## 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


# 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 v3.5. Il y aura vraisemblablement des nouvelles fonctionnalités et des modifications apportées aux fonctionnalités existantes au niveau de la version de développement (*-current*) qui ne seront pas couvertes par la présente FAQ.

La FAQ aux formats 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 :

- 7 - Contrôles du clavier et de l'affichage
- 8 - Questions Générales
- 9 - Migrer vers OpenBSD
- 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

## 6 - Mise en place du réseau



## 7 - Contrôles du clavier et de l'affichage



## 8 - Questions Générales

## 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 1.216 ]
$Translation: index.html,v 1.97 2004/10/15 13:05:04 saad Exp $
$OpenBSD: index.html,v 1.68 2004/10/15 15:53:37 saad 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 3.5.

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

OpenBSD 3.5 fonctionne sur les plates-formes suivantes :

- alpha - FTP uniquement
- amd64 - CD bootable *Une nouveauté de la version 3.5 !*
- cats - FTP only *Une nouveauté de la version 3.5 !*
- hp300 - FTP uniquement
- hppa - FTP uniquement
- i386 - CD bootable
- mac68k - FTP uniquement
- macppc - CD bootable

- mvme68k - FTP uniquement
- mvme88k - FTP uniquement - *Une nouveauté de la version 3.5 !*
- 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 fonctionne 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.html 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 4.4.0, non encombré avec les modifications de licence : l'environnement X Window, avec des correctifs locaux. 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 et 3.3.2. 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 `comp35.tgz`.
- Perl 5.8.2, avec des correctifs et des améliorations crées par l'équipe OpenBSD.
- Le serveur web Apache 1.3.29. L'équipe OpenBSD a ajouté le confinement chroot par défaut , la révocation de privilèges, et d'autres améliorations sécurité. mod_ssl 2.8.16 et DSO sont aussi inclus.
- OpenSSL 0.9.7c, avec des correctifs et des améliorations de la part de l'équipe OpenBSD
- Le processeur de texte Groff 1.15.
- Le serveur de messagerie Sendmail 8.12.11.
- Le serveur DNS BIND 9.2.3. 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
- Arla-current
- OpenSSH 3.8.1

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 3.5 ?

La liste complète des changements apportés à OpenBSD 3.4 pour créer OpenBSD 3.5 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 3.5 et qui sont familières des versions antérieures :

- **La plate-forme i386 est désormais en ELF depuis OpenBSD 3.4.** 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. 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.
- **La limite des 8G pour la partition racine a été supprimée.** La plate-forme i386 supporte désormais un démarrage à

partir de n'importe quel endroit du disque supporté par le BIOS. Ceci veut dire que la limite des 8G des versions précédentes n'est plus d'actualité. Un partitionnement est bien entendu toujours hautement recommandé.

- **Démarrage PXE.** Les plates-formes i386 et amd64 supportent désormais le PXE booting pour effectuer l'installation.
- **Nouvelles plates-formes.** De nouvelles plates-formes sont supportées par OpenBSD 3.5 :
  - cats, une carte de développement à base de StrongARM
  - amd64, le processeur AMD 64 bit, et
  - mvme88k, systèmes à base de processeurs RISC de la série Motorola 88000.
- **sparc64 utilise désormais GCC 3.3.2.** La plate-forme sparc64 utilise maintenant GCC 3.3.2 au lieu de GCC 2.95.3, utilisé sur les autres plates-formes. Une réinstallation complète est hautement recommandée pour mettre à jour les systèmes existants. Les nouvelles plates-formes cats et amd64 utilisent aussi GCC 3.2.2. Des ajouts locaux tels que ProPolice et d'autres améliorations ont, bien entendu, aussi été incorporés dans le nouveau GCC.
- **Nouveaux utilisateurs et groupes.** Plusieurs nouveaux utilisateurs et groupes ont été ajoutés à OpenBSD pour la séparation de privilèges. Lors des mises à jour, il faut s'assurer de mettre à jour le répertoire `/etc` tel que c'est indiqué dans upgrade-minifaq.

# 1.10 - Puis-je utiliser OpenBSD comme station de travail ?

Cette question est souvent posée exactement de cette manière, sans aucune précision sur le sens de "station de travail". La seule personne capable de répondre à cette question est vous, étant donné que la réponse dépend de vos besoins et de vos attentes.

Bien qu'OpenBSD possède une bonne réputation en tant que système d'exploitation "serveur", il peut être utilisé sur les stations de travail. Plusieurs applications destinées à des "stations de travail" sont disponibles à travers les ports et les packages. Comme pour n'importe quelle autre décision relative à un système d'exploitation, la question est la suivante : est-ce qu'OpenBSD peut remplir telle fonction de telle manière ? Il vous appartient de répondre à cette question.

# 1.11 - Pourquoi *ProductX* est/n'est pas inclus ?

Souvent, des personnes demandent pourquoi un produit particulier est ou n'est pas inclus dans OpenBSD. La réponse est basée sur deux éléments : les souhaits des développeurs et la compatibilité avec les objectifs du projet. Un produit ne sera pas inclus simplement parce qu'il est bien -- il doit aussi être "libre" à utiliser, à distribuer et à modifier selon nos standards. Un produit doit aussi être stable et sûr -- un numéro de version plus grand n'est pas toujours synonyme d'un meilleur produit.

La licence est souvent le plus grand problème : nous voulons qu'OpenBSD reste utilisable par n'importe quelle personne n'importe où au monde pour n'importe quel but.

Quelques questions communes sur les produits tiers :

- **Pourquoi Sendmail est inclus, il est "connu pour être non sécurisé"** Sendmail a un historique sécurité imparfait, cependant les auteurs de Sendmail et les mainteneurs ont été très réceptifs pour retravailler leur code pour le rendre beaucoup plus sécurisé (et ceci est une réponse malheureusement peu commune). L'historique sécurité récent de Sendmail n'est pas si différent de celui de certaines alternatives supposées "plus sécurisées".
- **Pourquoi Postfix n'est pas inclus ?** La licence n'est pas libre et donc ce logiciel ne peut être inclus.
- **Pourquoi qmail ou djbdns ne sont pas inclus ?** Licence, ou absence de licence : l'incapacité de distribuer une version modifiée de ce logiciel fait qu'il ne peut être inclus.
- **Pourquoi Apache est inclus ? beaucoup de personnes n'en veulent pas !** Parce que les développeurs le veulent.
- **Pourquoi une version plus récente d'Apache n'est pas incluse ?** La licence sur les versions plus récentes est inacceptable.
- **Pourquoi XFree86 v4.4 n'est pas inclus ?** Les modifications apportées à la licence dans la version v4.4 sont inacceptables. Ainsi, et comme beaucoup de projets de logiciel libre principaux, OpenBSD utilise la dernière version libre, qui est "pratiquement" la v4.4.

Dans la plupart des cas, ces sujets ont été discutés de manière trop détaillée sur les listes de diffusion, veuillez consulter les

archives si vous avez besoin de plus d'informations.

Bien entendu, si vous souhaitez utiliser un de ces packages dans la mesure où votre utilisation est compatible avec la licence de ces produits, personne ne vous en empêchera (ça ne serait pas vraiment très "libre" si on le faisait, n'est-ce pas ?). Cependant, vos besoins pourraient changer -- vous ne voudriez peut-être pas développer une "Killer Application" que vous ne pouvez pas vendre, distribuer, ou utiliser pour devenir riche parce que vous y avez incorporé des éléments non libres.

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

---


www@openbsd.org

Originally [OpenBSD: faq1.html,v 1.63 ]
$Translation: faq1.html,v 1.42 2004/10/25 13:49:34 saad Exp $
$OpenBSD: faq1.html,v 1.30 2004/10/25 13:55:37 saad Exp $

[Index de la FAQ] [Section 1 - Introduction à OpenBSD] [Section 3 - Obtenir OpenBSD]

# 2 - Autres Ressources d'Information OpenBSD

---

### Table des matières



---

## 2.1 - Pages Web intéressantes

Le site officiel pour le projet OpenBSD se trouve à : http://www.OpenBSD.org.

Beaucoup d'informations utiles s'y trouvent en ce qui concerne tous les aspects du projet OpenBSD.

L'OpenBSD Journal est un site de nouvelles et d'opinions autour d'OpenBSD.

Plusieurs utilisateurs ont mis en place des sites et des pages avec des informations spécifiques à OpenBSD. Un bon moteur de recherche vous facilitera la vie, comme le fera aussi une dose raisonnable de scepticisme. Comme d'habitude, n'utilisez pas aveuglément des commandes que vous ne comprenez pas.

## 2.2 - Listes de discussion

Le projet OpenBSD maintient plusieurs listes de discussion auxquelles les utilisateurs peuvent s'inscrire. Pour s'inscrire à une liste, il suffit d'envoyer un message à majordomo@openbsd.org. Cette adresse correspond à un service d'inscription automatique. Dans le corps du message, sur une seule ligne, vous devez inclure une commande d'inscription pour la liste désirée. Par exemple :

```
subscribe announce
```

Le gestionnaire automatique de la liste vous répondra, pour obtenir une confirmation de votre part afin que d'autres personnes ne vous inscrivent pas à une avalanche de courriel non sollicité. Le message contiendra des instructions pour différentes méthodes de confirmation, y compris un lien vers la page web du serveur de listes, la réponse au message de confirmation ou la réponse à `majordomo@openbsd.org`. Utilisez la méthode qui vous convient. Il est à noter que les trois techniques précitées impliquent un nombre identifiant unique et limité dans le temps, tel que `A56D-70D4- 52C3`, encore une fois pour s'assurer que *you* êtes réellement la personne qui a demandé l'inscription à la liste de diffusion (c'est le *vrai* "opt-in").

Une fois que vous avez confirmé votre intention de vous joindre à la liste, vous serez immédiatement ajouté à celle-ci, et le gérant vous l'indiquera.

Pour se désabonner d'une liste, il vous faut encore envoyer un message à majordomo@openbsd.org. Cela ressemblera à :

```
unsubscribe announce
```

Si vous avez des difficultés avec le système des listes de discussion, veuillez d'abord lire le fichier d'aide qui peut être obtenu en envoyant un message à majordomo@openbsd.org avec dans le corps de celui-ci : "help".

Votre inscription aux listes de diffusion OpenBSD peut aussi être maintenue à travers l'interface Web disponible à l'adresse http://lists.openbsd.org

Parmi les listes de discussions les plus populaires, on peut citer :

- **announce** - Annonces importantes. Il s'agit d'une liste à faible débit.
- **security-announce** - Annonces des problèmes de sécurité.
- **misc** - Questions et réponses générales d'utilisateurs. C'est la liste la plus active et elle devrait être la liste par "défaut" pour la plupart des questions.
- **tech** - Sujet techniques pour les développeurs OpenBSD et les utilisateurs avancés. Merci de diriger les questions des nouveaux utilisateurs et liés à l'installation vers *misc*, pas *tech*. Merci de ne pas cruciposter dans *misc* et *tech*.
- **bugs** - Bugs reçu par l'intermédiaire de sendbug(1) et discussions à leur sujet.
- **source-changes** - Liste recueillant le changements du réceptacle CVS.
- **ports** - Discussions à propos de l'arbre des ports OpenBSD.
- **ports-changes** - Liste automatique des changements dans le réceptacle source CVS relatifs aux ports.
- **advocacy** - Discussions sur la promotion d'OpenBSD.

Avant d'envoyer une question à **misc** ou toute autre liste de diffusion, parcourez les archives, pour y voir les questions fréquemment posées de façon répétée. Bien que cela puisse être la première fois que vous rencontrez un problème ou que vous avez une question, les autres participants de la liste peuvent avoir vu la même question plusieurs fois en plusieurs semaines et peuvent ne pas apprécier de la voir, une fois encore. Si votre question concerne du matériel, *envoyez toujours un dmesg(8) !*

Vous pouvez consulter plusieurs archives, d'autres lignes de conduite relatives aux listes de diffusion ainsi que d'autres informations dans la page des listes de diffusion.

Une liste de diffusion non officielle qui pourrait intéresser les nouveaux utilisateurs d'OpenBSD et d'Unix est la liste appellée OpenBSD Newbies.

## 2.3 - Pages du manuel

OpenBSD est fourni avec une abondante documentation sous la forme de pages du manuel, ainsi que d'autres documents relatifs à des applications spécifiques. Un effort considérable est fourni pour s'assurer que les pages du manuel sont à jour et correctes. Dans tous les cas, les pages du manuel sont à considérer comme la source faisant autorité concernant les informations sur OpenBSD.

Pour accéder aux pages du manuel ainsi qu'au reste de la documentation, assurez-vous d'avoir installé les ensembles de fichiers `man35.tgz` et `misc35.tgz`.

Voici une liste des pages du manuel les plus utiles pour les nouveaux utilisateurs :

**Débutants**

- **afterboot(8)** - les choses à vérifier après le premier redémarrage complet
- **help(1)** - aide pour les nouveaux utilisateurs et administrateurs
- **hier(7)** - agencement des systèmes de fichiers.
- **man(1)** - affiche les pages du manuel.
- **intro(1)** - introduction aux commandes générales, voir aussi les introductions aux autres sections du manuel : intro(2), intro(3), intro(4) (remarque : intro(4) est spécifique à chaque plate-forme), intro(5), intro(6), intro(7), intro(8), et intro(9).
- **adduser(8)** - commande pour ajouter de nouveaux utilisateurs.
- **vipw(8)** - édite le fichier de mots de passe principal.
- **disklabel(8)** - lecture et écriture du label disk pack.
- **reboot, halt(8)** - arrêt et redémarrage du système.
- **shutdown(8)** - arrêt du système à un temps donné.
- **dmesg(8)** - réaffichage des messages de démarrage du noyau
- **sudo(8)** - ne pas se logger en root mais exécute des commandes avec des droits root.
- **mg(1)** - éditeur de texte fonctionnant comme emacs

**Pour des utilisateurs plus avancés**

- **boot(8)** - procédures de bootstrapping système.
- **login.conf(5)** - format du fichier de configuration des classes de connexion.
- **ifconfig(8)** - configure les paramètres de l'interface réseau.
- **netstat(1)** - affiche le statut du réseau.
- **boot_config(8)** - comment modifier la configuration noyau au démarrage.
- **release(8)** - compile et prépare une distribution OpenBSD.
- **sendbug(1)** - envoi d'un rapport de bogues (PR) sur OpenBSD à un site central.
- **sysctl(8)** - obtenir ou paramétrer des états noyau.
- **style(9)** - guide du code source noyau OpenBSD.

Vous pouvez trouvez toutes les pages du manuel OpenBSD sur le web sur http://www.openbsd.org/cgi-bin/man.cgi ainsi que sur votre ordinateur si vous installez l'archive de fichiers `man35.tgz`.

En général, si vous connaissez le nom d'une commande ou d'une page de manuel, vous pouvez la lire en tapant `"man commande"`. Par exemple : `"man vi"` pour obtenir la page de manuel à propos de l'éditeur vi. Si vous ne connaissez pas le nom d'une commande ou si `"man commande"` ne trouve pas la page du manuel vous pouvez chercher dans la base de données des pages du manuel en tapant `` `apropos quelque_chose' `` ou `"man -k quelque_chose"`, où "quelque_chose" est une expression qui a des chances d'apparaître dans le titre de la page de manuel que vous recherchez. Par exemple :

```
# apropos "time zone"
tzfile (5) - time zone information
zdump (8) - time zone dumper
zic (8) - time zone compiler
```

Les numéros entre parenthèses indique la section du manuel dans laquelle cette page a été trouvée. Dans certains cas, vous pourrez trouver des pages du manuel avec des noms identiques dans différentes sections du manuel. Par exemple, supposons que vous souhaitiez connaître le format des fichiers de configuration du démon cron. Une fois que vous connaissez la section du manuel pour la page que vous recherchez, tapez `"man n commande"`, où n est le numéro de la section correspondante.

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

En plus des pages du manuel UNIX, il existe une documentation imprimable (inclue dans le jeu de fichiers `misc35.tgz`). Celle-ci se trouve dans le répertoire `/usr/share/doc`. Vous pouvez formater chacune des parties de la documentation avec un "make" dans le répertoire correspondant. Le répertoire `psd` contient le document *Programmer's Supplementary Documents*. Le répertoire `smm` contient le document *System Manager's Manual*. Le répertoire `usd` contient le document *UNIX User's Supplementary Documents*. Vous pouvez effectuer votre "make" dans les trois sous répertoires ou vous pouvez sélectionner une section spécifique d'un document et faire le "make" dans le répertoire correspondant.

Certaines parties sont vides. Par défaut, les documents seront formatés au format PostScript, utilisable pour l'impression. La taille de la sortie PostScript peut-être importante (attendez vous à une augmentation de volume de l'ordre de 250-300%). Si vous n'avez pas accès à une imprimante PostScript (ou à un visualiseur PostScript), vous pouvez toujours formater les documents de façon à les lire sur un terminal. Chaque sous- répertoire à documents possède une cible de compilation pour compiler des copies ASCII de ces documents (appelés `paper.txt') qui peuvent être générés avec make(1). Par exemple :

```
# cd /usr/share/doc/usd/04.csh
# make paper.txt
# more paper.txt
```

Les privilèges super-utilisateur peuvent être nécessaires pour compiler les documents dans ces répertoires. Un **make clean** aura pour effet de supprimer tous les documents générés par un make précédent. Veuillez consulter /usr/share/doc/README pour plus de détails concernant les documents dans /usr/share/doc/.

Les pages du manuel UNIX sont généralement plus à jour que les documents imprimables. Mais ceux-ci peuvent expliquer des applications complexes avec plus de détails que ne le font les pages du manuel.

Pour beaucoup de personnes, avoir une version papier d'une page du manuel est utile. Voici les indications pour obtenir une version imprimable d'une page du manuel.

## Comment est-ce que j'affiche le code source d'une page du manuel (par exemple, un de ces fichiers finissant par un numéro, comme tcpdump.8) ?

On les trouve dans les sources. Les pages du manuel non formatées se trouvent dans les sources et elles seront souvent mises à jour par l'utilisation de CVS. Pour les visualiser, il faut juste taper :

```
# nroff -Tascii -mandoc <file> | more
```

## Comment est-ce que j'obtiens une page de manuel sans caractères de contrôle ou de formatage ?

Il est utile d'obtenir une page du manuel sans caractères non imprimables.
Exemple :

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

## Comment puis-je obtenir une copie PostScript d'une page du manuel ?

Remarquez que `<file>` doit être le fichier source de la page du manuel (certainement un fichier finissant par un numéro comme `tcpdump.8`). Les versions PostScript des pages du manuel ont une belle apparence. Elles peuvent être imprimées ou visualisées à l'aide d'un programme comme gv (GhostView). GhostView se trouve dans l'arbre des ports. Utilisez les options de commande suivantes de nroff(1) pour obtenir une version PostScript à partir d'une page de manuel système d'OpenBSD :

```
# nroff -Tps -mandoc <file> > outfile.ps
```

## Comment générer des copies compressées des pages du manuel ?

Pour les personnes qui compilent leur système à partir des sources, il existe un certain nombre d'options permettant de contrôler la manière dont les pages du manuel sont construites. Ces options figurent dans /etc/mk.conf (il peut être nécessaire de créer ce fichier) et sont incluses durant les compilations système. Une option particulièrement utile permet de générer des pages du manuel compressées pour économiser de l'espace disque. Celles-ci peuvent être consultées de la manière habituelle avec la commande man. Pour cela, ajoutez la ligne suivante à `/etc/mk.conf` :

```
MANZ=yes
```

Une autre option utile permet de générer les pages du manuel aux formats PostScript et ASCII. Cette option est `MANPS=yes`, à ajouter dans /etc/mk.conf. Voir mk.conf(5) pour plus de détails.

## Que sont les fichiers info ?

Une partie de la documentation d'OpenBSD est sous forme de fichiers info. Ces fichiers se trouvent typiquement sous `/usr/share/info`. C'est une forme de documentation alternative fournie par GNU. Plusieurs fichiers info sont plus à jour que les pages du manuel fournis par GNU et peuvent être visualisés à l'aide de la commande info(1) Par exemple, pour voir les informations sur le compilateur GNU, gcc(1), tapez :

```
# info gcc
```

Après avoir utilisé info, vous allez vraiment apprécier nos pages du manuel !

## Comment afficher les pages du manuel en couleur dans un XTerm ?

Le fichier de configuration par défaut de xterm(1) n'affiche pas les pages du manuel en couleur. Afin d'avoir un affichage couleur, copiez le fichier `/etc/X11/app-defaults/XTerm-color` dans votre répertoire personnel et renommez-le en "`.Xdefaults`". Veillez à ne pas écraser un paramétrage particulier dans "`.Xdefaults`". Ce fichier contient tous les paramétres dont vous aurez besoin pour activer les couleurs sous XTerm. Cependant, trois lignes doivent être décommentées auparavant :

```
!*VT100*colorULMode: on
!*VT100*underLine: off
!*VT100*colorBDMode: on
```

Le reste du fichier permet de choisir les couleurs pour plusieurs paramétres. Les lignes applicables aux pages du manuel sont :

```
*VT100*colorUL: yellow
*VT100*colorBD: white
```

Ce qui produit des pages du manuel avec une couleur infernale, adoptez la à votre convenance : pouvons nous suggérer rouge pour "colorUL" et magenta pour "colorBD"? Il existe aussi un afficheur de pages de manuel sous X11, xman(1), qui fournit une interface alternative (graphique) aux pages du manuel. Consultez les pages du manuel de xterm et xman pour plus d'informations.

## Comment écrire ma propre page du manuel ?

Si vous souhaitez écrire votre propre page du manuel pour une application que vous avez écrite, un tutoriel est fourni dans mdoc.samples(7). Il existe aussi un guide de référence bien pratique dans mdoc(7).

## 2.4 - Rapporter les bugs

Avant de crier "Au bug!", merci de bien vouloir vous assurer que c'est vraiment le cas. Par contre, si vous ne comprenez pas comment telle ou telle chose est implémentée par OpenBSD ou comment elle fonctionne, et vous ne trouvez pas comment résoudre le problème à l'aide des pages du manuel ou le site OpenBSD, utilisez les listes de diffusion (généralement misc@openbsd.org) pour demander de l'aide. Si c'est votre première expérience avec OpenBSD, soyez réaliste : vous n'avez probablement pas découvert un bug inconnu. Notez aussi que du matériel défectueux peut mimer un bug logiciel. Alors prenez le temps de vérifier l'état de votre matériel avant de décider que vous avez trouvé un "bug".

Finalement, avant de soumettre un rapport de bug, veuillez lire http://www.openbsd.org/report.html.

Faire un rapport de bug correct est l'une des plus grandes responsabilités conférée aux utilisateurs. Des informations très détaillées sont nécessaires pour diagnostiquer les bugs les plus sérieux. Les développeurs reçoivent fréquemment des rapports de bugs par mail du style :

```
From: joeuser@exemple.com
To: bugs@openbsd.org
Subject: AU SECOURS!!!

J'ai un PC et il ne démarre pas !!!!!! C'est un 486 !!!!!
```

Heureusement la plupart des gens savent que de tels rapports sont rapidement effacés. Tous les rapports de bugs doivent contenir des informations détaillées. Si Joe User souhaite que son bug soit corrigé, il faudrait que son rapport ressemble à ça :

(Remarque : voir report.html pour de plus amples informations sur la façon de créer et d'envoyer des rapports de bug. Une description de votre matériel est utile si vous pensez que votre problème est lié au matériel. Une copie de votre dmesg(8) est généralement suffisante. Ensuite une description de votre problème est indispensable)

```
From: smartuser@exemple.com
To: bugs@openbsd.org
Subject: 3.3-beta panique sur une SPARCStation2

OpenBSD 3.2 installé depuis un CD-ROM officiel fonctionnait bien
sur cette machines.

Après avoir effectué une installation propre de la version
3.3-beta depuis un mirroir FTP, le système panique
aléatoirement après une certaine période
d'utilisation, et de façon prévisible et rapidement en
démarrant X.

Voici le dmesg :

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


Ceci est la panique lorsque j'essaye de démarrer 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

Merci beaucoup !
```

Consultez report.html concernant la manière de créer et d'envoyer des rapports des bogues. Des informations détaillées à propos de votre matériel sont nécessaires si vous pensez que le bogue peut être, *de quelque manière que se soit*, lié à votre matériel ou à sa configuration. Habituellement, un dmesg(8) est suffisant. Une description détaillée de votre problème est nécessaire. Vous remarquerez que le dmesg décrit le matériel, le texte explique pourquoi Smart User pense que sont système n'est pas défecteux, (3.2 fonctionnait convenablement), comment ce "crash" était causé (en démarrant X), et le détail des commandes "`ps`" et "`trace`" afin de déboguer. Dans ce cas- ci, Smart User a fourni ces détails capturés via une console serie, si vous ne savez pas le faire, vous devrez utiliser une feuille et un

stylo afin de rentranscrire le "crash". (L'exemple ci- dessus était un vrai problème qui affectait les systèmes sun4c, et les informations fournies dans le précédant rapport ont aidés à le résoudre).

Si notre ami Smart User possède un système OpenBSD fonctionnel et qu'il veut soumettre un rapport de bug, il peut le faire en utilisant l'utilitaire sendbug(1) qui enverra le rapport au système de suivi des bugs GNATS. Évidemment, vous ne pouvez utiliser sendbug(1) si votre système ne démarre pas mais utiliser le dès que c'est possible. Vous aurez toujours à inclure des informations détaillées sur ce qui s'est passé, la configuration de votre système et comment reproduire le problème. La commande sendbug(1) nécessite que votre système puisse envoyer des messages électroniques sur l'Internet. Notez que le serveur de messagerie utilise la fonctionnalité "greylisting" de spamd(8). Il peut alors s'écouler jusqu'à une demi-heure avant que le serveur de messagerie n'accepte votre rapport. Soyez donc patient.

Après avoir envoyé un rapport de bogue via sendbug(1), vous serez averti de son état par e-mail. Vous pourriez être contacté par les développeurs concernant plus d'information ou pour des correctifs qui doivent être testés. Vous pouvez également consulter les archives de la liste de diffusion `bugs@openbsd.org`, plus de détails se trouvant sur la page des listes de diffusion, ou consultez la base de données des rapports de bogues en-ligne sur notre Système de Consultation des Bogues.

## Comment fournir plus d'informations utiles aux développeurs

Voici quelques astuces supplémentaires :

**Vous avez perdu le "Panic message" ?**
Sous certaines conditions, vous pouvez perdre le tout premier message d'une panique système, et qui donne la raison de la panique. Ce message est très important, vous devez donc l'inclure dans votre rapport. Vous pouvez le retrouver en utilisant la commande "x/s *panicstr" (eXamine String *panicstr) dans ddb> comme suit :

```
ddb> x/s *panicstr
0:      kernel: page fault trap, code=0
ddb>
```

Dans ce cas, le message fût "Kernel: page fault trap, code=0"

**Remarque pour les systèmes SMP :**
Vous devez obtenir une "trace" pour chaque processeur et les insérer dans votre rapport :

```
ddb{0}> trace
pool_get(d05e7c20,0,dab19ef8,d0169414,80) at pool_get+0x226
fxp_add_rfabuf(d0a62000,d3c12b00,dab19f10,dab19f10) at
fxp_add_rfabuf+0xa5
fxp_intr(d0a62000) at fxp_intr+0x1e7
Xintr_ioapic0() at Xintr_ioapic0+0x6d
--- interrupt ---
idle_loop+0x21:
ddb{0}> machine ddb 1
Stopped at      Debugger+0x4:   leave
ddb{1}> trace
Debugger(d0319e28,d05ff5a0,dab1bee8,d031cc6e,d0a61800) at Debugger+0x4
i386_ipi_db(d0a61800,d05ff5a0,dab1bef8,d01eb997) at i386_ipi_db+0xb
i386_ipi_handler(b0,d05f0058,dab10010,d01d0010,dab10010) at
i386_ipi_handler+0x
4a
Xintripi() at Xintripi+0x47
--- interrupt ---
i386_softintlock(0,58,dab10010,dab10010,d01e0010) at
i386_softintlock+0x37
Xintrltimer() at Xintrltimer+0x47
--- interrupt ---
idle_loop+0x21:
ddb{1}>
```

Répétez la commande "`machine ddb x`" suivie de "`trace`" pour chaque processeur de votre machine.

[Index de la FAQ] [Section 1 - Introduction à OpenBSD] [Section 3 - Obtenir OpenBSD]

---


www@openbsd.org

Originally [OpenBSD: faq2.html,v 1.78 ]
$Translation: faq2.html,v 1.33 2004/10/25 11:39:46 saad Exp $
$OpenBSD: faq2.html,v 1.27 2004/10/25 12:38:57 jufi 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 : Commandes.

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 : T-shirts OpenBSD.

## 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 `cd35.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, HTTP ou AFS

Il y a plusieurs miroirs internationaux offrant un accès FTP et HTTP 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 :

Page FTP OpenBSD.

# 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 :

AnonCVS OpenBSD.

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 :

OpenBSD CTM.

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.org/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 1.45 ]
$Translation: faq3.html,v 1.23 2004/10/25 09:48:43 saad Exp $
$OpenBSD: faq3.html,v 1.18 2004/10/25 12:38:57 jufi Exp $

[Retour à l'Index principal] [Section 3 - Obtenir OpenBSD] [Section 5 - Construire le Système à partir des Sources]

# 4 - OpenBSD 3.5 Guide d'Installation

---

### Table des Matières



---

## 4.1 - Présentation de la procédure d'installation OpenBSD

OpenBSD a une procédure d'installation robuste, adaptable et peut être installé depuis une simple disquette. La plupart des architectures suivent une procédure d'installation similaire ; cependant quelques détails diffèrent. Dans tous les cas, il vous est vivement conseillé de lire le document INSTALL spécifique à votre architecture se trouvant dans le dossier *"platform"* sur le CD-ROM ou les sites FTP (par exemple, `i386/INSTALL.i386`, `mac68k/INSTALL.mac68k` ou `sparc/INSTALL.sparc`).

Sur la plupart des architectures, l'installation OpenBSD utilise un noyau spécial avec un certain nombre d'utilitaires et de scripts d'installation inclus dans un disque RAM préchargé. Après le démarrage de ce noyau, le système d'exploitation est extrait depuis plusieurs fichiers tar(1) (`.tgz`) compressés. Il existe de nombreux moyens de démarrer ce noyau d'installation :

- **Disquette :** Des images de disquettes utilisables pour créer une disquette d'installation depuis un autre système Compatible Unix ou sur un système

DOS/Windows sont fournies. Les noms de fichiers typiques sont `floppy35.fs`, bien que plusieurs architectures possèdent de multiples images de disquettes.
- **CD-ROM :** Sur plusieurs architectures une image CD-ROM (`cd35.iso`) est fournie, autorisant la création d'un CD-ROM amorçable. Il contient simplement le noyau d'installation - les fichiers d'installation doivent toujours être téléchargés via FTP ou une autre source. Vous pouvez, bien sûr, créer votre propre CD-ROM avec les fichiers et outils que vous désirez.
- **bsd.rd :** Le noyau RAM, créé pour le démarrage depuis une partition OpenBSD existante ou à travers le réseau.
- **Réseau:** Quelques architectures supportent le démarrage à travers un réseau.
- **Ecrire une image de système de fichiers sur disque :** Une image de système de fichiers qui peut être écrite depuis une partition existante, et qui sera ensuite amorçable.
- **Cassettes amorçable :** Certaines architectures supportent l'amorçage depuis des cassettes. Ces cassettes peuvent être créées en suivant les instructions du fichier INSTALL.*platform*.

Toutes les architectures ne supportent pas chacune des options de démarrage :

- **alpha :** Disquette, CD-ROM, Ecriture d'une image de disquette sur le disque dur.
- **amd64 :** Disquette, CD-ROM, Réseau.
- **cats :** CD-ROM.
- **hp300 :** CD-ROM, Réseau.
- **hppa :** Réseau.
- **i386 :** Disquette, CD, Réseau.
- **mac68k :** Installé (et démarré) en utilisant des outils lancés sur Mac OS. Voir INSTALL.mac68k pour plus de détails.
- **macppc :** CD-ROM, Réseau.
- **mvme68k :** Réseau, Cassette amorçable.
- **mvme88k :** Réseau, Cassette amorçable.
- **sparc :** Disquette, CD-ROM, Réseau, Ecriture d'une image sur une partition swap existante, Cassette amorçable.
- **sparc64 :** Disquette (U1/U2 uniquement), CD-ROM, Réseau, Ecriture d'une image sur une partition existante.
- **vax :** Disquette, Réseau.

Toutes les installations autres que mac68k peuvent aussi utiliser un noyau bsd.rd lors d'une mise à jour ou d'une réinstallation.

Une fois le noyau démarré, vous avez plusieurs options pour obtenir les Packages d'installation. Une fois de plus, toutes les architectures ne supportent pas toutes ces options.

- **CD-ROM :** Bien sûr, nous préférons que vous utilisiez les CD-ROMS Officiels, mais pour des besoins spéciaux, vous pouvez aussi créer les vôtres.
- **FTP :** L'un des différents sites miroirs FTP OpenBSD ou votre serveur FTP local contenant les packages.
- **HTTP :** L'un des différents sites miroirs HTTP OpenBSD ou votre serveur web local contenant les packages.
- **Partition de disque local :** Dans la plupart des cas, vous pouvez installer les packages depuis une autre partition de votre disque dur local. Par exemple, sur i386, vous pouvez installer à partir d'une partition FAT ou d'un CD-ROM formaté en format ISO9660, Rock Ridge ou Joliet. Dans certains cas, vous devrez manuellement monter le système de fichiers avant de l'utiliser.
- **NFS :** Quelques architectures supportent les montages NFS de packages.
- **Cassette :** Les packages peuvent aussi être lus depuis une cassette formatée.

## 4.2 - Vérifications avant l'installation

Avant de commencer votre installation, vous devriez avoir une idée de ce que vous aller devoir faire. Vous devriez connaître ces différents éléments au minimum :

- Nom de la machine
- Matériel installé et disponible
  - La compatibilité de votre architecture grâce à la page de compatibilité.
  - Dans les cas d'utilisation de matériel ISA, vous devrez aussi connaître les paramètres matériels et confirmer qu'ils sont tels qu'OpenBSD les requiert.
- La méthode d'installation qui sera utilisée (CD-ROM, FTP, etc.)
- Comment le système sera-t'il mis à jour et patché ?
  - Si cela est fait localement, vous devrez bénéficier de suffisamment d'espace libre pour stocker l'arbre des sources et le construire.
  - Autrement, vous aurez besoin d'accéder à une autre machine pour y construire une "release" patchée.
- Partitionnement de disque désiré.
  - Y a t'il des données à sauvegarder quelque part ?
  - OpenBSD coexistera t'il sur ce système avec d'autres OS ? Si oui, comment seront-il démarrés ? Aurez vous besoin d'installer un gestionnaire de démarrage ?
  - La totalité du disque sera t'elle utilisée par OpenBSD ou voulez-vous conserver une partition ou un OS (ou de la place pour un autre) ?
  - Comment voulez vous partitionner la partie réservée à OpenBSD de votre disque ?
- Paramètres Réseau, si vous n'utilisez pas DHCP:
  - Nom de domaine.

    - Adresse des Serveurs de Nom de Domaine (DNS)
    - Adresse IP et masque de sous-réseau pour chaque NIC
    - Adresse de la passerelle
- Lancerez vous le système de fenêtres X ?

# 4.3 - Créer un média d'installation OpenBSD amorçable

Comme exemples, nous allons regarder les images d'installation disponibles pour les architectures i386 et sparc.

L'architecture i386 dispose de quatre images de disques séparées parmis lesquelles choisir :

- **`floppy35.fs`** (PC de bureau) supporte la plupart des NICs PCI et ISA, des adaptateurs IDE et des adaptateurs simples SCSI et supporter quelques matériels PCMCIA. La *plupart* des utilisateurs utiliserons cette image.
- **`floppyB35.fs`** (Serveurs) supporte plusieurs contrôleurs RAID et quelques adaptateurs SCSI courants. Toutefois le support pour certains adaptateurs SCSI et plusieurs NICs EISA et ISA a été retiré.
- **`floppyC35.fs`** (Portables) supporte les matériels CardBus et PCMCIA trouvés sur la plupart des portables.
- **`cdrom35.fs`** est, en effet, la combinaison des trois disques de démarrage. Il peut être utilisé pour créer des disquettes amorçables de 2.88M, ou plus couramment, comme une image d'amorçe de CD enregistrables personnalisés.
- **`cd35.iso`** est une image ISO9660 qui peut être utilisée pour créer un CD amorçable avec la plupart des logiciels de création de CD-ROM sur la plupart des architectures. Ce `cdrom35.iso` est un format "prêt-à-graver".

Oui, il peut y avoir des situations dans lesquelles un disque d'installation est requis pour supporter votre adaptateur SCSI et un autre disque pour votre adaptateur réseau. Heureusement, ce cas est rare, et peut être généralement contourné.

L'architecture sparc dispose de trois images disques d'installation permis lesquelles choisir :

- **`floppy35.fs:`** Supporte les systèmes équipés d'un lecteur de disquettes.
- **`cd35.iso`** Une image ISO utilisable pour créer votre propre CD afin de d'amorcer les systèmes SPARC à l'aide un CD-ROM.
- **`miniroot35.fs`** Peut être écrite sur une partition swap et démarrée.

## 4.3.1 - Créer des disquettes de démarrage sur Unix

Pour créer une disquette formatée, utilisez la commande fdformat(1) pour chacun des formats et recherchez les secteurs défectueux..

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

Si votre sortie est identique à celle ci-dessus, votre disque est correct. Cependant, si vous ne voyez pas tous les "V" alors votre disque est probablement endommagé et vous devriez en essayer un autre.

Notez que certains systèmes compatibles UNIX ont des commandes différentes pour le formattage des disquettes. Réferrez-vous à votre documentation système pour connaître la procédure exacte.

Une fois que vous avez une disquette propre et formatée, il est l'heure d'écrire l'image d'installation sur celle-ci. Pour cela, vous pouvez recourir à l'utilitaire dd(1). Un exemple d'utilisation de `dd(1)` est fourni ci-dessous :

```
# dd if=floppy35.fs of=/dev/rfd0c bs=32k
```

Une fois l'image écrite, vérifiez qu'elle à été correctement copiée et qu'elle est identique à l'image d'origine avec la commande cmp(1). Si la disquette est identique à l'image, vous ne verrez qu'une nouvelle invite.

```
# cmp /dev/rfd0c floppy35.fs
```

## 4.3.2 - Créer des disquettes de démarrage sur Windows ou DOS

Cette section décrit comment ecrire des images d'installation sur une disquette à partir de Windows ou DOS. Vous pouvez obtenir les utilitaires mentionnés ci-dessous depuis le dossier tools de n'importe quel miroir FTP, ou depuis le dossier `3.5/tools` directement sur le CD1 du lot de CD-ROM OpenBSD.

Pour préparer la disquette sous MS-DOS ou Windows, utilisez tout d'abord les utilitaires natifs pour formatter le disque.

Pour écrire l'image d'installation sur la disquette préparée vous pouvez utiliser *rawrite*, *fdimage*, ou *ntrw*. *rawrite* ne fonctionnera pas sur Windows NT, 2000 ou XP.

Notez que `FDIMAGE.EXE` et `RAWRITE.EXE` sont toutes deux des applications MS-DOS, et sont donc limitées à la convention de nomage de fichiers MS-DOS "8.3". Comme `floppyB35.fs` et `floppyC35.fs` ont des noms de fichiers longs, vous devrez trouver comment votre système a sauvegardé ces derniers dans le format "8.3" avant d'utiliser `FDIMAGE.EXE` ou `RAWRITE.EXE` pour créer les images de disquettes de démarrage.

Exemple d'utilisation de *rawrite* :

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

Enter source file name: floppy35.fs
Enter destination drive: a
Please insert a formatted diskette into drive A: and press -ENTER- : Enter
```

Exemple d'utilisation de *fdimage* :

```
C:\> fdimage -q floppy35.fs a:
```

Exemple d'utilisation de *ntrw* :

```
C:\> ntrw floppy35.fs a:
3.5", 1.44MB, 512 bytes/sector
bufsize is 9216
1474560 bytes written
```

### 4.3.3 - Créer un CD-ROM amorçable

Vous pouvez créer un CD-ROM soit en utilisant le fichier `cd35.iso`, ou dans le cas de l'architecture i386, vous pouvez aussi utiliser `cdrom35.fs` comme image de disquette amorçable pour démarrer un système i386 depuis le CD-ROM. Les détails exacts ici présents sont laissés au lecteur afin de l'aider à se décider sur les outils mis à sa disposition.

Voici quelques-uns des outils présents dans OpenBSD :

- mkhybrid(8)
- cdrecord, qui est une partie de la collection cdrtools dans le Système de Ports OpenBSD.

## 4.4 - Démarrer le média d'installation OpenBSD

### Démarrer sur i386

Démarrer une image d'installation sur l'architecture PC i386 n'est pas nouveau pour la plupart des gens. Si vous utilisez une disquette, insérez la simplement dans le lecteur et démarrez le système. L'image d'installation va se charger si le démarrage depuis la disquette est activé dans votre BIOS. Si vous souhaitez démarrer depuis un CD-ROM, vous devrez allez dans votre BIOS système et autoriser le démarrage depuis celui-ci. Quelques anciens BIOS ne supportent pas cette option, et vous devrez utiliser une disquette pour démarrer votre image d'installation. Ne vous inquiétez pas ; bien que vous démarriez depuis la disquette, vous pourrez installer depuis le CD.

Vous pouvez aussi démarrer bsd.rd depuis une partition OpenBSD existante, ou depuis le réseau en utilisant le procédé de démarrage PXE.

### Démarrer sur sparc/sparc64

**NOTE :** Sur l'architecture sparc64, seules les machines SBus (Ultra 1, Ultra 2) sont amorçables depuis une disquette.

Pour démarrer depuis une disquette, placez la disquette contenant l'image d'installation OpenBSD dans le lecteur. Utilisez ensuite la commande suivante pour démarrer :

```
ok boot floppy
```

Pour démarrer depuis un CD-ROM, placez le CD-ROM OpenBSD dans le lecteur. Si votre Sun n'a qu'un lecteur de CD-ROM, alors allez simplement à l'invite de démarrage et tapez :

```
ok boot cdrom
```

Bien sûr, cela ne fonctionnera que dans le nouveau mode de commande. Si vous possédez l'ancien mode de commande (une flêche droite), tapez 'n' pour le nouveau mode de commande. (Si vous utilisez un ancien sparc qui est antérieur à sun4c, vous n'avez probablement pas de nouveau mode de commande. Dans ce cas, vous allez devoir expérimenter.) Si vous avez de multiples lecteurs CD-ROM, vous devez démarrer depuis le bon. Essayez `probe-scsi` depuis le nouveau mode de commande.

```
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
```

Regardez depuis lequel vous souhaitez démarrer. Notez le numéro de cible ("Target").

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

# 4.5 - Installer OpenBSD

## 4.5.1 - Commencer l'installation

Quelque soit votre méthode de démarrage, il est temps de l'utiliser. Pendant la procédure de démarrage, le noyau et tous les programmes utilisés pour installer OpenBSD sont chargés en mémoire. Le problème le plus courant au démarrage est une disquette endommagée ou un problème d'alignement de lecteur. La disquette de démarrage est très compactée -- un seul mauvais bloc causera des problèmes.

A presque tout moment de la procédure d'installation OpenBSD, vous pouvez arrêter la procédure en frappant CTRL-C et pourrez la relancer sans redémarrer en lanĉant `install` à l'invite shell.

Si votre démarrage réussi, vous verrez beaucoup de messages textes défiler. Ces textes, dans beaucoup d'architectures en blanc sur bleu, représentent le dmesg, du noyau qui donne les matériels trouvés et leur emplacement. Ne vous souciez pas de retenir ce message, une copie est conservée dans `/var/run/dmesg.boot`. Sur quelques architectures, SHIFT+PUG va vous permettre d'examiner le texte ayant défilé à l'écran.

Ensuite, vous verrez ce qui suit :

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

Avec cela, nous atteignons notre première question. Dans la plupart des cas, vous avez ces trois options :

- **Install :** Charge OpenBSD dans le système, remplaçant un éventuel autre système se trouvant là. Notez qu'il est possible de laisser certaines partitions intactes dans cette procédure, comme `/home`, mais dans les autres cas, considérez que tout le reste sera écrasé.
- **Upgrade :** Installer un nouveau package de fichiers d'installation sur cette machine, mais ne remplacer aucune information de configuration, donnée utilisateur ou programme additionnel. Aucun formatage de disque n'est réalisé, ni le dossier /etc ni /var ne sont remplacés. Quelques notes importantes :
  - Vous n'aurez pas l'option d'installer le package `etc35.tgz`. Après l'installation, vous devrez manuellement greffer `etc35.tgz` sur votre système avant qu'il puisse être complètement fonctionnel. C'est une étape importante qui doit être réalisée, car dans le cas contraires certains services clés (tel que pf(4)) ne démarrerons pas.
  - La procédure de mise à jour n'est pas faite pour sauter des "releases" ! Tandis que ce saut fonctionnera souvent, il n'est pas supporté. Pour OpenBSD 3.5, mettre à jour de version 3.4 à la version 3.5 est la seule mise à jour supportée. Si vous mettez à jour depuis une version plus vielle, une ré-installation complète est recommandée.
- **Shell :** Parfois, vous devez réaliser des réparations ou de la maintenance sur un système qui ne démarrera pas (ou ne devra pas démarrer) un noyau normal. Cette option vous autorisera cette maintenance sur le système.

Parfois, vous ne verrez pas l'option "Upgrade" listée. Après un évènement *"flag day"*, il n'est pas possible de mettre à jour directement ; l'on doit recréer le système depuis le début.

Dans cet exemple, nous allons faire une installation mais la procédure de mise à jour est similaire.

```
Welcome to the OpenBSD/i386 3.5 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] Entrée
Do you wish to select a keyboard encoding table? [no] Entrée
```

Dans la plupart des cas, le type de terminal par défaut est approprié ; cependant si vous utilisez une console série pour l'installation, ne prenez pas l'option par défaut, répondez de manière appropriée.

Si ne vous choisissez pas une table d'encodage, une disposition de clavier US est choisie.

```
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? [no] y
```

Si vous sélectionnez l'option par défaut ici, le proccessus d'installation va se terminer et vous envoyer sur une invite de commande shell.

### 4.5.2 - Configurer les disques

Configurer les disques sur OpenBSD varie dépendamment des architectures. Pour i386 et macppc, la configuration est faite en deux étapes. D'abord, la partition OpenBSD du disque dur est définie avec fdisk(8), ensuite cette partition est divisée en partitions OpenBSD avec disklabel(8).

Quelques utilisateurs peuvent être désorientés par la terminologie utilisée ici. Nous utiliserons le mot "partition" dans deux sens différents. Il y a deux niveaux de partitionnement pour certaines architectures sous OpenBSD, la première, peut être considérée comme le partitionnement des Systèmes d'Exploitation, qui permet à de multiples systèmes d'exploitation de définir leur espace sur le disque, et la seconde définie comment la partition OpenBSD est subdivisée en plusieurs

systèmes de fichiers individuels. La première couche est visible comme une partition de disque pour DOS, Windows ou tout autre système d'exploitation pouvant coexister avec d'autres systèmes sur les machines déscendantes de l'AT d'IBM. La seconde couche de partitionnement est visible seulement par OpenBSD et les systèmes d'exploitation pouvant lire directement un système de fichiers OpenBSD.

```
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
```

Le disque "root" est le disque depuis lequel le système démarrera et normalement sur lequel l'espace "swap" réside. Souvent, ce sera l'option par défaut -- si ce n'est pas le cas, vous devrez savoir comment forcer votre ordinateur à démarrer depuis un disque non standard. Les disques IDE vont être désignés comme `wd0`, `wd1`, etc., tandis que les disques SCSI et les matériels RAID apparaîtrons comme `sd0`, `sd1`, et ainsi de suite. Les disques que OpenBSD peut utiliser sont listés ici -- si vous avez des lecteurs qui ne sont pas montrés, votre matériel est soit non supporté, soit mal configuré.

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

Si vous répondez "yes" à cette question, l'intégralité du disque sera allouée à OpenBSD. Le résultat sera un "Master Boot Record" valide et une table de partition écrite sur le disque -- une partition, de la taille du disque entier, configuré avec le type de partition OpenBSD, et indiqué comme partition amorçable. Ce sera un choix courant pour la plupart des utilisations en production d'OpenBSD ; cependant, il y a quelques systèmes sur lesquels cela ne doit pas être fait. La plupart des systèmes Compaq, des ordinateurs portables, quelques DELL et d'autres systèmes utilisant une partition "maintenance" ou "Suspend to Disk", qui devrait être gardée intacte. Si votre système a d'autres partitions, quelque soit leur type, que vous ne voulez pas effacer, ne répondez pas "yes" à la question précédente.

Dans le cas de cet exemple, nous considêrerons que le disque doit être partagé entre OpenBSD et une partition Windows 2000 existante, donc nous répondrons l'option par défaut "no", qui nous enverra dans le programme fdisk(8). Vous pouvez obtenir plus d'informations sur fdisk(8) ici.

**Note importante :** Les utilisateurs avec un disque de grande capacité (plus de 8Go sur un nouveau i386, cependant sur des vielles machines et différentes architectures, souvent plus petites) voudrons surement consulter cette section avant d'aller plus loin.

```
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>
```

Quelques commandes doivent être expliquées :

- **`r`** ou **`reinit :`** Efface la table des partitions existantes, crée une grande partition OpenBSD, la définie active et installe le code MBR OpenBSD. Cela équivaut à répondre "yes" à la question "`use *all* of ...`" précédente.
- **`p`** ou **`print :`** Montre la table des partitions courante en secteurs. "`p m`" montrera la table en méga-octets, "`p g`" la montrera en giga-octets.
- **`e`** ou **`edit :`** Editer ou modifier une entrée de la table.
- **`f`** ou **`flag :`** Marquer une partition comme active, ce sera celle depuis laquelle le système démarrera.
- **`exit`** et **`quit :`** Soyez prudents, car certains utilisateurs ont l'habitude d'utiliser "exit" et "quit" qui ont des sens opposés.

Il est nécessaire de le rappeler, une erreur à ce stade entrainera de lourdes pertes de données. Si vous comptez faire cette manipulation sur un disque avec des données importantes, il serait plus judicieux de vous exercer sur un disque plus "disponible", en plus d'avoir une bonne sauvegarde.

Notre disque ici à une partition de 1.5Go pour Windows 2000 (utilisant le système de fichiers FAT). En regardant les informations ci-dessous, nous pouvons voir que la partition Windows occupe le lecteur jusqu'au cylindre 202. Nous allons donc allouer le reste du disque à OpenBSD en commençant au cylindre 203. Vous pouvez aussi calculer le secteur de début d'Openbsd étant 3069360, en ajoutant le secteur de début de partition existante (63) et sa taille (3069297).

Vous pouvez éditer la disposition du disque sur chaque Cylindre/Tête/Secteur ou simplement en secteurs bruts. Ce qui est le plus simple dépend de ce que l'on fait ; dans notre cas, travailler autour d'une partition existante, utiliser le format CHS sera probablement plus simple. Si vous créez la première partition sur le disque, utiliser les secteurs bruts sera probablement plus aisé.

```
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>
```

Il est important que la première partition saute la première piste du disque, dans notre cas, qu'elle commence au secteur `63`. Si une partition OpenBSD est crée en ayant pour origine le secteur `0`, la table de partition finira écrasée par le "Partition Boot Record" de la partition OpenBSD. Le système sera toujours amorçable, mais il sera très difficile à maintenir, et ce type de configuration n'est *ni recommandé ni supporté*.

Notez que l'invite a changée pour inclure un astérisque ('*') indiquant que nous avons des changements non enregistrés. Comme nous pouvons le voir à partir de la sortie `p m` nous n'avons pas altéré notre partition Windows, nous avons correctement alloué le reste du disque à OpenBSD et les partitions ne se chevauchent pas. Nous avons terminé. Presque.

Ce que nous n'avons pas fait est de définir active la partition pour que la machine l'amorce lors du prochain redémarrage :

```
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>
```

Maintenant, nous sommes prêts à enregistrer nos changements :

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

### Créer le "disklabel"

La prochaine étape est d'utiliser disklabel(8) pour diviser la partition OpenBSD. Plus de détails concernant disklabel(8) peuvent être trouvés sur la 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.
>
```

Une fois de plus, certaines commandes doivent être expliquées :

- **p** - Montre le partitionnement courant à l'écran, vous pouvez utiliser les arguments `k, m` ou `g` pour obtenir des kilo-octets, méga-octets ou giga-octets.
- **D** - Efface tout partitionnement existant au préalable, crée un nouveau partitionnement par défaut qui recouvre juste la partition OpenBSD courante. Ceci peut être utile si le disque contenait préalablement un partitionnement, et que la partition OpenBSD avait été recréée avec une taille différente -- l'ancien partitionnement n'aurais pas été supprimé, et aurait pu être déroutant.
- **m** - Modifie une entrée existante dans un partitionnement. Ne surestimez pas ce que cette option pourra vous permettre. Alors que vous pourrez modifier la taille d'une partition "disklabel", cela ne changera PAS le système de fichiers sur le disque. Utiliser cette option et vouloir redimensionner une partition existante est une bonne solution pour perdre de grandes quantités de données.

Diviser votre disque proprement est important. La réponse à la question, "Comment dois-je partitionner mon système ?" est "Exactement comme vous en avez besoin". Cela variera d'applications en applications. Il n'y à pas de réponse universelle. Si vous n'êtes pas sur de comment vous voulez partitionner votre système, regardez cette discussion.

Dans ce système, nous avons près de 17Go disponibles pour OpenBSD. Cela représente beaucoup d'espace, et nous n'aurons probablement pas besoin d'autant. Nous n'utiliserons donc pas les tailles minimales absolues. Nous préférerons avoir quelques centaines de mega-octets en trop plutôt qu'un kilo-octets en moins.

Sur le disque "root", deux partitions 'a' et 'b' **doivent** être créées. La procédure d'installation ne démarrera pas tant que ces deux partitions ne seront pas disponibles. 'a' sera utilisée comme racine du système de fichiers (/) et 'b' sera utilisé comme zone de "swap".

Après quelques réflexions, nous décidons de créer juste assez de partitions pour autoriser la création des systèmes de fichiers séparés qui est recommandée (`/, /tmp, /var, /usr, /home`) et une partition de "swap" :

- ***wd0a:*** */* (root) - 150Mo. Devrait être plus que suffisant.
- ***wd0b:*** (swap) - 300Mo.
- ***wd0d: /tmp*** - 120Mo. `/tmp` utilisé pour construire certains logiciels, 120Mo sera probablement suffisant pour la plupart des utilisations.
- ***wd0e: /var*** - 80Mo. Si ce fût un serveur web ou de mail, nous aurions fait une partition plus large, mais, ce n'est pas ce que nous souhaitons réaliser.
- ***wd0g: /usr*** - 2Go. Nous voulons que cette partition sois assez grande pour charger quelques applications utilisateurs, et en plus être capable de mettre à jour et reconstruire le système si voulu ou nécessaire. La Racine des "Ports" sera très bien ici, occupant moins de 100Mo avant la construction des "ports".
- ***wd0h: /home*** - 4Go. Cela permettra l'abondance de fichiers utilisateurs.

Maintenant, si vous ajoutez tout cela, vous verrez que plus de 10Go d'espace est inutilisé ! De l'espace inutilisé ne dérange en rien, et donne la flexibilité d'agrandir une de nos partitions dans le futur si le besoin se fait sentir. Vous voulez plus de */tmp* ? Pas de problèmes, créez en un nouveau dans l'espace non utilisé, changez le */etc/fstab* et le problème est réglé.

```
> 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
```

Vous noterez qu'il y a une partition *c* que nous avons vraissemblablement ignorée. Cette partition représente votre disque entier ; n'essayez pas de la modifier. Vous noterez aussi que la partition *i* n'as pas été définie par nous ; elle est la partition existante Windows 2000. Les partitions n'ont pas de lettre prédéfinie particulière -- à l'exception de *a* (racine), *b* (swap) et *c* (disque entier), le reste des partitions (jusqu'a la lettre *p*) est disponible pour l'utilisation que vous désirez en faire.

Si vous regardez de près à la sortie de disklabel, vous verrez que le taux de "RPM" est probablement faux. Ceci est historique ; la vitesse du disque n'est utilisée nulle part dans le système. Ne vous inquiétez pas de cela.

### Configurez vos points de montage et formatez vos systèmes de fichiers

Maintenant vient la configuration finale de vos points de montage. Si vous les avez configurés à travers disklabel(8), cette étape consiste juste en la vérification de vos sélections ; autrement, vous pouvez le spécifier maintenant.

```
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? [no] 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)
```

Vous devez vous demander pourquoi la procédure d'installation vous demande encore vos points de montage. Cela vous permet de vous abolir d'erreurs ou omissions sur les points de montage spécifiés pendant la création du partitionnement. Par exemple, la procédure d'installation va automatiquement supprimer les points de montages identiques que vous avez entrés lors du partionnement. Le programme "disklabel" vous autorisera de telles erreurs, ils doivent donc être vérifiés après la terminaison du programme "disklabel". Les points de montages en doubles qui ont été supprimés deviendrons des partitions sans point de

montage, auxquelles vous devez assigner de nouveaux points de montage si vous souhaitez utiliser l'espace.

Notez que la question "Are you really sure that you are ready to proceed?" a pour réponse par défaut "no", vous devrez donc délibérément lui indiquer de démarrer la procédure et de formatter vos partitions. Si vous choisissez "no", vous serez simplement renvoyé vers une invite de commande shell et pourrez relancer l'installation en tapant la commande "install", ou en redémarrant simplement depuis votre média d'amorçage.

A ce moment tous les systèmes de fichiers sont formatés pour vous. Cela peut prendre un certain temps, dépendamment de la taille des partitions et de la vitesse du disque.

### 4.5.3 - Configurer le nom d'hôte du système ("hostname")

Vous devez maintenant configurer le nom d'hôte du système ("hostname"). Cette valeur, avec le nom de domaine DNS (spécifié ci-dessous), sera sauvée dans le fichier `/etc/myname`, qui est utilisé pendant le démarrage normal pour configurer le nom d'hôte du système. Si vous ne configurez pas le nom de domaine du système, la valeur par défaut 'my.domain' sera utilisée.

Il est important de configurer ce nom maintenant, car il sera utilisé lorsque que les clés cryptographiques nécessaires au système serons générées lors du premier démarrage suivant l'installation. Cette génération intervient que le réseau soit configuré ou pas.

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

### 4.5.4 - Configurer le réseau

Maintenant il est temps de configurer votre réseau. Le réseau doit être configuré si vous voulez procéder à une installation par FTP ou NFS car elle sera basée sur les informations que vous aller entrer. Voici une entrevue de la section de configuration réseau de la procédure d'installation.

```
Configure the network? [yes] 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? [no] 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? [yes] Enter
Default route? (IP address, 'dhcp' or 'none') 199.185.137.128
add net default: gateway 199.185.137.128
Edit hosts with ed? [no] Enter
Do you want to do any manual network configuration? [no] Enter
```

Dans l'exemple précédent nous utilisons une adresse IP statique. Comme indiqué, il est possible d'utiliser le "`dhcp`" à la place sur la plupart des architectures (sauf Alpha), si votre environnement le supporte. Dans le cas du DHCP, la plupart des informations seront obtenues depuis le serveur distant DHCP ; vous aurez la possibilité de les confirmer. Voici un exemple de configuration réseau, cette fois cela est réalisé en utilisant DHCP :

```
Configure the network? [yes] 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? [no] 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? [yes] 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? [no] Enter
Do you want to do any manual network configuration? [no] Enter
```

**NOTE : Une** seule interface peut être facilement configurée en utilisant DHCP pendant l'installation. Si vous essayez de configurer plus d'une interface en utilisant DHCP vous allez au devant d'erreurs. Vous devrez configurer manuellement les autres interfaces après l'installation.

Maintenant, configurez le mot de passe pour le compte root :

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

Utilisez un mot de passe sécurisé pour le compte root. Vous créerez d'autres comptes utilisateurs après le démarrage du système. Extrait de 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 - Choisir le média d'installation

Une fois que votre réseau sera configuré, le script d'installation vous donnera la possibilité de faire des ajustements manuels à la configuration. Ensuite les systèmes de fichiers que vous avez créés seront montés et le mot de passe root configuré. Cela préparera vos disques à l'installation des packages de fichiers OpenBSD.

Ensuite, vous aurez la possibilité de choisir le média d'installation. Les options sont listées ci-dessous.

```
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.
```

Dans cet exemple nous installons depuis le CD-ROM. Cela nous donnera une liste des matériels sur notre machine identifiés comme lecteurs de CD-ROM. La plupart des gens n'en auront qu'un. Si vous en avez besoin, soyez sûr d'indiquer le bon matériel que vous souhaitez utiliser pour installer OpenBSD.

**NOTE :** Toutes les sources possibles sont listées, mais toutes ne sont pas forcement disponibles sur votre système. Ex. (N)fs est montré mais toutes les architectures n'autorisent pas les installations NFS. Si vous choisissez une source qui n'est pas disponnible, vous recevrez un message d'erreur et vous aurez la possibilité de choisir une autre source pour vos packages d'installation.

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

Pathname to the sets?  (or 'done') [3.5/i386] Enter
```

Ici, on vous demande dans quel dossier se trouvent les fichiers d'installation, celui-ci est `3.5/i386/` sur les CD-ROM officiels.

### 4.5.6 - Choisir les packages

Maintenant il est temps de choisir quels packages vous aller installer. Vous pouvez avoir une description de ces fichiers dans la section suivante. Les fichiers que le programme d'installation détecte sont montrés à l'écran. Votre travail est simplement de spécifier quels fichiers vous souhaitez installer. Par défaut tous les packages ne concernant pas X sont sélectionnés ; cependant, certaines personnes peuvent préférer se limiter au strict minimum requis pour démarrer OpenBSD, qui devrait être `base35.tgz`, `etc35.tgz` et `bsd`. D'autres vondrons installer tous les packages. L'exemple suivant est celui d'une installation complète.

```
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] base35.tgz
        [X] etc35.tgz
        [X] misc35.tgz
        [X] comp35.tgz
        [X] man35.tgz
        [X] game35.tgz
        [ ] xbase35.tgz
        [ ] xshare35.tgz
        [ ] xfont35.tgz
        [ ] xserv35.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] base35.tgz
        [X] etc35.tgz
        [X] misc35.tgz
        [X] comp35.tgz
        [X] man35.tgz
        [X] game35.tgz
        [X] xbase35.tgz
        [X] xshare35.tgz
        [X] xfont35.tgz
        [X] xserv35.tgz
```

Vous pouvez faire toutes sortes de combinaisons ici -- `-x*` déselectionnera tous les composants X. Dans notre cas, nous allons installer tous les packages. Bien que le système se lancera avec un minimum de packages, la sélection par défaut ou une installation complète sont recommandées. Plus de détails sur la sélection des packages sont disponibles ici.

Une fois que vous avez choisi les packages que vous désirez, il vous sera demandé si vous êtes d'accord pour décompresser puis installer ces derniers. Une barre de progression sera affichée pour vous informer du temps nécessaire. Les temps varient ennormément en fonction du système sur lequel vous installez OpenBSD,

les packages que vous avez sélectionné et la vitesse du média source. Cela va de quelques minutes à plusieurs heures.

```
File Name? (or 'done') [done] Enter
Ready to install sets? [yes] Enter
Getting bsd ...
100% |**************************************************|  4735 KB    00:03
Getting bsd.rd ...
100% |**************************************************|  4275 KB    00:02
Getting base35.tgz ...
100% |**************************************************| 30267 KB    00:21
Getting etc35.tgz ...
100% |**************************************************|  1545 KB    00:01
Getting misc35.tgz ...
100% |**************************************************|  1909 KB    00:01
Getting comp35.tgz ...
100% |**************************************************| 17074 KB    00:13
Getting man35.tgz ...
100% |**************************************************|  6139 KB    00:04
Getting game35.tgz ...
100% |**************************************************|  2534 KB    00:01
Getting xbase35.tgz ...
100% |**************************************************| 10940 KB    00:06
Getting xshare35.tgz ...
100% |**************************************************|  1656 KB    00:02
Getting xfont35.tgz ...
100% |**************************************************| 31160 KB    00:21
Getting xserv35.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')
```

A ce point, vous pouvez spécifier des fichiers additionnels venant d'autres sources (incluant les packages personnalisés) si vous le souhaitez, ou tapez '`done`' si vous avez installé tous les packages dont vous avez besoin.

### 4.5.7 - Terminer l'installation

Ensuite, plusieurs questions vous seront posées concernant les paramètres de votre système une fois installé. La première est si sshd(8) doit être lancé au démarrage. La plupart du temps, vous souhaiterez que sshd(8) se lance, mais occasionnellement vous ne le souhaiterez pas. Si votre configuration n'a aucune nécessité de sshd(8), il y a un petit avantage de sécurité théorique à ne pas le lancer.

```
Do you wish sshd(8) to be started by default? [yes] y
```

On vous demande maintenant si vous souhaitez lancer X sur votre système. Si vous répondez 'Y', `/etc/sysctl.conf` sera modifié pour inclure la ligne `machdep.allowaperture=1` ou `machdep.allowaperture=2`, dépendamment de votre architecture. Sous certaines architectures cette question ne sera pas posée du tout.

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

Votre dernière tâche est d'entrer votre "time zone". Dépendemment de l'endroit où votre machine réside, il peut y avoir plusieurs réponses équivalentes à la question. Dans l'exemple suivant, nous utilisons `US/Eastern`, mais l'on pourrait aussi utiliser `EST5EDT` ou `US/Michigan` et obtenir le même résultat. Appuyer sur `?` à l'invite vous aidera dans vos choix.

```
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.
```

Si vous êtes intéressé par un horodatage vraiment précis, vous devriez lire ceci.

Les étapes suivantes permettent au système de créer le dossier `/dev` (ce qui pourra prendre du temps sur certains systèmes, surtout si vous avez peu de RAM), et installer les blocs d'amorce.

```
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 est maintenant installé sur votre système et prêt pour son premier démarrage, mais avant cela...

#### Avant de redémarrer

A ce point, votre système est installé et prêt à être redémarré et configuré pour votre service. Avant de faire cela, cependant, il serait sage de consulter la page d'Errata pour voir s'il existe des bugs qui vous pourraient vous concerner.

#### Après avoir redémarré

Une des premières choses que vous devrez lire après avoir installé votre système est afterboot(8).

Vous devriez aussi trouver ces différents liens utiles :

- Ajouter des utilisateurs sur OpenBSD
- Configuration initiale du réseau
- Page des manuels de commandes populaires/utiles
- Page de manuel OpenBSD sur le web
- Le système de Ports et de Packages OpenBSD pour installer des logiciels, aussi bien que ceci et ceci

**Une dernière chose...**

Les développeurs OpenBSD vous demandent de leur Envoyer une copie de votre "dmesg". Celle-ci est vraiment utile à ces derniers, et au final, à tous les utilisateurs.

# 4.6 - Quels sont les fichiers nécessaires à l'installation ?

L'installation complète d'OpenBSD est divisée en plusieurs *packages de fichiers* séparés. Toutes les utilisations ne requièrent pas tous les packages. Voici une vue d'ensemble de chacun :

- ***bsd*** - Ceci est le noyau. **Requis**
- ***bsd.mp*** - Noyau pour les systè multi-processeurs (SMP) (seulement disponible pour quelques plateformes, uniquement *-current*)
- ***bsd.rd*** - Kernel de disque RAM
- ***base35.tgz*** - Contient le système de base OpenBSD **Requis**
- ***etc35.tgz*** - Contient tous les fichiers de /etc **Requis**
- ***comp35.tgz*** - Contient le compilateur et ses outils, en-têtes et librairies **Recommandé**
- ***man35.tgz*** - Contient les pages de manuels **Recommandé**
- ***misc35.tgz*** - Contient différentes informations et documentations de configuration
- ***game35.tgz*** - Contient les jeux pour OpenBSD
- ***xbase35.tgz*** - Contient l'installation de base pour X11
- ***xetc35.tgz*** - Contient les fichiers de configuration /etc/X11 et /etc/fonts (uniquement *-current*)
- ***xfont35.tgz*** - Contient le serveur de fontes X11 et les fontes
- ***xserv35.tgz*** - Contient les serveur X de X11
- ***xshare35.tgz*** - Contient les pages de manuels, les options de localisations, les inclusions, etc. pour X

Les packages **etc35.tgz** et **xetc35.tgz** ne sont pas installés lors d'une mise à mais uniquement lors d'une installation complète, de ce fait, toute configuration que vous ferez sera conservée. Vous devrez mettre à jour vos dossiers `/etc`, `/dev` et `/var` manuellement.

# De combien d'espace disque ai-je besoin pour une installation OpenBSD ?

Ce qui suit est le minimum suggéré pour les tailles de système de fichiers lors d'une installation complète. Les chiffres incluent un peu d'espace supplémentaire vous permettant de lancer un système connecté à l'internet.

- Ces valeurs sont les valeurs minimales.
- Si vous voulez installer un certain nombre de logiciels tierces, faites vous une grosse partition `/usr` ! Triplez **au moins** ces valeurs !
- Pour un système qui devra garder un nombre de mails et de pages web conséquent (enregistrés respectivement, dans `/var/mail` et `/var/www`) vous aurez besoin d'une grande partition `/var` , ou de les stocker dans des partitions différentes.
- Pour un système multi-utilisateurs qui pourrait générer beaucoup de journaux, vous devrez avoir une partition `/var` de taille confortable (`/var/log`).
- Si vous pensez réconstruire le noyau ou le système depuis les sources, vous devrez avoir une partition `/usr` volumineuse, **au moins** de 800Mo-1Go comme indiqué ci-dessous.

En lisant cela, gardez à l'esprit que les répertoires `/usr` et `/usr/X11R6` sont souvent tous deux parties du même système de fichiers, qui est `/usr`, du fait qu'il n'y a pas de gros avantage à les séparer en deux systèmes de fichiers.

| 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 |

En complément, il est recommandé qu'une partition `/tmp` soit utilisée. La partition `/tmp` est utilisée dans la compilation des ports, parmis d'autres choses, donc la taille que vous lui donnez dépendra de ce que vous en ferrez. 50Mo devraient être suffisants pour la plupart des gens, mais quelques grosses applications peuvent nécessiter 100Mo ou plus d'espace dans `/tmp`.

Lorsque vous être dans l'editeur de "disklabel", vous pourriez choisir de faire votre système de telle sorte qu'il n'ai qu'une partition 'a' (système de fichier principal) et 'b' (zone de "smap"). Le système de fichiers 'a' que vous avez configuré dans "disklabel" va devenir votre partition racine, qui devrait être la somme des trois valeurs précédentes (`/`, `/usr`, et `/var`) plus un peu de place pour `/tmp`. La partition 'b' que vous configurez devient automatiquement votre partition swap système -- nous recommandons un minimum de 32Mo mais si vous avez un peu de place disponnible, réservez au minimum 64Mo. Si vous avez beaucoup de place disponnible, faite une partition de 256Mo, ou même 512Mo.

L'espace "swap" est utilisé pour sauvegarder les copies de la mémoire dans l'éventualité d'un crash(8). Si cela entre en ligne de considération pour vous, votre zone de swap devra être au moins égale à la taille de mémoire vive dont vous pourrez disposer dans le système. Notez que lors du redémarage, savecore(8) essayera de sauvegarder le contenu de la partition swap dans un fichier stocké dans `/var/crash` donc une fois de plus, si cela est une priorité pour vous, votre partition `/var` devra avoir suffisement d'espace disponnible pour enregistrer ces fichiers de copie mémoire.

Il y a cinq raisons principales a utiliser des systèmes de fichiers séparés plutot que de n'en utiliser qu'un seul pour stocker tout dans un seul ou deux systèmes de fichiers :

- **Sécurité** : Vous pouvez marquer certains systèmes de fichiers d'un 'nosuid', 'nodev', 'noexec', 'readonly', etc. Cela est maintenant fait par la procédure d'installation, si vous utilisez les partitions décrites ci-dessus.
- **Stabilité** : Un utilisateur, ou un programme se conduisant mal, peut remplir votre système de fichiers s'ils ont les droits d'écriture pour le faire. Vos programmes critiques, qui bien sûr sont lancés sur un système de fichier à part, ne seront pas interrompus.
- **Vitesse** : Un système de fichiers sur lequel on écris souvent peut devenir quelque peu fragmenté. (Par chance, le système de fichiers ffs, que OpenBSD utilise, n'est pas enclin à être très fragmenté.)
- **Intégrité** : Si un système de fichiers est corrompu pour quelque raison que ce soit, les autres restent en bon état.
- **Taille** : Beaucoups de machines ont une limite sur la taille de la zone d'un disque d'où la ROM de démarrage peut charger le noyau. Dans certains cas, cette limite peut être vraiment basse (504Mo pour un ancien 486), dans d'autre cas, elle peut être plus grande (par exemple, 2Go, 8Go ou 128Go sur les systèmes i386). Comme le système peut être installé n'importe où sur la partition racine, la totalité de la partition racine devrait être sur cette zone. Pour plus de détails, regardez cette section. Une bonne ligne de conduite peut être de conserver votre partition / inférieure à 2Go, dans le cas où vous ne savez pas si votre architecture (et votre machine) peuvent en supporter plus (ou moins !).

Quelques remarques sur le partitionnement :

- Pour votre première tentative sur un système d'expérimentation, une grosse partition `/` et une zone "swap" devraient être plus simples jusqu'à ce que vous sachiez combien de place vous est nécessaire. En faisant cela vous sacrifierez certaines fonctions de sécurité d'OpenBSD qui requièrent des systèmes de fichiers séparés pour `/, /tmp, /var, /usr` et `/home`.
- Un système exposé à l'internet ou d'autres forces hostiles devrait avoir une partition `/var` séparée (et peut être même une partition `/var/log` séparée) pour la journalisation.
- Une partition `/home` peut être intéressante. Nouvelle version de l'OS ? Supprimez et rechargez tout le reste, conservez votre partition `/home` intacte. Rappelez vous de garder une copie de vos fichiers de configuration tout de même.
- Une partition séparée pour tout ce qui risque d'accumuler une grosse quantité de fichiers qui devront être supprimée peut être plus rapide à reformater re-créér que de supprimer les fichiers. Regardez la Minifaq de mise à jour pour un exemple (`/usr/obj`).
- Si vous souhaitez reconstruire votre système depuis les sources pour quelque raison que ce soit, le sources seront dans `/usr/src`. Si vous ne faites pas une partition séparée pour `/usr/src`, soyez sûr que `/usr` a suffisament de place.
- Un fait souvent oublié : vous n'avez **pas** besoin d'allouer toute la place sur un disque quand vous configurez un système ! Etant donné qu'il est maintenant compliqué de trouver un disque dur plus petit que 20Go, il devient sensé de laisser une certaine place du disque non allouée. Si vous dépassez la taille d'une partition, vous pouvez allouer une nouvelle partition depuis l'espace non utilisé, dupliquer votre partition existante sur la nouvelle, changer /etc/fstab pour pointer vers votre nouvelle partition, la remonter, vous avez maintenant plus d'espace.
- Si vous créez vos partitions avec une taille trop proche de la taille minimum requise, vous allez probablement le regretter plus tard lorsqu'il sera temps de mettre à jour votre système.
- Si vous autorisez les utilisateurs à écrire dans `/var/www` (ex., pages web personnelles), vous devriez le placer dans une partition separée, vous pouvez par exemple utiliser les quotas pour restreindre la place qu'il utiliserons, s'ils remplissent la partition, les autres parties du système ne seront pas affectées.

## 4.8 - Multiboot OpenBSD/i386

Le "Multibooting" est le fait d'avoir plusieurs systèmes d'exploitation sur le méme ordinateur, et de pouvoir choisir depuis lequel vous souhaitez démarrer. Ce n'est *pas* une tâche triviale ! Si vous ne comprennez pas ce que vous êtes en train de faire, vous finirez par perdre une somme conséquente de données sur votre ordinateur. Les nouveaux utilisateurs OpenBSD sont *vivement* encouragés a démarrer avec un disque dur vierge et sur une machine dédiée, afin d'essayer la configuration désirée sur un système qui n'est pas en production avant d'installer une configuration "multiboot" sur une machine de production. La FAQ 14 donne plus d'informations sur la procédure d'amorçage OpenBSD.

Voici plusieurs options pour le "multiboot" :

### Configurer la partition active

C'est probablement la solution la plus négligée, et parfois la plus intéressante pour le "multiboot". Configurez simplement comme partition active, la partition d'OS depuis laquelle vous souhaitez démarrer par défaut au prochain démarrage. Chaque OS offre un programme pour faire ceçi ; celui d'OpenBSD est fdisk(8), des programmes portant des noms similaires sont disponibles sous Windows 9x et DOS, et la plupart des autres systèmes d'exploitation. Ceci peut être très utile pour les OS ou systèmes long à arrêter et redémarrer -- vous pouvez le configurer et lancer la procédure de redémarrage, ensuite aller faire un tour, prendre une tasse de café, et revenir devant le système démarré comme vous le souhaitiez -- pas d'attente du Moment Magique pour choisir le système d'exploitation désiré.

### Disquette d'amorçage

Si vous avez un système qui utilise OpenBSD non fréquement (ou que vous ne voulez pas que les autres utilisateurs de l'ordinateur notent que quoi que ce soit ai changé), vous pouvez utiliser une disquette d'amorçage. Utilisez simplement l'une des disquettes d'installation standard d'OpenBSD, et créez un fichier `/etc/boot.conf` (oui, vous devrez aussi créer un dossier `/etc` sur la disquette) ayant le contenu suivant :

```
boot hd0a:/bsd
```

pour que le système démarre sur le disque dur 0, la partition OpenBSD 'a' et le fichier de noyau `/bsd`. Notez que vous pouvez aussi démarrer d'autres disques avec une ligne comme : "`boot hd2a:/bsd`" pour lancer le troisième disque dur de votre système. Pour lancer OpenBSD, insérez la disquette dans le lecteur et redémarrez. Pour lancer un autre système d'exploitation, éjectez la disquette et redémarrez.

Dans ce cas, le programme boot(8) chargé depuis la disquette, cherche et lis `/etc/boot.conf`. L'instruction "`boot hd0a:/bsd`" indique à boot(8) depuis quel endroit charger le noyau -- dans ce cas, le premier disque dur que le BIOS voit. Gardez à l'esprit que seulement un petit fichier (`/boot`) est chargé depuis la disquette -- le système charge le noyau entier depuis le disque dur, cela ne ralenti que de quelques secondes la procédure de démarrage.

### Windows NT/2000/XP NTLDR

Pour un "multiboot" entre OpenBSD et Windows NT/2000/XP, vous pouvez utilisez NTLDR, le chargeur de démarrage que NT utilise. Pour "multi-booter" avec NT, vous aurez besoin d'un copie de votre "Partition Boot Record" (PBR) OpenBSD. Après avoir lancé "installboot" vous pouvez en obtenir une copie dans un fichier en utilisant dd(1):

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

Maintenant démarrez sous NT et mettez `openbsd.pbr` dans C:. Ajoutez une ligne comme celle-ci à la fin du fichier `C:\BOOT.INI` :

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

Quand vous redémarrerez, vous devriez être en mesure de choisir OpenBSD dans le menu de chargement NT. D'autres informations sur le NTLDR sont disponibles dans le "NTLDR Hacking Guide".

Sur Windows XP vous pouvez aussi éditer les informations en utilisant la "GUI" ; consultez le XP Boot.ini HOWTO.

Des programmes faisant la plupart de ce travail sont à votre disposition, par exemple BootPart. Ce programme peut être lancé depuis Windows NT/2000/XP, et ira chercher le PBR OpenBSD, le mettre dans votre partition NT/2000/XP partition, et le rajouter dans `C:\BOOT.INI`.

La procédure d'installation et de mise à jour d'OpenBSD va réinstaller le chargeur de démarrage, dont la localisation est codée dans le PBR, donc si vous réinstallez ou mettez à jour OpenBSD, vous devrez recommencer l'étape ci-dessus pour prendre une nouvelle copie du PBR OpenBSD.

Note : Le chargeur de démarrage Windows NT/2000/XP est seulement capable de démarrer des systèmes depuis le premier disque dur. Vous ne pouvez pas l'utiliser pour charger OpenBSD depuis le second disque sur un système.

### Autres chargeurs de démarrage

D'autres utilisateurs de chargeurs de démarrage OpenBSD ont inclus avec succès GAG, OS-BS, The Ranish Partition Manager et GRUB.

### OpenBSD et Linux (i386)

Veuillez vous réferer au INSTALL.linux, qui donne les instructions en profondeur pour faire fonctionner OpenBSD avec Linux.

# 4.9 - Envoyer votre dmesg à dmesg@openbsd.org après l'installation

Rappelez vous, il est important pour les développeurs OpenBSD de garder une trace de quels métériels fonctionnent, et de quels matériels ne fonctionnent pas parfaitement.

Un commentaire de /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!
```

Soyez sur d'envoyer le mail depuis un compte sous lequel pour serez habilité à recevoir pour que les développeurs puissent vous contacter s'il ont quelque chose qu'il voudraient que vous testiez ou changiez pour que votre configuration fonctionne. Il n'est pas important d'envoyer le mail depuis la même machine que celle sur laquelle tourne OpenBSD, donc si cette dernière n'est pas en mesure de recevoir des mails, faites simplement :

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

et tranférez le message à

```
 dmesg@openbsd.org
```

Où `your-account@yourmail.dom` est votre compte de messagerie régulier. (ou tranférez votre sortie dmesg en utilisant FTP/scp/floppydisk/pigeon-porteur/...)

**NOTE** - N'envoyez que des dmesg concernant les noyaux GENERIC. Les noyaux personnalisés qui ont des drivers de matériels en moins ne sont pas utiles.

Notre aussi que les dmesgs sont reçus sur un ordinateur utilisant le système de filtrage de spam spamd. Cela peut causer le rejet temporaire de votre dmesg par les serveurs de mails. Soyez patient, après une demi-heure voire une heure, il sera reçu.

# 4.10 - Ajouter un package après l'installation

"Oh non ! J'ai oublié de rajouter un package quand j'ai fait l'installation !"

Parfois, vous réalisez que vous AURIEZ vraiment eu besoin de `comp35.tgz` (ou de n'importe quel composant système) après tout, mais vous ne l'avez pas réalisé quand vous avez installé votre système. Bonne nouvelle : Il y a deux voies relativement simples pour rajouter un package après l'installation initiale :

## En utilisant la procédure de mise à jour

Démarrez simplement votre média d'installation (CD-ROM ou disquette), et choisissez "Upgrade" (plutôt que "Install"). Quand vous aurez la liste des packages, sélectionnez simplement celui que vous avez oublié d'installer la première fois, choisissez la source, et laissez-le l'installer pour vous.

## En utilisant tar(1)

Les packages d'installation sont de simples fichiers compressés tar, et vous pouvez les décompresser vous même manuellement depuis la racine du système de fichiers.

```
# cd /
# tar xzvpf comp35.tgz
```

N'oubliez PAS l'option `'p'` ci-dessus qui restaurera correctement les permissions sur les fichiers !

Une méprise courante est de croire qu'il est possible d'utiliser pkg_add(1) pour rajouter des packages manquants. Cela ne fonctionne pas. pkg_add(1) est pour les fichiers packages, pas pour les fichiers archives tar comme les sets d'installation.

# 4.11 - Qu'est ce que 'bsd.rd' ?

**`bsd.rd`** est un noyau "RAM Disk". Ce fichier peut être vraiment intéressant ; beaucoup de développeurs prudents en plaçent un tout le temps à la racine de leur système.

Le noyau "RAM Disk" définit la racine du système de fichiers du noyau -- plus qu'un étant un disque physique, les utilitaires disponibles après l'ammorcage de `bsd.rd` sont enregistrés dans le kernel, et lancé depuis un système de fichiers basé en mémoire RAM. `bsd.rd` inclu aussi une floppée d'utilitaires de sauvetage vous permettant de faire de la maintenance système et de lancer une l'installation.

Sur certaines architectures, `bsd.rd` est actuellement la méthode d'installation pribilégiée -- vous placez ce noyau sur un système de fichiers, vous le démarrez, et lancez l'installation depuis ce dernier. Sur la plupart des architectures, si vous utilisez une ancienne version d'OpenBSD, vous pouvez obtenir une nouvelle version de `bsd.rd` par FTP, redémarrer à partir de lui, et installer la nouvelle version d'OpenBSD sans avoir besoin de quelque média amovible que ce soit.

Voici un exemple d'amorçage de `bsd.rd` sur un système i386 :

```
  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 . . .
```

Comme indiqué, vous allez être amené au programme d'installation, mais vous pouvez aussi aller à une invite de shell pour faire de la maintenance sur votre système.

La règle générale en lançant `bsd.rd` est de changer votre noyau d'amorçage de `/bsd` à `bsd.rd` quelque soit la méthode pour votre architecture.

# 4.12 - Problèmes d'installation courants

## 4.12.1 - Mon Compaq ne reconnait que 16Mo de RAM

*Certains* systèmes Compaq rencontrent un problème où la mémoire RAM n'est pas complètement détectée par le Chargeur d'ammorçage de second niveau OpenBSD et seulement 16Mo seront détectés et utilisés par OpenBSD. Ceçi peut être corrigé en créant/édiant le fichier `/etc/boot.conf`, ou en entrant des commandes à l'invite "`boot>`" avant qu'OpenBSD ne se charge. Si vous avez une machine avec 64Mo de RAM, mais qu'OpenBSD n'en a détecté que 16Mo, la commande devrait être :

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

pour ajouter 48Mo (`0x3000000`) après les premiers 16Mo (`0x1000000`). Typiquement, si vous avez une machine avec ce problème, vous devriez entrer la commande précédente d'abord dans l'invite `boot>` du CD-ROM ou de la Disquette, charger la disquette, rédémarrer et créer un fichier `/etc/boot.conf` avec la ligne précédente pour que dans les démarrages suivants OpenBSD reconnaisse toute la mémoire disponnible.

Une mise à jour ROM règlera ce problème sur *certains* systèmes.

## 4.12.2 - Mon i386 ne démarre pas après l'installation

Votre installation à eu l'air de bien se dérouler, mais lors de votre premier démarrage, vous ne voyez aucun signes montrant qu'OpenBSD essaye de démarrer. Il y a plusieurs problèmes courant pouvant expliquer ce problème :

- **Aucune partition n'a été définie active dans fdisk(8).** Pour corriger cela, relancez la machine en utilisant une disquette de démarrage ou tout autre média, et marquez une partition comme active. Regardez ici et ici
- **Aucun chargeur de démarrage valide n'a été installé sur le disque.** Si vous avez répondu "Y" à la question "Use entire disk for OpenBSD?" pendant

l'installation, ou utilisé l'option "`reinit`" de fdisk(8), l'ammorce OpenBSD à été installée sur le "Master Boot Record" du disque ; autrement, le programme d'amorce est conservé intact. Ce sera un problème si aucun autre programme d'amorce n'existe. Une solution est de démarrer le média d'installation une nouvelle fois, basculer dans le shell et invoquer la commande fdisk(8) pour mettre à jour le MBR depuis la ligne de commande :

```
# fdisk -u wd0
```

Note : l'option "update" du mode interactif ("-e") de fdisk n'écrira pas les bits de signature requis pour rendre le disque amor&ccdeil;able.

- **Dans quelques rares cas, quelque chose s'est mal déroulé dans l'installation du chargeur de démarrage de stage 2.** La réinstallation du chargeur de démarrage de stage 2 est vue ici.

### 4.12.3 - Ma (vielle et lente) machine à démarrée, mais bloque pendant la procédure ssh-keygen

Il semble que votre machine fonctionne correctement, mais prend juste beaucoup de temps pendant la procédure de génération de clés ssh. Une SPARCStation2 ou un Macintosh Quadra peut prendre 45 minutes ou plus pour terminer les trois étapes ssh-keygen(1), quelques machines peuvent même prendre plus de temps. Laissez le simplement terminer ; cela n'est réalisé qu'une fois par installation.

### 4.12.4 - J'ai le message "Failed to change directory" pendant l'installation

Quand vous faites l'installation d'un "snapshot" pendant le stage -*beta* du cycle de développement OpenBSD, vous devriez voir ceci :

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

Cela est normal et souhaité pendant la version précédent la sortie officielle dans le cycle. Le programme d'installation cherche la liste FTP sur le premier serveur FTP dans un dossier qui ne sera pas disponible avant la date de "release", donc vous obtiendrez les messages précédents.

Utilisez simplement la liste de miroirs FTP pour trouver votre site mirroir FTP favori, et entrez manuellement son nom lorsque cela vous est demandé.

**Note : Vous ne devriez pas voir cela si vous installez une version *"-release"* ou depuis un CD-ROM.**

### 4.12.5 - Quand je m'identifie j'obtiens "login_krb4-or-pwd: Exec format error"

Kerberos IV à été retiré d'OpenBSD 3.4, mais si vous avez fait une mise à jour, les anciens binaires de Kerberos IV seront toujours sur votre système. Ceçi est un problème lié à l'architecture i386, car les anciens fichiers Kerberos sont au format a.out, et donc impossible à lancer sur le noyau standart ELF (sur lequel les émulations de format a.out sont désactivées, comme mentionné ici). Si vous rencontrez ce problème , vous devrez écraser la méthode d'authentification krb4 lorsque vous vous identifierez :

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

login: joeuser:passwd
password:
```

Vous pouvez utiliser la même syntaxe "*`username`*`:passwd`" pour une connexion ssh et avec su(1) pour accéder à votre système Maintenant éditez `/etc/login.conf`, et retirez les références à krb4.

### 4.12.6 - Ma table de partition fdisk est vide !

Occasionellement, un utilisateur trouvera un système fonctionnant, mais en faisant un `fdisk wd0`, il trouvera une table de partitions vide (ou poluée). Cela est usuellement du à la création d'une partition dans fdisk(8) ayant un offset de zero secteurs, au lieu de l'offset d'une piste qu'elle est sencée avoir. (note : cela ne concerne que les plateformes i386 et amd64. Les autres plateformes requièrents des offsets différents, certaines n'en requièrent pas) Le système démarre ensuite en utilisant le PBR, pas le MBR.

Tandis que cette configuration peut fonctionner, cela peut causer des problèmes de maintenance et devrait être corrigé. Pour corriger ce problème, le système de fichiers doit généralement être recrée depuis le dÃ©but (si vous savez VRAIMENT ce que vous faites, vous devrier être en mesure de recréer juste le disklabel et le MBR et ne perdre que la première partition OpenBSD du disque).

## 4.13 - Personnaliser la procédure d'installation

### Fichier `siteXX.tgz`

Les scripts d'installation/mise à jour d'OpenBSD autorisent la création d'un set utilisateur nommé "`siteXX.tgz`", où XX représente la version (ex. 35). Le fichier `siteXX.tgz` est, comme les autres packages, une compression gzip(1) d'archive tar(1) dans la racine est '/' et est décompressé comme les autres avec les options `xzpf`. Ce package sera installé en dernier, après tous les autres packages.

Ce package vous permet d'ajouter et/ou écraser des fichiers installés dans les packages 'normaux' et donc de personnaliser l'installation ou la mise à jour.

Quelques exemples d'utilisation de fichier `siteXX.tgz` :

- Créer un fichier siteXX.tgz qui contient tous les changements que vous avez fait depuis la première installation de OpenBSD. Ensuite, si vous avez à récréer le système sélectionnez simplement siteXX.tgz pendant la procédure de réinstallation et toutes les modifications que vous avez faites seront répliquées sur le nouveau système.
- Créer une série de dossiers spécifiques machine qui contiennent chacun un fichier siteXX.tgz dans lequel se trouvent les fichiers spécifiques à la machine. L'installation de machines (ex. machines avec des cartes graphiques différentes) d'une catégorie particulière peut être faite en choisissant le fichier siteXX.tgz approprié.
- Mettez les fichiers que vous paramétrez constamment dans un fichier siteXX.tgz similaire -- fichiers `/etc/skel`, `/etc/pf.conf`, `/var/www/conf/httpd.conf`, `/etc/rc.conf`, etc.

### Scripts `install.site/upgrade.site`.

A la dernière étape de la procédure d'installation/mise à jour, le script cherche dans la racine un `install.site` ou un `upgrade.site` d'un nouveau système ou d'une nouvelle mise à jour, dépendemment de la procédure en cours, et lande le script dans un environnement chrooté de la racine système de l'installation/de la mise à jour. Rapellez vous, la mise à jour est faite depuis un système de fichier démarré donc votre système de fichiers cible est actuellement monté dans `/mnt`. Cependant, votre script peut être écrit tel quel, comme s'il était écrit dans la racine normale de votre système de fichiers. Comme ce script est lancé après que tous les fichiers aient été installés, vous avez un système totalement fonctionnel (bien que lancé en mode mono-utilisateur) quand votre script est invoqué.

Notez que le script `install.site` devra être placé dans un fichier `siteXX.tgz`, tandis que le script `upgrade.site` pourra être placé à la racine du système de fichier avant la mise à jour ou bien être placé lui aussi dans fichier `siteXX.tgz`.

Ce script peut être utilisé pour faire tout ce qui est possible dans un script.

- Supprimer des fichiers qui sont installés/mis à jour et que vous ne souhaitez pas présent dans votre système.
- Supprimer/mettre à jour/installer les packages que vous souhaitez sur le système installé.
- Faire une sauvegarde/archive immédiate de votre nouveau système avant de l'exposer au reste du monde.

La combinaison de `siteXX.tgz` et de `install.site/upgrade.site` à pour but de donner de larges capacités de personnalisation sans avoir à créer ses propres packages d'installation.

## 4.14 - Comment puis-je installer plusieurs systèmes identiques ?

Voici quelques utilitaires que vous pouvez utiliser lorsque vous avez plusieurs systèmes OpenBSD identiques à déployer.

### Les fichiers `siteXX.tgz` and `install/upgrade.site`

Voir l'article précédent.

### Restauration depuis dump(8)

Sur la plupart des architectures, le média de démarrage inclut le programme restore(8) qui peut être utilisé pour restaurer une sauvegarde faite par dump(8). Ainsi, vous pouvez démarrer depuis disquettes, CD, ou fichier bsd.rd, ensuite fdisk, disklabel, et restore pour restaurer la confiuration désirée depuis une bande ou autre média, et installer les blocs de d'amorçe. Plus de détails ici.

### Image de disque

Malheureusement, il n'existe pas de package d'image de disque attentif au FFS qui peuvent faire une image contenant simplement l'espace de disque utilisé. La plupart des solutions d'image de disque traiterons la partition OpenBSD comme une partition "générique", et pourront simplement faire l'image de l'intégralité du disque. Cela rejoint souvent notre but, mais souvent avec d'enormes quantités d'espace perdu -- une partition `/home` de 10Go vide demandera 10Go d'espace

dans l'image, même s'il n'y a aucun fichier à l'intérieur. Tandis que vous pouvez typiquement installer une image de disque sur un disque plus grand, vous ne pourrez pas l'installer sur un disque de plus petite taille.

Si cela est pour vous acceptable, vous devriez trouver dans la commande dd tout ce dont vous avez besoin, autorisant la copie d'un disque vers un autre, secteur-par-secteur. Celui-ci vous donnera souvent les mêmes fonctionalités que les programmes commerciaux, sans le prix.

## 4.15 - Comment puis-je obtenir un dmesg(8) pour rapporter un problème d'installation ?

Lorsque vous reportez un problème, il est important d'inclure le dmesg(8) complet du système. Cependant, souvent lorsque vous en avez besoin, c'est car le système ne fonctionne pas proprement ou ne s'installe pas donc vous n'aurez pas de disque, de réseau ou autre ressource vous permettant d'envoyer votre dmesg à la mailing liste appropriée. Il y a d'autres façons de faire cela, cependant :

- **Disquette :** Les disques d'amorces et CD-ROM ont assez d'outils pour permettre d'enregistrer votre dmesg sur une disquette MSDOS pour le lire sur une autre machine. Mettez une disquette MSDOS formatée dans votre lecteur de disquette et tapez les commandes suivantes :

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

  Si vous avez un autre système OpenBSD, vous pouvez aussi écrire sur une disquette compatible OpenBSD -- souvent, la disquette d'amorce à encore assez d'espace libre pour contenir le dmesg. Dans ce cas, retirez le "`-t msdos`" ci-dessus.

- **Console série :** Utiliser une console série et capturer la sortie sur un autre ordinateur est souvent la meilleure solution d'obtenir des informations de diagnostic - particulièrement si l'ordinateur "panic" immédiatement après le démarrage. Aussi bien qu'un second ordinateur, vous aurez besoin d'un cable série (souvent un cable null-modem), et un émulateur de terminal pouvant capturer la sortie de l'écran dans un fichier. Des informations sur la configuratio d'une console série sont données ailleur dans la FAQ; dans le but de capturer le log de l'installation, les commandes suivantes sont souvent suffisantes.

  **i386**

  A l'invite de démarrage tapez :

```
boot> set tty com0
```

  Cela indiquera à OpenBSD d'utiliser le premier port série (souvent appelé COM1 ou COMA dans la documentation du PC) en tant que console série. La bande passante par défaut est 9600 bauds.

  **Sparc/Sparc64**

  Ces machines utiliseront automatiquement une console série si elles sont lancées sans clavier. Si vous avez un clavier et un écran attachés, vous pouvez toujours forcer le système à utiliser une console série avec l'invocation suivante à l'invite `ok`.

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

- **FTP :** Dans certaines circonstances, vous aurez la possibilité d'utiliser le client ftp(1) sur le disque d'amorce ou le CD-ROM pour envoyer le dmesg sur un serveur FTP local, où vous pourrez le récupérer ensuite.

## 4.16 - Mettre à jour/Réinstaller OpenBSD/i386 en utilisant `bsd.rd-a.out`

Il est normalement possible de réaliser des mises à jours et des installations en utilisant le noyau the bsd.rd. Cependant, avec OpenBSD 3.4, l'architecture i386 change de format exécutable de a.out à ELF, donc les anciens chargeur de démarrage (OpenBSD 3.3 et précédents) ne peuvent pas lancer le nouveau format du noyau `bsd.rd`.

Pour contourner ce problème, et autoriser les mises à jours en utilisant `bsd.rd`, une version a.out de `bsd.rd` à été réalisée et mise à disposition sur la distribution FTP. Ce fichier, `bsd.rd-a.out`, peut être chargé depuis OpenBSD 3.3 et précédents, mais est un noyau OpenBSD 3.5 authentique, incluant le nouveau chargeur de démarrage ELF, et peut donc être utilisé pour démarrer un système OpenBSD/i386 3.5 depuis une version précédente.

Téléchargez simplement `bsd.rd-a.out` et placez le à la racine du système de votre machine. Démarrez le, au lieu des noyaux normaux `bsd` ou `bsd.rd` comme montré ici (en spécifiant `bsd.rd-a.out` comme votre noyau de démarrage, bien sûr).

Si vous voulez installer *"-current"*, il vous est recommandé d'installer un système "3.5-release" minimal (`base35.tgz, etc35.tgz, bsd`), et ensuite réinstaller en utilisant le fichier *-snapshot* `bsd.rd`.

[Retour à l'Index principal] [Section 3 - Obtenir OpenBSD] [Section 5 - Construire le Système à partir des Sources]

---


www@openbsd.org

Originally [OpenBSD: faq4.html,v 1.169 ]
$Translation: faq4.html,v 1.18 2004/10/15 13:05:04 saad Exp $
$OpenBSD: faq4.html,v 1.10 2004/10/15 15:53:37 saad 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



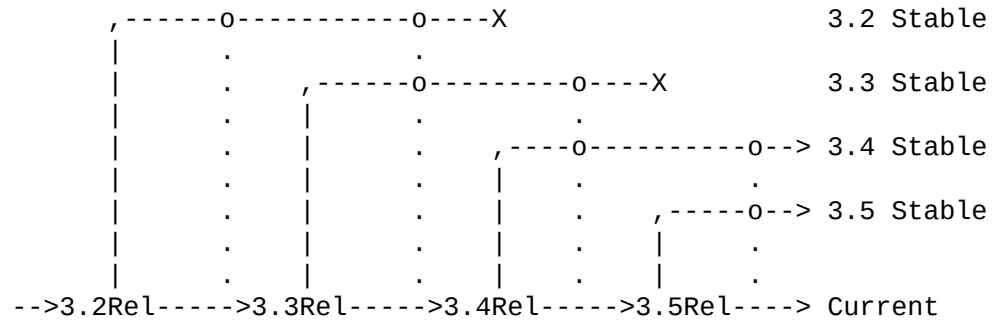
---

## 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 *3.2-stable* a été supprimée à la sortie de *3.4-release*, et que la branche *3.3-stable* a été supprimée à la sortie de *3.5-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 *3.5-release* vers *3.5-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 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 *3.5-current* (ou un snapshot), puis décider que c'est trop dangereux, et revenir vers *3.5-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 depend
# 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. Si config vous dit d'exécuter '`make clean`', alors faites le avant d'exécuter '`make depend`'.

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 3.5 (GENERIC) #34: Mon Mar 29 12:24:55 MST 2004
    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 3.5 (GENERIC) #34: Mon Mar 29 12:24:55 MST 2004
    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 - Le réseau]

---


www@openbsd.org

Originally [OpenBSD: faq5.html,v 1.97 ]
$Translation: faq5.html,v 1.39 2004/10/25 09:47:53 saad Exp $
$OpenBSD: faq5.html,v 1.32 2004/10/25 12:38:57 jufi Exp $

[Index de la FAQ] [Section 5 - Construire le Système à partir des Sources] [Section 7 - Contrôles du clavier et de l'affichage]

# 6 - Le réseau

---

### Table des matières



---

## 6.1 - Avant d'aller plus loin

Afin de permettre une meilleure compréhension de ce document, vous devriez lire et assimiler au moins partiellement la section Construire le Système à partir des Sources de la FAQ ainsi que le manuel ifconfig(8) et netstat(1) .

Si vous êtes administrateur de réseau et que vous mettez en place des protocoles de routage, si vous utilisez OpenBSD en tant que routeur, ou si vous souhaitez en savoir plus sur les réseaux IP, vous devriez lire " Understanding IP Addressing" (Comprendre l'adressage IP). Il s'agit d'un excellent document. Vous pouvez vous appuyer sur celui-ci afin de vous aider à travailler sur les réseaux IP, surtout si vous en avez un ou plusieurs sous votre responsabilité.

Si vous travaillez sur des applications telles que des serveurs web, des serveurs ftp et des serveurs de messagerie, vous pourriez bénéficier de la lecture des RFCs. A priori, vous ne pourrez pas toutes les lire. Choisissez les sujets qui vous intéressent ou les technologies que vous utilisez sur votre réseau. Lisez-les et voyez comment ces technologies fonctionnent. Les RFCs standardisent beaucoup (plusieurs milliers) de protocoles Internet et la façon dont ils sont censés fonctionner.

## 6.2 - Configuration initiale du réseau

### 6.2.1 - Identifier et configurer vos interfaces réseau

Pour commencer, vous devez d'abord identifier votre interface réseau. Sous OpenBSD, les interfaces sont nommées à partir du

type de la carte réseau, pas du type de la connexion. Vous pouvez voir l'initialisation de votre carte pendant la procédure de démarrage, ou après celle-ci en utilisant la commande dmesg(8) . Vous avez aussi la possibilité de voir votre carte réseau grâce à l'utilisation de la commande ifconfig(8). Voici par exemple la sortie de la commande dmesg pour une carte Intel Fast Ethernet qui utilise le nom de périphérique 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
```

Si vous ne connaissez pas le nom du périphérique associé à votre carte, regardez dans la liste des plates-formes actuellement supportées par votre architecture. Vous pourrez y trouver le nom des cartes les plus courantes et leur équivalent sous OpenBSD. Ajoutez au nom alphabétique du périphérique (par exemple fxp) le numéro assigné par le noyau et vous aurez le nom de votre interface (par exemple fxp0).

Vous pouvez connaître quelles interfaces réseaux ont été identifiées à l'aide de l'utilitaire ifconfig(8). La commande suivante montrera toutes les interfaces réseau d'un système. Cet exemple montre qu'il n'y a qu'une seule interface ethernet physique: 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
carp0: flags=0<> mtu 1500
carp1: flags=0<> mtu 1500
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
```

Comme vous pouvez le voir, à ce point ifconfig(8) nous fourni beaucoup plus d'informations que nécessaire. Mais nous pouvons tout de même voir notre interface. Dans l'exemple précédent, la carte est déja configurée. Ceci est évident de part la présence d'une configuration réseau IP pour fxp0, à savoir "inet 10.0.0.38 netmask 0xffffff00 broadcast 10.0.0.255". Les indicateurs **UP** et **RUNNING** sont également présents.

Finalement, vous noterez que d'autres interfaces sont activées par défaut. Il s'agit d'interfaces virtuelles servant différentes fonctions. Les manuels suivants les décrivent:

- lo - Interface "Loopback"
- pflog - Interface d'enregistrement du filtre de paquet (PF)
- sl - Interface SLIP (protocole Internet sur ligne série)
- ppp - Protocole point à point
- tun - Interface de tunnel
- enc - Interface d'encapsulation
- bridge - Interface de pont Ethernet
- vlan - Interface d'encapsulation EEE 802.1Q
- gre - Interface d'encapsulation GRE/MobileIP
- gif - Interface générique de tunnel IPv4/IPv6
- carp - Interface du protocole de redondance d'adresse (CARP)

Si votre interface n'est pas configurée, la première chose à faire est de créer le fichier */etc/hostname.xxx*, où xxx représente le nom de votre interface. A partir des informations des exemples précédents, le nom du fichier sera */etc/hostname.fxp0*. Le format de ce fichier est simple:

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

(Beaucoup plus d'informations sur la syntaxe de ce fichier sont disponibles dans le manuel hostname.if(5).)

Un fichier de configuration typique pour une interface IPv4 ressemblera à:

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

Vous pouvez également spécifier le type de media d'une connexion Ethernet pour, par exemple, forcer le mode 100baseTX full-duplex.

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

(Bien entendu, vous ne devriez jamais forcer le mode full duplex à moins que les deux extrémités de la connexion ne soient configurées ainsi ! Si ce n'est pour des besoins spécifiques, la configuration du type de media n'est pas nécessaire.)

Vous pouvez aussi vouloir utiliser des indicateurs spécifiques à une certaine interface. Le format du fichier hostname ne change que très peu.

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

La prochaine étape consiste à définir votre passerelle par défaut. Pour ce faire, renseignez simplement le fichier */etc/mygate* avec l'adresse IP de votre passerelle. Ceci permettra de configurer automatiquement votre route par défaut au démarrage. A présent, vous pouvez indiquer l'adresse de vos serveurs de noms et renseigner le fichier */etc/hosts* (voir la page de manuel hosts(5) ). Pour configurer vos serveurs de noms, vous devez créer un fichier nommé */etc/resolv.conf*. Vous pouvez en savoir plus sur le format de ce fichier dans la page de manuel resolv.conf(5). Voici un exemple d'utilisation standard. Dans cet exemple, vos serveurs de noms sont 125.2.3.4 et 125.2.3.5. Vous faites également partie du domaine "example.com".

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

A présent, vous pouvez soit redémarrer, soit lancer le script **/etc/netstart**. Vous pouvez l'exécuter simplement en tapant (en tant que 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
```

Notez que plusieurs erreurs sont apparues. En exécutant ce script, vous reconfigurez certaines choses qui le sont déjà. De fait, certaines routes sont déja présentes dans la table de routage du noyau. A présent, votre système devrait être en état de fonctionnement. Une nouvelle fois, vous pouvez vérifier que votre interface a été correctement configurée avec ifconfig(8). Vous pouvez également vérifier vos routes via netstat(1) ou route(8) . Si vous avez des problèmes de routage, vous pouvez utiliser l'option -n de la commande route(8) qui affichera l'adresse IP plutôt que d'effectuer une requête DNS et d'afficher le nom d'hôte. Voici un exemple qui vous permettra de voir vos tables de routage en utilisant ces deux programmes.

```
$ 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 - Mettre en place une passerelle OpenBSD

Voici les informations nécessaires à la mise en place d'une passerelle OpenBSD (appelé aussi routeur). Si vous devez installer OpenBSD pour en faire un routeur Internet, nous vous suggérons de lire les instructions sur la mise en place du filtre de paquets (plus loin) afin de bloquer le trafic non-autorisé. Avec le peu de disponibilité d'adresses IPv4 de la part des fournisseurs d'accès à Internet ainsi que des registres Internet régionaux, vous pourriez vouloir vous renseigner sur la translation d'adresses (NAT) afin

d'économiser votre adressage IP.

Le noyau GENERIC est déja configuré pour permettre le routage IP, mais celui-ci doit être explicitement activé. Vous pouvez l'activer avec l'utilitaire sysctl(8). Afin d'autoriser le routage de façon permanente, éditez le fichier /etc/sysctl.conf et ajoutez-y la ligne suivante.

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

Pour prendre en compte ce changement sans redémarrer, vous utiliserez directement l'utilitaire sysctl(8). Souvenez-vous que ce changement ne sera pas sauvegardé au redémarrage et que vous devrez être root pour utiliser cette commande.

```
# sysctl net.inet.ip.forwarding=1
net.inet.ip.forwarding: 0 -> 1
```

A présent, modifiez les routes sur les hôtes aux deux extrémités. Il existe beaucoup d'autres usages d'OpenBSD en tant que routeur avec l'aide de programmes comme routed(8), gated, mrtd, et zebra. Sous OpenBSD, zebra, gated et mrtd sont disponibles en tant que ports. OpenBSD supporte différentes interfaces T1, HSSI, ATM, FDDI, Ethernet, et série (PPP/SLIP).

## 6.2.3 - Configurer les alias sur une interface

OpenBSD possède un mécanisme simple pour la mise en place d'alias IP sur une interface. Pour ce faire, il suffit d'éditer le fichier */etc/hostname.<if>*. Ce fichier est lu au boot par le script /etc/rc(8) faisant partie de la séquence de démarrage rc. Admettons par exemple qu'un utilisateur utilise l'interface **dc0** et se trouve sur le réseau 192.168.0.0. Autres informations importantes:

- l'adresse IP pour dc0 est 192.168.0.2
- le masque de sous-réseau est 255.255.255.0

Notes sur les alias. Sous OpenBSD, vous utilisez le nom de l'interface uniquement. Il n'y a pas de différence entre le premier et le second alias. A la différence d'autres systèmes d'exploitation, OpenBSD ne s'y réfère pas en tant que dc0:0, dc0:1. Si vous vous référez à une adresse IP d'alias avec ifconfig ou si vous ajoutez un alias, soyez sûr d'utiliser la commande "`ifconfig int alias`" au lieu de "`ifconfig int`". Vous pouvez supprimer les alias en utilisant "`ifconfig int delete`".

En admettant que vous utilisiez plusieurs adresses avec alias sur le même sous-réseau IP, le masque correspondant à chaque alias devient 255.255.255.255. Il n'est pas nécessaire d'utiliser le masque correspondant à l'adresse IP primaire de l'interface. Dans cet exemple, */etc/hostname.dc0*, deux alias sont configurés pour le périphérique dc0, qui lui-même possède l'adresse 192.168.0.2 avec un masque de 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
```

Après avoir édité ce fichier, il suffit de redémarrer pour que les changements prennent effet. Mais vous pouvez aussi créer les alias à la main en utilisant l'utilitaire ifconfig(8). Pour créer le premier alias, lancez la commande suivante:

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

Pour voir les alias, utilisez la commande suivante:

```
$ 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 - Comment filtrer et utiliser un pare-feu sous OpenBSD ?

Sous OpenBSD, le filtrage du trafic TCP/IP ainsi que la translation d'adresses (NAT) est pris en charge par "Packet Filter" (filtre de paquet ; auquel nous nous referrerons à présent sous le nom de PF) . PF est aussi capable de normaliser et de traiter le trafic TCP/IP, d'offrir une gestion de la bande passante et une prioritarisation des paquets ainsi que d'être utilisé dans la création de puissants et flexibles pare-feu. Tout ceci est décrit dans le **Guide de l'Utilisateur PF**.

# 6.4 - DHCP

Le protocole d'attribution dynamique des adresses (DHCP) permet de configurer les interfaces réseaux automatiquement. OpenBSD peut être serveur DHCP (afin de configurer les autres machines), client DHCP (afin de recevoir sa configuration à partir d'une autre machine) et, dans certains cas, les deux.

## 6.4.1 Client DHCP

Pour utiliser le client DHCP dhclient(8) inclus avec OpenBSD, éditez le fichier `/etc/hostname.xl0` (sous-entendu que l'interface Ethernet soit xl0. La vôtre pouvant être aussi bien ep0 que fxp0 ou encore autre chose ! ) Tout ce que vous avez besoin d'écrire dans ce fichier est '`dhcp`':

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

Ceci aura pour effet de démarrer automatiquement le client DHCP au démarrage. OpenBSD se verra attribuer son adresse IP, sa passerelle par défaut et ses serveurs de noms (DNS) à partir du serveur DHCP.

Si vous souhaitez démarrer le client dhcp à partir de la ligne de commande, vérifiez bien que `/etc/dhclient.conf` existe, puis lancez la commande:

```
# dhclient fxp0
```

Où `fxp0` représente l'interface que vous voulez configurer par dhcp.

Peu importe la façon dont vous démarrez dhclient, vous pouvez éditer `/etc/dhclient.conf` afin de ne **pas** mettre à jour vos DNS à l'aide du serveur dhcp en décommentant les lignes 'request' (il y a des exemples de configurations, mais vous devez les décommenter afin de changer le comportement par défaut de dhclient.)

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

et **supprimez** domain-name-servers. Bien sûr, vous pouvez également supprimer hostname ainsi que d'autres options.

## 6.4.2 Serveur DHCP

Pour utiliser OpenBSD en tant que serveur DHCP dhcpd(8) , éditez le fichier `/etc/rc.conf.local` afin qu'il contienne la

ligne `dhcpd_flags="-q"`. Placez le nom des interfaces sur lesquelles vous souhaitez que le serveur écoute, dans le fichier `/etc/dhcpd.interfaces`.

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

Ensuite, éditez `/etc/dhcpd.conf`. Les options sont plutôt explicites.

```
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;
}
```

Ceci indique à vos clients DHCP que le domaine à ajouter aux requêtes DNS est example.com (ainsi, si un utilisateur lance la commande 'telnet joe', il sera renvoyé vers joe.example.com). Les clients auront comme serveurs DNS 192.168.1.3 et 192.168.1.5. Pour les hôtes présents sur le sous-réseau correspondant à l'interface du serveur OpenBSD, à savoir dans la plage 192.168.1.0/24, ils se verront attribuer une adresse IP entre 192.168.1.32 et 192.168.1.127. Leur passerelle par défaut sera 192.168.1.1.

Si vous souhaiter démarrez dhcpd(8) à partir de la ligne de commandes, une fois `/etc/dhcpd.conf` configuré, tapez:

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

La ligne incluant `touch` est nécessaire afin de créer un fichier `dhcpd.leases` vide avant que dhcpd(8) ne démarre. Les scripts de démarrage d'OpenBSD se chargeront de créer ce fichier au boot, mais si vous lancez dhcpd(8) manuellement, vous devez au préalable créer ce fichier. `fxp0` représente l'interface sur laquelle vous voulez répondre aux requêtes DHCP. L'option `-q` fait démarrer le serveur dhcpd(8) en mode silencieux, sinon celui-ci est très verbeux.

Si vous prévoyez de servir DHCP à des machines Windows, pour pourriez souhaiter donner à ces clients une adresse de serveur 'WINS'. Pour ce faire, ajoutez simplement la ligne suivante dans votre fichier `/etc/dhcpd.conf`:

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

(où `192.168.92.55` est l'adresse IP de votre serveur Windows ou Samba.) Reportez vous à dhcp-options(5) Si vos clients DHCP ont besoin de plus d'options, reportez vous au manuel dhcp-options(5).

# 6.5 - PPP

Le protocole point à point est généralement utilisé afin de créer une connexion modem vers votre FAI (fournisseur d'accès à Internet). Sous OpenBSD, deux solutions existent.

- pppd(8) - Qui est l'implémentation noyau du démon ppp.
- ppp(8) - Qui est l'implémentation ppp en espace utilisateur ("userland").

Le premier que nous allons aborder est le démon PPP en espace utilisateur. Pour commencer, vous aurez besoin de certaines informations concernant votre FAI. En voici une liste non-exhaustive.

- Le numéro de téléphone de votre FAI
- Votre serveur de noms
- Votre nom d'utilisateur et votre mot de passe
- Votre passerelle

Certaines de ces options ne sont pas obligatoires mais elles vous aideront à la mise en place de ppp. Le fichier de configuration du démon PPP est /etc/ppp/ppp.conf. Selon votre situation, les nombreux fichiers présents dans */etc/ppp* peuvent vous aider à la mise en place de votre configuration. Vous devriez jeter un oeil à ce répertoire.

Si vous n'utilisez pas le noyau GENERIC, soyez sûr d'inclure cette ligne dans le fichier de configuration de votre noyau:

```
pseudo-device   tun
```

## Configuration initiale - pour PPP(8)

La configuration initiale de PPP nécessite l'édition du fichier */etc/ppp/ppp.conf*. Ce fichier n'existe pas par défaut, mais vous pouvez vous inspirer du fichier */etc/ppp/ppp.conf.sample* afin de créer votre propre *ppp.conf*. Je commencerai par une configuration simple et généralement très employée. Voici rapidement un petit fichier *ppp.conf* définissant quelques valeurs par défaut:

```
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"
```

La section définie par le marqueur `default:` sera exécutée à chaque fois. Cette section recense des informations importantes. "set log" défini le niveau de verbosité des logs. Il peut être changé: référez-vous au manuel de ppp(8) pour plus d'informations concernant les niveaux d'enregistrement des différents évènements. Le périphérique utilisé est indiqué par "set device". Il s'agit du port sur lequel est présent notre modem. Dans cet exemple, le modem est branché sur le port com 2. Ainsi, le port com 1 sera indiqué par */dev/cua00*. La vitesse de la connexion est précisée par "set speed" et "set dial" renseigne nos paramètres de numérotation. Avec ceci, nous pouvons changer l'expiration ("timeout") de la connexion, etc. Ceci étant, cette ligne ne devrait pas tellement varier.

Maintenant, nous pouvons avancer et configurer les informations spécifiques à notre FAI. Pour ce faire, nous allons rajouter une autre section en-dessous de **default:**. Le marqueur définissant cette nouvelle section pourra être ce que vous désirez - le plus simple étant d'utiliser le nom de votre FAI. Ici, nous utiliserons **myisp:** pour indiquer le commencement de la section correspondant à notre FAI. Voici une configuration simple contenant le nécessaire afin de nous connecter:

```
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
```

Ici nous avons fourni les éléments essentiels concernant ce FAI. La première option, "set phone", donne le numéro de téléphone du FAI. Nos options d'ouverture de session sont renseignées par "set login". Notre "timeout" est égal à 5 ; ce qui signifie que notre tentative de connexion expirera après 5 secondes si aucune porteuse n'est trouvée. Dans le cas contraire, "login:" sera envoyé avec notre nom d'utilisateur et notre mot de passe.

Dans cet exemple, notre nom d'utilisateur est: ppp ; notre mot de passe est: ppp. Ces valeurs doivent être changées. La ligne "set timeout" permet de couper la connexion après 120 secondes de non-utilisation. L'option "set ifaddr" est un peu plus compliquée. En voici une explication plus complête.

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

La ligne précédente se définit avec le format suivant: "**set ifaddr [myaddr[/nn] [hisaddr[/nn] [netmask [triggeraddr]]]]**". La première IP désigne l'adresse IP que nous souhaitons nous voir attribuer. Si vous avez une adresse IP statique, vous devez l'indiquer ici. Dans notre exemple, nous utilisons la notation /0 qui dit qu'aucun "bit" de cette adresse ne doit forcément correspondre et que celle-ci peut être changée. La deuxième adresse IP est celle du FAI. La troisième option est notre masque de sous-réseau, ici défini à 255.255.255.0. Si triggeraddr est renseigné, il remplacera myaddr comme adresse IP utilisée pour la négociation IPCP initiale. Cependant, seule une adresse incluse dans la gamme d'adressage correspondant à myaddr sera acceptée. Ceci peut être utile dans la négociation avec certaines implémentations PPP qui n'attribuent pas d'adresse IP à moins que l'initiateur de la connexion ne demande explicitement l'adresse ``0.0.0.0''.

L'option suivante "add default HISADDR" attribue comme route par défaut l'adresse IP de votre FAI. Cette entrée permet d'ajuster automatiquement notre route par défaut en cas de changement d'adresse IP de celui-ci. "enable dns" est utilisé afin de récupérer la liste des serveurs DNS du FAI. N'utilisez PAS cette option si vous avez votre propre serveur DNS local car ppp empêchera les requêtes vers celui-ci en remplaçant les lignes "nameserver" de votre fichier */etc/resolv.conf*.

### Utiliser PPP(8)

Maintenant que *ppp.conf* est configuré, nous pouvons essayer d'initier une connexion vers notre FAI. Je détaillerai certaines des options les plus utilisées:

- `ppp -auto myisp` - Cette commande lancera ppp, configurera vos interfaces, vous connectera à votre FAI et se placera en arrière-plan.
- `ppp -ddial myisp` - Cette commande est similaire à -auto mais si votre connexion est rompue, ppp essayera de se reconnecter automatiquement.

Lancer */usr/sbin/ppp* sans options vous placera en mode interactif. A partir de là, vous pouvez intéragir directement avec le modem ; ceci peut être très utile pour corriger les erreurs dans votre fichier *ppp.conf*.

### Autres options de ppp(8)

Dans certaines situations, vous pourriez avoir besoin de lancer certaines commandes au lancement ou à la coupure de votre connexion. Si vous vous trouvez dans cette situation, il existe deux fichiers que vous pouvez créer: */etc/ppp/ppp.linkup* et */etc/ppp/ppp.linkdown*. Des exemples de configuration sont disponibles ici:

- ppp.linkup
- ppp.linkdown

Plus d'informations peuvent être trouvées dans la section "User PPP" du Manuel de Référence ("Handbook") de FreeBSD.

# 6.6 - Optimisation des paramètres réseau

## 6.6.1 - Comment configurer le noyau pour avoir un plus grand nombre d'essais et augmenter le délai d'expiration des sessions TCP ?

Généralement, vous utiliserez ceci en cas de problèmes de routage ou de connexion. Bien sûr, pour que cette configuration soit la

plus effective, les deux extrémités de la connexion doivent utiliser des valeurs similaires.

Pour optimiser ceci, utilisez `sysctl` et augmentez les valeurs de:

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

Avec sysctl -a, vous pouvez voir les valeurs courantes de ces paramètres (ainsi que beaucoup d'autres). Pour en changer un, tapez par exemple `sysctl net.inet.tcp.keepidle=28800`.

### 6.6.2 - Comment activer les émissions ("broadcasts") dirigées ?

Normalement, vous ne devriez pas utiliser ceci. Cette option permet à quelqu'un de diriger le trafic vers la ou les adresses "broadcast" des réseaux sur lesquels vous êtes connecté si vous utilisez OpenBSD en tant que routeur.

Dans certaines situations, sur des réseaux fermés, cette option peut-être utile, surtout lors de l'utilisation de vieilles implémentations du protocole NetBIOS. Il s'agit d'un autre sysctl. `sysctl net.inet.ip.directed-broadcast=1` active cette option. Lisez la section sur les "smurf attacks" si vous voulez savoir pourquoi cette option est désactivée par défaut.

### 6.6.3 - Je ne veux pas que le noyau alloue dynamiquement un port donné

Il existe également un sysctl pour ceci. D'après sysctl(8):

```
Définie la liste des ports TCP ne devant pas être alloués
dynamiquement par le noyau. Ceci peut être utile afin d'éviter
l'appropriation d'un port spécifique dont un autre programme a besoin
pour fonctionner. Les éléments listés peuvent être
séparés par une virgule et/ou un espace.

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

Il est aussi possible d'ajouter ou de retirer des ports de la liste courante.

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

## 6.7 - Utilisation simple de NFS

NFS ("Network File System") est utilisé afin de partager un système de fichiers à travers un réseau. Voici un certain nombre de manuels à lire avant la mise en place d'un serveur NFS:

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

Cette section détaillera les étapes nécessaires à la mise en oeuvre d'une configuration NFS simple. Cet exemple détaille un serveur et ses clients NFS sur un LAN. Il ne s'agit pas d'étudier la sécurisation de NFS. Nous assumerons que le filtre de paquets (PF) ou qu'un autre type de pare-feu est configuré ain d'empêcher les accès extérieurs. Si vous autorisez l'accès à votre serveur NFS depuis l'extérieur et que vous hébergez des données sensibles, nous vous conseillons fortement d'utiliser IPsec. Autrement certaines personnes pourraient voir ce qui transite dans votre trafic NFS. Quelqu'un pourrait également usurper l'adresse IP que vous

autorisez à se connecter à votre serveur NFS. Plusieurs types d'attaques peuvent en découler. Mais lorsqu'IPsec est correctement configuré, il offre une protection contre ces attaques.

Autre note concernant la sécurité. Ne vous contentez pas juste d'ajouter un système de fichiers dans */etc/exports* sans mettre en place une liste recensant les hôtes autorisés à se connecter. Sans une liste d'hôtes autorisés à monter un répertoire particulier, n'importe qui, capable de vous atteindre, sera en mesure de monter vos systèmes de fichiers exportés par NFS.

portmap(8) doit être lancé afin que NFS puisse fonctionner. A partir d'OpenBSD 3.2 Portmap(8) est désactivé par défaut, vous devez donc ajouter la ligne

```
portmap=YES
```

dans le fichier rc.conf.local(8) puis redémarrer.

Cette configurarion consiste en un serveur d'adresse IP **10.0.0.1**. Ce serveur n'offrira NFS qu'aux clients présents sur ce réseau. La première étape pour installer un serveur NFS est de renseigner votre fichier */etc/exports*. Ce fichier liste les systèmes de fichiers que vous souhaitez rendre accessibles par NFS et définit qui peut y accéder. Beaucoup d'options sont disponibles pour l'édition de ce fichier et vous devriez lire la page de manuel exports(5). Pour cet exemple, notre fichier */etc/exports* ressemblera à:

```
#
# 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
```

Ceci signifie que le système de fichiers local `/work` sera accessible par NFS. L'option `-alldirs` permet au client de monter n'importe quel répertoire sous le point de montage `/work`. L'option `-ro` exporte le système de fichiers en lecture seule. Les deux derniers arguments spécifient que seuls les clients présents dans le réseau 10.0.0.0 utilisant un masque de 255.255.255.0 seront autorisés à monter ce système de fichiers. Ceci est important pour certains serveurs accessibles sur différents réseaux.

Une fois que votre fichier */etc/exports* est configuré, vous pouvez mettre en place votre serveur NFS. Tout d'abord, vérifiez que les options NFSSERVER & NFSCLIENT sont bien présentes dans votre fichier de configuration noyau. (Le noyau GENERIC inclus ces options.) Ensuite, il vous faut inclure la ligne `nfs_server=YES` dans le fichier */etc/rc.conf.local*. Ceci aura pour effet de démarrer nfsd(8) et mountd(8) au redémarrage. A présent, vous pouvez lancer les démons manuellement. Ils doivent être lancés sous root et vous devez vérifier que portmap(8) est bien démarré sur votre système. Voici un exemple de lancement de nfsd(8) servant les requêtes TCP et UDP et utilisant 4 démons. Vous devriez utiliser un nombre approprié de démons NFS en fonction du nombre maximum de connexions concurrentes que vous souhaitez servir.

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

Vous devez non seulement démarrer le serveur nfsd(8) mais également mountd(8). Il s'agit du service qui se charge de passer les requêtes de montage à NFS. Pour démarrer mountd(8), soyez sûr qu'un fichier mountdtab vide existe puis lancez le démon:

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

Si vous faites des changements au fichier /etc/exports alors qu'NFS est déja en fonction, vous devez en informer mountd ! Lancez-lui simplement un signal HUP:

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

### Statistiques NFS

Maintenant, vous pouvez vérifier que tous ces démons sont lancés et enregistrés avec RPC. Pour ce faire, utilisez 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
```

Dans le cadre d'une utilisation normale, d'autres utilitaires peuvent vous permettre de voir ce qui se passe au niveau NFS. Un de ces utilitaires est showmount(8), qui permet de voir ce qui est monté et par qui. Il existe aussi nfsstat(8) qui affiche beacoup plus de statistiques. Pour utiliser showmount(8), essayez la commande `/usr/bin/showmount -a hôte`. Par exemple:

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

### Monter des systèmes de fichiers NFS

Les systèmes de fichiers NFS doivent être montés par mount(8), ou plus précisément, mount_nfs(8). Pour monter le système de fichiers */work* sur 10.0.0.1 vers le système de fichiers local */mnt*, il vous suffit de lancer la commande suivante (notez qu'il n'est pas nécessaire d'utiliser une adress IP ; mount résoudra les noms d'hôtes):

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

Pour que ce système de fichiers soit monté au démarrage, ajoutez la ligne suivante dans votre */etc/fstab*:

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

Il est important que vous utilisiez `0 0` à la fin de la ligne afin que votre machine ne lance pas un fsck sur ce système de fichiers NFS au boot !!! Les autres options standards, comme noexec, nodev, nosuid peuvent être employées lorsqu'elles sont applicables. Par exemple:

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

Ainsi, aucun périphérique ni programme setuid présent sur le serveur NFS ne peut compromettre la sécurité du client. Si vous ne montez pas de programmes que vous souhaitez utiliser sur le client NFS, ajoutez l'option noexec à cette ligne.

## 6.8 - Mise en place d'une connexion PPTP sous OpenBSD

**NOTE:** Cette section ne s'applique pas à **TOUS** les fournisseurs ADSL, mais beaucoup d'informations peuvent y être glanées. Cette configuration a été confirmée comme fonctionnelle pour Inode, un fournisseur d'accès autrichien et KPN, un fournisseur

d'accès hollandais.

Pour commencer, vous devez installer pptp. Le port est situé sous */usr/ports/net/pptp*. Lisez la FAQ 8, Qu'est que l'arborescences des ports ? pour de plus amples renseignements concernant l'arbre des ports sous OpenBSD.

Bien qu'il n'ait pas besoin de périphérique gre(4) , PPTP utilise l'encapsulation GRE. Afin de permettre au noyau de recevoir les paquets encapsulés par GRE, lancez la commande suivante:

```
# sysctl net.inet.gre.allow=1
```

Puis ajoutez la ligne suivante dans votre fichier */etc/sysctl.conf*:

```
net.inet.gre.allow=1
```

Ce qui permettra de sauvegarder ce changement au redémarrage.

Une fois le paquetage **pptp** installé et le fichier */etc/sysctl.conf* édité, vous devez renseigner un certain nombre de fichiers afin de configurer votre connexion. Ce paquetage utilise le ppp(8) présent dans OpenBSD, donc si vous êtes familier avec l'utilisation de celui-ci, la mise en place vous semblera similaire. Référez-vous également à la FAQ 6, Protocole Point à Point.

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

En ce qui concerne le fichier */etc/ppp/options*, la configuration suivante devrait répondre à la plupart des besoins:

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

`LOGINNAME` doit être remplacé par votre identifiant de connexion.

Insérez une ligne similaire à celle qui suit dans */etc/ppp/pap-secrets*:

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

Où LOGINNAME représente votre identifiant de connexion et PASSWORD votre mot de passe. 10.0.0.138 étant l'adresse IP de votre modem au cas où vous utiliseriez l'ADSL, etc. Faites attention que ce fichier ne soit accessible que par root et en lecture seule (mode 600).

## 6.8.1 - Assigner une adresse IP à votre interface

Dans l'exemple précédent, votre modem avait une adresse préconfigurée de 10.0.0.138. Nous devons maintenant assigner une adresse à NOTRE interface. Il est préférable d'utiliser une adresse proche de celle donnée à votre MODEM ou l'adresse IP statique qui vous est assignée. Plus d'information sur la mise en place des interfaces est disponible dans la FAQ 6, Configuration initiale du réseau.

Une fois votre interface configurée, vous devriez être en mesure de créer une connexion pptp à l'aide de la commande:

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

Puisque ppp(8) est utilisé, deux processus sont démarrés. Vous pouvez arrêter pptp en tuant ces deux processus:

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

Il est recommandé d'ouvrir `/var/log/messages` dans un autre terminal afin de diagnostiquer d'éventuels problèmes.

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

Nous vous conseillons également de placer les commandes de démarrage dans le fichier */etc/rc.local* afin que la connexion soit lancée à chaque fois que la machine démarre.

# 6.9 - Mise en place d'un pont ("bridge") avec OpenBSD

Un pont ("bridge ") est un lien entre deux (ou plusieurs) réseaux séparés. A l'inverse d'un routeur, les paquets sont transférés à travers le pont de manière "invisible" -- au niveau logique, les deux segments de réseau semblent n'en faire qu'un de chaque côté du pont. Le pont ne transfèrera que les paquets passant d'un segment à l'autre, ce qui entre autres, permet de réduire le trafic sur un réseau complexe tout en permettant à n'importe quel noeud d'accéder à un autre en cas de besoin.

Notez qu'à cause de sa nature "invisible", une interface faisant partie d'un pont peut, ou non ,posséder une adresse IP. Si c'est le cas, l'interface aura deux modes d'opération, l'un comme faisant parti du pont et l'autre se comportant comme une interface normale. Si aucune interface ne possède d'adresse IP, le pont fera transiter le trafic mais ne sera pas administrable par le réseau (ce qui peut être une fonctionnalité voulue).

## Exemple d'application d'un pont

Un des mes racks possède un certain nombre d'anciens systèmes, lesquels ne sont pas équipés de carte réseau 10BASE-TX. Bien qu'ils possèdent des ports AUI ou AAUI, ma réserve de transmetteurs est limitée à des câbles coaxiaux. Une des machines de cette rangée est un serveur d'accès terminal sous OpenBSD, toujours allumée et connectée au réseau à haut débit. L'ajout d'une seconde carte équipée d'un port coaxial permettra d'utiliser cette machine comme un pont vers le réseau coaxial.

Ce système a, pour le moment, deux cartes réseau, une Intel EtherExpress/100 (`fxp0` ) et une carte 3c590-Combo (`ep0` ) pour le port coaxial. `fxp0` fait le lien avec le reste du réseau et possède donc une adresse IP, `ep0`, ne faisant quant à elle que du "bridging", n'en possède pas. Les machines connectées sur le segment coaxial communiqueront comme les autres présentes sur le reste du réseau. A présent, voyons comment arriver à ce résultat.

Le fichier `hostname.fxp0` contient les informations concernant la carte `fxp0`. Cette machine est configurée pour DHCP, donc le fichier ressemble à celui-ci:

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

Ici, aucune surprise.

Comme vous pouvez le deviner, la configuration de la carte `ep0` est un peu différente:

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

Ici, nous demandons au système d'activer cette interface en utilisant ifconfig(8) et de la configurer pour du 10BASE-2 (coaxial). Aucune adresse ou information similaire n'est nécessaire pour cette interface. Les options possibles pour la carte `ep` sont disponibles en détail dans le manuel.

A présent, il nous faut paramétrer le pont. Un pont est initialisé par l'existence d'un fichier du type `bridgename.bridge0`. Dans ma situation, voici un exemple possible:

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

Cela indique de mettre en place un pont constitué de deux interfaces, fxp0 et ep0 et de l'activer. Est-ce que l'ordre dans lequel les cartes sont énoncées est important ? Non, souvenez-vous qu'un pont est très symétrique -- les paquets entrent et sortent dans les deux directions.

C'est tout ! Redémarrez et vous aurez un pont fonctionnel.

## Le filtrage sur un pont ("filtering bridge")

Bien qu'il existe certainement des usages pour de simples ponts du genre évoqué, il est probable que vous souhaitiez FAIRE quelquechose avec les paquets qui le traversent. Comme vous pouvez vous en douter, Packet Filter peut être utilisé pour restreindre le trafic traversant votre pont (pont filtrant).

Gardez à l'esprit que de part la nature d'un pont, les mêmes données traversent les deux interfaces, ce qui signifie que vous n'avez besoin de filtrer que sur l'une d'entre elles. Vos déclaration par défaut "Pass all" ressembleront à l'exemple suivant:

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

Maintenant, admettons que je souhaite filtrer le trafic dirigé vers les vieux systèmes évoqués précédemment, ne permettant qu'aux protocoles Web et SSH de les atteindre. Dans ce cas, nous allons autoriser tout le trafic entrant et sortant sur l'interface ep0, mais nous filtrerons sur l'interface fxp0 en utilisant "keep state" pour prendre en charge les données retournées:

```
# Autoriser le trafic entrant et sortant sur ep0
pass in quick on ep0 all
pass out quick on ep0 all

# Bloquer tout le trafic sur fxp0
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
```

Notez que cette règle bloquera tout, à l'exception des requêtes entrantes HTTP et SSH, vers la machine qui fait le pont ainsi qu'en

direction des autres noeuds "derrière" elle. Il est possible d'obtenir d'autres résultats en filtrant sur l'autre interface.

Pour surveiller et contrôler le pont que vous venez de créer, servez-vous de la commande brconfig(8) qui peut aussi être utilisée pour créer un pont après le démarrage.

### Astuces sur les ponts

- Il est HAUTEMENT recommandé de ne filtrer que sur une seule interface. Bien qu'il soit possible de filtrer sur les deux, vous devez vraiment en connaître toutes les implications avant de pouvoir le faire de la bonne manière.
- En utilisant l'option *blocknonip* de brconfig(8) ou dans bridgename.bridge0, vous pouvez empêcher le trafic non-IP (comme IPX ou NETBEUI) de passer oûtre votre filtre. Dans certaines situations, cela peut-être important et vous devez savoir qu'un pont fonctionne pour tous les types de trafic, pas seulement IP.
- Un pont requiert que les interfaces réseau soient en mode "Promiscuous" -- elles écoutent TOUT le trafic réseau, pas seulement celui leur étant dirigé. Ceci augmentera la charge du processeur et du bus. Certaines cartes ne fonctionnent pas correctement dans ce mode, le circuit TI ThunderLAN (tl(4)) en est un exemple.

## 6.10 - Comment démarrer en utilisant PXE ? (i386, amd64)

PXE ("Preboot Execution Environment", environnement d'exécution avant démarrage) permet de démarrer un ordinateur à partir du réseau plutôt que d'un disque dur, une disquette ou un CD-ROM. A l'origine, cette technologie a été développée par Intel mais est maintenant supportée par la plupart des contrôleurs réseau et des constructeurs. Sachez qu'il existe plusieurs protocoles de démarrage par le réseau, PXE étant relativement récent. Traditionnellement, un démarrage PXE est effectué en utilisant des ROMs présentes sur la carte réseau ou la carte mère elle-même, mais plusieurs disquettes permettant de démarrer en PXE sont disponibles sur différentes sources. Beaucoup de ROMs présentes sur des anciens contrôleurs supportent le démarrage en réseau mais sont imcompatibles avec PXE ; s'il est équipé de tels contrôlleurs, un système OpenBSD/i386 ou amd64 ne pourra pas être démarré à par le réseau.

### Comment fonctionne un démarrage PXE ?

Tout d'abord, il serait sage de savoir comment se déroule le processus de démarrage d'OpenBSD/i386 ? sur les platformes i386 et amd64. Au démarrage, chaque interface compatible PXE émet une requête DHCP en broadcast sur le réseau. Le serveur DHCP lui attribue alors une adresse IP en lui indiquant l'emplacement du fichier à exécuter sur le serveur tftp(1). Ce fichier se charge ensuite de gérer le reste du démarrage. Sous OpenBSD, pxeboot, remplace le fichier boot(8) standard. Il est alors capable de charger et d'exécuter un noyau (comme `bsd` ou `bsd.rd`) à partir du serveur tftp(1).

### Comment le mettre en place ?

Le point évident est que vous avez besoin d'une machine ou d'un contrôleur compatible avec un démarrage PXE. Certaines documentations précisent que toutes les cartes modernes sont compatibles PXE, mais c'est tout simplement faux -- de nombreux systèmes à bas prix n'incluent pas de ROMs PXE ou utilisent un ancien protocole de démarrage sur réseau. Vous avez également besoin d'un serveur DHCP configuré ainsi qu'un serveur TFTP.

En admettant qu'une machine sous OpenBSD serve les fichiers de démarrage (ceci n'est PAS obligatoire), le fichier dhcpd.conf de votre serveur DHCP devra contenir la ligne suivante:

```
filename "pxeboot";
```

afin de pouvoir offrir ce fichier de démarrage à une station de travail. Par exemple:

```
shared-network LOCAL-NET {
```

```
        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;
                filename "pxeboot";
                range 192.168.1.32 192.168.1.127;
                default-lease-time 86400;
                max-lease-time 90000;
        }
}
```

Vous devrez aussi activer le service tftpd(8) . Pour ce faire, il suffit de configurer inetd(8) . L'installation standard d'OpenBSD fournit une ligne d'exemple dans `inetd.conf` qui devrait vous satisfaire:

```
#tftp  dgram   udp  wait  root  /usr/libexec/tftpd   tftpd -s /tftpboot
```

Retirez simplement le caractère '#' et envoyez à inetd(8) un signal -HUP afin que celui-ci relise son fichier `/etc/inetd.conf`. Les fichiers accessibles par tftpd(8) sont contenus dans un répertoire particulier, ici `/tftpboot`, que nous utiliserons pour cet exemple. Bien évidemment, ce répertoire doit être créé et contenir les fichiers nécessaires. Pour un démarrage PXE, vous n'aurez typiquement besoin que de quelques fichiers:

- pxeboot, le chargeur de démarrage (offre la même fonction que boot pour un démarrage sur disque).
- bsd.rd, le noyau d'installation ou `bsd`, un noyau adapté.
- /etc/boot.conf, le fichier de configuration de boot.

Notez que `/etc/boot.conf` n'est nécessaire qu'au cas où vous souhaiteriez démarrer un noyau ne se nommant pas `bsd`, ou que les options par défaut de pxeboot ne vous conviennent pas (par exemple si vous utilisez une console série). Vous pouvez tester votre serveur tftpd(8) en utilisant un client tftp(1) afin de vérifier que vous pouvez bien récupérer les fichiers nécessaires.

Une fois vos serveurs DHCP et TFTP démarrés, vous êtes prêt pour un essai. Vous devez activer PXE sur votre système ou votre carte réseau ; consultez la documentation fournie avec votre matériel. Une fois que PXE est activé, vous devriez voir apparaître des lignes similaires à celles-ci:

```
Intel UNDI, PXE-2.0 (build 067)
Copyright (C) 1997,1998 Intel Corporation

For Realtek RTL 8139(X) PCI Fast Ethernet Controller v1.00 (990420)

DHCP MAC ADDR: 00 E0 C5 C8 CF E1
CLIENT IP: 192.168.1.76  MASK: 255.255.255.0  DHCP IP: 192.168.1.252
GATEWAY IP: 192.168.1.1
probing: pc0 com0 com1 apm pxe![2.1] mem[540k 28m a20=on]
disk: hd0*
net: mac 00:e0:c5:c8:cf:e1, ip 192.168.1.76, server 192.168.1.252
>> OpenBSD/i386 PXEBOOT 1.00
boot>
```

A présent, vous obtenez l'invite de commandes standard d'OpenBSD. Tapez simplement "`bsd.rd`" pour récupérer le fichier homonyme à partir du serveur TFTP.

```
>> OpenBSD/i386 PXEBOOT 1.00
boot> bsd.rd
booting tftp:bsd.rd: 4375152+733120 [58+122112+105468]=0x516d04
entry point at 0x100120
```

```
    Copyright (c) 1982, 1986, 1989, 1991, 1993
            The Regents of the University of California.  All rights reserved.

    Copyright (c) 1995-2004 OpenBSD.  All rights reserved.
http://www.OpenBSD.org

    OpenBSD 3.5 (RAMDISK_CD) #25: Thu Mar 19 23:16:56 EST 2004
      ...
```

Le noyau d'installation bsd.rd va maintenant se lancer.

### Est-il possible de démarrer d'autres noyaux que `bsd.rd` en utilisant PXE ?

Oui, bien qu'avec les outils actuellement présents dans OpenBSD 3.5, le démarrage PXE ne soit essentiellement prévu que pour installer le système d'exploitation.

## 6.11 - Protocole de redondance d'adresse commune (CARP)

### 6.11.1 - Qu'est-ce que CARP et comment fonctionne-t-il ?

CARP ("Common Address Redundancy Protocol") est un outil aidant à la redondance du système en ayant plusieurs ordinateurs créant une interface réseau unique entre eux, afin que si l'un d'eux ne fonctionne plus, un autre puisse répondre à sa place ; cet outil permet aussi de mettre en place un certain partage de charge entre les différents systèmes. CARP est une avancée par rapport au VRRP ("Virtual Router Redundancy Protocol" - protocole de redondance de routeur virtuel) standard. Il a été développé une fois que VRRP fut considéré comme non suffisamment libre à cause d'un brevet Cisco pouvant le couvrir. Pour plus d'information sur les origines de CARP et les problèmes légaux entourant VRRP, rendez-vous sur cette page.

Afin d'éviter tout problème légal, Ryan McBride (avec l'aide de Michael Shalayeff, Marco Pfatschbacher et Markus Friedl) a conçut CARP de manière à être fondamentalement différent. Si l'inclusion de la cryptographie reste le changement le plus visible, il n'en reste pas moins un seul parmi beaucoup d' autres.

Comment fonctionne-t-il ? CARP est un protocole multicast. Il regroupe plusieurs systèmes physiques en une ou plusieurs adresses virtuelles. L'un d'eux est le maître et répond aux paquets destinés au groupe, alors que les autres se comportent comme des "hot spares" (remplacement à chaud et automatique du maître). Peut importe les adresses IP et MAC de l'interface physique locale, les paquets envoyés vers l'adresse CARP ereviennent avec les informations CARP.

A intervalles configurés, le maître annonce sont état sur le port IP 112. Si le maître est déconnecté, les autres systèmes du groupe CARP commencent à annoncer leur présence. L'hôte qui parvient à s'annoncer le plus fréquemment devient le nouveau maître. Lorsque le système principal est reconnecté, il se transforme par défaut en hôte de secours, bien qu'il soit possible de configurer un hôte spécifique en tant que maître par défaut lorsque cela est faisable (ex. un hôte est un rapide Sun Fire V120 et les autres sont des SPARCstation IPCs, comparativement plus lentes).

Bien que les équipements à haute redondance et tolérance de pannes minimisent le besoin de CARP, ils ne le suppriment pas. Il n'existe pas de matériel à tolérance de pannes capable de gérer le fait que quelqu'un débranche le cordon d'alimentation ou que l'administrateur système tape `reboot` dans la mauvaise fenêtre. L'utilisation de CARP aide à rendre transparent une mise à jour ou un redémarrage du point de vue des utilisateurs, ainsi que le test d'un programme ou une mise à jour matérielle: si cela ne fonctionne pas, vous pouvez basculer sur le reste du groupe jusqu'à ce que le problème soit résolu.

Il existe cependant certaines situations où CARP ne pourra pas vous venir en aide. La conception de CARP est telle que les membres d'un groupe doivent appartenir au même réseau physique et que chaque interface doive avoir une véritable adresse IP fixe en plus de l'adresse IP CARP statique. De même, les services nécessitant une connexion constante au serveur (tels que SSH ou IRC) ne seront pas transférés de manière transparente aux autres systèmes, bien que dans ce cas, CARP puisse aider à minimiser le

temps d'arrêt. CARP, lui-même, ne synchronise pas les données entre applications, ceci doit être fait au travers de "procédés alternatifs" tels que pfsync(4) (pour du filtrage redondant), rsync, pour dupliquer manuellement les données entre les machines, ou tout ce qui semble approprié à votre application.

## 6.11.2 - Configuration

Le contrôle de CARP s'effectue en deux endroits: sysctl(8) et ifconfig(8). Regardons tout d'abord les sysctls.

La première variable sysctl, `net.inet.carp.allow`, définie si l'hôte gère ou non les paquets CARP. Bien évidemment, ceci est nécessaire à l'utilisation de CARP. Cette variable sysctl est activée par défaut.

La seconde, `net.inet.carp.arpbalance`, est utilisée pour la balance de charge ("load balancing"). Si cette fonctionnalité est activée, CARP effectue une empreinte (hashage) de l'IP originatrice de la requête. Cette empreinte est ensuite utilisée pour sélectionner à partir du groupe l'hôte virtuel qui prendra en charge cette requête. Cette fonctionnalité est désactivée par défaut.

La troisième, `net.inet.carp.log`, enregistre les erreurs CARP. Désactivée par défaut.

La quatrième, `net.inet.carp.preempt` active la sélection naturelle parmi les hôtes CARP. Le plus à même d'effectuer le travail (à savoir, celui qui est capable de s'annoncer le plus fréquemment) deviendra le maître. Celle-ci est désactivée par défaut, ce qui signifie qu'un système qui n'est pas maître ne tentera pas de (re)gagner ce status.

Toutes ces variables sysctl sont documentées dans sysctl(3).

Pour ce qui concerne le reste de la configuration de CARP, nous dépendrons d' ifconfig(8). Deux des quatre commandes spécifiques à CARP, `advbase` et `advskew`, définissent l'intervalle entre les annonces CARP. La formule (en secondes) est `advskew` divisée par 255 puis ajoutée à `advbase`. `advbase` peut être utilisée pour diminuer le trafic réseau ou autoriser une plus grande latence avant qu'un hôte de sauvegarde ne prenne le relais ; `advskew` vous permet de contrôler quel hôte sera le maître sans trop de délais de basculement (si cela est nécessaire).

Ensuite, `pass` crée un mot de passe et `vhid` définit le numéro d'identification d'hôte virtuel du groupe CARP. Vous devez assigner un numéro unique pour chaque groupe CARP, même s'ils partagent (pour des raisons de balance de charge) la même adresse IP. CARP est limité à 255 groupes.

Voyons l'emploi de ces différents paramètres dans une configuration simple. Admettons que vous déployiez deux serveurs Web identiques, *rachael* (192.168.0.5) et *pris* (192.168.0.6), afin de remplacer un ancien système à l'adresse 192.168.0.7. Les commandes:

```
rachael# ifconfig carp0 create
rachael# ifconfig carp0 vhid 1 pass tyrell 192.168.0.7
```

créent l'interface carp0 et lui donnent un vhid de 1, le mot de passe *tyrell*, et l'adresse IP 192.168.0.7. Afin de rendre ces changements permanents après un redémarrage, vous pouvez créer un fichier `/etc/hostname.carp0` ressemblant à:

```
inet 192.168.0.7 255.255.255.0 192.168.0.255 vhid 1 pass tyrell
```

Faites la même chose sur *pris*. Le système démarrant en premier son interface CARP deviendra le maître.

Notez que sur une machine possédant plusieurs interfaces, l'interface CARP est sur le même sous-réseau que l'interface physique.

Mais admettons que vous ne déploiiez pas cette solution à partir de zéro. *Rachael* était déja en place à l'adresse 192.168.0.7.

Comment allez-vous gérer cette situation ? Heureusement, CARP ne pose pas de problème si un système possède la même adresse IP sur son interface physique et dans un groupe CARP, donc il n'est pas nécessaire de changer les commandes précédentes. Cependant, il est plus propre d'avoir une adresse IP différente pour chaque systèm, cela rend les accès et la surveillance individuels bien plus simples.

Ajoutons un nouveau niveau de complexité ; nous souhaitons que *rachael* reste maître lorsque cela est possible. Plusieurs raisons peuvent nous pousser à cela: différences matérielles, simple préjudice, "si ce système n'est pas maître, c'est qu'il y a un problème", ou simplement connaître le maître par défaut sans avoir à recourir à des scripts envoyant par courriel l'analyse de la sortie d'ifconfig.

Sur *rachael*, nous utiliserons la variable sysctl créée précédemment puis nous éditerons `/etc/sysctl.conf` afin de rendre ce paramètre permanent.

```
rachael# sysctl net.inet.carp.preempt=1
```

Configurons également *pris*:

```
pris# ifconfig carp0 advskew 100
```

Ceci décale les annonces de *pris*, ce qui signifie que *rachael* sera maître s'il est fonctionnel.

Notez que si vous utilisez PF sur une machine CARP, vous devez utiliser "proto carp" sur toutes les interfaces concernées, avec une ligne similaire à:

```
pass on fxp0 proto carp keep state
```

### 6.11.3 - Balance de charge ("load balancing")

Effectuons une avance rapide de quelques mois. Notre entreprise de l'exemple précédent a grossi au point qu'un seul serveur Web interne arrive tout juste à gérer la charge. Que faire ? CARP à la rescousse. Il est temps d'essayer la balance de charge. Créons une nouvelle interface et un nouveau groupe CARP sur *rachael*:

```
rachael# ifconfig carp1 create
rachael# ifconfig carp1 vhid 2 advskew 100 pass bryant
192.168.0.7
```

Sur *pris*, nous allons également créer un nouveau groupe et une nouvelle interface, puis activer la variable sysctl "preempt":

```
pris# ifconfig create carp1
pris# ifconfig carp1 vhid 2 pass bryant 192.168.0.7
pris# sysctl net.inet.carp.preempt=1
```

A présent nous avons deux groupes CARP avec la même adresse IP. Chaque groupe est dirigé vers un hôte différent, ce qui signifie que *rachael* restera maître du premier groupe, mais que *pris* lui succédera comme maître dans le second.

Tout ce qu'il nous reste à faire désormais est d'activer la variable contrôlant la balance de charge sur les deux machines, comme nous l'avons vu précédemment:

```
# sysctl net.inet.carp.arpbalance=1
```

Bien que ces exemples ne soient que pour un cluster de deux machines, le même principe reste valable avec plus de systèmes. Notez en revanche qu'il n'est pas assuré que vous parveniez à une distribution de 50/50 sur les deux machines: CARP utilise une empreinte de l'IP originatrice afin de savoir quel système répondra à la requête, indépendamment de la charge.

### 6.11.4 - Plus d'information sur CARP

- carp(4)
- ifconfig(8)
- sysctl(8)
- sysctl(3)
- "Firewall Failover with pfsync and CARP" (Redondance de pare-feu avec pfsync et CARP) par Ryan McBride

[Index de la FAQ] [Section 5 - Construire le Système à partir des Sources] [Section 7 - Contrôles du clavier et de l'affichage]

---


www@openbsd.org

Originally [OpenBSD: faq6.html,v 1.200 ]
$Translation: faq6.html,v 1.32 2004/10/23 09:08:53 ajacoutot Exp $
$OpenBSD: faq6.html,v 1.18 2004/10/23 10:18:07 jufi Exp $

[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, amd64, cats, hppa, i386, macppc, sparc, sparc64 and vax.

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, amd64 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 *(amd64, 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? *(amd64, i386, some Alpha)*

On amd64, 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? *(amd64, i386)*

amd64 and 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 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.

## amd64 and 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.

### amd64 and 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 -- when using this feature, some BIOSs may prevent the bootloader from seeing the serial port, and thus, the kernel will not be told to use it. Some BIOSs have an option to "Continue Console Redirection after POST" (Power On Self Test), this should be set to "OFF", so the boot loader and the kernel can handle their own console. Unfortunately, this feature is not universal.
- 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 amd64 and 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 amd64 and 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`".

# 7.9 - EVERYTHING I TYPE AT THE LOGIN PROMPT IS IN CAPS!

This is a feature, not a bug, actually.

Virtually all Unix commands and user names are entered using all lowercase. However, some very old terminals were only capable of uppercase characters, making them difficult, if not impossible, to use with Unix. As a workaround, if you entered your user name in all uppercase, getty(8) would assume your terminal was "lowercase challenged", and simply interpret everything you type as lowercase, while echoing it as uppercase. If you have a mixed-case or uppercase password, this will make login impossible.

Hitting CTRL-D at the login prompt will cause getty(8) to terminate, and init(8) will relaunch a new one, which will accept uppercase and lowercase properly.

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

---


www@openbsd.org

$OpenBSD: faq7.html,v 1.65 2004/10/20 23:04:26 nick Exp $

[FAQ Index] [To Section 7 - Keyboard and Display controls] [To Section 9 - Migrating to OpenBSD]

# 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 more information about configuring and using X on your platform, see the `/usr/X11R6/README` file on your installed system.

# 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/3.5/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/3.5/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-0.54-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-0.54-no_x11
  [...snip...]
===>  Building package for mtr-0.54-no_x11
Creating package /usr/ports/packages/i386/All/mtr-0.54-no_x11.tgz
Using SrcDir value of /usr/ports/net/mtr/w-mtr-0.54-no_x11/fake-i386-no_x11/usr/
local
Creating gzip'd tar ball in '/usr/ports/packages/i386/All/mtr-0.54-no_x11.tgz'
===>  Installing mtr-0.54-no_x11 from /usr/ports/packages/i386/All/mtr-0.54-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-4.0.2        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-20030709      An enhanced version of ircII, the Internet Relay Chat client
```

```
ispell-3.2.06       An interactive spelling checker.
tin-1.6.2           TIN newsreader (termcap based)
procmail-3.22       A local mail delivery agent.
strobe-1.06         Fast scatter/gather TCP port scanner
lsof-4.69           Lists information about open files.
ntp-4.1.74          Network Time Protocol Implementation.
ncftp-3.1.7         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/3.5/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-4.0.2.tgz
Adding screen-4.0.2.tgz
Install script: /var/tmp/pkginfo.N2mDFKTA6pq/+INSTALL screen-4.0.2 PRE-INSTALL
extracting /usr/local/bin/screen
extracting /usr/local/info/screen.info
exec install-info --info-dir=/usr/local/info /usr/local/info/screen.info
extracting /usr/local/man/man1/screen.1
extracting /usr/local/share/examples/screen/screenrc-sample
extracting /usr/local/share/screen/utf8encodings/01
extracting /usr/local/share/screen/utf8encodings/02
extracting /usr/local/share/screen/utf8encodings/03
extracting /usr/local/share/screen/utf8encodings/04
extracting /usr/local/share/screen/utf8encodings/18
extracting /usr/local/share/screen/utf8encodings/19
extracting /usr/local/share/screen/utf8encodings/a1
extracting /usr/local/share/screen/utf8encodings/bf
extracting /usr/local/share/screen/utf8encodings/c2
extracting /usr/local/share/screen/utf8encodings/c3
extracting /usr/local/share/screen/utf8encodings/c4
extracting /usr/local/share/screen/utf8encodings/c6
extracting /usr/local/share/screen/utf8encodings/c7
extracting /usr/local/share/screen/utf8encodings/c8
extracting /usr/local/share/screen/utf8encodings/cc
```

```
extracting /usr/local/share/screen/utf8encodings/cd
extracting /usr/local/share/screen/utf8encodings/d6
Install script: /var/tmp/pkginfo.N2mDFKTA6pq/+INSTALLscreen-4.0.2 POST-INSTALL

+---------------
| The screen-4.0.2 configuration file, /etc/screenrc,
| has been installed.  Please view this file and change
| the configuration to meet your needs.
+---------------

/usr: 911029 bytes
```

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/3.5/packages/i386/screen-4.0.2.tgz
Adding ftp://ftp3.usa.openbsd.org/pub/OpenBSD/snapshots/packages/i386/screen-4.0.2.tgz

+---------------
| The screen-4.0.2 configuration file, /etc/screenrc,
| has been installed.  Please view this file and change
| the configuration to meet your needs.
+---------------
```

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
mpg123-0.59rp2     mpeg audio 1/2 layer 1, 2 and 3 player
nmap-3.50          port scanning large networks
ircII-20030709     enhanced version of ircII (internet relay chat)
screen-4.0.2       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-4.0.2

+---------------
| To completely deinstall the screen-4.0.2 package you need to perform
| this step as root:
|
|           rm -f /etc/screenrc
|
| Do not do this if you plan on re-installing screen-4.0.2
| 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 3.5 (GENERIC) #34: Mon Mar 29 12:24:55 MST 2004
    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.
#
```

## 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:

```
# skeyinit -E
```

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 passphrase*, and is **NOT** your system password. Your secret passphrase must be at least 10 characters. We suggest using a memorable phrase containing several words as the secret passphrase. Here is an example user being added.

```
$ skeyinit
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.
Password:
[Adding ericj with md5]
Enter new secret passphrase:
Again secret passphrase:

ID ericj skey is otp-md5 100 oshi45820
Next login password: HAUL BUS JAKE DING HOT HOG
```

One line of particular importance in here is *ID ericj skey is otp-md5 100 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).
- *100* - This is your sequence number. This is a number from 100 down to 1. Once it reaches one, another secret passphrase must be created by running skeyinit(1).
- *oshi45820* - This is the key.

But of more immediate importance is your one-time password. Your one-time password consists of 6 small words, combined together this is your one-time password, spaces and all. The one-time password printed by skeyinit cannot be used to login (there is a usage for this first one-time password, see skeyinit(1)). To be able to log in, a one-time password corrsponding to the challenge printed by the login process has to be computed using skey(1). The next section will show how to do that.

## Actually using S/Key to login.

By now your skey has been initialized. You're ready to login. Here is an example session using S/Key to login. To perform an S/Key login, you append **:skey** to your login name.

```
$ 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 3.5 (GENERIC) #34: Mon Mar 29 12:24:55 MST 2004
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 one-time password? Well, to compute the one-time password, 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?

When you are logging in, the login process prints a line containing the needed information, which you can use to generate a one-time password on the spot using another trusted computer accesses by a secure channel, by copy-pasting the line into a command shell:

```
otp-md5 96 oshi45820
```

After typing your passphrase, your one-time password will be printed, which you can then copy-paste to the S/Key Password prompt to log in. Not only is *otp-md5* a description of the hash used, it is also an alternate name for the skey(1) command.

If you already are logged in and want to generate a one-time password for the next login, use skeyinfo(1), it will tell you what to use for the next login. For example here, I need to generate another one-time 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
```

So, the simplest way to generate the next S/Key password is just:

```
$ `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 or a trusted local computer 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 3.5 (GENERIC) #34: Mon Mar 29 12:24:55 MST 2004

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)

SMP is not supported on OpenBSD 3.5-release or 3.5-stable, however SMP support has been added to OpenBSD/i386-current and OpenBSD/amd64-current.

A separate SMP kernel, "`bsd.mp`", is provided with the install file sets, which can be selected at install time. It is suggested that you test booting this kernel before renaming it to "`bsd`" to make it your default kernel.

It is hoped that other SMP-capable platforms will be supported in the future. On most other 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.

## 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)
- Mozilla and Firefox Feature-filled browsers. Mozilla includes a many non-browser features (mail client, IRC client, etc.), Firefox is just a browser, based on Mozilla. Works on alpha, amd64, i386, macppc, sparc, and sparc64 platforms.

**Console (Text mode) Browsers**

- w3m Has table and frame support (also has a graphical mode).
- 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.

## 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 3.5 (GENERIC) #34: Mon Mar 29 12:24:55 MST 2004
    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 use config(8) to change your 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 to OpenBSD]

---


www@openbsd.org

$OpenBSD: faq8.html,v 1.152 2004/10/20 23:04:26 nick 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 d'autres systèmes d'exploitation Unix-like

Bien qu'OpenBSD soit un système d'exploitation Unix-like très traditionnel et très familier pour les personnes ayant utilisé d'autres systèmes d'exploitation Unix-like, il existe d'importantes différences. Les nouveaux utilisateurs d'OpenBSD doivent se baser sur leur propre expérience : si votre connaissance d'Unix se limite à une certaine expérience avec une variante de Linux, vous pourrez trouver OpenBSD "étrange". Soyez rassuré, Linux parait aussi "étrange" pour quelqu'un qui a commencé son expérience d'Unix avec OpenBSD. Vous devez faire la distinction entre le "standard" et votre expérience.

Si vous avez appris Unix à l'aide d'un des bons ouvrages sur Unix en général, et si vous avez saisi la "philosophie Unix" puis étendu votre connaissance à un plate-forme particulière, vous trouverez qu'OpenBSD est un Unix "vrai" et très familier. Si vous avez appris Unix en utilisant un procédé de type "saisis ceci au clavier pour faire cela" ou un livre tel que "Apprendre PinkBeenie v8.3 en 31.4 Heures", puis vous vous êtes dit que vous "connaissez Unix", vous trouverez certainement OpenBSD très différent.

Une différence importante entre OpenBSD et plusieurs autres systèmes d'exploitation est la documentation. Les développeurs OpenBSD sont très fiers des pages de manuel du système. Les pages de manuel sont *les*sources de référence de la documentation OpenBSD -- ce qui n'est pas le cas de cette FAQ ou des pages maintenues indépendamment du projet ou des "HOWTO"s etc. Lorsqu'un développeur fait une modification au niveau système, on attend de sa part qu'il mette à jour les pages du manuel en même temps que la modification du code système, et pas "après" ou "losqu'il aura le temps de le faire" ou lorsque "quelqu'un se plain". Une page de manuel existe pour virtuellement chaque programme, utilitaire, pilote, fichier de configuration et ainsi de suite dans le système de base. On attend de la part de l'utilisateur qu'il prenne le temps d'effectuer des recherches dans les pages du manuel avant de demander de l'aide sur les listes de diffusion.

Voici quelques unes des différences les plus communes entre OpenBSD et d'autres variantes Unix.

- OpenBSD est un Unix du style "BSD" assez pur, qui suit la conception 4.4BSD de manière rapprochée. Linux et SCO Unix sont des systèmes du système "System V". Certains systèmes d'exploitation Unix-like (y compris quelques distributions Linux) font un mélange de plusieurs caractéristiques propres à Sys7 et à BSD. Cela prête à confusion particulièrement dans les scripts de démarrage, pour lesquels OpenBSD utilise le système traditionnel rc(8) 4.4BSD.
- OpenBSD est un *système* complet, destiné à être maintenu de manière homogène. Ce n'est pas un "noyau plus quelques utilitaires" qui peuvent être mis à jour séparemment. Si vous n'arrivez pas à maintenir votre système (noyau, utilitaires et applications) homogène, de mauvaises choses peuvent arriver.
- Etant donné que plusieurs applications ne sont pas développées pour être compilées directement et s'exécuter dans un environnement OpenBSD, OpenBSD a une arborescence de ports, un système qui permet aux utilisateurs de facilement acquérir du code, le modifier pour qu'il soit compilable sur OpenBSD, installer des dépendances, le compiler, et l'installer et le supprimer de manière standardisée et maintenable. Des packages pré-compilés sont créés et distribués par l'équipe de portage OpenBSD. Les utilisateurs sont encouragés à utiliser des packages au lieu de compiler les leurs.
- OpenBSD utilise CVS pour les modifications des sources. OpenBSD a été un pionnier d'anonymous CVS, qui permet à n'importe qui d'extraire l'arborescence complète des sources pour n'importe quelle version d'OpenBSD (à partir de 2.0 jusqu'à la version actuelle, y compris n'importe quelle révision de n'importe quel fichier située entre ces deux versions) à n'importe quel moment, et vous pouvez accéder aux modifications les plus récentes à peine quelques heures après qu'elles aient été effectué. Il y a aussi une There is also a very convenient and easy to use interface web pour CVS très utile et facile d'utilisation.
- OpenBSD produit une version officielle disponible sur CD et par FTP tous les six mois selon un agenda prédéfini. Des snapshots pour toutes les plates-formes supportées sont créés de manière semi-régulière à partir du code de développement le plus récent. Un des buts du projet OpenBSD est de garder l'arborescence des sources compilable à tout moment et de faire en sorte que le système issu de la compilation de cette arborescence soit utilisable. L'arborescence peut connaître des problèmes de compilation mais c'est à titre exceptionnel et ces problèmes sont résolus rapidement.
- OpenBSD contient une cryptographie forte, qui ne peut être incluse dans des systèmes d'exploitation provenant de certains pays.
- OpenBSD fait l'objet d'un audit sécurité lourd et continu pour s'assurer de la qualité (et donc, de la sécurité) du code.
- Le noyau d'OpenBSD est `/bsd`.
- Les noms des disques durs sont du type `/dev/wd` (IDE) et `/dev/sd` (SCSI ou équipements à émulation SCSI)
- /sbin/ifconfig sans arguments sous Linux fournit l'état de toutes les interfaces, mais sous OpenBSD (et plusieurs autres systèmes d'exploitation), vous aurez besoin du drapeau `-a`.
- /sbin/route sans arguments sous Linux fournit l'état de toutes les routes actives, mais sous OpenBSD (et plusieurs autres systèmes d'exploitation), vous aurez besoin du paramètre "*show*" ou utilisez la commande "`netstat -r`".
- OpenBSD NE supporte pas les Systèmes de Fichiers à Journaux tels que ReiserFS, JFS d'IBM ou XFS de SGI. Au lieu de cela, nous utilisons la fonctionnalité Soft Updates du très robuste Unix Fast File System (FFS) pour atteindre les objectifs de performance et de stabilité.
- OpenBSD inclut Packet Filter (PF), et non pas ipfw, ipchains, netfilter, iptables, ou ipf. Ce qui veut dire que la Traduction d'Adresses IP (connu sous le nom d'IP- Masquerading sous Linux), la gestion de la bande passante et le filtrage est effectué via pfctl(8), pf(4), et pf.conf(5). Consultez le Guide de l'Utilisateur PF pour des informations détaillées sur la configuration.
- Les adresses des interfaces sont stockées dans les fichiers /etc/hostname.<interfacename> (par exemple, `/etc/hostname.dc0` pour une interface réseau utilisant le pilote dc(4)). Ces fichiers peuvent contenir un nom d'hôte (résolu à partir de /etc/hosts) au lieu d'une adresse IP.
- Le nom de la machine est dans le fichier /etc/myname
- La passerelle par défaut est stockée dans le fichier /etc/mygate
- Le shell utilisateur par défaut d'OpenBSD est /bin/sh, qui est pdksh, le shell Korn du domaine public en mode Bourne Shell. Les autres shells inclus sont csh (le shell par défaut de *root*) et ksh. Les shells tels que *bash* et *tcsh* peuvent être ajoutés à partir des packages ou installés à partir des ports. Les utilisateurs familiers avec *bash* sont encouragés à essayer ksh(1) avant d'installer bash sur leur système -- il a la plupart des fonctionnalités que les utilisateurs utilisent dans *bash*.
- La gestion des mots de passe est différente de celle de la plupart des autres systèmes d'exploitation Unix-like. Les mots de passe sont stockés dans le fichier master.passwd(5) qui ne peut être lu que par *root*. Ce fichier ne doit être modifié que par le programme vipw.
- Les périphériques sont appelés selon le pilote et pas selon le type. Par exemple, il n'existe pas de périphériques eth*. Pour une carte Ethernet NE2000, ça serait ne0 et xl0 pour des périphériques Ethernet 3Com Etherlink XL ou Fast Etherlink XL etc. Tous ces pilotes ont des pages de manuel dans la section 4. Ainsi, pour en savoir plus sur les messages

que votre pilote 3c905 affiche, vous pouvez faire "`man 4 xl`".

- OpenBSD/i386 utilise un système de partitionnement de disque à "deux couches", où la première couche est la partition fdisk, visible depuis le BIOS, et avec laquelle la plupart des utilisateurs d'ordinateurs compatibles IBM sont familiers. La seconde couche est le disklabel, un système de partionnement BSD traditionnel. OpenBSD supporte un maximum de 15 partitions disklabel sur un disque, toutes dans la même partition fdisk. Ceci permet à OpenBSD/i386 de coexister avec d'autres systèmes d'exploitation, y compris d'autres systèmes d'exploitation Unix-like. OpenBSD doit faire partie des 4 partitions "primaires".
- Certains systèmes d'exploitation vous encouragent à adapter votre noyau à votre machine. Les utilisateurs d'OpenBSD sont encouragés à utiliser le noyau standard GENERIC fourni et testé par les développeurs. Les utilisateurs souhaitant "adapter" et "optimiser" causent souvent plus de problèmes qu'ils n'en résolvent et aucun ne leur sera fourni par les développeurs.
- L'équipe de développement OpenBSD travaille dur pour maintenir la politique de copyright et la securité du projet. Pour cette raison, certaines nouvelles versions de logiciels qui ne remplissent pas les objectifs de licence ou de sécurité ne sont pas intégrées à OpenBSD, et pourraient ne jamais l'être. La sécurité et les licences libres ne seront jamais négligées au profit d'un numéro de version plus grand.

## 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 système de type "Sixth Edition") 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 "Sixth Edition".

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 "Sixth Edition" (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 "Sixth Edition". 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 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 1.66 ]

$Translation: faq9.html,v 1.28 2004/10/25 09:47:39 saad Exp $
$OpenBSD: faq9.html,v 1.21 2004/10/25 12:38:57 jufi Exp $

[Index de la FAQ] [Section 9 - Migrer vers OpenBSD] [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 système :

- 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`" à "`/etc/shells`". A partir de là, lorsque vous affectez "`/usr/bin/false`" à un utilisateur, il ne sera plus capable d'ouvrir une session interactive sur le système néamoins il pourra utiliser le service ftp. Vous souhaiterez peut-être aussi restreindre l'accès en Confiner les utilisateurs à leur répertoire HOME avec ftpd(8).

## 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)

Par défaut, lorsque les utilisateurs se connectent en ftp, ils peuvent aller dans n'importe quel répertoire du système, dans la mesure où les contrôles d'accès leur permettent. Dans certains cas, ce comportement peut ne pas être souhaitable. Il est possible de retreindre les utilisateurs ftp à leur répertoire HOME en utilisant "chroot".

Si vous voulez autoriser les connexions ftp en chroot, utilisez l'option **-A** de ftpd(8).

Si vous voulez utiliser chroot de manière plus fine, consultez "login capability infrastructure" et ftpd(8)

Les utilisateurs appartenant à une classe de connexion où la variable `ftp-chroot` est positionnée seront automatiquement mis dans un chroot. De plus, vous pouvez ajouter un nom d'utilisateur au fichier **/etc/ftpchroot** pour mettre ces utilisateurs dans un chroot. Un utilisateur a uniquement besoin d'être listé dans un de ces endroits.

# 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 *3.5-release* via cvs(1), vous utiliseriez les lignes de commande suivantes :

```
$ export CVSROOT=anoncvs@anoncvs5.usa.openbsd.org:/cvs
$ cvs co -rOPENBSD_3_5_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. Ce mode de configuration n'est pas adopté par la plupart des autres systèmes d'exploitation. 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 existants :** 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 existants :** 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` de httpd(8) dans *rc.conf*.

## Exemple de chroot(2) d'une application : wwwcount

Voyons comment mettre en place chroot(2) pour une application à travers un exemple. Cet exemple se base sur wwwcount, un compteur tout simple d'accès aux pages web disponible dans l'arborescence des ports. Bien qu'il soit un programme très efficace, wwwcount ne sait rien d'Apache chroot(2)é et ne fonctionnera pas dans un environnement chroot(2)é avec sa configuration de base.

Tout d'abord, nous installons un package pour wwwcount. Nous le configurons et le testons et c'est là que nous en déduisons qu'il ne semble pas fonctionner : Apache nous affiche le message "Internal Server Error". La première étape consiste à arrêter Apache et à le redémarrer avec l'option `-u` pour vérifier que le problème vient bien du chroot(2) et pas de la configuration système.

```
# apachectl stop
/usr/sbin/apachectl stop: httpd stopped
# httpd -u
```

Après avoir fait cela, nous constatons que le compteur fonctionne correctement, du moins après avoir changé les droits d'un répertoire afin qu'Apache (et les CGIs qu'il exécute) puisse écrire à des fichiers. Ainsi, nous sommes maintenant certains que le problème vient du chroot. Nous arrêtons alors et redémarrons Apache à nouveau en utilisant le chroot par défaut :

```
# apachectl stop
/usr/sbin/apachectl stop: httpd stopped
# httpd
```

Un bon point pour commencer serait de supposer que wwwcount utilise des librairies et d'autres fichiers qu'il ne peut accéder une fois dans le chroot. On peut utiliser à cet effet la commande ldd(1) pour trouver les dépendances dynamiques dont le CGI a besoin :

```
# cd /var/www/cgi-bin/
# ldd Count.cgi
Count.cgi:
        Start    End      Type Ref Name
        00000000 00000000 exe   1  Count.cgi
        03791000 237ca000 rlib  1  /usr/lib/libc.so.30.3
        03db4000 03db4000 rtld  1  /usr/libexec/ld.so
```

Ah ! voilà un problème. Deux fichiers ne sont pas disponibles dans l'environnement chroot(2). Alors, on les copie dans cet environnement :

```
# mkdir -p /var/www/usr/lib /var/www/usr/libexec
# cp /usr/lib/libc.so.30.3 /var/www/usr/lib
```

```
# cp /usr/libexec/ld.so /var/www/usr/libexec
```

Puis nous essayons le compteur à nouveau.

Au moins, le programme s'exécute maintenant et nous affiche des messages d'erreur directement : "Unable to open config file for reading". Nous avons bien progressé mais nous n'avons pas encore fini. Le fichier de configuration se trouve normalement dans le répertoire `/var/www/wwwcount/conf`, mais au sein de l'environnement chroot, le programme devrait le voir sous `/wwwcount/conf`. Nous avons donc deux options. Soit nous recompilons le programme pour qu'il tienne compte du nouveau fichier de configuration par défaut (où plutôt du chemin pour l'atteindre) soit nous déplaçons les fichiers de données. Vu que nous avons installé le programme à partir d'un package, nous prenons l'option 2, à savoir le déplacement des fichiers de données. Afin que nous puissons utiliser exactement la même configuration dans un environnement chroot(2)é ou pas, nous utiliserons un lien symbolique :

```
# mkdir -p /var/www/var/www
# cd /var/www/var/www
# ln -s ../../wwwcount wwwcount
```

Remarquez que le lien symbolique a été pensé pour fonctionner au sein du chroot. Nous testons notre programme à nouveau et nous voilà avec un autre problème. Maintenant wwwcount se plaint de ne pas trouver les fichiers "strip image" qu'il utilise pour afficher des messages. Après quelques recherches, nous nous rendons compte que ces fichiers sont stockés dans `/usr/local/lib/wwwcount`, nous devons donc les copier dans le chroot aussi.

```
# tar cf - /usr/local/lib/wwwcount | (cd /var/www; tar xpf - )
```

Nous testons à nouveau ... et ça marche !

Dans cet exemple, le programme était simple et malgré ça, nous avons constaté plusieurs problèmes. Certaines applications sont relativement simples et les mettre dans un chroot(2) n'a aucun sens. D'autres sont très complexes. Elles ne valent pas les efforts nécessaires pour les mettre en chroot(2) et même si c'était le cas, vous perdriez tous les avantages du chroot(2) après avoir copié la masse importante de fichiers dont elles ont besoin pour fonctionner. Même avec une application aussi simple que celle que nous avons utilisé pour cet exemple, l'accès au disque est nécessaire (pour mettre à jour les compteurs). Du coup, certains avantages de chroot(2) ont été perdus. De manière générale, le minimum de fichiers nécessaires pour faire fonctionner une application devraient être copiés dans le chroot. *On ne peut ou on ne doit pas chroot(2)er n'importe quelle application*

# 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 vers OpenBSD] [Section 11 - Optimisation des Performances]

---


www@openbsd.org

Originally [OpenBSD: faq10.html,v 1.107 ]
$Translation: faq10.html,v 1.29 2004/10/25 12:56:49 saad Exp $
$OpenBSD: faq10.html,v 1.24 2004/10/25 13:55:37 saad Exp $

[Index de La FAQ] [Section 10 - Gestion du Système] [Section 12 - Questions Spécifiques Aux Plates-Formes]

# 11 - Optimisation des Performances

---

### Table des matières



---

## 11.1 - E/S Disque

La vitesse des E/S sur les disques est un facteur significatif de la vitesse globale de votre machine. Ce facteur devient de plus en plus important quand votre machine héberge un environnement multi utilisateur (des utilisateurs de toutes les catégories tels que les utilisateurs qui se connectent de manière interactive et les utilisateurs qui voient votre machine comme un serveur de fichiers ou un serveur Web). Le stockage de données demande une attention constante et particulièrement quand vos partitions ne contiennent plus d'espace libre ou quand vos disques ne fonctionnent plus. OpenBSD possède plusieurs options pour augmenter la vitesse des opérations sur disque. De plus il fournit des fonctionnalités de tolérance aux pannes.

Table des matières



### 11.1.1 - CCD

La première option consiste à utiliser ccd(4) , le Pilote de Disques Concaténés. Il vous permet de grouper plusieurs partitions en un seul disque virtuel (ainsi vous pouvez rendre plusieurs disques visibles comme un seul disque). Ce concept est similaire au concept de LVM (gestion de volumes logiques) se trouvant dans plusieurs versions Unix commerciales.

Si vous utilisez le noyau GENERIC, ccd est déjà activé (dans `/usr/src/sys/conf/GENERIC`). Si vous avez personnalisé votre noyau, vous aurez éventuellement besoin de refaire la configuration de votre noyau. Dans tous les cas, la ligne suivante devra figurer dans votre fichier de configuration noyau :

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

La ligne précédente vous permet de configurer jusqu'à 4 périphériques ccd (disques virtuels). L'étape suivante consiste à choisir les partitions sur vos disques physiques que vous voulez consacrer à ccd. Utilisez disklabel pour marquer ces partitions en type 'ccd'. Sous certaines architectures, disklabel ne vous autorisera pas à effectuer cette opération. Dans ce cas, marquez les en 'ffs'.

Si vous utilisez ccd pour améliorer les performances à travers la technique du striping, il est à noter que vous n'aurez pas de performance optimale à moins que vous n'utilisiez le même modèle de disques avec le même paramétrage disklabel.

Editez le fichier /etc/ccd.conf afin qu'il ressemble à ce qui suit : (pour plus d'informations sur la configuration ccd, veuillez consulter ccdconfig(8))

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

Pour que vos modifications prennent effet, exécutez :

```
# ccdconfig -C
```

Aussi longtemps que /etc/cdd.conf existera, ccd se configurera automatiquement lors du démarrage de la machine. A ce point, vous avez un nouveau disque dénommé ccd0; combinaison de /dev/sd2e et /dev/sd3e. Pour le partitionner, utilisez disklabel normalement. Nous vous rappelons qu'il ne faut pas utiliser la partition 'c' comme une partition réelle sur laquelle vous pouvez stocker des données. Assurez-vous que vos partitions soient au moins décalées d'un cylindre par rapport au début du disque.

## 11.1.2 - RAID

raid(4) est une autre solution. Elle nécessitera l'utilisation de la commande raidctl(8) pour contrôler vos périphériques raid. le RAID OpenBSD est basé sur le port NetBSD du logiciel CMU RAIDframe par Greg Oster. OpenBSD supporte les niveaux RAID 0, 1, 4 et 5.

Comme pour ccd, le support raid doit être configuré dans le NOYAU. Mais contrairement à ccd, le support RAID n'est pas configuré par défaut dans GENERIC. Il doit compilé au niveau du noyau (le support RAID ajoute 500k à la taille d'un noyau i386) :

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

La configuration du RAID sous certains systèmes d'exploitation peut prêter à confusion et s'avérer difficile. Ce n'est pas le cas avec RAIDframe. Consultez les pages de manuel raid(4) et raidctl(8) Pour des informations complètes. Il y a plusieurs options et configurations possibles. Une explication détaillée est au-delà du périmètre du présent document.

## 11.1.3 - Soft Updates

Un autre outil qui peut être utilisé pour accélerer la vitesse de votre système est softupdates. La mise à jour des informations meta ou metainfo (qui a lieu quand vous créez ou supprimez des fichiers et des répertoires entre autres) est une des opérations les plus lentes du système de fichiers BSD traditionnel. Softupdates tente de mettre à jour les metainfo dans la RAM au lieu d'écrire chaque mise à jour de metainfo sur le disque. Une autre conséquence est que les metainfo sur le disque devraient être toujours complètes, mais pas forcément à jour.

Un crash système ne devrait donc pas nécessiter une opération fsck lors du démarrage de la machine, mais seulement une version d'arrière-plan de fsck(8) qui effectue des modifications aux metainfo en RAM (comme softupdates). Ce qui veut dire que le redémarrage d'un serveur est beaucoup plus rapide puisque vous n'avez pas besoin d'attendre que fsck finisse ! (OpenBSD ne possède pas cette fonctionnalité encore). Pour en savoir plus, consultez La FAQ Softupdates .

### 11.1.4 - Taille du cache namei()

Remarque : précédemment, la page du manuel options(4) recommandait de positionner l'option du noyau `NVNODE=integer`. Ce n'est plus recommandé. Au lieu de cela, utilisez la commande sysctl(8)

Le cache de traduction nom-vers-inode (name-to-inode ou namei()) contrôle la vitesse de la traduction chemin vers inode(5). Une valeur raisonnable de fixer une valeur pour le cache, si on venait à remarquer à l'aide d'un outil comme systat(1) , des erreurs d'allocation au niveau du cache, est d'examiner la valeur courante générée avec sysctl(8) , (qui appelle ce paramètre "`kern.maxvnodes`") et d'augmenter cette valeur soit jusqu'à ce que les le taux de réponse à partir du cache namei s'améliore soit jusqu'à ce qu'on détermine que le système ne bénéficie plus substantiellement de l'augmentation de la taille du cache namei. Une fois que la valeur est déterminée, vous pouvez la fixer au démarrage du système à l'aide de sysctl.conf(5).

## 11.2 - Choix matériels

*(Remarque - cette section est très centrée autour de l'architecture i386 ou PC. Quelque part cela signifie que les autres architectures ne vous donnent pas autant de choix! )*

La performance de vos applications dépend beaucoup de votre système d'exploitation et des fonctionnalités qu'il fournit. C'est peut-être une des raisons pour laquelle vous utilisez OpenBSD. La performance de vos applications dépend beaucoup de votre matériel aussi. Pour beaucoup de gens, le rapport prix/performance d'un nouveau PC doté d'un processeur Intel Pentium III ou AMD Athlon est bien meilleur que le rapport prix/performance d'une station Sun UltraSPARC 60! Et bien sûr, le prix de OpenBSD ne peut être battu.

Si vous allez acheter un PC, pièce par pièce ou assemblé, vous devez vous assurer que vous allez acheter des composants fiables. Dans le monde PC, ce n'est pas quelque chose de facile. **OpenBSD peut être lent et souvent planter à cause de mauvais composants, des composants non fiables ou un mauvais assortiment de composants**. Le meilleur conseil que nous pouvons vous donner est d'être très attentif, achetez des marques et des composants qui ont été testées par une autorité dans laquelle vous avez confiance. Parfois lorsque vous lésinez sur le prix d'un PC, vous perdez en qualité!

Il y a certaines choses qui vont vous aider à tirer le meilleur parti de votre matériel :

- **Utiliser plusieurs disques.** Au lieu d'acheter un disque de grande capacité, achetez plusieurs disques de moindre capacité. Bien que ça risque de vous coûter plus cher, distribuer la charge sur plusieurs têtes diminuera le temps nécessaire pour accéder aux données sur les disques. Et avec plus d'axes, vous obtiendrez un meilleur niveau de fiabilité et un accès aux données plus rapide avec le système RAID.

- **Utilisez du SCSI si vous avez besoin de très hautes vitesses en E/S disque.** Les disques IDE ont des vitesses de rotation comprises entre 5400TPM à 7200TPM. En utilisant des disques IDE haut de gamme, il n'est pas raisonnable de s'attendre à plus de 15 à 20 Mégaoctets de débit par seconde sur un seul disque. En utilisant des disques SCSI haut de gamme (des disques plus chers à 10000 RPM), vous pouvez atteindre des performances supérieures. Inversement, si vous allez utiliser des disques SCSI milieu ou bas de gamme, c'est une perte d'argent. Et la technologie IDE sera aussi bonne sinon meilleure.

  Si vous mettez en place un serveur, et que vous avez besoin de plus d'un disque, considérez la technologie SCSI. IDE vous limitent à deux disques par contrôleur. Des accès simultanés à ces deux disques peuvent avoir un impact négatif sur

les performances E/S de ces disques. SCSI "Wide" vous limitent à 15 par contrôleur et gère les accès simultanés mieux qu'IDE. Certes, la technologie SCSI est plus coûteuse, mais sa flexibilité et ses performances peuvent justifier ces coûts dans certains environnements.

- **Utiliser de la SDRAM au lieu de la DRAM.** Cette option est surtout valable pour les PCs. La plupart des autres architectures ne vous laissent pas le choix du type de RAM à utiliser. Vous aurez de meilleures performances avec de la SDRAM par rapport à de la DRAM (SIMMs). Si votre système supporte la RDRAM ou un autre type nouveau de RAM, alors vous êtes déjà en avance...

- **Utilisez de la mémoire ECC ou de la RAM à parité.** La parité ajoute certaines fonctionnalités pour vérifier si les données en RAM ont été corrompues. La mémoire ECC étend ces fonctionnalités et tente de corriger certaines erreurs de corruption de bits à la volée. Cette option s'applique surtout aux PCs. La plupart des autres architectures requièrent simplement de la mémoire ECC ou à parité. Plusieurs machines non-PC ne démarreront même pas avec de la RAM sans parité. Si vous n'utilisez pas de la mémoire ECC ou à parité, vous aurez peut-être de la corruption de données et d'autres anomalies. Plusieurs fabricantsde "RAM à faible coût pour PC" ne font même pas une variété ECC! Ce qui vous aidera à les éviter! Les fabricantsde PC vendent souvent plusieurs gammes de produits, organisées en "serveurs" et "stations de travail". Les serveurs vont contenir de la RAM ECC dans leur architecture. Les fabricantsde stations de travail Unix utilisent de la mémoire à parité (et maintenant ECC) depuis plusieurs années dans toutes leurs gammes de produits.

- **Evitez les périphériques ISA.** Alors que la plupart de gens évitent les périphériques ISA, parce qu'ils sont généralement difficiles à configurer et obsolètes, il en existe encore beaucoup. Si vous utilisez un bus ISA pour votre disque ou vos contrôleurs réseau (ou, encore plus grave, pour les deux), rappelez vous que le bus ISA en lui-même peut être un goulet d'étranglement. Si vous avez besoin de vitesse, choisissez le bus PCI. Bien entendu, il y a plusieurs cartes à bus ISA qui fonctionnement bien. Malheureusement, la plupart de ces cartes sont des cartes son ou des cartes série.

- **Evitez les cartes réseau PCI bas de gamme.** OpenBSD supporte une pléthore de cartes réseau PCI bas de gamme Ces cartes fonctionnement très bien en environnement domestique, et aussi dans des environnements d'entreprise ou de recherche à débit faible ou modéré. Mais si vous avez besoin de haut débit avec un faible impact sur votre serveur, il serait plus judicieux d'opter pour une carte réseau PCI de qualité. Malheureusement, certaines cartes coûteuses ne sont pas mieux que des cartes bas de gamme. Les adaptateurs Gigabit ont souvent de meilleures performances que les adaptateurs 10Mbps/100Mbps même lorsqu'ils sont utilisés sur des réseaux à vitesse plus lente vu leur meilleure mise en tampon.

# 11.3 - Pourquoi nous n'utilisons pas de montage asynchrone ("async mount") ?

Question : "Je fais simplement "mount -u -o async /" ce qui rend un package dont j'ai besoin (qui touche à une centaine de chose de temps à autre) utilisable. Pourquoi le montage asynchrone n'est pas vu d'un bon œil et n'est pas activé par défaut (comme c'est le cas sur d'autres versions d'Unix) ? C'est un mécanismesûrement plus simple et plus sûr d'améliorer les performances de certaines applications."

Réponse : " les montages asynchrones sont en effet plus rapides que des montages synchrones, mais ils sont aussi moins sûrs. Qu'arrive-t-il dans le cas d'une panne de courant ? Ou un problème matériel ? la quête de la vitesse ne doit pas sacrifier la fiabilité et la stabilité du système. Reportez-vous à la page du manuel de 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.
```

D'un autre côté, quand vous travaillez avec des données temporaires que vous pouvez recréer après un plantage, vous pouvez gagnez en vitesse en utilisant une partition à part montée en asynchrone, utilisée uniquement pour ce type de données. Encore une fois, n'effectuez cette opération *que si* vous ne voyez pas d'inconvénient à perdre toutes les données de cette partition si quelque chose va mal. Pour cette raison, les partitions mfs(8) sont montées en mode asynchrone vu que de toute façon, elles vont être écrasées et recrées après un redémarrage.

# 11.4 - Optimisation de la Résolution de Votre écran sous XFree86

*Remarque : La plupart des utilisateurs n'ont pas à se soucier de la création manuelle du ModeLine avec les versions récentes de X. CEPENDANT, cette opération est parfois nécessaire dans des situations unusuelles.*

Avec plusieurs moniteurs multi-sync, il est possible de configurer un serveur X avec une résolution acceptable. Si quelqu'un a déjà essayé de faire ceci avec les utilitaires standards xf86config ou XF86Setup, il est fort probable qu'il n'aie pas obtenu les meilleurs résultats possibles. Un des aspects les plus difficiles est de simplement configurer votre moniteur avec votre résolution préférée, puis de paramétrer la vitesse de balayage vertical à une valeur d'au moins 72-75 Hz, une valeur où le scintillement de l'écran est beaucoup moins visible à l'oeil humain. Inversement, qu'en est t'il quand vous essayez de positionner la vitesse de balayage vertical à une vitesse très faible ? Vous pouvez le positionner à 50 Hz de telle manière à pouvoir en capturer le contenu sur une vidéo sans scintillement, mais les méthodes pour effectuer cette opération ne sont pas évidentes avec les outils et la documentation standards de XFree86.

Enfin, il est possible (avec les nouveaux moniteurs du moins) d'utiliser des vitesses de balayage vertical de 85 Hz et plus, afin d'obtenir une image très propre et agréable à des résolutions normalement utilisées par beaucoup de gens sur des moniteurs VGA à faible coût (800x600, 1024x768, 1152x900, 1280x1024). Le serveur X XFree86 possède un mécanismequi vous permet de décrire en détail le mode vidéo que vous voulez utiliser : C'est le ModeLine. Un ModeLine a quatre sections. Un chiffre pour la dotclock (bande passante), quatre chiffres pour les temporisations horizontales, quatre chiffres pour les temporisations verticales, et une section optionnelle contenant une liste de drapeaux spécifiant d'autres caractéristiques du mode (telles que l'interlaçage, DoubleScan, ... Voir la page XF86Config(5) du manuel pour plus de détails concernant ModeLine).

La génération d'un ModeLine s'apparente à de la magie noire... Heureusement, il existe plusieurs scripts qui peuvent vous en générer un. Un de ces scripts est Colas XFree86 ModeLine Generator. Un autre est The XFree86 ModeLine Generator hébérgé par SourceForge et plusieurs autres sont disponibles sur Freshmeat. Avant de pouvoir utiliser ces générateurs ModeLine, vous devez trouver les limites de synchronisation verticale et horizontale de votre moniteur. C'est souvent documenté dans le manuel ou sur le site Web du constructeur. Si ne vous trouvez ni l'un ni l'autre, faites une recherche sur le Web avec le modèle et la marque du moniteur. Plusieurs personnes sympathiques ont rassemblé des listes avec ce type d'informations.

Par exemple, supposons que vous avez un moniteur Dell D1226H. Vous avez effectué des recherches en agonisant sur le site Web de Dell pour savoir qu'il a une plage de balayage horizontal de 30-95 kHz et 50-160 Hz en balayage vertical. Visitez la page du générateur de ModeLine et saisissez ces informations. La prochaine étape consiste à saisir la vitesse minimale de balayage vertical que vous voulez. N'importe quelle vitesse de 72 Hz et au delà devrait permettre d'obtenir un scintillement peu visible. Plus vous montez la vitesse, plus votre image est claire et précise.

Avec toutes ces informations, le script va générer un ModeLine pour toutes les résolutions 4x3 possibles que votre moniteur peut supporter, et au-dessus de la vitesse minimale de balayage verticale que vous avez saisi. Si quelqu'un avait saisi les spécifications de ce moniteur Dell avec une vitesse minimale de balayage verticale de 75 Hz, le script donnera quelque chose comme :

```
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
```

Bien que ce moniteur prétend atteindre 1600x1200 @ 75 Hz, le script ne dit pas que cette résolution peut être atteinte à 75 Hz. Si vous voulez précisément avoir 1600x1200, abaissez un peu votre vitesse minimale ... (Dans l'exemple qui suit, nous fixons cette vitesse à 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
```

Comme on peut le voir, ce moniteur peut faire 1600x1200 @ 74 Hz quand la vitesse de point (bande passante) est limitée à 200 Mhz. Positionnez la bande passante par rapport aux limites définies par le moniteur.

Une fois que vous avez vos ModeLines, mettez les dans le fichier /etc/X11/XF86Config. Mettez en commentaire les anciens ModeLines de telle façon à pouvoir les utiliser par la suite si les nouveaux ne fonctionnent pas. Ensuite choisissez votre résolution. D'abord vous devez déterminer si X tourne en mode accéléré (ce qui est le cas avec la plupart des cartes graphiques) afin de savoir quelle section `"Screen"` modifier dans le fichier XF86Config. Sinon, modifiez toutes les sections Screen.

```
Section "Screen"
   Driver          "Accel"
   Device          "Primary Card"
   Monitor         "Primary Monitor"
   DefaultColorDepth 32
   SubSection "Display"
      Depth        32
      Modes        "1280x1024" "1024x768"
   EndSubSection
```

La première fréquence que vous voyez après la ligne "Modes" est la fréquence avec laquelle X va démarrer. En appuyant sur les touches CTRL-ALT-MOINS DU PAVE NUMERIQUE, ou CTRL-ALT-PLUS DU PAVE NUMERIQUE, vous pouvez commuter entre les fréquences que vous avez spécifié sur cette ligne. Selon la section ci-dessus, X va essayer de démarrer en mode couleur 32 bits (via la directive DefaultColorDepth sans laquelle X va démarrer en mode couleur 8 bits). La première résolution qu'il essayera de charger est 1280x1024 (il suit l'ordre précisé dans la ligne Modes). Notez que "1280x1024" est simplement un label des valeurs du ModeLine.

Il est à noter que le script de génération du ModeLine possède des options qui permettent de rendre ces temporisations plus flexibles pour les vieux ou petits moniteurs. De même, il a la capacité de fournir des ModeLines pour des résolutions spécifiques. Selon le type de matériel que vous avez, il ne sera peut-être pas facile de l'utiliser avec les options par défaut. Si l'image est trop grande, trop large, ou trop petite, ou elle est décalée horizontalement ou verticalement, et les contrôles du moniteur ne sont pas suffisants pour corriger son apparence, vous pouvez utiliser xvidtune(1) afin d'ajuster le ModeLine pour mieux respecter les temporisations du moniteur.

Sur la plupart des moniteurs modernes, il n'y a pas de limite fixée sur la bande passante. C'est pourquoi elles ne sont plus listées dans les spécifications. Mais plus vous augmentez la bande passante, plus l'image devient floue. Ainsi, vous pouvez commencer par la bande passante (appelée aussi "dotclock") de votre carte pour tester (vous ne pouvez pas abimer votre moniteur comme ça) et diminuer la bande passante au fur et à mesure jusqu'à ce que vous obteniez une image précise.

Cette procédure est inutilement complexe. XFree86 4.0 améliore nettement ce processus puisque cette version possède des modes internes et est capable de découvrir les capacités des moniteurs "plug and play" à travers DCC et DCC2. 5A

Vous pouvez télécharger le script Colas XFree86 ModeLine Generator à l'adresse http://koala.ilog.fr/ftp/pub/Klone/. Vous aurez besoin de télécharger et de compiler l'interpréteur Klone. Cet interpréteur se trouve dans la collection de ports sous `lang/klone`. Les scripts sont dans le répertoire scripts de la distribution Klone (la collection de ports les installe dans /usr/local/klone/scripts).

Deux versions du script sont incluses. La première est une version CGI identique à la page Web ci-dessus. La seconde est une version non-CGI qui va lire votre fichier XF86Config dans sa totalité, décoder les spécifications du moniteur que vous avez saisi dans xf86config/XF86Setup (Réfléchissez, avez-vous réellement renseigné le fichier avec les spécifications de votre moniteur ou avez-vous gardé les valeurs génériques ?), et corriger les ModeLines existants en conséquence.

[Index de La FAQ] [Section 10 - Gestion du Système] [Section 12 - Questions Spécifiques Aux Plates-Formes]

---


www@openbsd.org

Originally [OpenBSD: faq11.html,v 1.54 ]
$Translation: faq11.html,v 1.21 2004/10/25 09:57:24 saad Exp $
$OpenBSD: faq11.html,v 1.16 2004/10/25 12:38:57 jufi Exp $

[Index de La FAQ] [Section 11 - Optimisation des Performances] [Section 14 - Configuration des Disques]

# 12 - Questions Spécifiques Aux Plates- Formes

---

## Table des matières



---

## 12.1 - Notes Générales sur le Matériel

### 12.1.1 - Périphériques PCI

- Les périphériques PCI sont pratiquement tous auto- configurés -- la machine et le système d'exploitation vont allouer des ressources aux cartes si nécessaire.
- Les interruptions peuvent être partagées sur le bus PCI. Non seulement ils le peuvent, mais le système va souvent être plus performant lorsque les IRQs sont partagés, et particulièrement sur les systèmes i386.
- Il existe plusieurs standard PCI différents. Occasionnellement, vous trouverez une carte de spécification PCI2.2 qui va juste fonctionner sur un système de spécification 2.1. De même, plusieurs cartes avec des ponts intégrés (telles que les cartes réseaux multi-port) ne fonctionneront pas correctement sur d'anciens systèmes.
- Le bus PCI supporte deux niveaux de signalisation, 3.3v et 5v. Les cartes fonctionnant avec la signalisation 3.3v ont une seconde "coupure" au niveau de leur connecteur PCI. La plupart des cartes PCI utilisent la signalisation 5v, utilisée par la plupart des ordinateurs. Les ordinateurs à carte unique Soekris (Net45x1 et Net4801) sont des machines communes qui ne supportent que la signalisation 3.3v.

### 12.1.2 - Périphériques ISA

- Les périphériques ISA ne peuvent partager de ressources et de manière générale, doivent être configurés manuellement en utilisant des paramètres qui ne rentrent pas en conflit avec d'autres périphériques système.
- Quelques périphériques ISA sont"Plug and Play" (isapnp(4)) -- si vous avez des problèmes avec de tels périphériques, vérifiez leur configuration dans votre dmesg(8), ISAPNP ne fonctionne pas toujours comme on le souhaite.
- De manière générale, si vous avez le choix évitez les cartes ISA au profit de cartes PCI. Les cartes ISA sont plus difficiles à configurer et ont un impact négatif plus grand sur les performances système.

### 12.1.3 - Un périphérique est "reconnu" mais il est marqué comme "`not configured`" dans le dmesg

Ceci veut clairement dire que votre périphérique n'est pas supporté par le noyau que vous utilisez. Vous ne serez donc pas en mesure de l'utiliser.

PCI et de nombreux autres types de périphériques offrent des informations d'identification afin que le système d'exploitation puise les reconnaître correctement et activer s'il y a lieu le support de ces périphériques. Ajouter ces informations est chose facile, ajouter le support d'un périphérique ne l'est pas. Souvent. Voici une partie d'un dmesg avec deux exemples de périphériques "not configured" :

```
...
vendor "Intel", unknown product 0x5201 (class network subclass
ethernet,
rev 0x03) at pci2 dev 9 function 1 not configured
...
"Intel EE Pro 100" rev 0x03 at pci2 dev 10 function 0 not configured
...
```

Dans le premier exemple (une carte réseau), le code du constructeur a été identifié ainsi que le type générique de la carte. Cependant, le modèle exact de cette carte n'a pu être identifié. Le second exemple montre une autre carte réseau qu'un développeur a vu et a saisi dans le fichier d'identification utilisée pour identifier la carte. Cependant, les cartes ne seront pas fonctionnelles dans les deux cas vu qu'elles sont affichées comme "`not configured`", ce qui veut dire qu'aucun pilote ne leur a été affecté.

#### Que puis-je faire dans le cas d'un périphérique `not configured` ?

- Si le périphérique ou la carte que vous avez ne vous est pas utile, vous pouvez ignorer les périphériques "`not configured`" puisqu'ils ne causeront aucun dommage à votre système. Certains périphériques spéciaux ne sont pas configurés à bon escient afin que le BIOS du système puisse les gérer.

- Dans certains cas, c'est juste une variante d'un périphérique supporté. Si c'est le cas, il serait relativement facile pour un développeur d'ajouter le support de cette nouvelle carte. Dans d'autres cas, c'est peut- être une puce ou une implémentation qui n'est absolument pas supportée (telle que dans les exemples précédents). Dans ce cas, un nouveau pilote devra être écrit mais ce n'est pas toujours possible. Des fois, le périphérique n'est pas entièrement documenté. Vous êtes bien entendu invité à écrire un pilote pour le périphérique vous-même si vous en avez les compétences.
- Si vous utilisez un noyau d'installation, le périphérique n'est peut-être pas supporté par la méthode d'installation que vous avez utilisé mais supporté par un disque de démarrage différent. C'est chose commune pour des utilisateurs de certaines cartes SCSI populaires qui ont mal lu les notes de bas de page sur la page consacrée à la plate-forme i386 et qui essaient de démarrer avec des disquettes de démarrage ne supportant pas leur carte SCSI au lieu d'utiliser la disquette adéquate.
- Si vous utilisez un noyau modifié, vous avez peut-être supprimé le support du périphérique dont vous avez besoin. De manière générale, la suppression de périphériques dans le noyau est une mauvaise idée.
- Avant de faire un rapport sur un périphérique "`not configured`", assurez-vous d'abord que vous avez utilisé le dernier snapshot disponible pour voir si le support de ce périphérique n'a pas été récemment ajouté. Puis vérifiez les archives des listes de diffusion pour voir si le problème n'a pas été discuté. Rappelez-vous cependant que si vous utilisez une ancienne version d'OpenBSD, vous devriez mettre à jour votre système pour bénéficier des derniers pilotes.

## 12.1.4 - J'ai une carte listée dans le matériel "supporté" mais elle ne fonctionne pas !

Malheureusement, beaucoup de constructeurs utilisent les numéros de modèle pour indiquer leur position sur le marché, au lieu de la nature technique du produit. Pour cette raison, vous avez peut-être acheté un produit avec le même nom et du même modèle qu'un produit listé dans les pages consacrées aux plates-formes, alors qu'en réalité c'est un produit entièrement différent qui ne fonctionne pas sous OpenBSD. Par exemple, plusieurs cartes réseau sans-fil sorties mises sur le marché il y a un certain temps utilisaient la puce Prism2 donc le pilote (wi(4)) Mais plus tard, lorsque des puces à bas prix sont devenues disponibles, plusieurs fabricants ont changé leur produit pour utiliser des puces pour lesquelles aucun pilote libre n'existait. En revanche, ils n'ont pas changé le nom de leur produit. Les cartes réseau sans-fil sont malheureusement loin d'êtres les seules dans ce cas.

## 12.1.5 - 12.1.5 - Quels sont les types de cartes Sans Fil supportées par OpenBSD ?

OpenBSD supporte un certain nombre de chipsets sans fil:

- awi(4) AMD 802.11 PCnet Mobile
- an(4) Aironet Communications 4500/4800
- wi(4) Prism2/2.5/3
- atw(4) ADMtek ADM8211 (*-current UNIQUEMENT*)

Les adaptateurs basés sur ces composants peuvent être utilisés de la même façon que n'importe quel autre adaptateur réseau pour connecter un système OpenBSD à un réseau sans fil existant (prière de consulter les pages du manuel pour des détails précis). Cependant, les cartes à base de Prism2 et Prism3, peuvent aussi être utilisées en mode "Host-Based Access Point", ce qui leur permet d'être utilisées comme un point d'accès sans fil pour votre réseau, intégré à votre pare-feu. Malheureusement, les cartes réseau à base de Prism2/Prism3 ne sont plus produites par les fabricants du marché de masse. En effet ces derniers utilisent maintenant des composants plus récents et à moindre coût souvent sans modifier le numéro de série. La plupart des fabricants des nouveaux composants ont choisi de ne pas publier la documentation nécessaire pour produire des pilotes libres et ouverts pour ces périphériques (vous pouvez les contacter et essayer de les convaincre de changer leur politique).

D'un autre côté, les adaptateurs Prism2/Prism3 sont encore disponibles sur les marchés d'occasion et de surplus et quelques usines comprennent la valeur d'un composant correctement documenté et compatible avec le logiciel libre et continuent de produire des adaptateurs à base de Prism de qualité professionnelle. Parmi celles-ci, on peut citer Netgate.com. De manière générale, si le fabricant n'identifie pas de manière explicite un produit récent comme étant un produit à base de Prism, vous pouvez estimer qu'il n'est pas compatible avec le pilote wi(4). Il est à noter que les nouveaux composants Prism (tels que le Prism-GT) ne sont pas encore supportés.

Une autre méthode pour permettre à un pare-feu OpenBSD de fournir un accès sans fil est d'utiliser une carte conventionnelle et un point d'accès externe en mode pont. Cette méthode a l'avantage de vous permettre de positionner l'antenne là où elle est la plus

efficace.

# 12.2 - DEC Alpha

[aucune information pour le moment]

# 12.3 - AMD 64

[aucune information pour le moment]

# 12.4 - CATS ARM development board

[aucune information pour le moment]

# 12.5 - HP300

[aucune information pour le moment]

# 12.6 - HPPA

[aucune information pour le moment]

# 12.7 - i386

## 12.7.1 - Cartes réseau ISA

Etant donné qu'OpenBSD fonctionne bien sur le matériel ancien, les utilisateurs finissent souvent par utiliser des cartes réseau NIC avec les systèmes OpenBSD. Le matériel ISA nécessité beaucoup plus de configuration et de compréhension que le matériel PCI. De manière générale, vous ne pouvez pas vous contenter d'insérer votre carte dans la machine et espérer qu'elle fonctionne par magie. Avec plusieurs machines, si votre périphérique ISA n'est pas en mode "Plug 'n' Play" (PNP), vous devez réserver les ressources que la carte utilise au niveau du BIOS.

#### 3Com 3C509B ep(4)

C'est une carte réseau ISA très performante, supportée par le périphérique ep(4). La version 'B' peut être distinguée de la version non-B par le nom figurant sur la carte et par une puce principale plus large sur la carte (approximativement 2.5cm sur un côté pour la version 'B' vs. 2cm sur un côté pour l'ancienne version), et elle fournira de meilleures performances sur un système chargé ou doté de deux cartes réseau. Les 3C509B sont livrées avec une configuration en mode PNP, qui malheureusement n'est pas conforme aux standards, et cause des problèmes dans le support isapnp(4) d'OpenBSD. L'adaptateur est d'abord sélectionné comme périphérique non-PNP, puis une seconde fois lorsque le support PNP devient disponible. Le résultat est l'affichage d'une carte réseau supplémentaire dans le dmesg. Le fonctionnement peut alors être correct ou problématique. Il est hautement recommandé de désactiver le mode PNP pour les cartes 3C509B et de configurer manuellement la carte avec des paramètres non conflictuels à l'aide des utilitaires DOS 3Com avant la configuration.

Le pilote ep(4) va trouver les cartes en utilisant n'importe quelle combinaison matérielle qui ne cause pas de conflit avec d'autres périphériques dans le système.

Si vous avez plusieurs cartes 3C509 dans votre système, il est recommandé d'inscrire l'adresse MAC sur la surface des cartes située à l'extérieur du boîtier et d'utiliser dmesg pour identifier les cartes.

Il est à noter que les cartes 3C509, 3C905 et 3C590 sont souvent confondues. La carte 3C509 est une carte ISA 10Mbps, les cartes 3C905 et 3C590 sont des cartes PCI.

#### NE2000

La première carte NE2000 a été développée au milieu des années 1980 par Novell. Depuis, plusieurs constructeurs ont produit des cartes très similaires généralement appelées "compatible NE2000" ou clones. La performance de ces cartes clones varie énormèment. Alors que quelques anciennes cartes "compatibles NE2000" ont de très bonnes performances, la plupart des cartes actuellement disponibles ont de faibles performances. Les cartes "compatible NE2000" sont supportées par le pilote ne(4) sous OpenBSD.

OpenBSD va bien gérer quelques cartes "compatible NE2000" capables d'utiliser ISAPNP lorsque le mode ISAPNP est activé. D'autres cartes vont devoir être configurés soit par le biais de cavaliers soit à l'aide d'un utilitaire de configuration sous DOS. Malheureusement, les premières cartes NE2000 n'avaient pas de support pour la configuration logicielle ou ISAPNP, il n'y a aucun standard -- vous avez besoin de l'utilitaire fourni au départ avec votre carte. Ce qui peut souvent être difficile à obtenir.

Le pilote ne(4) supporte trois configurations des cartes ISA NE2000 dans le noyau GENERIC OpenBSD :

```
ne0:  port 0x240 irq 9
ne1:  port 0x300 irq 10
ne2:  port 0x280 irq 9
```

Si ces paramètres ne sont pas acceptables, vous pouvez les ajuster en utilisant User Kernel Configuration (UKC) ou en compilant un noyau personnalisé.

Il est à noter que le pilote ne(4) est assez "idiot" -- seul le port E/S est sondé, l'IRQ correspondant est *supposée*. dmesg(8) ne va pas refléter l'IRQ réelle de l'adaptateur dans le cas des pilotes ISA ne(4). Si ce n'est l'IRQ réelle utilisée par votre carte, ça ne marchera pas.

Il est à noter qu'il existe des cartes non-ISA qui utilisent le pilote ne(4) -- des cartes PCI et PCMCIA existent. Ces notes ne s'appliquent pas à ces cartes qui sont auto-configurés.

## 12.7.2 - OpenBSD ne fonctionne pas sur mon système 0386/80386SX/80486SX !

Le 80386sx peut adresser au maximum 16M, ce qui est au plus près du minimum supporté par OpenBSD/i386. La plupart des systèmes 80386sx ne peuvent supporter plus de 8M de RAM, ce qui les placent dans la catégorie 'Pour Experts Uniquement", puisque des étapes non triviales et une seconde machine sont nécessaires pour commencer. Consultez aussi la prochaine section :

#### 80386

OpenBSD pourra fonctionner sur un système 80386 ou 80386sx SI ce système a un coprocesseur mathématique (Floating Point Unit, ou FPU) 80387 ou 80387sx. Malheureusement, ces FPUs ne sont pas très communs, plusieurs systèmes 80386 ne les auront donc pas. OpenBSD ne pourra pas fonctionner sans ce FPU sur plate-forme i386. Encore une fois, soyez conscient que c'est vraiment un processeur très faible pour un système d'exploitation comme OpenBSD qui utilise autant le chiffrement. Vous ne serez probablement satisfait des performances d'une telle machine pour une utilisation générale.

#### 80486SX

Le processeur 80486SX fût une version "bas de gamme" du 80486. Il lui manque le support matériel de la virgule flottante (comme le 80386) qu'OpenBSD nécessite. Heureusement, des processeurs complets 80486DX sont assez communs et constituent une mise à jour facile sur la plupart des systèmes.

## 12.7.3 - Mon dmesg affiche plusieurs périphériques partageant la même interruption (IRQ) !

C'est entièrement acceptable et, en vérité, même souhaitable pour les périphériques PCI. C'est une caractéristique de la conception du bus PCI. Certaines personnes diront que le partage de requêtes d'interruption est une mauvaise chose, Cependant ils confondent la situation avec un bus ISA (où le partage d'IRQs n'est pas permit) ou ils ont une expérience avec du mauvais matériel ou logiciel.

Les périphériques ISA ne peuvent pas partager des IRQs. Si vous trouvez des périphériques ISA partageant des IRQs, vous devez corriger le problème.

## 12.7.4 - Comment utiliser un clavier USB

OpenBSD/i386 pourra normalement utiliser un clavier et une souris USB sans problème après l'installation du système. Cependant, installer OpenBSD sur un système équipé d'un clavier USB peut s'avérer difficile vu que les noyaux utilisés lors de la phase d'installation n'ont pas tous les pilotes USB requis pour un support complet des claviers USB. Les machines étant différentes, vous aurez besoin de faire quelques expériences :

- Certains systèmes ont une option BIOS appelée "Legacy USB support" ou un nom similaire. Cette option permet au BIOS d'émuler un clavier PS/2 traditionnel. Si elle n'est pas activée, vous *ne* serez pas en mesure de saisir des commandes au niveau de l'invite de commandes `boot>`. Cependant, l'effet de cette option varie d'un système à l'autre. Certains systèmes fonctionneront mieux avec l'option activée alors que d'autres fonctionneront mieux si elle est désactivée. De même, certains systèmes pourront être plus facilement installés avec l'option activée, d'autres pourront être plus facilement installés avec l'option désactivée.
- Si vous utilisez "Legacy USB support", désactiver le support USB du noyau peut être nécessaire au niveau de l'utilitaire User Kernel Configuration (UKC). Ceci est possible en utilisant la commande "`disable uhci`" avant la fin de la procédure de démarrage.
- Les adaptateurs PS/2 vers USB pour clavier/souris peuvent ne pas fonctionner.
- Il se peut que vous *trouviez* le système plus stable avec une souris déconnectée jusqu'à ce que la procédure d'installation soit complétée.
- Dans le pire des cas, installez OpenBSD sur une autre machine, puis déplacez le disque dur sur une autre machine "legacy-free".

Une fois OpenBSD installé :

- L'utilisation d'UKC pour désactiver pckbd0 aura comme conséquence une diminution des messages de la catégorie "bruit noyau" (`pckdc: cmd failed` habituellement). Cependant, si vous effectuez une telle action, vous ne serez plus capable d'utiliser une souris PS/2.
- Les clavier et souris USB sont gérés comme n'importe quels clavier et souris PS/2 par X. Le pilote souris est "wscons".
- Actuellement, il n'y a aucun moyen pour configurer une console série sur un système "legacy-free" (pas de port série, parallèle ou PS/2). Si vous rencontrez des problèmes avec votre système, vous devrez notez les messages à la main.

La racine du problème vient de la variété de façons dont les PCs gérent l'USB aujourd'hui. C'est pourquoi vous auriez peut-être à faire des expériences avec votre système pour faire fonctionner le clavier USB correctement. Merci de contacter faq@openbsd.org si vous avez d'autres astuces bien documentées concernant les claviers USB.

## 12.7.5 - Mon clavier / ma souris n'arrête pas de se bloquer (ou d'avoir un comportement complètement erratique) !

Ces symptômes se produisent souvent lorsqu'on utilise un "boîtier switch" pour connecter plusieurs machines à un seul clavier, un seul écran et une seule souris. Vous pouvez essayer plusieurs "boîtier switch" de différentes marques/conceptions. Cependant, OpenBSD est plus sensible à la commutation de la souris que d'autres systèmes d'exploitation. Le problème vient plus souvent de la commutation de la souris. Si vous n'utilisez pas la souris, la solution est simple : ne connectez pas le câble souris à la machine. Si vous utilisez la souris, une solution de contournement est d'utiliser une souris par machine et de continuer à faire de la commutation clavier et écran. Si vous voulez uniquement avoir un accès console à la machine, vous devriez peut-être considérer l'utilisation d'une console série.

### 12.7.6 - Est que les WinModems sont supportés ?

Les WinModems sont des modems à bas prix qui s'appuient sur le processeur pour gérer le traitement du signal qui est normalement effectué au niveau matériel dans un "vrai" modem. Vu le nombre de composants WinModem incompatibles et typiquement non documentés, OpenBSD ne supporte pas les WinModems et ce n'est pas prêt de changer.

## 12.8 - Mac68k

### 12.8.1 - Mon système Mac68k ne semble pas fonctionner !

Les développeurs n'ont malheureusement accès qu'à un petit nombre de machines Mac86k différentes. C'est pour cette raison que le bon fonctionnement n'a été validé que pour quelques machines.

### 12.8.2 - Plus le temps passe, plus mon horloge dérive. Pourquoi ?

Ce dysfonctionnement est causé par un bogue matériel. OpenBSD utilise des interruptions d'horloge pour maintenir le temps, mais ces interruptions ont la plus basse priorité dans l'architecture Mac68k. Les interruptions d'horloge sont ainsi perdues sous une charge importante (activé disque ou réseau par exemple) et l'horloge Unix ne fonctionne pas comme il se doit. Les systèmes MacPPC n'ont pas ce problème.

Mac OS contourne ce problème en consultant continuellement l'horloge matérielle. En revanche, OpenBSD ne la consulte que lors de la phase de démarrage et l'ignore par la suite. Vous remarquerez peut-être que, lors d'un arrêt, le noyau n'a pas suffisamment confiance pour réinjecter le temps Unix dans l'horloge matérielle puisque ce problème de perte d'interruptions est bien connu.

Une solution simple consiste à utiliser rdate(8) régulièrement avec une entrée crontab. Vous pouvez aussi lancer rdate (8) à partir de votre fichier */etc/ppp/ppp.linkup* si vous n'êtes pas connecté de façon permanente et que vous êtes un utilisateur PPP.

Voir aussi : http://www.macbsd.com/macbsd/macbsd-docs/faq/faq-3.html#ss3.17

### 12.8.3 - Mon système Mac68k ne veut pas fonctionner avec deux disques

C'est un problème connu qui se manifeste habituellement par des plantages peu de temps après le passage du système en multi-utilisateur. Le plantage peut ne pas être lié au disque système. La solution consiste à utiliser une seul disque sur vos systèmes Mac68k. Une meilleure solution est la bienvenue !

### 12.8.4 - Le programme d'installation a planté durant l'installation

Le programme d'installation Mac68k exécuté par Mac OS n'installera pas de fichiers sur une "grande" partition. Si n'importe quelle partition sur laquelle vous effectuez l'installation a une taille plus grande que 500M, vous pouvez vous attendre à l'erreur étrange suivante :

```
Error on SCSIRead(), #5
pos = 0, i = 22, fs = /
alloccgblk: can't find blk in cyl
```

La solution consiste à installer d'abord un système minimal sur une "petite" partition root, démarrer OpenBSD et transférer les fichiers comme souhaité.

Supposons que vous voulez une partition / de 200M. Créez la partition root, créez les autres partitions désirées, utilisez newfs avec les utilitaires Mac OS. Installez etc34.tgz et base34.tgz sur la partition /.

Montez les autres partitions dans un point de montage temporaire et copiez les répertoires que vous voulez à ces partitions comme indiqué ici.

Maintenant, modifiez /etc/fstab, rebootez et défaites le reste des fichiers *.tgz tel que c'est documenté ici.

En suivant cette procédure, vous pouvez choisir n'importe quelle taille pour les partitions autres que /.

# 12.9 - MacPPC

## 12.9.1 - Pourquoi mon pilote bm(4) est si lent ?

Le pilote bm, supportant le composant BMAC utilisé sur certains systèmes MacPPC (y compris les premiers iMacs) a des problèmes en 100Mbps. Il est hautement recommandé de forcer le pilote à 10Mbps en utilisant l'option `"media 10baseT"` dans votre fichier `/etc/hostname.bm0` sinon forcez cette vitesse au niveau de votre hub ou de votre commutateur.

# 12.10 - MVME68k

[aucune information pour le moment]

# 12.11 - MVME88k

[aucune information pour le moment]

# 12.12 - SPARC

[aucune information pour le moment]

# 12.13 - UltraSPARC (sparc64)

## 12.13.1 - Mon système UltraSPARC ne veut pas démarrer à partir de l'image sur disquette

Seul les modèles Ultra 1/1e et Ultra 2 peuvent démarrer *n'importe* quel OS à partir d'une disquette. Utilisez les méthodes d'installation CD-ROM, Miniroot ou démarrage réseau pour effectuer votre installation.

# 12.14 - DEC VAX

[aucune information pour le moment]

[Index de La FAQ] [Section 11 - Optimisation des Performances] [Section 14 - Configuration des Disques]

---


www@openbsd.org

Originally [OpenBSD: faq12.html,v 1.57 ]
$Translation: faq12.html,v 1.25 2004/10/25 12:02:37 saad Exp $
$OpenBSD: faq12.html,v 1.18 2004/10/25 12:38:57 jufi Exp $

[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 labels 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, ensuring that you don't 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. In addition to the obvious, it also shows a trailing period ("."), which indicates this machine is capable of using LBA translation to boot. If the machine were incapable of using LBA translation, the above period would have have been replaced with a semicolon (";"), indicating CHS translation:

```
Using Drive 0, Partition 3;
```

Note that the trailing period or semicolon can be used as an indicator of the "new" OpenBSD MBR, introduced with OpenBSD 3.5.

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:

   ```
   Loading...
   ```

   printing a dot for every file system block it attempts to load. Again, the PBR shows if it is using LBA or CHS to load, if it has to use CHS translation, it displays a message with a semicolon:

   ```
   Loading;...
   ```

   The older (pre v3.5) biosboot(8) showed the message "`reading boot...`.
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 2.06
   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
Loading....                                      <- PBR
probing: pc0 com0 com1 apm mem[636k 190M a20=on] <- /boot
disk: fd0 hd0+
>> OpenBSD/i386 BOOT 2.06
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 3.5 (GENERIC) #34: Mon Mar 29 12:24:55 MST 2004
   ...
```

### 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 block number and offset of `/boot`'s inode 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 pointed to by the inode specified in it, which will almost certainly no longer be the desired second-stage boot loader! Since `/boot` is being read using BIOS calls, old versions of the PBR were sensitive to BIOS disk translation. If you altered the drive's geometry (i.e., took it out of one computer that uses CHS translation and moving it into one that uses LBA translation, or even changed a translation option in your BIOS), it would have *appeared 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 you would have had to run installboot(8) before the system could be rebooted. The new (as of OpenBSD 3.5 and later) PBR is much more tollerant to changes in translation.

As the PBR is very small, its range of error messages is pretty limited, and somewhat cryptic. Most likely messages are:

- **ERR R** -- BIOS returned an error when trying to read a block from the disk. Usually means exactly what it says: your disk wasn't readable.
- **ERR M** -- 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 meaning installboot(8) was run incorrectly, the /boot file was altered, or you have exceeded your BIOS's ability to read a large disk.

Other error messages are detailed in the biosboot(8) manual page 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 supports an individual file system of up to $2^{31}$-1, or 2,147,483,647 sectors, and as each sector is 512 bytes, that's a tiny amount less than 1T.

Of course, the ability of file system and the abilities of particular hardware are two different things. 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, and some older BIOSs will hang when they encounter a modern sized hard disk. 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. The boot process has to utilize (and is thus limited by) the system's boot ROM.

For this reason, the entire /bsd file (the kernel) must be located on the disk within the boot ROM addressable area. This means that on some older i386 systems, the root partition must be completely within the first 504M, but newer computers may have limits of 2G, 8G, 32G, 128G or more. It is worth noting that many relatively new computers which support larger than 128G drives actually have BIOS limitations of booting only from within the first 128G. You can use these systems with large drives, but your root partition must be within the first 128G.

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 "ERR M" or other problems on boot.

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.**

Some non-i386 users think they are immune to this, however most platforms have some kind of boot ROM limitation on disk size. Finding out for sure what the limit is, however, can be difficult.

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 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 the fdisk(8) 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.

## 14.14 - Why does `df(1)` tell me I have over 100% of my disk used?

People are sometimes surprised to find they have *negative* available disk space, or more than 100% of a partition in use, as shown by df(1).

When a partition is created with newfs(8), some of the available space is held in reserve from normal users. This provides a margin of error when you accidently fill the disk, and helps keep disk fragmentation to a minimum. Default for this is 5% of the disk capacity, so if the root user has been carelessly filling the disk, you may see up to 105% of the available capacity in use.

If the 5% value is not appropriate for you, you can change it with the tunefs(8) command.

[FAQ Index] [To Section 12 - Platform-Specific Questions]

---


www@openbsd.org

$OpenBSD: faq14.html,v 1.113 2004/10/20 23:04:26 nick Exp $