh-cn] [zh-tw]

Autres Documents

Upgrade-MiniFAQ
Ports et Packages
Guide de Tests des Ports
Comment Utiliser AnonCVS
Stable
Comment Utiliser CVSup
Pages de manuel
Faire un Rapport de Bogues
Listes de diffusion
Guide de l'Utilisateur PF
FAQ OpenSSH

Fichiers PDF

FAQ OpenBSD
Guide de l'Utilisateur PF

Fichiers texte

FAQ OpenBSD
FAQ PF

Retour à OpenBSD

OpenBSD

Documentation et Questions Fréquemment Posées

Problèmes Fréquemment Rencontrés Mises à jour récentes

Cette FAQ constitue une documentation visant à augmenter les pages de manuel, pages disponibles aussi bien sur le système installé qu'en ligne. La FAQ couvre la version la plus récente d'OpenBSD, actuellement la v3.4. Il est à noter que la version de développement (-current) d'OpenBSD n'est *pas* couverte par la présente FAQ.

La FAQ au format PDF et texte simple est disponible avec d'autres documents dans le répertoire **`pub/OpenBSD/doc`** des miroirs FTP (en anglais).

Remarque Importante : Les fichiers suivants ne sont pas traduits ou leur traduction ne correspond plus au contenu de la version anglaise actuelle :

- 2 - Autres Ressources d'Information OpenBSD
- 4 - Guide d'Installation
- 6 - Mise en place du réseau
- 7 - Contrôles du clavier et de l'affichage
- 8 - Questions Générales
- 11 - Optimisation des Performances
- 14 - Configuration des Disques

Les liens vers les fichiers ci-dessus ont été modifiés de façon à pointer vers la version anglaise actuelle.

Si vous souhaitez contribuez à l'effort de traduction, prière de consultez la page de traduction.

---

1 - Introduction à OpenBSD

9 - Migrer depuis Linux

10 - Gestion du Système

11 - Optimisation des Performances

12 - Questions Spécifiques Aux Plates-Formes

14 - Configuration des Disques

Guide de l'Utilisateur PF

Problèmes Fréquemment Rencontrés

Mises à jour récentes

Les mainteneurs de la FAQ sont :
Nick Holland, Eric Jackson, Wim Vandeputte et Chris Cappuccio.

Pour toute information concernant la traduction de cette FAQ et le reste du site web OpenBSD, consultez la page traduction.

Les questions et les commentaires concernant cette FAQ peuvent être envoyés à faq@openbsd.org. Les questions d'ordre général sur OpenBSD devront être envoyées à la liste de diffusion appropriée.

Retour à OpenBSD

Originally [OpenBSD: index.html,v 1.207]
$Translation: index.html,v 1.80 2004/04/22 05:10:31 saad Exp $
$OpenBSD: index.html,v 1.53 2004/04/23 06:30:59 jufi Exp $
"Si vous ne le trouvez pas dans l'index, veuillez regarder en détail le catalogue."
Sears, Roebuck, and Co., Consumer's Guide, 1897

[Index de La FAQ] [Section 2 - Autres Ressources d'Information OpenBSD]

1 - Introduction à OpenBSD

Table des matières

1.1 - Qu'est-ce qu'OpenBSD ?

Le projet OpenBSD fournit un système d'exploitation de type UNIX **LIBRE**, multi plates-formes et basé sur 4.4BSD. Nos objectifs concernent principalement l'exactitude, la sécurité, la standardisation et la portabilité. OpenBSD supporte l'émulation des binaires SVR4 (Solaris), FreeBSD, Linux, BSD/OS, SunOS et HP-UX.

Cette FAQ couvre spécifiquement la version la plus récente d'OpenBSD, la version 3.4.

1.2 - Sur quels systèmes OpenBSD fonctionne-t-il ?

OpenBSD 3.4 tourne sur les plates-formes suivantes :

- alpha - FTP uniquement
- hp300 - FTP uniquement
- hppa - FTP uniquement
- i386 - CD bootable
- mac68k - FTP uniquement
- macppc - CD bootable
- mvme68k - FTP uniquement
- mvme88k - Experimental, FTP uniquement
- sparc - CD bootable
- sparc64 - CD bootable

- vax

bootable indique qu'OpenBSD démarrera directement depuis le CD. Le CD démarre sur plusieurs types de plates-formes. Voir le chapitre 3 de cette FAQ pour les détails quant à l'obtention d'OpenBSD sur CD.

Les versions précédentes d'OpenBSD supportaient aussi :

- amiga - supprimé après la version 3.2
- sun3 - supprimé après la version 2.9
- arc - supprimé après la version 2.3
- pmax - supprimé après la version 2.7

A l'heure actuelle, OpenBSD ne supporte pas plus d'un processeur. Consultez FAQ 8, SMP pour plus d'informations.

1.3 - Est-ce qu'OpenBSD est vraiment libre ?

OpenBSD est totalement libre. Les binaires sont libres, le source est libre. Toutes les parties d'OpenBSD ont des copyright raisonnables permettant la libre redistribution. Cela comprend la possibilité de REUTILISER la plupart des sources d'OpenBSD, pour un usage personnel comme pour un usage commercial. OpenBSD ne possède pas de restrictions autres que celles spécifiées dans la licence BSD originelle. Les logiciels qui sont écrits avec une licence restrictive ne peuvent pas être inclus dans la distribution standard d'OpenBSD. Ceci dans le but de sauvegarder l'usage libre d'OpenBSD. Par exemple, OpenBSD peut être utilisé librement pour un usage personnel, pour un usage éducatif, par des institutions gouvernementales, par des associations à but non lucratif et par des organisations commerciales. OpenBSD peut être entièrement ou partiellement incorporé dans des produits commerciaux.

Pour plus d'informations sur les autres licences, veuillez lire : Politique Copyright d'OpenBSD.

Les développeurs du projet OpenBSD supportent celui-ci principalement grâce à leurs revenus. Ceci inclut le temps dépensé en programmation pour le projet, le matériel utilisé, les ressources réseaux utilisées pour que vous puissiez obtenir OpenBSD et le temps dépensé à répondre aux questions et à corriger les bugs trouvés par les utilisateurs. Les développeurs OpenBSD ne sont pas forcément riches et même de petites contributions de temps, de matériel ou autres peuvent faire de grosses différences.

1.4 - Pourquoi est-ce que j'utiliserai OpenBSD ?

Les nouveaux utilisateurs veulent souvent savoir si OpenBSD est supérieur à d'autres UNIX libres. Il est quasiment impossible de répondre à cette question qui est sujette à de nombreux et inutiles débats "religieux". Ne jamais, en aucune circonstance, poser cette question sur une des listes de discussion OpenBSD.

Ci-dessous, vous trouverez les raisons qui nous font penser qu'OpenBSD est un système utile. Maintenant, vous seul pouvez répondre à la question qui est de savoir s'il est utile pour vous.

- OpenBSD tourne sur de nombreuses plates- formes matérielles.
- OpenBSD est considéré par de nombreux professionnels de la sécurité comme étant le système de type UNIX le plus sûr qui soit. Ceci est du au fait que le code source fait régulièrement l'objet d'un audit exhaustif.
- OpenBSD est un système UNIX complet disponible gratuitement avec les sources.
- OpenBSD intègre les derniers outils en matière de sécurité pour construire des pare-feux et des réseaux privés virtuels dans un environnement distribué.
- OpenBSD bénéficie d'un développement fort et continu dans de nombreux domaines, donnant la possibilité de travailler sur des technologies émergentes avec une communauté internationale de programmeurs et d'utilisateurs.

1.5 - Comment puis-je aider OpenBSD ?

http://www.openbsd.org/fr/donations.h- tml ont contribué au projet OpenBSD. Elles sont citées dans la page des dons

OpenBSD a constamment besoin de plusieurs types de support de la part des utilisateurs. Si vous trouvez OpenBSD utile, nous vous invitons à trouver un moyen de contribuer. Si aucuns de ceux proposés ci-dessous ne vous convient, vous pouvez toujours en proposer d'autres en envoyant un courrier éléctronique à : donations@openbsd.org.

- Acheter un jeu de CD OpenBSD. Il comprend la version actuelle d'OpenBSD, et est bootable sur plusieurs plates-formes. Il permet aussi de financer le projet OpenBSD et permet d'éviter la consommation de bande passante lors d'un téléchargement à travers l'Internet. Ce jeu de trois CD peu onéreux incluent la totalité du code source. Rappelez vous que vos amis doivent avoir leurs propres CD !
- Financer le projet. Le projet a toujours besoin d'argent pour payer l'équipement, l'équipement réseau et la publication du CD. Presser des CD demande un investissement aux développeurs OpenBSD qu'ils ne sont pas toujours assurés de récupérer. Envoyer un courrier électronique à donations@openbsd.org pour savoir comment contribuer. Même les petits dons font de grandes différences.
- Donner des équipements. Le projet a toujours besoin de matériel, qu'il soit spécifique ou général. Des articles tels que les disques IDE ou SCSI et les barrettes de mémoire RAM sont toujours les bienvenus. Pour d'autres types de matériels tels que les systèmes complets ou les cartes mères, il est préférable de se renseigner au préalable. Écrivez à donations@openbsd.org pour l'envoi.
- Faites don de votre temps et de vos compétences. Les programmeurs qui aiment écrire des systèmes d'exploitation sont bien entendu les bienvenus mais il existe bien d'autres façons d'être utile. Suivez les listes de discussion et aidez à répondre aux questions des nouveaux utilisateurs.
- Aidez à maintenir la documentation à jour en soumettant de nouvelles rubriques pour la FAQ (à faq@openbsd.org). Créez un groupe d'utilisateurs local et faites découvrir OpenBSD à vos amis. Insistez pour utiliser OpenBSD au travail. Si vous êtes étudiant, parler d'utiliser OpenBSD à vos professeurs en tant qu'outil d'apprentissage pour les cours d'informatique ou de sciences de l'ingénieur. Il est aussi important de mentionner l'une des façons de ne pas "aider" le projet OpenBSD : ne perdez pas votre temps en guerres de religion inter systèmes d'exploitation. Cela n'aide en rien le projet à trouver de nouveaux utilisateurs et nuit aux liens que les développeurs auront liés avec des développeurs d'autres projets.

1.6 - Qui maintient OpenBSD ?

OpenBSD est maintenu par une équipe de développement disséminée à travers de nombreux pays. Le projet est coordonné par Theo de Raadt basé au Canada.

1.7 - La prochaine version d'OpenBSD est prévue pour quand ?

L'équipe de développement OpenBSD crée une nouvelle version tous les six mois, avec des dates de mise à disponibilité fixées au 1er Mai et au 1er Novembre. Vous pourrez obtenir plus d'informations concernant le cycle de développement ici.

1.8 - Quels logiciels sont inclus avec OpenBSD ?

OpenBSD est distribué avec un certain nombre d'applications tierces telles que :

- XFree86 4.3.0 l'environnement X Window. Les serveurs v3.3 sont aussi inclus pour la plate-forme i386 pour le support de chipsets graphiques additionnels. Installé par les ensembles d'installation `x*.tgz`.
- GCC 2.95.3 Compilateur GNU C. L'équipe OpenBSD a ajouté la technologie de protection de la pile Propolice, activée

par défaut et utilisée par toutes les applications intégrées au système d'exploitation ainsi que sur les applications compilées sur OpenBSD. Fait partie de l'ensemble d'installation `comp34.tgz`.
- Perl 5.8.0, avec des correctifs et des améliorations crées par l'équipe OpenBSD.
- Le serveur web Apache 1.3.28. L'équipe OpenBSD a ajouté le confinement chroot par défaut , la révocation de privilèges, et d'autres améliorations sécurité. mod_ssl 2.8 et DSO sont aussi inclus.
- OpenSSL 0.9.7b, avec des correctifs et des améliorations de la part de l'équipe OpenBSD
- Le processeur de texte Groff 1.15.
- Le serveur Sendmail 8.12.9 avec le correctif sécurité parse8.359.2.8.
- Le serveur DNS BIND 9.2.2. OpenBSD a implémenté plusieurs améliorations au processus de confinement chroot ainsi que d'autres amélioration sécurité.
- Le navigateur web en mode texte Lynx 2.8.4rel.1. Sont inclus le support HTTPS et des correctifs crées par l'équipe OpenBSD.
- Sudo v1.6.7p5, permettant aux utilisateurs d'exécuter des commandes individuelles avec les privilèges root.
- Ncurses 5.2.
- IPv6 KAME.
- Heimdal 0.6rc1 avec correctifs
- OpenSSH 3.7.1

Comme on peut le constater, l'équipe OpenBSD ajoute souvent des correctifs aux applications tierces (typiquement) pour améliorer la sécurité ou la qualité du code. Dans certains cas, l'utilisateur ne verra aucune différence dans le fonctionnement. Tandis que dans d'autres cas, il existe des différences opérationnelles qui peuvent impacter certains utilisateurs. Gardez ces améliorations en tête avant d'ajouter des versions différentes du même logiciel de manière aveugle.

Bien entendu, des applications supplémentaires peuvent être ajoutées avec le système des packages et des ports.

1.9 - Quoi de neuf dans OpenBSD 3.4 ?

La liste complète des changements apportés à OpenBSD 3.3 pour créer OpenBSD 3.4 se trouve ici, cependant voici quelques changements que l'équipe de développement OpenBSD juge importants à signaler aux personnes qui vont faire des mises à jour ou des installations d'OpenBSD 3.4 et qui sont familières des versions antérieures :

- **La plate-forme i386 est maintenant en ELF.** Le format de fichier exécutable ELF offre une plus grande flexibilité au niveau de l'agencement mémoire par rapport à l'ancien format a.out. Le format ELF était requis pour notre implémentation W^X pour i386. (W^X est un agencement de permissions mémoire fines, permettant de s'assurer que la mémoire pouvant être écrite par des programmes applicatifs ne peut être exécutable au même moment et vice versa. Ceci met la barre haute pour l'exploitation de débordements tampon et d'autres attaques : l'attaquant ne peut plus écrire du code en mémoire dans des endroits où il peut être exécuté.) La mise à jour en utilisant les sources n'est pas une option. Les mises à jour binaires sont possibles mais très difficiles. Elles nécessitent de désinstaller tous les packages avant la mise à jour et de les réinstaller après mise à jour. Il existe plusieurs problèmes potentiels à ce niveau, l'équipe de développement OpenBSD recommande HAUTEMENT la réinstallation complète de votre système. Il est à noter que l'émulation binaire a.out est fournie par sysctl aux applications binaires (uniquement) qui le nécessitent. Si vous effectuez une mise à jour, vous aurez très certainement besoin d'activer cette option.
- **Les émulations binaires sont désactivées par défaut.** Cette décision a été prise pour rendre plus difficile l'exécution d'un programme malicieux écrit pour une autre plate-forme OpenBSD. Ceci empêchera plusieurs ports de fonctionner correctement jusqu'à ce que l'émulation soit activée si nécessaire en utilisant des sysctls. Le noyau standard GENERIC comporte ces options. Elles sont juste désactivées. Aucune recompilation de noyau n'est nécessaire. Pour plus d'informations, veuillez consulter Cet article. Si vous effectuez une mise à jour, vous aurez sûrement besoin d'activer compat_aout.
- **L'authentification Kerberos IV a été supprimée.** Si vous optez pour une mise à jour plutôt qu'une réinstallation, vous devrez supprimer toute référence à krb4 dans `/etc/login.conf` *avant* d'effectuer la mise à jour, sinon vous pouvez avoir des difficultés pour vous connecter à une machine i386 en mode multi-utilisateur. Si vous avez déjà mis à jour votre système et vous rencontrez ce problème, vous pouvez vous connecter et corriger ce problème en suivant les instructions explicitées ici. La suppression des anciens modules krb4 de `/usr/libexec/auth` est conseillée.

[Index de La FAQ] [Section 2 - Autres Ressources d'Information OpenBSD]


www@openbsd.org

Originally [OpenBSD: faq1.html,v 1.57]
$Translation: faq1.html,v 1.34 2004/04/22 04:42:40 saad Exp $
$OpenBSD: faq1.html,v 1.22 2004/04/23 06:30:59 jufi Exp $

[FAQ Index] [To Section 1 - Introduction to OpenBSD] [To Section 3 - Obtaining OpenBSD]

2 - 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.

The OpenBSD Journal is an OpenBSD-focused news and opinion site.

Many users have set up sites and pages with OpenBSD specific information. As with everything on the Internet, a good search engine is going to make your life easier, as will a healthy dose of skepticism. As always, do not blindly enter commands you do not understand into your computer.

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, so that others can not subscribe you to a flood of unwanted e-mail. The message will include instructions for several different ways to confirm, including a list server web page link, responding to the confirmation message or responding to `majordomo@openbsd.org`. Use whatever method is convenient to you. You will note that all three techniques involve a unique and time limited identifying number, such as `A56D-70D4-52C3`, again to make sure *you* are really the person who requested this mail list subscription (this is *real* "opt-in").

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 help file which can be obtained by sending an e-mail message to majordomo@openbsd.org with a message body of "help".

Your subscription to the OpenBSD mail lists can also be maintained through the web interface at http://lists.openbsd.org

Some of the more popular OpenBSD mailing lists are:

- **announce** - Important announcements. This is a low-volume list.
- **security-announce** - Announcements of security issues. This is a low volume list.
- **misc** - General user questions and answers. This is the most active list, and should be the "default" for most questions.
- **bugs** - Bugs received via sendbug(1) and discussions about them.
- **source-changes** - Automated mailing of CVS source tree changes. Every time a developer commits a change to the OpenBSD source tree, CVS will send out a copy of the (usually brief) commit message via this list.
- **ports** - Discussion of the OpenBSD Ports Tree.
- **ports-changes** - Automated mailing of ports-specific CVS source tree changes.
- **advocacy** - Discussion on advocating OpenBSD, and topics that are just too off-topic for **misc**.

Before posting a question on **misc** or any other mailing list, please check the archives, for most common questions have been asked repeatedly. While it might be the first time you have encountered the problem or question, others on the mailing lists may have seen the same question several times in the last week, and may not appreciate seeing it again. If asking a question possibly related to hardware, *always include a dmesg(8)!*

You can find several archives, other mailing list guidelines and more information on the mailing lists page.

An unofficial mailing list that may be of interest to new users of OpenBSD and Unix is the OpenBSD Newbies list.

2.3 - Manual Pages

OpenBSD comes with extensive documentation in the form of manual pages, as well as longer documents relating to specific applications. Considerable effort is made to make sure the man pages are up-to-date and accurate. In all cases, the man pages are considered the authoritative source of information for OpenBSD.

To access the manual pages and other documentation, be sure that you installed the `man34.tgz` and `misc34.tgz` file sets.

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

Getting Started

- **afterboot(8)** - things to check after the first complete boot.
- **help(1)** - help for new users and administrators.
- **hier(7)** - layout of filesystems.
- **man(1)** - display the on-line manual pages.
- **intro(1)** - introduction to general commands, also see the intros to the other sections of the manual: intro(2), intro(3), intro(4) (note: intro(4) is platform specific), intro(5), intro(6), intro(7), intro(8), and intro(9).
- **adduser(8)** - command for adding new users.
- **vipw(8)** - edit the master password file.
- **disklabel(8)** - read and write disk pack label.
- **reboot, halt(8)** - stop and restart the system.
- **shutdown(8)** - close down the system at a given time.
- **dmesg(8)** - redisplay the kernel boot messages
- **sudo(8)** - don't log in as root, but run commands as root.

For more advanced users

- **boot(8)** - system bootstrapping procedures.
- **login.conf(5)** - format of the login class configuration file.
- **ifconfig(8)** - configure network interface parameters.
- **netstat(1)** - show network status.
- **boot_config(8)** - how to change kernel configuration at boot.
- **release(8)** - build an OpenBSD release.
- **sendbug(1)** - send a problem report (PR) about OpenBSD to a central support site.
- **sysctl(8)** - get or set kernel state.
- **style(9)** - OpenBSD kernel source code style guide.

You can find all the OpenBSD man pages on the web at http://www.openbsd.org/cgi-bin/man.cgi as well as on your computer if you install the `man34.tgz` file set.

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:

```
# 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.

```
# man -k cron
cron (8) - clock daemon
crontab (1) - maintain crontab files for individual users
crontab (5) - tables for driving cron
# man 5 crontab
```

In addition to the UNIX manual pages, there is a typesettable document set (included in the misc34.tgz file set). It lives in the /usr/share/doc directory. 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(1) 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:

```
# cd /usr/share/doc/usd/04.csh
# groff -Tascii -ms tabs csh.1 csh.2 csh.3 csh.4 csh.a csh.g > csh.txt
# 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 through the use of CVS, they will be updated. To view these pages simply :

```
# nroff -mandoc <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 [man_src_file] must be the man page source file (probably a file that ends in a number; e.g., 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. Use the following groff(1) command options for getting a PostScript version from an OpenBSD system man page:

```
# groff -mandoc -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 this:

```
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 expected someone to help find this bug, he or she would have supplied more information... something like this:

```
From: smartuser@example.com
To: bugs@openbsd.org
Subject: 3.3-beta panics on a SparcStation2

OpenBSD 3.2 installed from an official CD-ROM installed and ran fine
on this machine.

After doing a clean install of 3.3-beta from an FTP mirror, I find the
system randomly panics after a period of use, and predictably and
quickly when starting X.

This is the dmesg output:

OpenBSD 3.3-beta (GENERIC) #9: Mon Mar 17 12:37:18 MST 2003
    deraadt@sparc.openbsd.org:/usr/src/sys/arch/sparc/compile/GENERIC
real mem = 67002368
avail mem = 59125760
using 200 buffers containing 3346432 bytes of memory
bootpath: /sbus@1,f8000000/esp@0,800000/sd@1,0
mainbus0 (root): SUNW,Sun 4/75
cpu0 at mainbus0: CY7C601 @ 40 MHz, TMS390C602A FPU; cache chip bug
- trap page uncached
cpu0: 64K byte write-through, 32 bytes/line, hw flush cache enabled
memreg0 at mainbus0 ioaddr 0xf4000000
clock0 at mainbus0 ioaddr 0xf2000000: mk48t02 (eeprom)
timer0 at mainbus0 ioaddr 0xf3000000 delay constant 17
auxreg0 at mainbus0 ioaddr 0xf7400003
zs0 at mainbus0 ioaddr 0xf1000000 pri 12, softpri 6
zstty0 at zs0 channel 0 (console i/o)
zstty1 at zs0 channel 1
zs1 at mainbus0 ioaddr 0xf0000000 pri 12, softpri 6
zskbd0 at zs1 channel 0: reset timeout
zskbd0: no keyboard
zstty2 at zs1 channel 1: mouse
audioamd0 at mainbus0 ioaddr 0xf7201000 pri 13, softpri 4
audio0 at audioamd0
sbus0 at mainbus0 ioaddr 0xf8000000: clock = 20 MHz
dma0 at sbus0 slot 0 offset 0x400000: rev 1+
esp0 at sbus0 slot 0 offset 0x800000 pri 3: ESP100A, 25MHz, SCSI ID 7
scsibus0 at esp0: 8 targets
sd0 at scsibus0 targ 1 lun 0: <SEAGATE, ST1480 SUN0424, 8628> SCSI2 0/direct fixed
sd0: 411MB, 1476 cyl, 9 head, 63 sec, 512 bytes/sec, 843284 sec total
sd1 at scsibus0 targ 3 lun 0: <COMPAQPC, DCAS-32160, S65A> SCSI2 0/direct fixed
sd1: 2006MB, 8188 cyl, 3 head, 167 sec, 512 bytes/sec, 4110000 sec total
le0 at sbus0 slot 0 offset 0xc00000 pri 5: address 08:00:20:13:10:b9
le0: 16 receive buffers, 4 transmit buffers
cgsix0 at sbus0 slot 1 offset 0x0: SUNW,501-2325, 1152x900, rev 11
wsdisplay0 at cgsix0
wsdisplay0: screen 0 added (std, sun emulation)
fdc0 at mainbus0 ioaddr 0xf7200000 pri 11, softpri 4: chip 82072
fd0 at fdc0 drive 0: 1.44MB 80 cyl, 2 head, 18 sec
root on sd0a
```

```
rootdev=0x700 rrootdev=0x1100 rawdev=0x1102


This is the panic I got when attempting to start X:

panic: pool_get(mclpl): free list modified: magic=78746572; page 0xfaa93000;
 item addr 0xfaa93000
Stopped at      Debugger+0x4:   jmpl            [%o7 + 0x8], %g0
RUN AT LEAST 'trace' AND 'ps' AND INCLUDE OUTPUT WHEN REPORTING THIS PANIC!
DO NOT EVEN BOTHER REPORTING THIS WITHOUT INCLUDING THAT INFORMATION!
ddb> trace
pool_get(0xfaa93000, 0x22, 0x0, 0x1000, 0x102, 0x0) at pool_get+0x2c0
sosend(0x16, 0xf828d800, 0x0, 0xf83b0900, 0x0, 0x0) at sosend+0x608
soo_write(0xfac0bf50, 0xfac0bf70, 0xfac9be28, 0xfab93190, 0xf8078f24, 0x0)
at soo_write+0x18
dofilewritev(0x0, 0xc, 0xfac0bf50, 0xf7fff198, 0x1, 0xfac0bf70) at
dofilewritev+0x12c
sys_writev(0xfac87508, 0xfac9bf28, 0xfac9bf20, 0xf80765c8, 0x1000, 0xfac0bf70)
at sys_writev+0x50
syscall(0x79, 0xfac9bfb0, 0x0, 0x154, 0xfcffffff, 0xf829dea0) at syscall+0x220
slowtrap(0xc, 0xf7fff198, 0x1, 0x154, 0x1, 0xfac87508) at slowtrap+0x1d8
ddb> ps
   PID   PPID   PGRP    UID  S       FLAGS  WAIT       COMMAND
 27765   8819  29550      0  3         0x86  netio      xconsole
  1668  29550  29550      0  3      0x4086  poll       fvwm
 15447  29550  29550      0  3     0x44186  poll       xterm
  8819  29550  29550     35  3      0x4186  poll       xconsole
  1238  29550  29550      0  3      0x4086  poll       xclock
 29550  25616  29550      0  3      0x4086  pause      sh
  1024  25523  25523      0  3     0x40184  netio      XFree86
*25523  25616  25523     35  2     0x44104             XFree86
 25616  30876  30876      0  3      0x4086  wait       xinit
 30876  16977  30876      0  3      0x4086  pause      sh
 16977      1  16977      0  3      0x4086  ttyin      csh
  5360      1   5360      0  3        0x84  select     cron
 14701      1  14701      0  3     0x40184  select     sendmail
 12617      1  12617      0  3        0x84  select     sshd
 27515      1  27515      0  3       0x184  select     inetd
  1904      1   1904      0  2        0x84             syslogd
  9125      1   9125      0  3        0x84  poll       dhclient
     7      0      0      0  3    0x100204  crypto_wa  crypto
     6      0      0      0  3    0x100204  aiodoned   aiodoned
     5      0      0      0  3    0x100204  syncer     update
     4      0      0      0  3    0x100204  cleaner    cleaner
     3      0      0      0  3    0x100204  reaper     reaper
     2      0      0      0  3    0x100204  pgdaemon   pagedaemon
     1      0      1      0  3      0x4084  wait       init
     0     -1      0      0  3     0x80204  scheduler  swapper

Thank you!
```

See report.html for more information on creating and submitting bug reports. Detailed information about your hardware is necessary if you think the bug *could be in any way* related to your hardware or hardware configuration. Usually, dmesg(8) output is sufficient in this respect. A detailed description of your problem is necessary. You will note that the dmesg described the hardware, the text explained why Smart User thought the system was not broken, (ran 3.2 properly), how this crash was caused (starting X), and the output of the debugger's "`ps`" and "`trace`" commands. In this case, Smart User provided output captured on a serial console; if you can not do that, you will have to use paper and pencil to record the crash. (This was a real problem, and the information in the above report helped lead to a repair of this issue which impacted Sun4c systems.)

If Smart 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 send electronic mail successfully on the Internet.

After submitting a bug report via sendbug(1), you will be notified by e-mail about the status of the report. You may be contacted by developers for additional information or with patches that need testing. You can also monitor the archives of the `bugs@openbsd.org` mailing list, details on the mailing list page, or query the bug report database status at the on-line Bug Tracking System.

[FAQ Index] [To Section 1 - Introduction to OpenBSD] [To Section 3 - Obtaining OpenBSD]


www@openbsd.org

$OpenBSD: faq2.html,v 1.74 2004/04/21 02:11:29 nick Exp $

[Index de La FAQ] [Section 2 - Autres Ressources d'Information OpenBSD] [Section 4 - Guide d'Installation]

3 - Obtenir OpenBSD

Table des matières

3.1 - Acheter un CD OpenBSD

Acheter un CD OpenBSD est sans doute la meilleure manière de commencer. Veuillez consulter la page des commandes pour en obtenir une copie : http://www.openbsd.org/orders.html.

Il y a plusieurs bonnes raisons d'acheter un CD OpenBSD :

- La vente des CDs finance le développement d'OpenBSD. continu
- Le développement d'un système d'exploitation multi plates-formes nécessite beaucoup d'investissements en matériels.
- Votre contribution sous la forme de l'achat d'un CD à un véritable impact sur le développement futur.
- Le CD contient les binaires (et les sources) pour toutes les plates-formes supportées.
- Le CD est "bootable" sur plusieurs plates-formes et peut être utilisé pour démarrer une machine sans système pré-installé.
- Le CD est utile pour démarrer un système même si vous choisissez d'installer un "snapshot".
- Installer depuis le CD est plus rapide! De plus cela permet de préserver la bande passante.
- Les CDs OpenBSD sont toujours fournis avec de jolis autocollants. Votre système n'est pas vraiment complet sans eux. Vous ne pouvez les obtenir que si vous achetez le CD ou donner du matériel.

Si vous installer un version stable d'OpenBSD, vous devriez utiliser le CD.

3.2 - Acheter des T-shirts OpenBSD

OpenBSD propose des T-shirts pour votre plaisir vestimentaire. Vous pouvez les visualiser à : http://www.OpenBSD.org/tshirts.html.

3.3 - Est-ce qu'OpenBSD fournit une image ISO disponible en

téléchargement ?

Certains systèmes d'exploitation libres sont distribués sous forme d'images ISO. Ce *n'*est pas la méthode de distribution d'OpenBSD.

Le projet OpenBSD ne fournit pas en téléchargement les images ISO utilisées pour créer les les masters des CDs officiels. La raison est simplement que nous souhaiterions que vous achetiez les CDs, pour aider au financement des développements futurs dans OpenBSD. La disposition du CD-ROM officiel est copyright Theo de Raadt. Theo n'autorise pas les gens à redistribuer des images des CDs OpenBSD officiels. Comme incitation pour que les gens achètent les CDs, plusieurs extras sont aussi disponibles dans le paquetage (Dessins, Autocollants, etc).

Notez que seule la disposition du CD est sous copyright, OpenBSD lui-même est libre. Rien n'empêche quelqu'un d'autre de récupérer OpenBSD et de créer son propre CD. Si pour une raison quelconque vous souhaitez télécharger une image de CD, vous pouvez rechercher dans les archives des listes de diffusion pour des pistes possibles. Bien sur, toute image ISO d'OpenBSD disponible sur l'Internet est soit en violation du copyright de Theo de Raadt ou n'est pas une image officielle. Les sources pour les images non officielles peuvent être de confiance ou non; c'est à vous de le déterminer.

Nous suggérons que les personnes souhaitant télécharger OpenBSD gratuitement utilise l'option d'installation par FTP. Pour ceux qui ont besoin d'un CD bootable pour leur système, des images ISO des disquettes de démarrage (appelées `cd34.iso`) sont disponibles pour un certain nombre de plates-formes et qui permettront d'installer le système par FTP. Ces images ISO ont une taille de quelques megaoctets seulement, et contiennent uniquement les outils d'installation. Elles ne contiennent pas les jeux de fichiers constituant le système.

3.4 - Télécharger par FTP ou AFS

Il y a plusieurs miroirs internationaux offrant un accès FTP aux snapshots et versions stables d'OpenBSD. L'accès par AFS est aussi disponible. Il est préférable de toujours utiliser le site le plus proche de vous. Avant de commencer à télécharger cous pouvez utiliser ping (8) et traceroute(8) pour déterminer quel est le site miroir le plus proche et voir si celui-ci fonctionne correctement. Bien sur votre CD d'OpenBSD est toujours plus proche qu'un site miroir. Les informations sont contenues là :

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

3.4 - Obtenir le code source actuel

Le code source d'OpenBSD est disponible librement et gratuitement. La meilleure façon d'obtenir le code source est d'installer les sources qui se trouvent sur le CD et de configurer AnonCVS pour les mettre à jour régulièrement. Les informations à propos d'AnonCVS ainsi que la façon de le configurer se trouvent là :

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

Vous pouvez aussi consulter FAQ 8, CVS

Si vous n'avez pas assez de bande passante pour utiliser AnonCVS, ou si votre accès à l'Internet se fait par UUCP, vous pouvez utiliser CTM au lieu d'AnonCVS pour mettre vos sources à jour. Si c'est votre cas, il est vraiment important de commencer à l'aide d'un CD récent. Les informations au sujet de CTM, incluant sa configuration se trouvent là :

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

Une autre possibilité est d'obtenir le code source par l'intermédiaire du web. Vous pouvez l'obtenir par l'intermédiaire de cvsweb :

Http://www.openbsd.or g/cgi-bin/cvsweb/.

[Index de La FAQ] [Section 2 - Autres Ressources d'Information OpenBSD] [Section 4 - Guide d'installation OpenBSD]


www@openbsd.org

Originally [OpenBSD: faq3.html,v 1.42]
$Translation: faq3.html,v 1.19 2004/01/02 22:19:03 saad Exp $
$OpenBSD: faq3.html,v 1.14 2004/01/03 19:44:35 jufi Exp $

[FAQ Index] [To Section 3 - Obtaining OpenBSD] [To Section 5 - Building the System from Source]

4 - OpenBSD 3.4 Installation Guide

Table of Contents

4.1 - Overview of the OpenBSD installation procedure

OpenBSD has a robust and adaptable text-based installation procedure, and can be installed from a single floppy disk. Most platforms follow a similar installation procedure; however there are some differences in the details. In all cases, you are urged to read the platform-specific INSTALL document in the *platform* directory on the CD-ROM or FTP sites (for example, `i386/INSTALL.i386`, `mac68k/INSTALL.mac68k` or `sparc/INSTALL.sparc`).

On most platforms, the OpenBSD installation uses a special kernel with a number of utilities and install scripts embedded in a preloaded RAM disk. After this kernel is booted, the operating system is extracted from a number of compressed tar(1) (`.tgz`) files. There are several ways to boot this install kernel:

- **Floppy disk:** Floppy disk images are provided which can be used to create an install floppy on another Unix-like system, or on a DOS/Windows system. Typical file names are `floppy34.fs`, though several platforms have multiple floppy images available.
- **CD-ROM:** On several platforms a CD-ROM image (`cd34.iso`) is provided allowing creation of a bootable CD-ROM. This just contains the installation kernel - install files must still be retrieved via FTP or other source. You can, of course, build your own CD-ROM with whatever files and

tools you desire.
- **bsd.rd:** The RAM disk kernel, intended for booting off either an already existing OpenBSD partition or booting over the network.
- **Network:** Some platforms support booting over a network.
- **Writing a file system image to disk:** a filesystem image that can be written to an existing partition, and then can be booted.
- **Bootable Tape:** Some platforms support booting from tape. These tapes can be made following the INSTALL.*platform* instructions.

Not every platform supports all boot options:

- **alpha:** Floppy, CD-ROM, writing a floppy image to hard disk.
- **hp300:** CD-ROM, network.
- **hppa:** Network.
- **i386:** Floppy, CD.
- **mac68k:** Installed (and booted) using utilities running on Mac OS. See INSTALL.mac68k for details.
- **macppc:** CD-ROM, network.
- **mvme68k:** Network, bootable tape.
- **sparc:** Floppy, CD-ROM, network, writing image to existing swap partition, bootable tape.
- **sparc64:** Floppy (U1/U2 only), CD-ROM, network, writing image to existing partition.
- **vax:** Floppy, network.

All platforms other than mac68k can also use a bsd.rd to reinstall or upgrade.

Once the install kernel is booted, you have several options of where to get the install file sets. Again, not every platform supports every option.

- **CD-ROM:** Of course, we prefer you use the Official CD-ROM set, but for special needs, you can also make your own.
- **FTP:** Either one of the OpenBSD FTP mirror sites or your own local FTP server holding the file sets.
- **HTTP:** Either one of the OpenBSD HTTP mirror sites or your own local web server holding the file sets.
- **Local disk partition:** In many cases, you can install file sets from another partition on a local hard disk. For example, on i386, you can install from a FAT partition or a CD-ROM formatted in ISO9660, Rock Ridge or Joliet format. In some cases, you will have to manually mount the file system before using it.
- **NFS:** Some platforms support using NFS mounts for the file sets.
- **Tape:** File sets can also be read from a supported tape.

4.2 - Pre-installation checklist

Before you start your install, you should have some idea what you want to end up with. You will want to know the following items, at least:

- Machine name
- Hardware installed and available
  - Verify compatibility with your platform's hardware compatibility page
  - If ISA, you also need to know hardware settings, and confirm they are as OpenBSD requires.
- Install method to be used (CD-ROM, FTP, etc.)
- How will the system be updated and patched?
  - If done locally, you will need to have sufficient space available for the source tree and building it.
  - Otherwise, you will need access to another machine to build a patched release on.
- Desired disk layout
  - Does existing data need to be saved elsewhere?
  - Will OpenBSD co-exist on this system with another OS? If so, how both will be booted? Will you need to install a "boot manager"?
  - Will the entire disk be used for OpenBSD, or do you want to keep an existing partition/OS (or space for a future one)?
  - How do you wish to sub-partition the OpenBSD part of your disk?
- Network settings, if not using DHCP:
  - Domain name
  - Domain Name Server(s) (DNS) address
  - IP addresses and subnet masks for each NIC
  - Gateway address
- Will you be running the X Window System?

4.3 - Creating bootable OpenBSD install media

As examples, we will look at the installation images available for the i386 and sparc platforms.

The i386 platform has five separate installation disk images to choose from:

- **floppy34.fs** (Desktop PC) supports many PCI and ISA NICs, IDE and simple SCSI adapters and some PCMCIA support. *Most* users will use this image.
- **floppyB34.fs** (Servers) supports many RAID controllers, and some of the less common SCSI adapters. However, support for many standard SCSI adapters and many EISA and ISA NICS has been removed.
- **floppyC34.fs** (Laptops) supports the CardBus and PCMCIA devices found in many laptops.
- **cdrom34.fs** is, in effect a combination of all three boot disks. It can be used to make a bootable 2.88M floppy, or more commonly, as a boot image for a custom recordable CD.
- **cd34.iso** is an ISO9660 image that can be used to create a bootable CD with most popular CD-ROM creation software on most platforms. This is `cdrom34.fs` in a "ready-to-record" format.

Yes, there may be situations where one install disk is required to support your SCSI adapter and another disk is required to support your network adapter. Fortunately, this is a rare event, and can usually be worked around.

The sparc platform has three separate installation disk images to choose from:

- **floppy34.fs:** Supports systems with a floppy disk.
- **cd34.iso** An ISO image usable to make your own CD for booting SPARC systems with a CD-ROM.
- **miniroot34.fs** Can be written to a swap partition and booted.

4.3.1 - Creating floppies on Unix

To create a formatted floppy, use the fdformat(1) command to both format and check for bad sectors.

```
# fdformat /dev/rfd0c
Format 1440K floppy `/dev/rfd0c'? (y/n): y
Processing VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV done.
```

If your output is like the above example, then the disk is OK. However, if you do not see ALL "Vr"'s then the disk is most likely bad, and you should try a new one.

Note that some Unix-like systems have different commands for formatting floppies. Refer to your system's documentation for the exact procedure.

Once you have a clean, formatted floppy it is time to write the installation image to floppy. For this, you can use the dd(1) utility. An example usage of `dd(1)` is below:

```
# dd if=floppy34.fs of=/dev/rfd0c bs=32k
```

Once the image is written, check to make sure that the copied image is the same as the original with the cmp(1) command. If the diskette is identical to the image, you will just see another prompt.

```
# cmp /dev/rfd0c floppy34.fs
```

4.3.2 - Creating floppies on Windows or DOS

This section describes how to write the installation images to floppy disk under Windows or DOS. You can get the tools mentioned below from the tools directory on any of the ftp mirrors, or from the `3.4/tools` directory on CD1 of the OpenBSD CD set.

To prepare a floppy in MS-DOS or Windows, first use the native formatting tools to format the disk.

To write the installation image to the prepared floppy you can use *rawrite*, *fdimage*, or *ntrw*. *rawrite* will not work on Windows NT, 2000 or XP.

Note that `FDIMAGE.EXE` and `RAWRITE.EXE` are both MS-DOS applications, and thus are limited to MS-DOS's "8.3" file naming convention. As `floppyB34.fs` and `floppyC34.fs` have longer file names, you will have to find out how your system stored the file in "8.3 format" before using `FDIMAGE.EXE` or `RAWRITE.EXE` to make your boot floppies.

Example usage of *rawrite*:

```
C:\> rawrite
RaWrite 1.2 - Write disk file to raw floppy diskette

Enter source file name: floppy34.fs
Enter destination drive: a
Please insert a formatted diskette into drive A: and press -ENTER- : Enter
```

Example usage of *fdimage*:

```
C:\> fdimage -q floppy34.fs a:
```

Example usage of *ntrw*:

```
C:\> ntrw floppy34.fs a:
3.5", 1.44MB, 512 bytes/sector
bufsize is 9216
1474560 bytes written
```

4.3.3 - Making a CD-ROM

You can create a CD-ROM using either the `cd34.iso` file or, in the case of the i386 platform, you can also use the `cdrom34.fs` as the bootable floppy image that is used to boot an i386 system from CD-ROM. The exact details here are left to the reader to determine with the tools they have at their disposal.

Some of the tools in OpenBSD are:

- mkhybrid(8)
- cdrecord, part of the cdrtools collection in the OpenBSD Ports System.

4.4 - Booting OpenBSD install media

Booting i386

Booting an install image on the i386 PC platform is nothing new to most people. If you are using a floppy disk, simply insert the floppy into the floppy drive and boot the system. The install image will then load, provided floppy boot is enabled in your BIOS. If you want to boot from CD, you must go into your system's BIOS and set the boot options to allow booting from CD. Some older BIOSes do not have this option, and you must use a floppy for booting your installation image. Don't worry though; even if you boot from floppy you can still install from the CD.

Booting sparc/sparc64

NOTE: On the sparc64 platform, only the SBus machines (Ultra 1, Ultra 2) are bootable from floppy.

To boot from floppy, place the floppy disk with the OpenBSD installation image on it into the floppy drive. Then use the following command to boot from the floppy:

```
ok boot floppy
```

To boot from CD-ROM, place the OpenBSD CD-ROM disk into the drive. If your Sun only has one CD-ROM drive, then just go to the boot prompt, where you can '`boot cdrom`':

```
ok boot cdrom
```

Of course, this will only work in new command mode. If you are at the old command mode prompt (a right arrow), type 'n' for the new command mode. (If you are using an old sparc that is pre-sun4c, you probably don't have a new command mode. In this case, you need to experiment.) If you have multiple CD-ROM devices, you need to boot from the correct one. Try `probe-scsi` from the new command mode.

```
ok probe-scsi

Target 0
  Unit 0   Disk     QUANTUM LIGHTNING 365S
Target 1
  Unit 0   Removable Disk     QUANTUM EMPIRE_1080S
Target 3
  Unit 0   Removable Disk     Joe's CD-ROM
```

Figure out which disk is the CD-ROM you want to boot from. Note the target number.

```
ok boot /sbus/esp/sd@X,0
```

4.5 - Performing an install

4.5.1 - Starting the install

Whatever your means of booting is, it is now time to use it. During the boot process, the kernel and all of the programs used to install OpenBSD are loaded into memory. The most common problem when booting is a bad floppy disk or a drive alignment problem. The boot floppy is quite tightly packed -- any bad spot will cause problems.

At almost any point during the OpenBSD install process, you can terminate the current install attempt by hitting CTRL-C and can restart it without rebooting by running `install` at the shell prompt.

When your boot is successful, you will see a lot of text messages scroll by. This text, on many architectures in white on blue, is the dmesg, the kernel telling you what devices have been found, and where. Don't worry about remembering this text, as a copy is saved as `/var/run/dmesg.boot`. On some architectures, SHIFT+PGUP will let you examine text that has scrolled off the screen.

Then, you will see the following:

```
rootdev=0x1100 rrootdev=0x2f00 rawdev=0x2f02
erase ^?, werase ^W, kill ^U, intr ^C, status ^T
(I)nstall, (U)pgrade or (S)hell? i
```

And with that, we reach our first question. Most of the time, you have the three options shown:

- **Install:** load OpenBSD onto the system, overwriting whatever may have been there. Note that it is possible to leave some partitions untouched in this process, such as a `/home`, but otherwise, assume everything else is overwritten.
- **Upgrade:** Install a new set of install files on this machine, but do not overwrite any configuration information, user data, or additional programs. No disk formatting is done, nor are the /etc or /var directories overwritten. A few important notes:
  - You will not be given the option of installing the `etc34.tgz` file. After the install, you will have to manually merge the changes of `etc34.tgz` into your system before you can expect it to be fully functional. This is an important step which must be done, as otherwise certain key services (such as pf(4)) may not start.
  - The Upgrade process is not designed to skip releases! While this will often work, it is not supported. For OpenBSD 3.4, upgrading 3.3 to 3.4 is the only supported upgrade. If you have to upgrade from an older version, a complete reinstall is recommended.
  - The i386 platform has switched to the ELF binary format, and will require uninstalling all installed packages and ports before upgrade. Reinstallation is *highly* recommended over upgrade.
- **Shell:** Sometimes, you need to perform repairs or maintenance to a system which will not (or should not) boot to a normal kernel. This option will allow you to do maintenance to the system.

On occasion, you will not see the "Upgrade" option listed. After a *flag day* event, it is not possible to directly upgrade; one must rebuild the system from scratch.

In this example, we will do an install, but the upgrade process is similar.

```
Welcome to the OpenBSD/i386 3.4 install program.

This program will help you install OpenBSD in a simple and rational way. At
any prompt except password prompts you can run a shell command by typing
'!foo', or escape to a shell by typing '!'. Default answers are shown in []'s
and are selected by pressing RETURN. At any time you can exit this program by
pressing Control-C and then RETURN, but quitting during an install can leave
your system in an inconsistent state.

Specify terminal type: [vt220]
Do you wish to select a keyboard encoding table? [n] ENTER
```

In most cases, the default terminal type is appropriate; however if you are using a serial console for install, don't just take the default, respond appropriately.

If you do not select a keyboard encoding table, a US keyboard layout will be assumed.

```
IS YOUR DATA BACKED UP? As with anything that modifies disk contents, this
program can cause SIGNIFICANT data loss.

It is often helpful to have the installation notes handy. For complex disk
configurations, relevant disk hardware manuals and a calculator are useful.

Proceed with install? [n] y
```

If you take the default here, the install process will terminate and drop you to a shell prompt.

4.5.2 - Setting up disks

Setting up disks in OpenBSD varies a bit between platforms. For i386 and macppc, disk setup is done in two stages. First, the OpenBSD slice of the hard disk is defined using fdisk(8), then that slice is subdivided into OpenBSD partitions using disklabel(8).

Some users may be a little confused by the terminology used here. It will appear we are using the word "partition" in two different ways. This observation is correct. There are two layers of partitioning in several OpenBSD platforms, the first, one could consider the Operating System partitioning, which is how multiple OSs on one computer mark out their own space on the disk, and the second one is how the OpenBSD partition is sub-partitioned into individual filesystems. The first layer is visible as a disk partition to DOS, Windows, and any other OS that can coexist with other Operating Systems on the IBM AT descended machines. The second layer of partitioning is visible only to OpenBSD and those OSs which can directly read an OpenBSD filesystem.

```
Cool!  Let's get to it...

You will now initialize the disk(s) that OpenBSD will use. To enable all
available security features you should configure the disk(s) to allow the
creation of separate filesystems for /, /tmp, /var, /usr, and /home.

Available disks are: wd0.
Which one is the root disk? (or done) [wd0] Enter
```

The root disk is the disk the system will boot from, and normally where swap space resides. Usually, this will be the default -- if it isn't, you will need to know how to force your computer to boot from a non-standard disk. IDE disks will show up as `wd0`, `wd1`, etc., SCSI disks and RAID devices will show up as `sd0`, `sd1`, and so on. All the disks OpenBSD can find are listed here -- if you have drives which are not showing up, you have unsupported or improperly configured hardware.

```
Do you want to use *all* of wd0 for OpenBSD? [no] Enter
```

If you say "yes" to this question, the entire disk will be allocated to OpenBSD. This will result in a standard Master Boot Record and partition table being written out to disk -- one partition, the size of the entire hard disk, set to the OpenBSD partition type, and flagged as the bootable partition. This will be a common choice for most production uses of OpenBSD; however, there are some systems this should not be done on. Many Compaq systems, many laptops, some Dell and other systems use a "maintenance" or "Suspend to Disk" partition, which should be kept intact. If your system has any other partitions of any type you do not wish to erase, do not select "yes" to the above question.

For the sake of this example, we will assume the disk is to be split between OpenBSD and a pre-existing Windows 2000 partition, so we take the default of "no", which will take us into the fdisk(8) program. You can also get more information on fdisk(8) here.

Important Note: Users with a large hard disk (larger than 8G on a newer i386, though on older machines and different platforms, often much smaller) will want to see this section before going any further.

```
You will now create a single MBR partition to contain your OpenBSD data. This
partition must have an id of 'A6'; must *NOT* overlap other partitions; and
must be marked as the only active partition.

The 'manual' command describes all the fdisk commands in detail.

Disk: wd0       geometry: 2586/240/63 [39100320 Sectors]
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
*0: 06    0   1  1 -  202 239 63 [          63:     3069297 ] DOS > 32MB
 1: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
 2: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
 3: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
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)
        setpid          Set the identifier of a given table entry
        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>
```

A few commands are worthy of elaboration:

- **r** or **reinit:** Clears existing partition table, makes one big OpenBSD partition, flags it active, and installs the OpenBSD MBR code. Equivalent to saying "yes" to the "`use *all* of ...`" question.
- **u** or **update:** Replaces the current MBR code (if any) with the OpenBSD MBR loader. If your drive has never been used, this may be important.
- **p** or **print:** Displays the current partition table in sectors. "`p m`" will show the partition table in megabytes, "`p g`" will show it in gigabytes.
- **e** or **edit:** edit or alter a table entry.
- **f** or **flag:** Marks a partition as the active partition, the one that will be booted from
- **exit** and **quit:** Careful on these, as some users are used to "exit" and "quit" having opposite meanings.

It is worth pointing out once again, a error here will result in significant data loss. If you are going to do this on a drive with important data, it might be worth practicing on a "disposable" drive, in addition to having a good backup.

Our drive here has a 1.5G partition for Windows 2000 (using the FAT filesystem). Looking at the info from the above display, we can see that the Windows partition occupies through cylinder 202 on the drive. So, we are going to allocate the rest of the disk to OpenBSD, starting at cylinder 203. You could also calculate OpenBSD's starting sector of 3069360 by adding the existing partition's starting sector (63) and its size (3069297).

You can edit the drive layout in either Cylinder/Heads/Sectors form or just raw sectors. Which is easier depends upon what you are doing; in this case, working around an existing partition, using CHS format will probably be easier. If you are creating the first partition on the disk, just using raw sectors may be easier.

```
fdisk: 1> e 1
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
 1: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
Partition id ('0' to disable)  [0 - FF]: [0] (? for help) a6
Do you wish to edit in CHS mode? [n] y
BIOS Starting cylinder [0 - 2585]: [0] 203
BIOS Starting head [0 - 239]: [0] Enter
BIOS Starting sector [1 - 63]: [0] 1
BIOS Ending cylinder [0 - 2585]: [0] 2585
BIOS Ending head [0 - 239]: [0] 239
BIOS Ending sector [1 - 63]: [0] 63
fdisk:*1> p
Disk: wd0       geometry: 2586/240/63 [39100320 Sectors]
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
*0: 06    0   1  1 -  202 239 63 [          63:     3069297 ] DOS > 32MB
 1: A6  203   0  1 - 2585 239 63 [     3069360:    36030960 ] OpenBSD
 2: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
 3: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
fdisk:*1> p m
Disk: wd0       geometry: 2586/240/63 [19092 Megabytes]
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
*0: 06    0   1  1 -  202 239 63 [          63:        1499M] DOS > 32MB
 1: A6  203   0  1 - 2585 239 63 [     3069360:       17593M] OpenBSD
 2: 00    0   0  0 -    0   0  0 [           0:           0M] unused
 3: 00    0   0  0 -    0   0  0 [           0:           0M] unused
fdisk:*1>
```

It is important that the first partition skips the first track of the disk, in this case, starting on sector `63`. If an OpenBSD partition is created starting at offset `0`, this partition table will end up being overwritten by the OpenBSD partition's Partition Boot Record. The system will probably still be bootable, but it will be very difficult to maintain, and this configuration is *not recommended or supported*.

Note that the prompt changed to include an asterisk ('*') to indicate you have unsaved changes. As we can see from the output of `p m` we have not altered our Windows partition, we have successfully allocated the rest of the drive for OpenBSD, and the partitions do not overlap. We are in business. Almost.

What we haven't done is flagged the partition as active so the machine will boot OpenBSD on the next reboot:

```
fdisk:*1> f 1
Partition 1 marked active.
fdisk:*1> p
Disk: wd0       geometry: 2586/240/63 [39100320 Sectors]
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
 0: 06    0   1  1 -  202 239 63 [          63:     3069297 ] DOS > 32MB
*1: A6  203   0  1 - 2585 239 63 [     3069360:    36030960 ] OpenBSD
 2: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
 3: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
fdisk:*1>
```

And now, we are ready to save our changes:

```
fdisk:*1> w
Writing MBR at offset 0.
wd0: no disk label
fdisk: 1> q
```

Creating a disklabel

The next step is to use disklabel(8) to slice up the OpenBSD partition. More details on using disklabel(8) can be found in FAQ 14, disklabel.

```
Here is the partition information you chose:

Disk: wd0       geometry: 2586/240/63 [39100320 Sectors]
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
*0: 06    0   1  1 -  202 239 63 [          63:     3069297 ] DOS > 32MB
 1: A6  203   0  1 - 2585 239 63 [     3069360:    36030960 ] OpenBSD
 2: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
 3: 00    0   0  0 -    0   0  0 [           0:           0 ] unused

You will now create an OpenBSD disklabel inside the OpenBSD MBR
partition. The disklabel defines how OpenBSD splits up the MBR partition
into OpenBSD partitions in which filesystems and swap space are created.

The offsets used in the disklabel are ABSOLUTE, i.e. relative to the
start of the disk, NOT the start of the OpenBSD MBR partition.

disklabel: no disk label
WARNING: Disk wd0 has no label. You will be creating a new one.

# using MBR partition 1: type A6 off 3069360 (0x2ed5b0) size 36030960 (0x225c9f0)

Treating sectors 3069360-39100320 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.
        D         - set label to default.
        g [d|b]   - Use [d]isk or [b]ios geometry.
        m [part]  - modify existing partition.
        n [part]  - set the mount point for a 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.
        X         - toggle expert mode.
        z         - zero out partition table.
        ? [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.
>
```

Again, a few of these commands could use a little elaboration:

- **p** - displays (prints) the current disklabel to the screen, and you can use the modifiers `k, m` or `g` for kilobytes, megabytes or gigabytes.
- **D** - Clears any existing disklabel, creates a new default disklabel which covers just the current OpenBSD partition. This can be useful if the disk previously had a disklabel on it, and the OpenBSD partition was recreated to a different size -- the old disk label may not get deleted, and may cause confusion.
- **m** - Modifies an existing entry in a disklabel. Do not over estimate what this will do for you. While it may alter the size of a disklabel partition, it will NOT alter the filesystem on the drive. Using this option and expecting it to resize existing partitions is a good way of losing large amounts of data.

Slicing up your disk properly is important. The answer to the question, "How should I partition my system?" is "Exactly how you need it". This will vary from application to application. There is no universal answer. If you are unsure of how you want to partition your system, see this discussion.

In this system, we have over 17G available for OpenBSD. That's a lot of space, and it isn't likely we will need most of it. So, we will deliberately not use absolute minimum sizes. We would rather have a few hundred megabytes of unused space than a kilobyte too little.

On the root disk, the two partitions 'a' and 'b' **must** be created. The installation process will not proceed until these two partitions are available. 'a' will be used for the root filesystem (/) and 'b' will be used as swap space.

After a little thought, we decide to create just enough partitions to allow the creation of the recommended separate filesystems (`/, /tmp, /var, /usr, /home`) along with a swap partition:

- ***wd0a:*** */* (root) - 150M. Should be more than enough.
- ***wd0b:*** (swap) - 300M.
- ***wd0d: /tmp*** - 120M. `/tmp` is used for building some software, 120M will probably be enough for most things.
- ***wd0e: /var*** - 80M. If this were to be a web or mail server, we'd have made this partition much larger, but, that's not what we are doing.
- ***wd0g: /usr*** - 2G. We want this partition to be large enough to load quite a few user applications, plus be able to update and rebuild the system if desired or needed. The Ports tree will be here as well, which will take almost 100M of this space before ports are built.
- ***wd0h: /home*** - 4G. This will allow plenty of user file space.

Now, if you add those up, you will see over 10G of space is unused! Unused space won't hurt anything, and it gives us flexibility to enlarge things in the future if need be. Need more */tmp*? No problem, create a new one in the unused space, change */etc/fstab* and problem solved.

```
> p m
device: /dev/rwd0c
type: ESDI
disk: ESDI/IDE disk
label: ST320011A
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total sectors: 39102336
free sectors: 36030960
rpm: 3600

16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  a: 17593.2M  1498.7M    unused        0     0
  c: 19092.9M     0.0M    unused        0     0
  i:  1498.7M     0.0M     MSDOS
> d a
> a a
offset: [3069360] Enter
size: [36030960] 150M
Rounding to nearest cylinder: 307440
FS type: [4.2BSD] Enter
mount point: [none] /
> a b
offset: [3376800] Enter
size: [35723520] 300M
Rounding to nearest cylinder: 614880
FS type: [swap] Enter
> a d
offset: [3991680] Enter
size: [35108640] 120m
Rounding to nearest cylinder: 245952
FS type: [4.2BSD] Enter
mount point: [none] /tmp
```

```
> a e
offset: [4237632] Enter
size: [34862688] 80m
Rounding to nearest cylinder: 164304
FS type: [4.2BSD] Enter
mount point: [none] /var
> a g
offset: [4401936] Enter
size: [34698384] 2g
Rounding to nearest cylinder: 4194288
FS type: [4.2BSD] Enter
mount point: [none] /usr
> a h
offset: [8596224] Enter
size: [30504096] 4g
Rounding to nearest cylinder: 8388576
FS type: [4.2BSD] Enter
mount point: [none] /home
> p m
device: /dev/rwd0c
type: ESDI
disk: ESDI/IDE disk
label: ST320011A
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total sectors: 39102336
free sectors: 22115520
rpm: 3600

16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  a:   150.1M  1498.7M    4.2BSD     1024  8192    16  # /
  b:   300.2M  1648.8M      swap
  c: 19092.9M     0.0M    unused        0     0
  d:   120.1M  1949.1M    4.2BSD     1024  8192    16  # /tmp
  e:    80.2M  2069.2M    4.2BSD     1024  8192    16  # /var
  g:  2048.0M  2149.4M    4.2BSD     1024  8192    16  # /usr
  h:  4096.0M  4197.4M    4.2BSD     1024  8192    16  # /home
  i:  1498.7M     0.0M     MSDOS
> q
Write new label?: [y] Enter
```

You will note there is a *c* partition we seem to have ignored. This partition is your entire hard disk; don't attempt to alter it. You will also note the *i* partition wasn't defined by us; this is the pre-existing Windows 2000 partition. Partitions are not assigned any particular letters -- with the exception of *a* (root), *b* (swap) and *c* (entire disk), the rest of the partitions (through letter *p*) are available for use as you desire.

If you look closely at the output of the disklabel, you will note that your drive RPM rating is probably wrong. This is historical; the drive speed is not used in any way by the system. Do not worry about it.

Configuring your mount points and formatting your filesystems

Now comes the final configuration of your mount points. If you configured the mount points through disklabel(8), this step consists of just verifying your selections; otherwise, you can specify them now.

```
The root filesystem will be mounted on wd0a.
wd0b will be used for swap space.
Mount point for wd0d (size=122976k), none or done? [/tmp] Enter
Mount point for wd0e (size=82152k), none or done? [/var] Enter
Mount point for wd0g (size=2097144k), none or done? [/usr] Enter
Mount point for wd0h (size=4194288k), none or done? [/home] Enter
Mount point for wd0d (size=122976k), none or done? [/tmp] done
Done - no available disks found.

You have configured the following partitions and mount points:

wd0a /
wd0d /tmp
wd0e /var
wd0g /usr
wd0h /home

The next step creates a filesystem on each partition, ERASING existing data.
Are you really sure that you're ready to proceed? [n] y
/dev/rwd0a:     307440 sectors in 305 cylinders of 16 tracks, 63 sectors
        150.1MB in 20 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
/dev/rwd0d:     245952 sectors in 244 cylinders of 16 tracks, 63 sectors
        120.1MB in 16 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
/dev/rwd0e:     164304 sectors in 163 cylinders of 16 tracks, 63 sectors
        80.2MB in 11 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
/dev/rwd0g:     4194288 sectors in 4161 cylinders of 16 tracks, 63 sectors
        2048.0MB in 261 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
/dev/rwd0h:     8388576 sectors in 8322 cylinders of 16 tracks, 63 sectors
        4096.0MB in 521 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
/dev/wd0a on /mnt type ffs (rw, asynchronous, local, ctime=Thu Oct 10 21:
50:36 2 002)
/dev/wd0h on /mnt/home type ffs (rw, asynchronous, local, nodev, nosuid,
 ctime=Thu Oct 10 21:50:36 2002)
/dev/wd0d on /mnt/tmp type ffs (rw, asynchronous, local, nodev, nosuid,
ctime=Thu Oct 10 21:50:36 2002)
/dev/wd0g on /mnt/usr type ffs (rw, asynchronous, local, nodev, ctime=Th
u Oct 10 21:50:36 2002)
/dev/wd0e on /mnt/var type ffs (rw, asynchronous, local, nodev, nosuid,
ctime=Th u Oct 10 21:50:36 2002)
```

You may wonder why the installer again asks for mount points. This allows you to recover from any errors or omissions in the mount points specified during the creation of the disklabel. For instance, the installation process will automatically delete any duplicate mount points you enter during the configuration of the disklabel. The disklabel program will allow you to enter such duplicates, and thus they must be checked for after the disklabel program exits. The deleted duplicate mount points will result in partitions without mount points, that you must assign new mount points for if you wish to use the space.

Notice the "Are you really sure that you are ready to proceed?" question defaults to no, so you will have to deliberately tell it to proceed and format your partitions. If you chose no, you would simply be dropped into a shell and could start the install again by typing install, or just by rebooting again with your boot disk.

At this point all filesystems will formatted for you. This could take some time depending on the size of the partitions and the speed of the disk.

4.5.3 - Setting the system hostname

Now you must set the system hostname. This value, along with the DNS domain name (specified below), will be saved in the file `/etc/myname`, which is used during normal boot to set the hostname of the system. If you do not set the domain name of the system, the default value of 'my.domain' will be used.

It is important to set this name now, because it will be used when the cryptographic keys for the system are generated during the first boot after installation. This generation takes place whether the network is configured or not.

```
Enter system hostname (short form, e.g. 'foo'): puffy
```

4.5.4 - Configuring the network

Now it is time to configure your network. The network must be configured if you are planning on doing a ftp or nfs based install, considering it will be based upon the information you are about to enter. Here is a walk through of the network configuration section of the install process.

```
Configure the network? [y] Enter
Available interfaces are: fxp0.
Which one do you wish to initialize? (or 'done') [fxp0] Enter
Symbolic (host) name for fxp0? [puffy] Enter
The default media for fxp0 is
        media: Ethernet autoselect (100baseTX full-duplex)
Do you want to change the default media? [n] Enter
IP address for fxp0? (or 'dhcp') 199.185.137.55
Netmask? [255.255.255.0] Enter
Done - no available interfaces found.
DNS domain name? (e.g. 'bar.com') [my.domain] example.com
DNS nameserver? (IP address or 'none') [none] 199.185.137.1
Use the nameserver now? [y] Enter
Default route? (IP address, 'dhcp' or 'none') 199.185.137.128
add net default: gateway 199.185.137.128
Edit hosts with ed? [n] Enter
Do you want to do any manual network configuration? [n] Enter
```

In the above example, we use a static IP address. As indicated, you can use "dhcp" instead on most platforms (not Alpha), assuming your environment supports it. In the case of DHCP, most of the information will be grabbed from the remote DHCP server; you will be given a chance to confirm it. Here is a sample of the network configuration part of the install, this time done with DHCP:

```
Configure the network? [y] Enter
Available interfaces are: fxp0.
Which one do you wish to initialize? (or 'done') [fxp0] Enter
Symbolic (host) name for fxp0? [puffy] Enter
The default media for fxp0 is
        media: Ethernet autoselect (100baseTX full-duplex)
Do you want to change the default media? [n] Enter
IP address for fxp0? (or 'dhcp') dhcp
Issuing hostname-associated DHCP request for fxp0.
Internet Software Consortium DHCP Client 2.0pl5-OpenBSD
Listening on BPF/fxp0/00:08:c7:77:b4:6b
Sending on   BPF/fxp0/00:08:c7:77:b4:6b
Sending on   Socket/fallback/fallback-net
DHCPDISCOVER on fxp0 to 255.255.255.255 port 67 interval 1
DHCPOFFER from 199.185.137.128
DHCPREQUEST on fxp0 to 255.255.255.255 port 67
DHCPACK from 199.185.137.128
New Network Number: 199.185.137.0
New Broadcast Address: 199.185.137.255
bound to 199.185.137.55 -- renewal in 43200 seconds.
Done - no available interfaces found.
DNS domain name? (e.g. 'bar.com') [example.org] Enter
DNS nameserver? (IP address or 'none') [199.185.137.1] Enter
Use the nameserver now? [y] Enter
Default route? (IP address, 'dhcp' or 'none') [199.185.137.128] Enter
add net default: gateway 199.185.137.128
Edit hosts with ed? [n] Enter
Do you want to do any manual network configuration? [n] Enter
```

NOTE: Only **one** interface can easily be configured using DHCP during an install. If you attempt to configure more than one interface using DHCP you will encounter errors. You have to manually configure the additional interfaces after the installation.

Now, we set the password for the root account:

```
Password for root account? (will not echo) pAssWOrd
Password for root account? (again) pAssWOrd
```

Use a secure password for the root account. You will create other user accounts after the system is booted. From passwd(1):

```
The new password should be at least six characters long and not purely
alphabetic.  Its total length must be less than _PASSWORD_LEN (currently
128 characters).  A mixture of both lower and uppercase letters, numbers,
and meta-characters is encouraged.
```

4.5.5 - Choosing installation media

After your network is set up, the install script will give you a chance to make manual adjustments to the configuration. Then the filesystems you created will be mounted and a root password set. This will get your local disks ready for the OpenBSD packages to be installed upon them.

Next, you will get a chance to choose your installation media. The options are listed below.

```
You will now specify the location and names of the install sets you want to
load. You will be able to repeat this step until all of your sets have been
successfully loaded. If you are not sure what sets to install, refer to the
installation notes for details on the contents of each.

Sets can be located on a (m)ounted filesystem; a (c)drom, (d)isk or (t)ape
device; or a (f)tp, (n)fs or (h)ttp server.
Where are the install sets? c
Available CD-ROMs are: cd0.
```

In this example we are installing from CD-ROM. This will bring up a list of devices on your computer identified as a CD-ROM. Most people will only have one. If you need to, make sure you pick the device which you will use to install OpenBSD from.

NOTE: All possible sources for install sets are listed, but not all may be available on your system. e.g. (n)fs is shown but not all architectures allow NFS installations. If you choose a source that is not available, you will get an error message and be given the chance to choose another source for your installation sets.

```
Available CD-ROMs are: cd0.
Which one contains the install media? (or 'done') [cd0] Enter

Pathname to the sets?  (or 'done') [3.4/i386] Enter
```

Here, you are prompted for which directory the installation files are, which is `3.4/i386/` on the official CD-ROM.

4.5.6 - Choosing filesets.

Now it's time to choose which packages you will be installing. You can get a description of these files in the next section. The files that the install program finds will be shown to you on the screen. Your job is just to specify which files you want. By default all the non-X packages are selected; however, some people may wish to limit this to the bare minimum required to run OpenBSD, which would be `base34.tgz`, `etc34.tgz` and `bsd`. Others will wish to install all packages. The example below is that of a full install.

```
The following sets are available. Enter a filename, 'all' to select
all the sets, or 'done'. You may de-select a set by prepending a '-'
to its name.

        [X] bsd
        [ ] bsd.rd
        [X] base34.tgz
        [X] etc34.tgz
        [X] misc34.tgz
        [X] comp34.tgz
        [X] man34.tgz
        [X] game34.tgz
        [ ] xbase34.tgz
        [ ] xshare34.tgz
        [ ] xfont34.tgz
        [ ] xserv34.tgz
```

```
File Name? (or 'done') [bsd.rd] all

The following sets are available. Enter a filename, 'all' to select
all the sets, or 'done'. You may de-select a set by prepending a '-'
to its name.

        [X] bsd
        [X] bsd.rd
        [X] base34.tgz
        [X] etc34.tgz
        [X] misc34.tgz
        [X] comp34.tgz
        [X] man34.tgz
        [X] game34.tgz
        [X] xbase34.tgz
        [X] xshare34.tgz
        [X] xfont34.tgz
        [X] xserv34.tgz
```

You can do all kinds of nifty things here -- `-x*` would remove all X components, if you changed your mind. In this case, we are going to load all the sets. While the system will run with fewer sets, either the starting default or installing all sets is recommended. More details on selecting sets here.

Once you have successfully picked which packages you want, you will be prompted to make sure you want to extract these packages and they will then be installed. A progress bar will be shown that will keep you informed on how much time it will take. The times range greatly depending on what system it is you are installing OpenBSD on, the packages installed, and the speed of the source media. This part may from a few minutes to several hours.

```
File Name? (or 'done') [done] Enter
Ready to install sets? [y] Enter
Getting bsd ...
100% |***************************************************|  4735 KB    00:03
Getting bsd.rd ...
100% |***************************************************|  4275 KB    00:02
Getting base34.tgz ...
100% |***************************************************| 30267 KB    00:21
Getting etc34.tgz ...
100% |***************************************************|  1545 KB    00:01
Getting misc34.tgz ...
100% |***************************************************|  1909 KB    00:01
Getting comp34.tgz ...
100% |***************************************************| 17074 KB    00:13
Getting man34.tgz ...
100% |***************************************************|  6139 KB    00:04
Getting game34.tgz ...
100% |***************************************************|  2534 KB    00:01
Getting xbase34.tgz ...
100% |***************************************************| 10940 KB    00:06
Getting xshare34.tgz ...
100% |***************************************************|  1656 KB    00:02
Getting xfont34.tgz ...
100% |***************************************************| 31160 KB    00:21
Getting xserv34.tgz ...
100% |***************************************************| 15228 KB    00:11

Sets can be located on a (m)ounted filesystem; a (c)drom, (d)isk or (t)ape
device; or a (f)tp, (n)fs or (h)ttp server.
Where are the install sets? (or 'done')
```

At this point, you can pull additional files from other sources (including custom file sets) if desired, or hit '`done`' if you have installed all the file sets you need.

4.5.7 - Finishing up

You will now be asked if you plan to run X on this system. If you answer 'Y', `/etc/sysctl.conf` will be modified to include the line `machdep.allowaperture=1` or `machdep.allowaperture=2`, depending on your platform.

```
Do you expect to run the X Window System? [y] y
```

Your last task is to enter the time zone. Depending on where your machine lives, there are may be several equally valid answers for the question. In the example that follows, we used `US/Eastern`, but could also have used `EST5EDT` or `US/Michigan` and had the same result. Hitting `?` at the prompts will guide you through your choices.

```
Saving configuration files......done.
Generating initial host.random file ......done.
What timezone are you in? ('?' for list) [US/Pacific] ?
Africa/      Chile/       GB-Eire      Israel       NZ-CHAT      Turkey
America/     Cuba         GMT          Jamaica      Navajo       UCT
Antarctica/  EET          GMT+0        Japan        PRC          US/
Arctic/      EST          GMT-0        Kwajalein    PST8PDT      UTC
Asia/        EST5EDT      GMT0         Libya        Pacific/     Universal
Atlantic/    Egypt        Greenwich    MET          Poland       W-SU
Australia/   Eire         HST          MST          Portugal     WET
Brazil/      Etc/         Hongkong     MST7MDT      ROC          Zulu
CET          Europe/      Iceland      Mexico/      ROK          posix/
CST6CDT      Factory      Indian/      Mideast/     Singapore    posixrules
Canada/      GB           Iran         NZ           SystemV/     right/
What timezone are you in? ('?' for list) [US/Pacific] US
What sub-timezone of 'US' are you in? ('?' for list) ?
Alaska          Central         Hawaii          Mountain        Samoa
Aleutian        East-Indiana    Indiana-Starke  Pacific
Arizona         Eastern         Michigan        Pacific-New
Select a sub-timezone of 'US' ('?' for list): Eastern
Setting local timezone to 'US/Eastern'...done.
```

If you are concerned about very precise time, you may wish to read this.

The last steps are for the system to create the `/dev` directory (which may take a while on some systems, especially if you have a small amount of RAM), and install the boot blocks.

```
Making all device nodes...done.
Installing boot block...
boot: /mnt/boot
proto: /usr/mdec/biosboot
device: /dev/rwd0c
/usr/mdec/biosboot: entry point 0
proto bootblock size 512
room for 12 filesystem blocks at 0x16f
Will load 7 blocks of size 8192 each.
Using disk geometry of 63 sectors and 240 heads.
 0:  9 @(203 150 55) (3078864-3078872)
 1: 63 @(203 151 1) (3078873-3078935)
 2: 24 @(203 152 1) (3078936-3078959)
 3: 16 @(203 8 47) (3069910-3069925)
/mnt/boot: 4 entries total
using MBR partition 1: type 166 (0xa6) offset 3069360 (0x2ed5b0)
...done.

CONGRATULATIONS! Your OpenBSD install has been successfully completed!
To boot the new system, enter halt at the command prompt. Once the
system has halted, reset the machine and boot from the disk.
# halt
syncing disks... done

The operating system has halted.
Please press any key to reboot.
```

OpenBSD is now installed on your system and ready for its first boot, but before you do...

Before you reboot

At this point, your system is installed and ready to be rebooted and configured for service. Before doing this, however, it would be wise to check out the Errata page to see if there are any bugs that would immediately impact you.

After you reboot

One of your first things to read after you install your system is afterboot(8).

You may also find the following links useful:

- Adding users in OpenBSD
- Initial Network Setup
- Man Pages of popular/useful commands
- OpenBSD man pages on the Web
- The OpenBSD Ports and Packages system for installing software, as well as here and here

One last thing...

The OpenBSD developers ask you to Send in a copy of your dmesg. This is really appreciated by the developers, and ultimately, all users.

4.6 - What files are needed for installation?

The complete OpenBSD installation is broken up into a number of separate *file sets*. Not every application requires every file set. Here is an overview of each:

- ***bsd*** - This is the Kernel. **Required**
- ***bsd.rd*** - RAM disk kernel
- ***base34.tgz*** - Contains the base OpenBSD system **Required**
- ***etc34.tgz*** - Contains all the files in /etc **Required**
- ***comp34.tgz*** - Contains the compiler and its tools, libs. **Recommended**
- ***man34.tgz*** - Contains man pages **Recommended**
- ***misc34.tgz*** - Contains misc info, setup documentation
- ***game34.tgz*** - Contains the games for OpenBSD
- ***xbase34.tgz*** - Contains the base install for X11
- ***xfont34.tgz*** - Contains X11's font server and fonts
- ***xserv34.tgz*** - Contains X11's X servers
- ***xshare34.tgz*** - Contains manpages, locale settings, includes, etc for X

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

The following are minimum suggested filesystem 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.

- These are minimum values.
- If you plan to install a significant amount of third party software, make your `/usr` partition large! **At least** triple these values!
- For a system that handles lots of email or web pages (stored, respectively, in `/var/mail` and `/var/www`) you will want to make your `/var` partition significantly larger, or put them on separate partitions.
- For a multiuser system which may generate lots of logs, you will still want to make your `/var` partition significantly larger (`/var/log`).
- If you plan to rebuild the kernel or system from source, you will want to make the `/usr` partition significantly larger, **at least** 800M-1G larger than indicated below.

As you read this, keep in mind that `/usr` and `/usr/X11R6` are usually both parts of the same filesystem, that is, `/usr`, as there is no big advantage to making them into separate filesystems.

SYSTEM	/	/usr	/var	/usr/X11R6
alpha	80M	250M	25M	140M
hp300	80M	250M	25M	140M
hppa	100M	200M	25M	120M
i386	60M	250M	25M	140M
mac68k	80M	250M	25M	100M
macppc	80M	250M	25M	140M
mvme68k	80M	250M	25M	100M
sparc	80M	250M	25M	120M

sparc64	80M	250M	25M	100M
vax	100M	200M	25M	120M

In addition, it is recommended that a `/tmp` partition be used. The `/tmp` partition is used in the compiling of ports, among other things, so how big you make it depends on what you do with it. 50M may be plenty for most people, but some large applications may require 100M or more of `/tmp` space.

When you are in the disklabel editor, you may choose to make your entire system have just an 'a' (main filesystem) and 'b' (swap) . The 'a' filesystem which 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. If you have lots of disk space to spare, make this 256MB, or even 512MB.

Swap space is used to store system core dumps on in the event of a crash(8). If this is a consideration for you, your swap space should be slightly larger than the amount of main memory you are likely to ever have in the system. Note that upon reboot, savecore(8) will attempt to save the contents of the swap partition to a file in `/var/crash` so again, if this is a priority for you, your `/var` partition must have enough *free space* to hold these dump files.

There are five main reasons for using separate filesystems, instead of shoving everything into one or two filesystems:

- **Security**: You can mark some filesystems as 'nosuid', 'nodev', 'noexec', 'readonly', etc. This is now done by the install process, in fact, if you use the above described partitions.
- **Stability**: A user, or a misbehaved program, can fill a filesystem with garbage if they have write permissions for it. Your critical programs, which of course run on a different filesystem, do not get interrupted.
- **Speed**: A filesystem which gets written to frequently may get somewhat fragmented. (Luckily, the ffs filesystem, what OpenBSD uses, is not prone to heavy fragmentation.)
- **Integrity**: If one filesystem is corrupted for some reason then your other filesystems are still OK.
- **Size**: Many platforms have limits on the area of a disk where the boot ROM can load the kernel from. In some cases, this limit may be very small (504M for an older 486), in other cases, a much larger limit (8G on new i386 systems). As the kernel can end up anywhere in the root partition, the entire root partition should be within this area. For more details, see this section. A good guideline might be to keep your / partition completely below 2G, unless you know your platform (and particular machine!) can handle more (or less!) than that.

Some additional thoughts on partitioning:

- For your first attempt at an experimentation system, one big `/` partition and swap may be easiest until you know how much space you need. By doing this you will be sacrificing some of the default security features of OpenBSD that require separate filesystems for `/, /tmp, /var, /usr` and `/home`.
- A system exposed to the Internet or other hostile forces should have a separate `/var` (and maybe even a separate `/var/log`) for logging.
- A `/home` partition can be nice. New version of the OS? Wipe and reload everything else, leave your `/home` partition untouched. Remember to save a copy of your configuration files, though!
- A separate partition for anything which may accumulate a large quantity of files that may need to be deleted can be faster to reformat and recreate than to delete. See the upgrade-minifaq for an example (`/usr/obj`).
- If you wish to rebuild your system from source for any reason, the source will be in `/usr/src`. If you don't make a separate partition for `/usr/src`, make sure `/usr` has sufficient space.
- A commonly forgotten fact: you do **not** have to allocate all space on a drive when you set the system up! Since you will now find it a challenge to buy a new drive smaller than 20G, it can make sense to leave a chunk of your drive unallocated. If you outgrow a partition, you can allocate a new partition from your unused space, duplicate your existing partition to the new partition, change /etc/fstab to point to the new partition, remount, you now have more space.
- If you make your partitions too close to the minimum size required, you will probably regret it later, when it is time to upgrade your system.
- If you permit users to write to `/var/www` (i.e., personal web pages), you might wish to put it on a separate partition, so you can use quotas to restrict the space they use, and if they fill the partition, no other parts of your system will be impacted.

4.8 - Multibooting OpenBSD/i386

Multibooting is having several operating systems on one computer, and some means of selecting the which OS is to boot. It is *not* a trivial task! If you don't understand what you are doing, you may end up deleting large amounts of data from your computer. New OpenBSD users are *highly* encouraged to start with a blank hard drive on a dedicated machine, and then practice your desired configuration on a non-production system before attempting a multiboot configuration on a production machine. FAQ 14 has more information about the OpenBSD boot process.

When multibooting, the requirements of all operating systems must be met by your configuration. People often ask if there is a way around the 8G boot limit of OpenBSD. While there are some programs that claim to get around various limits of various operating systems, none of them are known to do this with current versions of OpenBSD.

Here are several options to multibooting:

Setting active partitions

This is probably the most overlooked, and yet, sometimes the best solution for multibooting. Simply set the active partition in whatever OS you are currently using to be the one you want to boot by default when you next boot. Virtually every OS offers a program to do this; OpenBSD's is fdisk(8), similar named programs are in Windows 9x and DOS, and many other operating systems. This can be highly desirable for OSs or systems which take a long time to shut down and reboot -- you can set it and start the reboot process, then walk away, grab a cup of coffee, and come back to the system booted the way you want it -- no waiting for the Magic Moment to select the next OS.

Boot floppy

If you have a system that is used to boot OpenBSD infrequently (or don't wish other users of the computer to note anything has changed), consider using a boot floppy. Simply use one of the standard OpenBSD install floppies, and create a `/etc/boot.conf` file (yes, you will also have to create an `/etc` directory on the floppy) with the contents:

```
boot hd0a:/bsd
```

to cause the system to boot from hard drive 0, OpenBSD partition 'a', kernel file `/bsd`. Note you can also boot from other drives with a line like: "`boot hd2a:/bsd`" to boot off the third hard drive on your system. To boot from OpenBSD, slip your floppy in, reboot. To boot from the other OS, eject the floppy, reboot.

In this case, the boot(8) program is loaded from the floppy, looks for and reads `/etc/boot.conf`. The "`boot hd0a:/bsd`" line instructs boot(8) where to load the kernel from -- in this case, the first HD the BIOS sees. Keep in mind, only a small file (`/boot`) is loaded from the floppy -- the system loads the entire kernel off the hard disk, so this only adds about five seconds to the boot process.

Windows NT/2000/XP NTLDR

To multiboot OpenBSD and Windows NT/2000/XP, you can use NTLDR, the boot loader that NT uses. To multi-boot with NT, you need a copy of your OpenBSD Partition Boot Record (PBR). After running installboot, you can copy it to a file using dd(1):

```
# 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 NTLDR at the NTLDR Hacking Guide.

On Windows XP you can also edit the boot information using the GUI; see the XP Boot.ini HOWTO.

Programs that do much of this for you are available, for example, BootPart. This program can be run from Windows NT/2000/XP, and will fetch the OpenBSD PBR, place it on your NT/2000/XP partition, and will add it to `C:\BOOT.INI`

The OpenBSD install and upgrade process will re-install the OpenBSD boot loader, which has its location coded in the PBR, so if you re-install or update your OpenBSD installation, you need to repeat the above process to fetch a new copy of the OpenBSD PBR.

Note: The Windows NT boot loader is only capable of booting OSs from the primary hard drive. You can not use it to load OpenBSD from the second drive on a system.

Other boot loaders

Some other bootloaders OpenBSD users have used successfully include GAG, OSBS, The Ranish Partition Manager and GRUB.

OpenBSD and Linux (i386)

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

4.9 - 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 configured!) and type something like:
 # dmesg | mail -s "Sony VAIO 505R laptop, apm works OK" dmesg@openbsd.org
so that we can see what kinds of configurations people are running.  As shown,
including a bit of information about your machine in the subject or the body
can help us even further.  We will use this information to improve device driver
support in future releases.  (Please do this using the supplied GENERIC kernel,
not for a custom compiled kernel, unless you're unable to boot the GENERIC
kernel).  The device driver information we get from this helps us fix existing
drivers. Thank you!
```

Make sure you send email from an account that is able to also receive email so developers can contact you 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 dmesgs. Custom kernels that have device drivers removed are not helpful.

4.10 - Adding a file set after install

"Oh no! I forgot to add a file set when I did the install!"

Sometimes, you realize you really DID need `comp34.tgz` (or any other system component) after all, but you didn't realize this at the time you installed your system. Good news: There are two easy ways to add file sets after the initial install:

Using the upgrade process

Simply boot your install media (CD-ROM or Floppy), and choose Upgrade (rather than Install). When you get to the lists of file sets to install, choose the sets you neglected to install first time around, select your source, and let it install them for you.

Using tar(1)

The install file sets are simply compressed tar files, and you can expand them manually from the root of the filesystem:

```
# cd /
# tar xzvpf comp34.tgz
```

Do NOT forget the `'p'` option in the above command in order to restore the file permissions properly!

One common mistake is to think you can use pkg_add(1) to add a missing file sets. This does not work. pkg_add(1) is for package files, not generic tar files like the install sets.

4.11 - What is 'bsd.rd'?

`bsd.rd` is a "RAM Disk" kernel. This file can be very useful; many developers are careful to keep it on the root of their system at all times.

Calling it a "RAM Disk kernel" describes the root filesystem of the kernel -- rather than being a physical drive, the utilities available after the boot of `bsd.rd` are stored in the kernel, and are run from a RAM-based filesystem. `bsd.rd` also includes a healthy set of utilities to allow you to do system maintenance and installation.

On some platforms, `bsd.rd` is actually the preferred installation technique -- you place this kernel on an existing filesystem, boot it, and run the install from it. On most platforms, if you have a running older version of OpenBSD, you can FTP a new version of `bsd.rd`, reboot from it, and install a new version of OpenBSD without using any removable media at all.

Here is an example of booting `bsd.rd` on an i386 system:

```
  Using Drive: 0 Partition: 3
  reading boot.....
  probing: pc0 com0 com1 apm mem[639k 255M a20=on]
  disk: fd0 hd0+
  >> OpenBSD/i386 BOOT 2.02
  boot> boot hd0a:/bsd.rd
. . . normal boot to install . . .
```

As indicated, you will be brought to the install program, but you can also drop to the shell to do maintenance on your system.

The general rule on booting `bsd.rd` is to change your boot kernel from `/bsd` to `bsd.rd` through whatever means used on your platform.

4.12 - Common installation problems

4.12.1 - My Compaq only recognizes 16M RAM

Some Compaq systems have an issue where the full system RAM is not detected by the OpenBSD second stage boot loader properly, and only 16M may be detected and used by OpenBSD. This can be corrected either by creating/editing `/etc/boot.conf` file, or by entering commands at the "`boot>`" prompt before OpenBSD loads. If you had a machine with 64M RAM, but OpenBSD was only detecting the first 16M, the command you would use would be:

```
machine mem +0x3000000@0x1000000
```

to add 48M (`0x3000000`) after the first 16M (`0x1000000`). Typically, if you had a machine with this problem, you would enter the above command first at the install floppy/CD-ROM's `boot>` prompt, load the system, reboot, and create a `/etc/boot.conf` file with the above line in it so all future bootings will recognize all available RAM.

It has also been reported that a ROM update will fix this on *some* systems.

4.12.2 - My i386 won't boot after install

Your install seemed to go fine, but on first boot, you see no sign of OpenBSD attempting to boot. There are a few common reasons for this problem:

- **No partition was flagged active in fdisk(8).** To fix this, reboot the machine using the boot floppy or media, and "flag" a partition as "active" (bootable). See here and here
- **No valid boot loader was ever put on the disk.** If you answer "Y" to the "Use entire disk for OpenBSD?" question during the install, or use the "`reinit`" or "`update`" options of fdisk(8), the OpenBSD boot record is installed on the Master Boot Record of the disk; otherwise, the existing master boot code is untouched. This will be a problem if no other boot record existed. The solution is to boot the install media again, drop to the shell and invoke fdisk(8) and use the "`update`" option.
- **In some rare occasions, something may go wrong with the second stage boot loader install.** Reinstalling the second stage boot loader is discussed here.
- **You mixed *a.out* and *ELF* boot disk and install files.** OpenBSD/i386 has transitioned from the older *a.out* format to the *ELF* format for binaries shortly after 3.3-release. The OpenBSD 3.4 boot loader can not boot an OpenBSD 3.3-release's kernel, nor can the 3.3-release boot loader boot an OpenBSD 3.4 kernel. You must use a boot disk that matches the version of OpenBSD you intend to install. Failure to follow this will show itself with a message such as "`failure(79)`" and/or "`Inappropriate file type or format`".
- **You ignored all the warnings and explanations about the 8G boot limit.** This results in a hang at the very start of the boot process, sometimes with the message, "`Bad magic`".

4.12.3 - My (older, slower) machine booted, but hung at the ssh-keygen steps

It is very likely your machine is running fine, just taking a while to do the ssh key generation process. A SparcStation2 or a Macintosh Quadra may take 45 minutes or more to complete the three ssh-keygen(1) steps, some machines will take even longer. Just let it finish; it is only done once per install.

4.12.4 - I got the message "Failed to change directory" when doing an install

When doing an FTP install of a snapshot during the *-beta* stage of the OpenBSD development cycle, you may see this:

```
Do you want to see a list of potential FTP servers?  [y] ENTER
Getting the list from 192.128.5.191 (ftp.openbsd.org)... FAILED
Failed to change directory.
Server IP address or hostname?
```

This is normal and expected behavior during this pre-release part of the cycle. The install program looks for the FTP list on the primary FTP server in a directory that won't be available until the release date, so you get the above message.

Simply use the FTP mirror list to find your favorite FTP mirror, and manually enter its name when prompted.

Note: You should not see this if you are installing *-release* or from CD-ROM.

4.12.5 - When I login, I get "login_krb4-or-pwd: Exec format error"

Kerberos IV has been removed from OpenBSD 3.4, but if you did an upgrade, the old Kerberos IV binaries still will be on your system. This is a problem on the i386 platform, as the old Kerberos files are in a.out, format and thus unable to run on the standard ELF kernel (which has a.out emulations disabled, as mentioned here). If you have encountered this problem, you need to override the krb4 authentication method when you log in:

```
OpenBSD/i386 (puffy.openbsd.org) (ttyC0)

login: joeuser:passwd
password:
```

You can use the same "*username*`:passwd`" syntax with an ssh connection and with su(1) to access your system. Now edit `/etc/login.conf`, and remove the krb4 references.

4.13 - Customizing the install process

`siteXX.tgz` file

The OpenBSD install/upgrade scripts allow the selection of a user-created set called "`siteXX.tgz`", where XX is the release version (e.g. 34). The `siteXX.tgz` file set is, like the other file sets, a gzip(1) compressed tar(1) archive rooted in '/' and is un-tarred like the other sets with the options `xzpf`. This set will be installed last, after all other file sets.

This file set allows the user to add to and/or override the files installed in the 'normal' sets and thus customize the installation or upgrade.

Some example uses of a `siteXX.tgz` file:

- Create a siteXX.tgz file that contains all the file changes you made since first installing OpenBSD. Then, if you have to re-create the system you simply select siteXX.tgz during the re-install and all of your changes are replicated on the new system.
- Create a series of machine specific directories that each contain a siteXX.tgz file that contains files specific to those machine types. Installation of machines (e.g. boxes with different graphics cards) of a particular category can be completed by selecting the appropriate siteXX.tgz file.
- Put the files you routinely customize in a same or similar way in a siteXX.tgz file -- `/etc/skel` files, `/etc/pf.conf`, `/var/www/conf/httpd.conf`, `/etc/rc.conf`, etc.

`install.site`/`upgrade.site` scripts

As the last step in the install/upgrade process, the scripts look in the root directory of the newly installed/upgraded system for `install.site` or `upgrade.site`, as appropriate to the current process, and runs this script in an environment chrooted to the installed/upgraded system's root. Remember, the

upgrade is done from a booted file system, so your target file system is actually mounted on `/mnt`. However, your script can be written as if it it is running in the "normal" root of your file system. Since this script is run after all the files are installed, you have almost full functionality of your system (though, in single user mode) when your script runs.

Note that the `install.site` script would have to be in a `siteXX.tgz file`, while the `upgrade.site` script could be put in the root directory before the upgrade, or could be put in a `siteXX.tgz` file.

The scripts can be used to do anything possible in a script.

- Remove files that are installed/upgraded that you don't want present on the system.
- Remove/upgrade/install the packages you want on the installed system.
- Do an immediate backup/archive of the new system before you expose it to the rest of the world.

The combination of `siteXX.tgz` and `install.site/upgrade.site` files is intended to give the user broad customization capabilities without having to build their own custom install sets.

4.14 - How can I install a number of similar systems?

Here are some tools you can use when you have to deploy a number of similar OpenBSD systems.

`siteXX.tgz` and `install/upgrade.site` files

See the above article.

Restore from dump(8)

On most platforms, the boot media includes the restore(8) program, which can be used to restore a backup made by dump(8). Thus, you could boot from a floppy, CD, or bsd.rd file, then fdisk, disklabel, and restore the desired configuration from tape or other media, and install the boot blocks. More details here.

Disk imaging

Unfortunately, there are no known disk imaging packages which are FFS-aware and can make an image containing only the active file space. Most of the major disk imaging solutions will treat an OpenBSD partition as a "generic" partition, and can make an image of the whole disk. This often accomplishes your goal, but usually with huge amounts of wasted space -- an empty, 10G `/home` partition will require 10G of space in the image, even if there isn't a single file in it. While you can typically install a drive image to a larger drive, you would not be able to directly use the extra space, and you would not be able to install an image to a smaller drive.

If this is an acceptable situation, you may find the dd command will do what you need, allowing you to copy one disk to another, sector-for-sector. This would provide the same functionality as commercial programs without the cost.

4.15 - How can I get a dmesg(8) to report an install problem?

When reporting a problem, it is critical to include the complete system dmesg(8). However, often when you need to do this, it is because the system is working improperly or won't install so you may not have disk, network, or other resources you need to get the dmesg to the appropriate mail list. There are other ways, however:

- **Floppy disk:** The boot disks and CD-ROM has enough tools to let you record your dmesg to an MSDOS floppy disk for reading on another machine. Place an MSDOS formatted floppy in your disk drive and execute the following commands:

  ```
  mount -t msdos /dev/fd0a /mnt
  dmesg >/mnt/dmesg.txt
  umount /mnt
  ```

  If you have another OpenBSD system, you can also write it to an OpenBSD compatible floppy -- often, the boot floppy has enough room on it to hold the dmesg. In that case, leave off the "`-t msdos`" above.

- **Serial Console:** See this article on setting up the serial console, then capture the output to a file.

- **FTP:** Under some circumstances, you may be able to use the ftp(1) client on the boot disk or CD-ROM to send the dmesg to a local FTP server, where you can retrieve it later.

4.16 - Upgrading/reinstalling OpenBSD/i386 using `bsd.rd-a.out`

It is normally possible to perform upgrades and installs using the bsd.rd kernel. However, with OpenBSD 3.4, the i386 platform switched executable format from a.out to ELF, so older boot loaders (OpenBSD 3.3 and before) are unable to run the new-format `bsd.rd` kernel.

To circumvent this problem, and allow upgrades to be performed using `bsd.rd`, an a.out version of `bsd.rd` has been made available on the FTP distribution. This file, `bsd.rd-a.out`, can be booted by OpenBSD 3.3 and below, but is a genuine OpenBSD 3.4 kernel, including the new ELF boot loader, so can be used to bootstrap OpenBSD/i386 3.4 from an older system.

Simply download `bsd.rd-a.out` and place it in your machine's root directory. Boot it instead of the normal `bsd` or `bsd.rd` kernels as shown here (specifying `bsd.rd-a.out` as your boot kernel, of course).

If you wish to install *-current*, it is recommended you first install a minimal 3.4-release system (`base34.tgz, etc34.tgz, bsd`), then reinstall using the *-snapshot* `bsd.rd` file.

[FAQ Index] [To Section 3 - Obtaining OpenBSD] [To Section 5 - Building the System from Source]


www@openbsd.org

$OpenBSD: faq4.html,v 1.167 2004/03/28 02:55:09 nick Exp $

[Index de La FAQ] [Section 4 - Guide d'Installation] [Section 6 - Mise en place du réseau]

5 - Construire le Système à partir des Sources

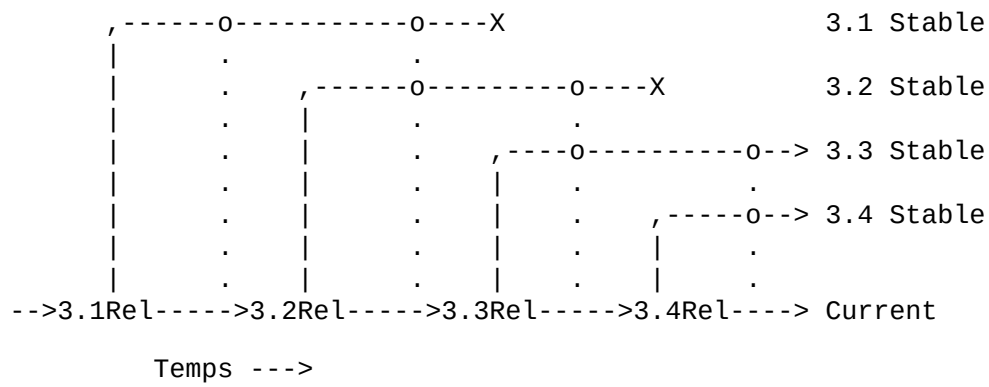
Table des matières

5.1 - Les Saveurs ("Flavors") OpenBSD

Il existe trois "saveurs" OpenBSD :

- **-release:** La version d'OpenBSD livrée tous les six mois sur CD.
- **-stable:** "release", plus les correctifs considérés comme critiques pour la sécurité et la fiabilité.
- **-current:** La version courante (ou de développement) d'OpenBSD, qui deviendra la prochaine "release".

Graphiquement, le développement de ces versions ressemble à ceci :



L'arborescence des sources est séparée en trois branches tous les six mois : *-current*, *-release*, et *-stable* -- *-release* devient un point gelé (une "Balise") dans l'historique de l'arborescence des sources - aucune modification n'est apportée à cette branche. Elle constitue ce qui figure sur les CDs et les serveurs FTP . *-Current* est la branche qui reçoit tous les travaux en cours, et devient la prochaine *-release* d'OpenBSD.

La branche *-stable* (aussi connue sous le nom de "branche des correctifs") est basée sur *-release*, et comme on peut le constater, c'est une "branche" à partir de la trajectoire de développement d'OpenBSD. Lorsque des correctifs importants sont crées pour *-current*, ils sont portés "en arrière" vers les branches *-stable*. Dans l'illustration ci-dessus, les lignes verticales en pointillé sont des correctifs de bogues incorporés aux branches *-stable*. Vous remarquerez aussi dans l'exemple ci-dessus que la branche *3.1-stable* a été supprimée à la sortie de *3.3-release*, et que la branche *3.2-stable* a été supprimée à la sortie de *3.4-release* -- les anciennes versions sont typiquement supportées durant deux "releases" au maximum. Le support d'anciennes versions nécessite des ressources et du temps, et alors que nous pourrions vouloir plutôt fournir un support continu pour les anciennes versions, nous préférerions nous concentrer sur les nouvelles fonctionnalités. La branche *-stable* est, par conception, très facile à construire à partir de *-release* de la même version (i.e., en allant de *3.4-release* vers *3.4-stable*).

La branche *-stable* est *-release* plus les correctifs listés dans la page des errata, ainsi que quelques correctifs ne nécessitant pas d'erratum. Habituellement, le fonctionnement de *-stable* est le même que celui de *-release* sur laquelle elle est basée. Si les pages du manuel doivent être modifiés, il est très probable que ces modifications ne feront pas partie de *-stable*. En d'autres termes, le support de nouveaux périphériques NE sera pas ajouté à *-stable*, et le support de nouvelles fonctionnalités sera très rarement ajouté sauf si c'est considéré comme très important.

Attention : *-current* change tout le temps. Elle change pratiquement toutes les minutes, et pourrait bien changer plusieurs fois le temps de télécharger le code source. Comme précédemment indiqué, aucune garantie n'est donnée quant à la compilation du système ou à son fonctionnement (bien entendu, nous espérons qu'il fonctionne). Il est complètement possible et pas si rare d'obtenir les sources *-current* et de voir leur compilation échouer et cinq minutes plus tard, il se pourrait que ça fonctionne correctement. **Si vous n'êtes pas prêt à gérer ce type de situation, prenez vos distances de *-current*.**

La plupart des utilisateurs devraient utiliser *-stable* ou *-release*. Cela dit, plusieurs personnes utilisent *-current* sur des systèmes en production, et il est important que certaines personnes fassent cela pour identifier des bogues et tester les nouvelles fonctionnalités. Cependant, si vous ne savez pas comment décrire, diagnostiquer et gérer proprement un problème, ne vous dites pas (ou à quelqu'un d'autre) que vous être entrain d'"aider le projet" en utilisant *-current*. "Ç ne marche pas !" n'est pas un rapport de bogue utile. "Les changements récents dans le pilote pciide a brisé la compatibilité avec mon interface IDE basée sur Slugchip, ci-joint le dmesg d'un système fonctionnel et un système ne fonctionnant pas..." peut être un rapport utile.

Des fois, les utilisateurs "normaux" veulent disposer des derniers développements et utiliser *-current*. La raison la plus commune de faire cela est que l'utilisateur possède un périphérique qui n'est pas supporté par *-release* (et donc, par *-stable* non plus), ou il souhaite utiliser une nouvelle fonctionnalité de *-current*. Dans ce cas, soit l'utilisateur utilise *-current* ou n'utilise pas le périphérique, et utiliser *-current* est peut-être l'option la plus logique. Cependant, il ne faut pas espérer que les développeurs vous tiennent la main.

Snapshots

Entre les versions formelles d'OpenBSD, des *snapshots* sont mis à disposition sur les sites FTP . Comme le nom l'indique, ce sont des images du code dans l'arborescence à l'instant où le créateur de l'image a pris une copie du code pour une plate-forme donnée. Il est à noter que, pour certaines plates-formes, il peut s'écouler des jours entiers avant que l'image soit complètement construite et mise à disposition. Aucune garantie n'est donnée quant au bon fonctionnement ou à la possibilité d'installer des snapshots. Souvent, une modification qui a besoin d'être testée peut enclencher le processus de création des snapshots. Quelques plates-formes ont des snapshots construits pratiquement tous les jours, d'autres en ont beaucoup moins fréquemment. Si vous souhaitez utiliser *-current*, un snapshot récent est tout ce dont vous aurez besoin, et mettre à jour un snapshot est généralement un bon point de départ avant de tenter de compiler *-current*.

Parfois, on demande s'il y a un moyen d'obtenir une copie exacte du code qui a servi à construire un snapshot. La réponse est non. Primo, il n'y a aucun intérêt. Secundo, les snapshots sont construits selon le souhait des développeurs, lorsque le planning le permet, et lorsque des ressources sont disponibles. Sur les plates-formes rapides, il est possible de créer plusieurs snapshots en un

jour. Sur les plates-formes lentes, la création d'un snapshot peut durer une semaine ou plus. Fournir des balises ou des marques dans l'arborescence des sources pour chaque snapshot peut s'avérer peu pratique.

Garder Les Composants Synchronisés

Il est important de comprendre qu'OpenBSD est un Système d'Exploitation, et il faut le prendre en tant que tel et non pas comme un noyau entouré d'un ensemble d'outils. Vous devez vous assurer que votre noyau, le "userland" (les utilitaires et fichiers complétant le noyau) et l'arborescence des `ports` sont synchronisés, autrement des choses désagréables peuvent arriver. Dit autrement (vu que les gens continuent à commettre les mêmes erreurs), vous ne pouvez pas utiliser des `ports` tout neufs sur un système datant d'il y a un mois, ou reconstruire un noyau à partir de *-current* et espérer qu'il fonctionne avec un "userland" *-release*. Oui, cela veut dire que vous aurez besoin de mettre à jour votre système si vous voulez utiliser un nouveau programme qui a a été rajouté aujourd'hui à l'arborescence des ports. OpenBSD n'a malheureusement que des ressources limitées.

Il faut aussi comprendre que lors de la mise à jour des sources, le processus de mise à jour est uniquement supporté **dans une seule direction uniquement : de l'ancien au nouveau**, et de *-stable* vers *-current*. Vous ne pouvez pas utiliser *3.4-current* (ou un snapshot), puis décider que c'est trop dangereux, et revenir vers *3.4-stable*. Vous ne pouvez compter que sur vous-même si vous choisissez un autre chemin que celui, supporté, consistant à réinstaller votre système proprement. Vous ne devez espérer aucune aide de la part de l'équipe de développement OpenBSD.

Oui, cela veut dire que vous devez prendre le temps de réfléchir avant d'utiliser *-current*.

5.2 - Pourquoi aurais-je besoin d'un noyau sur mesure?

En réalité, vous n'en avez très probablement pas besoin.

Un noyau sur mesure est un noyau construit à partir d'un fichier de configuration autre que `GENERIC`, le fichier de configuration fourni de base Un noyau sur mesure peut se baser sur du code source *-release*, *-stable* or *-current* comme c'est le cas pour le noyau `GENERIC`. Alors que la compilation de votre propre noyau `GENERIC` est supportée par l'équipe OpenBSD, la compilation de votre propre noyau sur mesure *ne* l'est pas.

Le fichier de configuration noyau OpenBSD standard (`GENERIC`) est conçu pour convenir à la plupart des utilisateurs. Bon nombre de personnes ont rendu leur système inopérant en essayant d'optimiser le noyau au lieu d'améliorer son fonctionnement. Il existe certaines personnes qui pensent qu'un noyau et un système d'exploitation doivent être taillés sur mesure pour obtenir des performances optimales. Ceci n'est pas vrai dans le cas d'OpenBSD. Seules les personnes très compétentes avec des applications très particulières doivent penser à faire un noyau et un système sur mesure.

Voici quelques raisons pour lesquelles vous devriez créer un noyau sur mesure :

- Vous savez vraiment ce que vous faites, et vous souhaitez utiliser OpenBSD sur une machine disposant de peu de ressources mémoire en supprimant tous les pilotes de périphériques dont vous n'avez pas besoin.
- Vous savez vraiment ce que vous faites, et vous souhaitez supprimer des options par défaut ou ajouter des options qui ne sont pas activées par défaut (et vous avez vraiment une bonne raison pour le faire).
- Vous savez vraiment ce que vous faites, et vous souhaitez activer des options expérimentales.
- Vous savez vraiment ce que vous faites, et vous avez un besoin spécifique auquel le noyau `GENERIC` ne répond pas. Si quelque chose ne marche pas comme prévu, vous n'allez pas demander à autrui le pourquoi du comment.

Voici quelques raisons pour lesquelles vous ne devez pas compiler un noyau sur mesure :

- Vous n'en avez pas besoin en temps normal.
- Votre système n'en sera pas plus rapide.
- Vous rendrez probablement votre machine moins fiable.
- Vous n'obtiendrez aucune aide de la part des développeurs.

- Tout problème rencontré devra être obligatoirement reproduit avec un noyau `GENERIC` avant que les développeurs ne le prennent au sérieux.
- Les autres utilisateurs et les développeurs vous riront au nez si vous cassez votre système.
- D'habitude, des options de compilation sur mesure exposent les problèmes de compilateur au lieu d'améliorer les performances du système.

La suppression de pilotes pourrait rendre plus rapide la phase de démarrage système. Cependant, elle peut compliquer la récupération suite à problème matériel. La suppression de pilotes est une tâche très souvent mal réalisée. La suppression de pilotes *ne rendra pas* votre système plus rapide de manière perceptible même si elle peut produire un noyau plus petit. La suppression des parties liées au déboguage et à la vérification d'erreurs peut améliorer les performances, mais rendra impossible l'analyse du système si quelque chose ne fonctionne plus ou pas.

Encore une fois, les développeurs ignorent d'habitude les rapports de bogue relatifs à des noyaux personnalisés, sauf si le problème peut être reproduit avec un noyau `GENERIC`. Vous aurez été prévenu.

5.3 - Options de configuration du noyau

La création d'un noyau OpenBSD est contrôlée par le biais de fichiers de configuration, se trouvant dans le répertoire `/usr/src/sys/arch/<arch>/conf/` par défaut. Toutes les architectures possèdent un fichier, `GENERIC`, qui peut être utilisé pour générer un noyau OpenBSD standard pour une plate-forme donnée. Il peut aussi y avoir d'autres fichiers de configuration qui peuvent être utilisés pour créer des noyaux avec des objectifs différents tels que la minimisation de l'utilisation de la RAM, les stations de travail "diskless", etc.

Le fichier de configuration est traité par config(8), qui crée et peuple un répertoire de compilation situé sous `../compile`. Pour une installation typique, le chemin absolu du répertoire serait situé sous `/usr/src/sys/arch/<arch>/compile/`. config(8) peut aussi créer un fichier Makefile, et d'autres fichiers requis pour créer avec succès un noyau.

Les options de configuration du noyau sont des options que vous ajoutez à la configuration de votre noyau pour activer certaines caractéristiques dans celui-ci. Ceci vous permet d'avoir exactement le support que vous voulez sans vous encombrer des pilotes inutiles. Il y a une multitude d'options qui vous permettront de personnaliser votre noyau. Veuillez consulter la page du manuel options(4) pour une liste complète des options. Vous pouvez aussi consulter les fichiers d'exemples de configurations qui sont disponibles pour votre architecture.

L'ajout, la suppression, ou la modification d'options dans votre noyau ne doivent être effectués que si vous avez une bonne raison pour le faire ! La seule configuration du noyau supportée par l'équipe OpenBSD est le noyau GENERIC, la combinaison d'options figurant dans les fichiers `/usr/src/sys/arch/<arch>/conf/GENERIC` et `/usr/src/sys/conf/GENERIC` *tels que livrés par l'équipe OpenBSD* (i.e. non édités). Emettre un rapport de bogues concernant un noyau personnalisé va dans la plupart des cas se résumer à une retour vous demandant d'essayer de reproduire le problème avec un noyau GENERIC. Les options ne sont pas toutes compatibles entre elles, et plusieurs options sont nécessaires au bon fonctionnement du système. Il n'y a aucune garantie quant au fonctionnement d'un noyau personnalisé que vous avez réussi à compiler.

Vous pouvez voir les fichiers de configuration spécifiques à une plate-forme donnée ici :

- Fichiers de Configuration du Noyau alpha
- Fichiers de Configuration du Noyau i386
- Fichiers de Configuration du Noyau macppc
- Fichiers de Configuration du Noyau sparc
- Fichiers de Configuration du Noyau sparc64
- Fichiers de Configuration du Noyau vax
- Fichiers de Configuration du Noyau hppa
- Other Arch's

Si vous lisez attentivement ces fichiers vous verrez une ligne du genre :

```
include "../../../conf/GENERIC"
```

Cela signifie que l'on fait référence à un autre fichier de configuration. Ce fichier comprend toutes les options qui ne sont pas dépendantes de l'architecture. Donc quand vous créez votre fichier de configuration, soyez sûr de regarder /sys/conf/GENERIC pour voir ce que vous voulez.

Toutes les options ci-dessous doivent être placées dans le fichier de configuration du noyau avec le format :

```
option      nom
```

Par exemple, pour utiliser l'option "DEBUG" dans le noyau, il faut mettre la ligne suivante :

```
option      DEBUG
```

Les options dans le noyau OpenBSD sont traduites en tant qu'options du préprocesseur, donc une option telle que DEBUG compilerait les sources avec l'option -DDEBUG. Ce qui est équivalent à placer un `#define DEBUG` à travers les sources du noyau.

Quelques fois, vous aurez peut-être besoin de désactiver une option précedemment définie dans le fichier "`src/sys/conf/GENERIC`" typiquement. Bien entendu, vous pouvez modifier une copie de ce fichier, mais une meilleure méthode consiste à utiliser la clause *rmoption*. Par exemple, si vous souhaitiez vraiment désactiver le débogueur intégré au noyau (*non recommandé !*), vous ajouteriez la ligne suivante :

```
rmoption DDB
```

à votre fichier de configuration du noyau. `option DDB` est définie dans `src/sys/conf/GENERIC`, mais la ligne `rmoption` ci-dessus la désactive.

Encore une fois, veuillez consulter options(4) pour plus d'informations concernant les spécificités de ces options. Il est à noter que plusieurs options possèdent leurs propres pages de manuel -- il faut toujours lire toutes les informations disponibles au sujet d'une option avant de l'ajouter ou la supprimer de votre noyau.

5.4 - Construire votre propre noyau

Toutes les instructions pour créer votre noyau sur mesure sont dans la page du manuel afterboot(8).

Pour compiler votre noyau depuis le cdrom vous aurez tout d'abord besoin d'avoir le code source. Ce dernier est disponible à partir du CD officiel (disque 3) et à partir des sites FTP. Cet exemple suppose que le CD3 est monté sur /mnt :

```
# cd /usr/src
# tar xvzf /mnt/src.tar.gz
```

Remarque : Si vous effectuez un téléchargement à partir des serveurs FTP, vous trouverez DEUX fichiers : *src.tar.gz* et *sys.tar.gz*. Le premier correspond à "userland" -- tout excepté le noyau, le second est le code source du noyau. Téléchargez et effectuez une extraction des deux comme expliqué ci-dessus vu que pour la plupart des utilisations, vous aurez besoin des deux. Sur le CD-ROM, ils sont combinés dans un seul fichier.

Maintenant, pour créer votre noyau personnalisé il est plus facile de partir du noyau GENERIC. Ce dernier est situé sous `/usr/src/sys/arch/$ARCH/conf/GENERIC`, où $ARCH est votre architecture. Il y a d'autres configurations en exemples dans le répertoire. Voici deux exemples vous montrant comment compiler votre noyau. Le premier exemple permet de compiler votre noyau avec des sources en lecture seule. Le second correspond à des sources accessibles en écriture.

```
# cd /somewhere
# cp /usr/src/sys/arch/$ARCH/conf/SOMEFILE .
# vi SOMEFILE   (pour effectuer les changements voulus)
# config -s /usr/src/sys -b . SOMEFILE
```

suivi par :

```
    # make depend
         - OU -
    # make clean
# make
```

Vous devez exécuter '`make depend`' lorsque vous effectuez n'importe quelle modification à votre arborescence des sources (y compris des mises à jour et des correctifs) (en d'autres termes, pratiquement toujours, sauf si vous avez besoin d'exécuter '`make clean`').

Si vous avez effectué des modifications aux options de configuration de votre noyau, et/ou effectué des modifications majeures à votre arborescence des sources, vous devriez utiliser '`make clean`' au lieu de '`made depend`'. Il est à noter qu'il est toujours de bon ton de faire un 'make clean', même si ça peut aboutir à des temps de compilation plus longs, vu que plus de choses ont besoin d'être reconstruites.

Pour compiler un noyau avec des sources accessibles en écriture il faut effectuer les actions suivantes :

```
# cd sys/arch/$ARCH/conf
# vi SOMEFILE (pour effectuer tous les changements voulus)
# config SOMEFILE (plus de détails sur cette # opération sont disponibles
ici : config(8))
# cd ../compile/SOMEFILE
# make
```

Où $ARCH est l'architecture que vous utilisez (par exemple i386). Vous pouvez aussi faire un **make depend** pour créer les dépendances pour la prochaine compilation de votre noyau.

Pour déplacer votre noyau à sa destination.

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

Pour démarrer avec votre ancien noyau au démarrage, il suffit juste de faire ceci

```
boot> bsd.old
```

votre ancien noyau sera alors chargé à la place de /bsd.

Parfois quand vous installez un nouveau noyau, il vous faudra installer de nouveaux blocs de démarrage. Pour ce faire, lisez FAQ 14, Installation des Bootblocks, qui vous donnera toutes les informations sur le bootloader OpenBSD.

5.5 - Configuration au démarrage

Parfois lorsque vous démarrez votre système vous remarquerez que votre noyau trouve vos périphériques mais à la mauvaise IRQ. Et avez-vous immédiatement besoin de ce périphérique. Sans recompiler votre noyau vous pouvez utiliser la configuration au démarrage de celui-ci. Cela permettra de corriger votre problème pour cette fois et uniquement pour cette fois. Si vous redémarrez le système il faudra recommencer la procédure. Il s'agit donc d'une méthode temporaire. Le problème devra être corrigé en recompilant votre noyau. De plus votre noyau à besoin de l'**option BOOT_CONFIG** dans la configuration du noyau. Le noyau GENERIC possède cette option.

Une grande partie de ce document peut-être trouvé dans la page du manuel boot_config(8).

Pour démarrer avec UKC (User Kernel Config), il faut spécifier l'option -c au démarrage.

```
boot> boot hd0a:/bsd -c
```

Où n'importe quel autre noyau que vous voulez démarrer. L'invite de commandes UKC apparaîtra. De là vous pourrez spécifier au noyau les périphériques que vous désirez modifier, ceux que vous voulez activer ou désactiver.

Voici un liste des commandes les plus utilisées dans UKC.

- `add` **`device`** - Ajoute un périphérique en en copiant un autre
- `change` **`devno | device`** - Modifie un ou plusieurs périphériques
- `disable` **`devno | device`** - Désactive un ou plusieurs périphériques
- `enable` **`devno | device`** - Active un ou plusieurs périphériques
- `find` **`devno | device`** - Trouver un ou plusieurs périphériques
- `help` - Rapide sommaire de ces commandes
- `list` - Liste TOUS les périphériques connus
- `exit/quit` - Continue le démarrage
- `show` **`[attr [val]]`** - Montre les périphériques avec un attribut et une valeur spécifiée optionnelle

Une fois que votre périphérique est configuré, utilisez quit ou exit pour continuer à démarrer. Après avoir fait ceci, vous devriez corriger la configuration de votre noyau et en compiler un nouveau. Voir Construire votre propre noyau pour plus de renseignements.

5.6 - Avoir plus d'informations pendant le démarrage

Obtenir plus d'informations pendant le démarrage peut-être très pratique lorsque vous essayez de régler certains problèmes lors de celui-ci. Si vous avez un problème et que vous aimeriez avoir plus d'informations, redémarrez votre machine. Quand vous obtenez l'invite "boot>quot; démarrez avec l'option -c. L'invite UKC> apparaîtra et ensuite :

```
UKC> verbose
autoconf verbose enabled
UKC> quit
```

Vous obtiendrez un démarrage extrêmement verbeux.

5.7 - Utilisation de config(8) pour changer le binaire du noyau

Les options **-e** et **-u** de config(8) peuvent être très utiles et vous évitent de perdre du temps à recompiler votre noyau. Le drapeau **-e**

vous permet de rentrer en configuration UKC alors que le système fonctionne. Les changements prendront effets au prochain redémarrage. Le drapeau **-u** permet de voir si des changements ont été effectués au noyau pendant le démarrage, signifiant que vous avez utilisé **boot -c** pour entrer en configuration UKC.

Les exemples suivants montrent la désactivation des périphériques ep* dans le noyau. Pour plus de sécurité, il est préférable d'utiliser l'option **-o** qui écrira les changements dans un fichier spécifié. Par exemple : **config -e -o bsd.new /bsd** écrira les changements dans bsd.new. Les exemples n'utilisent pas l'option **-o**, mais les changements sont ignorés et non écrit dans le binaire du noyau. Pour plus d'informations sur les messages d'erreur et d'avertissement, lire la page du manuel config(8) .

```
$ sudo config -e /bsd
OpenBSD 3.4 (GENERIC) #18: Wed Sep 17 03:34:47 MST 2003
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/GENERIC
warning: no output file specified
Enter 'help' for information
ukc> ?
        help                            Command help list
        add         dev                 Add a device
        base        8|10|16             Base on large numbers
        change      devno|dev           Change device
        disable     attr val|devno|dev  Disable device
        enable      attr val|devno|dev  Enable device
        find        devno|dev           Find device
        list                            List configuration
        lines       count               # of lines per page
        show        [attr [val]]        Show attribute
        exit                            Exit, without saving changes
        quit                            Quit, saving current changes
        timezone    [mins [dst]]        Show/change timezone
        nmbclust    [number]            Show/change NMBCLUSTERS
        cachepct    [number]            Show/change BUFCACHEPERCENT
        nkmempg     [number]            Show/change NKMEMPAGES
        shmseg      [number]            Show/change SHMSEG
        shmmaxpgs   [number]            Show/change SHMMAXPGS
ukc> list
  0 audio* at sb0|sb*|gus0|pas0|sp0|ess*|wss0|wss*|ym*|eap*|eso*|sv*|neo*|cmpci*
|clcs*|clct*|auich*|autri*|auvia*|fms*|uaudio*|maestro*|esa*|yds*|emu* flags 0x0
  1 midi* at sb0|sb*|opl*|opl*|opl*|opl*|ym*|mpu*|autri* flags 0x0
  2 nsphy* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*
|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|e
p*|ep* phy -1 flags 0x0
  3 nsphyter* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|
vr*|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep
*|ep*|ep* phy -1 flags 0x0
  4 qsphy* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*
|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|e
p*|ep* phy -1 flags 0x0
  5 inphy* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*
|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|e
p*|ep* phy -1 flags 0x0
  6 iophy* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*
|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|e
p*|ep* phy -1 flags 0x0
  7 eephy* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*
|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|e
p*|ep* phy -1 flags 0x0
  8 exphy* at aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*
|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|e
p*|ep* phy -1 flags 0x0
[...snip...]
```

```
ukc> disable ep
 67 ep0 disabled
 68 ep* disabled
 69 ep* disabled
155 ep0 disabled
156 ep0 disabled
157 ep* disabled
158 ep* disabled
210 ep* disabled
ukc> quit
not forced
```

Dans l'exemple ci-dessus, tous les périphériques ep* sont désactivés du noyau et ne seront donc pas testés. Dans certaines situations où vous aurez effectué ces changements au démarrage avec UKC et **boot -c**, il vous faudra les rendre définitifs. Pour ce faire, il faudra utiliser l'option **-u**. Dans l'exemple suivant, l'ordinateur a été démarré avec UKC et les périphériques wi(4) sont désactivés. Ètant donné que les changements fait par boot -c ne sont pas permanents, ceux-ci doivent être écrits sur le disque. Cet exemple montre comment écrire les changements effectués par boot -c dans un nouveau binaire noyau bsd.new.

```
$ sudo config -e -u -o bsd.new /bsd
OpenBSD 3.3 (GENERIC) #44: Sat Mar 29 13:22:05 MST 2003
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/GENERIC
Processing history...
105 wi* disabled
106 wi* disabled
Enter 'help' for information
ukc> quit
```

5.8 - Problèmes Usuels de Compilation

5.8.1 - La compilation s'est arrêtée avec une erreur "Signal 11"

La compilation d'OpenBSD et d'autres programmes à partir des sources est une tâche qui sollicite le matériel plus que la plupart des autres opérations, faisant un usage intensif du CPU, du disque et de la mémoire. Par conséquence, si votre matériel a des problèmes, ces derniers apparaîtront plus facilement durant une compilation. Les défaillances "Signal 11" sont `typiquement` causées par des problèmes matériel, et la plupart du temps à cause de problèmes de mémoirei. Mais ces défaillances peuvent aussi causées par le CPU, la carte mère, ou des problèmes de surchauffe. Votre système peut d'ailleurs être stable la plupart du temps et connaître des défaillances lors de la compilation de programmes.

Il est recommandé de réparer ou remplacer les composants à l'origine des défaillances; les problèmes pouvant se manifester sous d'autres formes plus tard. Si vous avez du matériel que vous souhaitez utiliser et qui ne vous pose aucun problème, installez simplement une snapshot ou une "release".

Pour plus d'informations, consultez la Faq Sig11.

5.8.2 - "make build" échoue en générant un message "cannot open output file snake: is a directory"

Ceci est le résultat de deux erreurs séparées :

- **Vous n'avez pas proprement récupéré ou mis à jour votre arborescence CVS.** Lorsque vous effectuez une opération "CVS checkout", vous devez utiliser l'option "`-P`", et lorsque vous mettez à jour votre arborescence des sources à partir de CVS, vous devez utiliser les options "`-Pd`" de cvs(1), tel que c'est documenté dans le guide anoncvs , l' upgrade-

minifaq et la FAQ. Ces options s'assurent que les nouveaux répertoires sont ajoutés et supprimés de l'arborescence suivant l'évolution d'OpenBSD.

- **Vous n'avez pas correctement crée le répertoire `obj` avant de commencer la compilation.** La compilation de l'arborescence sans répertoire `/usr/obj` n'est pas supportée.

Il est important de suivre soigneusement les instructions lors de l'obtention et de la mise à jour de votre arborescence des sources et lors de la compilationde votre arborescence.

5.9 - Comment compiler une "release" OpenBSD

Après avoir compilé votre système, vous pouvez créer les ensembles de fichiers OpenBSD qui peuvent être utilisés pour installer votre système sur une autre machine.

La procédure nécessaire pour effectuer cette opération est détaillée dans la page de manuel release(8)

[Index de La FAQ] [Section 4 - Guide d'Installation] [Section 6 - Mise en place du réseau]


www@openbsd.org

Originally [OpenBSD: faq5.html,v 1.93]
$Translation: faq5.html,v 1.31 2004/03/23 21:06:13 saad Exp $
$OpenBSD: faq5.html,v 1.25 2004/03/26 08:40:41 jufi Exp $

OpenBSD

[FAQ Index] [To Section 5 - Building the System from Source [To Section 7 - Keyboard and Display Controls]

6 - Networking

Table of Contents

6.1 - Before we go any further

For the bulk of this document, it helps if you have read and at least partially understood 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, especially when you deal with or are responsible for more than one network.

If you are working with applications such as web servers, ftp servers, and mail servers, you may benefit greatly by reading the RFCs. Most likely, you can't read all of them. Pick some topics that you are interested in, or that you use in your network environment. 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.

6.2 - Initial Network Setup

6.2.1 - Identifying and Setting Up Your Network Interfaces

To start off, you must first identify your network interface. In OpenBSD, interfaces are named for the type of card, not for the type of connection. You can see your network card get initialized during the booting process, or after the booting process using the dmesg(8) command. You also have the chance of seeing your network interface using the ifconfig(8) command. For example, here is the output of dmesg for a Intel Fast Ethernet network card, which uses the device name fxp.

```
fxp0 at pci0 dev 10 function 0 "Intel 82557" rev 0x0c: irq 5, address 00:02:b3:2b:10:f7
inphy0 at fxp0 phy 1: i82555 10/100 media interface, rev. 4
```

If you don't know what your device name is, please look at the supported hardware list for your platform. You will find a list of many common card names and their OpenBSD device names here. Combine the short alphabetical device name (such as fxp) with a number assigned by the kernel and you have an interface name (such as fxp0).

You can find out what network interfaces have been identified by using the ifconfig(8) utility. The following command will show all network interfaces on a system. This sample output shows us only one physical ethernet interface, an fxp(4).

```
$ ifconfig -a
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33224
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
```

```
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x5
lo1: flags=8008<LOOPBACK,MULTICAST> mtu 33224
fxp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        address: 00:04:ac:dd:39:6a
        media: Ethernet autoselect (100baseTX full-duplex)
        status: active
        inet 10.0.0.38 netmask 0xffffff00 broadcast 10.0.0.255
        inet6 fe80::204:acff:fedd:396a%fxp0 prefixlen 64 scopeid 0x1
pflog0: flags=0<> mtu 33224
pfsync0: flags=0<> mtu 2020
sl0: flags=c010<POINTOPOINT,LINK2,MULTICAST> mtu 296
sl1: flags=c010<POINTOPOINT,LINK2,MULTICAST> mtu 296
ppp0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
ppp1: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
tun0: flags=10<POINTOPOINT> mtu 3000
tun1: flags=10<POINTOPOINT> mtu 3000
enc0: flags=0<> mtu 1536
bridge0: flags=0<> mtu 1500
bridge1: flags=0<> mtu 1500
vlan0: flags=0<> mtu 1500
        address: 00:00:00:00:00:00
vlan1: flags=0<> mtu 1500
        address: 00:00:00:00:00:00
gre0: flags=9010<POINTOPOINT,LINK0,MULTICAST> mtu 1450
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif1: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif2: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif3: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
```

As you can see here, ifconfig(8) gives us a lot more information than we need at this point. But, it still allows us to see our interface. In the above example, the interface card is already configured. This is obvious because an IP network is already configured on fxp0, hence the values "inet 10.0.0.38 netmask 0xffffff00 broadcast 10.0.0.255". Also, the **UP** and **RUNNING** flags are set.

Finally, you will notice several other interfaces come enabled by default. These are virtual interfaces that serve various functions. The following manual pages describe them:

- lo - Loopback Interface
- pflog - Packet Filter Logging Interface
- sl - SLIP Network Interface
- ppp - Point to Point Protocol
- tun - Tunnel Network Interface
- enc - Encapsulating Interface
- bridge - Ethernet Bridge Interface
- vlan - IEEE 802.1Q Encapsulation Interface
- gre - GRE/MobileIP Encapsulation Interface
- gif - Generic IPv4/IPv6 Tunnel Interface

If you don't have your interface configured, the first step is to create the */etc/hostname.xxx* file, where the name of your interface will take the place of xxx. From the information in the examples above, the name would be */etc/hostname.fxp0*. The layout of this file is simple:

```
address_family address netmask broadcast [other options]
```

(Much more detail about the format of this file can be found in the hostname.if(5) man page.)

A typical interface configuration file, configured for an IPv4 address, would look like this:

```
$ cat /etc/hostname.fxp0
inet 10.0.0.38 255.255.255.0 NONE
```

You could also specify media types for Ethernet, say, if you wanted to force 100baseTX full-duplex mode.

```
inet 10.0.0.38 255.255.255.0 NONE media 100baseTX mediaopt full-duplex
```

(Of course, you should never force full duplex mode unless both sides of the connection are set to do this! In the absence of special needs, media settings should be excluded.)

Or, you may want to use special flags specific to a certain interface. The format of the hostname file doesn't change much!

```
$ cat /etc/hostname.vlan0
inet 172.21.0.0 255.255.255.0 NONE vlan 2 vlandev fxp1
```

The next step from here is to setup your default gateway. To do this, simply put the IP of your gateway in the file */etc/mygate*. This will allow for your gateway to be set upon boot. From here you should setup your nameservers, and your */etc/hosts* file (see the hosts(5) man page). To setup your nameservers, you will create a file called */etc/resolv.conf*. You can read more about the format of this file in the resolv.conf(5) man page. But for a standard usage, here is an example. In this example your domain servers are 125.2.3.4 and 125.2.3.5. You also belong in the domain "example.com".

```
$ cat /etc/resolv.conf
search example.com
nameserver 125.2.3.4
nameserver 125.2.3.5
lookup file bind
```

From here, you can either reboot or run the **/etc/netstart** script. You can do this by simply typing (as root):

```
# sh /etc/netstart
writing to routing socket: File exists
add net 127: gateway 127.0.0.1: File exists
writing to routing socket: File exists
add net 224.0.0.0: gateway 127.0.0.1: File exists
```

Notice that a few errors were produced. By running this script, you are reconfiguring things which are already configured. As such, some routes already exist in the kernel routing table. From here your system should be up and running. Again, you can check to make sure that your interface was setup correctly with ifconfig(8). You can also check your routes via netstat(1) or route(8). If you are having routing problems, you may want to use the -n flag to route(8) which prints the IP addresses rather than doing a DNS lookup and displaying the hostname. Here is an example of viewing your routing tables using both programs.

```
$ netstat -rn
Routing tables

Internet:
Destination        Gateway            Flags     Refs     Use    Mtu  Interface
default            10.0.0.1           UGS          0      86      -  fxp0
127/8              127.0.0.1          UGRS         0       0      -  lo0
127.0.0.1          127.0.0.1          UH           0       0      -  lo0
10.0.0/24          link#1             UC           0       0      -  fxp0
10.0.0.1           aa:0:4:0:81:d      UHL          1       0      -  fxp0
10.0.0.38          127.0.0.1          UGHS         0       0      -  lo0
224/4              127.0.0.1          URS          0       0      -  lo0

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

$ route show
Routing tables

Internet:
Destination      Gateway            Flags
default          10.0.0.1           UG
127.0.0.0        LOCALHOST          UG
localhost        LOCALHOST          UH
10.0.0.0         link#1             U
10.0.0.1         aa:0:4:0:81:d      UH
10.0.0.38        LOCALHOST          UGH
BASE-ADDRESS.MCA LOCALHOST          U
```

6.2.2 - Setting up your OpenBSD box as a Gateway

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 Packet 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 for information on conserving your IP address space.

The GENERIC kernel already has the ability to allow IP Forwarding, but needs to be turned on. You should do this using the sysctl(8) utility. To change this permanently you should edit the file /etc/sysctl.conf to allow for IP Forwarding. To do so add this line in that configuration file.

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

To make this change without rebooting you would use the sysctl(8) utility directly. Remember though that this change will no longer exist after a reboot, and needs to be run as root.

```
# 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 zebra, gated and mrtd. OpenBSD supports several T1, HSSI, ATM, FDDI, Ethernet, and serial (PPP/SLIP) interfaces.

6.2.3 - Setting up aliases on an interface

OpenBSD has a simple mechanism for setting up ip aliases on an interface. To do this simply edit the file */etc/hostname.<if>*. This file is read upon boot by the /etc/rc(8) script, which is part of the rc startup hierarchy. For the example, we assume that the user has an interface **dc0** and is on the network 192.168.0.0. Other important information:

- IP for dc0 is 192.168.0.2
- NETMASK is 255.255.255.0

A few side notes about aliases. In OpenBSD you use the interface name only. There is no difference between the first alias and the second alias. Unlike some other operating systems, OpenBSD doesn't refer to them as dc0:0, dc0:1. If you are referring to a specific aliased IP address with ifconfig, or adding an alias, be sure to say "`ifconfig int alias`" instead of just "`ifconfig int`" at the command line. You can delete aliases with "`ifconfig int delete`".

Assuming you are using multiple IP addresses which are in the same IP subnet with aliases, your netmask setting for each alias becomes 255.255.255.255. They do not need to follow the netmask of the first IP bound to the interface. In this example, */etc/hostname.dc0*, two aliases are added to the device dc0, which, by the way, was configured as 192.168.0.2 netmask 255.255.255.0.

```
# cat /etc/hostname.dc0
inet 192.168.0.2 255.255.255.0 media 100baseTX
inet alias 192.168.0.3 255.255.255.255
inet alias 192.168.0.4 255.255.255.255
```

Once you've made this file, it just takes a reboot for it to take effect. You can, however, bring up the aliases by hand using the ifconfig(8) utility. To bring up the first alias you would use the command:

```
# ifconfig dc0 inet alias 192.168.0.3 netmask 255.255.255.255
```

To view these aliases you must use the command:

```
$ ifconfig -A
dc0: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST>
        media: Ethernet manual
        inet 192.168.0.2  netmask 0xffffff00 broadcast 192.168.0.255
        inet 192.168.0.3 netmask 0xffffffff broadcast 192.168.0.3
```

6.3 - How do I filter and firewall with OpenBSD?

Packet Filter (from here on referred to as PF) is OpenBSD's system for filtering TCP/IP traffic and doing Network Address Translation. PF is also capable of normalizing and conditioning TCP/IP traffic and providing bandwidth control and packet prioritization, and can be used to create powerful and flexible firewalls. It is described in the **PF User's Guide**.

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 'request' 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 "example.com";
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 example.com (so, if the user types in 'telnet joe' then it will send them to joe.example.com). 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(8) from the command line, after editing /etc/dhcpd.conf, try:

```
# touch /var/db/dhcpd.leases
# dhcpd -q fxp0
```

The `touch` line is needed to create an empty `dhcpd.leases` file before dhcpd(8) can start. The OpenBSD startup scripts will create this file if needed on boot, but if you are starting dhcpd(8) manually, you must create it first. `fxp0` is an interface that you want to start serving DHCP on. The `-q` flag makes dhcpd(8) quiet; otherwise it is very noisy.

If you are serving DHCP to a Windows box, you may want dhcpd(8) 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 you're 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 probably 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 referring 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 idle timeout for the entire connection duration to 120 seconds. The "set ifaddr" line is a little tricky. Here is a more extensive explanation.

```
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 a 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) extras

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

Extended information can be found at FreeBSD Handbook entry on User PPP.

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 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
```

6.7 - Simple NFS usage

NFS, or Network File System, is used to share a filesystem over the network. A few choice man pages to read before trying to setup a NFS server are:

- nfsd(8)
- mountd(8)
- exports(5)

This section will go through the steps for a simple setup of NFS. This example details a server on a LAN, with clients accessing NFS on the LAN. It does not talk about securing NFS. We presume you have already setup packet filtering or other firewalling protection, to prevent outside access. If you are allowing outside access to your NFS server, and you have any kind of sensitive data stored on it, we strongly recommend that you employ IPsec. Otherwise, people can potentially see your NFS traffic. Someone could also pretend to be the IP address which you are allowing into your NFS server. There are several attacks that can result. When properly configured, IPsec protects against these types of attacks.

Another important security note. Don't just add a filesystem to */etc/exports* without some kind of list of allowed host(s). Without a list of hosts which can mount a particular directory, anyone on who can reach your host will be able to mount your NFS exports.

NFS depends upon portmap(8) to be running before it will operate. Portmap(8) is now off by default on OpenBSD 3.2 and later, so you must enable it in rc.conf(8) by changing the `portmap` line to read:

```
portmap=YES
```

and reboot to make it take effect.

The setup consists of a server with the ip **10.0.0.1**. This server will be serving NFS only to clients within that network. The first step to setting up NFS is to setup your */etc/exports* file. This file lists which filesystems you wish to have accessible via NFS and defines who is able to access them. There are many options that you can use in your */etc/exports* file, and it is best that you read the exports(5) man page. For this example we have an */etc/exports* that looks like this:

```
#
# NFS exports Database
# See exports(5) for more information.  Be very careful, misconfiguration
# of this file can result in your filesystems being readable by the world.
/work -alldirs -ro -network 10.0.0 -mask 255.255.255.0
```

This means that the local filesystem `/work` will be made available via NFS. `-alldirs` specifies that clients will be able to mount at any point under the `/work` mount point. `-ro` specifies that it will only be allowed to be mounted read-only. The last two arguments specify that only clients within the 10.0.0.0 network using a netmask of 255.255.255.0 will be authorized to mount this filesystem. This is important for some servers that are accessible by different networks.

Once your */etc/exports* file is setup, you can go ahead and setup your NFS server. You should first make sure that options NFSSERVER & NFSCLIENT are in your kernel configuration. (GENERIC kernel has these options included.) Next, you should set `nfs_server=YES` in */etc/rc.conf*. This will bring up both nfsd(8) and mountd(8) when you reboot. Now, you can go ahead and start the daemons yourself. These daemons need to be started as root, and you need to make sure that portmap(8) is running on your system. Here is an example of starting nfsd(8) which serves on both TCP and UDP using 4 daemons. You should set an appropriate number of NFS server daemons to handle the maximum number of concurrent client requests that you want to service.

```
# /sbin/nfsd -tun 4
```

Not only do you have to start the nfsd(8) server, but you need to start mountd(8). This is the daemon that actually services the mount requests on NFS. To start mountd(8), make sure an empty mountdtab file exists, and run the daemon:

```
# echo -n >/var/db/mountdtab
# /sbin/mountd
```

If you make changes to /etc/exports while NFS is already running, you need to make mountd aware of this! Just HUP it:

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

Checking Stats on NFS

From here, you can check to make sure that all these daemons are up and registered with RPC. To do this, use rpcinfo(8).

```
$ rpcinfo -p 10.0.0.1
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp    633  mountd
    100005    3   udp    633  mountd
    100005    1   tcp    916  mountd
    100005    3   tcp    916  mountd
    100003    2   udp   2049  nfs
```

```
    100003    3   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
```

During normal usage, there are a few other utilities that allow you to see what is happening with NFS. One is showmount(8), which allows you to view what is currently mounted and who is mounting it. There is also nfsstat(8) which shows much more verbose statistics. To use showmount(8), try `/usr/bin/showmount -a host`. For example:

```
$ /usr/bin/showmount -a 10.0.0.1
All mount points on 10.0.0.1:
10.0.0.37:/work
```

Mounting NFS Filesystems

NFS filesystems should be mounted via mount(8), or more specifically, mount_nfs(8). To mount a filesystem */work* on host 10.0.0.1 to local filesystem */mnt*, do this (note that you don't need to use an IP address; mount will resolve host names):

```
# mount -t nfs 10.0.0.1:/work /mnt
```

To have your system mount upon boot, add something like this to your */etc/fstab*:

```
10.0.0.1:/work /mnt nfs rw 0 0
```

It is important that you use `0 0` at the end of this line so that your computer does not try to fsck the NFS filesystem on boot!!!! The other standard security options, such as noexec, nodev, and nosuid, should also be used where applicable. Such as:

```
10.0.0.1:/work /mnt nfs rw,nodev,nosuid 0 0
```

This way, no devices or setuid programs on the NFS server can subvert security measures on the NFS client. If you are not mounting programs which you expect to run on the NFS client, add noexec to this list.

6.8 - Setting up a PPTP connection in OpenBSD

NOTE: This does not apply to **ALL** ADSL providers, but much information can be gleaned from the setup here. This is known to work for Inode, an ADSL provider in Austria.

To start off, you need to install pptp. The port is located at */usr/ports/net/pptp*. Read FAQ 8, Ports for more information on the OpenBSD ports tree.

Because of a conflict between the In-Kernel gre(4) support and pptp, you will need to re-compile your kernel, removing support for gre(4).

Patch to remove GRE(4) support.

```
Index: GENERIC
===================================================================
RCS file: /cvs/src/sys/conf/GENERIC,v
retrieving revision 1.86
diff -u -r1.86 GENERIC
--- GENERIC     14 Mar 2002 00:42:25 -0000      1.86
+++ GENERIC     17 May 2002 01:52:17 -0000
@@ -87,7 +87,7 @@
 pseudo-device  enc     1       # option IPSEC needs the encapsulation interface
 pseudo-device  bridge  2       # network bridging support
 pseudo-device  vlan    2       # IEEE 802.1Q VLAN
-pseudo-device  gre     1       # GRE encapsulation interface
+#pseudo-device gre     1       # GRE encapsulation interface
 #pseudo-device strip   1       # Starmode Radio IP interface

 pseudo-device  pty     64      # pseudo-terminals
```

To recompile your kernel, check out OpenBSD source via cvs (refer to AnonCVS web page for more information), apply the following patch, and rebuild your kernel as per FAQ 5, Building a kernel.

After you have the **pptp** package installed and a new kernel, you need to edit a few files to setup for your connection. This packages uses the in-house OpenBSD

ppp(8), so if you are familiar with ppp(8), much of the setup is the same. Also, refer to FAQ 6, PPP.

- 1 - /etc/ppp/options
- 2 - /etc/ppp/pap-secrets

For the */etc/ppp/options* file, a setup like below will most likely do all that you need:

```
# cat /etc/ppp/options
name "LOGINNAME"
noauth
noipdefault
defaultroute
debug
```

`LOGINNAME` should be replaced with your User-ID.

The */etc/ppp/pap-secrets* a line like:

```
# cat /etc/ppp/pap-secrets
LOGINNAME 10.0.0.138 PASSWORD
```

Where LOGINNAME is your User-ID and PASSWORD is your password. 10.0.0.138 is the IP assigned to your MODEM in the case that you are using ADSL, etc. Make sure this file stays readonly by root (mode 600).

6.8.1 - Assigning an address to your Network Interface

In the above example, our modem came with a preconfigured interface of 10.0.0.138. We now need to assign an address to OUR interface. It's best to pick an IP close to the one given by your MODEM, or use the static IP assigned to you. Read more about setting up interfaces in FAQ 6, Setup.

Once your interface is setup, you should be able to create a pptp connection with the command:

```
# /usr/local/sbin/pptp 10.0.0.138 &
```

Since this uses the in-house OpenBSD ppp(8), two processes are started. You can kill pptp by killing both these processes:

```
# kill -9 [pid of pppd]
$ kill -9 [pid of pptp]
```

It is recommended to open `/var/log/messages` in a extra terminal window, to recognize possible problems.

```
# tail -f /var/log/messages
```

We also suggest that you put the startup command in */etc/rc.local* so that you automatically connect on reboot.

6.9 - Setting up a network bridge in OpenBSD

A bridge is a link between two or more separate networks. Unlike a router, packets transfer through the bridge "invisibly" -- logically, the two network segments appear to be one segment to nodes on either side of the bridge. The bridge will only forward packets that have to pass from one segment to the other, so among other things, they provide an easy way to reduce traffic in a complex network and yet allow any node to access any other node when needed.

Note that because of this "invisible" nature, an interface in a bridge may or may not have an IP address of its own. If it does, the interface has effectively two modes of operation, one as part of a bridge, the other as a normal, stand-alone NIC. If neither interface has an IP address, the bridge will pass network data, but will not be externally maintainable (which can be a feature).

An example of a bridge application

One of my computer racks has a number of older systems, none of which have a built-in 10BASE-TX NIC. While they all have an AUI or AAUI connector, my supply of transceivers is limited to coax. One of the machines on this rack is an OpenBSD-based terminal server which is always on and connected to the high-speed network. Adding a second NIC with a coax port will allow me to use this machine as a bridge to the coax network.

This system has two NICs in it now, an Intel EtherExpress/100 (`fxp0`) and a 3c590-Combo card (`ep0`) for the coax port. `fxp0` is the link to the rest of my network and

will thus have an IP address, `ep0` is going to be for bridging only and will have no IP address. Machines attached to the coax segment will communicate as if they were on the rest of my network. So, how do we make this happen?

The file `hostname.fxp0` contains the configuration info for the `fxp0` card. This machine is set up using DHCP, so its file looks like this:

```
$ cat /etc/hostname.fxp0
dhcp NONE NONE NONE NONE
```

No surprises here.

The `ep0` card is a bit different, as you might guess:

```
$ cat /etc/hostname.ep0
up media 10base2
```

Here, we are instructing the system to activate this interface using ifconfig(8) and set it to 10BASE-2 (coax). No IP address or similar information needs to be specified for this interface. The options the `ep` card accepts are detailed in its man page.

Now, we need to set up the bridge. Bridges are initialized by the existence of a file named something like `bridgename.bridge0`. Here is an example for my situation here:

```
$ cat /etc/bridgename.bridge0
add fxp0
add ep0
up
```

This is saying set up a bridge consisting of the two NICs, fxp0 and ep0, and activate it. Does it matter which order the cards are listed? No, remember a bridge is very symmetrical -- packets flow in and out in both directions.

That's it! Reboot, and you now have a functioning bridge.

Filtering on a bridge

While there are certainly uses for a simple bridge like this, it is likely you might want to DO something with the packets as they go through your bridge. As you might expect, Packet Filter can be used to restrict what traffic goes through your bridge.

Keep in mind, by the nature of a bridge, the same data flows through both interfaces, so you only need to filter on one interface. Your default "Pass all" statements would look something like this:

```
pass in  on ep0  all
pass out on ep0  all
pass in  on fxp0 all
pass out on fxp0 all
```

Now, let's say I wish to filter traffic hitting these old machines, I want only Web and SSH traffic to reach them. In this case, we are going to let all traffic in and out of the ep0 interface, but filter on the fxp0 interface, using keep state to handle the reply data:

```
# Pass all traffic through ep0
pass in quick on ep0 all
pass out quick on ep0 all

# Block fxp0 traffic
block in  on fxp0 all
block out on fxp0 all

pass in quick on fxp0 proto tcp from any to any port {22, 80} \
     flags S/SA keep state
```

Note that this rule set will prevent anything but incoming HTTP and SSH traffic from reaching either the bridge machine or any of the other nodes "behind" it. Other results could be had by filtering the other interface.

To monitor and control the bridge you have created, use the brconfig(8) command, which can also be used to create a bridge after boot.

Tips on bridging

- It is HIGHLY recommended that you filter on only one interface. While it is possible to filter on both, you really need to understand this very well to do it right.
- By using the *blocknonip* option of brconfig(8) or in bridgename.bridge0, you can prevent non-IP traffic (such as IPX or NETBEUI) from slipping around your filters. This may be important in some situations, but you should be aware that bridges work for all kinds of traffic, not just IP.
- Bridging requires that the NICs be in a "Promiscuous mode" -- they listen to ALL network traffic, not just that directed at the interface. This will put a higher load on the processor and bus than one might expect. Some NICs don't work properly in this mode, the TI ThunderLAN chip (tl(4)) is an example of a chip that won't work as part of a bridge.

[FAQ Index] [To Section 5 - Building the System from Source] [To Section 7 - Keyboard and Display Controls]


www@openbsd.org

$OpenBSD: faq6.html,v 1.184 2004/02/06 01:07:52 nick Exp $

OpenBSD

[FAQ Index] [To Section 6 - Networking] [To Section 8 - General Questions]

7 - Keyboard and Display Controls

Table of Contents

7.1 - How do I remap the keyboard? *(wscons)*

The ports that use the wscons(4) console driver: alpha, hppa, i386, macppc, sparc, and sparc64.

With wscons(4) consoles, most options can be controlled using the wsconsctl(8) utility. For example, to change keymappings with wsconsctl(8) one would execute the following:

```
# wsconsctl -w keyboard.encoding=uk
```

In the next example, we will remap "Caps Lock" to be "Control L":

```
# wsconsctl -w keyboard.map+="keysym Caps_Lock = Control_L"
```

7.2 - Is there gpm or the like in OpenBSD?

For the alpha and i386 platforms, OpenBSD provides wsmoused(8), a port of FreeBSD's moused(8). It can be enabled automatically at startup by editing the appropriate line in rc.conf(8).

7.3 - Clearing the console each time a user logs out.

To do this you must add a line in `/etc/gettytab(5)`. Change the current section:

```
P|Pc|Pc console:\
        :np:sp#9600:
```

adding the line "`:cl=\E[H\E[2J:`" at the end, so that it ends up looking like this:

```
P|Pc|Pc console:\
        :np:sp#9600:\
        :cl=\E[H\E[2J:
```

7.4 - Accessing the Console Scrollback Buffer *(i386, some Alpha)*

On some platforms, OpenBSD provides a console scrollback buffer. This allows you to see information that has already scrolled past your screen. To move up and down in the buffer, simply use the key combinations `[SHIFT]+[PGUP]` and `[SHIFT]+[PGDN]`.

The default scrollback buffer, or the number of pages that you can move up and view, is 8. This is a feature of the vga(4) driver, so it will not work without a VGA card on any platform (many Alpha systems have TGA video).

7.5 - How do I switch consoles? *(i386, some Alpha)*

On i386 and Alpha systems with vga(4) cards, OpenBSD provides six virtual terminals by default, /dev/ttyC0 through /dev/ttyC5. ttyC4 is reserved for use by the X Window system, leaving five text consoles. You can switch between them using `[CTRL]+[ALT]+[F1]`, `[CTRL]+[ALT]+[F2]`, `[CTRL]+[ALT]+[F3]`, `[CTRL]+[ALT]+[F4]` and `[CTRL]+[ALT]+[F6]`.

The X environment uses ttyC4, `[CTRL]+[ALT]+[F5]`. When using X, the `[CTRL]+[ALT]+[F`*n*`]` keys will take you to the text screens; `[CTRL]+[ALT]+[F5]` will take you back to the graphical environment.

If you wish to have more than the default number of virtual consoles, use the wsconscfg(8) command to create screens for ttyC6, ttyC7 and above. For example:

```
wsconscfg -t 80x25 6
```

will create a virtual terminal for ttyC6, accessed by `[CTRL]+[ALT]+[F7]`. Don't forget to add this command to your `rc.local(8)` file if you want the extra screen the next time you boot the computer.

Note that you will not get a "`login:`" prompt on the newly-created virtual console unless you set it to "on" in `/etc/ttys(5)`, and either reboot or send init(8) a HUP signal using kill(1).

7.6 - How do I use a console resolution of 80x50? *(i386)*

i386 users normally get a console screen of 25 lines of 80 characters. However, many VGA video cards are capable of displaying a higher text resolution of 50 lines of 80 characters.

First, a font that supports the desired resolution must be loaded using the wsfontload(8) command. The standard 80x25 text screen

uses 8x16 pixel fonts; to double the number of lines we will have to use 8x8 pixel fonts.

After that, we will have to delete and recreate a virtual console at the desired screen resolution, using the wsconscfg(8) command.

This can be done automatically at boot by adding the following lines to the end of your `rc.local(8)` file:

```
wsfontload -h 8 -e ibm /usr/share/misc/pcvtfonts/vt220l.808
wsconscfg -dF 5
wsconscfg -t 80x50 5
```

As with any modification to your system configuration, it is recommended you spend some time with the man pages to understand what these commands do.

The first line above loads the 8x8 font. The second line deletes screen 5 (which would be accessed by `[CTRL]+[ALT]+[F6]`). The third line creates a new screen 5 with 50 lines of 80 characters each. If you do this, you will see your primary screen, and the other three default virtual consoles, come up in the standard 80x25 mode, but a new screen 5 at 80x50 accessible through `[CTRL]+[ALT]+[F6]`.

Remember that `[CTRL]+[ALT]+[F1]` is screen 0 (ttyC0). If you wish to alter other screens, simply repeat the delete and add screen steps for whichever screens you want running at the 80x50 resolution.

You should avoid changing screen 4 (ttyC4, `[CTRL]+[ALT]+[F5]`), which is used by X as a graphical screen. It is also not possible to change the resolution of the primary console device (i.e., ttyC0).

As one might expect, all these commands can also be entered at the command prompt, as root, or (better) using sudo(8).

Note: this will not work on all video cards. Unfortunately, not all video cards support the uploaded fonts that wscons(4) requires to achieve the 80x50 text mode. In these cases, you might wish to consider running X.

7.7 - How do I use a serial console?

There are many reasons you may wish to use a serial console for your OpenBSD system:

- Recording console output (for documentation).
- Remote management.
- Easier maintenance of a large quantity of machines
- Providing a useful dmesg from machines which might otherwise be difficult to get one from.
- Providing an accurate "trace" and "ps" output if your system crashes so developers can have a chance to fix the problem.

OpenBSD supports serial console on most platforms, however details vary greatly between platforms.

Note that serial interfacing is NOT a trivial task -- you will often need unusual cables, and ports are not standardized between machines, in some cases, not even consistent on one machine. It is assumed you know how to select the appropriate cable to go between your computer and the device acting as your serial terminal. A full tutorial on serial interfacing is beyond the scope of this article, however, we offer one hint: just because the ends plug in doesn't mean it will work.

/etc/ttys change

There are two parts to getting a functional serial console on an OpenBSD system. First, you must have OpenBSD to use your serial

port as a console for status and single user mode. This part is very platform dependent. Second, you must enable the serial port to be used as an interactive terminal, so a user can log into it when running multi-user. This part is fairly similar between platforms, and is detailed here.

Terminal sessions are controlled by the */etc/ttys* file. Before OpenBSD will give you a "`login:`" prompt at a device, it has to be enabled in */etc/ttys*, after all, there are other uses for a serial port other than for a terminal. In platforms which typically have an attached keyboard and screen as a console, the serial terminal is typically disabled by default. We'll use the i386 platform as an example. In this case, you must edit the line that reads:

```
tty00   "/usr/libexec/getty std.9600"   unknown off
```

to read:

```
tty00   "/usr/libexec/getty std.9600"   vt100   on secure
```

Here, `tty00` is the serial port we are using as a console. The "`on`" activates the *getty* for that serial port so that a "`login:`" prompt will be presented, the "`secure`" permits a root (uid 0) login at this console (which may or may not be what you desire), and the "`9600`" is the terminal baud rate. Note that you can use a serial console for install without doing this step, as the system is running in single user mode, and not using *getty* for login.

On some platforms and some configurations, you must bring the system up in single user mode to make this change if a serial console is all you have available.

i386

To direct the boot process to use the serial port as a console, create or edit your */etc/boot.conf* file to include the line:

```
set tty com0
```

to use the first serial port as your console. The default baud rate is 9600bps, this can be changed with a */etc/boot.conf* line using the stty option. This file is put on your boot drive, which could also be your install floppy, or the command can be entered at the `boot>` prompt from the OpenBSD second-stage boot loader for a one-time (or first time) serial console usage.

i386 notes:

- OpenBSD numbers the serial ports starting at *tty00*, DOS/Windows labels them starting at *COM1*. So, keep in mind *tty02* is *COM3*, not *COM2*
- Some systems may be able to operate without a video card in the machine, but certainly not all -- many systems consider this an error condition. Some machines will even refuse to work easily without a keyboard attached.
- Some systems are capable of redirecting all BIOS keyboard and screen activity to a serial port through a configuration option, so the machine can be completely maintained through the serial port. Your results may vary -- this port may not be available to the operating system once booted, thus requiring monitoring using TWO serial ports.
- PC compatible computers are not designed to be run from a serial console, unlike some other platforms. Even those systems that support a serial console usually have it as a BIOS configuration option -- and should the configuration information get corrupted, you will find the system looking for a standard monitor and keyboard again. You generally must have some way to get a monitor and keyboard to your i386 systems in an emergency.
- You will need to edit */etc/ttys* as above.
- Only the first serial port (com0) is supported for console on i386.

SPARC and UltraSPARC

These machines are designed to be completely maintainable with a serial console. Simply remove the keyboard from the machine, and the system will run serial.

SPARC and UltraSPARC notes

- The serial ports on a SPARC are labeled *ttya*, *ttyb*, etc.
- Unlike some other platforms, it is not necessary to make any changes to */etc/ttys* to use a serial console.
- The SPARC/UltraSPARC systems interpret a BREAK signal on the console port to be the same as a STOP-A command, and kicks the system back to the Forth prompt, stopping any application and operating system at that point. This is handy when desired, but unfortunately, some serial terminals at power-down and some RS-232 switching devices send something the computer interprets as a break signal, halting the machine. Test before you go into production.
- If you have a keyboard and monitor attached, you can still force the serial console to be used instead by using the following commands at the `ok` prompt:

```
ok setenv input-device ttya
ok setenv output-device ttya
ok reset
```

  If the keyboard and monitor (ttyC0) are active in */etc/ttys* (above), you can use the keyboard and monitor in X.

MacPPC

The MacPPC machines are configured for a serial console through OpenFirmware. Use the commands:

```
ok setenv output-device scca
ok setenv input-device scca
ok reset-all
```

Set your serial console to 57600bps, 8N1.

MacPPC notes

- Unfortunately, serial console is not directly possible on most MacPPCs. While most of these machines do have serial hardware, it isn't accessible outside the machine. Fortunately, a few companies offer add-on devices for several Macintosh models which will make this port available for use as a serial console (or other uses). Use your favorite search engine and look for "Macintosh internal serial port".
- You will have to change `tty00` in */etc/ttys* to `on` and set the speed to 57600 instead of the default of 9600 as detailed above in single user mode before booting multi-user and having the serial console functional.

Mac68k

Serial console is selected in the *Booter* program, under the "`Options`" pull-down menu, then "`Serial Ports`". Check the "Serial Console" button, then choose the Modem or Printer port. You will need a Macintosh modem or printer cable to attach to the Mac's serial ports. If you wish to have this as default, tell the Booter program to save your options.

Mac68k Notes

- The modem port is *tty00*, the printer port is *tty01*.
- The Mac68k doesn't turn on its serial port until called upon, so your breakout box may not show any signals on the Mac's

serial port until the OpenBSD boot process has started.
- You will have to enable the port (*tty00* or *tty01*) as indicated above.

7.8 - How do I blank my console? (wscons)

If you wish to blank your console after a period of inactivity without using X, you can alter the following wscons(4) variables:

- **`display.vblank`** set to `on` will disable the vertical sync pulse, which will cause many monitors to go into an "energy saver" mode. This will require more time to bring the screen back on, but will reduce energy consumption and heat production of newer monitors. When set to `off`, the display will blank, but the monitor will still be receiving the normal horizontal and vertical sync pulses, so the unblanking will be almost instant.
- **`display.screen_off`** determines the blanking time in thousandths of a second, i.e., 60000 would set the timeout to one minute.
- **`display.kbdact`** determines if keyboard activity will restore the blanked screen. Usually, this is desirable.
- **`display.outact`** determines if screen output will restore the blanked screen.

You can set these variables at the command line using the wsconsctl(8) command:

```
# wsconsctl -w display.screen_off=60000
display.screen_off -> 60000
```

or set them permanently by editing */etc/wsconsctl.conf* so these changes take place at next boot:

```
display.vblank=on               # enable vertical sync blank
display.screen_off=600000       # set screen blank timeout to 10 minutes
display.kbdact=on               # Restore screen on keyboard input
display.outact=off              # Restore screen on display output
```

The blanker is activated when either `display.kbdact` or `display.outact` is set to "`on`".

[FAQ Index] [To Section 6 - Networking] [To Section 8 - General Questions]


www@openbsd.org

$OpenBSD: faq7.html,v 1.59 2004/04/01 04:16:38 nick Exp $

OpenBSD

[FAQ Index] [To Section 7 - Keyboard and Display controls] [To Section 9 - Migrating from Linux]

8 - General Questions

Table of Contents

8.1 - 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.
   ```
   # fsck -p / && mount -uw /
   ```
3. If /usr is not the same partition that / is (and it shouldn't be) then you will need to mount it, also
   ```
   # fsck -p /usr && mount /usr
   ```
4. run passwd(1)
5. boot into multiuser mode... and *remember* your password!

8.2 - 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:

```
option APERTURE
```

is in your kernel configuration. [It is already in the GENERIC kernel]

Then you need to edit */etc/sysctl.conf* and set **machdep.allowaperture=2**. This will allow X to access the aperture driver. This would already be set if you said that you would be running X when asked during the install. 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 Online Documentation at http://www.xfree86.org/support.html.

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

CVS is the tool that 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.cvshome.org/. 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

You can retrieve the sources from one 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 FAQ 8, Ports of the OpenBSD FAQ.
- *XF4* - The XF4 module contains the source to compile XFree 4.

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 ssh(1), but some AnonCVS servers allow for the use of rsh. 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.

```
$ export CVSROOT=anoncvs@anoncvs.usa.openbsd.org:/cvs
$ 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 environment 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:

```
$ 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.

```
$ echo $CVSROOT
anoncvs@anoncvs.usa.openbsd.org:/cvs
$ cvs -q up -Pd www
Warning: Remote host denied X11 forwarding, perhaps xauth program could not be run on the server side.
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.

8.4 - 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/3.4/ports.tar.gz`. The most recent ports are available via the 'ports' cvs tree, or `/pub/OpenBSD/snapshots/ports.tar.gz`. For most of you however, packages will be a much better option. Packages are created from ports and are already compiled and ready to use. To read more on packages read FAQ 8, Packages.

Important note about keeping your system and ports in sync

OpenBSD has three "active" versions at any point in time:

- Release: What is on the CD.
- Stable: Release, plus security and reliability enhancements
- Current: The development version of OpenBSD.

DO NOT mix versions of Ports and OpenBSD!

If your system is Release, use the Release version of the ports tree. Don't try to use a -Current version of the Ports tree on a -Release or -Stable system. Not only is it not likely to work, you will irritate people when you ask for help about why "nothing seems to work!" Note that there is a -Stable branch of the Ports tree as well, where critical fixes to -Release ports will be made.

Yes, this really does mean a wonderful new port will not typically work on your "older" system -- even if that system was -current just a few weeks ago.

If you do not have the ports tree installed, you can download it via any of OpenBSD's FTP servers, or of course, from the CD-ROM. The file is `ports.tar.gz`, and you want to untar this in the /usr directory, which will create /usr/ports, and all the directories under it. For example:

```
$ ftp ftp://ftp.openbsd.org/pub/OpenBSD/3.4/ports.tar.gz
$ sudo cp ports.tar.gz /usr
$ cd /usr; sudo tar xzf ports.tar.gz
```

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`. If you are installing a snapshot of OpenBSD, you should use a matching snapshot of ports. Again, make sure you keep your ports tree and your OpenBSD system in sync.

What ports are available? How do i find them?

Use the ports tree to search for keywords. To do this use `make search key="searchkey"`. Here is an example of a search for 'samba':

```
$ make search key="samba"
  [...snip...]
Port:   amanda-client-2.4.2.2
Path:   misc/amanda,-client
Info:   network-capable tape backup (client only)
Maint:  Tom Schutter <t.schutter@att.net>
Index:  misc
L-deps:
B-deps: :devel/gmake gnuplot-*:math/gnuplot gtar-*:archivers/gtar samba-*:net/samba/stable
R-deps:
Archs:  any

Port:   samba-2.2.8a
Path:   net/samba/stable
Info:   SMB and CIFS client and server for UNIX
Maint:  The OpenBSD ports mailing-list <ports@openbsd.org>
Index:  net
L-deps: popt::devel/popt
B-deps: :devel/autoconf/2.13 :devel/metaauto
R-deps:
Archs:  any
  [...snip...]
```

Installing Ports

Ports are set up to be EXTREMELY easy to make and install. Here is an example showing how 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 above. Once you are in the dir of the program you want, you can just type `make install`. For example.

```
$ sudo make install
===>  Checking files for xfig-3.2.4
>> xfig.3.2.4.full.tar.gz doesn't seem to exist on this system.
>> Attempting to fetch /usr/ports/distfiles/xfig.3.2.4.full.tar.gz from http://w
ww.xfig.org/xfigdist/.
100% |**********************************************|  5042 KB    00:31
>> Checksum OK for xfig.3.2.4.full.tar.gz. (sha1)
```

```
===>  xfig-3.2.4 depends on: jpeg.62 - jpeg.62 missing...
===>  Verifying install for jpeg.62 in graphics/jpeg
===>  Checking files for jpeg-6b
>> jpegsrc.v6b.tar.gz doesn't seem to exist on this system.
>> Attempting to fetch /usr/ports/distfiles/jpegsrc.v6b.tar.gz from ftp://ftp.uu
.net/graphics/jpeg/.
'EPSV': command not understood.
100% |***********************************************|   598 KB    00:06
>> Checksum OK for jpegsrc.v6b.tar.gz. (sha1)
===>  Extracting for jpeg-6b
===>  Patching for jpeg-6b
===>  Configuring for jpeg-6b
checking for gcc... cc
checking whether the C compiler (cc -O2   ) works... yes
checking whether the C compiler (cc -O2   ) is a cross-compiler... no
checking whether we are using GNU C... yes
  [...snip...]
```

Using Flavors

Many of the applications in the ports tree support different install options, called *flavors*. If a port comes in multiple flavors, you can use these options simply by setting an environment variable before you compile the port. If multiple features are needed, the FLAVOR variable can be set to a space-delimited list of the supported and desired flavors. Currently, many ports have flavors that include database support, support for systems without X, or network additions like SSL and IPv6.

```
$ pwd
/usr/ports/net/mtr
$ make show=FLAVORS
no_x11
$ env FLAVOR="no_x11" make
===>  mtr-0.49-no_x11 depends on: gmake-3.80 - not found
===>  Verifying install for gmake-3.80 in devel/gmake
===>  Checking files for gmake-3.80
>> make-3.80.tar.gz doesn't seem to exist on this system.
>> Attempting to fetch /usr/ports/distfiles/make-3.80.tar.gz from ftp://ftp.gnu.
org/gnu/make/.
Unknown command.
100% |***********************************************|  1183 KB    00:07
>> Checksum OK for make-3.80.tar.gz. (sha1)
  [...snip...]
$ sudo env FLAVOR="no_x11" make install
===>  Faking installation for mtr-0.49-no_x11
  [...snip...]
===>  Building package for mtr-0.49-no_x11
Creating package /usr/ports/packages/i386/All/mtr-0.49-no_x11.tgz
Using SrcDir value of /usr/ports/net/mtr/w-mtr-0.49-no_x11/fake-i386-no_x11/usr/
local
Creating gzip'd tar ball in '/usr/ports/packages/i386/All/mtr-0.49-no_x11.tgz'
===>  Installing mtr-0.49-no_x11 from /usr/ports/packages/i386/All/mtr-0.49-no_a
x11.tgz
```

Listing Installed ports/packages

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

```
$ /usr/sbin/pkg_info
zsh-4.1.1           The Z shell.
screen-3.9.15       A multi-screen window manager.
emacs-21.3          GNU editing macros.
tcsh-6.12.00        An extended C-shell with many useful features.
bash-2.05b          The GNU Borne Again Shell.
zip-2.3             Create/update ZIP files compatible with pkzip.
ircII-20030314      An enhanced version of ircII, the Internet Relay Chat client
ispell-3.2.06       An interactive spelling checker.
tin-1.6.1           TIN newsreader (termcap based)
procmail-3.22       A local mail delivery agent.
strobe-1.06         Fast scatter/gather TCP port scanner
lsof-4.68           Lists information about open files.
ntp-4.1.74          Network Time Protocol Implementation.
ncftp-3.1.5p0       ftp replacement with advanced user interface
nmh-1.0.4p1         The New MH mail handling program
bzip2-1.0.2         A block-sorting file compressor
```

Other Information

More information about the ports can be found in the ports(7) man page and on the Ports page.

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

8.5 - 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

Where to find packages

If you are a smart user and bought an OpenBSD CD set, then packages can be found on one of the three CDs, depending on your architecture. If you don't have an OpenBSD CD in your possession you can download packages from any of the ftp mirrors. You can get a list of mirrors http://www.openbsd.org/ftp.html. Packages are located at */pub/OpenBSD/3.4/packages* from there packages are broken down depending on architecture.

Installing Packages

To install packages, the utility pkg_add(1) is used. pkg_add(1) is an extremely easy utility to use, in the following two examples pkg_add(1) will be used to install a package. The first example will show pkg_add(1) installing a package that resides on a local disk, the second example will show an installation of a package via ftp. In both examples screen-3.9.15 will be installed.

Installing via local disk

```
$ sudo pkg_add -v screen-3.9.15.tgz
Requested space: 749864 bytes, free space: 2239117312 bytes in /var/tmp/instmp.cpsHA27596
Running install with PRE-INSTALL for `screen-3.9.15'
extract: Package name is screen-3.9.15
extract: CWD to /usr/local
extract: /usr/local/bin/screen-3.9.15
extract: execute 'ln -sf screen-3.9.15 /usr/local/bin/screen'
extract: /usr/local/man/man1/screen.1
extract: /usr/local/info/screen.info
extract: execute '[ -f /usr/local/info/dir ] || sed -ne '1,/Menu:/p' /usr/share/info/dir > /usr/local/info/dir'
extract: execute 'install-info /usr/local/info/screen.info /usr/local/info/dir'
extract: /usr/local/lib/screen/screencap
extract: /usr/local/lib/screen/screenrc
extract: CWD to .
Running mtree for `screen-3.9.15'
mtree -q -U -f +MTREE_DIRS -d -e -p /usr/local
Running install with POST-INSTALL for `screen-3.9.15'

+---------------
| The file /etc/screenrc has been created on your system.
| You may want to verify/edit its contents
|
| The file /usr/local/lib/screen/screencap contains a
| termcap like description of the screen virtual terminal.
| You may use it to update your terminal database.
| See termcap(5).
+---------------

Attempting to record package into `/var/db/pkg/screen
Package `screen-3.9.15' registered in `/var/db/pkg/screen-3.9.15'
```

In this example the **-v** flag was used to give a more verbose output. This option is not needed but is helpful for debugging and was used here to give a little more insight into what pkg_add(1) is actually doing. Notice however, that there are some valid messages given out mentioning */etc/screenrc*. Messages like this will be given to you whether or not you use the **-v** flag.

Installing via ftp

```
$ sudo pkg_add ftp://ftp.openbsd.org/pub/OpenBSD/3.4/packages/i386/screen-3.9.15.tgz
```

```
>>> ftp -o - ftp://ftp.openbsd.org/pub/OpenBSD/3.4/packages/i386/screen-3.9.15.tgz

+---------------
| The file /etc/screenrc has been created on your system.
| You may want to verify/edit its contents
|
| The file /usr/local/lib/screen/screencap contains a
| termcap like description of the screen virtual terminal.
| You may use it to update your terminal database.
| See termcap(5).
+---------------
```

In this example you can see that I installed the i386 package. You should substitute *i386* (above) with your architecture. Notice: Not all architectures have the same packages. Some ports don't work on certain architectures. In this example the **-v** flag wasn't used, so only NEEDED messages are shown.

Viewing and Deleting Installed Packages

The utility pkg_info(1) is used to view a list of packages that are already installed on your system. This is usually needed to find out the correct name of a package before you remove that package. To see what packages are installed on your system simple use:

```
$ pkg_info
mpg123-0.59rp1     mpeg audio 1/2 layer 1, 2 and 3 player
nmap-3.00          port scanning large networks
ircII-20030314     enhanced version of ircII (internet relay chat)
screen-3.9.15      multi-screen window manager
unzip-5.50r2       extract, list & test files in a ZIP archive
ntp-4.1.74         Network Time Protocol implementation
icb-5.0.9p1        Internet CB - mostly-defunct chat client
```

To delete a package, simple take the proper name of the package as shown by pkg_info(1) and use pkg_delete(1) to remove the package. In the below example, the screen package is being removed. Notice that on some occasions there are instructions of extra objects that need to be removed that pkg_delete(1) did not remove for you. As with the pkg_add(1) utility, you can use the **-v** flag to get more verbose output.

```
$ sudo pkg_delete screen-3.9.15

+---------------
| To completely deinstall the screen-3.9.15 package you need to perform
| this step as root:
|
|           rm -f /etc/screenrc
|
| Do not do this if you plan on re-installing screen-3.9.15
| at some future time.
+---------------
```

8.6 - Should I use Ports or Packages?

In general, you are HIGHLY advised to use packages over building an application from ports. The OpenBSD ports team considers packages to be the goal of their porting work, not the ports themselves.

Building a complex application from source is not trivial. Not only must the application be compiled, but the tools used to build it must be built. Unfortunately, OpenBSD, the tools, and the application are all evolving, and often, getting all the pieces working together is a challenge. Once everything works, a revision in any of the pieces the next day could render it broken. Every six months, as a new release of OpenBSD is made, an effort is made to test the building of every port on every platform, but during the development cycle it is likely that some ports will break.

In addition to having all the pieces work together, there is just the matter of time and resources required to compile some applications from source. A common example is CVSup, a tool commonly used to track the OpenBSD source tree. To install CVSup on a moderately fast system with a good Internet connection may take only about ten seconds -- the time required to download and unpack a single 511kB package file. In contrast, building CVSup on the same machine from source is a huge task, requiring many tools and bootstrapping a compiler, takes almost half an hour on the same machine. Other applications, such as Mozilla or KDE may take hours and huge amounts of disk space and RAM/swap to build. Why go through this much time and effort, when the programs are already compiled and sitting on your CD-ROM or FTP mirror, waiting to be used?

Of course, there are a few good reasons to use ports over packages in some cases:

- Distribution rules prohibit OpenBSD from distributing a package.
- You wish to modify or debug the application or study its source code.
- You need a FLAVOR of a port that is not built by the OpenBSD ports team.
- You wish to alter the directory layout (i.e., modifying PREFIX or SYSCONFDIR)

However, for most people and most applications, using packages is a far easier, and is the recommended way of adding applications to OpenBSD.

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

You need to set the kernel to always assume the floppy is attached, even if not detected during the hardware probe, by setting the 0x20 flag bit on fdc(4). This can be done by using User Kernel Config or config(8) to alter your kernel,

```
# config -e -f /bsd
OpenBSD 3.4 (GENERIC) #18: Wed Sep 17 03:34:47 MST 2003
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/GENERIC
Enter 'help' for information
ukc> change fd*
204 fd* at fdc0 drive -1 flags 0x0
change [n] y
drive [-1] ? ENTER
flags [0] ? 0x20
204 fd* changed
204 fd* at fdc0 drive -1 flags 0x20
ukc> q
Saving modified kernel.
#
```

You can also change this by adding "flags 0x20" at the end of the fd* entry in your Kernel Configuration file recompile your kernel. The line should read:

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

8.9 - OpenBSD Bootloader (*i386 specific*)

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 if that fails, it will try **/bsd.old**. You can specify a kernel by hand by typing:

```
boot> boot hd0a:/bsd
```

or

```
boot> b /bsd
```

This will boot the kernel named `bsd` from the 'a' partition of the first BIOS recognized hard disk.

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.

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

For further reading you can read boot(8)'s 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 in 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. The following algorithms are available:

- md4
- md5

- sha1
- rmd160.

Setting up S/Key - The first steps

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

```
# mkdir /etc/skey
```

Once that directory is in existence, you can initialize 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 S/Key *secret password*, and is **NOT** your system password. Your secret password must be at least 10 characters. We suggest using a memorable phrase containing several words as the secret password. Here is an example user being added.

```
$ 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. Starting with OpenBSD 3.0, S/Key logins work differently. For OpenBSD 3.0 and above you append **:skey** to your login name. For versions of OpenBSD previous to 3.0 you use **s/key** for the password at which time you are prompted for your S/Key password (the exception to this is ftpd(8) which will always present an S/Key challenge for S/Key-enabled user prior to OpenBSD < 3.0). The examples below assume OpenBSD 3.0 or higher.

```
$ ftp localhost
Connected to localhost.
220 oshibana.shin.ms FTP server (Version 6.5/OpenBSD) ready.
Name (localhost:ericj): ericj:skey
331- otp-md5 96 oshi45820
331 S/Key Password:
230- OpenBSD 3.4 (GENERIC) #18: Wed Sep 17 03:34:47 MDT 2003
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.
```

Note that I appended ":skey" to my username. This tells ftpd that I want to authenticate using S/Key. Some of you might have noticed that my sequence number has changed to *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).

```
$ skeyinfo
95 oshi45820
```

An even better way is to use **skeyinfo -v**, which outputs a command suitable to be run in the shell. For instance:

```
$ skeyinfo -v
otp-md5 95 oshi45820
```

Not only is *otp-md5* a description of the hash used, it is also an alternate name for the skey(1) command. So, the simplest way to generate the next S/Key password is simply:

```
$ `skeyinfo -v`
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password:
NOOK CHUB HOYT SAC DOLE FUME
```

Note the backticks in the above example.

I'm sure many of you won't always have a secure connection 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.

```
$ otp-md5 -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), ssh(1), and rlogin(1)

Using S/Key with telnet(1), ssh(1), or rlogin(1) is done in pretty much the same fashion as with ftp--you simply tack ":skey" to the end of your username. Example:

```
$ telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

OpenBSD/i386 (oshibana) (ttyp2)

login: ericj:skey
otp-md5 98 oshi45821
S/Key Password: SCAN OLGA BING PUB REEL COCA
Last login: Thu Oct  7 12:21:48 on ttyp1 from 156.63.248.77
OpenBSD 3.4 (GENERIC) #18: Wed Sep 17 03:34:47 MDT 2003

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.
$
```

8.12 - Does OpenBSD support SMP? (Symmetric Multi-Processor)

Not at this time. Work is on-going, there is an OpenBSD SMP project, though there is nothing usable yet, nor is there any time schedule for OpenBSD SMP support.

On most platforms, OpenBSD will run on an SMP system, but only utilizing one processor. The exception to this is the SPARC platform -- OpenBSD/sparc will sometimes require that extra MBus modules be removed for the system to boot. Multi-processor SPARC64 systems run as long as the base machine is supported.

How can I help?

- Write code. There is lots of work to do.
- Donate hardware. The goal of the OpenBSD SMP project is to support as many platforms as possible, including SPARC, SPARC64, Alpha, MacPPC and of course, the i386. Donations of desired hardware are welcome!
- Don't offer to "test". The SMP project is not part of the core OpenBSD project right now for a reason -- it isn't ready for production use yet, or even general "testing". Informing the developers that it didn't work on your hardware without code to make it work isn't useful at this point. Send in your dmesg, if a developer sees something interesting in your hardware, they can contact you for testing.

8.13 - I get Input/output error when trying to use my tty devices

You need to use /dev/cuaXX for connections initiated from the OpenBSD system, the /dev/ttyXX devices are intended only for terminal or dial-in usage. While it was possible to use the tty devices in the past, the OpenBSD kernel is no longer compatible with this usage.

From cua(4):

For hardware terminal ports, dial-out is supported through matching device nodes called calling units. For instance, the terminal called /dev/tty03 would have a matching calling unit called /dev/cua03. These two devices are normally differentiated by creating the calling unit device node with a minor number 128 greater than the dial-in device node. *Whereas the dial-in device (the tty) normally requires a hardware signal to indicate to the system that it is active, the dial-out device (the cua) does not, and hence can communicate unimpeded with a device such as a modem*. This means that a process like getty(8) will wait on a dial-in device until a connection is established. Meanwhile, a dial-out connection can be established on the dial-out device (for the very same hardware terminal port) without disturbing anything else on the system. The getty(8) process does not even notice that anything is happening on the terminal port. If a connecting call comes in after the dial-out connection has finished, the getty(8) process will deal with it properly, without having noticed the intervening dial-out action.

8.14 - What web browsers are available for OpenBSD?

Lynx, a text-based browser, is in the base system, and has SSL support. Other browsers in the ports tree, include (in no particular order):

Graphical (X) Browsers

- Dillo Minimal feature set, very fast and small, runs well on slower hardware.
- Konqueror Installed as part of the KDE desktop environment.
- Konqueror-embedded (konq-e) Konqueror, using only the KDE libraries rather than all of KDE.
- Netscape 4 For sparc and i386 only, not Open Source, no package available.
- Opera Commercial browser, i386 only.
- Amaya The W3C's browser and editor.
- Links+ Another fast and small graphical browser. (Also has a text-only mode)
- *(new for 3.4)* Mozilla and Firebird Feature-filled browsers. Mozilla includes a many non-browser features (mail client, IRC client, etc.), Firebird is just a browser, based on Mozilla. Works on alpha, i386, sparc, and sparc64 platforms.

Console (Text mode) Browsers

- links Has table support.

You will find all these in the ports collection. All the above mentioned browsers are located in `/usr/ports/www/` after the installation of the ports tree. Most are also available as pre-compiled packages, available on the FTP servers and on the CD-ROM. As most of the graphical browsers are very large and require quite some time to download and compile, one should *seriously* consider the use of packages where available.

Unfortunately, due to a bug in one of the libraries used by w3m, it does not work on 3.4-release on any platform.

8.15 - How do I use the mg editor?

Mg is a micro Emacs-style text editor included in OpenBSD. Micro means that it's small (Emacs is very large!) For the basics, read the mg(1) manual page and the tutorial, as included with the source code. For more interesting questions (such as, "I don't have a Meta key!") check out the Emacs FAQ.

Note that since mg is a small Emacs implementation, which is mostly similar to the text editor features of Emacs 17, it does not implement many of Emacs' other functionality. (Including mail and news functionality, as well as modes for Lisp, C++, Lex, Awk, Java, etc...)

8.16 - ksh(1) does not appear to read my `.profile`!

There are two likely reasons for ksh(1) to seemingly ignore a user's `.profile` file.

- `.profile` is not owned by the user. To fix for **username**,

```
# chown username ~username/.profile
```

- You are using ksh(1) from within X Window System

  Under xterm(1), argv[0] for ksh(1) is not prepended with a dash ("-"). Prepending a dash to argv[0] will cause csh(1) and ksh(1) to know they should interpret their login files. (For csh(1) that's `.login`, with a separate `.cshrc` that is always run when csh(1) starts up. With ksh(1), this is more noticeable because there is only one startup script, `.profile`. This file is ignored unless the shell is a login shell.)

  To fix this, add the line "`XTerm*loginShell: true`" to the file `.Xdefaults` in your home directory. Note, this file does not exist by default, you may have to create it.

```
$ echo "XTerm*loginShell: true" >> ~/.Xdefaults
```

  You may not have had to do this on other systems, as some installations of X Window System come with this setting as default. OpenBSD has chosen to follow the XFree86 behavior.

8.17 - Why does my `/etc/motd` file get overwritten when I modified it?

The `/etc/motd` file is edited upon every boot of the system, replacing everything up to, but not including, the first blank line with the system's kernel version information. When editing this file, make sure that you start after this blank line, to keep `/etc/rc` from deleting these lines when it edits `/etc/motd` upon boot.

8.18 - Why does www.openbsd.org run on Solaris?

Although none of the developers think it is particularly relevant, this question comes up frequently enough in the mailing lists that it is answered here. *www.openbsd.org* and the main OpenBSD ftp site are hosted at a SunSITE at the University of Alberta, Canada. These sites are hosted on a large Sun system, which has access to lots of storage space and Internet bandwidth. The presence of the SunSITE gives the OpenBSD group access to this bandwidth. This is why the main site runs here. Many of the OpenBSD mirror sites run OpenBSD, but since they do not have guaranteed access to this large amount of bandwidth, the group has chosen to run the main site at the University of Alberta SunSITE.

8.19 - I'm having problems with my PCI devices being detected

There exists a condition where some machines might not detect some PCI devices properly, or might freeze while detecting multiple NIC's in one machine. This is the fault of PCIBIOS, and involves a simple workaround to make it work properly. Simply enter the boot time configuration and disable PCIBIOS. An example is below:

```
boot> boot -c
Copyright (c) 1982, 1986, 1989, 1991, 1993
        The Regents of the University of California.  All rights reserved.
Copyright (c) 1995-2002 OpenBSD. All rights reserved.  http://www.OpenBSD.org

OpenBSD 3.4 (GENERIC) #18: Wed Sep 17 03:34:47 MDT 2003
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/RAMDISK_CD
cpu0: Intel Pentium III (Coppermine) ("GenuineIntel" 686-class, 128KB L2 cache)
1 GHz
cpu0: FPU,V86,DE,PSE,TSC,MSR,PAE,MCE,CX8,SYS,MTRR,PGE,MCA,CMOV,PAT,PSE36,MMX,FXS
R,SIMD
real mem  = 267956224 (261676K)
avail mem = 243347456 (237644K)
using 3296 buffers containing 13500416 bytes (13184K) of memory
User Kernel Config
UKC> disable pcibios
UKC> quit
[... snip ...]
```

Once this is done, you can follow the directions in FAQ 5, Building a Kernel to create a new kernel so that you don't have to worry about this in the future.

8.20 - Antialiased and TrueType fonts in XFree86

See this document.

8.21 - Does OpenBSD support any journaling filesystems?

No it doesn't. We use a different mechanism to achieve similar results that is called Soft Updates. Please read in FAQ 14 to get more details.

8.22 - Reverse DNS
- or -
Why is it taking so long for me to log in?

Many new users to OpenBSD experience a two minute login delay when using services such as ssh, ftp, or telnet. This can also be experienced when using a proxy, such as ftp-proxy, or when sending mail out from a workstation through sendmail.

This is almost always due to a reverse-DNS problem. DNS is Domain Name Services, the system the Internet uses to convert a name, such as "www.openbsd.org" into a numeric IP address. Another task of DNS is the ability to take a numeric address and convert it back to a "name", this is "Reverse DNS".

In order to provide better logging, OpenBSD performs a reverse-DNS lookup on any machine that attaches to it in many different ways, including ssh, ftp, telnet, sendmail or ftp-proxy. Unfortunately, in some cases, the machine that is making the connection does not have a proper reverse DNS entry.

An example of this situation:

A user sets up an OpenBSD box as a firewall and gateway to their internal home network, mapping all their internal computers to one external IP using NAT. They may also use it as an outbound mail relay. They follow the installation guidelines, and are very happy with the results, except for one thing -- every time they try to attach to the box in any way, they end up with a two minute delay before things happen.

What is going on:

From a workstation behind the NAT of the gateway with an unregistered IP address of 192.168.1.35, the user uses ssh to access the gateway system. The ssh client prompts for username and password, and sends them to the gateway box. The gateway then tries to figure out who is trying to log in by performing a reverse DNS lookup of 192.168.1.35. The problem is 192.168.0.0 addresses are for private use, so a properly configured DNS server outside your network knows it should have no information about those addresses. Some will quickly return an error message, in these cases, OpenBSD will assume there is no more information to be gained, and it will quickly give up and just admit the user. Other DNS servers will not return ANY response. In this case you will find yourself waiting for the OpenBSD name resolver to time out, which takes about two minutes before the login will be permitted to continue. In the case of ftp-proxy, some ftp clients will timeout before the reverse DNS query times out, leading to the impression that ftp-proxy isn't working.

This can be quite annoying. Fortunately, it is an easy thing to fix.

Fix, using `/etc/hosts`:

The simplest fix is to populate your `/etc/hosts` file with all the workstations you have in your internal network, and ensure that your `/etc/resolv.conf` file contains the line `lookup file bind` which ensures that the resolver knows to start with the `/etc/hosts` file, and failing that, to use the DNS servers specified by the "nameserver" lines in your `/etc/resolv.conf` file.

Your `/etc/hosts` file will look something like this:

```
::1 localhost.in.example.org localhost
127.0.0.1 localhost.in.example.org localhost
192.168.1.1 gw.in.example.org gw
192.168.1.20 scrappy.in.example.org scrappy
192.168.1.35 shadow.in.example.org shadow
```

Your `resolv.conf` file will look something like this:

```
search in.example.org
nameserver 24.2.68.33
nameserver 24.2.68.34
lookup file bind
```

A common objection to this is "But, I use DHCP for my internal network! How can I configure my `/etc/hosts`?" Rather easily, actually. Just enter lines for all the addresses your DHCP server is going to give out, plus any static devices:

```
::1 localhost.in.example.org localhost
127.0.0.1 localhost.in.example.org localhost
192.168.1.1 gw.in.example.org gw
192.168.1.20 scrappy.in.example.org scrappy
192.168.1.35 shadow.in.example.org shadow
192.168.1.100 d100.in.example.org d100
192.168.1.101 d101.in.example.org d101
192.168.1.102 d102.in.example.org d102
         [... snip ...]
192.168.1.198 d198.in.example.org d198
192.168.1.199 d199.in.example.org d199
```

In this case, I am assuming you have the DHCP range set to 192.168.1.100 through 192.168.1.199, plus the three static definitions as listed at the top of the file.

If your gateway must use DHCP for configuration, you may well find you have a problem -- dhclient will overwrite your `/etc/resolv.conf` every time the lease is renewed, which will remove the "lookup file bind" line. This can be solved by putting the line "lookup file bind" in the file `/etc/resolv.conf.tail`.

Fix, using a local DNS server

Details on this are somewhat beyond the scope of this document, but the basic trick is to setup your favorite DNS server, and make sure it knows it is authoritative for both forward and reverse DNS resolution for all nodes in your network, and make sure your computers (including your gateway) know to use it as a DNS server.

8.23 - Why do the OpenBSD web pages not conform to HTML4/XHTML?

The present web pages have been carefully crafted to work on a wide variety of actual browsers going back to browser versions 4.0 and later. We do not want to make these older pages conform to HTML4 or XHTML until we're sure that they will also work with older browsers; it's just not a priority. We welcome new contributors, but suggest you work on writing code, or on documenting new aspects of the system, not on tweaking the existing web pages to conform to newer standards.

8.24 - Why is my clock off by twenty-some seconds?

When using rdate(8) to synchronize your clock to a NTP server, you may find your clock is off by twenty-some seconds from your local definition of time.

This is caused by a difference between the UTC (Coordinated Universal Time, based on astronomical observations) time and TAI (International Atomic Time, based on atomic clocks) time. To compensate for variations in the earth's rotation, "leap seconds" are inserted into UTC, but TAI is unadjusted. These leap seconds are the cause of this discrepancy. For a more detailed description, search the web for "`leap seconds UTC TAI`".

Addressing the problem is fairly simple. In most countries you will get the correct time if you use the "`-c`" parameter to rdate(8) and use a time zone out of the directory `/usr/share/zoneinfo/right/`. For example, if you are located in Germany, you could use these commands:

```
# cd /etc && ln -sf /usr/share/zoneinfo/right/CET localtime
# rdate -ncv ptbtime1.ptb.de
```

In other countries, the rules may differ.

[FAQ Index] [To Section 7 - Keyboard and Display Controls] [To Section 9 - Migrating from Linux]


www@openbsd.org

$OpenBSD: faq8.html,v 1.143 2004/04/27 13:33:27 tom Exp $

[Index de La FAQ] [Section 8 - Questions Générales] [Section 10 - Gestion du Système]

9 - Astuces et conseils pour les utilisateurs de Linux (et d'autres OS libres Unix-Like)

Table Des Matières

Les utilisateurs Linux pourront trouver des informations supplémentaires a l'adresse suivante : http://sites.inka.de/mips/unix/bsdlinux.html.

9.1 - Astuces et conseils pour les utilisateurs de Linux (et d'autres OS libres Unix-Like)

Il y a plusieurs différences entre OpenBSD et Linux. Ces différences se manifestent, entre autres, dans la procédure de démarrage, l'utilisation des interfaces réseau et la gestion des disques. La plupart de ces différences sont bien documentées mais nécessitent une recherche dans les pages du manuel. Ce document fait office d'index de ces différences.

- OpenBSD possède une arborescence de ports. En effet, à l'heure actuelle il n'y a pas beaucoup d'applications qui sont natives à l'environnement OpenBSD. Le but de l'arborescence de ports est de permettre l'installation d'applications sur OpenBSD par des utilisateurs finaux et d'avoir plus d'applications intégrant les spécificités de l'environnement OpenBSD. Eventuellement, cette arborescence de ports peut être utilisée pour créer des packages binaires de manière simple.
- OpenBSD utilise CVS pour la gestion du code source. Dans l'environnement Linux, ce code source est disséminé à travers des distributions distinctes. OpenBSD a été le pionnier du CVS anonyme, qui permet à n'importe qui d'extraire l'arborescence complète des sources de n'importe quelle version de OpenBSD (de 2.0 à current, et toutes les révisions de tous les fichiers entre ces deux versions) à n'importe quel moment! Il y a aussi une agréable interface web à CVS très facile d'utilisation.
- OpenBSD diffuse périodiquement des snapshots pour les différentes architectures supportées et met à disposition une version officielle sur CD tous les 6 mois.
- OpenBSD contient de la CRYPTOGRAPHIE FORTE, que les systèmes d'exploitation basés aux Etats-Unis ne peuvent contenir (Voir http://www- .openbsd.org/fr/crypto.html). OpenBSD a aussi été soumis à un audit sécurité rigoureux et beaucoup de fonctionnalités de la sécurité ont déjà été implémentées dans le code source (IPSEC, KERBEROS).
- Le noyau de OpenBSD est /bsd.
- Le nom des disques durs est généralement */dev/wd*et */dev/sd* (SCSI ou périphériques ATA en émulation SCSI)
- /sbin/ifconfig sans arguments sous Linux donne l'état de toutes les interfaces. Sous OpenBSD, il est nécessaire

d'employer l'option -a.

- /sbin/route sans arguments sous Linux donne l'état de toutes les routes actives. Sous OpenBSD, vous aurez besoin du paramètre "*show*" ou utilisez la commande **netstat -r**.
- OpenBSD ne supporte pas les systèmes de fichiers journalisés tels que ReiserFS, JFS de IBM, ou XFS de SGI. Au lieu de cela, nous utilisons Soft Updates.
- OpenBSD est fourni avec Packet Filter (PF), et non pas ipfw, ipchains, ou ipf. Cela veut dire que :
  - La traduction d'adresses (connue sous le nom de "IP-Masquerading" sous Linux) est effectuée via pfctl (en utilisant l'option -N). (pfctl(8))
  - l'équivalent de ipfwadm est pfctl. (pfctl(8), pf(4), pf.conf(5))
  - Vous devriez consulter le Guide de l'Utilisateur PF pour une aide à la configuration détaillée.
- L'adresse d'une interface est stockée dans /etc/hostname.<interfacename>. Ce fichier peut contenir un nom au lieu d'une adresse IP.
- Le nom de la machine est stocké dans */etc/myname*
- Le routeur par défaut se trouve dans */etc/mygate*
- Le shell par défaut d'OpenBSD est /bin/sh, un Korn shell. Des shells tels que bash et tcsh peuvent être ajoutés par le biais des packages ou installés à partir de l'arborescence des ports.
- La gestion des mots de passe est fort différente. Les fichiers principaux ne sont pas les mêmes. (passwd(1), passwd(5))
- Les périphériques sont nommés d'après le pilote et non d'après leur type. Par exemple, il n'y a pas d'interfaces eth*. Ça serait ne0 pour une carte ethernet ne2000, et xl0 pour une interface 3Com Etherlink XL ou Fast Etherlink XL, etc.
- Les developpeurs OpenBSD fournissent des efforts considérables de maintenir les pages du manuel à jour et aussi précises que possible. Utilisez la commande man(1) pour rechercher des informations.

9.2 - Double démarrage de Linux et de OpenBSD

Oui c'est possible !

Lisez INSTALL.linux.

9.3 - Convertir votre fichier de mots de passe de Linux (ou de tout autre System-7) au format BSD

Tout d'abord, déterminez si votre fichier de mots de passe Linux est en mode shadow ou pas. Si c'est le cas, installez John the Ripper et utilisez son utilitaire unshadow pour faire fusionner les fichiers `passwd` et `shadow` en un seul fichier type System-7.

En utilisant votre fichier de mots de passe Linux, que nous allons appeler `linux_passwd`, vous devez rajouter "::0:0" entre les champs quatre et sept. awk(1) peut faire cela pour vous.

```
# cat linux_passwd | awk -F : # '{printf("%s:%s:%s:%s::0:0:%s:%s:%s\n", \
```

A partir de là, vous devriez éditer le fichier `new_passwd` et enlever les entrées correspondantes à root et d'autres entités système qui sont déjà présentes dans le fichier de mots de passe OpenBSD ou ne sont pas applicables à OpenBSD (toutes). De même, assurez vous qu'il n'y a pas des noms d'utilisateurs ou des ID utilisateurs dupliqués entre `new_passwd` et le fichier `/etc/passwd` de la machine OpenBSD. La manière la plus facile consiste à utiliser un nouveau `/etc/passwd`.

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

La dernière étape, `pwd_mkdb`, est nécessaire pour reconstruire les fichiers /etc/spwd.db et /etc/pwd.db. Cette commande crée aussi

un fichier de mots de passe au format System-7 (sans les mots de passe cryptés) dénommé /etc/passwd pour les programmes qui l'utilisent. OpenBSD utilise un algorithme de cryptage plus fort, blowfish, qui est rarement présent dans d'autres systèmes qui utilisent des fichiers de mots de passe au format System-7. Pour utiliser cet algorithme plus solide, dites aux utilisateurs d'utiliser 'passwd' et de changer leur mot de passe. Le nouveau mot de passe sera crypté avec l'algorithme par défaut (blowfish si vous n'avez pas édité /etc/passwd.conf). Ou, en utilisateur root, vous pouvez utiliser `passwd` *`username`*.

9.4 - Exécution des binaires Linux sous OpenBSD

OpenBSD/i386 est capable d'exécuter des binaires Linux lorsque le noyau est compilé avec l'option COMPAT_LINUX et le paramètre sysctl kern.emul.linux positionné. Si vous utilisez le noyau GENERIC (ce qui devrait être le cas normalement), l'option COMPAT_LINUX est incluse et vous aurez juste besoin de positionner le paramètre sysctl précité comme suit :

```
# sysctl -w kern.emul.linux=1
```

Pour que cette modification soit prise en compte à chaque redémarrage de la machine, supprimez le caractère "#" (commentaire) au début de la ligne

```
#kern.emul.linux=1      # enable running Linux binaries
```

dans le fichier /etc/sysctl.conf. Vous devez alors obtenir :

```
kern.emul.linux=1       # enable running Linux binaries
```

puis redémarrez votre système pour que cette modification puisse prendre effet.

Pour utiliser des binaires Linux qui ne sont pas statiquement liés (la plupart d'entre eux), vous devez suivre les instructions de la page du manuel compat_linux(8).

Un moyen simple d'obtenir la plupart des librairies Linux les plus communes est d'installer `redhat/base` à partir de la collection de ports. Pour plus d'informations concernant la collection de ports, consultez FAQ 8, Ports. Si vous avez déjà l'arborescence des ports, utilisez les commandes suivantes pour installer les librairies Linux :

```
# cd /usr/ports/emulators/redhat/base
# make install
```

9.5 - Accèder à vos fichiers Linux depuis OpenBSD

OpenBSD supporte le système de fichiers EXT2FS. Utilisez

```
# disklabel disk
```

(où *`disk`* est le nom du périphérique correspondant à votre disque, **`wd0`** par exemple) pour voir où se trouve votre partition Linux. Cependant, **n'utilisez pas** disklabel (8) ou fdisk (8) pour modifier le disklabel.

Pour plus d'informations concernant disklabel, veuillez consulter FAQ 14, Disklabel.

[Index de La FAQ] [Section 8 - Questions Générales] [Section 10 - Gestion du système]


www@openbsd.org

Originally [OpenBSD: faq9.html,v 1.58]
$Translation: faq9.html,v 1.18 2004/02/29 20:26:10 saad Exp $
$OpenBSD: faq9.html,v 1.15 2004/03/01 09:26:09 jufi Exp $

OpenBSD

[Index de la FAQ] [Section 9 - Migrer depuis Linux] [Section 11 - Optimisation des Performances]

10 - Gestion du Système

Table des matières

10.1 - Quand j'essaie de passer root à l'aide de su, on me dit que je suis dans le mauvais groupe

Les utilisateurs existant sur le système doivent être rajoutés au groupe `"wheel"` à la main. Ceci est fait pour des raisons de sécurité, et vous devriez apporter une attention toute particulière lorsque vous donnez l'accès à ce groupe à des utilisateurs. Sous OpenBSD, les utilisateurs appartenant au groupe `wheel` sont autorisés à utiliser le programme su(1) pour devenir root. Les utilisateurs n'appartenant pas à ce groupe ne peuvent pas utiliser su(1). Voici un exemple d'une entrée `/etc/group` pour mettre l'utilisateur **ericj** dans le groupe `"wheel"`.

Si vous ajoutez un utilisateur avec adduser(8), vous pouvez le mettre dans le groupe wheel en répondant wheel à la question "`Invite` *`user`* `into other groups:`". Ceci aura pour effet de rajouter l'entrée correspondante dans /etc/group qui ressemble à la ligne suivante :

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

Si vous cherchez un moyen pour limiter l'accès des utilisateurs aux privilèges du super utilisateur, sans pour autant les mettre dans le groupe `"wheel"`, utilisez sudo(8).

10.2 - Comment dupliquer un système de fichiers ?

Pour dupliquer votre système de fichiers, utilisez dump(8) et restore(8). Par exemple, pour dupliquer tout ce qu'il y a sous le répertoire SRC vers le répertoire DST, faites un :

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

dump est conçu pour vous fournir beaucoup de possibilités de sauvegarde, et c'est peut-être trop si vous voulez juste dupliquer une partie d'un système de fichiers (entier). La commande tar(1) peut être plus rapide pour ce genre d'opération. Le format est très similaire à celui de dump :

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

10.3 - Comment démarrer des services en même temps que le système ? (Vue d'ensemble de rc(8))

OpenBSD utilise un démarrage de type rc(8) . Il utilise seulement quelques fichiers clés pour le démarrage.

- /etc/rc - Script principal. Ne doit pas être édité.
- /etc/rc.conf - Fichier de configuration utilisé par */etc/rc* pour savoir quels services doivent être démarrés en même temps que le système
- /etc/rc.conf.local - Fichier de configuration servant à compléter /etc/rc.conf, ainsi vous ne devez pas toucher à /etc/rc.conf, ce qui est commode pour ceux qui (re)mettent souvent à jour leur système.
- /etc/netstart - Script pour initialiser le réseau. Ne devrait pas être édité.
- /etc/rc.local - Script utilisé pour l'administration locale. C'est là où les informations relatives à de nouveaux services ou des informations spécifiques à l'hôte doivent être stockées.
- /etc/rc.securelevel - Script utilisé pour exécuter des commandes qui doivent être exécutées avant que le niveau de sécurité ne change. Voir init(8)
- /etc/rc.shutdown - Script exécuté lors de l'arrêt de la machine. Mettez tout ce que vous voulez exécuter avant l'arrêt de la machine dans ce fichier. Voir rc.shutdown(8)

Comment fonctionne rc(8) ?

/etc/rc.conf (ou */etc/rc.conf.local*), */etc/rc.local* et */etc/rc.shutdown* sont les principaux fichiers à connaître par l'administrateur système. Pour comprendre le fonctionnement de la procédure rc(8), en voici le déroulement :

/etc/rc est appelé après le démarrage du noyau :

- Les systèmes de fichiers sont vérifiés. Cette vérification n'aura pas lieu si le fichier /etc/fastboot existe. Ce n'est certainement pas une bonne idée.
- Les variables de configuration sont lues à partir de */etc/rc.conf* et ensuite */etc/rc.conf.local*. Les paramètres dans rc.local.conf vont surpasser ceux se trouvant dans rc.conf.
- Les systèmes de fichiers sont montés
- */tmp* est nettoyé et les fichiers d'éditeurs sont préservés
- Le réseau est configuré à l'aide de */etc/netstart*
    - Les interfaces réseau sont montées.
    - Le nom d'hôte et le nom de domaine (ainsi que d'autres paramètres) sont positionnés
- Les services système sont démarrés
- Diverses vérifications sont effectuées (quota, savecore, etc).
- Les services locaux sont démarrés à partir de */etc/rc.local*

Démarrage des services fournis avec OpenBSD

La plupart des services fournis par défaut avec OpenBSD sont lancés au démarrage simplement en modifiant le fichier de configuration

/etc/rc.conf. Pour commencer, jetez un coup d'oeil au fichier /etc/rc.conf par défaut. Vous verrez des lignes similaires à la ligne suivante :

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

Une ligne telle que celle-ci montre que ftpd n'est pas lancé au démarrage du système (du moins pas à travers rc(8), lisez la FAQ Serveur FTP Anonyme pour plus d'informations). Dans tous les cas, chaque ligne est dotée d'un commentaire qui vous montrent les drapeaux utilisés dans le cadre d'une utilisation **NORMALE** du service. Cela ne veut pas dire que vous devez appeler ce service avec ces mêmes drapeaux. Vous pouvez toujours utiliser man(1) pour savoir comment démarrer un service donné de la manière que vous souhaitez. Par exemple, voici la ligne par défaut concernant httpd(8) :

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

D'après cet exemple, vous pouvez voir qu'aucun drapeau n'est nécessaire pour démarrer httpd normalement. Ainsi, la ligne " **httpd_flags=""** suffit. Mais pour démarrer httpd avec le support ssl (Reportez vous à la FAQ SSL ou à ssl(8)), vous devez démarrer httpd avec une ligne comme celle-ci : "httpd_flags="-DSSL"".

Une autre approche serait de ne jamais toucher à */etc/rc.conf*. Au contraire, créez le fichier */etc/rc.conf.local* et ne copiez que les lignes que vous comptez changer dans */etc/rc.conf* et modifiez-les comme vous voulez. Ceci peut rendre vos mises à jour futures plus faciles -- tous les changements se trouvant dans un fichier.

Démarrage et configuration des services locaux

Pour les services que vous installez via la collection de ports ou d'autres méthodes, vous devez utiliser le fichier */etc/rc.local*. Par exemple, j'ai installé un service fourni par l'applicatif /usr/local/sbin/daemonx. Je souhaite que ce service soit lancé au démarrage. Pour cela, je rajoute les lignes suivantes dans */etc/rc.local* :

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

(Si le service ne se détache pas automatiquement lors de son démarrage, souvenez-vous de rajouter "&" à la fin de la commande.)

A partir de là, ce service sera lancé au démarrage. Vous pourrez voir toutes les erreurs au démarrage. Un démarrage normal sans erreurs affichera le message suivant :

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

rc.shutdown

/etc/rc.shutdown est un script exécuté à l'arrêt de la machine. Toutes les tâches à effectuer avant l'arrêt du système devront être ajoutées à ce fichier. Si vous avez apm, vous pouvez aussi positionner "powerdown=YES". C'est l'équivalent de "shutdown -p".

10.4 - Pourquoi les utilisateurs sont interdits de relais quand ils envoient des mails à distance à travers mon système OpenBSD ?

Essayez ceci :

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

Le résultat ressemblerait à la ligne suivante :

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

Si ce fichier n'existe pas, créez le. Vous devez saisir les hôtes qui envoient des messages à distance en respectant la syntaxe suivante :

```
.domain.com    #Allow relaying for/to any host in domain.com
sub.domain.com #Allow relaying for/to sub.domain.com and any host in that domain
10.2           #Allow relaying from all hosts in the IP net 10.2.*.*
```

N'oubliez pas d'envoyer un signal 'HangUP' à sendmail (signal qui notifie la plupart des processus de relire leur fichier de configuration) :

```
# kill -HUP `head -1 /var/run/sendmail.pid`
```

Pour plus d'informations

- http://www.sendmail.org/~ca/email/relayingdenied.html
- http://www.sendmail.org/tips/relaying.html
- http://www.sendmail.org/antispam.html

10.5 - J'ai mis en place POP, mais j'ai des erreurs quand j'accède à ma messagerie via POP. Que puis-je faire ?

La plupart des problèmes rencontrés avec POP sont liés aux fichiers temporaires et aux fichiers verrous. Si votre serveur POP renvoie une erreur du type :

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

Essayez de positionner les permissions comme suit :

```
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
```

Vérifiez aussi que l'utilisateur possède son propre fichier /var/mail. Bien évidemment, ceci devrait être le cas (comme par exemple l'utilisateur joe qui possède /var/mail/joe) mais si ça n'a pas été configuré proprement, le problème viendrait de là !

Bien entendu, si vous donner l'accès à /var/mail en écriture au groupe mail, vous allez probablement vous exposer à des vagues et obscurs problèmes de sécurité. Il se pourrait que ça ne pose aucun problème mais on ne sait jamais (et particulièrement si vous êtes un site de haut vol, un FAI,...) ! Il existe plusieurs services POP de la collection de ports OpenBSD. Si possible, utilisez popa3d(8) disponible dans le système de base d'OpenBSD. Ou peut-être vous avez sélectionné les mauvaises options pour votre programme POP serveur (comme le dot locking). Ou vous avez peut-être simplement besoin de changer le répertoire dans lequel les verrous sont crées (bien que les opérations de verrouillage ne devraient être bénéfiques qu'au service POP).

PS: Il est à noter que OpenBSD n'a pas de groupe "mail". Vous devez en créer un, si nécessaire, dans le fichier */etc/group*. La ligne suivante devrait suffire :

```
mail:*:6:
```

10.6 - Pourquoi Sendmail ignore-t-il le fichier `/etc/hosts` ?

Par défaut, Sendmail utilise le DNS pour la résolution de nom, non le fichier `/etc/hosts`. Ce comportement peut être changé par l'usage du fichier `/etc/mail/service.switch`.

Si vous désirez interroger le fichier d'hôtes avant les serveurs DNS, créez un fichier `/etc/mail/service.switch` contenant les lignes suivantes :

```
hosts      files dns
```

Si vous désirez interroger QUE le le fichier d'hôtes, utilisez ce qui suit :

```
hosts      files
```

Envoyez un signal HUP à Sendmail :

```
# kill -HUP `head -1 /var/run/sendmail.pid`
```

et les changements prendront effet.

10.7 - Configurer HTTP en mode sécurisé à l'aide de SSL(8)

OpenBSD est fourni avec des librairies RSA et un service httpd supportant SSL. Pour utiliser SSL avec httpd(8), vous devez d'abord créer un certificat. Ce certificat sera stocké dans */etc/ssl/private/*. Les étapes décrites ici sont en partie prises de la page de manuel ssl(8). Lisez la pour plus d'informations. Cette partie de la FAQ s'intéresse seulement à la génération d'un certificat RSA pour les serveurs Web. Elle ne décrit pas les certificats serveur DSA. Pour plus d'informations à ce sujet, lisez la page de manuel ssl(8).

Pour commencer, vous aurez besoin de créer votre clé serveur et le certificat en utilisant OpenSSL :

```
# openssl genrsa -out /etc/ssl/private/server.key 1024
```

Ou si vous voulez que la clé soit cryptée avec un mot de passe que vous devez saisir à chaque démarrage des serveurs

```
# openssl genrsa -des3 -out /etc/ssl/private/server.key 1024
```

La prochaine étape consiste à générer une requête de signature de certificat qui est utilisée pour permettre à une autorité de certification (CA) de signer votre certificat. Pour cela, utilisez la commande suivante :

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

Le fichier *server.csr* pourra alors être communiqué à une autorité de certification qui signera la clé. Une de ces autorités est **Thawte Certification** que vous pourrez joindre à l'adresse http://www.thawte.com/.

Si vous ne pouvez pas vous permettre un tel service, ou si vous voulez auto signer le certificat, vous pouvez utiliser la commande suivante :

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

Avec */etc/ssl/server.crt* et */etc/ssl/private/server.key*, vous devez être désormais capable de démarrer httpd(8) avec le drapeau **-DSSL** (consultez la section à propos de rc(8) dans cette faq), activant ainsi les transactions https sur le port 443 de votre machine.

10.7 - J'ai effectué des changements dans /etc/passwd avec vi(1),

mais les changements ne semblent pas être pris en compte. Pourquoi ?

Si vous éditez */etc/passwd*, vos modifications seront perdues. OpenBSD génère */etc/passwd* dynamiquement avec pwd_mkdb(8). Le fichier principal de mots de passe sous OpenBSD est */etc/master.passwd*. D'après pwd_mkdb(8),

```
FILES
     /etc/master.passwd  fichier courant de mots de passe
     /etc/passwd         fichier de mots de passe au format Version 7
     /etc/pwd.db         fichier non sécurisé de mots de passe au format base de données
     /etc/pwd.db.tmp     fichier temporaire
     /etc/spwd.db        fichier sécurisé de mots de passe au format base de données
     /etc/spwd.db.tmp    fichier temporaire
```

Dans un fichier de mots de passe Unix traditionnel, toutes les informations y compris le mot de passe crypté de l'utilisateur sont à la disposition de n'importe quel utilisateur du système (et c'est la cible principale de programmes tels que Crack). 4.4BSD a introduit le fichier master.passwd qui a un format étendu (avec les options additionnelles par rapport à /etc/passwd). Ce fichier n'est accessible que pour root. Pour un accès plus rapide aux données, les appels à la librairie qui utilisent ce type d'informations accèdent normalement à /etc/pwd.db et à /etc/spwd.db.

OpenBSD met à votre disposition un outil qui vous permet d'éditer le fichier de mots de passe. Cet outil s'appelle vipw(8). vipw utilisera vi (ou votre éditeur favori tel que défini par $EDITOR) pour éditer /etc/master.passwd. Suite à vos modifications, vipw recréera /etc/passwd, /etc/pwd.db, et /etc/spwd.db qui tiendront compte de vos modifications. vipw verrouille aussi l'accès à ces fichiers de telle façon à en interdire l'accès à quiconque essaie d'en changer le contenu en même temps que vous.

10.8 - Comment je crée un compte utilisateur ? Ou je supprime un compte utilisateur ?

OpenBSD offre deux commandes pour facilement créer des comptes utilisateurs sur le systè :

- adduser(8)
- user(8)

La manière la plus facile pour créer un compte utilisateur sous OpenBSD est d'utiliser le script adduser(8). Ce script est paramétrable à travers le fichier */etc/adduser.conf*. Bien que c'est la méthode recommandée pour créer des comptes utilisateurs, il est toujours possible de les créer à la main en utilisant vipw(8). adduser(8) permet d'effectuer des vérifications sur la cohérence de */etc/passwd*, */etc/group*, et les bases de données shell. adduser(8) crée pour vous les entrées correspondantes et les répertoires $HOME. Il peut aussi envoyer un message de bienvenue aux utilisateurs. Le comportement de ce programme peut être adapté à vos besoins. Pour illustrer notre propos, prenons comme exemple la création du compte **testuser**. Le répertoire de cet utilisateur sera */home/testuser*. L'utilisateur fera partie du groupe **guest** comme groupe et aura un shell */bin/ksh* .

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

Reading /etc/shells
Reading /etc/login.conf
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 []: testuser
Enter full name []: Test FAQ User
Enter shell csh ksh nologin sh [sh]: ksh
Uid [1002]: Enter
Login group testuser [testuser]: guest
```

```
Login group is ``guest''. Invite testuser into other groups: guest no
[no]: no
Login class auth-defaults auth-ftp-defaults daemon default staff
[default]: Enter
Enter password []: Type password, then Enter
Enter password again []: Type password, then Enter

Name:        testuser
Password:    ****
Fullname:    Test FAQ User
Uid:         1002
Gid:         31 (guest)
Groups:      guest
Login Class: default
HOME:        /home/testuser
Shell:i      /bin/ksh
OK? (y/n) [y]: y
Added user ``testuser''
Copy files from /etc/skel to /home/testuser
Add another user? (y/n) [y]: n
Goodbye!
```

Pour supprimer des comptes utilisateurs, utilisez la commande rmuser(8). Cette commande supprimera toute chose relative à l'utilisateur. Elle supprimera son entrée crontab(1), son répertoire $HOME (s'il lui appartient), et son courrier. Bien évidemment, cette commande supprimera aussi les entrées correspondantes dans */etc/passwd* et */etc/group*. Comme exemple, nous allons utiliser cette commande pour supprimer le compte utilisateur précédemment crée. Notez que la commande vous demande l'identifiant du compte et si oui ou non elle doit supprimer le répertoire home de l'utilisateur.

```
# rmuser
Enter login name for user to remove: testuser
Matching password entry:

testuser:$2a$07$ZWnBOsbqMJ.ducQBfsTKUe3PL97Ve1AHWJ0A4uLamniLNXLeYrEie:1002
:31::0:0:Test FAQ User:/home/testuser:/bin/ksh

Is this the entry you wish to remove? y
Remove user's home directory (/home/testuser)? y
Updating password file, updating databases, done.
Updating group file: done.
Removing user's home directory (/home/testuser): done.
```

Créer des comptes utilisateurs via user(8)

Ces outils sont moins interactifs que la commande adduser(8), ce qui en facilite l'usage dans des scripts.

La liste complète des outils est :

- group(8)
- groupadd(8)
- groupdel(8)
- groupinfo(8)
- groupmod(8)
- user(8)
- useradd(8)
- userdel(8)
- userinfo(8)
- usermod(8)

Création effective des comptes utilisateurs

Etant donné que la commande user(8) n'est pas interactive, la manière la plus simple et la plus efficace pour créer des comptes utilisateurs est d'utiliser la commande adduser(8). La commande */usr/sbin/user* est seulement une interface aux autres commandes */usr/sbin/user**. Ainsi, dans l'exemple qui suit il est possible d'utiliser soit **user add** soit **useradd**. Le choix est votre et ne change rien au résultat.

Dans cet exemple, nous allons créer un compte avec les mêmes spécificités que le compte crée précédemment. useradd(8) est bien plus facile à utiliser si vous connaissez les paramètres par défaut avant de créer un compte utilisateur. Ces paramètres se trouvent dans le fichier */etc/usermgmt.conf* et peuvent être visualisés comme suit :

```
$ user add -D
group           users
base_dir        /home
skel_dir        /etc/skel
shell           /bin/csh
inactive        0
expire          Null (unset)
range           1000..60000
```

Ces paramètres vont être appliqués à chaque nouveau compte si vous ne changez pas leur valeur en utilisant des options en ligne de commande. Par exemple, dans notre cas nous voulons que l'utilisateur appartienne au groupe **guest** et non pas à **users**. Il est à noter que lors de la création des comptes utilisateurs, les mots de passe doivent être spécifiés sous leur forme cryptée en ligne de commande. Vous devez donc utiliser, au préalable, l'utilitaire encrypt(1) pour créer le mot de passe. Par exemple : Les mots de passe par défaut sous OpenBSD utilisent l'algorithme Blowfish avec 6 réitérations. Voici un exemple d'utilisation de la commande encrypt :

```
$ encrypt -p -b 6
Enter string:
$2a$06$YOdOZM3.4m6MObBXjeZtBOWArqC2.uRJZXUkOghbieIvSWXVJRzlq
```

Maintenant que nous avons le mot de passe crypté, nous sommes prêts à créer le compte utilisateur :

```
# user add -p '$2a$06$YOdOZM3.4m6MObBXjeZtBOWArqC2.uRJZXUkOghbieIvSWXVJRzlq' -u 1002 \
-s /bin/ksh -c "Test FAQ User" -m -g guest testuser
```

Remarque : Assurez vous d'utiliser " pour englober le mot de passe. L'utilisation de "" ne permet pas d'empêcher le shell d'interpréter le jeu de caractères correspondant au mot de passe avant de les communiquer à user(8). De même, assurez vous d'utiliser l'option **-m** si vous voulez créer le répertoire $HOME de l'utilisateur et copier les fichiers à partir de */etc/skel* vers $HOME.

Pour voir si le compte utilisateur a été correctement crée, nous pouvons recourir à plusieurs utilitaires. Voici quelques commandes pour vérifier rapidement que tout s'est bien passé :

```
$ ls -la /home
total 14
drwxr-xr-x   5 root      wheel   512 May 12 14:29 .
drwxr-xr-x  15 root      wheel   512 Apr 25 20:52 ..
drwxr-xr-x  24 ericj     wheel  2560 May 12 13:38 ericj
drwxr-xr-x   2 testuser  guest   512 May 12 14:28 testuser
$ id testuser
uid=1002(testuser) gid=31(guest) groups=31(guest)
$ finger testuser
Login: testuser                         Name: Test FAQ User
Directory: /home/testuser               Shell: /bin/ksh
Last login Sat Apr 22 16:05 (EDT) on ttyC2
No Mail.
No Plan.
```

En plus de ces commandes, user(8) fournit son propre utilitaire, appelé userinfo(8), qui permet d'afficher les caractéristiques d'un compte utilisateur :

```
$ userinfo testuser
login   testuser
passwd  *
uid     1002
groups  guest
change  Wed Dec 31 19:00:00 1969
class
gecos   Test FAQ User
dir     /home/testuser
shell   /bin/ksh
expire  Wed Dec 31 19:00:00 1969
```

Suppression des comptes utilisateurs

Pour supprimer des comptes utilisateurs avec la hiérarchie de commandes user(8), vous devez utiliser userdel(8). Cette commande est simple et efficace. Pour supprimer le compte précédemment crée, utilisez :

```
# userdel -r testuser
```

Notez bien l'option **-r** qui doit être spécifiée si vous voulez supprimer les répertoires $HOME aussi. Si vous voulez juste bloquer l'accès au compte sans supprimer des informations liées au compte, utilisez **-p** au lieu de **-r**.

10.10 - Comment puis-je créer un compte pour ftp uniquement ?

Il y a plusieurs méthodes pour effectuer cette opération. Une des manières les plus communes est d'ajouter */usr/bin/false* dans le fichier */etc/shells*. Et quand vous créez le compte utilisateur, attribuez lui le shell /usr/bin/false. Il sera alors capable d'utiliser les fonctionnalités ftp sans pour autant être capable de se connecter de manière interactive. Par défaut, adduser(8) affecte à l'utilisateur le répertoire */home/<user>*. Si c'est ce que vous souhaitez, laissez-le tel quel. Cependant, vous pouvez affecter à l'utilisateur le répertoire que vous souhaitez. De même, vous pouvez confiner l'utilisateur à son répertoire $HOME en ajoutant son nom à */etc/ftpchroot*. En utilisant l'option **-A** de ftpd(8), seuls les comptes listés dans ce fichier seront autorisés à se connecter !

10.11 - Mise en place des quotas

Les quotas sont utilisés pour limiter l'espace disque disponible pour les utilisateurs. Ce système peut être très utile si vous avez des ressources limitées. Les quotas peuvent être configurés par utilisateur et/ou par groupe.

La première étape pour configurer les quotas est de s'assurer que `option QUOTA` est présente dans votre configuration noyau. Cette option est incluse dans le noyau GENERIC. Ensuite, vous aurez besoin de marquer les systèmes de fichiers où les quotas sont utilisés dans le fichier `/etc/fstab`. Les mots clés `userquota` et `groupquota` doivent être utilisés pour marquer chaque système de fichiers où les quotas sont activés. Par défaut, les fichiers `quota.user` et `quota.group` seront crées à la racine des systèmes de fichiers où les quotas sont utilisés pour stocker les informations relatives à ces derniers. Si vous voulez les créer ailleurs, spécifiez un fichier avec l'option des quotas dans `/etc/fstab`, par exemple "`userquota=/var/quotas/quota.user`". Voici un exemple de `/etc/fstab` avec un système de fichiers avec quotas activés et le fichier de quotas se trouvant dans un endroit non-standard :

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

Maintenant, il faut configurer les quotas par utilisateur. A cette fin, nous utilisons la commande edquota(8). Une utilisation simple est **edquota <user>**. edquota(8) va utiliser vi(1) pour éditer les quotas à moins que la variable d'environnement EDITOR est positionnée pour charger un autre éditeur. Par exemple la commande :

```
# edquota ericj
```

Affichera un résultat similaire à :

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

Pour ajouter des limites, éditer là pour donner un résultat similaire à :

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

Notez que l'allocation de quotas est en blocs de 1k. Dans ce cas-ci, softlimit est fixé à 1000k et hardlimit à 1050k. softlimit est une limite qui permet au système de prévenir les utilisateurs quand ils l'ont dépassé. Ils auront alors jusqu'à la fin de leur période de grâce pour redescendre en dessous de cette limite. Les périodes de grâce peuvent être configurées à l'aide de l'option **-t** de edquota(8). Après la fin de la période de grâce, softlimit est géré comme hardlimit. Ce qui cause un échec d'allocation.

Une fois les quotas configurés, il faut les activer. Pour cela, utilisez la commande quotaon(8). Par exemple :

```
# quotaon -a
```

Cette commande analysera le contenu de `/etc/fstab` et activera les quotas sur les systèmes de fichiers où les options de quota sont positionnées. Maintenant que les quotas sont activés, vous pouvez les visualiser à l'aide de quota(1). Ainsi, la commande **quota <user>** fournit les informations concernant cet utilisateur. Si aucun argument n'est utilisé, quota vous fournira des statistiques sur les quotas. Par exemple :

```
# quota ericj
```

Afficherait :

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

Par défaut, les quotas positionnés dans `/etc/fstab` seront activés au démarrage. Pour les désactiver, utilisez :

```
# quotaoff -a
```

10.12 - Mise en place de Clients et Serveurs KerberosV

OpenBSD inclut KerberosV comme un composant pré-installé sur le système par défaut.

Pour plus d'information concernant KerberosV, sur votre système OpenBSD, utilisez la commande :

```
# info heimdal
```

10.13 - Mise en place d'un serveur FTP Anonyme

Le mode FTP anonyme permet à des utilisateurs sans compte d'accéder aux fichiers sur votre machine en utilisant le protocole de transfert de fichiers. Ce chapitre a pour but de fournir une vue d'ensemble de la configuration d'un serveur FTP anonyme, les logs générés, ...etc.

Création du compte FTP

La première étape consiste à créer un compte "ftp" sur votre système. Ce compte ne doit pas avoir de mot de passe utilisable. Dans cet exemple, nous allons considérer que /home/ftp est le répertoire correspondant au compte "ftp" mais vous n'êtes pas obligé de choisir la même chose. Quand le mode anonyme est utilisé, le démon ftp va se confiner au répertoire HOME de l'utilisateur 'ftp' (dans notre cas, ce répertoire est /home/ftp). Pour en savoir plus, lisez les pages du manuel ftp(8) et chroot(2). Voici un exemple de création du compte *ftp* en utilisant la commande adduser(8). Au préalable, nous avons besoin d'ajouter /usr/bin/false au fichier */etc/shells*. C'est le shell que nous allons attribuer à l'utilisateur ftp. Il ne permettra pas de connexion en login à ce compte même si nous configurons un mot de passe vide. Pour effectuer cette opération, il suffit de faire *echo /usr/bin/false >> /etc/shells*. Si en plus vous souhaitez que ce shell apparaisse dans la liste des choix proposés par adduser, vous devez modifier le fichier */etc/adduser.conf*.

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

Reading /etc/shells
Reading /etc/login.conf
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 []: ftp
Enter full name []: anonymous ftp
Enter shell csh false ksh nologin sh tcsh zsh [sh]: false
Uid [1002]: Enter
Login group ftp [ftp]: Enter
Login group is ``ftp''. Invite ftp into other groups: guest no
[no]: no
Login class auth-defaults auth-ftp-defaults daemon default staff
[default]: Enter
Enter password []: Enter
Set the password so that user cannot logon? (y/n) [n]: y

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

Configuration du répertoire

L'opération a crée, en plus de l'utilisateur, le répertoire */home/ftp*. C'est ce que nous voulons mais nous avons besoin d'effectuer quelques modifications pour préparer le système à héberger le service FTP anonyme. Ces modifications sont expliquées dans la page du manuel ftp(8).

Vous n'avez pas besoin de créer un répertoire /home/ftp/usr ou /home/ftp/bin.

- */home/ftp* - C'est le répertoire principal. Il doit être possédé par root avec les permissions 555.
- */home/ftp/etc* - Ce répertoire est optionnel et non recommandé. Son seul but est de fournir des informations sur les comptes utilisateurs existant. Si vous voulez que les fichiers de votre répertoire de ftp soient associés à de vrais utilisateurs, vous devez copier /etc/pwd.db et /etc/group dans ce répertoire. Les permissions du répertoire doivent être 511. Les permissions sur les deux fichiers doivent être 444. Ils sont utilisés pour fournir une correspondance entre des nombres et les noms attribués aux comptes utilisateurs et groupes. Il n'y a pas de mots de passe dans pwd.db. Tous les mots de passe sont stockés dans spwd.db alors ne copiez pas ce fichier.
- */home/ftp/pub* - C'est le répertoire standard pour mettre les fichiers que vous voulez partagez. Ce répertoire doit avoir les

permissions 555.

Il est à noter que tous ces répertoires doivent être la propriété de "root". Voici à quoi doivent ressembler les répertoires après leur création :

```
# pwd
/home
# 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 11:34 etc
dr-xr-xr-x  2 root  ftp    512 Jul  6 11:33 pub

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 ..
```

Démarrage du serveur et logs

Vous pouvez choisir d'exécuter ftpd soit à partir de inetd soit directement de le lancer directement via les scripts rc. L'exemple ci-après représente le cas où le service est démarré à partir de inetd.conf. Tout d'abord, nous devons nous familiariser avec quelques options de ftpd. La ligne par défaut dans */etc/inetd.conf* est :

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

Comme vous pouvez le voir, ftpd est invoqué avec *-US*. Ces options vont permettre de loguer les connexions anonymes dans */var/log/ftpd* et les sessions courantes dans */var/run/utmp*. Ce qui permet de voir ces sessions via who(1). Dans certains cas, on souhaitera fournir un accès anonyme et désactiver ftp pour les utilisateurs du système. Pour cela, il faut utiliser l'option *-A* de ftpd. Voici une ligne d'invocation de ftpd en mode anonyme exclusif. On utilise aussi *-ll* qui logue chaque connexion vers syslog en plus des commandes ftp get, retrieve, etc.

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

Remarque - Les personnes gérant des serveurs ftp à HAUT trafic ne devraient pas invoquer ftpd à partir de inetd.conf. La meilleure option consiste à commenter la ligne correspondant à ftpd dans /etc/inetd.conf et à démarrer ftpd à partir de rc.conf avec l'option *-D*. Ce qui va démarrer ftpd en tant que démon. Ce mode de fonctionnement est beaucoup moins coûteux et plus rapide que le démarrage via inetd. La ligne correspondant à ftpd dans rc.conf ressemblerait à :

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

Bien évidemment, cette méthode ne fonctionnera que si ftpd est commenté dans */etc/inetd.conf* et en veillant qu'inetd ait bien relu sont fichier de configuration.

Autres fichiers importants

- */etc/ftpwelcome* - Ce fichier contient le message de bienvenue qui sera affiché aux personnes qui se connectent sur votre serveur ftp.
- */etc/motd* - Ce fichier contient le message qui sera affiché aux utilisateurs une fois authentifiés sur votre serveur ftp.

- *.message* - Ce fichier peut être mis dans n'importe quel répertoire. Il contient un message qui sera affiché lorsque l'utilisateur entre dans le répertoire où ce fichier se trouve.

10.13 - Confiner les utilisateurs à leur répertoire HOME avec ftpd(8)

le démon ftpd(8) livré avec OpenBSD est configuré par défaut pour gérer cela de manière très facile. Cette action est accomplie par le biais du fichier **/etc/ftpchroot**. Puisqu'on ne peut pas toujours faire confiance aux utilisateurs, il sera peut-être nécessaire de les confiner à leur répertoire HOME. Ce mode de fonctionnement n'est pas activé par défaut. Voici à quoi ressemble le mode par défaut :

```
$ ftp localhost
Connected to localhost.
220 oshibana FTP server (Version 6.4/OpenBSD) ready.
Name (localhost:ericj): ericj
331 Password required for ericj.
Password: *********
230- OpenBSD 3.4 (GENERIC) #18: Wed Sep 17 03:34:47 MDT 2003
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> cd /
250 CWD command successful.
ftp> ls
227 Entering Passive Mode (127,0,0,1,60,7)
150 Opening ASCII mode data connection for 'file list'.
altroot
bin
dev
etc
home
mnt
root
sbin
stand
tmp
usr
var
bsd
sys
boot
226 Transfer complete.
ftp> quit
221 Goodbye.
```

Comme vous pouvez le voir, l'accès est permis à tout le serveur. Dans un monde parfait où on peut faire confiance aux utilisateurs, il n'y a pas de mal à ça mais c'est loin d'être le cas dans la réalité. Pour confiner un utilisateur à son répertoire HOME, il suffit simplement d'ajouter son nom au fichier **/etc/ftpchroot**. Voici un exemple pour montrer cette restriction appliquée à l'utilisateur "ericj" :

```
$ cat /etc/ftpchroot
#       $OpenBSD: faq10.html,v 1.18 2004/03/26 08:40:41 jufi Exp $
#
# list of users (one per line) given ftp access to a chrooted area.
# read by ftpd(8).
```

```
ericj
```

Cette opération est suffisante pour empêcher l'utilisateur "ericj" de sortir de son propre répertoire comme vous pouvez le voir à travers l'exemple suivant. Le répertoire / correspond maintenant à son répertoire HOME !

```
$ ftp localhost
Connected to localhost.
220 oshibana FTP server (Version 6.4/OpenBSD) ready.
Name (localhost:ericj): ericj
331 Password required for ericj.
Password: *********
230 User ericj logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /
250 CWD command successful.
ftp> ls
227 Entering Passive Mode (127,0,0,1,92,171)
150 Opening ASCII mode data connection for 'file list'.
.login
.mailrc
.profile
.rhosts
.ssh
.cshrc
work
mail
src
226 Transfer complete.
ftp> quit
221 Goodbye.
```

10.15 - Appliquer des correctifs sous OpenBSD

Le code source de OpenBSD est en constante évolution et amélioration. En même temps, des correctifs aux problèmes les plus communs sont crées et diffusés au public. Ces correctifs apparaissent sur la page web des errata. Ils sont séparés en catégories. Ces catégories correspondent à des correctifs qui concernent différentes architectures ou à des correctifs indépendants de l'architecture.

Cependant, il est à noter qu'il n'y a pas de correctifs pour les nouvelles fonctionnalités ajoutées à OpenBSD. Les correctifs corrigent uniquement des problémes de stabilité ou de sécurité qui doivent être réglés très rapidement, bien que le choix final revienne à l'administrateur.

Comme exemple, nous allons appliquer un correctif de sécurité obtenu à partir de la page web des errata à talkd(8).

Quelle est la différence entre ces correctifs et ce qui se trouve dans l'arborescence CVS ?

Tous les correctifs postés sur la page web des errata concernent uniquement l'arborescence des sources de la dernière version diffusée. Les autres correctifs qui concernent l'arborescence actuelle de CVS peuvent contenir certaines modifications qui ne sont peut-être pas désirables sur la version de production.

Préparer le système pour l'application des correctifs

Les correctifs d'OpenBSD sont distribués sous la forme de fichiers diff. Ces fichiers sont des fichiers texte qui contiennent les différences par rapport au code source d'origine. Ils ne sont **PAS** distribués sous forme binaire. Cela veut dire que pour appliquer les correctifs, vous devez avoir à disposition sur votre système le code source de la version **RELEASE** d'OpenBSD. Cela ne veut pas dire que vous avez besoin de tout le code source du système d'exploitation OpenBSD pour appliquer les correctifs à votre système. Mais vous devez avoir le code source correspondant au composant à corriger. Par exemple, si vous corrigez le noyau vous devez avoir tout le code source au noyau.

cvs(1) est un outil très pratique pour télécharger uniquement les sources dont vous avez besoin en utilisant des serveurs cvs anonymes situés un peu partout dans le monde. Vous pouvez avoir une liste des serveurs disponibles dans la page CVS Anonyme.

Pour télécharger le code source de talkd(8) correspondant à la version *3.4-release* via cvs(1), vous utiliseriez les lignes de commande suivantes :

```
$ export CVSROOT=anoncvs@anoncvs5.usa.openbsd.org:/cvs
$ cvs co -rOPENBSD_3_4_BASE src/libexec/talkd/
cvs server: Updating src/libexec/talkd
U src/libexec/talkd/announce.c
U src/libexec/talkd/talkd.c
U src/libexec/talkd/talkd.h
```

Pour connaître le chemin CVS du code dont vous avez besoin, reportez- vous à la ligne *Index:*. Dans notre cas, le chemin CVS est *src/libexec/talkd/*. Il faut toujours vérifier la révision de OPENBSD_version_number_BASE. Sans "_BASE", vous obtiendrez le code correspondant à la branche stable qui contient probablement d'autres modifications qui peuvent interférer avec la procédure d'application des correctifs. Si vous synchroniser vos sources par rapport à la branche stable, les correctifs sont normalement déjà inclus dans le code source. Cependant, nous vous conseillons de toujours effectuer une vérification. Vous pouvez voir dans la page consacrée aux changements dans OpenBSD-current les correctifs qui ont été appliqués à la branche stable. Si les correctifs n'ont pas encore été appliqués, vous devez télécharger les sources de la dernière RELEASE en utilisant les commandes précédentes.

Pour les utilisateurs ayant acheté des CDs OpenBSD officiels, vous pouvez obtenir le code source directement à partir du CD. Reportez vous à l'encart livré avec votre CD. Vous n'aurez à ce moment-là besoin d'obtenir les sources via anoncvs.

```
Apply by doing:
        cd /usr/src
        patch -p0 < 026_talkd.patch
        cd libexec/talkd
        make obj && make depend && make && make install

Index: libexec/talkd/announce.c <------ Chemin CVS
===================================================================
RCS file: /cvs/src/libexec/talkd/announce.c,v
retrieving revision 1.8
retrieving revision 1.9
diff -u -r1.8 -r1.9
--- libexec/talkd/announce.c    1998/08/18 03:42:10     1.8
+++ libexec/talkd/announce.c    2000/07/06 00:01:45     1.9
@@ -160,6 +160,6 @@
                *(bptr++) = '\n';
        }
        *bptr = '\0';
-       fprintf(tf, big_buf);
+       fprintf(tf, "%s", big_buf);
        fflush(tf);
 }
```

Une fois que vous avez obtenu les sources, vous pouvez télécharger le correctif que vous mettrez sous *src/*

Application des correctifs

```
$ cd /usr/src
$ patch -p0</path/to/026_talkd.patch
Hmm...  Looks like a unified diff to me...
The text leading up to this was:
--------------------------
|Apply by doing:
|       cd /usr/src
```

```
|       patch -p0 < 026_talkd.patch
|       cd libexec/talkd
|       make obj && make depend && make && make install
|
|Index: libexec/talkd/announce.c
|===================================================================
|RCS file: /cvs/src/libexec/talkd/announce.c,v
|retrieving revision 1.8
|retrieving revision 1.9
|diff -u -r1.8 -r1.9
|--- libexec/talkd/announce.c   1998/08/18 03:42:10     1.8
|+++ libexec/talkd/announce.c   2000/07/06 00:01:45     1.9
--------------------------
Patching file libexec/talkd/announce.c using Plan A...
Hunk #1 succeeded at 160. <------------ Patch Succeeded
done
$ cd /usr/src/libexec/talkd/
$ ls
CVS             announce.c      print.c         table.c         talkd.c
Makefile        announce.c.orig process.c       talkd.8         talkd.h
$ make obj && make depend && make
making /home/ericj/lsrc/src/libexec/talkd/obj
mkdep -a /home/ericj/lsrc/src/libexec/talkd/talkd.c /home/ericj/lsrc/sr
c/libexec/talkd/announce.c /home/ericj/lsrc/src/libexec/talkd/process.c
/home/ericj/lsrc/src/libexec/talkd/table.c /home/ericj/lsrc/src/libexec
/talkd/print.c
cc -O2     -c /home/ericj/lsrc/src/libexec/talkd/talkd.c
cc -O2     -c /home/ericj/lsrc/src/libexec/talkd/announce.c
cc -O2     -c /home/ericj/lsrc/src/libexec/talkd/process.c
cc -O2     -c /home/ericj/lsrc/src/libexec/talkd/table.c
cc -O2     -c /home/ericj/lsrc/src/libexec/talkd/print.c
cc   -o ntalkd talkd.o announce.o process.o table.o print.o
nroff -Tascii -mandoc /home/ericj/lsrc/src/libexec/talkd/talkd.8 > talk
d.cat8
$ sudo make install
install -c -s -o root -g bin  -m 555 ntalkd /usr/libexec
install -c -o root -g bin -m 444 talkd.cat8 /usr/share/man/cat8/talkd.0
/usr/share/man/cat8/ntalkd.0 -> /usr/share/man/cat8/talkd.0
```

Une fois cette opération effectuée, redémarrez le service.

10.16 - Parlez moi de chroot() Apache ?

Sous OpenBSD, le serveur httpd(8) d'Apache est chroot(2)é par défaut. Etant un grand pas en avant du point de vue de la sécurité, cela peut créer des problè si vous n'y êtes pas préparé.

Qu'est-ce qu'un chroot ?

Une application chroot(2)ée est bloquée dans un répertoire spécifique et ne peut errer dans les autres répertoires de l'arbre du système de fichiers et voit ce répertoire comme sont répertoire / (root). Dans le cas d'httpd(8), le programme démarre, ouvre ses fichiers log, ouvre ses ports TCP (bien qu'il n'accepte pas encore de données), et lis sont fichier de configuration. Ensuite, il se fixe lui-même dans le répertoire */var/www* et baisse ses privilèges. Ce qui veut dire que tous les fichiers servis et utilisés par Apache, doivent être dans le répertoire */var/www*. Cela aide considérablement la sécurité -- si il devait y avoir un problème de sécurité, les dégats seraient confinés dans un seul répertoire avec les permissions de "lecture seule" et aucune ressource pour causer des problèmes.

Qu'est-ce que cela signifie pour l'utilisateur ?

En gros, chroot(2)er Apache est quelque chose de nouveau, et donc beaucoup d'applications et de configurations système ne fonctionneront plus comme avant.

- **Hiérarchie historique du système de fichiers :** Les serveurs mis à jour depuis d'anciennes versions d'OpenBSD peuvent avoir les fichiers web situés dans les répertoires HOME des utilisateurs, ce qui clairement ne fonctionnera pas dans un environnement chroot(2)é étant donné qu'httpd(8) ne peut atteindre le répertoire */home*. Les administrateurs découvriront peut-être que leur partition /var/www existente est trop petite pour accueillir tous les fichiers web. Vos options sont dès lors de restructurer votre système ou de ne pas utiliser l'option du chroot(2). Vous pouvez, bien evidemment, utiliser des liens symboliques dans le répertoire HOME de l'utilisateur pointant vers les sous-répertoires dans */var/www*, mais vous ne pouvez PAS utiliser des liens dans */var/www* pointant vers une autre partie du système de fichier -- ceci ne peut fonctionner dû au chroot(2). Notez que si vous voulez que vos utilisateurs aient un accès FTP chroot(2)é, ceci ne fonctionnera pas plus étant donné que le chroot FTP va (à nouveau) vous empêcher d'accéder aux destinations de ces liens symboliques. Une solution est donc, de ne pas utiliser */home* comme répertoire HOME pour ces utilisateurs mais plutot quelque chose similaire à */var/www/home*.
- **Rotation des fichiers log :** Normalement, une rotation des fichiers log est réalisée en renommant les anciens fichiers, ensuite en envoyant à httpd(8) un signal SIGUSR1 qui forcera Apache à fermer ses anciens fichiers logs et à en ouvrir de nouveaux. Ceci n'est désormais plus possible, étant donné qu'httpd(8) n'a plus la possibilité d'ouvrir ses propres fichiers log une fois que ses privilèges ont baissés. httpd(8) doit donc être stoppé et relancé :

  ```
  # apachectl stop && apachectl start
  ```

  Il existe néanmoins d'autres techniques, notamment logguer vers un pipe(2), et utiliser un programme extérieur afin de réaliser une rotation des fichiers à la fin du pipe.
- **Modules Apache existant :** Pratiquement, ils se lanceront tous, cependant, certains pourraient ne pas fonctionner correctement dans le chroot(2), et pourraient causer des problèmes lors de " `apachectl restart`", générant une erreur, causant httpd(8) à se stopper.
- **CGIs existant :** La plupart ne fonctionneront pas tels quels. Ils auront certainement besoin de programmes ou de librairies se trouvant hors de */var/www*. Certains peuvent être corrigés en étant compilés statiquement (n'ayant pas besoin de librairies se trouvant dans un autre répertoire), la plupart peuvent être corrigés en copiant dans */var/www* les fichiers nécessaires à l'application, bien que cette manoeuvre soit non triviale et requiert certaines notions de programmation -- la plupart des utilisateurs trouveront plus facile de ne pas employer le chroot(2) en attendant une mise à jour de ces CGIs.

Dans certains cas, les applications ou les configurations peuvent être changées pour fonctionner dans le chroot. Dans d'autres cas, vous devrez tout simplement retirer cette option en utilisant l'option `u` d'Apache dans *rc.conf*

10.17 - Je n'aime pas le shell root standard !

Le shell par défaut sur OpenBSD de l'utilisateur *root* est csh, dû principalement à la tradition. Il n'est pas spécialement requis qu'OpenBSD ait csh(1) comme shell pour l'utilisateur root (lisez néanmoins avant de le changer).

Certains utilisateurs venant d'un système d'exploitation "Unix-like" trouvent csh(1) peu familier et demande dès lors si ils peuvent le changer et comment. Il y a plusieurs options :

- **Ne vous authentifiez pas directement en tant que *root*** Entre su et sudo, il ne devrait y avoir que peu de raisons pour que les utilisateurs s'authentifient en tant que *root* pour la plupart des applications après l'installation initiale.
- **Invoquez votre shell favori après le login :** Si vous aimez ksh(1) ou n'importe quel autre shell, invoquez le simplement depuis votre shell par défaut.
- **Changez le shell de l'utilisateur root :** Ceci peut être fait en utilisant chsh ou vipw.

Une directive Unix traditionelle est d'utiliser pour l'utilisateur root des shells compilés statiquement, car si votre système démarre en mode utilisateur unique, les partitions non-root ne seront pas montées et les shells liés dynamiquement ne seront pas capable d'accéder aux librairies situées dans la partition `/usr`. Ceci n'étant pas très important pour OpenBSD, car le système vous demandera de choisir un shell lors d'un démarrage en mode utilisateur unique, le shell par défaut étant sh. Les trois shells standards sous OpenBSD (csh, sh et ksh) sont liés statiquement et donc utilisables en mode utilisateur unique.

Il est parfois dit qu'il ne faut pas changer le shell de l'utilisateur root, bien qu'il n'y ait aucune raison de ne pas le faire sous OpenBSD. Mais encore une fois, ceci ne devrait pas être une raison -- ne vous authentifiez pas directement en tant que root.

10.18 - Que puis-je faire d'autre avec *ksh* ?

Sous OpenBSD, ksh est pdksh, le Shell Korn du Domaine Public (Public Domain Korn Shell), et est le même binaire que sh.

Les utilisateurs qui sont à l'aise avec *bash*, souvent utilisé sur les systèmes Linux, trouveront probablement ksh très familier. Ksh(1) offre la plupart des options habituellement utilisées avec *bash*, notamment l'achèvement des commandes avec la touche tab, l'édition de la ligne de commande et l'historique via les touches flèchées, et CTRL-A/CTRL-E pour aller au début/à la fin de la ligne de commande. Si vous désirez d'autres options de *bash*, *bash* peut être installé soit via les ports ou soit via les packages.

Le prompt de *ksh* peut être facilement changé de manière à fournir plus d'informations que le simple "$ " par défaut en modifiant la variable `PS1`. Par exemple, en insérant la ligne suivante :

```
export PS1='$PWD $ '
```

dans votre `/etc/profile`, cela produira le prompt suivant :

```
/home/nick $
```

Consultez le fichier `/etc/ksh.kshrc`, qui inclut plusieurs options utiles ainsi que des exemples, et qui peut être invoqué dans les fichiers `.profile` de vos utilisateurs.

[Index de la FAQ] [Section 9 - Migrer depuis Linux] [Section 11 - Optimisation des Performances]


www@openbsd.org

Originally [OpenBSD: faq10.html,v 1.100]
$Translation: faq10.html,v 1.23 2004/03/25 21:03:18 saad Exp $
$OpenBSD: faq10.html,v 1.18 2004/03/26 08:40:41 jufi Exp $

OpenBSD

[FAQ Index] [To Section 10 - System Management] [To Section 12 - Platform-Specific Questions]

11 - Performance Tuning

Table of Contents

11.1 - 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 or when your disks fail. OpenBSD has several options to increase the speed of your disk operations and provide fault tolerance.

Table Of Contents

11.1.1 - 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 (in `/usr/src/sys/conf/GENERIC`). If you have customized your kernel, you may need to return it to your kernel configuration. Either way, a line such as this should be in your configuration file:

```
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 which partitions on your real disks 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 usable partitions are at least one cylinder off from the beginning of the disk.

11.1.2 - 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, so 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 available, and a detailed explanation is beyond the scope of this document.

11.1.3 - 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(8) 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, as 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.

11.1.4 - Size of the namei() cache

Note: previously, the options(4) manual page recommended to set the `NVNODE=integer` kernel option. This is no longer recommended; you should now use the sysctl(8) command instead.

The name-to-inode translation (a.k.a., namei()) cache controls the speed of pathname to inode(5) translation. A reasonable way to derive a value for the cache, should a large number of namei() cache misses be noticed with a tool such as systat(1), is to examine the system's current computed value with sysctl(8), (which calls this parameter "`kern.maxvnodes`") and to increase this value until either the namei() cache hit rate improves or it is determined that the system does not benefit substantially from an increase in the size of the namei() cache. After the value has been determined, you can set it at system startup time with sysctl.conf(5).

11.2 - 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 IV or AMD Athlon processor is much better than the Price/Performance ratio of a Sun UltraSparc 60! Of course, the price of OpenBSD can't be beaten.

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, and 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 large disk, buy multiple smaller 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 if you need very high disk IO speeds.** IDE disks normally run at 5400 RPM to 7200 RPM. Using high end IDE disks, it may be unreasonable to expect more than 15 to 20 megabytes per second of throughput from a single disk. Using high end SCSI disks (higher cost 10k RPM to 15k RPM disks), you can achieve performance higher than this. Conversely, if you are using medium or low end SCSI disks, this is a waste of money, and IDE will serve you just as well, if not better.

  If you are building a server, and you need more than one drive, you may want to consider SCSI. IDE limits you to two disks per controller. Concurrent access to these two disks may have a negative impact on the I/O performance of these disks. Wide SCSI limits you to 15 per controller, and has better support for concurrent access than IDE. While SCSI costs more, the flexibility and performance can justify these costs in some environments.

- **Use SDRAM instead of DRAM.** 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. You will get better performance with SDRAM versus DRAM (SIMMs). If your system supports RDRAM, DDR or some other new type of RAM, then you are even further ahead...

- **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 into 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.

- **Avoid cheap PCI network adapters.** OpenBSD supports a plethora of cheap PCI network adapters. These adapters work great in home systems, and also low or moderate throughput business and research environments. But, if you need high throughput and low impact on your server, you are better off buying a quality PCI network adapter. Unfortunately, some expensive brand adapters (such as the 3com XL series) are not much better than the cheap adapters. One favourite 10/100Mbps adapter is the Intel EtherExpress PRO/100.

11.3 - 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) usable. Why is async mounting frowned upon and not on by default (as it is in some other unixen)? Isn't it a much simpler, and therefore, a safer way of improving performance in some applications?"

Answer: "Async mounts is indeed faster than 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 reliability and the stability of the system. Check the man page 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 can gain speed by using a separate partition for that data only, mounted async. Again, do this *only if* you don't mind the loss of all the data in the partition when something goes wrong. For this reason, mfs(8) partitions are mounted asynchronously, as they will get wiped and recreated on a reboot anyway.

11.4 - Tuning your monitor resolution under XFree86

Note: Most users do NOT need to worry about manually creating a ModeLine in modern versions of X. HOWEVER, sometimes it is needed for unusual situations.

Getting an X server working at an acceptable resolution with many multi-sync monitors is possible. If anyone has tried to do this with the standard xf86config or XF86Setup utilities, they probably didn't get the best possible results. One of the more painful aspects is simply getting your monitor running with your preferred resolution, and then getting the vertical scan rate set to at least 72-75 Hz, a rate where the screen flicker is much less visible to humans. Conversely, what if you are trying to set the vertical scan rate so it is very low? You can set it at 50 Hz so that it can be captured on to video without flicker, but the methods to do this are non-obvious with the standard XFree86 tools and documentation.

Finally, at the resolutions many people normally use with inexpensive VGA monitors (800x600, 1024x768, 1152x900,

1280x1024), it is possible (at least on newer monitors) to use vertical scan rates of 85Hz and above, to achieve an extremely clean, palatable picture. The XFree86 X server has a mechanism which allows you to describe in detail the video mode you want to use, this is the ModeLine. A ModeLine has four sections, a single number for the pixel clock, four numbers for horizontal timings, four numbers for vertical timings, and an optional section with a list of flags specifying other characteristics of the mode (such as Interlace, DoubleScan, and more... see the XF86Config(5) manual page for more ModeLine details).

Generating a ModeLine is a black art... Luckily, there are several scripts which can do this for you. One is Colas XFree86 ModeLine Generator. Another is The XFree86 Modeline Generator hosted at SourceForge, and there are several others available on Freshmeat. Before you can use these ModeLine generators, you need to figure out the vertical and horizontal sync limits for your monitor. This is often documented in the manual, or on the manufacturer's web site. If you can't find either of those, simply search the web for the monitor make and model, several people have been kind enough to compile lists with this information.

For example, say you have a Dell D1226H monitor. You searched in agony at Dell's web site to find that it has a 30-95 kHz horizontal scan range, and a 50-160 Hz vertical scan range. Visit the ModeLine generator page, enter this information. Next, you need to enter the minimum vertical scan rate you want. Any rate at or above 72 Hz should generally have low visible flicker. As you go higher, the clearer and crisper your screen image becomes.

With all of these bits of information, the script will generate a ModeLine for every possible 4x3 resolution which your monitor can support, above the minimum vertical scan rate which you enter. If someone enters the Dell specs above and a 75 Hz vertical scan minimum, the script gives out something like the following:

```
ModeLine "320x240" 20.07 320 336 416 448 240 242 254 280 #160Hz
ModeLine "328x246" 20.86 328 344 424 456 246 248 260 286 #160Hz
...
ModeLine "816x612" 107.39 816 856 1056 1136 612 614 626 652 #145Hz
ModeLine "824x618" 108.39 824 864 1064 1144 618 620 632 658 #144Hz
ModeLine "832x624" 109.38 832 872 1072 1152 624 626 638 664 #143Hz
...
ModeLine "840x630" 109.58 840 880 1080 1160 630 632 644 670 #141Hz
ModeLine "848x636" 110.54 848 888 1088 1168 636 638 650 676 #140Hz
...
ModeLine "1048x786" 136.02 1048 1096 1336 1432 786 788 800 826 #115Hz
ModeLine "1056x792" 136.58 1056 1104 1344 1440 792 794 806 832 #114Hz
ModeLine "1064x798" 137.11 1064 1112 1352 1448 798 800 812 838 #113Hz
...
ModeLine "1432x1074" 184.07 1432 1496 1816 1944 1074 1076 1088 1114 #85Hz
ModeLine "1576x1182" 199.86 1576 1648 2008 2152 1182 1184 1196 1222 #76Hz
ModeLine "1584x1188" 198.93 1584 1656 2016 2160 1188 1190 1202 1228 #75Hz
```

Now, this monitor claims to do 1600x1200 @ 75 Hz, but the script does not say this is within 75 Hz. So, if you really want exactly 1600x1200, go down a notch with your minimum vertical rate... (Here, we go down to 70 Hz)

```
ModeLine "1592x1194" 197.97 1592 1664 2024 2168 1194 1196 1208 1234 #74Hz
ModeLine "1600x1200" 199.67 1600 1672 2032 2176 1200 1202 1214 1240 #74Hz
ModeLine "1608x1206" 198.65 1608 1680 2040 2184 1206 1208 1220 1246 #73Hz
ModeLine "1616x1212" 197.59 1616 1688 2048 2192 1212 1214 1226 1252 #72Hz
ModeLine "1624x1218" 199.26 1624 1696 2056 2200 1218 1220 1232 1258 #72Hz
ModeLine "1632x1224" 198.15 1632 1704 2064 2208 1224 1226 1238 1264 #71Hz
ModeLine "1640x1230" 199.81 1640 1712 2072 2216 1230 1232 1244 1270 #71Hz
ModeLine "1648x1236" 198.64 1648 1720 2080 2224 1236 1238 1250 1276 #70Hz
```

Here, we see the monitor really does 1600x1200 @ 74 Hz when the dot clock (bandwidth) is limited to 200MHz. Set the bandwidth according to the limits defined by the monitor.

Once you have your ModeLines, put them into your /etc/X11/XF86Config file. Comment out the old ModeLines, so that you can

use them again if the new ones don't work. Next, choose what resolution you actually want to run at. First, figure out if X is running in accelerated mode (which it does with most video cards), so you know which "Screen" section of the XF86Config to modify. Or, just modify all of the Screen sections.

```
Section "Screen"
   Driver          "Accel"
   Device          "Primary Card"
   Monitor         "Primary Monitor"
   DefaultColorDepth 32
   SubSection "Display"
      Depth        32
      Modes        "1280x1024" "1024x768"
   EndSubSection
```

The first resolution you see after the "Modes" keyword is the resolution that X is going to start in. By pressing CTRL-ALT-KEYPAD MINUS, or CTRL-ALT-KEYPAD PLUS, you can switch between any resolutions that you list here. According to the section above, X will try to start in 32-bit color mode (via the DefaultColorDepth directive, without it X will start in 8-bit color mode.) The first resolution it will try to use is 1280x1024 (it follows the order of the Modes line.) Note that "1280x1024" is just a label for the values in the ModeLine.

Note that the ModeLine generator script has options to relax its timings for older or smaller monitors, and also has the ability to provide ModeLines for specific resolutions. Depending on the type of hardware you have, it may not be very easy to use with the default options. If the picture is too tall, too wide, or too small, or is shifted horizontally or vertically, and the controls of the monitor aren't enough to correct its appearance, once can use xvidtune(1) to adjust the ModeLine to better fit with the monitor's timings.

On most modern monitors, there is no fixed limit on the bandwidth, thus they are often not listed anymore in the specs. What happens is that the more you go up in bandwidth, the fuzzier the screen image becomes. So you may want to put in the bandwidth of your card (also named "dotclock") to test (you cannot damage your monitor this way), and go progressively down in BW down to have a nice crisp image.

If this seems needlessly complex, that's because it is. XFree86 4.0 addresses this, and makes this process much easier since it has several built-in modes and is capable of reading back capabilities from "plug and play" monitors through DDC and DDC2.

You can download the Colas XFree86 ModeLine Generator script at: http://koala.ilog.fr/ftp/pub/Klone/. You need to grab the Klone interpreter, and compile it. It is in the ports as `lang/klone`. The scripts exist under the scripts directory in the Klone distribution. (The port installs them to /usr/local/lib/klone/scripts.)

There are two versions of the script included, the first is a CGI version identical to the web page above. The second is a non-CGI version which will take your complete XF86Config file, decode the monitor specs that you entered into xf86config/XF86Setup (Now, think, did you actually enter the specs for your monitor or just choose generic ones?), and fix the existing ModeLines accordingly.

[FAQ Index] [To Section 10 - System Management] [To Section 12 - Platform-Specific Questions]


www@openbsd.org

$OpenBSD: faq11.html,v 1.50 2004/04/21 00:02:10 nick Exp $

[FAQ Index] [To Section 11 - Performance Tuning] [To Section 14 - Disk Setup]

12 - Platform-Specific Questions

Table of Contents

12.1 - General hardware notes

12.1.1 - PCI devices

- PCI devices are mostly self-configuring -- the computer and OS will allocate resources to the cards as required.
- Interrupts can be shared on the PCI bus. Not only can they be, the system will often perform better when the IRQs are shared, especially on i386 systems.
- There are several different PCI bus standards. You will occasionally find a PCI2.2 specification card that will just not work in a PCI2.1 specification system. Also, many cards with on-board bridges (such as, multi-port network cards) will

not work well in older systems.
- The PCI bus supports two levels of signaling, 3.3v and 5v. Cards that work with 3.3v signaling have a second notch cut in their PCI connector. Most PCI cards use 5v signaling, which is used by most computers. The Soekris single-board computers (Net45x1 and Net4801) are commonly-encountered computers that only support 3.3v signaling.

12.1.2 - ISA devices

- ISA devices cannot share resources, and in general, must be manually configured to settings that don't conflict with other devices in the system.
- Some ISA devices are "Plug and Play" (isapnp(4)) -- if you have any problem with these devices, though, verify their configuration in your dmesg(8), ISAPnP doesn't always work as desired.
- In general, if you have a choice, most people are best advised to avoid ISA cards in favor of PCI. ISA cards are more difficult to configure and have a much greater negative impact on the system's performance.

12.2 - DEC Alpha

[nothing yet]

12.3 - HP300

[nothing yet]

12.4 - HPPA

[nothing yet]

12.5 - i386

12.5.1 - ISA NICs

As OpenBSD runs well on older hardware, users often will end up using ISA NICs on OpenBSD systems. ISA hardware requires much more configuration and understanding than does PCI hardware. In general, you can't just stuff the card in the computer and expect it to magically work. In many machines, if your ISA device is not in a "Plug 'n' Play" (PNP) mode, you must reserve the resources the card uses in the system's BIOS.

3Com 3C509B ep(4)

This is an excellent performing ISA NIC, supported by the ep(4) driver. The 'B' version can be distinguished from the non-B version by labeling on the card and by the larger "main" chip on the board (approximately 2.5cm on a side for the 'B' version, vs. 2cm on a side on the older version), and will provide better performance on a loaded or dual network card system. The 3C509B ships configured in a PNP mode, which unfortunately does not comply with standards, and causes problems in OpenBSD's isapnp(4) support. The adapter is picked up first as a non-PNP device, then again after the PNP support comes on-line, resulting in an extra NIC showing in the dmesg. This may work fine, or it may cause other problems. It is highly recommended that the 3C509B cards have PNP mode disabled and manually configured to non-conflicting settings using the 3Com DOS-based configuration utilities before configuration.

The ep(4) driver will pick the cards up at any hardware combination that does not conflict with other devices in the system.

If you have multiple 3C509 cards in your system, it is recommended that you label the cards' spine with the MAC address, and use the dmesg to identify which is which.

Note that the 3C509, the 3C905 and the 3C590 are often confused. The 3C509 is a 10Mbps ISA card, the 3C905 and 3C590 are PCI cards.

NE2000

The original NE2000 NIC was developed in the mid-1980s by Novell. Since then, many manufacturers have produced cards that are very similar, which are generally called NE2000-compatibles, or clones. Performance of these clone cards varies greatly. While some older NE2000-compatible cards performed very well, many of the currently-available ones perform poorly. NE2000-compatibles are supported by the ne(4) driver in OpenBSD.

OpenBSD will handle some ISAPNP-capable NE2000-compatible cards well if the ISAPNP mode is turned on. Other cards will have to be set using either jumpers or a DOS-based configuration utility. Unfortunately, as the original NE2000 cards did not have software configuration or ISAPNP support, there are no standards for this -- you need the utility that will have been originally supplied with your specific card. This can often be difficult to obtain.

The ne(4) driver supports three configurations of the ISA NE2000 card in the GENERIC OpenBSD kernel:

```
ne0:  port 0x240 irq 9
ne1:  port 0x300 irq 10
ne2:  port 0x280 irq 9
```

If these settings are not acceptable, you can adjust them using User Kernel Configuration (UKC) or by building a customized kernel.

Note that the ne(4) driver is fairly "dumb" -- only the I/O port is probed, if any of the above I/O addresses is detected, the corresponding IRQ is *assumed*. dmesg(8) will not reflect the actual IRQ of the adapter in the case of ISA ne(4) drivers. If this is not the actual IRQ your card is set to, it will not work.

Note that there are non-ISA cards that use the ne(4) driver -- PCI and PCMCIA ne(4) cards exist. These notes do not apply to them, these devices are auto-configuring.

12.5.2 - OpenBSD won't work on my 80386/80386SX/80486SX system!

80386sx

The 80386sx has a maximum addressing range of 16M, which is at the very low end of OpenBSD/i386's support. Most 80386sx systems can't support more than 8M of RAM, which places them in the "For Experts Only" category, as some non-trivial steps and a second computer are required to get going. Also, see the next section:

80386

OpenBSD 3.4 will run on an 80386 or 80386sx system IF it has a 80387 or 80387sx hardware math coprocessor (Floating Point Unit, or FPU). Unfortunately, these FPUs were not common, so many 80386 systems will not have them. OpenBSD will not run without the FPU on the i386 platform. Again, be aware that this is a very small amount of processor for a crypto-intensive operating system like OpenBSD. You aren't likely to be happy with the performance of such a machine for general use.

80486SX

The 80486SX chip was a "low-cost" version of the 80486, which lacked the hardware floating point support (like the 80386) OpenBSD 3.4 requires. Fortunately, full 80486DX chips are fairly available, and is an easy upgrade in most systems.

12.5.3 - My dmesg shows multiple devices sharing the same Interrupt (IRQ)!

This is entirely acceptable, and in fact, even desirable for PCI devices. This is a design feature of the PCI bus. Some people will say that sharing interrupt requests (IRQs) is bad, however they are either confusing the situation with the ISA bus (where IRQ sharing is not permitted), or past experience with broken hardware or software.

ISA devices can not share IRQs. If you find ISA devices sharing IRQs, you must correct this problem.

12.5.4 - How do I use a USB keyboard?

OpenBSD/i386 will usually use a USB keyboard and mouse without difficulty after installation. However, installing OpenBSD on a system with a USB keyboard can be difficult, as the install kernels do not have the full complement of USB drivers required for complete USB keyboard support. As machines vary, you may need to experiment:

- Some systems have a BIOS option for "Legacy USB support" or something similar. This allows the BIOS to emulate a traditional PS/2 keyboard. If this is not enabled, you will *not* be able to enter commands at the `boot>` prompt. However, some systems may run better with this feature turned on, some with it turned off. Some may install better with it in one setting and run better with it in the other. Some machines do not have this option.
- If you use "Legacy USB support", you may find it works better to disable kernel USB support in User Kernel Configuration (UKC) by entering "`disable uhci`" before completing boot.
- PS/2 keyboard/mouse-to-USB adapters may not work.
- You *may* find the system is more stable with the mouse disconnected until the system is fully installed.
- In the worst case situation, you may find it easier to install OpenBSD on another computer, then move the hard disk to your "legacy-free" machine.

Once you have OpenBSD installed:

- Using UKC to disable pckbd0 may result in less "kernel message noise" (usually, `pckdc: cmd failed`), though if you do this, you will not be able to use a PS/2 mouse.
- The USB keyboard and mouse are handled just like a PS/2 keyboard and mouse by X. Use "wscons" for your mouse driver.
- At the moment, there is no way to set up a serial console on a "legacy-free" (no serial, parallel or PS/2 port) system. If you have difficulty with your system, you will have to record messages with pencil and paper.

Unfortunately, there are a variety of ways in which USB is supported on PCs right now, so you may have to do some experimenting with your system to get the USB keyboard working properly. Please contact faq@openbsd.org with any other well-documented USB keyboard tips you find.

12.5.5 - My keyboard/mouse keeps locking up (or goes crazy)!

This is most often seen when using a "switch box" to attach multiple computers to one keyboard, monitor and mouse. You can experiment with different brand and design switch boxes, but OpenBSD seems to be more sensitive to switching the mouse than some other operating systems. The problem is usually just the switching of the mouse. If you are not using the mouse, the solution is simple: don't attach the mouse cable to the computer. If you are using the mouse, an easy solution is "one mouse per computer", and switch just the keyboard and monitor. If you just want console access to the machine, you may wish to consider using a serial console instead.

12.6 - Mac68k

12.6.1 - My Mac68k system doesn't seem to work!

Unfortunately, we have a small number of different Mac68k machines in the hands of OpenBSD developers, and thus, only a few are known to work well at this moment. If you can help restore support to currently broken systems, your code would be welcome!

12.6.2 - Why is my Mac68k 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 the Mac68k 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. MacPPC systems do not have this issue.

Mac OS 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.

A simple solution is to run rdate(8) on a regular basis, by having a crontab entry for it. Another good place to launch rdate (8) is in your */etc/ppp/ppp.linkup* file if you are not permanently connected and are a PPP user.

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

12.6.3 - My Mac68k system won't work with two disks

Yes, unfortunately, this is a known problem. This usually shows itself by crashes shortly after the system goes multi-user, and the crash may not be obviously related to the disk system. The solution is to only use one drive on your Mac68k systems. A proper fix would be welcomed!

12.6.4 - The installer crashed during install

The Mac68k installer running on Mac OS will not install files to a "large" partition. If any partition you are installing to is more than around 500M in size, you can expect the following strange error:

```
Error on SCSIRead(), #5
pos = 0, i = 22, fs = /
alloccgblk: can't find blk in cyl
```

The solution to this is to first install a minimal system in a "small" root partition, then boot OpenBSD and relocate things as desired.

So, let's say you want a 200M / partition. Create the root partition, create the other partitions you want, newfs them all using the Mac OS utilities. Install etc34.tgz and base34.tgz to this / partition.

Mount your other partitions on a temporary mount point, and copy over the directories you want to them, as demonstrated here.

Now, modify /etc/fstab, reboot, and unpack the rest of the *.tgz files as documented here.

Following this process, any size can be achieved for the non-root partitions.

12.7 - MacPPC

12.7.1 - Why is my bm(4) network adapter so slow?

The bm driver, supporting the BMAC chip used on some MacPPC systems (including early iMacs) has issues when run at 100Mbps. It is highly recommended that you force the driver to 10Mbps by using a `"media 10baseT"` option in your `/etc/hostname.bm0` file, or otherwise force it to 10Mbps at your hub or switch.

12.8 - MVME68k

[nothing yet]

12.9 - SPARC

[nothing yet]

12.10 - UltraSPARC (sparc64)

12.10.1 - My UltraSPARC won't boot from the floppy image

Only the Ultra 1/1e and Ultra 2 can boot *any* OS from floppy disk. Use CD-ROM, Miniroot, or network boot to do your installation instead.

12.11 - DEC VAX

[nothing yet]

[FAQ Index] [To Section 11 - Performance Tuning] [To Section 14 - Disk Setup]

---

www@openbsd.org

$OpenBSD: faq12.html,v 1.50 2004/03/15 18:50:46 saad Exp $

OpenBSD

[FAQ Index] [To Section 12 - Platform-Specific Questions]

14 - Disk Setup

Table of Contents

14.1 - Using OpenBSD's disklabel(8)

Table of Contents

What is disklabel(8)?

First be sure to read the disklabel(8) man page.

Disklabels are created to allow an efficient interface between your disk and the disk drivers contained within the kernel. Labels hold certain information about your disk, like your drive geometry and information about your filesystems. This is then used by the bootstrap program to load the drive and to know where filesystems are contained on the drive. Labels are also used in conjunction with the filesystems to create a more efficient environment. You can read more in-depth information about disklabel by reading the disklabel(5) man page.

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(8), 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!

disklabel(8) during OpenBSD's install

One of the major parts of OpenBSD's install is your initial creation of labels. This comes (for i386 users) directly after using fdisk(1). During the install you use disklabel to create your separate labels which will contain your separate mountpoints. During the install, you can set your mountpoints from within disklabel(8), but this isn't completely necessary considering you will be prompted later to confirm you choices. But it does make your install go just a little smoother.

Since this is during the install you won't have any existing labels, and they will need to be created. The first label you will create is the label 'a'. This label SHOULD be your where / will be mounted. You can see recommended partitions that should be created and their sizes by reading FAQ 4, Space Needed. For servers it is recommended that you create at least these label's separately. For desktop users creating one mountpoint at / will probably suffice. When initially

creating your root partition ('a' label), keep in mind that you will need SOME space left for your swap label. Now that the basics have been explained, here is an example of using disklabel during an install. In this first example it is assumed that OpenBSD will be the only operating system on this computer, and that a full install will be done.

```
If this disk is shared with other operating systems, those operating systems
should have a BIOS partition entry that spans the space they occupy completely.
For safety, also make sure all OpenBSD file systems are within the offset and
size specified in the 'A6' BIOS partition table.  (By default, the disklabel
editor will try to enforce this).  If you are unsure of how to use multiple
partitions properly (ie. separating /,  /usr, /tmp, /var, /usr/local, and other
things) just split the space into a root and swap partition for now.

# using MBR partition 3: type A6 off 63 (0x3f) size 4991553 (0x4c2a41)

Treating sectors 63-16386300 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)
> d a
> a a
offset: [63] <Enter>
size: [16386237] 64M
Rounding to nearest cylinder: 131040
FS type: [4.2BSD] <Enter>
mount point: [none] /
fragment size: [1024] <Enter>
block size: [8192] <Enter>
cpg: [16] <Enter>
> a b
offset: [131103] <Enter>
size: [16255197] 64M
Rounding to nearest cylinder: 131040
FS type: [swap] <Enter>
```

At this point we have created a 64M root partition mounted at /, and a 64Meg swap partition. Notice that the offset starts at sector 63. This is what you want. When it comes to the size, disklabel will show your size in sectors, however, you don't need to enter sizes in the same format. Like the example above you can enter sizes in the manner of *64 Megabytes = 64M* and *2 Gigabytes = 2G*. Disklabel will then round to the nearest cylinder. In the example above you will also notice that disklabel assumes that label 'b' will be a swap. This is a correct assumption as the GENERIC kernel is set to look for swap on label 'b', and you should just follow this guideline and use 'b' as your swap area.

The next example will take you through the creation of two more labels. This means that it's not a complete install, as the size of these won't be enough to install OpenBSD to its fullest. Showing the creation of all the partitions would just be repetitive.

```
> a d
offset: [262143] <Enter>
size: [16124157] 64M
Rounding to nearest cylinder: 131040
FS type: [4.2BSD] <Enter>
mount point: [none] /tmp
fragment size: [1024] <Enter>
block size: [8192] <Enter>
cpg: [16] <Enter>
> a e
offset: [393183] <Enter>
size: [15993117] 64M
Rounding to nearest cylinder: 131040
FS type: [4.2BSD] <Enter>
mount point: [none] /var
fragment size: [1024] <Enter>
block size: [8192] <Enter>
cpg: [16] <Enter>
```

In the above example, there are two things you might notice. One being that the offset is automatically figured out for you to be the next in order. When doing an install of this sort, you won't need to mess with changing the offsets at all. Another difference you might notice will be that label 'c' has been skipped. This is done for a reason, which is that label 'c' is a label that represents the whole disk. For this reason you shouldn't deal with label 'c' in any way.

Once all your labels have been created all that's left to do is write the labels to disk, and move on in the installation process. To write everything and quit disklabel (and continue with the install) do:

```
> w
> q
```

Common uses for disklabel(8)

Once your system is installed, you shouldn't need to use disklabel too often. But some times you will need to use disklabel when adding, removing or restructuring your disks. One of the first things you will need to do is view your current disklabel. To do this, simply type:

```
# disklabel wd0 >----- Or whatever disk device you'd like to view

# 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*)
```

The above command simply allows you to view the existing disklabel, and assuring that you dont mess anything up. (Which we all need sometimes.) But to be able to make changes you must use the -E option with disklabel like so:

```
# disklabel -E wd0
```

This will bring you to a prompt, the same as the one that you used during the OpenBSD install. Probably the single most important command at this prompt is '?'. This will give you a list of possible options pertaining to disklabel. You can even view the entire disklabel(8) man page with the 'M' command. From this prompt, you will do all of your adding, deleting and changing of partitions. For additional information read the disklabel(8) man page.

14.2 - Using fdisk(8)

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

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, use:

```
# fdisk sd0
```

Which will give an output similar to this:

```
Disk: sd0       geometry: 553/255/63 [8883945 Sectors]
```

```
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
*0: A6    3   0  1 -  552 254 63 [       48195:     8835750 ] OpenBSD
 1: 12    0   1  1 -    2 254 63 [          63:       48132 ] Compaq Diag.
 2: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
 3: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
```

In this example we are viewing the fdisk output of the first SCSI 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)
        setpid          Set the identifier of a given table entry
        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.
- **setpid** Change the partition identifier of the given partition table entry. This command is particularly useful for reassigning an existing partition to OpenBSD.
- **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. In the below example we're adding a third SCSI drive to the system.

```
# 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! Fstype for OpenBSD 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).

```
# newfs sd2a
```

Or whatever your disk was named as per OpenBSD's disk numbering scheme. (Look at the output from dmesg(8) 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/sd2a /u
```

Finally, add it to /etc/fstab(5).

```
/dev/sd2a /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(5) file to show that the /usr/local partition is now /dev/sd2a (your freshly formatted partition.) Example:

```
/dev/sd2a /usr/local ffs rw 1 1
```

Reboot into single user mode with **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 voila, 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).)

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 space setup. The file must not reside on a filesystem which has SoftUpdates enabled (they are disabled by default). To start out, you can see how much swap you currently have and how much you are using with the swapctl(8) utility. You can do this by using the command:

```
$ 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.

```
$ 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

```
$ sudo chmod 600 /var/swap
$ 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.

```
$ 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.

```
$ cat /etc/fstab
/dev/wd0a / ffs rw 1 1
/var/swap /var/swap swap sw 0 0
```

Swapping via a vnode device

This is a more permanent 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.

```
$ 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.

```
$ 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(8).

```
$ 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.

```
$ sudo swapctl -a /dev/vnd0c
$ 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

Soft Updates is based on an idea proposed by Greg Ganger and Yale Patt and developed for FreeBSD by Kirk McKusick. SoftUpdates imposes a partial ordering on the buffer cache operations which permits the requirement for synchronous writing of directory entries to be removed from the FFS code. Thus, a large performance increase is seen in disk writing performance.

The potential of background fsck(8), using Soft Updates is not yet realised in OpenBSD, so fsck(8) is still required after an unclean shutdown. This may be changed in future versions.

To use Soft Updates, your kernel must have

option FFS_SOFTUPDATES

compiled in, this is already in place on GENERIC.

Enabling soft updates must be done with a mount-time option. When mounting a partition with the mount(8) utility, you can specify that you wish to have soft updates enabled on that partition. Below is a sample */etc/fstab(5)* entry that has one partition *sd0a* that we wish to have mounted with soft updates.

```
/dev/sd0a / ffs rw,softdep 1 1
```

Note to sparc users: Do not enable soft updates on sun4 or sun4c machines. These architectures support only a very limited amount of kernel memory and cannot use this feature. However, sun4m machines are fine.

14.6 - How does OpenBSD/i386 boot?

The boot process for OpenBSD/i386 is not trivial, and understanding how it works can be useful to troubleshoot a problem when things don't work. There are four key pieces to the boot process:

1. ***Master Boot Record (MBR):*** The Master Boot Record is the first physical sector (512 bytes) on the disk. It contains the primary partition table and a small program to load the Partition Boot Record (PBR). Note that in some environments, the term "MBR" is used to refer to only the code portion of this first block on the disk, rather than the whole first block (including the partition table). It is critical to understand the meaning of "initialize the MBR" -- in the terminology of OpenBSD, it would involve rewriting the entire MBR sector, not just the code, as it might on some systems. You will rarely want to do this. Instead, use fdisk(8)'s "-u" command line option ("`fdisk -u wd0`").

   While OpenBSD includes an MBR, you are not obliged to use it, as virtually any MBR can boot OpenBSD. The MBR is manipulated by the fdisk(8) program, which is used both to edit the partition table, and also to install the MBR code on the disk.

   OpenBSD's MBR announces itself with the message:

   ```
   Using Drive: 0 Partition: 3
   ```

   showing the disk and partition it is about to load the PBR from.
2. ***Partition Boot Record (PBR):*** The Partition Boot Record, also called the PBR or biosboot(8) (after the name of the file that holds the code) is the first

physical sector of the OpenBSD partition of the disk. The PBR is the "first-stage boot loader" for OpenBSD. It is loaded by the MBR code, and has the task of loading the OpenBSD second-stage boot loader, boot(8). Like the MBR, the PBR is a very tiny section of code and data, only 512 bytes, total. That's not enough to have a fully filesystem-aware application, so rather than having the PBR locate `/boot` on the disk, the BIOS-accessible location of `/boot` is physically coded into the PBR at installation time.

The PBR is installed by installboot, which is further described later in this document. The PBR announces itself with the message:

```
reading boot....
```

printing the dots as it reads sectors from the disk.

3. ***Second Stage Boot Loader, /boot:*** `/boot` is loaded by the PBR, and has the task of accessing the OpenBSD file system through the machine's BIOS, and locating and loading the actual kernel. boot(8) also passes various options and information to the kernel.

   boot(8) is an interactive program. After it loads, it attempts to locate and read `/etc/boot.conf`, if it exists (which it does not on a default install), and processes any commands in it. Unless instructed otherwise by `/etc/boot.conf`, it then gives the user a prompt:

   ```
   probing: pc0 com0 com1 apm mem[636k 190M a20=on]
   disk: fd0 hd0+
   >> OpenBSD/i386 BOOT 2.02
   boot>
   ```

   It gives the user (by default) five seconds to start giving it other tasks, but if none are given before the timeout, it starts its default behavior: loading the kernel, `bsd`, from the root partition of the first hard drive. The second-stage boot loader probes (examines) your system hardware, through the BIOS (as the OpenBSD kernel is not loaded). Above, you can see a few things it looked for and found:
   - **pc0** - the standard keyboard and video display of a i386 system.
   - **com0, com1** - Two serial ports
   - **apm** - Advanced Power Management BIOS functions
   - **636k 190M** - The amount of conventional (below 1M) and extended (above 1M) memory it found
   - **fd0 hd0+** - The BIOS disk devices found, in this case, one floppy and one hard disk.

   The '+' character after the "hd0" indicates that the BIOS has told `/boot` that this disk can be accessed via LBA. When doing a first-time install, you will sometimes see a '*' after a hard disk -- this indicates a disk that does not seem to have a valid OpenBSD disk label on it.
4. ***Kernel: /bsd*:** This is the goal of the boot process, to have the OpenBSD kernel loaded into RAM and properly running. Once the kernel has loaded, OpenBSD accesses the hardware directly, no longer through the BIOS.

So, the very start of the boot process could look like this:

```
Using Drive: 0 Partition: 3                     <- MBR
reading boot....                                <- PBR
probing: pc0 com0 com1 apm mem[636k 190M a20=on] <- /boot
disk: fd0 hd0+
>> OpenBSD/i386 BOOT 2.02
boot>
booting hd0a:/bsd 4464500+838332 [58+204240+181750]=0x56cfd0
entry point at 0x100120

[ using 386464 bytes of bsd ELF symbol table ]
Copyright (c) 1982, 1986, 1989, 1991, 1993       <- Kernel
        The Regents of the University of California.  All rights reserved.
Copyright (c) 1995-2003 OpenBSD.  All rights reserved.  http://www.OpenBSD.org

OpenBSD 3.4 (GENERIC) #18: Wed Sep 17 03:34:47 MDT 2003
   ...
```

What can go wrong

- **Bad/invalid/incompatible MBR:** Usually, a used hard disk has some MBR code in place, but if the disk is new or moved from a different platform, AND you don't answer "Yes" to the "Use entire disk" question of the installation process, you may end up with a disk without a valid MBR, and thus, will not be bootable, even though it has a valid partition table.

  You may install the OpenBSD MBR on your hard disk using the fdisk program. Boot from your install media, choose "Shell" to get a command prompt:

  ```
  # fdisk -u wd0
  ```

  You may also install a specific MBR to disk using fdisk:

```
# fdisk -u -f /usr/mdec/mbr wd0
```

which will install the file `/usr/mdec/mbr` as your system's MBR. This particular file on a standard OpenBSD install happens to be the standard MBR that is also built into fdisk, but any other MBR could be specified here.

- **Invalid `/boot` location installed in PBR:** When installboot(8) installs the partition boot record, it writes the physical location of `/boot`'s disk sectors into the PBR. Therefore, deleting and replacing `/boot` without re-running installboot(8) will render your system unbootable, as the PBR will load whatever happens to be at the blocks specified in it, which will almost certainly no longer be the second-stage boot loader! Since `/boot` is being read using BIOS calls, if you alter the drive's geometry (i.e., taking it out of one computer that uses CHS translation and moving it into one that uses LBA translation, or even changing a translation option in your BIOS), it will *appear to the BIOS* to be in a different location (a different numerical block must be accessed to get the same data from the disk), so again, you will have to run installboot(8) before the system can be rebooted. As the PBR is very small, its range of error messages is pretty limited:
  - **Read Error** -- BIOS returned an error when trying to read a block from the disk. Could be a physical problem with the disk, or an invalid sector was called for (i.e., geometry problem).
  - **Bad magic** -- An invalid magic(5) number was read in the second-stage bootloader's header. This generally means whatever it was that was read in was NOT `/boot`. Usually, this is due to a drive geometry problem, having replaced `/boot` on your disk or ignoring the 8G or BIOS limitations of your system, and having `/boot` end up somewhere the PBR can't reach it.

For more information on the i386 boot process, see

- boot_i386(8)
- http://www.ata-atapi.com/hiw.htm Hale Landis' "How it Works" documents.

14.7 - What are the issues regarding large drives with OpenBSD?

OpenBSD has support for file systems of sizes much larger than any currently or soon to be available hard disks, however there are limitations on some interfaces which are smaller than the theoretical maximum of OpenBSD.

Not all hardware combinations are possible, of course. A new 250G IDE hard disk will not work on an older (pre >137G standards) interfaces, and some very old SCSI adapters have been seen to have problems with more modern drives. You must respect the abilities of your hardware, of course.

Partition size and location limitations

Unfortunately, the full ability of the OS isn't available until AFTER the OS has been loaded into memory, and the booting process introduces limits of its own. The boot process has to utilize (and is thus limited by) the system's boot ROM. The OpenBSD i386 first-stage boot loader (biosboot(8)) also has its own internal 8G limitation, from an older BIOS limit.

For this reason, the entire /bsd file (the kernel) must be located on the disk within the boot ROM addressable area, or within the first 8G of the disk, whichever is smaller. This means that on some older i386 systems, the root partition must be completely within the first 504M, but for most newer computers, the root partition may be anywhere within the first 8G.

Note that it is possible to install a 40G drive on an old 486 and load OpenBSD on it as one huge partition, and think you have successfully violated the above rule. However, it might come back to haunt you in a most unpleasant way:

- You install on the 40G / partition. It works, because the base OS and all its files (including /bsd) are within the first 504M.
- You use the system, and end up with more than 504M of files on it.
- You upgrade, build your own kernel, whatever, and copy your new /bsd over the old one.
- You reboot.
- You get a message such as "bad magic"

Why? Because when you copied "over" the new /bsd file, it didn't overwrite the old one, it got relocated to a new location on the disk, probably outside the 504M range the BIOS supported. The boot loader was unable to fetch the file /bsd, and the system hung.

To get OpenBSD to boot, the boot loaders (biosboot(8) and `/boot` in the case of i386) and the kernel (`/bsd`) must be within the boot ROM's supported range, and within their own abilities. To play it safe, the rule is simple:

The entire root partition must be within the computer's BIOS (or boot ROM) addressable space or within the first 8G, whichever is smaller.
Note: the OpenBSD/i386 8G limit has been removed in -current. You must still respect your system's BIOS limit, however

This is another good reason to partition your hard disk, rather than using one large partition.

fsck(8) time and memory requirements

Another consideration with large file systems is the time and memory required to fsck(8) the file system after a crash or power interruption. One should not put a 120G file system on a system with 32M of RAM and expect it to successfully fsck(1) after a crash. A rough guideline is the system should have at least 1M of available memory for every 1G of disk space to successfully fsck the disk. The time required to fsck the drive may become a problem as the file system size expands.

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 drive (or use a bootable CD-ROM) 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)
```

14.9 - Preparing for disaster: Backing up and Restoring from tape

Introduction:

If you plan on running what might be called a production server, it is advisable to have some form of backup in the event one of your fixed disk drives fails.

This information will assist you in using the standard dump(8)/restore(8) utilities provided with OpenBSD. A more advanced backup utility called "Amanda" is also available through ports for backing up multiple servers to one tape drive. In most environments dump(8)/restore(8) is enough. However, if you have a need to backup multiple machines to one tape, Amanda might be worth investigating in the future.

The device examples in this document are for a configuration that uses both SCSI disks and tape. In a production environment, SCSI disks are recommended over IDE due to the way in which they handle bad blocks. That is not to say this information is useless if you are using an IDE disk or other type of tape drive, your device names will simply differ slightly. For example sd0a would be wd0a in an IDE based system.

Backing up to tape:

Backing up to tape requires knowledge of where your file systems are mounted. You can determine how your filesystems are mounted using the mount(8) command at your shell prompt. You should get output similar to this:

```
# mount
/dev/sd0a on / type ffs (local)
/dev/sd0h on /usr type ffs (local)
```

In this example, the root (/) filesystem resides physically on sd0a which indicates SCSI fixed disk 0, partition a. The /usr filesystem resides on sd0h, which indicates SCSI fixed disk 0, partition h.

Another example of a more advanced mount table might be:

```
# mount
/dev/sd0a on / type ffs (local)
/dev/sd0d on /var type ffs (local)
/dev/sd0e on /home type ffs (local)
/dev/sd0h on /usr type ffs (local)
```

In this more advanced example, the root (/) filesystem resides physically on sd0a. The /var filesystem resides on sd0d, the /home filesystem on sd0e and finally /usr on sd0h.

To backup your machine you will need to feed dump the name of each fixed disk partition. Here is an example of the commands needed to backup the simpler mount table listed above:

```
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0a
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0h
# mt -f /dev/rst0 rewind
```

For the more advanced mount table example, you would use something similar to:

```
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0a
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0d
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0e
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0h
# mt -f /dev/rst0 rewind
```

You can review the dump(8) man page to learn exactly what each command line switch does. Here is a brief description of the parameters used above:

- **0** - Perform a level 0 dump, get everything
- **a** - Attempt to automatically determine tape media length
- **u** - Update the file /etc/dumpdates to indicate when backup was last performed
- **f** - Which tape device to use (/dev/nrst0 in this case)

Finally which partition to backup (/dev/rsd0a, etc)

The mt(1) command is used at the end to rewind the drive. Review the mt man page for more options (such as eject).

If you are unsure of your tape device name, use dmesg to locate it. An example tape drive entry in dmesg might appear similar to:

```
st0 at scsibus0 targ 5 lun 0: <ARCHIVE, Python 28388-XXX, 5.28>
```

You may have noticed that when backing up, the tape drive is accessed as device name "nrst0" instead of the "st0" name that is seen in dmesg. When you access st0 as nrst0 you are accessing the same physical tape drive but telling the drive to not rewind at the end of the job and access the device in raw mode. To back up multiple file systems to a single tape, be sure you use the non-rewind device, if you use a rewind device (rst0) to back up multiple file systems, you'll end up overwriting the prior filesystem with the next one dump tries to write to tape. You can find a more elaborate description of various tape drive devices in the dump man page.

If you wanted to write a small script called "backup", it might look something like this:

```
echo "  Starting Full Backup..."
/sbin/dump -0au -f /dev/nrst0 /dev/rsd0a
/sbin/dump -0au -f /dev/nrst0 /dev/rsd0d
/sbin/dump -0au -f /dev/nrst0 /dev/rsd0e
/sbin/dump -0au -f /dev/nrst0 /dev/rsd0h
echo
echo -n "  Rewinding Drive, Please wait..."
mt -f /dev/rst0 rewind
echo "Done."
echo
```

If scheduled nightly backups are desired, cron(8) could be used to launch your backup script automatically.

It will also be helpful to document (on a scrap of paper) how large each file system needs to be. You can use df -h to determine how much space each partition is currently using. This will be handy when the drive fails and you need to recreate your partition table on the new drive.

Restoring your data will also help reduce fragmentation. To ensure you get all files, the best way of backing up is rebooting your system in single user mode. File systems do not need to be mounted to be backed up. Don't forget to mount root (/) r/w after rebooting in single user mode or your dump will fail when trying to write out dumpdates. Enter bsd -s at the boot> prompt for single user mode.

Viewing the contents of a dump tape:

After you've backed up your file systems for the first time, it would be a good idea to briefly test your tape and be sure the data on it is as you expect it should be.

You can use the following example to review a catalog of files on a dump tape:

```
# /sbin/restore -tvs 1 -f /dev/rst0
```

This will cause a list of files that exist on the 1st partition of the dump tape to be listed. Following along from the above examples, 1 would be your root (/) file system.

To see what resides on the 2nd tape partition and send the output to a file, you would use a command similar to:

```
# /sbin/restore -tvs 2 -f /dev/rst0 > /home/me/list.txt
```

If you have a mount table like the simple one, 2 would be /usr, if yours is a more advanced mount table 2 might be /var or another fs. The sequence number matches the order in which the file systems are written to tape.

Restoring from tape:

The example scenario listed below would be useful if your fixed drive has failed completely. In the event you want to restore a single file from tape, review the restore man page and pay attention to the interactive mode instructions.

If you have prepared properly, replacing a disk and restoring your data from tape can be a very quick process. The standard OpenBSD install/boot floppy already contains the required restore utility as well as the binaries required to partition and make your new drive bootable. In most cases, this floppy and your most recent dump tape is all you'll need to get back up and running.

After physically replacing the failed disk drive, the basic steps to restore your data are as follows:

- Boot from the OpenBSD install/boot floppy. At the menu selection, choose Shell. Write protect and insert your most recent back up tape into the drive.
- Using ther fdisk(8)r command, create a primary OpenBSD partition on this newly installed drive. Example:

  ```
  # fdisk -e sd0
  ```

  See fdisk FAQ for more info.
- Using the disklabel command, recreate your OpenBSD partition table inside that primary OpenBSD partition you just created with fdisk. Example:

  ```
  # disklabel -E sd0
  ```

  (Don't forget swap, see disklabel FAQ for more info)
- Use the newfs command to build a clean file system on each partition you created in the above step. Example:

  ```
  # newfs /dev/rsd0a
  # newfs /dev/rsd0h
  ```

- Mount your newly prepared root (/) file system on /mnt. Example:

  ```
  # mount /dev/sd0a /mnt
  ```

- Change into that mounted root file system and start the restore process. Example:

```
# cd /mnt
# restore -rs 1 -f /dev/rst0
```

- You'll want this new disk to be bootable, use the following to write a new MBR to your drive. Example:

```
# fdisk -i sd0
```

- In addition to writing a new MBR to the drive, you will need to install boot blocks to boot from it. The following is a brief example:

```
# cp /usr/mdec/boot /mnt/boot
# /usr/mdec/installboot -v /mnt/boot /usr/mdec/biosboot sd0
```

- Your new root file system on the fixed disk should be ready enough so you can boot it and continue restoring the rest of your file systems. Since your operating system is not complete yet, be sure you boot back up with single user mode. At the shell prompt, issue the following commands to unmount and halt the system:

```
# umount /mnt
# halt
```

- Remove the install/boot floppy from the drive and reboot your system. At the OpenBSD boot> prompt, issue the following command:

```
boot> bsd -s
```

- The bsd -s will cause the kernel to be started in single user mode which will only require a root (/) file system.
- Assuming you performed the above steps correctly and nothing has gone wrong you should end up at a prompt asking you for a shell path or press return. Press return to use sh. Next, you'll want to remount root in r/w mode as opposed to read only. Issue the following command:

```
# mount -u -w /
```

- Once you have remounted in r/w mode you can continue restoring your other file systems. Example:

```
(simple mount table)
# mount /dev/sd0h /usr; cd /usr; restore -rs 2 -f /dev/rst0

(more advanced mount table)
# mount /dev/sd0d /var; cd /var; restore -rs 2 -f /dev/rst0
# mount /dev/sd0e /home; cd /home; restore -rs 3 -f /dev/rst0
# mount /dev/sd0h /usr; cd /usr; restore -rs 4 -f /dev/rst0
```

- You could use "**restore rvsf**" instead of just rsf to view names of objects as they are extracted from the dump set.
- Finally after you finish restoring all your other file systems to disk, reboot into multiuser mode. If everything went as planned your system will be back to the state it was in as of your most recent back up tape and ready to use again.

14.10 - Mounting disk images in OpenBSD

To mount a disk image (ISO images, disk images created with dd, etc) in OpenBSD you must configure a vnd(4) device. For example, if you have an ISO image located at */tmp/ISO.image*, you would take the following steps to mount the image.

```
# vnconfig svnd0 /tmp/ISO.image
# mount -t cd9660 /dev/svnd0c /mnt
```

Notice that, since this image is a CD image you must specify type of *cd9660* when mounting it. This is true, no matter what type, e.g. you must use type ffs when mounting disk images.

To unmount the image use the following commands.

```
# umount /mnt
# vnconfig -u svnd0
```

For more information, refer to the vnconfig(8) man page.

14.11 - Help! I'm getting errors with IDE DMA!

DMA IDE transfers, supported by pciide(4) are unreliable with many combinations of hardware. Until recently, most "mainstream" operating systems that claimed to support DMA transfers with IDE drives did not ship with that feature active by default due to unreliable hardware. Now many of these same machines are being used for OpenBSD.

OpenBSD is aggressive and attempts to use the highest DMA Mode it can configure. This will cause corruption of data transfers in some configurations because of buggy motherboard chipsets, buggy drives, and/or noise on the cables. Luckily, Ultra-DMA modes protect data transfers with a CRC to detect corruption. When the Ultra-DMA CRC fails, OpenBSD will print an error message and try the operation again.

```
wd2a:  aborted command, interface CRC error reading fsbn 64 of 64-79
(wd2 bn 127; cn 0 tn 2 sn 1), retrying
```

After failing a couple times, OpenBSD will downgrade to a slower (hopefully more reliable) Ultra-DMA mode. If Ultra-DMA mode 0 is hit, then the drive downgrades to PIO mode.

UDMA errors are often caused by low quality or damaged cables. Cable problems should usually be the first suspect if you get many DMA errors or unexpectedly low DMA performance. It is also a bad idea to put the CD-ROM on the same channel with a hard disk.

If replacing cables does not resolve the problem and OpenBSD does not successfully downgrade, or the process causes your machine to lock hard, or causes excessive messages on the console and in the logs, you may wish to force the system to use a lower level of DMA or UDMA by default. This can be done by using UKC or config(8) to change the flags on the wd(4) device.

14.13 - RAID options for OpenBSD

RAID (Redundant Array of Inexpensive Disks) gives an opportunity to use multiple drives to give better performance, capacity and/or redundancy than one can get out of a single drive alone. While a full discussion of the benefits and risks of RAID are outside the scope of this article, there are a couple points that are important to make here:

- RAID has nothing to do with backup.
- By itself, RAID will not eliminate down-time.

If this is new information to you, this is not a good starting point for your exploration of RAID.

Software Options

OpenBSD includes RAIDframe, a software RAID solution. Documentation for it can be found in the following places:

- FAQ 11, RAID
- RAIDframe Homepage
- man page for raidctl(8)
- man page for raid(4)

The root partition can be directly mirrored by OpenBSD using the "Autoconfiguration" option of RAIDframe.

Hardware Options

Many OpenBSD platforms include support for various hardware RAID products. The options vary by platform, see the appropriate hardware support page (listed here).

Another option available for many platforms is one of the many products which make multiple drives act as a single IDE or SCSI drive, and are then plugged into

a standard IDE or SCSI adapter. These devices can work on virtually any hardware platform that supports either SCSI or IDE.

Some manufacturers of these products:

- Arco
- Accusys
- Maxtronic
- Infortrend

(Note: these are just products that OpenBSD users have reported using -- this is not any kind of endorsement, nor is it an exhaustive list.)

Non-Options

An often asked question on the mail lists is "Are the Promise or HighPoint IDE RAID controllers supported?". The answer is "No". These cards and chips are not true hardware RAID controllers, but rather BIOS-assisted boot of a software RAID. As OpenBSD already supports software RAID in a hardware-independent way, there isn't much desire among the OpenBSD developers to implement special support for these cards.

[FAQ Index] [To Section 12 - Platform-Specific Questions]


www@openbsd.org
$OpenBSD: faq14.html,v 1.104 2004/04/23 16:18:57 nick Exp $