Langue : fr [équipes]

de en nl pl pt

Autres Documents

Guide de Mise à Niveau
Suivre -current
Guide de Tests des Ports
Comment Utiliser AnonCVS
Stable
Comment Utiliser CVSup
Pages de manuel
Faire un Rapport de Bogues
Listes de diffusion
Guide de l'Utilisateur PF
FAQ OpenSSH

Fichiers PDF

FAQ OpenBSD
Guide de l'Utilisateur PF

Fichiers texte

FAQ OpenBSD
FAQ PF

Retour à OpenBSD

OpenBSD

Documentation et Questions Fréquemment Posées

Problèmes Fréquemment Rencontrés Mises à jour récentes

Cette FAQ constitue une documentation visant à agrémenter les pages de manuel, ces dernières étant disponibles aussi bien sur le système installé qu'en ligne. La FAQ couvre la version la plus récente d'OpenBSD, actuellement la v3.9. Il y aura vraisemblablement des nouvelles fonctionnalités ou 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 (en anglais) dans le répertoire **`pub/OpenBSD/doc`** des miroirs FTP.

---

7 - Contrôles du clavier et de l'affichage

8 - Questions Générales

9 - Migrer vers OpenBSD

10 - Gestion du Système

12 - Questions Spécifiques Aux Plates-Formes

Guide de l'Utilisateur PF

Problèmes Fréquemment Rencontrés

Mises à jour récentes

- FAQ mise à jour pour OpenBSD 3.9
- FAQ 8 - Puis-je utiliser le langage de programmation "L" sous OpenBSD ? - nouveau
- Guide de Mise à Jour - nouveau
- FAQ 15 - Paquetages et Ports - nouveau
- FAQ 13 - utilisation de Java et Flash - nouveau
- 14.16 - Est-il possible d'accéder aux données présentes sur des systèmes de fichiers autres que FFS ? - nouveau

---

Les mainteneurs de la FAQ sont :
Nick Holland, Joel Night et Steven Mestdagh. Parmi les contributeurs additionnels, on compte 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

$OpenBSD: index.html,v 1.122 2006/09/13 11:40:59 jufi Exp $
"Si vous ne le trouvez pas dans l'index, veuillez regarder en détail le catalogue."
Sears, Roebuck, and Co., Consumer's Guide, 1897

[Index de La FAQ] [Section 2 - Autres sources 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, 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 3.9.

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

OpenBSD 3.9 fonctionne sur les plates-formes suivantes :

- alpha - FTP uniquement
- amd64 - CD bootable
- cats - FTP uniquement
- hp300 - FTP uniquement
- hppa - FTP uniquement
- i386 - CD bootable
- luna88k - FTP uniquement
- mac68k - FTP uniquement

- macppc - CD bootable
- mvme68k - FTP uniquement
- mvme88k - FTP uniquement
- sgi - FTP uniquement
- sparc - CD bootable
- sparc64 - CD bootable
- vax - FTP uniquement
- zaurus - FTP uniquement

CD 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

De temps à autre, on nous demande pourquoi nous supportons autant de machines exotiques. Tout simplement "parce que nous le voulons". Si un nombre suffisant de personnes compétentes (des fois, c'est une seule personne compétente !) souhaitent maintenir le support d'une plate-forme alors celle-ci sera supportée. Il y a des avantages pratiques au support de multiples plates-formes : lorsque de nouvelles plates-formes sont supportées, l'arbre des sources ne contient pas a priori de bogues relatifs à la portabilité ou à la conception du système. Les plates-formes supportées par OpenBSD incluent des processeurs 32 bit et 64 bit, des machines en "big" et "little" endian, ainsi que d'autres de conception différente. Et bien entendu, le support de plates-formes exotiques nous a aidé à produire un code de meilleure qualité pour les plates-formes les plus communes.

1.3 - OpenBSD est-il réellement 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 trop 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.

Des personnes nous demandent parfois si le fait que notre travail soit utilisé dans des produits commerciaux nous dérange. La réponse est la suivante, nous préférons que notre code de qualité soit *largement utilisé* plutôt que des vendeurs de logiciels commerciaux réécrivent des solutions alternatives incompatibles et mal codées, pour répondre à des problématiques déjà résolues. Par exemple, SSH est largement utilisé à cause de la liberté qui lui attrait, et l'est bien plus que si des restrictions avaient été appliquées à l'utilisation du code d'OpenSSH.

Cela ne veut pas dire que nous opposons à des donations financières ou à un support matériel en remerciement. En fait, il est aberrant de voir le peu de support de toute sorte que nous recevons des sociétés dépendant d'OpenBSD pour leurs produits, même si aucune compensation n'est exigée.

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

Les développeurs du projet OpenBSD le supportent 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 ainsi que le temps dépensé à répondre aux questions et à corriger les bugs trouvés par les utilisateurs. Les développeurs OpenBSD ne sont pas riches et même de petites contributions de temps, de matériel ou autres peuvent apporter de grosses différences.

1.4 - Pourquoi utiliserais-je 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 vous sera utile.

- 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.
- OpenBSD tente de minimiser le besoin de personnalisation. Pour la majorité des utilisateurs, OpenBSD "fonctionne tout court" sur leur matériel pour leur application. Non seulement la personnalisation est rarement nécessaire, mais elle est activement déconseillée.

1.5 - Comment puis-je aider OpenBSD ?

Nous sommes redevables aux personnes et organisations qui ont contribuées au projet OpenBSD. Les donateurs sont cités sur 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 aucun de ceux proposés ci-dessous ne vous convient, vous pouvez toujours en proposer d'autres en envoyant un courrier électronique 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 d'éviter la consommation de bande passante lors d'un téléchargement à travers l'Internet. Ce jeu de trois CD peu onéreux inclut 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 les équipements, les connexions réseau et la publication des CD. Presser des CD demande un investissement aux développeurs OpenBSD qu'ils ne sont pas toujours assurés de récupérer. Envoyez 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 convenir des modalités d'expédition.
- 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, proposez 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 - Pour quand la prochaine version d'OpenBSD est-elle prévue ?

L'équipe de développement OpenBSD crée une nouvelle version tous les six mois, avec des dates de mise à disposition fixées à Mai et 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 :

- X.org 6.9.0, l'environnement X Window, avec des correctifs locaux. Les serveurs v3.3 sont aussi inclus avec la plate-forme i386 pour le support de chipsets graphiques additionnels. XFree86 est installé en utilisant les fichiers `x*.tgz` des ensembles d'installation.
- GCC 2.95.3 et 3.3.5. 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. Gcc fait partie de `comp39.tgz` présent dans l'ensemble d'installation.
- Perl 5.8.6, avec des correctifs et des améliorations créés 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 relatives à la sécurité. mod_ssl 2.8.16 et DSO sont aussi inclus.
- OpenSSL 0.9.7g, 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 8.13.4 avec libmilter.
- Le serveur DNS BIND 9.3.1. OpenBSD a implémenté plusieurs améliorations au processus de confinement chroot ainsi que d'autres amélioration sécurité.
- Le navigateur web en mode texte Lynx 2.8.5rel.4. Sont inclus le support HTTPS et des correctifs créés par l'équipe OpenBSD.
- Sudo v1.6.8p9, permettant aux utilisateurs d'exécuter des commandes individuelles avec les privilèges root.
- Ncurses 5.2.
- IPv6 KAME.
- Heimdal 0.7 avec correctifs.
- Arla 0.35.7.
- OpenSSH 4.3.
- gdb 6.3.

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. Vous pourrez obtenir un numéro de version plus récent mais un système moins sûr en retour.

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

1.9 - Quoi de neuf dans OpenBSD 3.9 ?

La liste complète des changements apportés à OpenBSD 3.8 pour créer OpenBSD 3.9 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 3.9 et qui sont familières des versions antérieures :

- **Augmentation significative des archives d'installation** sur presque toutes les plates-formes. Cela a été effectué afin d'améliorer le débogage des crashes car à présent gdb(1) offre de nombreuses informations supplémentaires sans impacter sur la vitesse du code. Cela peut causer des problèmes aux personnes ayant partitionné leur disque en utilisant les valeurs minimums et augmente substantiellement l'espace nécessaire à la compilation du système à partir des sources.
- **ftp-proxy(8) a été réécrit** et fonctionne de façon légèrement différente de l'ancien, ce qui nécessite quelques modifications mineures de vos fichiers pf.conf et inetd.conf comme expliqué en détail dans le Guide de Mise à niveau.
- **La plupart des modules binaires X tiers ne fonctionneront plus.** Cela posera problème si vous utilisez le fichier "mga_hal_drv.o" fourni par MGA nécessaire dans certaines configurations multi affichages. D'autres exemples existent certainement.
- **Mises à jour ipsec.conf(5)** Lorsque les **règles ike** ne définiront pas de paramètres pour le mode principal, ipsecctl(8) choisira le cryptage AES au lieu de 3DES. Afin de continuer à fonctionner avec les machines de versions plus anciennes, ajoutez ces lignes à leur fichier ipsec.conf(5) afin de basculer vers les nouvelles valeurs par défaut :

```
main auth hmac-sha1 enc aes
```

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 paquetages. 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 le *ProduitX* est-il/n'est-il 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" d'utilisation, de distribution et de modification selon nos standards. Un produit doit aussi être stable et sûr -- un numéro de version plus récent 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 dans le monde et dans n'importe quel but.

Une autre considération majeure est les souhaites des développeurs. Les développeurs OpenBSD sont les juges ultimes de ce qui est intégré ou non dans le projet. Le fait qu'une application soit "bien" ne signifie pas forcément que le projet OpenBSD veuille y allouer les ressources pour le maintenir, ou qu'ils soient enjoués par sa présence au sein du projet.

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 ?*
  Sa licence n'est pas libre et il ne peut donc pas être inclus.
- *Pourquoi qmail ou djbdns ne sont pas inclus ?*
  Licence ou, absence de licence : l'incapacité de distribuer une version modifiée de ces logiciels fait qu'ils ne peuvent être inclus.
- *Pourquoi Apache est-il 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 bzip2 n'est pas inclus à la place de gzip ?*
  Les performances sont horribles, et le bénéfice est minimal. L'impact sur les plates-formes plus lentes, comme m68k ou VAX, ne serait pas acceptable.

Dans la plupart des cas, ces sujets ont été discutés de manière plus que 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 paquetages et dans la mesure où votre utilisation est compatible avec la licence de ces produits, personne ne vous en empêchera (ça ne serait pas vraiment synonyme de "liberté" de le faire, 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 sources d'Information OpenBSD]


www@openbsd.org

$OpenBSD: faq1.html,v 1.59 2006/09/07 22:06:56 saad Exp $

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

2 - Autres sources d'Information OpenBSD

Table des matières

2.1 - Pages Web intéressantes

Le site officiel pour le projet OpenBSD se trouve à l'adresse : 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.

OpenBSDsupport.org regroupe de la documentation maintenue par des utilisateurs et de qualité variable, mais qui a le mérite de couvrir des sujets ne faisant pas partie de la présente FAQ ou de toute autre documentation officielle.

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 sur 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 identifiant unique et limité dans le temps, tel que `A56D-70D4-52C3`, encore une fois pour s'assurer que *vous* 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 service 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. Il s'agit d'une liste à faible débit.
- **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.
- **bugs** - Bugs reçus par l'intermédiaire de sendbug(1) et discussions à leur sujet.
- **source-changes** - Liste recevant automatiquement les modifications apportées à l'arbre des sources CVS. Dès qu'un développeur modifie (commit) l'arbre des sources d'OpenBSD, CVS envoie une copie (généralement brève) de son message commit à cette liste.
- **ports** - Discussions à propos de l'arbre des ports OpenBSD.
- **ports-changes** - Liste recevant automatiquement les modifications apportées à l'arbre des sources CVS qui concernent les ports.
- **advocacy** - Discussions sur la promotion d'OpenBSD et les messages hors-sujet dans **misb**..

Avant d'envoyer une question à **misc** ou toute autre liste de diffusion, parcourez les archives, pour y voir les questions fréquemment posées. 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 appelée OpenBSD Newbies.

2.3 - Pages de manuel

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

Pour accéder aux pages du man ainsi qu'au reste de la documentation, assurez-vous d'avoir installé les tarballs `man39.tgz` et `misc39.tgz` de l'ensembles de fichiers.

Voici une liste des pages de 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 de manuel.
- **intro(1)** - introduction aux commandes générales, voir aussi les introductions aux autres sections de 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 disklabel.
- **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)** - exécuter des commandes en root mais sans se loguer en root.
- **mg(1)** - éditeur de texte fonctionnant comme emacs.

Pour des utilisateurs plus avancés

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

Vous pouvez trouver toutes les pages de manuel OpenBSD sur le web à l'adresse http://www.openbsd.org/cgi-bin/man.cgi ainsi que sur votre ordinateur si vous installez l'archive de fichiers `man39.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 de l'éditeur vi. Si vous ne connaissez pas le nom d'une commande ou si `"man commande"` ne trouve pas la page de manuel vous pouvez chercher dans la base de données des pages de 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
```

Le numéro entre parenthèses indique la section de manuel dans laquelle cette page a été trouvée. Dans certains cas, vous pourrez trouver des pages de 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 de 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 de manuel UNIX, il existe une documentation imprimable (inclue dans le jeu de fichiers `misc39.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 présents dans /usr/share/doc/.

Les pages de 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 de manuel.

Il est utile pour bon nombre de personnes de disposer d'une version papier d'une page de manuel. Voici les indications pour obtenir une version imprimable d'une page de manuel.

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

On les trouve dans les sources. Les pages de 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 de manuel sans caractères non imprimables.
Exemple :

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

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

Remarquez que `<file>` doit être le fichier source de la page de manuel (certainement un fichier finissant par un numéro comme `tcpdump.8`). Les versions PostScript des pages de manuel ont une belle apparence. Elles peuvent être imprimées ou visualisées à l'aide d'un programme comme gv (GhostView). GhostView est disponible dans collection de Paquetages. Utilisez les options suivantes de la commande 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 de 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 de 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 de 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 de manuel aux formats PostScript et ASCII. Cette option, `MANPS=yes`, est 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 de 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 de manuel !

Comment afficher les pages de manuel en couleur dans un XTerm ?

Le fichier de configuration par défaut de xterm(1) n'affiche pas les pages de 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 paramètres 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 paramètres. Les lignes applicables aux pages de manuel sont :

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

Ce qui produit des pages de manuel avec une couleur infernale, adaptez-la à votre convenance : pouvons nous nous permettre de 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 de manuel. Consultez les pages de manuel de xterm et xman pour plus d'informations.

Comment écrire ma propre page de manuel ?

Si vous souhaitez écrire votre propre page de manuel pour une application que vous avez écrite, un tutorial 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 réellement 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 de manuel ou du site OpenBSD, utilisez les listes de discussion (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, merci de 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@example.com
To: bugs@openbsd.org
Subject: HELP!!!

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

```
From: smartuser@example.com
To: bugs@openbsd.org
Subject: 3.3-beta panics on a SPARCStation2

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

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

This is the dmesg output:

OpenBSD 3.3-beta (GENERIC) #9: Mon Mar 17 12:37:18 MST 2003
    deraadt@sparc.openbsd.org:/usr/src/sys/arch/sparc/compile/GENERIC
real mem = 67002368
avail mem = 59125760
using 200 buffers containing 3346432 bytes of memory
bootpath: /sbus@1,f8000000/esp@0,800000/sd@1,0
```

```
mainbus0 (root): SUNW,Sun 4/75
cpu0 at mainbus0: CY7C601 @ 40 MHz, TMS390C602A FPU; cache chip bug
- trap page uncached
cpu0: 64K byte write-through, 32 bytes/line, hw flush cache enabled
memreg0 at mainbus0 ioaddr 0xf4000000
clock0 at mainbus0 ioaddr 0xf2000000: mk48t02 (eeprom)
timer0 at mainbus0 ioaddr 0xf3000000 delay constant 17
auxreg0 at mainbus0 ioaddr 0xf7400003
zs0 at mainbus0 ioaddr 0xf1000000 pri 12, softpri 6
zstty0 at zs0 channel 0 (console i/o)
zstty1 at zs0 channel 1
zs1 at mainbus0 ioaddr 0xf0000000 pri 12, softpri 6
zskbd0 at zs1 channel 0: reset timeout
zskbd0: no keyboard
zstty2 at zs1 channel 1: mouse
audioamd0 at mainbus0 ioaddr 0xf7201000 pri 13, softpri 4
audio0 at audioamd0
sbus0 at mainbus0 ioaddr 0xf8000000: clock = 20 MHz
dma0 at sbus0 slot 0 offset 0x400000: rev 1+
esp0 at sbus0 slot 0 offset 0x800000 pri 3: ESP100A, 25MHz, SCSI ID 7
scsibus0 at esp0: 8 targets
sd0 at scsibus0 targ 1 lun 0: <SEAGATE, ST1480 SUN0424, 8628> SCSI2 0/direct fixed
sd0: 411MB, 1476 cyl, 9 head, 63 sec, 512 bytes/sec, 843284 sec total
sd1 at scsibus0 targ 3 lun 0: <COMPAQPC, DCAS-32160, S65A> SCSI2 0/direct fixed
sd1: 2006MB, 8188 cyl, 3 head, 167 sec, 512 bytes/sec, 4110000 sec total
le0 at sbus0 slot 0 offset 0xc00000 pri 5: address 08:00:20:13:10:b9
le0: 16 receive buffers, 4 transmit buffers
cgsix0 at sbus0 slot 1 offset 0x0: SUNW,501-2325, 1152x900, rev 11
wsdisplay0 at cgsix0
wsdisplay0: screen 0 added (std, sun emulation)
fdc0 at mainbus0 ioaddr 0xf7200000 pri 11, softpri 4: chip 82072
fd0 at fdc0 drive 0: 1.44MB 80 cyl, 2 head, 18 sec
root on sd0a
rootdev=0x700 rrootdev=0x1100 rawdev=0x1102


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

panic: pool_get(mclpl): free list modified: magic=78746572; page 0xfaa93000;
 item addr 0xfaa93000
Stopped at      Debugger+0x4:   jmpl            [%o7 + 0x8], %g0
RUN AT LEAST 'trace' AND 'ps' AND INCLUDE OUTPUT WHEN REPORTING THIS PANIC!
DO NOT EVEN BOTHER REPORTING THIS WITHOUT INCLUDING THAT INFORMATION!
ddb> trace
pool_get(0xfaa93000, 0x22, 0x0, 0x1000, 0x102, 0x0) at pool_get+0x2c0
sosend(0x16, 0xf828d800, 0x0, 0xf83b0900, 0x0, 0x0) at sosend+0x608
soo_write(0xfac0bf50, 0xfac0bf70, 0xfac9be28, 0xfab93190, 0xf8078f24, 0x0)
at soo_write+0x18
dofilewritev(0x0, 0xc, 0xfac0bf50, 0xf7fff198, 0x1, 0xfac0bf70) at
dofilewritev+0x12c
sys_writev(0xfac87508, 0xfac9bf28, 0xfac9bf20, 0xf80765c8, 0x1000, 0xfac0bf70)
at sys_writev+0x50
syscall(0x79, 0xfac9bfb0, 0x0, 0x154, 0xfcffffff, 0xf829dea0) at syscall+0x220
slowtrap(0xc, 0xf7fff198, 0x1, 0x154, 0x1, 0xfac87508) at slowtrap+0x1d8
ddb> ps
   PID   PPID   PGRP    UID  S       FLAGS  WAIT       COMMAND
 27765   8819  29550      0  3        0x86  netio      xconsole
  1668  29550  29550      0  3      0x4086  poll       fvwm
 15447  29550  29550      0  3     0x44186  poll       xterm
  8819  29550  29550     35  3      0x4186  poll       xconsole
  1238  29550  29550      0  3      0x4086  poll       xclock
```

```
 29550  25616  29550      0  3      0x4086  pause      sh
  1024  25523  25523      0  3     0x40184  netio      XFree86
*25523  25616  25523     35  2     0x44104             XFree86
 25616  30876  30876      0  3      0x4086  wait       xinit
 30876  16977  30876      0  3      0x4086  pause      sh
 16977      1  16977      0  3      0x4086  ttyin      csh
  5360      1   5360      0  3        0x84  select     cron
 14701      1  14701      0  3     0x40184  select     sendmail
 12617      1  12617      0  3        0x84  select     sshd
 27515      1  27515      0  3       0x184  select     inetd
  1904      1   1904      0  2        0x84             syslogd
  9125      1   9125      0  3        0x84  poll       dhclient
     7      0      0      0  3    0x100204  crypto_wa  crypto
     6      0      0      0  3    0x100204  aiodoned   aiodoned
     5      0      0      0  3    0x100204  syncer     update
     4      0      0      0  3    0x100204  cleaner    cleaner
     3      0      0      0  3    0x100204  reaper     reaper
     2      0      0      0  3    0x100204  pgdaemon   pagedaemon
     1      0      1      0  3      0x4084  wait       init
     0     -1      0      0  3     0x80204  scheduler  swapper

Thank you!
```

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 son système n'est pas défectueux, (3.2 fonctionnait convenablement), comment ce "crash" était causé (en démarrant X), et inclut le détail des commandes "`ps`" et "`trace`" afin de permettre aux développeurs de déboguer. Dans ce cas, Smart User a fourni ces détails capturés via une console serie, si vous ne savez pas le faire, vous pouvez utiliser une feuille et un stylo afin de retranscrire le "crash". (L'exemple ci-dessus était un problème réel qui affectait les systèmes sun4c, et les informations fournies dans le précédant rapport ont aidé 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 utilisez-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 pourrez être contacté par les développeurs pour fournir de plus amples informations ou tester des correctifs. Vous pouvez également consulter les archives de la liste de diffusion `bugs@openbsd.org`, de plus amples informations se trouvant sur la page des listes de discussion, ou consulter 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" ?
Dans certaines conditions, vous pouvez perdre le tout premier message d'une panique système, et il s'agit de celui 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 "show panic" dans ddb> comme suit :

```
ddb> show panic
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

$OpenBSD: faq2.html,v 1.45 2006/09/07 22:06:56 saad 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 jeu de CD OpenBSD

Acheter un jeu de 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 jeu de CD OpenBSD :

- La vente des jeux CDs finance le développement d'OpenBSD.
- 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 jeu de CD a un véritable impact sur le développement futur.
- Les CDs contiennent les binaires (et les sources) pour toutes les plates-formes supportées.
- Les CDs sont "bootables" sur plusieurs plates-formes et peuvent être utilisés pour démarrer une machine sans système pré-installé.
- Les CD sont utiles pour démarrer un système même si vous choisissez d'installer un "snapshot".
- Installer depuis les CDs est plus rapide ! De plus cela permet de préserver la bande passante.
- Les jeux de 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 un jeu de CD ou donnez du matériel.
- Les jeux de CDs OpenBSD contiennent un assortiment de packages populaires et utiles. Ceux-ci sont suffisamment complets pour installer un environnement de travail et de développement sans aucune connexion réseau.

Pour installer une Révision (Release) d'OpenBSD, vous devriez utiliser un jeu de CD officiel.

3.2 - Acheter des T-shirts OpenBSD

OpenBSD propose des T-shirts pour votre plaisir vestimentaire. Vous pouvez les visualiser sur la page : 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 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. Pour inciter les gens à acheter les CDs, plusieurs extras sont aussi inclus 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 sûr, toute image ISO d'OpenBSD disponible sur l'Internet l'est 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 aux personnes souhaitant télécharger OpenBSD gratuitement d'utiliser 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 `cd39.iso`) sont disponibles pour un certain nombre de plates-formes et permettront d'installer le système par FTP. Ces images ISO ont une taille de quelques mégaoctets 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 vous 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 sûr votre CD d'OpenBSD est toujours plus proche qu'un site miroir. Les informations sont disponibles ici :

Page FTP OpenBSD.

3.4 - Obtenir le code source actuel

Le code source d'OpenBSD est disponible librement et gratuitement. La meilleure façon de l'obtenir 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 ici :

AnonCVS OpenBSD.

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

$OpenBSD: faq3.html,v 1.34 2006/09/07 22:06:56 saad Exp $

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

4 - Guide d'Installation d'OpenBSD 3.9

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`).

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 à partir d'une source autre que ce disque RAM préchargé. 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 `floppy39.fs`, bien que plusieurs architectures possèdent de multiples images de disquettes.
- **CD-ROM :** Sur plusieurs architectures une image CD-ROM (`cd39.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 du disque RAM peut être démarré à partir d'une partition OpenBSD existante (i.e., une mise à jour ou une réinstallation).
- **Réseau:** Quelques architectures supportent le démarrage à travers un réseau (en utilisant par exemple PXE ou d'autres méthodes de démarrage 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 amorçables :** 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 :** Booté 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 paquetages 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 paquetages.
- **HTTP :** L'un des différents sites miroirs HTTP OpenBSD ou votre serveur web local contenant les paquetages.
- **Partition de disque local :** Dans la plupart des cas, vous pouvez installer les paquetages 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 au 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 paquetages.
- **Cassette :** Les paquetages peuvent aussi être lus depuis une cassette formatée. Ces cassettes peuvent être créées en suivant les instructions du fichier `INSTALL.`*platform*.

4.2 - Vérifications avant l'installation

Avant de commencer votre installation, vous devriez avoir une idée de ce que vous allez 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 corrigé ?
  - 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 souhaitez 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 six images de disques séparées parmi lesquelles choisir :

- **`floppy39.fs`** (PC de bureau) supporte la plupart des cartes PCI et ISA, des adaptateurs IDE et des adaptateurs SCSI simples ainsi que quelques matériels PCMCIA. La plupart des utilisateurs utiliseront cette image.
- **`floppyB39.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.
- **`floppyC39.fs`** (Portables) supporte les matériels CardBus et PCMCIA trouvés sur la plupart des portables.
- **`cdrom39.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'amorce de CD inscriptibles personnalisés.
- **`cd39.iso`** est une image ISO9660 qui peut être utilisée pour créer un CD amorçable à l'aide de la plupart des logiciels de création de CD-ROM sur la plupart des architectures.
- **`cdemu39.iso`** est une image ISO9660 se servant de l'émulation de disquette pour démarrer. Elle utilise pour cela l'image `cdrom39.fs` de 2.88M. Il faut espérer que peu de personnes auront à utiliser cette image -- la plupart utiliseront `cd39.iso`, n'utilisez `cdemu39.iso` que si `cd39.iso` ne fonctionne pas pour vous.

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 parmi lesquelles choisir :

- **`floppy39.fs:`** Supporte les systèmes équipés d'un lecteur de disquettes.
- **`cd39.iso`** Une image ISO utilisable pour créer votre propre CD afin d'amorcer les systèmes SPARC à l'aide un CD-ROM.
- **`miniroot39.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 VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV 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 possèdent des commandes différentes pour le formatage des disquettes. Référez-vous à votre documentation système pour connaître la procédure exacte.

Une fois que vous avez une disquette propre et formatée, il est temps 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=floppy39.fs of=/dev/rfd0c bs=32k
```

Une fois l'image écrite, vérifiez qu'elle a 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 apparaître qu'une nouvelle invite de commande.

```
# cmp /dev/rfd0c floppy39.fs
```

4.3.2 - Créer des disquettes de démarrage sur Windows ou DOS

Cette section décrit comment écrire 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 `3.9/tools` du CD1 du lot de CD-ROMS OpenBSD.

Pour préparer la disquette sous MS-DOS ou Windows, utilisez tout d'abord les utilitaires natifs pour formater 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 nommage de fichiers MS-DOS "8.3". Comme `floppyB39.fs` et `floppyC39.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: floppy39.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 floppy39.fs a:
```

Exemple d'utilisation de *ntrw* :

```
C:\> ntrw floppy39.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 `cd39.iso`, ou dans le cas des architectures i386 et amd64, vous pouvez aussi utiliser `cdrom39.fs` comme image de disquette amorçable pour démarrer un système i386 depuis le CD-ROM. On laissera ici au lecteur le soin de déterminer lui-même les détails en fonction des 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 et de Paquetages OpenBSD.

4.4 - Démarrer le média d'installation OpenBSD

Démarrer sur i386/amd64

Démarrer une image d'installation sur les architectures PC i386 et amd64 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 aller 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 la procédure 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 `boot cdrom` :

```
ok boot cdrom
```

Bien sûr, cela ne fonctionnera que dans le nouveau mode de commande. Si vous possédez l'ancien mode de commande (une flèche 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

Quelle que 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 avoir à redémarrer en lançant `install` à l'invite shell.

Si votre démarrage réussi, vous verrez beaucoup de messages défiler. Ces textes, dans beaucoup d'architectures en blanc sur bleu, représentent le dmesg, du noyau qui cite les matériels trouvés et leur emplacement. Ne vous souciez pas de retenir ce message, une copie est conservée dans `/var/run/dmesg.boot`.

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 paquetage 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 remarques importantes :
  - Vous n'aurez pas la possibilité d'installer le paquetage `etc39.tgz`. Après l'installation, vous devrez >manuellement greffer `etc39.tgz` sur votre système avant qu'il puisse être complètement fonctionnel. C'est une étape importante qui doit être réalisée, car dans le cas contraire certains services clés (tel que pf(4)) ne démarreront pas.
  - La procédure de mise à jour n'est pas faite pour sauter des "releases" ! Bien que ce saut fonctionnera souvent, il n'est pas supporté. Pour OpenBSD 3.9, mettre à jour de la version 3.8 à la version 3.9 est la seule mise à jour supportée. Si vous mettez à jour depuis une version plus vieille, il est fortement recommandé de mettre à jour en versions intermédiaires d'abord. Cependant, si votre version est très ancienne, une réinstallation complète est recommandée.

  Plus d'informations concernant la mise à jour d'une version à une autre se trouvent ici.
- **Shell :** Parfois, vous devez réaliser des réparations ou de la maintenance sur un système qui ne pourra pas (ou ne devra pas) démarrer un noyau normal. Cette option vous permettra cette maintenance sur le système. Un certain nombre d'utilitaires importants sont présents sur le media de démarrage.

Parfois, vous ne verrez pas l'option "Upgrade" listée. Après un *"jour drapeau"*, 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 3.9 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
kbd(8) mapping? ('L' for list) [none] 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 vous ne 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 processus d'installation va se terminer et vous envoyer sur une invite de commande shell.

Les notes d'installation auxquelles nous nous référons ici sont présentes sur les CDs d'installation ainsi que les serveurs FTP dans le fichier `INSTALL.<plat>`, où <*plat*> est votre plate-forme, par exemple pour `i386`.

4.5.2 - Configurer les disques

Configurer les disques sur OpenBSD varie un peu en fonction des architectures. Pour i386, amd64, macppc, zaurus et cats, 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 sous OpenBSD pour les architectures citées précédemment, 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 supportant cette disposition. 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] Entrée
```

Le disque "root" est le disque depuis lequel le système démarrera et normalement sur lequel l'espace "swap" réside. Les disques IDE vont être désignés comme `wd0`, `wd1`, etc., tandis que les disques SCSI et les matériels RAID apparaîtront 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] Entrée
```

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, configurée avec le type de partition OpenBSD, et indiquée 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, certains 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, quels que soient leurs types, que vous ne voulez pas effacer, ne répondez pas "yes" à la question précédente. En revanche, si votre système possède un disque tout neuf qui n'a jamais été utilisé, vous devrez probablement répondre "yes" (ou utiliser l'option "update" de fdisk) afin d'inscrire un enregistrement et une signature de boot valide.

Dans le cas de cet exemple, nous considérerons 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 grande que la capacité généralement disponible lorsque votre machine a été fabriquée) voudront sûrement 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
*0: 0B    0   1  1 -  202 239 63 [          63:     3069297 ] Win95 FAT-32
 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
        swap            Swap two partition entries
        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 existante, crée une grande partition OpenBSD, la définit 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.
- **`u`** or **`update :`** Met à jour le MBR avec le code de boot d'OpenBSD, similaire à "`reinit`", si ce n'est que cela ne modifie pas la table des partitions existante.
- **`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 entraînera 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 a 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 que le secteur de début d'OpenBSD est 3069360 en additionnant le secteur de début de la partition existante (63) et sa taille (3069297).

Vous pouvez éditer la disposition du disque en utilisant soit la notation Cylindre/Tête/Secteur, soit la notation 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] Entrée
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
 *0: 0B    0   1  1 -  202 239 63 [          63:        1499M] Win95 FAT-32
 *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
 0: 0B    0   1  1 -  202 239 63 [          63:     3069297 ] Win95 FAT-32
 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>
```

Sur les plates-formes utilisant fdisk, Il est important que la première partition saute la première piste du disque, *dans notre cas*, qu'elle commence au secteur `63`. Ceci varie d'une machine à une autre et d'un disque à un autre. 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 la partition active 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:
        ? [cmnd]  - this message or command specific help.
        a [part]  - add new partition.
        b         - set OpenBSD disk boundaries.
        c [part]  - change partition size.
        D         - set label to default.
        d [part]  - delete partition.
        e         - edit drive parameters.
        g [b|d|u] - use [b]ios, [d]isk or [u]ser geometry.
        M         - show entire OpenBSD man page for disklabel.
        m [part]  - modify existing partition.
        n [part]  - set the mount point for a partition.
        p [unit]  - print label.
        q         - quit and save changes.
        r         - recalculate free space.
        s [path]  - save label to file.
        u         - undo last change.
        w         - write label to disk.
        X         - toggle expert mode.
        x         - exit without saving changes.
        z         - zero out partition table.

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).
        '%' for percent of total disk size, '&' for percent of free space.
        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 a pas de réponse universelle. Si vous n'êtes pas sûr de la façon dont vous voulez partitionner votre système, regardez cette discussion.

Sur 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 disposer de quelques centaines de mega-octets en trop plutôt que d'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 utilisée 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` est utilisé pour construire certains logiciels, 120Mo sera probablement suffisant pour la plupart des cas de figure.
- ***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*** - 6Go. Nous voulons que cette partition soit assez grande pour charger quelques applications utilisateurs, et en plus être capable de mettre à jour et reconstruire le système si nécessaire. La Racine des "Ports" sera très bien ici, occupant approximativement 160Mo avant la construction des "ports". Si vous prévoyez la compilation de nombreuses applications à partir des sources, en vous servant des ports plutôt que des paquetages pré-compilés vous aurez sûrement besoin de plus de place ici.
- ***wd0h: /home*** - 4Go. Cela permettra de stocker beaucoup de fichiers utilisateurs.

Maintenant, si vous additionnez tout cela, vous verrez que plus de 6Go 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* ? créez une nouvelle partition dans l'espace inutilisé, formatez la avec newfs(8), et changez */etc/fstab* pour monter la partition dans */tmp*. Problème 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] Entrée
size: [36030960] 150m
Rounding to nearest cylinder: 307440
FS type: [4.2BSD] Entrée
mount point: [none] /
> a b
8ffset: [3376800] Entrée
size: [35723520] 300m
Rounding to nearest cylinder: 614880
FS type: [swap] Entrée
> a d
offset: [3991680] Entrée
size: [35108640] 120m
Rounding to nearest cylinder: 245952
FS type: [4.2BSD] Entrée
mount point: [none] /tmp
```

```
> a e
offset: [4237632] Entrée
size: [34862688] 80m
Rounding to nearest cylinder: 164304
FS type: [4.2BSD] Entrée
mount point: [none] /var
> a g
offset: [4401936] Entrée
size: [34698384] 6g
Rounding to nearest cylinder: 12582864
FS type: [4.2BSD] Entrée
mount point: [none] /usr
> a h
offset: [16984800]] Entrée
size: [22115520] 4g
Rounding to nearest cylinder: 8388576
FS type: [4.2BSD] Entrée
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   2048 16384   16 # /
  b:        300.2M       1648.8M    swap
  c:      19092.9M          0.0M  unused      0     0
  d:        120.1M       1949.1M  4.2BSD   2048 16384   16 # /tmp
  e:         80.2M       2069.2M  4.2BSD   2048 16384   16 # /var
  g:       6144.0M       2149.4M  4.2BSD   2048 16384   16 # /usr
  h:       4096.0M       8293.4M  4.2BSD   2048 16384   16 # /home
  i:       1498.7M          0.0M   MSDOS
> q
Write new label?: [y] Entrée
```

Vous noterez qu'il y a une partition *c* que nous avons vraisemblablement 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 valeur 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 les spécifier maintenant.

```
  Mount point for wd0d (size=122976k)? (or 'none' or 'done') [/tmp] Entrée
  Mount point for wd0e (size=82152k)? (or 'none' or 'done') [/var] Entrée
  Mount point for wd0g (size=4194288k)? (or 'none' or 'done') [/usr] Entrée
  Mount point for wd0h (size=4194288k)? (or 'none' or 'done') [/home] Entrée
  Mount point for wd0d (size=122976k)? (or 'none' or 'done') [/tmp] done
  No more disks to initialize.

  OpenBSD filesystems:
  wd0a /
  wd0d /tmp
  wd0e /var
  wd0g /usr
  wd0h /home

  The next step *DESTROYS* all existing data on these partitions!
  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 1 cyl groups (306 c/g, 150.61MB/g, 19328 i/g)
  /dev/rwd0d:     245952 sectors in 244 cylinders of 16 tracks, 63 sectors
          120.1MB in 1 cyl groups (244 c/g, 120.09MB/g, 15360 i/g)
  /dev/rwd0e:     164304 sectors in 163 cylinders of 16 tracks, 63 sectors
          80.2MB in 1 cyl groups (164 c/g, 80.72MB/g, 10368 i/g)
  /dev/rwd0g:     12582864 sectors in 12483 cylinders of 16 tracks, 63 sectors
          6144.0MB in 39 cyl groups (328 c/g, 161.44MB/g, 20608 i/g)
  /dev/rwd0h:     8388576 sectors in 8322 cylinders of 16 tracks, 63 sectors
          4096.0MB in 26 cyl groups (328 c/g, 161.44MB/g, 20608 i/g)
  /dev/wd0a on /mnt type ffs (rw, asynchronous, local, ctime=Sat Oct  7
19:49:44 2
  006)
  /dev/wd0h on /mnt/home type ffs (rw, asynchronous, local, nodev,
nosuid, ctime=S
  at Oct  7 19:49:44 2 006)
  /dev/wd0d on /mnt/tmp type ffs (rw, asynchronous, local, nodev,
nosuid, ctime=Sa
  t Oct  7 19:49:44 2006)
  /dev/wd0g on /mnt/usr type ffs (rw, asynchronous, local, nodev,
ctime=Sat Oct  7
   19:49:44 2006)
  /dev/wd0e on /mnt/var type ffs (rw, asynchronous, local, nodev,
nosuid, ctime=Sa
  t Oct  7 19:49:44 2006)
```

Vous devez vous demander pourquoi la procédure d'installation vous demande encore vos points de montage. Cela permet d'éviter des erreurs ou des omissions sur les points de montage spécifiés pendant la création du partitionnement. Par exemple, la procédure d'installation va automatiquement supprimer les points de montage identiques que vous avez entrés lors du partitionnement. Le programme "disklabel" vous autorisera de telles erreurs, ces points de montage doivent donc être vérifiés après l'exécution du programme "disklabel". Ceux qui sont en doubles et qui ont été supprimés deviendront 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 de formatage de 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.

Durant cette étape tous les systèmes de fichiers sont formatés. Cela peut prendre un certain temps, en fonction 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 les clés cryptographiques nécessaires au système seront 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 passer à la configuration de votre réseau. Le réseau doit être configuré si vous voulez procéder à une installation par FTP ou NFS car celle-ci sera basée sur les informations que vous aller entrer. Voici un aperçu de la section de configuration réseau de la procédure d'installation. Dans notre exemple, nous connecterons une interface (fxp0) à un modem câble, configurée à l'aide de DHCP. L'autre interface sera connectée à notre réseau interne et configurée statiquement.

```
Configure the network? [yes] Enter
Available interfaces are: fxp0 xl0.
Which one do you wish to initialize? (or 'done') [fxp0] xl0
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
IPv4 address for fxp0? (or 'dhcp') 199.185.137.55
Netmask? [255.255.255.0] Enter
IPv6 address for fxp0? (or 'rtsol' or 'none') [none] Enter
Available interfaces are: fxp0.
Which one do you wish to initialize? (or 'done') [fxp0] Enter
Symbolic (host) name for fxp0? [puffy] Enter
The media options for fxp0 are currently
        media: Ethernet autoselect (10baseT)
Do you want to change the media options? [no] Enter
IPv4 address for fxp0? (or 'none' or 'dhcp') dhcp
Issuing hostname-associated DHCP request for fxp0.
DHCPDISCOVER on fxp0 to 255.255.255.255 port 67 interval 1
DHCPOFFER from 73.34.136.1
DHCPREQUEST on fxp0 to 255.255.255.255 port 67
DHCPACK from 73.34.136.1
bound to 69.241.244.76 -- renewal in 1800 seconds.
IPv6 address for fxp0? (or 'rtsol' or 'none') [none] Enter
No more interfaces to initialize.
DNS domain name? (e.g. 'bar.com') [my.domain] example.com
DNS nameserver? (IP address or 'none') [68.87.77.130 68.87.72.130] Enter
Use the nameserver now? [yes] Enter
Default route? (IP address, 'dhcp' or 'none') [dhcp] Enter
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 votre réseau configuré, le script d'installation vous donne la possibilité de faire des ajustements manuels à la configuration.

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

```
Let's install the sets!
Location of sets? (cd disk ftp http or 'done') [cd] Entrée
Available CD-ROMs are: cd0.
```

Dans cet exemple nous installons depuis le CD-ROM. Une liste des matériels sur notre machine identifiés comme lecteurs de CD-ROM nous sera proposée. 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.

REMARQUE : Les options d'installation ne sont pas toutes disponibles sur toutes les plates-formes. Dans le cas présenté, la plate-forme OpenBSD/i386 ne supporte pas les installations via NFS c'est pour cela que cette option n'est pas affichée dans la liste d'options d'installation.

```
Available CD-ROMs are: cd0.
Which one contains the install media? (or 'done') [cd0] Entrée
Pathname to the sets?  (or 'done') [3.9/i386] Entrée
```

Ici, on vous demande dans quel dossier se trouvent les fichiers d'installation, celui-ci est `3.9/i386/` sur les CD-ROM officiels.

4.5.6 - Choisir les sets de fichiers

Il est maintenant temps de choisir quels sets de fichiers vous aller installer. Vous pouvez avoir une description de ces sets dans la section suivante. Les fichiers que le programme d'installation détecte sont montrés à l'écran. Votre travail consiste simplement à spécifier ceux que vous souhaitez installer. Par défaut toutes les archives ne concernant pas X sont sélectionnées; certains utilisateurs expérimentés peuvent préférer se limiter au strict minimum requis pour démarrer OpenBSD, qui devrait se résumer à `base39.tgz`, `etc39.tgz` et `bsd`. La plupart des utilisateurs voudront installer toutes les archives. L'exemple suivant illustre une installation complète.

```
Select sets by entering a set name, a file name pattern or 'all'. De-select
sets by prepending a '-' to the set name, file name pattern or 'all'. Selected
sets are labeled '[x]'.

        [X] bsd
        [X] bsd.rd
        [ ] bsd.mp
        [X] base39.tgz
        [X] etc39.tgz
        [X] misc39.tgz
        [X] comp39.tgz
        [X] man39.tgz
        [X] game39.tgz
        [ ] xbase39.tgz
        [ ] xetc39.tgz
        [ ] xshare39.tgz
        [ ] xfont39.tgz
        [ ] xserv39.tgz
Set name? (or 'done') [bsd.mp] all

        [X] bsd
        [X] bsd.rd
        [X] bsd.mp
        [X] base39.tgz
        [X] etc39.tgz
        [X] misc39.tgz
        [X] comp39.tgz
        [X] man39.tgz
        [X] game39.tgz
        [X] xbase39.tgz
        [X] xetc39.tgz
        [X] xshare39.tgz
        [X] xfont39.tgz
        [X] xserv39.tgz
```

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

Une fois que vous avez choisi les paquetages 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 énormément en fonction du système sur lequel vous installez OpenBSD, les paquetages que vous avez sélectionnés et la vitesse du média source. Cela va de quelques minutes à plusieurs heures.

```
Set name? (or 'done') [done] Entrée
Ready to install sets? [yes] Entrée
Getting bsd ...
100% |**************************************************|  5332 KB    00:07
Getting bsd.rd ...
100% |**************************************************|  4622 KB    00:03
Getting bsd.mp ...
100% |**************************************************|  5374 KB    00:04
Getting base39.tgz ...
100% |**************************************************| 39523 KB    00:32
Getting etc39.tgz ...
100% |**************************************************|  1126 KB    00:01
Getting misc39.tgz ...
100% |**************************************************|  2222 KB    00:02
Getting comp39.tgz ...
100% |**************************************************| 73524 KB    01:03
Getting man39.tgz ...
100% |**************************************************|  7258 KB    00:06
Getting game39.tgz ...
100% |**************************************************|  2538 KB    00:02
Getting xbase39.tgz ...
100% |**************************************************| 10313 KB    00:09
Getting xetc39.tgz ...
100% |**************************************************| 90387       00:00
Getting xshare39.tgz ...
100% |**************************************************|  2028 KB    00:02
Getting xfont39.tgz ...
100% |**************************************************| 32457 KB    00:28
Getting xserv39.tgz ...
100% |**************************************************| 19410 KB    00:26
Location of sets? (cd disk ftp http or 'done') [done] Entrée
```

A ce point, vous pouvez spécifier des fichiers additionnels venant d'autres sources (incluant les paquetages personnalisés) si vous le souhaitez, ou taper '`done`' si vous avez installé tous les paquetages 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. Dans la plupart des cas, vous souhaiterez que sshd(8) soit lancé, mais dans de rares cas non. Si votre configuration n'a aucune nécessité de sshd(8), il y a un petit avantage de sécurité théorique à ne pas le lancer.

```
        Start sshd(8) by default? [yes] y
```

(Si vous changez d'avis plus tard, modifiez `/etc/rc.conf.local` ou `/etc/rc.conf`.)

Vous avez à présent la possibilité de lancer OpenNTPD au boot. OpenNTPD est un moyen peu coûteux permettant de garder une horloge système synchronisée, et la configuration par défaut est suffisante dans la plupart des cas.

```
        Start ntpd(8) by default? [no] y
```

(Si vous changez d'avis plus tard, modifiez `/etc/rc.conf.local` ou `/etc/rc.conf`.)

Sur certaines plates-formes, 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`, en fonction de votre architecture. Sous certaines architectures cette question ne sera pas posée du tout. Si vous ne comptez pas utiliser X sur ce système ou si vous n'êtes pas sûr, répondez 'N' vu que vous pouvez facilement changer la valeur ultérieurement en éditant `/etc/sysctl.conf` au besoin. Il existe un avantage potentiel à laissé le pilote d'aperture xf86(4) désactivé car le moteur graphique d'une carte vidéo moderne pourrait être utilisé pour altérer la mémoire, passant outre le contrôle du processeur.

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

Ensuite, on vous demande si vous souhaitez vous servir d'une console série, plutôt que d'un clavier et un écran standards, sur cet ordinateur. Si vous choisissez "`yes`" et répondez à deux autres questions, `/etc/boot.conf` et `/etc/ttys` seront édités de manière appropriée pour vous. La plupart des utilisateurs choisiront la réponse par défaut, c'est à dire **no**.

```
Change the default console to com0? [no] Entrée
```

Votre dernière tâche est de paramétrer votre fuseau horaire ("timezone"). Selon 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) [Canada/Mountain] ?
Africa/      Chile/       GB-Eire      Israel       NZ-CHAT      UCT
America/     Cuba         GMT          Jamaica      Navajo       US/
Antarctica/  EET          GMT+0        Japan        PRC          UTC
Arctic/      EST          GMT-0        Kwajalein    PST8PDT      Universal
Asia/        EST5EDT      GMT0         Libya        Pacific/     W-SU
Atlantic/    Egypt        Greenwich    MET          Poland       WET
Australia/   Eire         HST          MST          Portugal     Zulu
Brazil/      Etc/         Hongkong     MST7MDT      ROC          posix/
CET          Europe/      Iceland      Mexico/      ROK          posixrules
CST6CDT      Factory      Indian/      Mideast/     Singapore    right/
Canada/      GB           Iran         NZ           Turkey       zone.tab
What timezone are you in? ('?' for list) [Canada/Mountain] 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 dernières étapes sont pour le système de créer le répertoire `/dev` (ce qui pourrait prendre un peu de temps sur certains systèmes, surtout si vous avez peu de RAM), et d'installer les "boot blocks".

```
Making all device nodes...done.
Installing boot block...
boot: /mnt/boot
proto: /usr/mdec/biosboot
device: /dev/rwd0c
/usr/mdec/biosboot: entry point 0
proto bootblock size 512
/mnt/boot is 3 blocks x 16384 bytes
fs block shift 2; part offset 3069360; inode block 152, offset 4136
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 pourraient vous concerner.

Une astuce pouvant être utilisée pour une configuration "d'avant le premier démarrage" consiste à lancer :

```
# /mnt/usr/sbin/chroot /mnt
```

au prompt du shell. Ceci règlera vos points de montage comme ils le seraient après un redémarrage normal du système fraîchement installé. Vous pouvez à présent réaliser des tâches de configuration basique du système, comme l'ajout d'utilisateurs, le changement des points de montage, etc.

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 Paquetages OpenBSD pour installer des logiciels.

Une dernière chose...

Les développeurs OpenBSD vous demandent de leur Envoyer une copie de votre "dmesg". Celui-ci leur est vraiment utile, et bénéficie 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 *paquetages de fichiers* séparés. Toutes les utilisations ne requièrent pas tous les paquetages. Voici une vue d'ensemble de chacun :

- ***bsd*** - Ceci est le noyau. **Requis**
- ***bsd.mp*** - Noyau pour les systèmes multiprocesseurs (SMP) (seulement sur certaines plates-formes)
- ***bsd.rd*** - Kernel de disque RAM
- ***base39.tgz*** - Contient le système de base OpenBSD **Requis**
- ***etc39.tgz*** - Contient tous les fichiers de /etc **Requis**
- ***comp39.tgz*** - Contient le compilateur et ses outils, en-têtes et bibliothèques **Recommandé**
- ***man39.tgz*** - Contient les pages de manuels **Recommandé**
- ***misc39.tgz*** - Contient différentes informations et documentations de configuration
- ***game39.tgz*** - Contient les jeux pour OpenBSD
- ***xbase39.tgz*** - Contient les fichiers de base pour X11
- ***xetc39.tgz*** - Contient les fichiers de configuration /etc/X11 et /etc/fonts
- ***xfont39.tgz*** - Contient le serveur de fontes X11 et les fontes
- ***xserv39.tgz*** - Contient les serveurs X de X11
- ***xshare39.tgz*** - Contient les pages de manuels, les options de localisations, les inclusions, etc. pour X

Les paquetages **etc39.tgz** et **xetc39.tgz** ne sont pas installés lors d'une mise à niveau 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.

Même si vous n'avez pas l'intention de faire tourner X, certains paquetages nécessitent que les bibliothèques graphiques présentes dans **xbase39.tgz** soient installées sur votre système. Cette dépendance peut facilement être résolue en installant **xbase39.tgz**, le reste de X n'étant pas nécessaire.

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

Evidemment, la réponse à cette question dépend de l'utilisation que vous voulez faire du système. Cependant, ces nombres peuvent être utilisés comme un point de *départ* :

```
(root)          60MB
/usr            420MB (no X) ou 550MB (avec X)
/var            25MB
/tmp            50MB
swap            32MB
```

Ce sont les tailles de systèmes de fichiers pour une installation complète. Ces nombres incluent un peu d'espace supplémentaire, et correspondent à une installation typique d'une machine connectée à Internet et à usage personnel.

Gardez les choses suivantes à l'esprit :

- Ces valeurs sont les valeurs minimales. L'espace disque est relativement bon marché aujourd'hui et essayer de réduire votre système afin de le faire tenir sur le plus petit disque possible ne vaut généralement pas la peine. Pour les applications spécifiques les valeurs précédentes peuvent être rabaissées mais vous aurez besoin d'expérimentation.

- Ces valeurs n'incluent PAS l'arbre des ports
- Si vous voulez installer un certain nombre de logiciels tiers, faites-vous une grosse partition `/usr` ! Sa taille dépend des applications que vous comptez installer.
- 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 reconstruire le noyau ou le système depuis les sources, vous devrez avoir une partition `/usr` volumineuse. 4G est une taille correcte.
- La compilation de ports depuis les sources pourrait nécessiter beaucoup d'espace sur les partitions `/usr` et `/tmp`. C'est une des raisons pour lesquelles nous conseillons d'utiliser des paquetages pré-compilés.
- La partition `/tmp` est entre autres utilisée lors de la compilation de ports, et la taille à lui donner dépend de l'utilisation que vous voulez en faire. 50Mo devraient être suffisants pour la plupart des gens, mais quelques grosses applications peuvent nécessiter 100Mo ou plus d'espace dans `/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 disponible, réservez au minimum 64Mo. Si vous avez beaucoup de place disponible, faite une partition de 256Mo, ou même 512Mo. D'un autre côté, si vous utilisez un périphérique de stockage flash, vous souhaiterez probablement ne pas avoir de partition swap du tout. Beaucoup de personnes suivent une vieille règle stipulant que la partition de swap doit être d'une taille double à celle de la quantité de RAM. Cette règle n'a aucun sens. Sur un système moderne, il y a beaucoup de swap, la plupart des gens ne veulent pas que leur système swap. Utilisez ce qui correspond à vos besoins.
- Les espaces swap et `/var` sont utilisés pour stocker les "core dumps" du système lors 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. Lors du redémarrage, savecore(8) essaiera 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 suffisamment d'*espace disponible* pour enregistrer ces fichiers de copie mémoire. Soyez réaliste -- peu de développeurs voudront jeter un oeil à votre fichier de dump de 1GB, et il est peu probable que le dump serve à quiconque si vous ne prévoyez pas de le faire vous-même.

Il y a plusieurs bonnes raisons pour utiliser des systèmes de fichiers séparés plutôt que de n'en utiliser qu'un ou deux pour tout stocker :

- **Sécurité** : Vous pouvez marquer certains systèmes de fichiers d'un 'nosuid', 'nodev', 'noexec', 'readonly', etc. Cela est 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'il a 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 écrit souvent peut devenir quelque peu fragmenté. (Par chance, le système de fichiers ffs utilisé par OpenBSD 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** : Beaucoup 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 comprise dans 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`. Cependant, vous ne devriez probablement pas mettre votre première installation OpenBSD en production.
- Un système exposé à l'Internet ou à 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 supprimés peut être plus rapide à reformater puis re-créér que de supprimer les fichiers. Regardez la FAQ de la compilation du code source pour un exemple (`/usr/obj`).
- Si vous souhaitez reconstruire votre système depuis les sources pour quelque raison que ce soit, les sources seront dans `/usr/src`. Si vous ne faites pas une partition séparée pour `/usr/src`, soyez sûr que `/usr` est suffisamment grand.
- 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 portion du disque non allouée. Si vous dépassez la taille d'une partition, vous pouvez allouer une nouvelle partition dans l'espace non utilisé, dupliquer votre partition existante sur la nouvelle, changer /etc/fstab pour pointer vers votre nouvelle partition, la remonter, vous disposez maintenant de 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 créez des partitions très larges, il est à signaler que la vérification des systèmes de fichiers à l'aide de fsck(8) nécessite approximativement 1M de RAM par gigaoctet, et peut être très consommatrice en temps voire infaisable sur des systèmes anciens et lents (merci de consulter cette section.
- Si vous autorisez les utilisateurs à écrire dans `/var/www` (ex., pages web personnelles), vous devriez le placer dans une partition séparée, vous pouvez par exemple utiliser les quotas pour restreindre la place qu'ils utiliseront, de manière à ce que 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 même ordinateur, et de pouvoir choisir depuis lequel vous souhaitez démarrer. Ce n'est *pas* une tâche triviale ! Si vous ne comprenez 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 ceci ; 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 peu fréquemment (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 lit `/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 ralentit 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 utiliser NTLDR, le chargeur de démarrage que NT utilise. Pour "multi-booter" avec NT, vous aurez besoin d'une 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), en suivant une procédure similaire à la procédure suivante :

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

Remarque : ceci est une excellente occasion pour vous rappeler que saisir bêtement des commandes auxquelles vous ne comprenez rien est vraiment une mauvaise idée. La commande ci-dessus ne fonctionnera pas directement sur la plupart des machines. Au lecteur de l'adapter à son équipement.

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 mettra dans votre partition NT/2000/XP partition, et le rajoutera dans `C:\BOOT.INI`.

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 référer 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 matériels 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 sûr 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 afin 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 transférez le message à

```
 dmesg@openbsd.org
```

Où `your-account@yourmail.dom` est votre compte de messagerie régulier. (ou transférez votre sortie dmesg en utilisant FTP/scp/disquette/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.

Notez 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 paquetage après l'installation

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

Parfois, vous réalisez que vous AURIEZ vraiment eu besoin de `comp39.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 paquetage 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 paquetages, 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 paquetages 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 comp39.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 sets d'installation manquants. Cela ne fonctionne pas. pkg_add(1) est l'outil de gestion des paquetages pour installer des applications tierces. Cet outil prend en compte les fichiers paquetages et non des archives tar génériques telles que les ensembles 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 conservent un tout le temps à la racine de leur système.

Le noyau "RAM Disk" définit la racine du système de fichiers du noyau -- plutôt qu'être stockés sur un disque physique, les utilitaires disponibles après l'amorçage de `bsd.rd` sont enregistrés dans le kernel, et lancés depuis un système de fichiers basé en mémoire RAM. `bsd.rd` comporte aussi une floppée d'utilitaires vous permettant de faire de la maintenance système et de lancer une installation.

Sur certaines architectures, `bsd.rd` est actuellement la méthode d'installation privilégiée -- 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 2.10
  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` quelle que soit la méthode pour votre architecture.

4.12 - Problèmes d'installation courants

4.12.1 - Mon Compaq ne reconnaît que 16Mo de RAM

Certains systèmes Compaq rencontrent un problème où la mémoire RAM n'est pas complètement détectée par le Chargeur d'amorçage de second niveau OpenBSD et seulement 16Mo seront détectés et utilisés par OpenBSD. Ceci peut être corrigé en créant/éditant 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, redémarrer et créer un fichier `/etc/boot.conf` avec la ligne précédente pour que dans les démarrages suivants OpenBSD reconnaisse toute la mémoire disponible.

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

4.12.2 - Mon i386 ne démarre pas après l'installation

Votre installation a eu l'air de bien se dérouler, mais lors de votre premier démarrage, vous ne voyez aucun signe montrant qu'OpenBSD essaye de démarrer. Plusieurs problèmes courants peuvent expliquer ce phénomène :

- **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'amorce OpenBSD a 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 ("-e") interactif de fdisk n'écrira pas les bits de signature requis pour rendre le disque amorçable.
- **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 (vieille et lente) machine a démarré, 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 *plusieurs heures* ou plus pour terminer les trois étapes ssh-keygen(1), Laissez le simplement terminer ; cela n'est réalisé qu'une fois par installation.

La taille de la clé par défaut a été augmentée pour OpenBSD 3.8, en conséquence de quoi les temps de génération de clés sont beaucoup plus élevés que ce qu'ils furent. Les utilisateurs possédant des machines très lentes souhaiteront peut-être générer leurs clés sur une autre machine, les placer dans l'archive site39.tgz, et les installer avec le reste des "file sets".

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

Quand vous faites l'installation d'un snapshot durant la phase *-beta* du cycle de développement OpenBSD, vous devriez voir ceci :

```
Do you want to see a list of potential FTP servers?  [yes] Entrée
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 précédant 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", vous obtiendrez donc les messages précédents.

Utilisez simplement la liste de miroirs FTP pour trouver votre site miroir 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 - Ma table de partition fdisk est corrompue ou vide !

Occasionnellement, un utilisateur trouvera un système fonctionnant, mais en faisant un `fdisk wd0`, il trouvera une table de partitions vide (ou polluée). 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 sensée avoir. (note : cela ne concerne que les plates-formes i386 et amd64. Les autres plates-formes requièrent des offsets différents, certaines n'en requièrent pas). Le système démarre ensuite en utilisant le PBR, pas le MBR.

Bien 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 recréé depuis le début (si vous savez VRAIMENT ce que vous faites, vous devriez 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. 39). Le fichier `siteXX.tgz` est, comme les autres paquetages, une compression gzip(1) d'archive tar(1) dont la racine est '/' et est décompressé comme les autres avec les options `xzpf`. Ce paquetage sera installé en dernier, après tous les autres paquetages.

Ce paquetage vous permet d'ajouter et/ou écraser des fichiers installés dans les paquetages '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.local`, 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, selon la procédure en cours, et lance le script dans un environnement chrooté de la racine système de l'installation/de la mise à jour. Rappelez-vous, la mise à jour est faite depuis un système de fichiers démarré donc votre système de fichiers cible est actuellement monté dans `/mnt`. Cependant, votre script peut être écrit tel quel à cause du chroot, 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 fichiers avant la mise à jour ou bien être placé lui aussi dans fichier `siteXX.tgz`.

Ce script peut être utilisé pour faire de nombreuses choses.

- Supprimer des fichiers qui sont installés/mis à jour et que vous ne souhaitez pas présents dans votre système.
- Supprimer/mettre à jour/installer les paquetages que vous souhaitez sur le système installé.
- Faire une sauvegarde/archive immédiate de votre nouveau système avant de l'exposer au reste du monde.
- Utilisez rdate(8) pour paramètrer l'horloge du système.

La combinaison de `siteXX.tgz` et de `install.site/upgrade.site` a pour but de donner de larges capacités de personnalisation sans avoir à créer ses propres paquetages 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` et `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 configuration désirée depuis une bande ou autre média, et installer les blocs de d'amorce. Plus de détails ici.

Image de disque

Malheureusement, il n'existe pas de paquetage d'image de disque reconnaissant le FFS, et qui pourrait faire une image contenant simplement l'espace de disque utilisé. La plupart des solutions d'image de disque traiteront 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'énormes 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 fournira souvent les mêmes fonctionnalités que les programmes commerciaux, sans le prix.

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

Lorsque vous rapportez un problème, il est important d'inclure le dmesg(8) complet du système. Souvent cependant, vous en avez besoin parce que le système ne fonctionne pas correctement ou ne s'installe pas, donc vous n'aurez pas de disque, pas de réseau ou manquerez d'une 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'amorçage et les CD-ROM ont assez d'outils pour permettre d'enregistrer votre dmesg sur une disquette MSDOS afin de 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 a 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 pour 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 câble série (souvent un câble null-modem), et d'un émulateur de terminal pouvant capturer la sortie de l'écran dans un fichier.

  Des informations sur la configuration d'une console série sont données ailleurs 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, et sous réserve de paramétrage correct de votre réseau, 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.

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


www@openbsd.org

$OpenBSD: faq4.html,v 1.56 2006/10/25 07:40:55 jufi 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

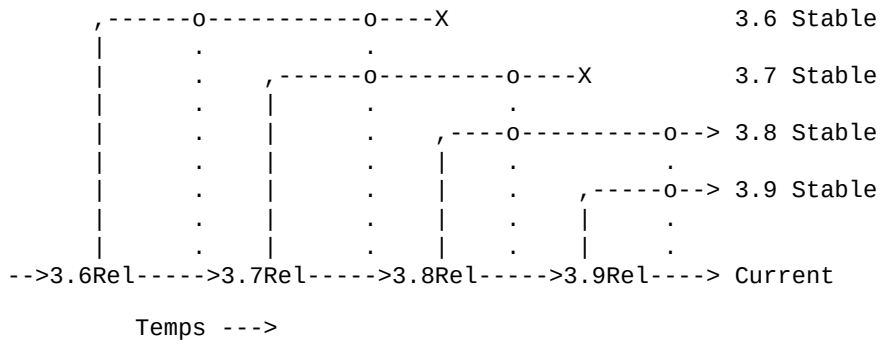
Table des matières

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 pour laquelle le développement est réalisé, et qui deviendra la prochaine "release".

Schématiquement, le développement de ces versions ressemble à ceci :



-Current: La version courante pour laquelle le développement est réalisé, et qui deviendra la prochaine *-release* d'OpenBSD. Tous les six mois, lorsqu'une version d'OpenBSD est révisée, *-current* est tagguée, et devient *-release* : un point gelé dans l'histoire de l'arbre des sources. Chaque *-release* demeure inchangée; c'est pour cela qu'on la trouve sur les CDs et serveurs FTP.

-Stable est basée sur *-release*, et est une branche du chemin de développement principal d'OpenBSD. Quand des corrections très importantes sont faites sur *-current*, elles sont "backportées" (intégrées) aux branches *-stable*; à cause de celà, *-stable* est aussi connue sous le nom de *branche patch*. Dans l'illustration ci-dessus, la ligne verticale en pointillés symbolise les corrections de bogues incorporées aux branches *-stable*. Vous remarquerez aussi dans l'exemple ci-dessus que la branche *3.6-stable* a été supprimée à la sortie de *3.8-release*, et que la branche *3.7-stable* a été supprimée à la sortie de *3.9-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 préférerons nous concentrer sur les nouvelles fonctionnalités. La branche *-stable* est, par conception, très facile à construire à partir de *-release* de la même version (c'est à dire en allant de *3.9-release* vers *3.9-stable*).

La branche *-stable* est *-release* à laquelle on a ajouté 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 de manuel doivent être modifiées, il est très probable que ces modifications ne fassent 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 cela est considéré comme très important.

Il est utile de préciser que le nom "*-stable*" ne signifie pas pour autant que *-current* est instable. *-current* change et évolue, alors que *-stable* ne change que très peu, et vous n'aurez pas à apprendre à nouveau votre système ou à changer des fichiers de configuration. En fait, notre préoccupation étant de faire continuellement évoluer OpenBSD, vous devriez trouver *-current* plus performante que *-stable*.

Attention : *-current* change tout le temps. Elle change pratiquement toutes les minutes. Bien que les développeurs travaillent sans cesse afin que le système se compile toujours et qu'il n'y ait pas de bogues majeurs, il est tout à fait possible de récupérer les sources de *-current* et que celles-ci ne soient pas compilables, alors que tout fonctionne correctement cinq minutes plus tard. Il y a aussi des "jours drapeau" et des changements majeurs du système que les développeurs gèrent avec des outils en un, mais qui empêchent une quelconque mise à jour basée sur les sources. **Si vous n'êtes pas préparé à les traiter, rester à l'écart 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*. "Ça ne marche pas !" n'est pas un rapport de bogue utile. "Les changements récents dans le pilote pciide ont 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 qu'il souhaite utiliser une nouvelle fonctionnalité de *-current*. Dans ce cas, soit l'utilisateur utilise *-current* soit il 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 ne 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 point de

départ nécessaire avant de tenter de compiler *-current* depuis les sources.

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 été rajouté aujourd'hui à l'arborescence des ports. OpenBSD n'a malheureusement que des ressources limitées.

Il faut aussi comprendre que 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 *3.9-current* (ou un snapshot), puis décider que c'est trop dangereux, et revenir vers *3.9-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 devrais-je compiler mon système depuis les sources ?

Actuellement, il y a de fortes chances pour que vous n'en ayez pas besoin.

Voici quelques raisons pour NE PAS compiler votre systèmes depuis les sources :

- La compilation de votre propre système à des fins de mise à niveau n'est pas supportée.
- Vous n'obtiendrez pas de meilleurs performances en compilant votre système.
- Le changement des options de compilation est plus enclin à briser votre système qu'a l'améliorer.

Voici quelques raisons d'avoir besoin de compiler depuis les sources :

- Tester ou développer de nouvelles fonctionnalités.
- La compilation du système est éprouvante pour l'ordinateur, ceci peut être un bon moyen de s'assurer du bon fonctionnement de ce dernier.
- Vous souhaitez suivre la branche stable.
- Vous souhaitez construire un système OpenBSD hautement personnalisé pour des utilisations spécifiques.

L'équipe OpenBSD réalise fréquemment des instantanés basés sur le code de *-current* pour toutes les plates-formes. Dans la plupart des cas, cela vous suffira pour utiliser *-current*.

La raison la plus courante de compiler depuis les sources est de suivre la branche *-stable*, pour laquelle la compilation depuis les sources est la seule issue supportée...

5.3 - Compilation d'OpenBSD depuis les sources

5.3.1 - Aperçu du processus de compilation

La compilation d'un système OpenBSD depuis les sources implique un certain nombre d'étapes :

- 5.3.2 - Installer ou mettre à niveau depuis les binaires les plus récents
- 5.3.3 - Téléchargement des sources appropriées

- 5.3.4 - Compilation du noyau
- Compilation du "userland" ("make build").

Il ya deux étapes supplémentaires que certains utilisateurs pourraient vouloir réaliser, ceci dépend de la raison de la compilation et de la présence ou non de X :

- Compilation d'une Release.
- Compilation de X.

5.3.2 - Installer ou mettre à niveau depuis les binaires les plus récents

La première étape dans la compilation depuis les sources est de s'assurer d'avoir le binaire le plus récent installé. Utilisez ce tableau afin de savoir ou vous en êtes, ou vous voulez aller, et avec quel binaire commencer :

Vous êtes au	**But**	**Mise à niveau binaire vers**	**ensuite ...**
Ancienne -release	Nouvelle release	Dernière release	C'est fait !
-release	-stable	Dernière release	Récupérer & compiler *-stable*
Ancienne -stable	-stable	Nouvelle release	Récupérer & compiler *-stable*
-release	-current	Dernier instantané	Récupérer (optionnel) & compiler *-current*
Ancienne -current	-current	Dernier instantané	Récupérer (optionnel) & compiler *-current*

Il est recommandé d'installer le binaire via l'option "Upgrade" du média d'installation. Si ce n'est pas possible, vous pouvez aussi décompacter les binaires comme décris ici. Vous pouvez faire le processus complet de mise à niveau sans soucis, incluant la création d'utilisateurs ou d'autres changements de répertoires de `/etc`.

5.3.3 - Téléchargement des sources appropriées

Le source d'OpenBSD est géré en utilisant le système de contrôle de versions CVS, et cvs(1) est utilisé pour récupérer les sources désirées sur votre machine locale, à des fins de compilation. Ceci peut être réalisé en utilisant le serveur AnonCVS (une machine proposant une copie du dépôt CVS utilisé par le projet OpenBSD) et ce publiquement, ou depuis une archive locale que vous maintenez en utilisant les programmes CVSup, ou CVSync, disponibles dans les paquetages. CVSup peut aussi être utilisé en mode "checkout", mais cela n'est pas couvert par le présent document. Si vous avez plusieurs machines sur lesquelles vous désirez maintenir le code source, vous devriez trouver utile d'avoir un dépôt CVS local, créé et maintenu en utilisant CVSup ou CVSync.

Après avoir décidé quel Serveur AnonCVS vous allez utiliser, vous devez effectuer un "checkout" (récupération) de l'arbre des sources, que vous maintiendrez par la suite en lançant des "updates" (mises à jour), afin de récupérer les fichiers les plus récents dans votre arbre local.

La commande CVS(1) dispose de nombreuses options, certaines d'entre elles sont ***requises*** afin de récupérer et de mettre à jour un arbre. Les autres commandes peuvent résulter en un arbre cassé. Le fait de comprendre et de suivre les directions est ici important.

Suivis de *-current*

Dans ce cas, nous supposons que vous utilisez un serveur AnonCVS publique, **anoncvs@anoncvs.example.org:/cvs**. Nous supposons également que vous utilisez le shell sh(1), vous devrez dans le cas contraire ajuster certaines commandes.

Pour récupérer un arbre des sources CVS de *-current*, vous pouvez utiliser la commande suivante :

```
# cd /usr
# export CVSROOT=anoncvs@anoncvs.example.org:/cvs
# cvs -d$CVSROOT checkout -P src
```

Une fois l'arbre récupéré, vous pourrez plus tard le mettre à jour avec :

```
# cd /usr
# export CVSROOT=anoncvs@anoncvs.example.org:/cvs
# cvs -d$CVSROOT up -Pd
```

Suivis de *-stable*

Si vous souhaitez récupérer une "branche" alternative de l'arbre, comme la branche *-stable*, vous devez utiliser le modificateur "`-r`" :

```
# cd /usr
# export CVSROOT=anoncvs@anoncvs.example.org:/cvs
# cvs -d$CVSROOT checkout -rOPENBSD_3_9 -P src
```

Ceci aura pour effet de récupérer les fichiers sources de la branche OPENBSD_3_9, aussi connue sous le nom de "Branche patchée" ou "-stable". Vous pourrez mettre à jour le code de façon similaire :

```
# cd /usr/src
# export CVSROOT=anoncvs@anoncvs.example.org:/cvs
# cvs -d$CVSROOT up -rOPENBSD_3_9 -Pd
```

CVS est vraiment agréable car il permet de suivre une balise dans les fichiers récupérés, vous n'avez ainsi pas à vous rappeler de la partie "`-rOPENBSD_3_9`" de la ligne de commande, ceci sera mémorisé jusqu'a ce que vous l'effaciez ou en spécifiez un autre via l'option "`-A`" de "`update`". Cependant, il est probablement plus correct de fournir trop d'infos que pas assez sur vos lignes de commande CVS.

L'arbre "`src`" ayant le seul à avoir été montré jusqu'a présent, vous pouvez faire les mêmes étapes pour "`XF4`" et "`ports`". Toutes les parties d'OpenBSD devant être en synchronisation, tous les arbres utilisés devraient être mis à jour en même temps. Vous pouvez allier plusieurs récupérations sur une seule ligne (*-stable* est montré) avec :

```
# cd /usr/src
# export CVSROOT=anoncvs@anoncvs.example.org:/cvs
# cvs -d$CVSROOT checkout -rOPENBSD_3_9 -P src ports XF4
```

Cependant, les mises à jour doivent être faites répertoire par répertoire.

A ce niveau, que vous ayez suivi *-stable* ou *-current* vous devriez avoir un arbre des sources correct. Soyez attentif à ce que vous récupérez -- vous pourriez compiler *-current* en pensant compiler *-stable*.

Pré-chargement de l'arbre des sources : `src.tar.gz, sys.tar.gz`

Au lieu de télécharger l'arbre des sources dans sa totalité à partir d'un serveur AnonCVS, vous pouvez le "pré-charger" à partir des fichiers des sources se trouvant sur le CD d'OpenBSD ou sur les serveurs FTP. Vous économiseriez ainsi beaucoup de temps et de bande passante. Ceci est particulièrement vrai si utilisez *-stable*, étant donné le peu de différences entre cette version et *-release*.

Pour extraire l'arbre des sources à partir du CD vers */usr/src* (en supposant que le CD est monté dans /mnt) :

```
# cd /usr/src; tar xzf /mnt/src.tar.gz
# cd /usr; tar xzf /mnt/XF4.tar.gz
# tar xzf /mnt/ports.tar.gz
```

Les fichiers sources téléchargeables depuis les serveurs FTP sont séparés en deux fichiers pour minimiser le temps nécessaire à leur téléchargement pour les personnes souhaitant travailler avec telle ou telle partie de l'arbre uniquement. Ces deux fichiers sont `sys.tar.gz`, qui contient les fichiers utilisés pour créer le noyau, et `src.tar.gz` qui contient toutes les autres applications "userland", hormis l'arbre des ports et les sources X11. Cependant, et de manière générale, vous aurez besoin des deux. En supposant que vous les ayez téléchargé `src.tar.gz` et `sys.tar.gz` dans `/usr` :

```
# cd /usr/src
# tar xzf ../sys.tar.gz
```

```
# tar xzf ../src.tar.gz
# cd /usr
# tar xzf XF4.tar.gz
# tar xzf ports.tar.gz
```

L'extraction de toutes les parties de l'arbre source n'est pas obligatoire mais si on souhaite garder un système cohérent, il est conseillé de les extraire toutes.

Astuces CVS courantes

Comme indiqué précédemment, certaines options sont obligatoires afin d'obtenir un arbre `src` d'OpenBSD valide. L'option "`-P`" ci-dessus est l'une d'entre elle : elle "prune" (efface) les répertoires vides. Avec les années, des répertoires ont été créés puis supprimés, et les noms de ces répertoires sont parfois utilisés actuellement pour des fichiers. Sans l'option "`-P`", votre arbre NE compilera PAS.

Même chose pour l'option `-d` sur une commande 'update' -- elle crée les nouveaux répertoires ayant pu être ajoutés au dépôt depuis votre dernier "checkout".

Les utilisateurs de CVS expérimentés auront pu se demander pourquoi le CVSROOT est précisé et utilisé dans cet exemple, alors que cvs(1) enregistre la situation du serveur dans l'arbre ainsi obtenu. Ceci est correct, cependant, un utilisateur devra souvent outrepasser le serveur enregistré, et de nombreuses personnes recommandent de *toujours* spécifier le dépôt à utiliser. Il est aussi à noter que si la variable d'environnement CVSROOT peut être utilisée directement par cvs(1), elle est utilisée uniquement si rien ne viens la remplacer, et la spécification de la ligne de commande est prépondérante.

Dans cet exemple, notez que `-Pd` est utilisé comme paramètre de la commande `up`. Ceci est une autre de ces options REQUISES, elle permettra aux nouveaux répertoires qui n'existaient pas dans la récupération précédente d'être créés lors du processus de mise à jour.

Il est souvent utile d'utiliser un `.cvsrc` dans votre répertoire home afin de spécifier les options par défaut. Un fichier `.cvsrc` d'exemple :

```
$ more ~/.cvsrc
cvs -q -danoncvs@anoncvs.example.org:/cvs
diff -up
update -Pd
checkout -P
```

Ce fichier ordonnera à cvs(1) d'utiliser le serveur `anoncvs@anoncvs.example.org:/cvs`, supprimant les sorties souvent inutiles ("`-q`" pour "quiet", tranquille) pour toutes les opérations, la commande "cvs up" utilisant par défaut `-Pd`, la commande "cvs diff" réalisant par défaut des "diffs unifiés" suite à l'option "`-u`", et un "cvs checkout" utilisera l'option "`-P`". Tandis que cela est confortable, si vous oubliez que ce fichier existe, ou si vous essayez de lancer ces commandes auxquelles vous êtes habitué sans ce fichier, vous rencontrerez des problèmes.

L'arbre des sources est constitué d'un grand nombre de petits fichiers. Il est donc conseillé d'activer soft updates sur la partition où se trouve cet arbre afin d'améliorer significativement les performances.

5.3.4 - Compilation du noyau

Nous supposerons que vous désirez construire un noyau standard (GENERIC ou GENERIC.MP). Normalement, c'est ce que vous devriez faire. *N'*essayez pas de construire un noyau personnalisé si vous ne maîtrisez pas le processus de compilation standard.

Evidemment, le noyau est un composant TRES dépendant du matériel du système. Les sources du noyau sont dans le répertoire `/usr/src/sys`. Certaines parties du code du noyau d'OpenBSD sont utilisées sans distinction de plate-forme, alors que d'autres sont très spécifiques à un processeur ou une architecture. Si vous regardez dans le répertoire `/usr/src/sys/arch/`, vous verrez certaines choses quelque peu intrigantes -- par exemple, il y a des répertoires `mac68k`, `m68k` et `mvme68k`. Dans ce cas, les systèmes mvme68k et mac68k utiliseront tous deux le même processeur, mais les machines sur lesquelles ils sont basés sont très différentes, et elles demandent ainsi un noyau très différent (il y a bien plus de choses qu'un processeur dans un ordinateur !). Cependant, certaines parties du noyau sont communes, elles demeurent dans le répertoire m68k. Si vous compilez simplement un noyau, les répertoires de l'architecture de base comme `m68k` ne sont pas un soucis, vous devriez travailler uniquement avec les répertoires de "l'architecture composant", comme `mvme68k`.

Les noyaux compilés sont basés sur les fichiers de configuration du noyau, qui se trouvent dans le répertoire `/usr/src/sys/arch/<votre plate-forme>/conf`. La compilation du noyau consiste à l'utilisation du programme config(8) pour créer et peupler un répertoire compile, qui se terminera dans `/usr/src/sys/arch/<votre plate-forme>/compile/<nom du noyau>`. Pour cet exemple, nous supposerons que

vous utilisez la plate-forme i386 :

```
# cd /usr/src/sys/arch/i386/conf
# config GENERIC
# cd ../compile/GENERIC
# make clean && make depend && make
    [...beaucoup de sortie...]
# make install
```

Remplacez "`i386`" sur la première ligne par le nom de votre plate-forme. La commande machine(1) peut vous donner le nom de la plate-forme sur laquelle vous êtes afin de pouvoir utiliser "`cd /usr/src/sys/arch/`machine`/conf`" à la place de la première ligne.

A ce stade, redémarrez votre machine afin d'activer le nouveau noyau. Notez que le nouveau noyau devrait être lancé avant l'étape suivante, ce que vous savez si vous avez suivis les explications ci-dessus sur la mise à niveau vers les instantanés les plus récents. Parfois, les APIs changent cependant, et l'ancien noyau sera dans l'incapacité de lancer de nouvelles applications, mais les nouveaux noyaux supportent en général les applications plus anciennes.

Variation du processus ci-dessus : arbre des sources en lecture seule

Parfois, vous voudrez peut-être vous assurez que `/usr/src/sys` reste intacte. Ceci peut être fait en utilisant les procédés suivants :

```
$ cd /somewhere
$ cp /usr/src/sys/arch/i386/conf/GENERIC .
$ config -s /usr/src/sys -b . GENERIC
$ make clean && make depend && make
   ... beaucoup de messages affichés ...
```

Remarquez que vous pouvez compiler un noyau sans accès root, mais vous devez être root pour pouvoir l'installer.

5.3.5 - Compilation du "userland"

Il y a une marche à suivre spécifique afin que cela fonctionne, sans quoi vous vous amuserez à trouver d'où les problèmes viennent.

- Nettoyez votre répertoire `/usr/obj` et reconstruisez les liens symboliques :

  ```
  # rm -rf /usr/obj/*
  # cd /usr/src
  # make obj
  ```

  Remarquez que l'utilisation du répertoire `/usr/obj`est obligatoire. Echouer à cette étape avant la compilation laissera votre arbre `src`dans un mauvais état.

- Soyez sûr que tous les répertoires nécessaires sont présents.

  ```
  # cd /usr/src/etc && env DESTDIR=/ make distrib-dirs
  ```

- Compilez le système :

  ```
  # cd /usr/src
  # make build
  ```

  Ceci compile et installe tous les outils du "userland" dans un ordre approprié. Cette étape est très coûteuse en temps -- une machine très rapide devrait la réaliser sous une heure, une machine très lente pourrait mettre plusieurs jours. Lorsque cette étape est terminée, vous avez donc de nouveaux binaires à leurs places sur votre système.

- **Si vous compilez *-current* :** Mettez à jour `/dev` et `/etc`, avec les changements listés dans current.html. Si vous suivez -stable d'après le

processus de mise à niveau ou via une installation du binaire correspondant, cette étape n'est ni désirée ni nécessaire.

5.4 - Compilation d'une Révision

Qu'est ce qu'une "release" (révision), et pourquoi voudrais-je en créer une ?

Une "release" est le jeu de fichiers complet pouvant être utilisé pour installer OpenBSD sur un ordinateur. Si vous n'avez qu'une seule machine sous OpenBSD, vous n'avez pas de réelle motivation à construire une release, le processus ci-dessus vous apportant tout ce dont vous avez besoin. Un exemple d'utilisation du processus de release pourrait être de compiler une *-stable* sur une machine puissante, et de faire une release installable sur tous vos ordinateurs.

Le processus de release utilise les binaires créés dans le répertoire `/usr/obj` lors du processus ci-dessus, vous devez donc accomplir celui-ci au préalable, et rien ne doit perturber le répertoire `/usr/obj`. Ceci peut être un problème si vous utilisez un disque mémoire pour `/usr/obj` avec de petites performances lors du processus de compilation, vous ne voudrez pas redémarrer l'ordinateur entre les étapes de compilations et de "release" !

Le processus de release requiert deux répertoires de travail, appelés DESTDIR et RELEASEDIR. Tous les fichiers faisant partie d'une installation OpenBSD "propre" seront copiés à l'endroit correct au sein de DESTDIR. Ils seront tar(1)és puis placés dans RELEASEDIR. A la fin du processus, RELEASEDIR arborera la release OpenBSD intégralement. Le processus de release utilisera aussi `/mnt`, ce point ne doit donc pas être utilisé par autre chose pendant le processus de release. A titre d'exemple, nous utiliserons DESTDIR à `/usr/dest` et RELEASEDIR à `/usr/rel`.

Deux utilitaires qui ne sont pas dans le système de base d'OpenBSD, crunch et crunchgen(1), sont utilisés pour créer un exécutable unique composé de plusieurs binaires. Le nom qui l'invoque détermine quel composant binaire est utilisé. C'est un peu comme si plusieurs fichiers de programmes individuels étaient concentrés sur un ramdisk noyau qui existe sur les disquettes et autres médias d'installation. *Ces utilitaires doivent être traités lorsque le processus de release est lancé* Ils n'ont besoin d'être installés qu'une seule fois, mais certaines personnes oublient parfois cette étape, et ces programmes sont compilés rapidement, certaines personnes optent pour compiler crunch et crunchgen à chaque fois qu'ils utilisent le script.

Vous devez avoir les privilèges root pour créer une release.

Faire une release

Tout d'abord, si cela n'a pas encore été fait sur la machine, compilez crunch et crunchgen :

```
# cd /usr/src/distrib/crunch && make obj depend all install
```

A présent, définissons nos variables d'environnement DESTDIR et RELEASEDIR :

```
# export DESTDIR=/usr/dest
# export RELEASEDIR=/usr/rel
```

Nettoyons DESTDIR et créons le répertoire :

```
# test -d ${DESTDIR} && mv ${DESTDIR} ${DESTDIR}.old && rm -rf ${DESTDIR}.old &
# mkdir -p ${DESTDIR} ${RELEASEDIR}
```

RELEASEDIR ne doit pas forcément être vide lors du lancement du processus mais, cependant, s'il y a des changement dans les fichiers de la release ou dans leurs noms, les anciens fichiers seront conservés. Vous voudrez ainsi certainement effacer ce répertoire.

Passons à présent à la release elle-même :

```
# cd /usr/src/etc
# make release
```

Une fois la release construite, il peut être bon de la vérifier afin d'être sûr que les fichiers tar reflètent bien le contenu de DESTDIR. La sortie de cette étape devrait être très courte :

```
# cd /usr/src/distrib/sets
# sh checkflist
```

Vous avez à présent un jeu de fichiers complet et fonctionnel dans RELEASEDIR. Ces fichiers peuvent être utilisés afin d'installer ou de mettre à jour OpenBSD sur d'autres machines.

Les instructions faisant foi sur la création d'une release sont dans release(8).

Remarque : Si vous souhaitez distribuer les fichiers résultants par HTTP pour être utilisables par les scripts de mise à jour ou d'installation, vous aurez besoin d'ajouter un fichier "index.txt". Ce fichier contient la liste de tous les fichiers constituant votre release fraîchement créée.

```
# /bin/ls -1 >index.txt
```

5.5 - Compilation de X

X utilise possède un processus de compilation différent du reste de l'arbre OpenBSD, étant basé sur imake, plutôt que sur le make(1) standard. Une conséquence de ceci est qu'il n'y a pas de répertoire "obj", les binaires générés finissent mélangés au code source, ce qui peut poser des problèmes (ou du moins des sorties excessives) avec cvs(1). Une solution à ce problème est d'utiliser lndir(1) afin de créer un répertoire ombre contenant des liens symboliques vers le répertoire des sources actuel pour l'arbre XF4.

i386 seulement : La plate-forme i386 nécessite l'installation des paquetages "tcl" et "tk" avant la compilation de X (ils se trouvent dans l'arbre des ports à `/usr/ports/lang/tcl/8.4/` et `/usr/ports/x11/tk/8.4/`. L'installation du paquetage "tk" installera "tcl" en tant que dépendance). Comme d'habitude, l'installation qu'un paquetage est plus rapide que l'installation de ces applications depuis les sources. Un échec de l'installation de ces paquetages avant de compiler X est quelque chose de frustrant, car le système fonctionnera un certain temps sans afficher la moindre erreur.

Afin de compiler X en utilisant le "répertoire ombre" de `/usr/Xbld` (compiler et installer les nouveaux binaires dans les bons répertoires), suivez ces étapes :

```
# rm -rf /usr/Xbld
# mkdir -p /usr/Xbld
# cd /usr/Xbld
# lndir ../XF4
   [...beaucoup de sortie...]
# make build
   [...beaucoup de sortie...]
```

Création d'une release X

Ceci est similaire au processus de release principal. Après avoir compilé X avec succès, vous définirez DESTDIR et RELEASEDIR, de la même manière qu'évoqué précédemment. RELEASEDIR peut être le même répertoire que la principale RELEASEDIR système, mais DESTDIR sera effacé et reconstruit lors de ce processus. Si cela est fait avec attention, ce n'est pas un problème, mais l'utilisation d'une DESTDIR séparée est plus sûr.

Pour cet exemple, nous utiliserons les DESTDIR et RELEASEDIR comme `/usr/Xbld/dest` et `/usr/Xbld/rel`, respectivement. Ceci doit être fait après le processus de compilation ci-dessus.

```
# export DESTDIR=/usr/Xbld/dest
# export RELEASEDIR=/usr/Xbld/rel
# cd /usr/Xbld
# rm -rf dest
# mkdir dest rel
# make release
```

Si vous prévoyez de faire à la fois une compilation et une release de X, vous pouvez utiliser une cible make(1) différente, "b-r", qui réalise à la fois la compilation et les étapes de release. La cible "b-r" suppose que DESTDIR et RELEASEDIR sont les sous-répertoires "`rel`" et "`dest`" au sein de votre sous-répertoire de compilation :

```
# rm -rf /usr/Xbld
# mkdir -p /usr/Xbld /usr/Xbld/dest /usr/Xbld/rel
# cd Xbld
# lndir ../XF4
   [...beaucoup de sortie...]
# make b-r
   [...beaucoup de sortie...]
```

5.6 - 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* ou *-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.

5.7 - Options de configuration du noyau

Nous partons du fait que vous avez lu ci-dessus, et que vous aimez la douleur. Il est aussi supposé que vous avez un but ne pouvant être atteint ni avec Configuration au démarrage (UKC>), ni avec config(8)urer un noyau GENERIC. Si les deux possibilités sont fausses, vous devriez vous en tenir à utiliser GENERIC. Vraiment.

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 de 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 ! N'éditez pas le fichier de configuration `GENERIC` !! 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 bogue concernant un noyau personnalisé va dans la plupart des cas se résumer à une réponse 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. Il n'y a aucune garantie sur le fait qu'un noyau pouvant être "config(1)uré" puisse être compilé.

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
- Autres architectures

Si vous lisez attentivement ces fichiers, vous verrez une ligne comme :

```
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
```

ou

```
option      nom=valeur
```

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.

Vous aurez peut-être parfois besoin de désactiver une option précédemment 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.

Compiler un noyau sur mesure

Dans ce cas, nous compilerons un noyau supportant les cartes série ISA multi-port boca(4). Cette carte n'est pas dans le noyau GENERIC, du fait qu'elle entre en conflit avec d'autres drivers. Une autre raison courante de construire un noyau personnalisé est d'utiliser RAIDframe, trop gros pour être dans le noyau par défaut. Il y a deux moyens courants de faire un noyau personnalisé : copier le fichier de configuration GENERIC vers un autre nom et l'éditer, ou créer un fichier "wrapper" qui inclut le noyau standard GENERIC ainsi que toutes les options dont vous avez besoin, et qui ne sont pas dans GENERIC. Dans ce cas, votre fichier "wrapper" pourrait ressembler à :

```
include "arch/i386/conf/GENERIC"

boca0  at       isa? port 0x100 irq 10     # BOCA 8-port serial cards
pccom* at       boca? slave ?
```

Les deux lignes au regard des cartes boca(4) sont copiées depuis les lignes commentées dans `GENERIC`, avec l'IRQ ajusté comme requis. L'avantage d'utiliser ce fichier "wrapper" est que les changements imprévus dans GENERIC sont automatiquement mis à jour comme lors de toute mise à jour du code. L'inconvénient est qu'on ne peut supprimer un périphérique (cependant, c'est en général une mauvaise idée).

Un autre moyen de générer un noyau personnalisé est de faire une copie du `GENERIC` standard, en lui donnant un autre nom, et en l'éditant selon les besoins. L'inconvénient est que les futures mises à jour du fichier de configuration GENERIC devront être fusionnés dans votre copie, ou vous devrez refaire votre fichier de configuration.

Dans un autre événement, après avoir fait votre fichier de configuration personnalisé, utilisez config(8) et construisez le noyau comme documenté ci-dessus.

Les instructions complètes pour la création de votre propre noyau sont dans la page de manuel afterboot(8).

5.8 - Configuration au démarrage

Lorsque vous démarrez votre système il est possible que vous remarquiez que votre noyau trouve vos périphériques mais à la mauvaise IRQ. Et que vous aillez immédiatement besoin de ce périphérique. Sans recompiler votre noyau vous pouvez utiliser la configuration au démarrage de celui-ci. Cela vous 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 utilisant config(8). 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 trouvée dans la page de manuel boot_config(8).

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

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

ou n'importe quel autre noyau que vous voulez démarrer. L'invite de commandes UKC apparaîtra, et 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 votre noyau configuré, utilisez `quit` ou `exit` et continuez le démarrage. Une fois votre système démarré, vous devriez rendre la modification permanente au niveau de votre binaire du noyau, tel que c'est décrit dans utilisation de config(8) pour changer le binaire du noyau

5.9 - 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 ne sont pas écrits dans le binaire du noyau. Pour plus d'informations sur les messages d'erreur et autres avertissements, lisez la page de manuel config(8) .

```
$ sudo config -e /bsd
OpenBSD 3.9 (GENERIC) #617: Thu Mar  2 02:26:48 MST 2006
    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. Étant 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 noyau binaire bsd.new.

```
$ sudo config -e -u -o bsd.new /bsd
OpenBSD 3.9 (GENERIC) #617: Thu Mar  2 02:26:48 MST 2006
    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
```

5.10 - Obtention d'une sortie plus verbeuse lors du démarrage

L'obtention d'une sortie plus verbeuse peut s'avérer être très utile lors des tentatives de résolution de problème au boot. Si vous avez un problème inhérent à votre disquette de boot et désirez obtenir plus d'informations, redémarrez. A l'arrivée au prompt "boot>", bootez avec l'option -c. Ceci vous permettra d'être dans UKC>, puis de faire :

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

Vous aurez à présent une sortie très verbeuse au démarrage.

5.11 - Problèmes courants, astuces et questions lors de la compilation et de la construction

La plupart du temps, les problèmes lors de la compilation sont causés par le fait de ne pas suivre précisément les indications précédentes. Cependant, il peut exister occasionnellement de véritables problèmes lors de la compilation de *-current* à partir du snapshot le plus récent mais les erreurs lors de la compilation de *-release* ou *-stable* sont presque toujours dues à une erreur de l'utilisateur.

Les problèmes les plus courants sont :

- Ne pas démarrer à partir du binaire approprié, essayer de mettre à jour à partir des sources ou penser qu'un snapshot vieux d'une semaine est "suffisamment proche".

- Récupérer la mauvaise branche de l'arbre CVS.
- Ne pas suivre la procédure.
- Tenter d'adapter ou d'"optimiser" votre système.

Voici tout de même certains problèmes auxquels vous pourriez être confronté :

5.11.1 - 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 mémoire. 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 un snapshot ou une "révision".

Pour plus d'informations, consultez la Faq Sig11.

5.11.2 - "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), comme documenté ci-dessus. Ces options s'assurent que les nouveaux répertoires sont ajoutés ou supprimés au fur et à mesure qu'OpenBSD évolue.
- **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 la récupération et de la compilation de votre arbre.

5.11.3 Mon système sans IPv6 ne fonctionne pas !

Oui. Nous vous prions de ne pas faire de modifications au système de base pour lesquelles vous ne comprenez pas toutes les conséquences. Un "petit" changement dans le noyau peut avoir un impact très large sur le reste entier du système. Relisez s'il vous plaît ceci.

5.11.4 - Oops ! J'ai oublié de créer le répertoire `/usr/obj` avant de commencer !

En faisant un "make build" avant de faire un "make obj", you devrez en terminer avec les fichiers objets accumulés dans votre répertoire `/usr/src`. C'est une mauvaise chose. Si vous désirez essayer d'avoir à télécharger l'arbre des sources une nouvelle fois, vous pouvez essayer les manipulations suivantes afin de nettoyer les fichiers objets :

```
# cd /usr/src
# find . -type l -name obj | xargs rm
# make cleandir
# rm -rf /usr/obj/*
# make obj
```

5.11.5 - Conseil : Placer `/usr/obj` sur sa propre partition

Si vous compilez souvent, vous trouverez plus rapide de mettre `/usr/obj` sur sa propre partition. Le bénéfice est simple, il est typiquement plus rapide de :

```
# umount /usr/obj
# newfs VotrePartitionObj
# mount /usr/obj
```

que de faire "rm -rf /usr/obj".

5.11.6 - Comment ne pas compiler certaines parties de l'arbre ?

Vous souhaiterez peut-être de pas compiler certaines parties de l'arbre, typiquement parce que vous l'avez remplacé par une application incluse dans un paquetage, ou parce que vous voulez créer une installation "plus petite" pour diverses raisons. La solution est d'utiliser l'option SKIPDIR de /etc/mk.conf.

Note : il est de ce fait possible de compiler des systèmes cassés. Les résultats de cette option ne sont pas supportés par le projet OpenBSD.

5.11.7 - Ou puis-je avoir plus d'informations sur le processus de compilation ?

- release(8)
- afterboot(8)
- mk.conf(5)
- `/usr/src/Makefile`
- Branches Patchées (*-stable*)
- (pour X) `/usr/X11R6/README` sur votre système

5.11.8 - Je ne vois aucun snapshot sur le site FTP. Ou sont-ils passés ?

Les snapshots peuvent être retirés lorsqu'ils deviennent anciens (ou obsolètes) ou lorsque la sortie d'une *-release* approche.

5.11.9 - Comment puis-je amorcer sur une nouvelle version du compilateur (*gcc*)?

Vous devriez vraiment installer le dernier snapshot.

OpenBSD supporte désormais deux compilateurs dans l'arbre, gcc v3.3.5 par la plupart des plates-formes, et également gcc v2.95.3 utilisé par des plates-formes qui n'ont pas encore été converties, ou qui ne seront pas converties à cause des manques du support de gcc3 ou de ses performances pauvres.

Les deux compilateurs sont dans des parties différentes de l'arbre :

- gcc3: `/usr/src/gnu/usr.bin/gcc`
- gcc2: `/usr/src/gnu/egcs/gcc`

Parce que la mise à niveau du compilateur est un peu comme le problème du poulet et de l'oeuf, les changements sur le compilateur intégré à l'arbre demandent une attention toute particulière. Vous devez construire le compilateur deux fois -- la première construction produit un compilateur qui génère un nouveau code mais utilise l'ancien code pour fonctionner, la deuxième construction construit un nouveau compilateur intégralement. En général, vous voudrez réaliser la procédure suivante :

```
Si votre plate-forme utilise gcc 2.95.3 :
   # rm -r /usr/obj/gnu/egcs/gcc/*
   # cd /usr/src/gnu/egcs/gcc
    - ou -
Si votre plate-forme utilise gcc 3.3.5 :
   # rm -r /usr/obj/gnu/usr.bin/gcc/*
   # cd /usr/src/gnu/usr.bin/gcc

Procédure commune de compilation pour v3.3.5 ou v2.95.3
# make -f Makefile.bsd-wrapper clean
# make -f Makefile.bsd-wrapper obj
# make -f Makefile.bsd-wrapper depend
```

```
# make -f Makefile.bsd-wrapper
# make -f Makefile.bsd-wrapper install
# make -f Makefile.bsd-wrapper clean
# make -f Makefile.bsd-wrapper depend
# make -f Makefile.bsd-wrapper
# make -f Makefile.bsd-wrapper install
```

Puis lancer un make build classique.

5.11.10 - Quel est le meilleur moyen de mettre à jour `/etc`, `/var`, et `/dev` ?

En tant que politique, les logiciels dans l'arbre OpenBSD ne modifient pas de fichiers dans `/etc` automatiquement. Ceci signifie que les modifications à réaliser incombent *toujours* à l'administrateur. Les mises à niveau n'y font pas exception. Pour mettre à jour les fichiers dans ces répertoires, déterminez tout d'abord les changements qui ont été opérés sur les fichiers de la base (distribution) et répétez manuellement ces changements.

Par exemple, pour voir les fichiers dans l'arbre qui ont changé récemment, faîtes un :

```
# cd /usr/src/etc
# ls -lt |more
```

Pour voir tous les changements dans le répertoire `/etc` entre des versions arbitraires d'OpenBSD, vous pouvez utiliser CVS. Par exemple, pour voir les changements entre 3.8 et 3.9, faîtes un :

```
# cd /usr/src/etc
# cvs diff -u -rOPENBSD_3_8 -rOPENBSD_3_9
```

Pour voir les changements entre 3.9 et *-current* ("HEAD"), utilisez :

```
# cd /usr/src/etc
# cvs diff -u -rOPENBSD_3_9 -rHEAD
```

Le script /dev/MAKEDEV n'est pas mis à jour automatiquement lors du processus de compilation, mais il est cependant installé comme partie de la mise à jour binaire. En règle générale, c'est une bonne idée que de copier (si besoin est) et de lancer ce script depuis votre arbre des sources lors de la mise à niveau :

```
# cd /dev
# cp /usr/src/etc/etc.`machine`/MAKEDEV ./
# ./MAKEDEV all
```

Une fois que vous avez identifié les changements, appliquez à nouveau ceci à votre arbre local, en préservant toute configuration que vous auriez pu faire.

Les changements typiques de `/etc` à considérer, entre les release concernent :

- Additions à `/etc/protocols` et `/etc/services`
- Nouveaux sysctls (regardez `/etc/sysctl.conf`)
- Changements aux travaux de cron par défaut. Regardez `/etc/daily`, `/etc/weekly`, `/etc/monthly`, et `/etc/security`
- Tous les scripts rc, incluant netstart
- Les changements de périphériques, regardez ci-dessus
- Les changements de la hiérarchie des fichiers dans `/etc/mtree`, voyez ci-dessus
- Les nouveaux utilisateurs (`/etc/passwd`) et groupes (`/etc/group`)

Ces changements sont résumés dans upgrade39.html (pour aller vers 3.9-release) ou current.html (pour aller vers *-current*).

5.11.11 - Y a t'il un moyen facile de faire des changements sur tous les fichiers de la hiérarchie ?

De temps en temps, des fichiers et répertoires sont ajoutés, ou supprimés dans le fichier hierarchy. Aussi, le possesseur de l'information pour les portions du système de fichiers peut changer. Un moyen facile de s'assurer que votre hiérarchie est à jour est d'utiliser l'outil mtree(8).

Tout d'abord, procurez vous les dernières sources, en faisant :

```
# cd /usr/src/etc/mtree
# install -c -o root -g wheel -m 600 special /etc/mtree
# install -c -o root -g wheel -m 444 4.4BSD.dist /etc/mtree
# mtree -qdef /etc/mtree/4.4BSD.dist -p / -u
```

Votre hiérarchie de fichiers devrait à présent être à jour.

5.11.12 - La compilation de l'environnement X a échoué !

Une cause très commune d'échec de la compilation sur un système i386 est l'oubli de l'installation des paquetages "tcl" et "tk" avant cette opération, tel que précédemment indiqué dans la section Compilation de X.

5.11.13 - Puis-je cross-compiler ? Pourquoi pas ?

Les outils de cross-compilation sont présent dans le système, et sont utilisés par les développeurs lors de la création d'un nouveau port. Cependant, ils ne sont pas maintenus pour une utilisation généralisée.

Lorsque les développeurs créent le support d'une nouvelle plateforme, l'un des premiers gros tests est la compilation native. Compiler le système depuis les sources augmente très fortement la charge de l'OS et de la machine, et permet de voir à quel point le système fonctionne correctement. Pour cette raison, OpenBSD réalise les compilations sur les plateformes que la compilation concerne, ceci est appelé le "native building". Sans compilation native, il serait bien plus difficile de s'assurer que les différentes plateformes sont stables et qu'elles ne font pas que booter.

[Index de La FAQ] [Section 4 - Guide d'Installation] [Section 6 - Le réseau]


www@openbsd.org

$OpenBSD: faq5.html,v 1.72 2006/10/12 08:24:27 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 mieux comprendre 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 en fonction du type de la carte réseau, et non en fonction 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
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, ifconfig(8) nous fourni ici beaucoup plus d'informations que nécessaire. Mais nous pouvons tout de même voir notre interface. Dans l'exemple précédent, la carte est déjà 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. En ce basant sur les 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 172.21.0.31 255.255.255.0 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). Si vous utilisez DHCP, soyez sûr de lire 6.4 - DHCP ainsi que le manuel resolv.conf.tail(5). Mais pour une utilisation standard, voici un exemple.

Dans celui-ci, 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 déjà 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 déjà 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 OpenBGPD (qui fait partie du projet OpenBSD), routed(8), mrtd, zebra et quagga. Sous OpenBSD, zebra et mrtd sont disponibles en tant que ports. OpenBGPD et routed sont inclus dans le système de base. 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/netstart(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 référerons 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 réseau 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 le client DHCP, 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.

En changeant les options dans le fichier dhclient.conf(5), vous indiquez au client DHCP la façon de créer votre fichier resolv.conf(5). Le client DHCP écrase toutes les données déjà présentes dans resolv.conf(5) en utilisant les informations envoyées par le serveur DHCP. De fait, tous les changements que vous aurez apportés manuellement seront perdus.

Il existe deux mécanismes pour empêcher cela :

- `OPTION MODIFIERS` (**default**, **supersede**, **prepend**, et **append**) vous permet de modifier n'importe quelles options du fichier dhclient.conf(5).
- `resolv.conf.tail(5)` vous permet d'ajouter ce que vous souhaitez au fichier resolv.conf(5) créé par dhclient(8).

Imaginions que vous souhaitiez utiliser le client DHCP mais que vous vouliez ajouter `lookup file bind` au fichier resolv.conf(5) créé par dhclient(8). Il n'existe pas de telle option dans le fichier `dhclient.conf` et vous devrez donc utiliser `resolv.conf.tail` pour préserver ce changement.

```
# echo "lookup file bind" > /etc/resolv.conf.tail
```

A présent, votre fichier resolv.conf(5) devrait inclure "lookup file bind" à la fin.

```
nameserver 192.168.1.1
nameserver 192.168.1.2
lookup file bind
```

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=""`. 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 fxp0
```

La ligne invoquant `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.

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 au manuel dhcp-options(5) si vos clients DHCP ont besoin de plus d'options.

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) - implémentation noyau du démon ppp.
- ppp(8) - implémentation ppp en espace utilisateur ("userland").

Chacune des implémentations (ppp et pppd) offrent des fonctionnalités similaires de différentes façons. pppd utilise le pilote ppp(4) inclus dans le noyau alors que ppp utilise tun(4) dans l'espace utilisateur. Ce document ne couvrira que l'implémentation en espace utilisateur du démon PPP car il est plus simple à utiliser et diagnostiquer. 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 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.

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 événements. 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.

Nous pouvons maintenant continuer 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 complète.

```
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*.

A la place des méthodes traditionnelles de connexion, certains FAI utilisent à présent l'authentification CHAP ou PAP. Si c'est le cas, notre configuration sera un peu différente:

```
myisp:
set phone 1234567
```

```
set authname ppp
set authkey ppp
set login
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
```

Dans l'exemple précédent, nous spécifions notre nom d'utilisateur (ppp) et notre mot de passe à l'aide des options authname et authkey, respectivement. Il n'est pas nécessaire de spécifier le type d'authentification CHAP ou PAP utilisé, car il sera négocié automatiquement. "set login" signifie simplement de tenter de se connecter en utilisant le nom d'utilisateur et le mot de passe précisés précédemment.

Utiliser PPP(8)

Maintenant que *ppp.conf* est configuré, nous pouvons essayer d'initier une connexion vers notre FAI. Je détaillerais 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.

Si les commandes précédentes ne fonctionnent pas, lancez */usr/sbin/ppp* sans options, ce qui vous placera en mode interactif. Les options peuvent être spécifiées une par une afin de diagnostiquer les erreurs et autres problèmes. En utilisant la configuration précédente, ppp enregistrera les événements dans /var/log/ppp.log. Ce fichier de log ainsi que la page de manuel contiennent des informations utiles.

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 configurations sont disponibles ici :

- ppp.linkup
- ppp.linkdown

variations sur ppp(8)

De nos jours, beaucoup de FAI offrent des services xDSL plus rapides que les méthodes traditionnelles de connexion. Ceux-ci incluent des variantes de type ADSL et SDSL. Bien qu'aucun appel à distance n'est lieu, la connexion reste basée sur le protocole point-à-point. En voici des exemples:

- PPPoE
- PPPoA
- PPTP

PPPoE/PPPoA

Le protocole d'encapsulation de PPP dans Ethernet (PPPoE - "Point to Point Protocol over Ethernet") permet d'envoyer des paquets PPP à l'intérieur de trames Ethernet. Le protocole d'encapsulation de PPP dans ATM (PPPoA - "Point to Point Protocol over

ATM") est généralement utilisé dans les réseaux ATM tels qu'au Royaume Uni ou en Belgique.

En résumé cela signifie que vous pouvez établir une connexion vers votre FAI à l'aide d'une carte Ethernet standard et d'un modem DSL-Ethernet (à l'inverse des modems USB).

Si vous possédez un modem compatible PPPoE/PPPoA, il est possible de le configurer afin qu'il initie lui-même la connexion. Dans le cas contraire, si le modem est capable de fonctionner en mode "bridge", il est possible d'activer ce mode afin que celui-ci fasse "transiter" les paquets vers une machine exécutant un logiciel PPPoE (voir plus loin).

Sous OpenBSD, l'interface logicielle PPPoE/PPPoA principale est pppoe(8) , qui est l'implémentation en espace utilisateur (de la même manière que ppp(8) , précédemment décrit). Une implémentation noyau de PPPoE, pppoe(4) , a été ajoutée dans OpenBSD.

PPTP

Le "Point to Point Tunneling Protocol" (PPTP) est un protocole propriétaire Microsoft. Un client pptp s'interfaçant avec pppd(8) est disponible et est capable de se connecter à un réseau privé virtuel PPTP (VPN) utilisé par certains fournisseurs d'accès câble ou xDSL. Le logiciel pptp quant à lui doit être installé en utilisant les paquets ou les ports. De plus amples instructions sur la façon de mettre en place et d'utiliser pptp sont disponibles dans la page de manuel installée avec le paquet pptp.

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 paramètre sysctl. `sysctl net.inet.ip.directed-broadcast=1` active cette option. Lisez la section sur les "attaques de type smurf" 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 paramètre sysctl pour cela. 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
```

6.6.4 - Comment augmenter les performances sur les liens à grande vitesse et à fort trafic ?

Si vous avez des soucis de performances sur des liens WAN à grande vitesse lorsque vous transférez beaucoup de données, vous devriez pouvoir optimiser ces performances en éditant les valeurs sysctls suivantes :

```
net.inet.tcp.recvspace
net.inet.tcp.sendspace
```

Essayez une valeur comme 65536 au lieu de 16384. Notez que très peu de personnes verront un gain réel. N'ajustez ces valeurs que si vous obtenez des performances en deçà de vos attentes.

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 afin d'interdire 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. Sous OpenBSD, portmap(8) est désactivé par défaut, vous devez donc ajouter la ligne

```
portmap=YES
```

dans le fichier rc.conf.local(8) pour le lancer au démarrage. Il peut également être lancé manuellement :

```
# /usr/sbin/portmap
```

Cette configuration 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 */etc/exports* 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 et utilisant un masque de 255.255.255.0 seront autorisés à monter ce système de fichiers. Ceci est important pour certains serveurs accessibles dans 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 par 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 déjà 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 beaucoup 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 adresse IP ; mount résoudra les noms d'hôtes) :

```
# mount -o ro -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 ro 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 ro,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.

6.9 - 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 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
```

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 quelque chose 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 déclarations 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 incompatibles avec PXE ; s'il est équipé de tels contrôleurs, un système OpenBSD/i386 ou amd64 ne pourra pas être démarré 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 plateformes 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 (cela n'étant 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. Si vous tapez simplement "`bsd.rd`" pour récupérer le fichier `bsd.rd` 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) 1995-2006 OpenBSD.  All rights reserved.  http://www.OpenBSD.org

OpenBSD 3.9 (RAMDISK_CD) #1025: Thu Mar  2 02:43:29 MST 2006
  ...
```

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, 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'informations 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). Peu importe les adresses IP et MAC de l'interface physique locale, les paquets envoyés vers l'adresse CARP reviennent avec les informations CARP.

A intervalles paramétrables, 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 Sun Fire V120 rapide 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 avec une adresse IP fixe, bien que depuis l'introduction de la directive carpdev il n'y ait plus besoin d'adresse IP pour les interfaces physiques. 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 paramètres sysctls.

La première variable sysctl, `net.inet.carp.allow`, définit 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 originaire 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). Les commandes `advbase` et `advskew`, spécifiques à CARP, définissent l'intervalle entre les annonces CARP. La formule (en secondes) est `advskew` divisée par 256 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.

Enfin, `carpdev` spécifie l'interface appartenant à ce groupe CARP particulier. Par défaut, n'importe quelle interface possédant une adresse IP dans le même sous-réseau que celui assigné à l'interface CARP sera utilisée.

Voyons l'emploi de ces différents paramètres dans une configuration simple. Admettons que vous déployez 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 carpdev fxp0 \
    192.168.0.7 netmask 255.255.255.0
```

créent l'interface carp0 et lui donnent un vhid de 1, le mot de passe *tyrell*, l'adresse IP 192.168.0.7 de masque 255.255.255.0 et assignent fxp0 en tant qu'interface membre. 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 carpdev
fxp0
```

Notez qu'il faut ajouter l'adresse de diffusion ("broadcast") en plus du vhid et du mot de passe. Ce champ est nécessaire et est souvent cause d'erreurs en cas d'oubli.

Faites la même chose sur *pris*. Le système démarrant en premier son interface CARP deviendra le maître (en admettant que "preempt" est désactivé ; sinon, c'est l'inverse qui se produit).

Mais admettons que vous ne déployez pas cette solution à partir de zéro. *Rachael* était déjà en place à l'adresse 192.168.0.7. Comment allez- vous gérer cette situation ? Heureusement, CARP sait gérer ce genre de situation. Vous assignez simplement l'adresse à l'interface CARP et laissez vide l'adresse de l'interface physique spécifiée par le mot-clef `carpdev'. Cependant, il est plus propre d'avoir une adresse IP différente pour chaque système, 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 carpdev fxp0 \
    192.168.0.7 netmask 255.255.255.0
```

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

```
pris# ifconfig carp1 create
pris# ifconfig carp1 vhid 2 pass bryant carpdev fxp0 \
    192.168.0.7 netmask 255.255.255.0
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'adresse IP d'origine 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

6.12 - Utiliser OpenNTPD

Une horloge précise est importante pour plusieurs applications informatiques. Cependant, plusieurs personnes ont remarqué que leur montre à 5 euros est plus précise que leur ordinateur à 2000 euros. Certes, connaître la date et l'heure est une chose importante. Mais souvent, il est plus important de synchroniser plusieurs machines afin qu'elles s'accordent sur la date et l'heure actuelles. Durant un temps, ntp.org a produit une application "Network Time Protocol" (RFC1305, RFC2030), disponible dans le système des ports, et qui peut être utilisée pour synchroniser les horloges des machines à travers Internet. Cependant, c'est un programme difficile à configurer, dont le code est difficile à auditer, et qui nécessite une quantité de mémoire vive conséquente. En somme, c'est un programme qui rend bien des services à certaines personnes mais qui est loin d'être une solution pour tous les utilisateurs.

OpenNTPD est une tentative pour résoudre certains de ces problèmes en créant un programme compatible NTP qui soit facile à administrer, sûr, simple et qui vous permette d'avoir une horloge exacte sur votre ordinateur. Le service ntpd(8) d'OpenBSD est contrôlé travers le fichier de configuration `/etc/ntpd.conf` facile à comprendre.

En activant tout simplement ntpd(8) dans rc.conf.local, l'horloge de votre ordinateur avancera légèrement et restera synchronisée avec les serveurs de pool.ntp.org, une collection de serveurs de temps publics. Une fois votre horloge réglée de manière précise, ntpd la gardera à un haut degré d'exactitude. Cependant, si votre horloge est décalée de plus de quelques minutes, il est *vivement* recommandé de la régler plus précisément au démarrage car la synchronisation d'une horloge très décalée pourrait prendre des jours, voire des semaines. Vous pouvez la régler manuellement en utilisant l'option "`-s`" de ntpd ou tout autre moyen permettant de mettre l'horloge de votre système à l'heure.

6.12.1 - "Mais OpenNTPD n'est pas aussi exact que l'application de ntp.org !"

C'est probablement vrai. Ce n'est pas un des objectifs de la conception d'OpenNTPD. Ce programme a été d'abord conçu pour être libre, simple, fiable et sécurisé. Si vous avez réellement besoin d'une précision de l'ordre de la micro- seconde, alors OpenNTPD n'est pas pour vous. Utilisez plutôt le ntpd de ntp.org. Ce dernier restera disponible dans les ports et comme paquetage. Nous n'avons aucun plan ou souhait de rendre OpenNTPD bouffi en y intégrant toutes les fonctionnalités possibles et imaginables.

6.12.2 - "Quelqu'un a prétendu qu'OpenNTPD était 'nuisible' !"

Certaines personnes n'ont pas compris le but d'OpenNTPD, à savoir un moyen de synchroniser l'horloge de votre ordinateur qui soit simple, sécurisé et facile à maintenir. Si la maintenance d'un temps aussi exact que possible est une chose importante, un certain nombre d'utilisateurs ont rapporté de meilleurs résultats avec OpenNTPD par rapport au programme ntpd de ntp.org. Si la sécurité est importante, le code d'OpenNTPD est beaucoup plus lisible (et donc auditable) et a été écrit en utilisant les appels de fonction natifs d'OpenBSD tels que strlcpy, au lieu d'utiliser des fonctions plus portables telles que strcpy. Il a été écrit avec des principes sécurité dès le début. La sécurité n'a pas été ajoutée par la suite. Si le fait d'amener autant de personnes que possible à utiliser la synchronisation du temps est une chose précieuse, OpenNTPD rend une telle entreprise très facile pour beaucoup de personnes. Si cela est "nuisible", alors nous souhaitons ce genre de nuisance.

Il y a des applications pour lesquelles le programme ntpd de ntp.org est plus approprié, cependant nous croyons qu'OpenNTPD est plus que suffisant pour une large majorité d'utilisateurs.

Une réponse plus complète donnée par l'un des mainteneurs d'OpenNTPD et concernant ce fait est disponible ici.

6.12.3 - Pourquoi mes autres machines ne peuvent pas se synchroniser avec OpenNTPD?

Par défaut, ntpd(8) n'écoute sur aucune adresse. Ainsi, pour l'utiliser en tant que serveur, vous devez décommenter la ligne "`#listen on *`" dans /etc/ntpd.conf et redémarrer le service ntpd(8). Bien entendu, si vous préférez écouter sur une adresse particulière plutôt que sur toutes celles disponibles, remplacez "*" par l'adresse correspondante.

Lorsque ntpd(8) écoute il se peut que les autres machines ne puissent toujours pas se synchroniser ! Lorsque que le service ntpd(8) vient juste de démarrer (par exemple si vous venez de le redémarrer suite à un changement dans ntpd.conf) il se peut qu'il refuse la synchronisation des clients en attendant que lui-même ajuste sa propre horloge a un certain degré de stabilité. Lorsque ntpd(8) considère que son information temporelle est stable, il l'annonce par le message "clock now synced" dans `/var/log/daemon`. Même si l'horloge système est assez correcte au démarrage, cela peut prendre jusqu'à 10 minutes avant d'être synchronisé et des heures voire des jours si celle-ci est décalée.

6.13 - 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
- ath(4) pilote pour chipset Atheros IEEE 802.11a/b/g.
- iwi(4) Intel PRO/Wireless 2200BG/2225BG/2915ABG IEEE 802.11a/b/g.
- ipw(4) Intel PRO/Wireless 2100 IEEE 802.11b.
- atu(4) Atmel AT76C50x USB IEEE 802.11b.
- ral(4) et ural(4) [USB] Ralink Technology RT25x0 IEEE 802.11a/b/g.
- rtw(4) Realtek 8180 IEEE 802.11b.

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 les détails précis). Certaines cartes 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.

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.

Notez que pour utiliser les cartes à base de chipset Intel, vous devrez récupérer les firmwares dont Intel refuse la libre distribution, ce qui empêche qu'ils soient inclus dans OpenBSD. Contactez Intel afin de leur faire savoir ce que vous pensez de cette situation ou de les informer sur le produit que vous avez acheté à la place.

D'autres constructeurs tels que Broadcom, Texas Instrument et Connexant se sont activement battus contre nos tentatives de développer des pilotes libres pour leurs produits. Nous vous encourageons à respecter leur souhait et à ne pas acheter leurs produits. Realtek, Ralink, Atmel et ADMtek font de bons produits et supportent la communauté Open Source dans sa volonté de créer des pilotes libres, ils ont gagné notre support.

[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

$OpenBSD: faq6.html,v 1.71 2006/10/29 10:58:52 jufi Exp $

[Index de La FAQ] [Section 6 - Administration réseau] [Section 8 - Questions diverses]

7 - Contrôles du clavier et de l'affichage

Table des matières

7.1 - Comment puis-je redéfinir le clavier ? *(wscons)*

En utilisant le pilote console wscons(4) : alpha, amd64, cats, hppa, i386, mac68k, macppc, sparc, sparc64 et vax.

Avec wscons(4), plusieurs options peuvent être configurées en utilisant l'utilitaire wsconsctl(8). Par exemple, pour changer la configuration des touches du clavier avec wsconsctl(8) vous devez exécuter ceci :

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

Dans l'exemple suivant, on redéfinit la touche "Caps Lock" en "Control L" :

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

7.2 - OpenBSD dispose t-il d'un support de la souris en mode console ?

Pour les plates-formes alpha, amd64 et i386, OpenBSD fournit wsmoused(8), un port du démon moused(8) de FreeBSD. Il peut être lancé au démarrage en éditant la ligne appropriée dans rc.conf(8).

7.3 - Effacer la console à chaque fois qu'un utilisateur se déconnecte.

Pour ce faire, il vous faut ajouter une ligne dans `/etc/gettytab(5)`. Changer la section :

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

Ajouter une ligne "`:cl=\E[H\E[2J:`" à la fin, ce qui donne :

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

7.4 - Accéder au tampon de la console *(amd64, i386, quelques Alpha)*

Sur quelques plates-formes, OpenBSD offre la possibilité de revenir en arrière dans la console. Ceci permet de voir les informations inscrites précédemment sur votre écran. Pour monter et descendre dans le tampon, utilisez simplement les combinaisons de touches `[SHIFT]+[PGUP]` et `[SHIFT]+[PGDN]`.

La valeur par défaut de pages que vous pouvez consulter en revenant en arrière dans la console est de 8. Ceci est une fonctionnalité du driver vga(4), donc cela ne fonctionnera pas sans carte vga quelque soit la plate-forme (beaucoup de systèmes Alpha ont une carte vidéo de type TGA).

Par manque d'espace, les noyaux d'installation ne fournissent pas la possibilité de revenir en arrière dans la console. Changer de console effacera le tampon de retour en arrière.

7.5 - Comment puis-je changer de console ? *(amd64, i386, Zaurus, quelques Alpha)*

Sur amd64, i386 et quelques systèmes Alpha, avec les cartes vga(4), OpenBSD fournit six terminaux virtuels par défaut, allant de /dev/ttyC0 à /dev/ttyC5. ttyC4 est réservée pour l'utilisation de X, laissant donc cinq consoles texte. Vous pouvez passer de l'une à l'autre en utilisant `[CTRL]+[ALT]+[F1]`, `[CTRL]+[ALT]+[F2]`, `[CTRL]+[ALT]+[F3]`, `[CTRL]+[ALT]+[F4]` et `[CTRL]+[ALT]+[F6]`.

L'environnement X utilise ttyC4, `[CTRL]+[ALT]+[F5]`. Lorsque vous utilisez X, les combinaisons `[CTRL]+[ALT]+[F`*n*`]` vous basculerons sur les consoles textes ; `[CTRL]+[ALT]+[F5]` vous ramènera à l'environnement graphique.

Si vous souhaitez avoir plus de console virtuelles que le nombre par défaut, utilisez la commande wsconscfg(8) pour créer des consoles pour ttyC6, ttyC7 et au delà. Par exemple :

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

Créera un terminal virtuel pour ttyC6, accessible par `[CTRL]+[ALT]+[F7]`. N'oubliez pas d'ajouter cette commande à votre fichier `rc.local(8)` si vous souhaitez cette console supplémentaire au prochain redémarrage de votre ordinateur.

Notez que vous n'obtiendrez pas une invite "`login:`" sur la nouvelle console virtuelle tant que vous ne l'aurez pas mis à "on" dans `/etc/ttys(5)`, puis au choix redémarré ou envoyé un signal HUP à init(8) par le biais de kill(1).

Sur le Zaurus, deux terminaux virtuels (/dev/ttyC0 and /dev/ttyC1) sont disponibles par défaut. Ils sont accessibles à l'aide des combinaisons de touches `[ALT]+[CALENDAR]` et `[ALT]+[ADDRESS]` (La touche `[ALT]` se situe à gauche de la touche `[CTRL]` key).

7.6 - Comment puis-je utiliser une résolution console de 80x50 ? *(amd64, i386, quelques Alpha)*

Les utilisateurs de systèmes i386, amd64 et Alpha (VGA) ont normalement une console de 25 lignes de 80 caractères. Cependant, beaucoup de cartes vidéo VGA sont capables d'afficher une résolution supérieure à 50 lignes de 80 caractères.

Premièrement, une police supportant la résolution désirée doit être chargée en utilisant la commande wsfontload. L'écran standard 80x25 utilise des polices de 8x16 pixels, pour doubler la résolution verticale, nous allons devoir utiliser des polices de 8x8 pixels.

Ensuite, nous allons devoir supprimer et recréer une console virtuelle de la résolution désirée en utilisant la commande wsconscfg.

Ceci peut être fait automatiquement en ajoutant les lignes suivantes à la fin de votre fichier `rc.local` :

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

Comme pour toutes les configurations de votre système, il est recommandé de prendre le temps de lire les pages de manuels (man) afin de comprendre ce que font les commandes ci-dessus.

La première commande ci-dessus charge une police 8x8. La seconde supprime la console 5 (qui aurait été accessible via `[CTRL]+[ALT]+[F6]`). La troisième crée une nouvelle console 5 avec une résolution de 50 lignes de 80 caractères chacune. Si vous procédez de la sorte, vous verrez votre console principale, ainsi que les trois autres consoles par défaut, en mode standard 80x25, mais une nouvelle console 5 en mode 80x50 accessible via `[CTRL]+[ALT]+[F6]`.

Rappelez vous que `[CTRL]+[ALT]+[F1]` est la console 0 (ttyC0). Si vous désirez modifier les autres consoles, réalisez simplement les étapes de suppression et d'ajout des consoles pour lesquelles vous désirez une résolution de 80x50.

Vous devriez éviter de modifier la console 4 (ttyC4, `[CTRL]+[ALT]+[F5]`), qui est utilisée par X en tant que console graphique. Il n'est pas non plus possible de changer la résolution de la première console (exemple ttyC0).

Comme certains ont pu le remarquer, toutes ces commandes peuvent être tapées sur la ligne de commande, en tant que root, ou (mieux), en utilisant sudo(8).

Note : ceci ne fonctionnera pas avec toutes les cartes vidéo. Malheureusement, toutes les cartes vidéo ne supportent pas les polices fournies qui sont nécessaires à wscons pour parvenir à configurer le mode texte 80x50. Dans ces cas là vous devriez reconsidérer le fait d'utiliser X.

7.7 - Comment puis-je utiliser une console série ?

Il y a plusieurs raisons pour que vous souhaitiez utiliser une console série sur votre système OpenBSD :

- Enregistrer la sortie de la console (pour de la documentation).
- Administration distante.
- Maintenance simplifiée d'un grand nombre de machines.
- Obtenir un très utile dmesg de machines depuis lesquelles cette obtention aurait été difficile.
- Obtenir une sortie "trace" et de "ps" si votre système se crash, pour que les développeurs aient une chance de régler le problème.

OpenBSD supporte les consoles séries sur la plupart des plates-formes, cependant certains détails varient considérablement entre plates-formes.

Notez que l'interfaçage série n'est PAS une tâche triviale -- vous serez souvent obligé d'utiliser des câbles peu communs, et les ports ne sont pas standardisés entre machines, dans certains cas, parfois même incompatibles sur une machine. Nous considérons que vous savez comment utiliser le câble adéquat entre votre ordinateur et le matériel faisant office de terminal série. Un tutorial complet sur l'interfaçage série sortirait du cadre de ce document, cependant, nous vous donnons un petit conseil : ce n'est pas parce que les prises se connectent que cela fonctionnera.

Modification de */etc/ttys*

Deux conditions doivent êtres réunies pour obtenir une console série fonctionnelle sur un système OpenBSD. Premièrement, vous devez indiquer à OpenBSD d'utiliser le port série comme console pour les modes "status" et "single user". Cette étape dépend de la plate-forme que vous utilisez. Ensuite, vous devez activer votre port série comme terminal interactif, de façon à ce qu'un utilisateur puisse s'y connecter en mode multi-utilisateurs. Cette partie est relativement similaire sur les différentes plates-formes, et est détaillée ici.

Les sessions de terminaux sont contrôlées par le fichier */etc/ttys*. Avant qu'OpenBSD ne vous présente une invite "`login:`" sur un périphérique, celui-ci doit être activé dans */etc/ttys*, après tout, il y à d'autres utilisations possibles que celle d'un terminal pour un port série. Sur les plates-formes qui ont un clavier et un écran attaché comme console, le terminal série est désactivé par défaut. Nous utiliserons la plate-forme i386 comme exemple. Dans ce cas vous devez éditer la ligne suivante :

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

ainsi (par exemple) :

```
tty00   "/usr/libexec/getty std.9600"   vt220   on secure
```

Ici, `tty00` est le port série que nous utilisons comme console. `vt220` représente l'entrée termcap(5) équivalente à votre terminal (d'autres options similaires pourraient inclure `vt100`, `xterm`, etc.). Le "`on`" active le *getty* pour le port série afin que l'invite "`login:`" soit présentée, le "`secure`" permet une connexion root (uid 0) sur cette console (ce qui peut être, ou pas, ce que vous souhaitez), et le "`9600`" est le débit en baud du terminal. Essayez de ne pas augmenter le débit en baud du terminal jusqu'au maximum de ce que votre matériel peut supporter car cela risque de poser plus de problèmes que de bénéfices. La plupart des systèmes ont une vitesse "par défaut" (supportée par défaut par la ROM de démarrage et/ou le chargeur de démarrage, généralement 9600), utilisez celle-ci à moins que vous n'ayez une raison valable de faire autrement.

Notez que vous pouvez utiliser une console série pour l'installation sans devoir réaliser tout cela étant donné que le système est lancé en mode "single user" et n'utilise pas *getty* pour la connexion.

Sur certaines plates-formes ou configurations, vous devrez démarrer le système en mode "single user" pour faire cette modification si vous n'avez rien d'autre qu'une console série.

amd64 et i386

Pour que la procédure de démarrage utilise le port série comme console, créez ou éditez votre fichier */etc/boot.conf* pour y inclure la ligne :

```
set tty com0
```

afin d'utiliser votre premier port série comme console. La bande passante en baud par défaut est 9600bps, ceci peut être changé par une ligne du fichier */etc/boot.conf* en utilisant l'option stty. Ce fichier est stocké dans votre média de démarrage, qui peut aussi être votre disquette d'installation, la commande peut être entrée à l'invite `boot>` depuis le chargeur de démarrage de second stage OpenBSD pour une première utilisation de la commande série.

notes pour amd64 et i386 :

- OpenBSD numérote les ports séries en commençant à *tty00*, les labels DOS/Windows commencent eux à *COM1*. Gardez donc à l'esprit que *tty02* est *COM3*, et non *COM2*
- Quelques systèmes pourront fonctionner sans carte vidéo présente dans la machine, mais sûrement pas tous -- la plupart des systèmes considèrent cela comme une erreur. Certaines machines refuserons même de fonctionner sans clavier attaché.
- Certains systèmes sont capables de rediriger les activités BIOS du clavier et de l'écran sur un port série au travers d'une option de configuration, pour que la machine puisse être complètement maintenue par le port série. Les résultats que vous obtiendrez varieront, quelques BIOSs peuvent empêcher le chargeur de démarrage de voir le port série, et donc le noyau ne saura pas qu'il doit l'utiliser. Certains BIOS ont une option "Continuer la redirection de la console après le POST" (Power On Self Test), cela devra être désactivé, pour que le chargeur de démarrage et le noyau puissent disposer de leur propre console. Malheureusement cette option n'est pas universelle.
- Les ordinateurs compatibles PC, ne sont pas pensés dans le but d'être utilisés depuis une console série, à la différence d'autres plates-formes. Même sur les systèmes qui supportent une console série, celle-ci doit être activée dans la configuration BIOS -- et si les informations de configuration de celui-ci sont corrompues, vous serez face à un système qui recherche de nouveau un moniteur et un clavier. Vous devrez généralement, disposer d'un moyen d'obtenir un clavier et un moniteur pour vos systèmes amd64 et i386 en cas d'urgence.
- Vous devrez éditer */etc/ttys* comme décrit ci-dessus.
- Seul le premier port série (com0) est supporté pour une utilisation en console sur amd64 et i386.

SPARC et UltraSPARC

Ces machines sont construites dans le but d'être totalement configurables par console série. Retirez simplement le clavier de la machine et le système lancera une console série.

Notes SPARC et UltraSPARC

- Les ports sur un SPARC sont appelés *ttya*, *ttyb*, etc.
- A la différence des autres plates-formes, il n'est pas nécessaire de faire des changements à */etc/ttys* pour utiliser une console série.
- Les systèmes SPARC/UltraSPARC interprètent le signal BREAK sur le port série de la même manière que la commande STOP-A, et renvoient le système à l'invite "Forth", stoppant toute application et le système lui même à ce point. Cela est intéressant lorsque souhaité, mais malheureusement, quelques terminaux série lors de l'arrêt et quelques matériels de switch RS-232 envoient quelques codes à l'ordinateur que celui-ci interprète comme un signal BREAK, arrêtant la machine. Testez cela avant de passer la machine en production.
- Si vous avez un clavier et une souris connectés, vous pouvez toujours forcer l'utilisation de la console série en utilisant les commandes suivantes à l'invite `ok` :

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

Si le clavier et l'écran (ttyC0) sont activés dans */etc/ttys* (ci-dessus), vous pouvez utiliser le clavier et le moniteur dans X.

MacPPC

Les machines MacPPC, sont configurées pour utiliser une console série au travers de "OpenFirmware". L'utilisation des commandes

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

Vous configurera une console série à 57600bps, 8N1.

Notes MacPPC

- Malheureusement, l'utilisation des consoles séries n'est pas directement possible sur la plupart des MacPPCs. Alors que la plupart des machines auront le matériel série, il n'est pas accessible à l'extérieur de la machine. Heureusement, quelques sociétés offrent des matériel tiers pour la plupart des modèles Macintosh qui rendront ce port accessible pour une utilisation de console série (ou autre). Utilisez votre moteur de recherche préféré et recherchez "Macintosh internal serial port".
- Vous devrez positionner `tty00` dans */etc/ttys* à `on` et mettre la vitesse à 57600 plutôt que la valeur par défaut 9600 comme décrit ci-dessus dans le mode "single user" avant de démarrer en mode multi-utilisateurs et obtenir une console série fonctionnelle.

Mac68k

La console série est sélectionnée dans le programme *Booter*, sous le menu "`Options`", puis "`Serial Ports`". Cochez le bouton "Serial Console", puis choisissez le port du modem ou de l'imprimante. Vous aurez besoin d'un modem Macintosh ou d'un câble d'imprimante à attacher à un port série Mac. Si vous souhaitez que ces paramètres soient utilisés par défaut, demandez au programme "Booter" de sauvegarder vos options.

Notes Mac68k

- Le port du modem est *tty00*, celui de l'imprimante *tty01*.
- Le Mac68k n'active pas son port série tant que cela n'est pas demandé, aussi, votre machine n'enverra aucun signal sur la console série avant que la procédure de démarrage OpenBSD ne commence.
- Vous devrez activer le port (*tty00* ou *tty01*) comme indiqué ci-dessus.

7.8 - Comment effacer la console ? (wscons)

Si vous souhaitez effacer votre console après une certaine période d'inactivité sans utiliser X, vous pouvez modifier les variables wscons(4) suivantes :

- **`display.vblank`** réglé à `on` désactivera les rafraîchissements verticaux, ce qui mettra la plupart des moniteurs dans le mode "économie d'énergie". Il faudra plus de temps pour retrouver l'affichage, mais cela réduira la consommation d'énergie. Quand il est réglé à `off`, l'affichage sera désactivé, mais le moniteur continuera de recevoir les signaux de synchronisation verticaux et horizontaux, donc l'affichage sera très vite réactivé.
- **`display.screen_off`** règle le nombre de nettoyages en millisecondes, par exemple, 60000 serait le temps pour une minute.
- **`display.kbdact`** détermine si l'activité du clavier provoquera le rafraîchissement de l'écran. En général, ceci est

souhaité.
- **`display.outact`** détermine si une sortie texte sur l'écran provoquera le retour de l'affichage.

Vous pouvez positionner ces variables en ligne de commande en utilisant l'utilitaire wsconsctl(8) :

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

ou les positionner de manière permanente en éditant le fichier */etc/wsconsctl.conf* de manière à ce que les paramètres soient repris au prochain redémarrage :

```
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
```

Le vidage est activé à partir du moment où `display.kbdact` ou `display.outact` sont positionnés à "`on`".

7.9 - TOUT CE QUE JE TAPPE A LA CONNEXION EST EN MAJUSCULES !

En réalité, ceci est une fonctionnalité, non un bogue.

Virtuellement toutes les commandes et les noms d'utilisateurs sont entrés en minuscules. Cependant, quelques très vieux terminaux sont simplement incapables d'afficher des caractères en minuscules, ce qui rend leur utilisation difficile, voir impossible à utiliser avec UNIX. Pour pallier à cela, si vous utilisez un nom d'utilisateur entièrement en majuscules, getty(8) assumera que votre terminal ne supporte pas les minuscules, et interprétera simplement tout ce que vous tapez comme des minuscules alors qu'il affichera tout en majuscules. Si votre mot de passe est composé de minuscules et de majuscules, la connexion sera impossible.

L'appui sur CTRL-D à la connexion provoquera l'arrêt de getty(0) et init(8) en relancera un nouveau, qui acceptera correctement les minuscules et les majuscules.

[Index de La FAQ] [Section 6 - Administration réseau] [Section 8 - Questions diverses]


www@openbsd.org

$OpenBSD: faq7.html,v 1.25 2006/10/12 08:24:27 jufi Exp $

[Index de la FAQ] [Section 7 - Contrôle du clavier et de l'affichage] [Section 9 - Migrer vers OpenBSD]

8 - Questions Générales

Table des matières

8.1 - J'ai oublié mon mot de passe root..... Que faire !

Quelques étapes pour y remédier

1. Démarrez en mode "single-user". Pour les architectures i386, saisissez "boot -s" à l'invite de démarrage.
2. montez les partitions.

```
# fsck -p / && mount -uw /
```

3. Si /usr n'est pas la même partition que celui de / (recommandé !), alors vous aurez également besoin de la monter.

```
# fsck -p /usr && mount /usr
```

4. Exécutez passwd(1)
5. Démarrez en mode multi-utilisateurs... et *rappelez-vous* de votre mot de passe !

8.2 - X ne veut pas démarrer, et j'ai de nombreux messages d'erreur

La cause habituelle des problèmes avec X vient souvent de la configuration de machdep.allowaperture avec sysctl(8).

Vous devez éditer */etc/sysctl.conf* and positionner la valeur **machdep.allowaperture=2** (ou **1**, selon votre plateforme). Cela permettra à X d'accéder au pilote d'ouverture (aperture) xf86(4), au prochain redémarrage. Cette valeur ne peut pas être changée après le démarrage. En revanche, elle peut être définie dès l'installation si vous répondez "Y" lorsque l'on vous demande si vous comptez utiliser X Window ("Do you expect to run the X Window System?").

Sous OpenBSD, le pilote d'ouverture (aperture) doit être activé sur les plates-formes alpha, amd64, cats, i386, macppc et sparc64 afin de permettre l'accès à la carte vidéo. D'autres plates-formes utilisent un moyen plus sûr pour gérer le système vidéo et n'ont pas besoin de ceci (qui n'est pas inclut dans leur noyau). Si vous ne prévoyez pas d'utiliser X sur votre système, il est recommandé de ne pas activer le pilote d'ouverture (aperture).

Pour plus d'informations sur la configuration et l'utilisation de X sur votre plate-forme, consultez le fichier`/usr/X11R6/README` de votre installation.

8.3 - Puis-je utiliser le langage de programmation "L" sous OpenBSD ?

Plusieurs langages de programmation communs sont supportés soit au niveau du système de base (et plus spécifiquement dans les jeux de fichiers `baseXX.tgz` et `compXX.tgz`), soit au niveau du système de paquetages et de ports. Il est recommandé d'installer le jeu de fichiers nécessaire ou le paquetage contenant le compilateur spécifique que vous souhaitez employer au lieu de l'installation à partir des sources. Pour certains compilateurs, l'installation à partir des sources requièrent d'importantes ressources système. Cette opération est souvent inutile sauf si vous avez des besoins spécifiques ou aucun paquetage n'est disponible.

Le tableau suivant cherche à fournir une vue d'ensemble des compilateurs pour différents langages, où vous pouvez les trouver et si des problèmes ou des limitations relatifs à leur utilisation existent. Certains de ces compilateurs ne sont disponibles que pour certaines plates-formes. Pour voir les plates-formes sur lesquelles fonctionne tel ou tel compilateur, il suffit d'examiner un résultat de recherche dans l'arborescence des ports et de noter ce qui est mentionné au niveau de "Archs". Une autre méthode consiste à inspecter directement le contenu du Makefile du port et d'y chercher `ONLY_FOR_ARCHS`, `NOT_FOR_ARCHS`, `BROKEN`, etc.

Remarque : Pour améliorer l'utilisation, cet article fournit une liste alphabétique sans distinguer entre différentes catégories de langages de programmation. Ceci n'est en aucun cas une liste exhaustive de tout ce qui est disponible ou peut fonctionner sous OpenBSD. Si vous relevez des inexactitudes ou des problèmes autres que ceux listés ci-dessous, prière de nous en faire part.

Langage	Où ?	Remarques
Awk	`base39.tgz`, awk(1)	
	lang/gawk	GNU awk

C, C++	`comp39.tgz`, gcc(1)	Les compilateurs C/C++ dans le système de base ont été audités et contiennent plusieurs améliorations sécurité (telles que ProPolice) activées par défaut. Merci de bien vouloir consulter gcc-local(1) pour de plus amples détails. Ces compilateurs émettrons des alertes lors de l'utilisation de fonctions non sûres telles que sprintf(), strcpy(), strcat(), tmpnam(), etc. La plupart des plates-formes utilisent gcc 3.3.5. Les autres restent en gcc 2.95.3 pour le moment.
C, C++	lang/gcc	Ces compilateurs n'ont pas bénéficié d'un audit sécurité et ne contiennent pas les améliorations sécurité que possèdent les compilateurs du système de base. Les compilateurs sont renommés `egcc`, `eg++`, etc. pour éviter la confusion avec ceux du système de base.
Caml	lang/ocaml	Objective Caml
COBOL	lang/open-cobol	
Fortran	`comp39.tgz`, g77(1)	Support du Fortran 77 uniquement.
	lang/gcc	Fortran 95 est aussi supporté par `egfortran` dans gcc 4.0 et supérieur. Ce nouveau compilateur est disponible en tant que sous-paquetage (g95) de gcc.
Haskell	lang/ghc	
	lang/nhc98	
Java	devel/jdk	JDK Sun - aucun paquetage disponible; consultez les instructions ci-dessous.
	lang/jikes	Compilateur rapide, fonctionne bien. Il nécessite un "run-time jar", la version bytecode de toute l'API standard.
	devel/eclipse	IDE très fourni; Fonctionne avec Sun JDK
Lisp	lang/clisp	
Lua	lang/lua	Bibliothèques Lua additionnelles et outils auxiliaires sont disponibles dans l'arbre des ports.
Perl	`base39.tgz`, perl(1)	De nombreux modules Perl sont disponibles dans l'arbre des ports, regardez donc ceux-là avant d'installer des modules depuis CPAN.
PHP	www/php4	De nombreux sous-paquetages sont disponibles pour des modules PHP différents.
	www/php5	
Prolog	lang/gprolog	Compilateur GNU Prolog.
Python	lang/python	D'autres ports utilisent Python 2.3 par défaut.
Ruby	lang/ruby	
Scheme	lang/scm	
	shells/scsh	
Smalltalk	lang/squeak	
Tcl	lang/tcl	

Compilation du JDK Sun

A cause de la licence restrictive SCSL de Sun, OpenBSD n'est pas en mesure de livrer des paquetages binaires du JDK. cela signifie que vous devez vous-même le compiler depuis les ports. Notez que vous aurez besoin d'une grande quantité de RAM pour que la compilation réussisse.

Les ports du JDK sont dans le sous-répertoire `devel/jdk` de l'arbre des ports. Vous pouvez choisir entre les différentes versions de celui-ci, toutes dans leurs sous-répertoires. Seules les versions 1.3 et 1.4 possèdent un plugin "browser". Lorsque vous tapez `make`, vous verrez un message vous demandant de récupérer manuellement les archives de distribution sur le site de Sun. Avant

que vous ne puissiez faire cela, vous devez vous enregistrer sur ce site, et accepter la licence. C'est la raison pour laquelle le mécanisme des ports ne peut les télécharger automatiquement.

Une fois les archives de distribution et patchs téléchargés, copiez les dans le répertoire `/usr/ports/distfiles` et débutez le processus de compilation en saisissant `make` dans le répertoire du port.

Le JDK requiert un compilateur Java 2 en état de fonctionnement afin devant agir en tant que bootstrap. Si vous trouvez cela fastidieux, n'hésitez pas à demander à Sun pourquoi ils ne fournissent pas une version native pour OpenBSD. L'émulation Linux sur OpenBSD est réservée aux systèmes i386, et le JDK ne peut ainsi être compilé que sur i386. Le mécanisme des ports devrait prendre en charge l'installation des fichiers nécessaires ainsi que le réglage `kern.emul.linux=1`. Pour davantage d'informations, consultez s'il vous plaît Linux emulation dans la page de manuel compat_linux(8), ainsi que FAQ 9 - Exécution des binaires Linux sous OpenBSD. Notez que cette émulation Linux n'est nécessaire que lors de la compilation du JDK, aboutissant à un JDK OpenBSD natif. **Vous n'avez pas besoin de cette émulation pour utiliser le JDK natif.**

Après plusieurs heures, la compilation finira. Continuez simplement avec `make install` pour installer le JDK.

Autres outils de développement

Il existe plusieurs autres outils de développement dans le système de base ou en paquetages ou ports. Quelques exemples :

- Shells Unix : ksh et csh dans le système de base, plusieurs autres dans le sous-répertoire `shells` de l'arbre des ports.
- lint(1) : un programme de vérification C qui a été grandement amélioré sous OpenBSD 3.9. Les versions compatibles lint ("linted versions") des librairies système sont également fournies.
- Utilitaires "make" : Le traditionnel make(1) BSD est dans le système de base et l'arbre des ports contient d'autres versions qui sont nécessaires pour la compilation de certains logiciels.
- outils graphiques : plusieurs outils graphiques populaires (tels que GTK+, Tk, Qt, wxWidgets, ...) ont été portés sur OpenBSD. Vous pourrez les trouver dans le sous-répertoire `x11` de l'arbre des ports.
- Systèmes de contrôle de versions : GNU CVS tel qu'il est utilisé par le projet OpenBSD est dans le système de base et l'arbre des ports en contient quelques autres. Notez qu'OpenCVS est en cours de développement.

8.4 - Qu'est-ce que l'arborescence des ports ?

Veuillez consulter FAQ 15, Utilisation des ports.

8.5 - Qu'est ce que les paquetages?

Veuillez consulter FAQ 15, Gestion des paquetages.

8.6 - Dois-je utiliser les Ports ou les Paquetages ?

Veuillez consulter FAQ 15.

8.8 - Y'a t-il un moyen d'utiliser mon lecteur de disquette alors qu'il n'était pas connecté durant la phase de démarrage ?

Vous devez indiquer au noyau de toujours supposer que le lecteur de disquette est installé, même si il n'est pas détecté pendant la phase de démarrage. Pour ce faire, l'option 0x20 doit être positionnée dans fdc(4). Ceci peut-être fait en utilisant User Kernel

Config ou config(8) pour modifier votre noyau,

```
# config -e -f /bsd
OpenBSD 3.9 (GENERIC) #617: Thu Mar  2 02:26:48 MST 2006
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/GENERIC
Enter 'help' for information
ukc> change fd*
254 fd* at fdc0 drive -1 flags 0x0
change [n] y
drive [-1] ? ENTER
flags [0] ? 0x20
254 fd* changed
254 fd* at fdc0 drive -1 flags 0x20
ukc> q
Saving modified kernel.
#
```

8.9 - Chargeur de démarrage OpenBSD (*spécifique à i386 et amd64*)

Lorsque vous démarrez votre système OpenBSD, vous avez probablement remarqué l'invite de commande.

```
boot>
```

La plupart des utilisateurs n'auront pas à intervenir ici. Le système démarre même si aucune commande n'est entrée. Mais parfois surviennent des problèmes, ou certaines fonctions spéciales sont requises. C'est à cet endroit que ces options doivent être entrées. Avant de vous lancer, vous devriez lire la page de manuel boot(8). Ici, nous allons au-delà de la plupart des commandes utilisées pour le chargeur de démarrage.

Avant de débuter, si aucune commande n'est utilisée, le chargeur de démarrage essaiera automatiquement de démarrer **/bsd**. Si cela échoue, il essaiera **/obsd**, puis en cas de nouvelle échec **/bsd.old**. Vous pouvez spécifier un noyau en tapant :

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

ou

```
boot> b /bsd
```

Ceci démarrera le noyau nommé `bsd` depuis la partition 'a' du premier disque reconnu par le BIOS.

Voici une brève liste d'options que vous pouvez utiliser avec le noyau OpenBSD.

- **-a** : Vous permet de spécifier un autre disque racine après le démarrage du noyau.
- **-c** : Vous permet d'entrer la planification de démarrage. Lisez la section planification de démarrage de la faq.
- **-s** : Vous permet de démarrer en mode utilisateur unique (single user).
- **-d** : Cette option est utilisée pour copier le noyau dans le ddb. Vous devez avoir DDB compilé dans le noyau.

Ces options sont saisies dans le format : **boot [image [-acds]]**

Pour plus d'informations, lisez la page de manuel boot(8).

8.10 - Utilisation de S/Key sur votre système OpenBSD

S/Key est un système d'authentification basé sur des mots de passe à usage unique. Il peut être utile pour les personnes n'ayant pas la possibilité d'utiliser un canal chiffré protégeant leurs identifiants en transit, tel qu'un canal établi par ssh(1).

WARNING: Les systèmes d'authentification basés sur des mots de passe à usage unique protègent uniquement l'information relative à l'authentification. Ils ne permettent pas de se prémunir contre l'usage des écoutes clandestines du réseau pour accéder à des informations confidentielles. De plus, si vous vous connectez à un système sûr A, il est recommandé de le faire depuis un autre système sûr B afin de s'assurer que personne n'accède au système A en enregistrant toutes vos saisies clavier ou en capturant et/ou en forgeant les entrées/sorties de vos terminaux.

Le système S/Key génère une séquence de mots passe à usage unique à partir de la *phrase secrète* d'un utilisateur ainsi qu'un défi transmis par le serveur à l'aide d'une *fonction de hachage* sûre. Le système est sûr uniquement si la phrase secrète n'est jamais transmise sur le réseau. **L'initialisation ou la modification de la phrase secrète DOIT donc se faire à travers un canal sûr**, tel que ssh(1) ou la console.

L'implémentation de S/Key sur OpenBSD peut utiliser plusieurs algorithmes de hachage à sens unique. Les algorithmes suivants sont disponibles :

- md4
- md5
- sha1
- rmd160.

Mise en place de S/Key - Les premières étapes

Pour commencer, le répertoire */etc/skey* doit exister. Si ce répertoire n'existe pas, le super utilisateur doit le créer. Cela peut simplement se faire en tapant :

```
# skeyinit -E
```

Une fois ce répertoire créé, vous pouvez initialiser votre S/Key. Pour cela vous devez utiliser skeyinit(1). Etant donné que skeyinit(1) vous demandera de saisir votre phrase secrète S/Key, vous devez l'exécuter **à travers un canal sûr**, tel que nous l'avons précédemment expliqué. Le programme vous rappellera même de le faire. Avec skeyinit(1), votre mot de passe vous sera d'abord demandé. C'est le même mot de passe que vous utilisez pour vous connectez au système. Une fois authentifié avec votre mot de passe système, vous devrez saisir votre phrase secrète S/Key. Ce n'est **PAS** votre mot de passe système. votre phrase secrète devra comporter au moins 10 caractères. Nous vous suggérons d'utiliser des phrases de plusieurs mots comme mot de passe. Voici un exemple d'ajout d'utilisateur :

```
$ skeyinit
Reminder - Only use this method if you are directly connected
      or have an encrypted channel. If you are using telnet,
      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
```

La ligne *ID ericj skey is otp-md5 100 oshi45820* est particulièrement intéressante. Elle fournit beaucoup d'informations à l'utilisateur comme indiqué ci-après.

- *otp-md5* - Ceci indique quelle fonction à sens unique a été utilisée pour créer votre mot de passe à usage unique (otp).
- *100* - C'est votre numéro de séquence. Ce nombre part de 100 et se décrémente jusqu'à 0. Une fois ce numéro égal à 1, un autre mot de passe doit être généré, et ce grâce à skeyinit(1).
- *oshi45820* - C'est la clé.

Mais le plus important c'est votre prochain mot de passe. Il consiste en 6 petits mots, qui combinés, constituent votre prochain mot de passe à usage unique, espaces compris. Ce mot de passe affiché par skeyinit ne peut être utilisé pour vous connecter (il existe une procédure pour l'utilisation du premier mot de passe à usage unique, lisez skeyinit(1)). Pour être capable de vous connecter, un mot de passe à usage unique correspondant à celui affiché par le processus d'identification doit être calculé en utilisant skey(1). La section suivante montre comment s'y prendre.

Utilisation de S/Key pour se connecter.

Maintenant votre skey initialisé. Vous êtes prêt à vous connecter. Voici l'exemple d'une session utilisant S/Key pour se connecter : Pour exécuter une connexion avec S/Key, vous devez spécifier **:skey** comme votre identifiant.

```
$ 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 3.9 (GENERIC) #617: Thu Mar  2 02:26:48 MST 2006
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.
```

Notez que j'ai mis ":skey" pour mon nom d'utilisateur. Ceci indique à ftpd que je veut être identifié en utilisant S/Key. Certains d'entre vous ont dû remarquer que le numéro de séquence a été changé en *otp-md5 96 oshi45820*. C'est parce que j'ai utilisé S/Key plusieurs fois pour m'identifier. Mais comment avoir votre mot de passe à usage unique ? Alors, pour calculer le mot de passe à usage unique, vous aurez besoin du numéro de séquence que vous utilisez et de votre clé. Vous être probablement en train de chercher comment vous rappeler le numéro de séquence.

Lorsque vous vous connectez, le processus d'identification affiche une ligne contenant les informations dont vous avez besoin, ce dont vous utilisez pour générer un mot de passe à usage unique à travers un autre canal sécurisé, en copiant la ligne dans une console :

```
otp-md5 96 oshi45820
```

Après avoir entré votre mot de passe, votre mot de passe à usage unique sera affiché. Vous pouvez le copier dans l'invite de S/Key

lorsqu'il vous demande un mot de passe. Non seulement c'est *otp-md5* la description du hachage utilisé, mais c'est aussi un autre nom d'utilisateur pour la commande skey(1) .

Si vous êtes connecté et souhaitez générer un autre mot de passe à usage unique pour la session suivante, utilisez skeyinfo(1), Il vous indiquera ce qu'il faudra utiliser pour la session suivante. Par exemple, je génère d'autres mots de passe à usage unique pour des utilisations ultérieurs (souvenez-vous que je le fait depuis un canal sécurisé).

```
$ skeyinfo
95 oshi45820
```

La meilleure façon de le faire est d''utiliser **skeyinfo -v**, ce qui fournira une commande appropriée à exécuter dans une console. Par exemple :

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

De même, la façon la plus simple de générer le mot de passe S/Key suivant est juste :

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

Remarquez les caractères (`) dans l'exemple précédent.

Il est probable que beaucoup d'entre vous n'ont pas de connexion sécurisée ou une machine locale sécurisée pour créer les mots de passe, et leurs création à travers un canal non sécurisé n'est pas admissible. Donc comment créer plusieurs mots de passe en une seule fois ? Vous pouvez indiquer à skey(1) le nombre de mot de passe que vous voulez créer. Vous pourrez alors les imprimer et les emporter avec vous.

```
$ otp-md5 -n 5 95 oshi45820
Reminder - Do not use this program while logged in via telnet.
Enter secret passphrase:
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
```

Il est à noter que le dernier mot de passe doit être le premier utilisé, car nous comptons de manière décroissante à partir de 100.

Utilisation de S/Key avec telnet(1) et ssh(1)

L'utilisation de S/Key avec telnet(1) ou ssh(1) est à peu près identique qu'avec ftp -- vous devez simplement rajouter ":skey" à la fin de votre nom d'utilisateur. Exemple:

```
$ 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 3.9 (GENERIC) #617: Thu Mar  2 02:26:48 MST 2006

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 - OpenBSD supporte-t-il le SMP (Système Multiprocesseurs) ?

SMP est supporté sur les plates-formes OpenBSD/i386 et OpenBSD/amd64.

Un noyau SMP séparé, "`bsd.mp`", est fournit avec les paquetages d'installation, il peut être sélectionné pendant l'installation. Il est recommandé de tester ce noyau avant de le renommer en "`bsd`" pour en faire votre noyau par défaut.

Nous espérons que les autres plates-formes compatibles SMP soient supportées dans le futur. Sur la plupart des plates-formes, OpenBSD démarrera correctement sur un système SMP, mais n'utilisera qu'un seul processeur. L'exception est la plate-forme SPARC -- OpenBSD/sparc qui nécessitera parfois le retrait des modules MBus additionnels pour que le système démarre. Le systèmes SPARC64 Multiprocesseurs fonctionneront si la machine est supportée.

8.13 - Parfois, j'ai des erreurs d'entrées/sorties lorsque j'essaie d'utiliser mes périphériques tty

Vous devez utiliser /dev/cuaXX pour les connexions initialisées depuis le système OpenBSD, les périphériques /dev/ttyXX sont utilisables seulement pour les terminaux ou les connexions en dial. Alors qu'il était possible d'utiliser ces périphériques autrefois, le noyau OpenBSD n'est plus compatible avec ce type d'utilisation.

Extrait de 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 - Quels sont les navigateurs Web disponibles pour

OpenBSD ?

Lynx, un navigateur en mode texte supportant le SSL est disponible dans le système de base. Les autres sont présents dans l'arborescence des Ports, incluant (sans ordre particulier):

Navigateurs Graphique (X)

- Konqueror Installé avec l'environnement de bureau KDE.
- Konqueror-embedded (konq-e) Konqueror, n'utilisant que les bibliothèques KDE plutôt que l'ensemble de KDE.
- Links+ Un autre navigateur graphique rapide et léger (dispose également d'un mode console).
- Firefox et Mozilla sont des navigateurs disposant de beaucoup de fonctionnalités. Mozilla inclut beaucoup de fonctionnalités sortant du cadre d'un navigateur (client mail, client IRC, etc.), Firefox est simplement un navigateur basé sur Mozilla. Fonctionnent sur plates-formes alpha, amd64, i386, macppc, sparc et sparc64.
- Opera Navigateur commercial, i386 seulement.
- Amaya L'éditeur/navigateur du W3C.
- Netscape 4 Pour sparc et i386 seulement, n'est pas open source, aucun paquetage disponible.

Navigateurs en mode console

- elinks Riche en fonctionnalités, capable d'afficher les cadres et les tableaux, hautement configurable.
- w3m Supporte les tableaux et les frames. (dispose aussi d'un mode graphique).
- links Supporte les tables.

Vous les trouverez tous dans la collection de paquetages. Ils sont tous présents dans `/usr/ports/www/` après l'installation de l'arborescence des ports. Beaucoup sont aussi disponibles en paquetages précompilés, présents sur les serveurs FTP et sur le CD-ROM. Etant donné que la plupart des navigateurs graphiques pèsent lourd et prennent beaucoup de temps à télécharger et à compiler, vous devriez *sérieusement* prendre en compte que l'utilisation de paquetages précompilés était disponible.

8.15 - Comment utiliser l'éditeur mg ?

MG est un éditeur de texte minimal dans le style d'Emacs, inclus dans OpenBSD. Minimal signifie qu'il est léger (Emacs est très lourd). Pour les bases, lisez la page de manuel de mg(1) ainsi que le tutoriel, inclus dans le code source. Pour des questions plus intéressantes (comme , "Je n'ai pas de Meta-Clé !") voyez la FAQ Emacs.

Notez que mg est une petite implémentation de Emacs, qui est souvent similaire aux fonctionnalités d'édition de texte de Emacs 17, et n'implémente donc pas les autres fonctionnalités (comprenant les outils de mail et de news, aussi bien que les modes pour Lisp, C++, Lex, Awk, Java, etc...)

8.16 - ksh(1) ne semble pas lire mon fichier `.profile`!

Il y a deux raison possibles pour que ksh(1) semble ignorer le fichier `.profile` d'un utilisateur.

- `.profile` n'appartient pas à l'utilisateur. Pour corriger pour **utilisateur**,

```
# chown utilisateur ~utilisateur/.profile
```

- Vous utilisez ksh(1) depuis X Window System

dans une console xterm(1), argv[0] pour ksh(1) n'est pas précédé d'un trait d'union ("-"). Ajouter un trait d'union pour argv[0] précise à csh(1) et ksh(1) qu'ils doivent interpréter leurs fichiers de connexion. (Pour csh(1) c'est `.login`, avec un `.cshrc` séparé qui est toujours interprété quand csh(1) est lancé. Avec ksh(1), cela est plus perceptible car il n'y a qu'un fichier de démarrage, `.profile`. Ce fichier est ignoré si le shell n'est pas un shell de connexion.)

Pour corriger cela, ajoutez la ligne "`XTerm*loginShell: true`" au fichier `.Xdefaults` de votre dossier home. Note, ce fichier n'existe pas par défaut, vous devrez le créer.

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

Vous ne devriez pas avoir à faire cela sur d'autres systèmes, étant donné que certaines installations de X Window System viennent avec leur paramétrage par défaut. OpenBSD à choisi de suivre la ligne de conduite de X.

8.17 - Pourquoi le fichier `/etc/motd` est-il écrasé alors que je l'ai modifié ?

Le fichier `/etc/motd` est édité à chaque démarrage du système, remplaçant tout jusqu'a la première ligne vide du fichier, exclue, par les informations de version du noyau. Lorsque vous éditez ce fichier, soyez sûr que vous commencez après cette ligne vide, pour empêcher `/etc/rc` de supprimer ces lignes de `/etc/motd` au démarrage.

8.18 - Pourquoi le site www.openbsd.org est-il hébergé sur une machine Solaris ?

Même si aucun développeur ne pense que cela ai un rapport quelconque, cette question revient suffisamment souvent pour trouver une réponse ici. *www.openbsd.org* et le site FTP OpenBSD principal résident sur SunSITE dans l'université d'Alberta, au Canada. Ce sites sont hébergés sur un puissant système Sun, qui a accès à une grosse capacité de stockage et à une grosse bande passante. La présence de ce SunSITE permet au groupe OpenBSD d'accéder à cette bande passante. C'est pourquoi le site OpenBSD principal y est placé. La plupart des sites OpenBSD secondaires tournent sous OpenBSD, mais étant donné qu'ils ne peuvent garantir l'accès à cette large bande passante, le groupe à choisi d'héberger le site principal sur le SunSITE de l'université d'Alberta au Canada.

8.20 - Les polices anti-aliasées et "TrueType" avec X

Voyez ce document.

8.21 - Est-ce que OpenBSD supporte des systèmes de fichiers journalisés ?

Non il n'en supporte aucun. Nous utilisons un mécanisme différent pour obtenir des résultats similaires, celui-ci est appelé Soft Updates. Veuillez lire la FAQ 14 - Soft Updates pour obtenir plus de détails.

8.22 - Reverse DNS
- ou -
pourquoi cela prend autant de temps lorsque je me connecte ?

Beaucoup de nouveaux utilisateurs sur OpenBSD rencontrent un délai de connexion de deux minutes lorsqu'ils utilisent des services tels que ssh, ftp, ou telnet. Cela peut aussi être rencontré lorsque l'on utilise un proxy comme ftp-proxy, ou encore pendant l'envoi d'un email depuis une station de travail avec sendmail.

Ceci est le plus souvent du à un problème de reverse-DNS. DNS est le Serveur de Nom de Domaine (Domaine Name Serveur), le système qu'Internet utilise pour convertir un nom, comme "www.openbsd.org" en une adresse numérique. Une autre possibilité de DNS est de pouvoir prendre une adresse numérique et la reconvertir en "nom", c'est ce qu'on appelle le "Reverse-DNS".

Pour fournir un meilleur système de connexion, OpenBSD réalise un reverse-DNS sur chaque machine qui lui est attachée de différentes façons, incluant ssh, ftp, telnet, sendmail ou ftp-proxy. Malheureusement dans certains cas, la machine créant la connexion ne possède pas une entrée reverse DNS correcte.

Un exemple de cette situation :

Un utilisateur met en place une machine OpenBSD comme firewall et passerelle pour son réseau local, connectant tous ses ordinateurs interne au travers d'une unique adresse en utilisant NAT. Celui-ci peut aussi l'utiliser comme un relais de mail sortant. Il suit les notes d'installation, et est satisfait du résultat, à l'exception d'une petite chose -- chaque fois qu'il tente une requête sur la machine de quelque manière que ce soit, il doit attendre deux minutes avant qu'il ne se passe quoi ce soit.

Ce qu'il se passe :

Pour une station de travail derrière le NAT de la passerelle avec une adresse IP non enregistrée étant 192.168.1.35, l'utilisateur utilise ssh pour accéder au système de la passerelle. Le client ssh demande le nom d'utilisateur et le mot de passe, et les envoit à la machine passerelle. La passerelle tente ensuite de savoir qui a essayé de se connecter et réalise une requête reverse DNS de 192.168.1.35. Le problème est que les adresses 192.168.0.0 sont des adresses réservées à un usage privé, et un serveur DNS hors de votre réseau sait qu'il ne doit pas avoir d'information sur ces adresses. Certains vont rapidement retourner un message d'erreur, dans ce cas, OpenBSD comprendra qu'il n'y a pas plus d'informations à obtenir, et continuera rapidement en acceptant l'utilisateur. D'autres serveurs DNS ne retournerons AUCUNE réponse. Dans ce cas, vous vous retrouverez à attendre que le resolveur d'adresses OpenBSD dépasse son délai d'attente, ce qui prendra deux minutes avant que la connexion soit autorisée. Dans le cas de ftp-proxy, certains clients vont dépasser le délai imparti avant que la requête DNS ne dépasse le sien, donnant l'impression que ftp-proxy ne fonctionne pas.

Cela peut être relativement ennuyant. Heureusement, il existe une solution simple pour corriger cela.

Correction, en utilisant `/etc/hosts` :

La solution la plus simple est de partager votre fichier `/etc/hosts` entre toutes les stations de travail de votre réseau et vous assurer que le fichier `/etc/resolv.conf` contient bien la ligne `lookup file bind` qui confirme que le resolveur va bien considérer en premier lieu le fichier `/etc/hosts`, avant d'interroger les serveurs DNS spécifiques indiqués par les lignes "nameserver" dans votre fichier `/etc/resolv.conf`.

Votre fichier `/etc/hosts` ressemblera à quelque chose comme ceci :

```
::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
```

Votre fichier `resolv.conf` ressemblera à quelque chose comme ceci :

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

Une objection courante est de dire "Mais, j'utilise DHCP pour mon réseau local ! Comment puis-je configurer mon `/etc/hosts` ?" Très simple, en réalité. Entrez juste les lignes pour toutes les adresses que votre serveur DHCP va délivrer, ainsi que tous les périphériques static :

```
::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
```

Dans ce cas, je présume que votre plage DHCP est définie entre 192.168.1.100 et 192.168.1.199, ainsi que trois définitions statiques au début du fichier.

Si votre passerelle doit utiliser DHCP pour sa configuration, vous devriez rencontrer un problème -- dhclient va écraser votre fichier `/etc/resolv.conf` à chaque renouvellement de paramètres, cela supprimera la ligne "lookup file bind". Cela peut être résolu en ajoutant la ligne "lookup file bind" au fichier `/etc/resolv.conf.tail`.

Correction, en utilisant un serveur DNS local

Les détails sur cette mise en place, sortent du cadre de ce document, mais la procédure basique est de configurer votre serveur DNS favori, et de vous assurer qu'il a autorité pour les résolutions "forward" et "reverse" pour tous les points du réseau, et que vos ordinateurs (ainsi que votre passerelle) utilisent ce serveur DNS.

8.23 - Pourquoi les pages web d'OpenBSD ne sont-elles pas conformes au HTML4/XHTML ?

Les présentes pages Web ont été écrites avec précaution pour fonctionner sur une large variété de navigateurs actuels, pour les versions 4.0 et suivantes. Nous ne souhaitons pas rendre ces anciennes pages conformes aux normes HTML4 ou XHTML tant que nous ne serons pas sur qu'elles fonctionneront aussi avec les anciens navigateurs ; ce n'est tout simplement pas une priorité. Nous accueillons les nouveaux contributeurs, mais nous vous suggérons de travailler à écrire du code, ou documenter de nouveaux aspects du système, et non de vous concentrer sur la conformité aux nouveaux standards des pages existantes.

8.24 - Mon horloge est décalée de plusieurs heures. Pourquoi ?

Lorsque vous utilisez rdate(8) pour synchroniser votre horloge à un serveur NTP, vous pourrez observer que votre horloge avance/retarde d'une vingtaine de secondes par rapport à votre définition locale de temps.

Ceci est causé par une différence entre le temps UTC (Coordinated Universal Time, basé sur des observations astronomiques) et le temps TAI (International Atomic Time, basé sur des horloges atomiques). Pour compenser les variations dans la rotation de la Terre, des secondes sont insérées dans UTC, mais TAI n'est pas ajusté. Pour une description plus détaillée, cherchez "`leap seconds UTC TAI`" sur le Web.

Corriger le problème est simple. Dans beaucoup de pays, vous obtiendrez la bonne heure si vous utilisez le paramètre "`-c`" à rdate(8) et utilisez une zone dans le dossier `/usr/share/zoneinfo/right/`. Par exemple, si vous habitez en Allemagne, vous pouvez utiliser ces commandes :

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

Dans d'autres pays, les règles peuvent différer.

8.25 - Mon horloge avance/retarde de plusieurs heures. Pourquoi ?

Par défaut, OpenBSD suppose que l'horloge matérielle de votre équipement est paramétrée de telle façon à indiquer l'heure UTC (Universal Coordinated Time) au lieu de l'heure locale.

Si cela représente un problème (si, par exemple, vous avez un double boot avec d'autres systèmes d'exploitation), vous pouvez changer le comportement par défaut à l'aide de config(8). Par exemple, pour configurer OpenBSD de telle façon à utiliser une horloge matérielle paramétrée en US/Eastern (5 heures de moins que UTC, c'est-à-dire 300 minutes) :

```
# config -ef /bsd
OpenBSD 3.9 (GENERIC) #617: Thu Mar  2 02:26:48 MST 2006
     deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/GENERIC
Enter 'help' for information
ukc> timezone 300
timezone = 300, dst = 0
ukc> quit
Saving modified kernel.
```

Veuillez consulter options(4) et cherchez l'option "TIMEZONE=value" pour plus d'informations.

Normalement, le fuseau horaire est paramétré à l'installation. Si vous avez besoin de modifier le fuseau horaire, vous pouvez créer un nouveau lien symbolique vers le fichier de fuseau horaire adéquat dans `/usr/share/zoneinfo`. Par exemple, pour paramétrer la machine de telle façon à utiliser EST5EDT comme le nouveau fuseau horaire local :

```
# ln -fs /usr/share/zoneinfo/EST5EDT /etc/localtime
```

Veuillez consulter également :

- date(1)
- 8.24 - Mon horloge est décalée de plusieurs heures. Pourquoi ?
- OpenNTPD

[Index de la FAQ] [Section 7 - Contrôle du clavier et de l'affichage] [Section 9 - Migrer vers OpenBSD]


www@openbsd.org

$OpenBSD: faq8.html,v 1.51 2006/09/26 09:49:58 jufi 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 "lorsqu'il aura le temps de le faire" ou lorsque "quelqu'un se plaint". 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 séparément. 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 paquetages pré-compilés sont créés et distribués par l'équipe de portage OpenBSD. Les utilisateurs sont encouragés à utiliser des paquetages au lieu de compiler les leurs.
- OpenBSD utilise CVS pour suivre 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 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/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 par défaut d'OpenBSD est /bin/ksh, qui est pdksh, le shell Korn du domaine public. Les autres shells inclus sont csh et sh. Les shells tels que *bash* et *tcsh* peuvent être ajoutés à partir des paquetages 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 partitionnement 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 à partir de paquetages ou ports (`security/john`) 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", \
> $1,$2,$3,$4,$5,$6,$7); }' > new_passwd
```

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 bibliothèques Linux les plus communes est d'installer `redhat/base` à partir du site miroir FTP le plus proche. Pour plus d'informations concernant les paquetages et le système de ports, veuillez consulter FAQ 15 - Le système de ports et de paquetages d'OpenBSD. Pour installer le paquetage mentionné ci-dessus, utilisez les commandes suivantes :

```
# export PKG_PATH=ftp://your.ftp.mirror/pub/OpenBSD/3.9/packages/i386
# pkg_add redhat_base-8.0p4.tgz
```

A partir d'OpenBSD 3.7, pkg_add(1) exécutera automatiquement sysctl pour paramétrer correctement kern.emul.linux à la bonne valeur après avoir ajouté le paquetage. Cependant, il ne modifie pas `/etc/sysctl.conf`. Si vous voulez activer l'émulation Linux par défaut, vous devez modifier la variable kern.emul.linux dans ce fichier.

9.5 - Accéder à vos fichiers Linux depuis OpenBSD

OpenBSD supporte le système de fichiers EXT2FS. Pour plus d'informations, veuillez consulter la FAQ 14.

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


www@openbsd.org

$OpenBSD: faq9.html,v 1.42 2006/09/08 20:49:53 saad Exp $

[Index de la FAQ] [Section 9 - Migrer vers OpenBSD] [Section 12 - Questions Spécifiques Aux Plates-Formes]

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 au groupe `"wheel"` 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 lors de la mise à jour du 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.
- 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. Lisez la page man correspondante 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 les paquetages 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 :

```
# grep relay-domains /etc/mail/sendmail.cf
```

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).

Note : 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 bibliothèques 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* avec la clef correspondante 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 style "6th Edition"
     /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 bibliothèque 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)

Il est toujours possible de créer des utilisateurs à la main en utilisant vipw(8), mais cela complique la plupart des étapes.

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*. 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]: Entrée
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]: Entrée
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:       /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, néanmoins 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 ftpd(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
```

Ensuite vous êtes prêt pour ajouter l'utilisateur *ftp*.

```
# 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]: Entrée
Login group ftp [ftp]: Entrée
Login group is ``ftp''. Invite ftp into other groups: guest no
[no]: no
Login class auth-defaults auth-ftp-defaults daemon default staff
[default]: Entrée
Enter password []: Entrée
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(8) soit de le lancer directement via les scripts rc. Les exemples suivants vous montreront le service lancé via 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/ftpd       ftpd -llUSA
```

Note : 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.local 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.local 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

Même avec OpenBSD, des bogues apparaissent de temps à autre. Certaines bogues causent des problèmes de fiabilité (par exemple, quelque chose peut amener le système à ne plus fonctionner correctement). D'autres bogues causent des problèmes de sécurité (qui peuvent permettre à d'autres personnes d'utiliser votre machine de façon inattendue). Lorsqu'un bogue critique est trouvé, le correctif sera mis en place au niveau de l'arborescence du code source *-current*. Ce correctif sera ensuite propagé vers les versions maintenues d'OpenBSD. Ces correctifs apparaissent sur la page web des errata. Ils sont séparés en correctifs "communs" applicables à toutes les plates-formes, et en correctifs applicables à une ou plusieurs plates-formes mais pas toutes.

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 problèmes de stabilité ou de sécurité qui doivent être réglés très rapidement sur les systèmes impactés (mais pas tous les systèmes vu que l'application de tel ou tel correctif dépend des applications utilisées).

Il existe trois façons d'installer les correctifs sur votre système :

- **Mettez à jour votre système en *-current*.** Vu que tous les correctifs sont appliqués systématiquement au code source de la branche *-current*, la mise à jour de votre système avec le dernier snapshot disponible est une excellente façon d'installer les correctifs. Cependant, l'utilisation de la branche *-current* n'est pas adaptée à tout le monde.
- **Mettez à jour votre système en *-stable*.** Vous pouvez faire ceci en téléchargeant votre arborescence du code source ou en la mettant à jour en utilisant la branche *-stable* appropriée puis en recompilant le noyau et le reste du système d'exploitation. C'est probablement la méthode la plus aisée mais la recompilation de tout le système prend du temps. De plus, le téléchargement complet du code source peut prendre du temps aussi si vous avez une bande passante limitée.
- **Corrigez, compilez et installer les fichiers impactées individuellement.** C'est la méthode que nous allons aborder ci-après de manière détaillée. Bien que cette méthode nécessite peu de bande passante et prend, typiquement, peu de temps par rapport à un téléchargement ou une mise à jour via cvs(1) et une compilation complète de tout le système, elle peut parfois être la plus difficile à utiliser. En effet, il n'y a pas une procédure universelle à suivre pour la mise à jour. Certaines fois vous devez mettre en oeuvre un correctif pour une application donnée puis recompiler et installer cette dernière. D'autres fois, vous aurez peut-être à recompiler des sections entières de l'arborescence des sources si le problème est localisé au niveau d'une bibliothèque.

Encore une fois, la correction de fichiers individuels n'est pas toujours simple. Pensez à utiliser la deuxième méthode décrite plus haut et suivre la branche *-stable* (dite aussi la branche "des correctifs") d'OpenBSD. L'utilisation combinée de ces différentes méthodes est possible si vous comprenez exactement comment ça fonctionne. Les nouveaux utilisateurs devront choisir une seule méthode.

Quelle est la différence entre les correctifs de la page des errata et ce qui existe au niveau

de la base de données CVS ?

Tous les correctifs postés sur la page web des errata concernent uniquement l'arborescence des sources de la version indiquée dans cette page. 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. Ceci est important : Si vous avez installé un snapshot et que vous avez téléchargé les arborescences du code source au moment où vous avez obtenu le snapshot, il se peut que si vous essayer d'appliquer un des correctifs publiés, ça ne marche pas à cause d'une modification du code source.

Application des correctifs

Les correctifs d'OpenBSD sont distribués sous la forme de fichiers diff unifiés. 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. De manière générale, vous devriez avoir à disposition l'arborescence complète du code source. Si votre machine héberge une version à partir d'un CDROM officiel, l'arborescence du code source est disponible sur le disque 3. Elle est aussi disponible sous forme de fichiers à partir des serveurs FTP. Nous allons désormais supposer que vous avez l'arborescence complète à votre disposition.

A titre d'exemple, nous allons appliquer le correctif 001 pour OpenBSD qui corrige un bogue au niveau du pilote st(4) qui gère les lecteurs de bandes magnétiques. Sans ce correctif, la restoration de sauvegardes est assez difficile. Les personnes utilisant un lecteur de bandes magnétiques ont *besoin* de ce correctif. Les autres personnes peuvent s'en passer. Jetons un coup d'oeil à ce correctif :

```
# more 001_st.patch
Apply by doing:
        patch -p0 < 001_st.patch

Rebuild your kernel.
Index: sys/scsi/st.c
===================================================================
RCS file: /cvs/src/sys/scsi/st.c,v
retrieving revision 1.41
retrieving revision 1.41.2.1
diff -u -p -r1.41 -r1.41.2.1
--- sys/scsi/st.c       1 Aug 2004 23:01:06 -0000       1.41
+++ sys/scsi/st.c       2 Nov 2004 01:05:50 -0000       1.41.2.1
@@ -1815,7 +1815,7 @@ st_interpret_sense(xs)
        u_int8_t skey = sense->flags & SSD_KEY;
        int32_t info;

-       if (((sense->flags & SDEV_OPEN) == 0) ||
+       if (((sc_link->flags & SDEV_OPEN) == 0) ||
            (serr != 0x70 && serr != 0x71))
                return (EJUSTRETURN); /* let the generic code handle it */
```

Comme vous pouvez le constater, le début du patch inclut de brèves instructions sur la façon de l'appliquer. Nous admettrons que vous avez placé ce patch dans le répertoire `/usr/src` auquel cas les étapes suivantes seront utilisées :

```
# cd /usr/src
# patch -p0 < 001_st.patch
Hmm...  Looks like a unified diff to me...
The text leading up to this was:
--------------------------
|Apply by doing:
|        cd /usr/src
|        patch -p0 < 001_st.patch
|
|Rebuild your kernel.
|
|Index: sys/scsi/st.c
|===================================================================
|RCS file: /cvs/src/sys/scsi/st.c,v
```

```
|retrieving revision 1.41
|retrieving revision 1.41.2.1
|diff -u -p -r1.41 -r1.41.2.1
|--- sys/scsi/st.c      1 Aug 2004 23:01:06 -0000       1.41
|+++ sys/scsi/st.c      2 Nov 2004 01:05:50 -0000       1.41.2.1
--------------------------
Patching file sys/scsi/st.c using Plan A...
Hunk #1 succeeded at 1815.              <-- Ce message indique le
succès de l'opération
done
```

Le message "`Hunk #1 succeeded`" indique que le correctif a été appliqué avec succès. Plusieurs correctifs sont plus complexes que l'exemple utilisé, et impliqueront de multiples "hunks" et fichiers. Dans ce cas, il faudrait que vous vous assuriez que tous les "hunks" ont été appliqués avec succès pour tous les fichiers concernés. Si ce n'est pas le cas, ceci veut normalement dire que votre arborescence du code source n'est pas bonne, que vous n'avez pas suivi les instructions, ou que le correctif ne correspond pas au correctif originel. Les correctifs sont très sensibles aux espaces blancs -- un copier/coller depuis votre navigateur changera la plupart du temps les caractères de tabulation en espaces ou modifiera les espaces blancs de telle façon à rendre le correctif inutilisable.

Vous pouvez maintenant recompiler le noyau comme d'habitude, l'installer et redémarrer le système.

Les correctifs ne s'appliquent pas systématiquement au noyau. Dans certains cas, vous devrez recompiler des utilitaires. Dans d'autres, vous devrez recompiler tous les utilitaires liés statiquement à une bibliothèque sujette à correction. Suivez les instructions dans l'en-tête du correctif, et si vous n'êtes pas certain, recompilez tout le système.

Les correctifs qui n'impactent pas directement votre utilisation du système n'ont pas besoin d'être appliqués normalement. Par exemple, si vous n'aviez pas de lecteur de bande magnétique dans votre système, vous ne bénéficierez pas du correctif ci-dessus. Cependant, les correctifs sont supposés être appliqués dans l'ordre. Il est donc possible qu'un correctif ultérieur dépend d'un correctif apparu plutôt. Soyez conscient de ce mode de fonctionnement si vous sélectionnez la méthode consistant à choisir vous-même vos correctifs. Si vous avez un doute, appliquez-les tous, dans l'ordre de leur mise à disposition.

10.16 - Parlez moi de chroot(2) 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 problèmes 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 son 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 lit son 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*. Dans la configurarion par défaut d'OpenBSD, tous les fichiers du répertoire */var/www* sont en lecture seule pour l'utilisateur qui fait tourner Apache, *www*. Cela aide considérablement la sécurité ; si il devait y avoir un problème de sécurité, les dégâts 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'administrateur ?

En gros, chroot(2)er Apache est quelque chose qui n'est pas adopté par la plupart des autres systèmes d'exploitation. Beaucoup d'applications et de configurations système ne fonctionneront plus comme avant sans aménagement. De plus, il faut se souvenir que sécurité et commodité sont souvent incompatibles. L'implémentation d'Apache sous OpenBSD ne sacrifie pas la sécurité aux profit des capacités ou de la "facilité".

- **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 existante 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 évidemment, 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 plutôt quelque chose similaire à */var/www/users*. Les liens symboliques sont totalement utilisables à l'intérieur du chroot(2) mais ceux-ci doivent être relatifs et non absolus.

- **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 que httpd(8) n'a pas la possibilité d'ouvrir les fichiers de log en écriture une fois que ses privilèges ont été abaissés. httpd(8) doit donc être stoppé et relancé. Cela demande parfois quelques secondes pour que tous les processus enfants se terminent, ce qui est obligatoire avant que httpd(8) ne puisse redémarrer ; une possibilité de rotation des logs pourrait être :

```
# apachectl stop
  rename your log files
# apachectl start ; sleep 10 ; apachectl start
```

  Oui, la dernière ligne tente de redémarrer Apache immédiatement et dans le cas où cela ne fonctionnerait pas, on attend quelques secondes puis on rééssaye. Et oui, cela signifie bien que pendant quelques secondes chaque fois que vous effectuerez une rotation des logs, votre serveur web sera inaccessible. Bien que cela puisse poser des problème, n'importe quelle tentative pour permettre à httpd(8) de rouvrir des fichiers après avoir chroot(2)é ira à l'encontre de l'intérêt même du chroot ! 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 bibliothèques se trouvant hors de */var/www*. Certains peuvent être corrigés en étant compilés statiquement (n'ayant pas besoin de bibliothèques 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 une certaine connaissance du programme.
- **Options de montage du système de fichiers :** Par défaut sous OpenBSD, votre partition */var* est monté avec les options `nosuid` et `nodev`/ Si vous tentez d'utiliser une application à l'intérieur du chroot, vous serez peut-être amené à changer ces options. Bien entendu, il est possible que vous ayez besoin d'effectuer ces changements même si vous n'utiliser pas l'option du chroot.

Dans certains cas, les applications ou les configurations peuvent être changées pour fonctionner dans le chroot(2). 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 les paquetages. 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 le paquetage 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 bibliothèques 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 paquetage, nous prenons l'option 2, à savoir le déplacement des fichiers de données. Afin que nous puissions 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 !

Notez que nous n'avons copié que les fichiers strictement nécessaires au bon fonctionnement. En général, seuls les fichiers nécessaires à l'application doivent être copiés dans le chroot.

Dois-je utiliser chroot ?

Dans l'exemple précédent, le programme est relativement simple et pourtant nous avons rencontré différents types de problèmes.

Toutes les applications ne peuvent ou ne devraient pas être chroot(2)ées.

Le but est la mise en place d'un serveur web sécurisé et le chroot(2)age n'en est qu'un outil, pas un but en soi. Souvenez-vous, la configuration initiale de l'Apache chroot(2)é sous OpenBSD fair en sorte que l'utilisateur sous lequel le programme httpd(8) tourne ne peut pas lancer de programme, ne peut pas modifier de fichiers et ne peut pas prendre l'identité d'un autre utilisateur. Réduire ces restrictions diminuera votre sécurité, chroot ou pas.

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 les avantages du chroot(2) après avoir copié la masse importante de fichiers dont elles ont besoin pour fonctionner. Ainsi le programme OpenWebMail a besoin de pouvoir lire et écrire dans le répertoire mail, le répertoire home de l'utilisateur et doit pouvoir travailler en tant que n'importe quel utilisateur du système. Essayer le mettre cette application en chroot serait inutile car vous serez obliger de désactiver tous les bénéfices du chroot(2)age. Même avec des applications aussi simples que le compteur précédent, il doit pouvoir écrire sur le disque (pour garder la trace de ses compteurs) et donc, une *partie* du bénéfice du chroot(2) est perdue.

Toute application qui doit fonctionne sous root ne vaut pas le coup d'être chroot(2)ée puisque root peut généralement s'échapper du chroot(2).

N'oubliez pas, si la procédure de chroot pour votre application est trop complexe vous pourriez ne pas mettre à jour votre système aussi souvent qu'il le faudrait. Ceci pourrait rendre votre système MOINS sécurisé qu'un système plus facilement administrable et dont le chroot est désactivé.

10.17 - Puis-je changer le shell de l'utilisateur root ?

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.

Le shell par défaut sur OpenBSD de l'utilisateur *root* est ksh.

Une directive Unix traditionnelle 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 bibliothèques 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.

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 fléchées, 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 paquetages ou soit via les ports.

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.

A partir d'OpenBSD 3.7, ksh(1) a été amélioré. Un certain nombre de caractères spéciaux ont été ajoutés au niveau de la chaîne primaire de l'invite de commandes, PS1. Ces caractères sont similaires à ceux présents dans bash. Par exemple :

`\e` - Insertion d'un caractère d'échappement ASCII.
`\h` - Le nom d'hôte sans le nom de domaine.

- `\H` - Le nom d'hôte complet, avec le nom de domaine.
- `\n` - Insertion d'un caractère de retour à la ligne.
- `\t` - L'heure actuelle, sur 24 heures, au format HH:MM:SS.
- `\u` - Le nom de l'utilisateur actuel.
- `\w` - Le répertoire de travail actuel. $HOME est abrégé en `~'.
- `\W` - La racine du répertoire de travail actuel.

(consultez la page du manuel ksh(1) pour plus de détails !)

Vous pouvez par exemple utiliser la commande suivante :

```
export PS1="\n\u@\H\n\w $ "
```

pour disposer d'une invite de commandes très parlante mais dont l'utilité n'est que relative.

[Index de la FAQ] [Section 9 - Migrer vers OpenBSD] [Section 12 - Questions Spécifiques Aux Plates-Formes]


www@openbsd.org

$OpenBSD: faq10.html,v 1.53 2006/09/13 11:40:59 jufi Exp $

[Index de La FAQ] [10 - Gestion du Système] [Section 13 - Multimédia]

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.

12.2 - 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 énormément. 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 !

80386sx

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.

12.7.5 - 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 "switch box" (appelé aussi commutateur KVM) pour connecter plusieurs machines à un seul clavier, un seul écran et une seule souris. Vous pouvez essayer plusieurs "switch box" 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. Une autre solution de contournement consiste à utiliser un adaptateur PS/2 vers USB, permettant à OpenBSD de voir une souris de type USB, 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.

12.7.7 - Qu'est-il arrivé au support RAID des cartes Adaptec (aac)?

Adaptec a refusé de fournir une documentation précise et utile pour leurs contrôleurs RAID (aac(4)), à base de FSA. Ces contrôleurs RAID semblent être très buggés. Cette documentation est donc nécessaire pour créer un pilote utilisable. Vu que le pilote était très peu fiable, il a été supprimé du noyau GENERIC.

Je peux toujours compiler mon propre noyau avec le support aac(4), n'est-ce pas ?

Bien sûr. Mais quelle partie de l'expression "très peu fiable" n'avez-vous pas compris ? Ce n'est pas une fonctionnalité "expérimentale" mais un pilote connu pour être défectueux. Il fonctionnera peut-être correctement avec certaines variations de matériel mais nous ne vous recommandons pas de parier vos données sur ce pilote.

12.7.8 - Ma carte ami(4) ne supporte qu'un seul disque logique !

ami(4) contient un bogue connu qui cause une corruption de données si vous utilisez plus d'un seul volume sur certains contrôleurs. Pour les contrôleurs ayant ce problème, OpenBSD vous limitera à un seul disque logique. Vous aurez alors un message dans votre dmesg qui ressemble au message suivant :

```
ami0: FW A.04.03, BIOS vA.04.03, 4MB RAM
ami0: 3 channels, 16 targets, 2 logical drives
ami0: firmware buggy, limiting access to first logical disk
scsibus0 at ami0: 1 targets
```

12.8 - Mac68k

12.8.1 - Plus le temps passe, plus mon horloge dévie de l'heure exacte. Pourquoi ?

Cela était causé par un bogue de longue durée, fixé dans OpenBSD 3.9.

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.

12.13.2 - J'obtiens le message "partition extends past end of unit" dans disklabel

Le disklabel BSD ne peut décrire une géométrie disque supérieur à 8Go sous sparc et sparc64, bien que les entrées disklabel individuelles peuvent être plus larges.

A chaque fois que vous exécutez disklabel(8), ce dernier effectue des vérifications pour s'assurer que les entrées disklabel correspondent bien à sa compréhension de la géométrie disque. Seulement, comme il voit une géométrie tronquée, il vous le signale et ne vous permettra pas d'éditer des entrées au delà de cette zone de 8Go sauf si vous lui dites d'utiliser la géométrie réelle. Vous pouvez faire cela à l'aide de la commande 'g' de l'éditeur en ligne de commande de disklabel(8) et dites lui d'utiliser "[d]isk geometry" :

```
# disklabel -E wd0
# Inside MBR partition 3: type A6 start 63 size 17912412
   [...]
Initial label editor (enter '?' for help at any prompt)
> g
[d]isk, [b]ios, or [u]ser geometry: [d] d
> w
> q
```

Vous aurez toujours des messages d'avertissement, mais vous pourrez configurer et utiliser votre disque comme vous le souhaitez. Une meilleure solution nécessiterait d'être compatible avec les systèmes existants déjà utilisés ainsi qu'avec Solaris s'exécutant sur des disques de capacité supérieure à 8Go mais cette solution n'existe pas aujourd'hui.

12.14 - DEC VAX

12.14.1 - Puis-je utiliser le simulateur VAX SIMH ?

Oui !

Le simulateur VAX SIMH peut être utiliser pour effectivement émuler un un vrai VAX. Les instructions sont disponibles ici.

[Index de La FAQ] [10 - Gestion du Système] [Section 13 - Multimédia]


www@openbsd.org

$OpenBSD: faq12.html,v 1.43 2006/09/08 20:49:53 saad Exp $

[Index de la FAQ] [Section 12 - Questions spécifiques aux plates-formes] [Section 14 - Configuration des disques]

13 - Multimedia

Table des matières

13.1 - Comment configurer mon périphérique audio ?

Les périphériques concernant l'audio sous OpenBSD sont : `/dev/audio`, `/dev/sound`, `/dev/audioctl` et `/dev/mixer`. Pour une belle introduction à la couche de support audio, vous pouvez lire la page de manuel audio(4).

Tous les matériels audio supportés sont déjà inclus dans le noyau GENERIC, il n'y a donc pas besoin de configuration supplémentaire ou d'installation de drivers. Pour trouver des informations spécifiques à la puce de votre carte son, vous devrez déterminer quelle puce vous avez. Les puces supportées peuvent être trouvées sur la page des matériels supportés pour votre plate-forme. Si vous avez déjà OpenBSD lancé, cherchez l'apparition d'un driver de carte son dans la sortie de la commande dmesg(1), et lisez le manuel du driver pour des informations plus spécifiques comme les options et autres détails sur celui-ci. Voici un exemple d'une puce audio dans une sortie dmesg :

```
auich0 at pci0 dev 31 function 5 "Intel 82801BA AC97" rev 0x04: irq 10, ICH2 AC97
ac97: codec id 0x41445360 (Analog Devices AD1885)
ac97: codec features headphone, Analog Devices Phat Stereo
audio0 at auich0
```

Vous pouvez tester si votre matériel audio fonctionne correctement en lui envoyant un fichier audio (souvent d'extension .au). Si vous ne disposez pas de fichier audio, vous pouvez envoyer n'importe quel fichier texte ou binaire sur le périphérique :

```
$ cat filename > /dev/audio
```

Si vous entendez quelque chose (si ne n'est pas un fichier audio le son ne sera pas agréable), cela signifie que la puce de votre matériel est supportée par OpenBSD et fût reconnue et configurée par le noyau lors du démarrage.

Remarque : Toutes les variantes et utilisations de chaque puce n'ont pas été testées ou débuguées.

Si vous n'avez entendu aucun son après avoir lancé la commande précédente, il peut y avoir plusieurs raisons possibles :

- Le fichier que vous avez envoyé est tellement petit que vous avez du mal à entendre quelque chose. Essayez encore avec un fichier d'au moins 10 Ko.
- La puce à été configurée et reconnue correctement, mais le son est muet et doit être réactivé pour que vous puissez entendre quelque chose, regardez ci-dessous.
- La puce à été reconnue mais n'a pas été configurée correctement. Peut être avez vous une vielle carte ISA qui doit être configurée avec une adresse I/O et une valeur IRQ différente pour éviter les conflits avec d'autres matériels. Vous pouvez simplement essayer différentes combinaisons en utilisant la Configuration au démarrage.
- Votre carte/puce n'est pas totalement supportée.

Notez que même si vous avez entendu du son, cela ne signifie pas forcément que tout fonctionnera comme espéré. Si vous rencontrez des problèmes en jouant du son, voici d'autres vérifications à effectuer :

- Trouver des informations à propos de votre périphérique audio. Utilisez la documentation ou effectuez une recherche sur Internet afin de connaître ses spécifications. Celles-ci pourraient bien vous aider à trouver l'origine du problème.
- Veillez à bien activer ("unmute") **toutes** les sorties audio (voir ci-dessous) avant de rapporter un problème. Parfois les sorties listées ne sont pas en rapport avec la réalité, ainsi la sortie headphones (casque) peut être mélangée avec la sortie line (ligne).
- Il est possible qu'un pilote non optimal soit rattaché à votre périphérique et que vous puissiez obtenir de meilleurs résultats en utilisant un autre pilote. Il ne s'agit pas du problème le plus simple à découvrir mais jetez un oeil attentif la sortie de votre dmesg(1) afin de trouver les lignes correspondant à votre carte son. Si vous voyez plusieurs pilotes audio tentant de s'attacher au périphérique, testez-les un par un en n'en laissant qu'un seul activé. Pour cela, utilisez la Configuration au démarrage.
- Vous possédez un périphérique audio qui ne supporte qu'un taux d'échantillonnage fixe. Dans ce cas, il existe plusieurs utilitaires audio qui supportent le ré-échantillonnage comme indiqué dans Jouer différents types de formats audio.

Pour configurer vos périphériques audio, comme la vitesse de lecture, vous pouvez utiliser audioctl(1). Pour configurer le volume sonore et les autres paramètres de mixage, vous pouvez utiliser mixerctl(1). Ces deux utilitaires sont fournis avec le système de base.

Par exemple, pour configurer le volume sonore des canaux gauche et droite à 200, vous pourriez lancer la commande (voir note 2 plus bàs) :

```
$ mixerctl outputs.master=200,200
outputs.master: 255,255 -> 207,207
```

Remarquez que la valeur est devenue 207. La raison à cela est que le périphérique de ma carte audio est un codec AC'97, qui utilise seulement 5 bits pour le contrôle, ce qui ne laisse que 32 valeurs possibles. Evidemment, les autres matériels peuvent avoir une résolution différente.

Pour activer le canal "master", vous devriez faire

```
$ mixerctl outputs.master.mute=off
outputs.master.mute: on -> off
```

Pour que vos changements soient permanents vous devrez éditer */etc/mixerctl.conf*, par exemple :

```
$ cat /etc/mixerctl.conf
outputs.master=200,200
outputs.master.mute=off
outputs.headphones=160,160
outputs.headphones.mute=off
```

Note 1 : Vous pourriez voir que plus de sorties sont disponibles sur votre carte son ou votre carte mère. Ceci est du au fait que beaucoup de puces audio sont généralement utilisées pour connecter les jacks sur les sorties, et toute option de chaque puce audio n'est pas forcément accessible depuis le monde extérieur.

Note 2 : Sur votre système, il se peut que les sorties des périphériques soient désignées autrement. Ainsi, vous pourriez ne pas avoir "outputs.master" comme dans l'exemple précédent et devriez ajuster "outputs.output" ou encore autre chose. Cela dépend du pilote audio mais il est facile de trouver la dénomination correcte en listant tous les contrôles possibles avec :

```
$ mixerctl -a
```

13.2 - Jouer différents types de formats audio

Audio digitalisée

Formats audio sans perte (AU, PCM, WAV, FLAC, TTA)

Certains formats audio sans perte peuvent être joués sans qu'un logiciel tierce partie ne soit nécessaire, ils contiennent les enregistrements digitaux non compressés en flots d'octets. Ces formats incluent les fichiers Sun audio (AU), les fichiers raw PCM (sans en-têtes), et les RIFF WAV.

Pour jouer un tel fichier, vous devrez connaître ses paramètres principaux : le type d'encodage, le nombre de canaux, la fréquence, et le nombre de bit par morceau. Si vous ignorez ces paramètres, vous pourrez les retrouver en utilisant l'utilitaire : file(1).

```
$ file music.au
music.au:  Sun/NeXT audio data: 16-bit linear PCM, stereo, 44100 Hz

$ file music.wav
music.wav: Microsoft RIFF, WAVE audio data, 16 bit, stereo 44100 Hz
```

La seule chose qu'il vous reste à savoir sur ces fichiers d'exemple est qu'ils sont encodés dans le format d'octet "little-endian" utilisant la quantisation linéaire signée. Vous pouvez vérifier cela en lisant son en-tête avec hexdump(1). Si vous utilisez un fichier sans en-tête (raw), il n'y a aucun moyen de connaître les paramètres à l'avance. Configurez les paramètres suivants en utilisant audioctl(1).

```
play.encoding=slinear_le
play.rate=44100
play.channels=2
play.precision=16
```

Ensuite, envoyez le fichier audio au périphérique son :

```
$ cat music.au > /dev/sound
```

Si vous avez appliqué les bons paramètres, vous devriez entendre ce que vous souhaitiez entendre.

Remarque : Utilisez toujours `/dev/sound`, et non `/dev/audio`, si vous souhaitez que les paramètres appliqués restent en place.

Il y à d'autres utilitaires que vous pouvez utiliser, comme aucat(1), et `audio/waveplay` dans les paquetages et les ports. Bien sur, des logiciels populaires comme XMMS sont aussi capables de jouer ces fichiers aussi bien que d'autres formats;

A l'inverse de ce qui à été dit précédemment, il existe des format audio sans perte de qualité. Free Lossless Audio Codec (FLAC) et TTA en sont des exemples. L'implémentation FLAC à été portée sous OpenBSD et peut être trouvée dans `audio/flac` dans les paquetages et les ports.

Formats audio utilisant la compression par perte (Ogg Vorbis, MP3, WMA, AAC)

La compression par perte de qualité est souvent utilisée pour l'audio ou les autres fichiers multimédia. L'idée est qu'un certain volume de données est supprimé pendant la compression, en utilisant une méthode qui fait que le résultat compressé est toujours très exploitable et dispose d'une bonne qualité pour être joué. L'avantage de ces techniques est qu'ils autorisent de plus gros ratios de compression, ce qui résulte en une plus faible place disque utilisée et de plus besoins en bande passante.

Un bon exemple est le format libre, ouvert et non déposé Ogg Vorbis. Pour jouer des fichiers Ogg Vorbis, vous pouvez utiliser l'utilitaire ogg123, qui est inclus dans le paquetage `audio/vorbis-tools`. Par exemple :

```
$ ogg123 music.ogg

Audio Device:   Sun audio driver output

Playing: music.ogg
Ogg Vorbis stream: 2 channel, 44100 Hz
Time: 00:02.95 [02:21.45] of 02:24.40  (133.1 kbps)  Output Buffer  87.5%
```

Bien sur, les plugins Ogg Vorbis existent pour beaucoup d'autres logiciels audio.

Un autre exemple est le très populaire encodage MPEG-1 Audio Layer 3 (MP3), qui a, cependant, ses problèmes de redistribution et brevets. Beaucoup de programmes peuvent jouer des fichiers MP3, consultez la section `audio` du systèmes de paquetages et de ports et choisissez celui que vous souhaitez.

Que dire du format Windows Media Audio (WMA) ? Les fichiers de ce type peuvent être lus en utilisant `x11/mplayer` qui se sert de l'architecture FFmpeg.

Un bon point de départ pour en savoir plus sur les différents formats de fichiers audio est de lire l'article Wikipedia suivant : Audio file formats (en anglais).

Les périphériques audio au taux d'échantillonnage fixe

Certaines cartes son ne peuvent utiliser qu'un taux d'échantillonnage fixe. Ainsi, vous pourriez être en train d'essayer de lire un fichier à 22050 Hz au travers d'un périphérique sonore bloqué à 48000 Hz.

Il existe des utilitaires audio dans les paquetages et l'arbre des ports qui peuvent résoudre ce problème en ré-échantillonnant le son. Par exemple, `x11/mplayer` lancé avec l'option "`-srate`" permet de spécifier l'échantillonnage désiré. Vous devriez spécifier le taux utilisé par votre carte son. Le serveur de sons de KDE, `artsd`, ainsi que certains jeux supportent des options similaires. Référez-vous à la documentation de votre application afin de voir si celle-ci supporte le ré-échantillonnage.

Son synthétisé

MIDI

Le protocole "Musical Intrument Digital Interface" (MIDI) fournit une méthode standardisée et efficace pour représenter une performance musicale sous forme de données électroniques. Un fichier MIDI peut être très petit par rapport à un fichier audio traditionnel, car il ne contient que les informations nécessaires au synthétiseur pour jouer les sons.

Les deux technologies qui sont utilisées dans les synthétiseurs musicaux sont :

- Frequency Modulation (FM) Synthesis : une vielle méthode qui réalise une approximation de la courbe de chaque instrument en utilisant un certain nombre de paramètres; généralement de mauvaise qualité, peu coûteux à implémenter, et donc utilisé sur beaucoup de cartes son.
- Wavetable Synthesis : une nouvelle méthode utilisant de petits enregistrements digitalisés de vrais instruments; meilleure qualité de son, requiert de la mémoire intégrée à la carte, on la trouve sur des cartes un peu plus chères.

La plupart des informations concernant MIDI sur OpenBSD peuvent être trouvées dans la page de manuel midi(4).

L'outil principal pour prendre en charge les fichiers MIDI standards est midiplay(1). Pour obtenir une liste des périphériques MIDI disponibles sur votre système, essayez ceci :

```
$ midiplay -l
0: SB MIDI UART
1: SB Yamaha OPL3
2: PC speaker
```

Dans cet exemple, nous voyons une sortie UART sur laquelle un périphérique externe peut être connecté, le synthétiseur intégré Yamaha OPL3 FM et le classique vieux PC Speaker.

Remarque : Toutes les cartes ne disposent pas de synthétiseur MIDI intégré, vous pourriez ne voir qu'une sortie UART et un "PC speaker" listé.

Jouer un fichier MIDI standard, dans cet exemple à travers le synthétiseur OPL3, est aussi simple que :

```
$ midiplay -d 1 file.mid
```

Remarquez que nous avons spécifié le périphérique MIDI de numéro 1, car le périphérique 1 est utilisé par défaut.

Plus d'informations : Tutorial sur la synthétisation MIDI et de Musique

MOD

Un module "Soundtracker" est un format binaire qui mixe les plages audio avec des ordres de séquences, autorisant à jouer de très grands morceaux de musique digitalisée avec un qualité raisonnablement bonne.

Le moyen le plus simple de jouer vos fichiers MOD favoris sur OpenBSD est probablement d'utiliser le logiciel XMMS, disponible via les paquetages et les ports. Vous devriez installer le "subpackage" `-mikmod` pour XMMS afin de lui permettre d'utiliser la bibliothèque sonore MikMod, qui supporte les modules de formats MOD, S3M, IT et XM.

Vous trouverez aussi liste de "trackers" dans la section `audio` de la collection de ports et de paquetages, comme par exemple tracker, soundtracker. Avec ces trackers, vous ne pouvez pas seulement jouer mais aussi générer vos propres modules. Remarquez que tous les trackers ne sont pas supportés par les outils de l'arbre des ports. Vous êtes invités à envoyer un port de votre programme de tracker favori.

13.3 - Comment jouer des CDs dans OpenBSD ?

Pour jouer un CD audio, en utilisant la sortie analogique de votre lecteur CD-ROM, vous pouvez

- Utiliser la sortie casque audio, souvent présente sur la face avant de votre lecteur de cdrom.
- Connecter la sortie audio sur le connecteur noir de votre carte audio. Oui, il s'agit d'un câble supplémentaire près de celui des données (SCSI/IDE) et de celui de l'alimentation.

Un utilitaire pratique en ligne de commande appelé cdio(1), à été inclut dans le système de base. Appelé sans paramètres, il entrera en mode interactif. Si vous souhaitez jouer un CD-ROM, entrez :

```
$ cdio play
```

Cela lira depuis le premier lecteur CD-ROM, `cd0`, par défaut. Notez que l'utilisateur qui lance cdio doit bénéficier des permissions de lecture sur le périphérique CD-ROM (ex. `/dev/rcd0c`). Etant donné que par défaut ce périphérique n'est accessible en lecture que par root et le groupe "operator", pour plus de facilité vous pouvez ajouter l'utilisateur au groupe operator en éditant la ligne correspondante dans `/etc/group`. Une autre solution est de modifier les permissions du périphérique comme vous le souhaitez.

Vous aurez peut-être besoin d'activer l'entrée CD au niveau du mixeur de sons. Au même titre que les sorties, le nom réel de ce canal d'entrée varie selon les systèmes. Mais la commande à utiliser ressemble à la commande suivante :

```
$ mixerctl inputs.cd.mute=off
```

Si vous préférez une belle GUI, il y a beaucoup de lecteurs de CDs basés sur X11 dans la collection de paquetages et de ports. Consultez simplement la section `audio`.

13.4 - Puis-je utiliser OpenBSD pour enregistrer des séquences audio ?

Oui, pour le faire vous pouvez utiliser /dev/sound ou /dev/audio comme périphérique d'entrée.

Premièrement, configurez les paramètres d'enregistrement corrects à l'aide de audioctl(1), exemple :

```
record.encoding=mulaw
record.rate=8000
record.channels=1
record.precision=8
```

J'utilise ici une quantisation non uniforme avec l'algorithme mu-law, un seul canal, une fréquence de 8000 Hz et 8 bits par morceau. Les algorithmes mu-law et A-law sont spécialement utiles lors de la digitalisation de la voix, car il fournissent une meilleur efficacité au codage. Cela signifie que la qualité de l'enregistrement sera meilleure pour un nombre de bit par morceau donné, ou que moins de bits seront requis pour une qualité donnée.

Si vous décidez d'adopter les valeurs ci-dessus (seulement celles-ci !), vous pouvez utiliser le périphérique /dev/audio qui les utilise par défaut, donc vous n'avez pas besoin de les configurer explicitement ici.

Puis, soyez sur de sélectionner le bon matériel pour enregistrement et que la source n'est pas muette. Vous pouvez configurer les paramètres nécessaires en utilisant mixerctl(1). Par exemple :

```
inputs.mic.mute=on
inputs.mic.preamp=on
inputs.mic.source=mic0
```

```
record.source=mic
record.volume=255,255
record.volume.mute=off
record.mic=0
record.mic.mute=off
```

Dans cet exemple j'enregistrerais à partir d'un microphone. La pré-amplification a été activée, car sinon le son enregistré peut être quelque peu silencieux.

Pour faire l'enregistrement utilisez simplement cat(1) ou dd(1):

```
$ dd if=/dev/audio of=myvoice.raw
```

Appuyez sur [CTRL]-C pour mettre fin à l'enregistrement. La sortie est une séquence d'octets brute. Ce son peut être joué, comme expliqué dans Jouer différents formats audio. Pour un test rapide, en considérant que les paramètres audio sont configurés correctement :

```
$ dd if=myvoice.raw of=/dev/audio
```

Une fois de plus si vous spécifiez d'autres paramètres d'encodage, vous devrez utiliser le périphérique /dev/sound. Un autre exemple de paramètres d'encodage :

```
record.encoding=slinear_le
record.rate=22050
record.channels=2
record.precision=8
```

Le résultat sera un PCM avec une quantisation linéaire (uniforme) signée, enregistré dans l'ordre little-endian à une fréquence de 22050 Hz, en stéréo, et en utilisant 8 bits pour représenter une morceau. (2^8 = 256 niveaux de quantisation).

Note : Vous souhaiterez probablement convertir cet enregistrement brut (sans en-tête) en un format utilisable pour pour un traitement ultérieur. Lisez FAQ 13 - Conversion afin d'en savoir plus.

13.5 - Parlez moi de l'encodage Ogg Vorbis et MP3 ?

Ces formats ont déjà été mentionnés dans Jouer différents formats audio. Dans cette section nous donnerons une brève introduction à l'encodage de tels fichiers. Si vous êtes intéressé par le fonctionnement de ces codecs de compression audio, de plus amples informations peuvent être trouvées dans les articles Wikipedia concernant Vorbis et MP3.

Ogg Vorbis

L'encodage de formats audio raw, WAV ou AIFF en Ogg Vorbis peut être réalisé via l'utilitaire **oggenc**, faisant partie du paquetage `audio/vorbis-tools`, qui est disponible via le système de ports et de paquetages d'OpenBSD.

Disons que vous avez un certain nombre de fichier WAV à encoder, par exemple votre album favori que vous venez juste d'extraire de son CD. Pour encoder tous ces fichiers en utilisant une fréquence d'environ 192 kbps, vous pouvez utiliser une commande comme celle ci :

```
$ oggenc *.wav -b 192
```

Une fois terminé, cela vous fournira un ensemble de fichiers .ogg dans le dossier courant. Des exemples plus précis ainsi que les options d'encodage peuvent être trouvés dans le manuel oggenc.

MPEG-1 Audio Layer 3 (MP3)

Si pour quelque raison que ce soit, vous souhaitez utiliser le format MP3, vous pouvez utiliser "Lame ain't an MP3 encoder" (LAME), un outil d'éducation utilisé pour apprendre l'encodage MP3. Lame est inclut dans l'arbre des sources OpenBSD. Notez que pour des raisons de brevets MP3, vous ne trouverez pas ce paquetage dans le Set de CDs officiels.

Ci-dessous ce trouve un exemple simple d'encodage d'un fichier WAV avec une fréquence de 192 kbps :

```
$ lame -b 192 track01.wav track01.mp3
```

Pour toutes les options et les détails, merci de consulter le manuel fournit avec lame.

13.6 - Comment lire des vidéos DVDs sous OpenBSD ?

Pour l'instant, OpenBSD supporte les médias DVD à travers le système de fichiers ISO 9660 qui est aussi utilisé pour les CD-ROMs et, depuis la version 3.8, également à travers le système de fichiers plus récent, Universal Disk Format (UDF), présent sur certains DVDs. Cependant, presque tous les disques DVD-Video et DVD-ROM utilisent le format passerelle UDF, qui est une combinaison des systèmes de fichiers DVD MicroUDF (sous-ensemble de UDF 1.02) et ISO 9660. Ce format est utilisé afin de garantir la compatibilité ascendante.

Puisque la plupart des ordinateurs disposant d'un lecteur DVD-ROM utilisent un décodage logiciel, il est recommandé d'avoir au moins un processeur Pentium II 350-MHz ou équivalent pour disposer d'une bonne qualité d'écoute.

Quelques utilitaires de lecture DVD populaires ont été portés sur OpenBSD. Comme par exemple ogle et mplayer. Veuillez lire les instructions d'installation livrées avec le paquetage, car ces fichiers peuvent nécessiter plus de configuration. Avec ces utilitaires il est possible de lire des DVD directement en accédant au périphérique raw. Bien sur, il est possible de monter d'abord le DVD en utilisant mount_cd9660(8), et de jouer les fichiers sur ce système de fichiers ou un autre.

Remarque :

- Quasiment tous les DVDs que vous achetez sont encryptés en utilisant le "Content Scrambling System (CSS)". Pour pouvoir jouer ces DVDs encryptés, vous pouvez utiliser la bibliothèque **libdvd** disponible via les paquetages et les ports.
- Soyez averti qu'un code région doit être présent sur votre disque DVD. Cela ne devrait pas réellement être un problème lorsque les DVDs sont joués sur un ordinateur.

13.7 - Comment graver des CDs et DVDs ?

13.7.1 - Introduction et configuration de base

Vous devriez tout d'abord être sur que votre graveur CD/DVD a bien été reconnu et configuré par le noyau. La plupart des périphériques SCSI sont reconnus. Les matériels IDE/ATAPI et USB sont supportés au travers de l'émulation SCSI. Vous trouverez rapidement votre périphérique dans la sortie d'un dmesg(1). Cherchez simplement les lignes commençant par "cd", par exemple :

```
cd0 at scsibus0 targ 0 lun 0: <TOSHIBA, CD-ROM XM-5702B, 2826> SCSI0 5/cdrom removable
cd1 at scsibus1 targ 4 lun 0: <PLEXTOR, CD-R PX-R412C, 1.04> SCSI2 5/cdrom removable
```

Mais `cdrecord -scanbus` ne fonctionne pas!

Oui. OpenBSD utilise un espace de nomage de périphérique différent de celui du système d'exploitation pour lequel cdrecord a été écrit. Tous les périphériques doivent être dans la sortie dmesg, comme mentionné ci-dessus. L'information dont vous avez besoin se trouve ici.

Error: `mount_cd9660: /dev/cd2c on /mnt/cdrom: No such file or directory`

Par défaut, l'installeur OpenBSD ne crée que deux périphériques cd `cd0` et `cd1`. Pour commencer à utiliser votre périphérique `cd2`, vous devez créer le périphérique nécessaire pour celui-ci. La méthode recommandée pour cela est d'utiliser le script MAKEDEV(8) (Sélectionnez votre plate-forme) :

```
# cd /dev
# ./MAKEDEV cd2
```

Dans ce qui suit, nous accèderons principalement au graveur de CD/DVD à travers le périphérique *raw* et **non** le périphérique *block*.

Vérifier le fonctionnement du graveur de CD/DVD

Il est recommandé de vérifier que votre graveur cd CD/DVD fonctionne correctement. Dans cet exemple, j'utilise un graveur DVD USB 2.0 :

```
cd2 at scsibus2 targ 1 lun 0: <LITE-ON, DVDRW LDW-851S, GS0C> SCSI0 5/cdrom removable
```

Essayez de l'utiliser en y montant un CD/DVD. Si vous le souhaitez, vous pouvez aussi vérifier les taux de transfert que vous obtenez en copiant des fichiers sur votre disque dur. La commande time(1) sera votre meilleure amie.

Si quelque chose se passe mal ici et que vous obtenez des erreurs durant cette phase, il est important de les corriger avant de commencer à écrire un CD/DVD.

Je souhaite maintenant graver un CD ! Pouvons nous commencer ?

Auparavant, il est judicieux de garder quelques conseils en tête :

- Ne lancez pas de tâches utilisant lourdement le disque pendant que vous écrivez un CD/DVD. Cela réduira la vitesse d'écriture vers votre graveur CD/DVD. Si la vitesse descend plus bas que ce que le graveur attend pendant un temps trop long, son buffer sera vide. Ce phénomène est aussi connu sous le nom de "buffer underrun".
- Evitez les chocs pendant la gravure, cela pourrait faire glisser le laser de sa piste, et pourra conduire à des erreurs sur le disque.
- Tous les graveurs de DVD ne supportent pas tous les formats DVD, voyez ci-dessous.

13.7.2 - Graver des CDs

Créer des CD-ROMs de données

Tout d'abord, vous devrez créer un système de fichier ISO 9660 à mettre sur un CD-ROM. Pour cela vous pouvez utiliser l'utilitaire mkhybrid(8) fournit dans le système de base ou mkisofs du paquetage cdrtools qui peut être plus performant avec de grosses arborescences. Dans l'exemple suivant, nous utiliserons mkisofs bien que l'utilisation de mkhybrid soit similaire.

Comme exemple d'utilisation, j'ai essayé de sauvegarder les sources d'OpenBSD dans une image ISO 9660 :

```
$ mkisofs -R -o sys.iso /usr/src/sys

Using ALTQ_RMC.000;1 for  /usr/src/sys/altq/altq_rmclass_debug.h (altq_rmclass.h)
...
Using XFS_DEV-.000;1 for  /usr/src/sys/xfs/xfs_dev-common.c (xfs_dev-bsd.c)
 12.06% done, estimate finish Sun Mar 27 21:18:47 2005
 24.09% done, estimate finish Sun Mar 27 21:18:43 2005
...
 84.32% done, estimate finish Sun Mar 27 21:18:38 2005
 96.37% done, estimate finish Sun Mar 27 21:18:38 2005
Total translation table size: 0
Total rockridge attributes bytes: 0
Total directory bytes: 1822720
Path table size(bytes): 10674
Max brk space used 469000
41508 extents written (81 Mb)
```

L'option `-R` demande à `mkisofs` de rajouter des extensions Rock Ridge dans l'image ISO 9660. Le "Rock Ridge Interchange Protocol" a été crée pour supporter le système de fichiers POSIX à l'intérieur du système de fichiers ISO 9660, comme les longs nom de fichiers, les créateurs de fichiers, les permissions, les liens de fichiers, les fichiers de périphériques, les hiérarchies de fichiers profondes (plus de 8 niveaux de sous-dossiers), etc.

Si vous souhaitez que les longs noms de fichiers sur votre CD-ROM soient lisibles sur les systèmes Windows et DOS, vous devriez ajouter le paramètre `-J` pour inclure les extensions Joliet dans l'image ISO 9660.

Après avoir crée les systèmes de fichiers, vous pouvez le vérifier en montant l'image ISO 9660. Si tout se passe bien, vous êtes maintenant prêt à écrire le CD-R(W). Un paquetage souvent utilisé pour cela est le programme **cdrecord**, inclus dans le paquetage cdrtools, disponible dans le système de port et de paquetages OpenBSD.

Si vous utilisez un média non réinscriptible, comme un CD-R, il est souvent judicieux de faire un essai avant de graver effectivement le CD- ROM, dans le but de ne pas gaspiller de disques.

```
# cdrecord -v -dummy dev=/dev/rcd1c sys.iso
```

Si tout se passe bien, vous êtes prêt à graver l'image créée dans l'exemple précédant sur un CD-R(W) vierge. Vous pourrez utiliser une commande similaire à :

```
# cdrecord -v dev=/dev/rcd1c sys.iso
```

En utilisant les options spécifiées ci-dessus, nous demandons à cdrecord de fournir une sortie verbeuse et d'utiliser le second périphérique CD- ROM comme graveur de CD. Notez que nous passons le périphérique *raw* à cdrecord.

Pour vérifier que le CD-ROM a bien été écrit, vous pouvez le monter et vérifier que tout ce qui doit y être y est. Pour monter le système de fichier, vous devez utiliser le périphérique *block* pour le lecteur de CD-ROM, qui dans ce cas est toujours le graveur de CD :

```
# mount /dev/cd1c /mnt/cdrom
```

Créer des CDs audio

Pour graver des CDs audio, vous pouvez utiliser **cdrecord** avec le paramètre `-audio`, ou bien utilisant l'utilitaire cdrdao pour écrire dans le mode Disc-At-Once et éviter les trous entre les différentes pistes audio. Chacun de ces utilitaires sont disponibles au travers du système de ports et de paquetages OpenBSD.

Comme exemple, je ferais une copie de sauvegarde d'un de mes CDs de musique. Cela implique deux étapes :

1. Copier les pistes audio depuis votre CD original: vous pouvez faire cela a l'aide de logiciels comme **cdparanoia**, **cdda2wav** (inclus dans le paquetage `sysutils/cdrtools`), et **cdrdao**. Par exemple :

   ```
   # cdrdao read-cd --device /dev/rcd0c cdtracks.toc
   ```

   Cette commande va extraire uen série de fichiers WAV depuis votre premier lecteur cd CD-ROM vers votre disque dur, et compiler un fichier de "Table of Contents (TOC)" (Table de matières).
2. Ecrire les pistes audio sur un cd vierge comme mentionné ci-dessus en utilisant **cdrecord** ou **cdrdao**, exemple :

   ```
   # cdrdao write --device /dev/rcd1c cdtracks.toc
   ```

   Si vous le souhaitez, vous pouvez essayer la commande `cdrdao simulate`au préalable.

13.7.3 - Graver de DVDs

Il y a quelques détails importants à propos des DVD que vous devriez savoir avant de créer vos propres DVDs.

Remarques importantes :

- Si vous souhaitez vraiment tout savoir sur les DVD, je vous suggère de lire la très complète FAQ DVD.
- Cette section a été très peu testée et nous n'avons probablement pas testé toutes les combinaisons de médias et de graveurs possibles. Cependant nous avons eu où reçu des retours d'expériences positifs avec les formats de DVD suivants. Vous êtes invités à nous informer de vos succès ou échecs.

Differents formats de DVDs

Il y a plusieurs formats différents pour les DVD. Les plus utilisés sont les formats DVD-R, DVD-RW, DVD+R et DVD-RW (R signifie enregistrable une seule fois, RW signifie qu'il peut être réécrit quelques milliers de fois). Il existe d'autres standards plus compétitifs.

Un format différent est le DVD-RAM, qui a principalement été développé comme un lecteur de données et dispose de fonctions d'écriture de paquets avancées, autorisant à l'utiliser comme une sorte de disque dur optique. DVD-RAM n'est pas recommandé pour une utilisation vidéo car les vidéos sont écrites sur le disque dans un format non compatible avec les lecteurs de DVD normaux.

Le point le plus important est d'utiliser le média adapté à votre graveur de DVD. Si vous souhaitez la compatibilité avec d'autres lecteurs de DVD, veuillez lire cette section de la FAQ DVD.

Vitesse de gravure DVD

Il est important de spécifier que les indications de vitesses des DVDs diffèrent de celles des CD-ROMs. La table suivante en donne un aperçu :

Vitesse de Lecture/Ecriture DVD	Taux de transfert (Mo/s)	Vitesse de Lecture/Ecriture CD-R(W)
1x	1.32	9x
2x	2.64	18x
4x	5.28	36x
8x	10.57	72x

Comme on peut le voir dans le tableau, les taux de transfert sont relativement hauts, et vous devrez vérifier que votre bus (SCSI, (E)IDE/ATAPI, USB) est assez performant pour les supporter. Spécifiquement, les vielles interfaces USB 1.0 et 1.1 travaillent à des taux plus bas, avec des vitesses de transfert de 1.5 Mbit/s et 12 Mbit/s, respectivement. Cela signifie que l'USB 1.0 a une vitesse maximum de 178.8 kByte/s et USB 1.1 de 1.43 MB/s. USB 2.0 est plus rapide : 480 Mbit/s ou 57.2 MB/s. En général, les vitesses des bus SCSI et (E)IDE/ATAPI sont correctes.

Graver les DVD

Basiquement, le processus est très similaire à celui d'écrire des CD- R(W)s. Le logiciel, cependant, diffère. A ce moment, la meilleure option est **growisofs** depuis le paquetage `sysutils/dvd+rw-tools`. Cet utilitaire écrit une image ISO 9660 sur le média DVD. Tous les formats DVD sont supportés par dvd+rw-tools.

Dans le cas où vous voudriez plus d'informations sur le média présent dans votre graveur de DVD (par exemple si vous avez perdu le texte d'information dans la boite du cd ou êtes simplement désorganisé comme moi), vous pouvez utiliser l'utilitaire **dvd+rw-mediainfo**. Il y a deux options pour écrire le DVD:

- Pre-masteuriser une image ISO 9660 depuis vos données, stockant l'image sur votre disque dur; puis l'écrire sur le DVD.
- Ecrire une image ISO 9660 de vos données directement sur le DVD.

J'ai crée une image ISO 9660 pré-masteurisée depuis les modules CVS d'OpenBSD (src, XF4, ports et www) présents dans le dossier /cvs de mon disque. J'ai utilisé la commande suivante, qui ressemble beaucoup à celle que j'ai utilisé pour créer l'image CD-ROM précédente.

```
$ mkisofs -R -o cvs.iso /cvs
```

Si vous le souhaitez, testez le système de fichiers ISO 9660 en montant l'image. Pour écrire cette image (environ 2Go) sur un disque DVD, on peut utiliser :

```
# growisofs -dvd-compat -Z /dev/rcd2c=cvs.iso
Executing 'builtin_dd if=cvs.iso of=/dev/rcd2c obs=32k seek=0'
/dev/rcd2c: pre-formatting blank DVD+RW...
/dev/rcd2c: "Current Write Speed" is 4.1x1385KBps.
23822336/1545832448 ( 1.5%) @3.9x, remaining 5:19
42172416/1545832448 ( 2.7%) @3.9x, remaining 5:20
60522496/1545832448 ( 3.9%) @3.9x, remaining 4:54
...
1504706560/1545832448 (97.3%) @3.9x, remaining 0:07
1523318784/1545832448 (98.5%) @3.9x, remaining 0:04
1541898240/1545832448 (99.7%) @3.9x, remaining 0:00
/dev/rcd2c: flushing cache
/dev/rcd2c: writing lead-out
/dev/rcd2c: reloading tray
```

L'option `-Z` demande a growisofs de graver une session initiale sur le périphérique, qui dans ce cas est mon graveur de DVD, attaché à cd2. L'option `-dvd-compat` finalise le disque, ce qui signifie qu'aucune nouvelle session ne peut y être rajoutée. Cela devrait fournir une meilleure compatibilité avec les lecteurs DVD et quelques vielles unités DVD-ROM.

Remarquez comment growisofs indique la vitesse d'écriture, dans notre cas 3.9x, ce qui est ce que l'on peut attendre en combinant les vitesses du média et du graveur, comme indiqué par le dvd+rw-mediainfo.

Si vous ne disposez pas de suffisamment de place pour stocker une image de ISO 9660 four un DVD, vous pouvez écrire vos données directement sur le DVD. Essayons simplement une simulation de création du système de fichiers.

```
# growisofs -dry-run -Z /dev/rcd2c -R /cvs
```

Si cela fonctionne, retirez simplement l'option -dry-run et commencez l'écriture du DVD.

```
# growisofs -Z /dev/rcd2c -R /cvs
```

Il est aussi possible d'ajouter des données à un DVD existant, en utilisant l'option -M, qui fond une nouvelle session dans une session existante :

```
# growisofs -M /dev/rcd2c -R /mydata
```

Pour plus d'informations à propos de growisofs, consultez sa page de manuel.

Lorsque vous avez terminé d'écrire le DVD, montez le et vérifiez que tout ce que vous souhaitiez présent y soit effectivement.

Pourquoi je n'obtiens pas la vitesse que je veux ?

Au lieu de la vitesse d'écriture précédente, vous pourriez voir quelque chose comme ceci :

```
4784128/1545832448 ( 0.3%) @0.7x, remaining 26:50
7929856/1545832448 ( 0.5%) @0.7x, remaining 29:05
14123008/1545832448 ( 0.9%) @0.7x, remaining 27:06
...
```

ce qui est bien plus lent. Cela signifie qu'il ne transite pas assez d'informations sur l'un des bus que votre DVD utilise. Dans l'exemple précédent, le graveur de DVD USB fut installé sur une machine sur laquelle le driver ehci(4), utilisé par les contrôleurs USB 2.0, ne s'est pas initialisé correctement. Comme d'habitude, vous êtes invités à envoyer des patchs et résultats de vos tests. Le graveur de DVD fut ralenti par l'interface USB 1.1 plus lente, ce qui a réduit la vitesse de gravure. L'USB 1.1 est limité à 12 Mbit/s, ce qui correspond à 1.43 MB/s ou 1.08x en terme de vitesse DVD. Le graveur de DVD est descendu à une plus basse vitesse de gravure pour diminuer le risque de buffer overrun (défaut de données à écrire).

13.8 - Je voudrais mes fichiers multimédia au format FOO.

Conversion entre différents formats audios

Admettons que vous souhaitiez convertir le son enregistré avec FAQ 13 - Enregistrer des séquences audio. Cet enregistrement a été enregistré en format brut. Il serait utile de le convertir car le format brut ("raw") ne contient pas d'en-tête et les paramètres devront être spécifiés à chaque utilisation du fichier.

Un bon outil de conversion de format est `audio/sox`, disponible au travers des ports et des paquetages. **sox** supporte les formats AIFF, AU, MP3, Ogg Vorbis, RIFF WAV et raw, ainsi que certains autres plus exotiques. Ci-dessous vous trouverez un exemple de conversion de l'enregistrement vers le format RIFF WAV.

```
$ sox -U -c 1 -r 8000 -b myvoice.raw myvoice.wav
```

Notez que les paramètres spécifiés correspondent aux paramètres d'enregistrement spécifiés avant l'enregistrement. Cela n'est qu'un exemple. De nombreux autres programmes et bibliothèques peuvent être utilisés en conversion audio.

Note : Il n'est pas recommandé de convertir entre différents formats de compression dits destructeurs ("lossy"). Par exemple, les codecs MP3 et Vorbis suppriment certaines informations du fichier audio original. Ainsi, lorsque vous convertissez un fichier MP3 en Ogg Vorbis, le résultat final sonnera certainement moins bon que l'original.

Conversion entre différents formats vidéos

Il est important de faire une distinction entre

- le format du containeur - par exemple, MP4, OGG, MPEG, MOV, AVI, ASF.
- le codec vidéo - par exemple, MPEG-1, MPEG-2, codecs compatibles MPEG-4 (comme Xvid et DivX), FFmpeg, WMV, ... - lisez l'article Wikipedia sur les codecs vidéos pour en savoir plus.

Pour le moment, le support des containeurs MPEG et AVI est le plus abouti. Aucun utilitaire présent dans l'arbre des ports n'est encore capable de créer des containeurs MP4.

Il existe deux utilitaires populaires, `multimedia/transcode` et **mencoder** (qui fait partie de `x11/mplayer`). Ils utilisent ou peuvent utiliser la bibliothèque **libavcodec** disponible dans le port `graphics/ffmpeg` et qui génère un format de sortie de bonne qualité. Vous pouvez bien sûr utiliser **ffmpeg** directement. Il devrait aussi être possible d'utiliser l'encodeur XviD présent dans `multimedia/xvidcore`.

La documentation, qui vient avec ces paquetages sous la forme de pages de manuel ou de documents HTML dans `/usr/local/share/doc`, contient de nombreux exemples, c'est pourquoi il est HAUTEMENT recommandé de lire ces documents.

13.9 - Est-il possible de lire des fichiers en streaming sous OpenBSD ?

Oui, c'est possible. De nombreux streams audio et vidéos fonctionneront sans soucis sur un nombre limité de plates-formes. Certains, quant à eux, ne fonctionneront pas.

Ce document ne représente pas une réponse exhaustive sur la manière de faire fonctionner tous les formats de streaming possibles sur n'importe qu'elle architecture. Tout d'abord, vous devriez vous renseigner sur la technologie de streaming. Quoi qu'un peu daté, le chapitre à propos du streaming tiré du livre "Designing Web Audio" aux éditions O'Reilly représente un bon début.

La première chose à savoir est qu'il existe plusieurs protocoles de streaming. Le **protocole** de streaming définit la façon dont les flux seront envoyés à travers le réseau. Ils ont été développés afin de garantir une transmission audio/vidéo efficace en temps réel par internet. En bref, le protocole de streaming est un protocole applicatif (niveau 7) capable d'utiliser les protocoles de transport UDP ou TCP (niveau 4). Le protocole de transport UDP ("User Datagram Protocol") convient parfaitement pour ce type d'application puisqu'il n'effectue aucune retransmission de paquets ou autre type de charge réseau. Un certain nombre de protocoles spécialisés mais propriétaires ont été développés, par exemple Microsoft Media Services (MMS - services média de Microsoft) et le Real Time Streaming Protocol (RTSP - protocole de transmission de flux en temps réel). Comme nous le verrons, HTTP (qui utilise TCP) est également parfois utilisé, même s'il ne permet pas de transmettre des flux à une vitesse régulière comme UDP, RTSP et MMS.

Ensuite, il y a le **format** de streaming qui représente la façon dont les données audio/vidéo ont été organisées et peuvent être lues. Les formats de streaming les plus utilisés sont le MP3, Real Audio (RA, RM) et le Windows Media (ASF) qui sont tous des technologies propriétaires. Parfois, vous rencontrerez certains streams au format Ogg Vorbis.

Pour l'exemple, j'expliquerai, en quelques étapes, comment j'arrive à écouter Radio 1, une des stations radios nationales belges. Sous OpenBSD, les plugins (modules externes) de navigateurs ne sont pas disponibles, ce qui signifie que la mise en place se résume rarement en un "clic et lecture".

- Déterminer le protocole de streaming et le format.
  Ceci est généralement indiqué sur le site web sur lequel vous accédez au stream. Dans ce cas, je clic simplement sur le lien "Listen live" (écoutez en direct) à partir du site principal qui me répond que mon système d'exploitation n'est pas supporté. Ils sont suffisamment polis pour me signifier que je peux également écouter leur stream MP3 sans utiliser leur lecteur Flash qui nécessite un plugin. Ensuite une liste de liens vers les stations radios nationales apparaît, ce qui me permet de passer à la suite. Notez que j'ai dû utiliser un navigateur compatible JavaScript afin d'arriver ici.
- Déterminer l'URL précise.
  De nombreux sites internet lient vers un containeur (du type M3U, ASX, RAM) qui contient la localisation effective du flux. Enregistrez simplement ce fichier et récupérez l'URL à partir de celui-ci. Dans mon exemple, il s'agit de :

  ```
  $ ftp http://internetradio.vrt.be/dab/hoeluisteren/pc/help/gebruiksvoorwaarden/stream_11.m3U
  $ cat stream_11.m3U
  http://mp3.streampower.be/radio1-mid.mp3
  http://mp3.streampower.be/radio1-low.mp3
  http://mp3.streampower.be/radio1-high.mp3
  ```

  Visiblement, je peux même choisir entre différentes qualités de flux ("low, medium et high"). D'autres sites peuvent contenir du code JavaScript pour créer l'URL. Dans ce cas, le meilleur conseil serait : parcourez le source HTML ainsi que les scripts auxquels il se réfère. Il existe une bonne chance que vous soyez en mesure de reconstruire l'URL à partir de ces données.
- Pour lire ces flux, votre meilleure chance est probablement `x11/mplayer`, disponible en tant que paquetage et dans l'arbre des ports. Il supporte la plupart des protocoles et formats de streaming et fonctionne sur les plates-formes amd64, i386, powerpc et sparc64. Mais il existe des alternatives : **ogg123** dans `audio/vorbis-tools` (pour les streams Ogg Vorbis), `audio/mpg123` et `audio/mpg321` (pour les streams MP3), XMMS dans `audio/xmms` et le client Videolan dans `x11/vlc`. Continuons l'exemple :

  ```
  $ mplayer http://mp3.streampower.be/radio1-mid.mp3
  ```

- Si vous le souhaitez, vous pouvez faciliter cela en ajoutant un alias dans votre fichier `.profile` :

  ```
  alias radio1='mplayer http://mp3.streampower.be/radio1-mid.mp3'
  ```

Les flux Windows Media (ASF) fonctionneront la plupart du temps bien qu'ils puissent contenir des données aux formats supportés uniquement grâce au port

`graphics/win32-codecs` qui ne tourne que sous i386 ('pkg_info win32-codecs' vous donnera la liste des codecs supportés). Certains flux Real Audio peuvent fonctionner sous i386 en utilisant **mplayer** en conjonction avec les ports `graphics/win32-codecs` et `emulators/redhat/base` (lisez ce fil de la liste de discussion des ports).

13.10 - Puis-je utiliser un plugin Java avec mon navigateur ? (spécifique à i386)

Le plugin Java fait partie du kit de développement Java ("Java Development Toolkit" - JDK). Pour des raisons de licence ,OpenBSD ne peut pas distribuer de paquetages binaires du JDK. Cela signifie que vous devrez le compiler à partir des ports. De plus amples informations sur la compilation du JDK sont disponibles dans FAQ 8 - Languages de programmation. Une fois le paquet créé, vous pouvez installer le JDK complet ou simplement l'environnement d'exécution Java ("Java Runtime Environment" - JRE) qui se trouve dans un sous-paquet ("subpackage") et contient le plugin web.

A la fin de l'installation, des instructions seront affichées afin d'utiliser le plugin Java avec Firefox ou un navigateur basé sur Mozilla. Créez le lien comme expliqué et vous devriez voir le plugin Java lorsque vous entrerez "about:plugins" dans la barre d'adresse.

Note : Le plugin java a seulement été testé avec les navigateurs Firefox et Mozilla. S'il fonctionne également pour vous avec un autre navigateur, n'hésitez pas à nous le faire savoir.

13.11 - Puis-je utiliser un plugin Flash avec mon navigateur ? (spécifique à i386)

Le plugin Flash est distribué par Macromedia sous forme binaire uniquement. Macromedia ne fourni pas de plugin natif pour OpenBSD mais il existe un plugin pour Linux qui peut être utilisé sous émulation. Ce plugin n'est disponible que pour les plateformes i386.

Avant de poursuivre, vous devriez vous renseigner sur l'émulation Linux dans la page de manuel compat_linux(8) ainsi que FAQ 9 - Exécution des binaires Linux sous OpenBSD.

Si vous avez assimilé ces documents mais n'avez pas encore installé les fichiers nécessaires, ajoutez simplement le paquetage redhat. En admettant que vous ayez correctement défini la variable `PKG_PATH` (voyez FAQ 15),

```
# pkg_add redhat_base-8.0p8
```

La première commande définira automatiquement `kern.emul.linux=1`, mais pas de façon permanente. Si vous avez besoin de l'émulation Linux en permanence, vous devez le spécifier dans `/etc/sysctl.conf`, comme expliqué dans FAQ 9 - Exécution des binaires Linux sous OpenBSD.

Une autre chose que vous devriez savoir est que les bibliothèques partagées ainsi que les modules Linux ne peuvent pas être utilisés avec des exécutables OpenBSD, ce qui signifie que vous aurez également besoin d'un navigateur Linux.

Un candidat serait le navigateur Opera, disponible dans l'arbre des ports. OpenBSD ne distribue pas de paquetage car la licence d'Opéra n'est pas claire à ce sujet. Cependant, l'installation ne devrait pas prendre de temps puisqu'il est distribué sous forme binaire par Opera Software. Après cela vous pouvez facilement installer le plugin Flash en utilisant les ports.

```
# cd /usr/ports/www/opera
# make install
# cd /usr/ports/www/opera-flashplugin
# make install
```

Note : La dernière étape devrait suffire en elle-même car le système des ports devrait installer les dépendances automatiquement. Cependant, dans un souci de clarté nous séparons le processus en plusieurs étapes afin d'en faciliter la compréhension.

Si vous avez suivi ce guide, le plugin Flash devrait être listé lorsque vous tapez "about:plugins" dans la barre d'adresse.

[Index de la FAQ] [Section 12 - Questions spécifiques aux plates-formes] [Section 14 - Configuration des disques]


www@openbsd.org

$OpenBSD: faq13.html,v 1.31 2006/09/22 12:14:39 jufi Exp $

[Index de la FAQ] [Section 13 - Multimédia] [Section 15 - Paquetages et Ports]

14 - Configuration des disques

Table des matières

14.1 - Utilisation de disklabel(8) sous OpenBSD

Qu'est-ce que disklabel(8) ?

Pour commencer, lisez le manuel de disklabel(8).

La façon de configurer les disques sous OpenBSD diffère légèrement selon les plates-formes. Sous i386, amd64, macppc, zaurus et cats, la configuration se déroule en deux étapes. Tout d'abord, la tranche contenant OpenBSD sur le disque est définie en utilisant fdisk(8) et ensuite divisée en partitions OpenBSD avec disklabel(8).

Toutes les autres plates-formes utilisent disklabel(8) pour la gestion complète des partitions OpenBSD. Les plates-formes qui utilisent aussi fdisk(8) placent toutes les partitions disklabel(8) dans une tranche fdisk.

Les labels contiennent certaines informations sur votre disque, comme sa géométrie ou les systèmes de fichiers présents sur celui-ci. Ils

contiennent également des données sur le disque lui-même comme la vitesse de rotation, l'ordonnancement etc, présentent pour des raisons historiques et souvent incorrectes. Il est inutile de s'en inquiéter. Le disklabel est utilisé par le programme de bootstrap pour accéder aux disques et connaître le type de systèmes de fichiers présents. Vous pouvez accéder à de plus amples informations sur disklabel en lisant le manuel disklabel(5).

Sur certaines plateformes, l'utilisation de disklabel permet de passer outre les limitations de partitionnement liées aux types d'architectures. Par exemple, sur i386, vous pouvez avoir 4 partitions primaires mais avec disklabel(8), vous utilisez une de ces partitions 'primaires' pour stocker *toutes* vos partitions OpenBSD (par exemple, 'swap', '/', '/usr', '/var', etc) et il vous reste encore 3 partitions disponibles pour d'autres systèmes d'exploitation !

disklabel(8) à l'installation d'OpenBSD

Une partie importante de l'installation d'OpenBSD est la création initiale des labels. A l'installation, vous utiliserez disklabel(8) afin de créer les différentes partitions. Vous pourrez définir vos points de montage à partir de disklabel(8) et le changer plus tard pendant l'installation ou même après celle-ci.

Il n'existe pas de "bonne" façon de créer un label disque mais il en existe beaucoup de mauvaises. Avant de créer un label sur votre disque, lisez ce chapitre sur le partitionnement.

Pour un exemple d'utilisation de disklabel(8) pendant l'installation, référez-vous à la configuration des disques du guide d'installation.

Utilisations de disklabel(8) après l'installation

Après l'installation, un des usages les plus courants de disklabel(8) est de vérifier la disposition de votre disque. La commande suivante affichera le disklabel courrant sans le modifier :

```
# disklabel wd0 <-- Selon le disque que vous souhaitez
vérifier
# Inside MBR partition 3: type A6 start 63 size 29880837

# /dev/rwd0c:
type: ESDI
disk: ESDI/IDE disk
label: Maxtor 51536H2
flags:
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total sectors: 29888820
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # microseconds
track-to-track seek: 0  # microseconds
drivedata: 0

16 partitions:
#             size        offset  fstype [fsize bsize  cpg]
  a:        614817            63  4.2BSD   2048 16384  328 # Cyl     0*-   609
  b:        409248        614880    swap                    # Cyl   610 -  1015
  c:      29888820             0  unused      0     0      # Cyl     0 - 29651*
  d:       6291936       1024128  4.2BSD   2048 16384  328 # Cyl  1016 -  7257
  e:        409248       7316064  4.2BSD   2048 16384  328 # Cyl  7258 -  7663
  f:       1024128       9822960  4.2BSD   2048 16384  328 # Cyl  9745 - 10760
```

```
  h:       2097648       7725312  4.2BSD   2048 16384  328 # Cyl  7664 -  9744
```

Notez que, pour le moment, ce disque n'a qu'une partie de son espace disponible allouée. Disklabel offre deux modes d'édition différents, un mode de commandes (utiliser à l'installation d'OpenBSD) et un éditeur complet tel que vi(1). Le mode de commandes peut paraître plus simple puisqu'il vous guide à travers les différentes étapes et propose une aide intégrée, mais le mode éditeur possède également une véritable utilité.

Ajoutons une partition au système précédent.

> *Attention : chaque fois que vous touchez à votre disklabel, vous mettez en danger toutes les données présentes sur le disque. Soyez sûr que vos données soient sauvegardées avant d'éditer un disklabel existant !*

Nous utiliserons le mode de commandes intégré qui est lancé avec l'option "-E" de disklabel(8)

```
# disklabel -E wd0
...
> a k
offset: [10847088]
size: [19033812] 2g
Rounding to nearest cylinder: 4194288
FS type: [4.2BSD]
> p m
device: /dev/rwd0c
type: ESDI
disk: ESDI/IDE disk
label: Maxtor 51536H2
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total bytes: 14594.2M
free bytes: 7245.9M
rpm: 3600

16 partitions:
#             size        offset  fstype [fsize bsize  cpg]
  a:        300.2M           0.0M  4.2BSD   2048 16384  328 # Cyl     0*-   609
  b:        199.8M         300.2M    swap                   # Cyl   610 -  1015
  c:      14594.2M           0.0M  unused      0     0      # Cyl     0 - 29651*
  d:       3072.2M         500.1M  4.2BSD   2048 16384  328 # Cyl  1016 -  7257
  e:        199.8M        3572.3M  4.2BSD   2048 16384  328 # Cyl  7258 -  7663
  f:        500.1M        4796.4M  4.2BSD   2048 16384  328 # Cyl  9745 - 10760
  h:       1024.2M        3772.1M  4.2BSD   2048 16384  328 # Cyl  7664 -  9744
  k:       2048.0M        5296.4M  4.2BSD   2048 16384   16 # Cyl 10761 - 14921
> q
Write new label?: [y]
```

Dans ce cas, disklabel(8) a été capable de calculer un bon point de départ pour la partition. La plupart du temps ce sera le cas mais si vous avez des "trous" dans votre disklabel (ex. vous avez supprimez votre partition ou vous aimez simplement vous compliquer la vie) il vous faudra prendre le temps de calculer un offset correct. Notez que même si disklabel(8) effectue certaines vérifications il est très possible de faire de grosses erreurs. Faites attention et comprenez bien les valeurs que vous utilisez.

Sur la plupart des plates-formes OpenBSD, il y 16 partitions disklabel disponibles, notées de "a" à "p" (certains systèmes "spécialisés" n'en ont que huit). Chaque disklabel doit avoir une partition 'c' avec un "fstype" défini à "unused" qui couvre l'ensemble du disque physique. Si ce n'est pas le cas de votre disklabel, il doit être fixé, l'option "D" (plus bas) peut être utilisée. Ne tentez jamais d'utiliser la partition "c" pour autre chose que l'accès en mode raw des secteurs du disque et n'essayez pas de créer un système de fichiers sur "c". Sur le périphérique de démarrage et seulement sur celui-ci, "a" est réservé pour la partition racine et "b" pour le swap. Tous les autres

périphériques peuvent utiliser l'ensemble des quinze partitions en dehors de 'c' pour leur système de fichiers.

Trucs et astuces sur disklabel

- **Obtenir de l'aide :** en mode commandes, taper "?" affichera une liste des commandes disponibles. "M" affichera la page manuel de disklabel(8).
- **Valeurs par défaut :** dans certains cas, vous pourriez avoir besoin de commencer avec disklabel par défaut. La commande "D" écrasera l'ancien disklabel comme s'il n'avait jamais existé pour le remplacer par un nouveau.
- **Dupliquer un disklabel :** dans certains cas, vous pourriez avoir besoin de dupliquer le partitionnement d'un disque sur un autre mais par précisément (par exemple, vous voudriez avoir les même partitions mais sur des disques de taille différente). Utilisez l'option '-e' (mode éditeur en plein écran) de disklabel(8) pour enregistrer les partitions du disque "source", copiez-les sur le nouveau disque, supprimez la partition 'c' du disque "source", enregistrez et vous aurez copié la disposition des partitions sur le sur le nouveau disque sans altérer ses paramètres de base.
- (sparc/sparc64) **ne placez pas la partition swap au début du disque.**
- (i386, amd64) **laissez la première piste ("track") libre :** sur certaines plateformes, vous devriez laisser la première piste logique inutilisée dans disklabel(8) et fdisk(8). Cette recommandation est parfois injustement précisée par "démarrez les partitions au secteur 63", ce qui n'est SEULEMENT valable que si elle correspond bien à la taille d'une piste sur votre matériel. N'assumez en rien que cela soit vrai, disklabel vous dira quel est le nombre de secteurs par piste. De nombreuses autres plates-formes s'attendent à voir les partitions OpenBSD démarrer au secteur 0.
- **Disques sans disklabel :** si un périphérique ne possède pas de disklabel OpenBSD mais a déjà un autre système de fichiers (par exemple, un disque avec un système de fichiers FAT32 déjà existant), OpenBSD va en "créer" un en mémoire qui posera les bases d'un disklabel à enregistrer sur le disque. Cependant, si un disklabel OpenBSD est créé et enregistrer sur les disque et qu'un système de fichiers non OpenBSD est ajouté par la suite, le disklabel ne sera pas automatiquement mis à jour. Vous devrez l'éditer vous-même si vous souhaitez que OpenBSD puisse y accéder. Plus d'informations sont disponibles plus bas.
- **"q" ou "x":** Pour des raisons historiques, en mode de commandes, "q" enregistre les changements et quitte le programme alors que "x" quitte sans sauvegarder. C'est l'opposé de ce que beaucoup de gens peuvent voir sous d'autres environnements. disklabel(8) demandera confirmation avant d'enregistrer les changements mais n'affichera aucune alerte si "x" est utilisée.

14.2 - Utilisation de fdisk(8) sous OpenBSD

Avant de commencer, lisez le manuel de fdisk(8).

fdisk(8) est utilisé sur certaines plates-formes (i386, amd64, macppc, zaurus et cats) pour créer une partition reconnue par la ROM de démarrage du système dans laquelle les partitions disklabel d'OpenBSD peuvent être inscrites. Les autres plateformes n'ont pas besoin d'utiliser fdisk(8). fdisk(8) peut également être utilisé pour la manipulation du MBR ("Master boot record") pouvant avoir un impact sur tous les systèmes présents sur la machine. A l'inverse de certains programmes dont les fonctionnalités sont proches de celles de fdisk, sous OpenBSD cette commande assume que vous savez ce que vous voulez faire et en règle générale n'interviendra pas dans vos décisions, ce qui en fait un outil très puissant. Il vous laissera également faire des choses que vous n'aviez pas forcément prévues et doit donc être utilisé avec précaution.

Normalement, seule une partition fdisk OpenBSD sera inscrite sur le disque. Cette partition sera divisée par disklabel en différentes partitions systèmes de fichiers OpenBSD.

Pour simplement voir votre table de partition avec fdisk, utilisez :

```
# fdisk sd0
```

Ce qui vous donnera une sortie similaire à celle-ci :

```
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
```

Dans cet exemple, nous voyons la sortie de fdisk concernant le premier lecteur SCSI. Nous pouvons voir les partitions OpenBSD (A6). Le * indique que la partition OpenBSD est amorçable.

Dans l'exemple précédent, nous avons simplement accédé à une information. Qu'en est-il si nous souhaitons éditer notre table de partition ? Et bien, pour ce faire, nous devons utiliser l'option **- e**. Ceci fera apparaître une invite de commandes pour interagir avec 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
        swap            Swap two partition entries
        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>
```

Voici un aperçu des commandes disponibles lorsque vous utilisez l'option **-e**.

- **help** Affiche la liste des commandes utilisables par fdisk en mode d'édition interactif.
- **reinit** Initialise la copie en mémoire du bloc de démarrage actuellement sélectionné. Pratique pour initialiser une partition OpenBSD sur tout le disque, mettre à jour le code de démarrage et, de manière générale, préparer le système pour OpenBSD (et OpenBSD seulement).
- **disk** Affiche la géométrie actuelle du disque détectée par fdisk. Vous pouvez l'éditer selon vos désirs.
- **setpid** Change l'identifiant de partition de l'entrée sélectionnée de la table de partition.
- **edit** Edite l'entrée sélectionnée de la table dans la copie mémoire du bloc de démarrage actuel. Vous pouvez entrer les valeurs en géométrie BIOS ou en secteurs, offsets et tailles.
- **flag** Rend amorçable la partition sélectionnée. Seule une entrée peut être marquée comme amorçable. Si vous souhaitez démarrer à partir d'une partition étendue, vous devez la marquer comme amorçable.
- **update** Met à jour le code machine dans la copie mémoire du bloc de démarrage actuellement sélectionné.
- **select** Sélectionne et charge en mémoire le bloc de démarrage désigné par la table de partition étendue dans le bloc de démarrage actuel.
- **swap** Echange deux entrées MBR afin que vous puissiez le réordonner.
- **print** Affiche la copie en mémoire actuelle du bloc de démarrage et de son MBR sur le terminal.
- **write** Ecrit la copie en mémoire actuelle du bloc de démarrage sur le disque. Vous serez amené à confirmer cette action.
- **exit** Quitte le niveau actuel de fdisk, revenant à la copie précédente du bloc de démarrage ou sortant du programme si une telle copie n'existe pas.
- **quit** Quitte le niveau actuel de fdisk, revenant à la copie précédente du bloc de démarrage ou sortant du programme si une telle copie n'existe pas. A l'inverse d'"exit", le bloc de démarrage modifié est inscrit sur le disque.
- **abort** Quitte le programme sans sauvegarder les changements.

trucs et astuces fdisk

- fdisk(8) permet d'éditer les partitions soit en mode secteurs purs, soit en utilisant les Cylindres/Têtes/Secteurs. Ces deux options sont disponibles pour une raison ; certaines tâches sont plus simples à effectuer d'une façon ou d'une autre. Ne vous limitez pas à l'utilisation d'une seule de ces deux options.
- Un disque vierge aura besoin qu'on y inscrive le code de démarrage dans le MBR avant qu'il ne puisse booter. Pour ce faire, vous pouvez utiliser les options "reinit" ou "update". Si vous ne le faites pas, vous écrirez une table de partitions valide avec fdisk mais vous ne pourrez pas démarrer. Si vous ne savez pas d'où vient de code de démarrage présent sur votre disque, il est possible que vous souhaitiez le mettre à jour de toutes façons.
- Si votre système possède une partition de "maintenance" ou de "diagnostique", il est recommandé de la laisser en place ou de l'inscrire avant d'installer OpenBSD.
- Pour des raisons historiques, "q" enregistre les changements et quitte le programme alors que "x" quitte sans sauvegarder. C'est l'opposé de ce que beaucoup de gens peuvent voir sous d'autres environnements. disk(8) ne demandera aucune confirmation avant d'enregistrer les changements alors utilisez cette commande avec précaution

14.3 - Ajout de nouveaux disques sous OpenBSD

Une fois votre disque **CORRECTEMENT** installé, vous devez utiliser fdisk(8) (*i386 seulement*) et disklabel(8) afin de le configurer pour OpenBSD.

Pour les utilisateurs i386, commencez avec fdisk. Les autres architectures peuvent ignorer cette étape. Dans l'exemple suivant, nous ajouterons un troisième disque SCSI au système.

```
# fdisk -i sd2
```

Cette commande va initialiser la "véritable" table de partition du disque pour un usage exclusif par OpenBSD. Ensuite, vous devez créer un disklabel. Ceci risque de sembler confus.

```
# 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)
```

Tout d'abord, ignorez la partition 'c', celle-ci est toujours présente et est nécessaire au fonctionnement de programmes comme disklabel ! Le type de système de fichiers ("fstype") pour OpenBSD est 4.2BSD. Le nombre total de secteurs ("total sectors") représente la taille du disque. Admettons qu'il s'agit d'un disque de 3 gigabytes. 3 gigabytes en terme de constructeur de disques représente 3000 megabytes. Divisons 6185088 par 3000 (on utilise bc(1)). Vous obtenez 2061. Ainsi, pour décider des tailles des partitions a, d, e, f, g, ... multipliez simplement X par 2061 afin d'avoir X megabytes d'espace sur cette partition. L'offset de votre première partition doit être égal à "sectors/track" comme rapporté par disklabel. Pour nous, cette valeur est égale à 63. L'offset de chaque partition suivante résultera de la combinaison de la taille et de l'offset de chacune d'elle (sauf la partition 'c' qui n'intervient pas dans cette équation).

Ou, si vous souhaitez simplement une partition couvrant tout le disque, afin d'offrir un espace de stockage web, un répertoire d'utilisateurs ("home") ou autre chose, ôtez simplement la valeur "sectors/track" à la taille totale du disque. 6185088-63 = 6185025. Votre partition est :

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

Si tout ceci vous semble inutilement complexe, vous pouvez simplement utiliser disklabel -E pour entrer dans le même mode de

partitionnement dont vous disposiez à l'installation ! Dans celui- ci, vous pouvez utiliser "96M" pour spécifier "96 megabytes" (ou, si vous possédez un disque suffisamment gros, 96G pour 96 gigabytes). En revanche, le mode -E utilise la géométrie BIOS du disque, pas sa véritable géométrie, et souvent ces deux valeurs sont différentes. Pour passer outre cette limitation, tapez 'g d' pour "geometry disk" (géométrie disque). Les autres options sont 'g b' for "geometry bios" (géométrie bios) et 'g u' pour "geometry user" (géométrie utilisateur) ou simplement, ce que le label affichait avant que disklabel n'effectue des changements.

Voilà qui était dense. Mais tout n'est pas terminé. Vous devez à présent créer un système de fichiers sur ce disque en utilisant newfs(8).

```
# newfs sd2a
```

Ou autre, selon le nom donné à votre disque par le système de nommage d'OpenBSD (regardez la sortie de dmesg(8) pour voir comment OpenBSD a nommé votre disque).

Maintenant, il vous faut décider où vous souhaitez monter cette partition nouvellement créée. Disons que vous vouliez la monter sur /u. Tout d'abord, créez le répertoire /u. Puis montez-le.

```
# mount /dev/sd2a /u
```

Enfin, ajoutez-le a /etc/fstab(5).

```
/dev/sd2a /u ffs rw 1 1
```

Et si vous souhaitiez migrer un répertoire existant comme /usr/local ? Vous devriez monter le nouveau disque sur /mnt et utiliser `cpio - pdum` pour copier /usr/local vers le répertoire /mnt. Editez le fichier /etc/fstab(5) afin de préciser que votre partition /usr/local est maintenant /dev/sda2 (votre nouvelle partition formatée). Exemple :

```
/dev/sd2a /usr/local ffs rw 1 1
```

Redémarrez en mode single user avec **boot -s**, déplacez le répertoire /usr/local vers /usr/local-backup (ou si vous vous sentez en veine, supprimez-le) et créez un répertoire /usr/local vide. Enfin, redémarrez le système et voilà, les fichiers sont présents !

14.4 - Comment créer un espace de pagination dans un fichier

(Note : si vous souhaitez ajouter un espace de pagination dans un fichier parce que vous recevez des erreurs du type "virtual memory exhausted", vous devriez d'abord essayer d'augmenter la limite des processus avec unlimit(1) pour csh ou ulimit(1) pour sh)

Il n'est pas nécessaire de recompiler le noyau pour utiliser un fichier comme espace de pagination, même si cela peut être effectué à des fins de personnalisation, cette faq vous montrera les deux façons de faire.

Paginer dans un fichier.

L'utilisation d'un fichier d'échange est le moyen le plus simple et le plus rapide pour augmenter l'espace de pagination (swap) disponible. Ce fichier ne doit pas résider sur un système de fichiers dont l'option "SoftUpdates" est activée (cette option est désactivée par défaut). Pour commencer, vous pouvez voir la taille totale et utilisée de votre swap actuel en utilisant l'utilitaire swapctl(8). Pour ce faire, lancez la commande :

```
$ swapctl -l
Device      512-blocks     Used    Avail Capacity  Priority
swap_device      65520        8    65512     0%    0
```

Cela vous affiche les différents dispositifs utilisés pour la pagination ainsi que leurs statistiques actuelles. Dans l'exemple précédent, il n'y a qu'un seul dispositif nommé "swap_device". Il s'agit de l'espace disque prédéfini utilisé pour paginer (partition b visible sous disklabel). Comme vous pouvez le constater dans l'exemple précédent, cet espace n'est pas beaucoup utilisé pour le moment. Mais pour les besoins de ce document, nous imaginerons que 32M supplémentaires sont nécessaires.

La première chose à faire pour configurer un fichier en tant que dispositif de pagination est de créer ce fichier. Le mieux est d'utiliser la commande dd(1). Voici un exemple montrant la création d'un fichier */var/swap* de 32M.

```
$ 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)
```

Une fois que cela est fait, nous pouvons activer la pagination vers ce fichier. Pour ce faire, utilisez la commande suivante :

```
$ sudo chmod 600 /var/swap
$ sudo swapctl -a /var/swap
```

A présent, vérifions que ce fichier a bien été ajouté à la liste des dispositifs de pagination disponibles.

```
$ 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%
```

Maintenant que le fichier de pagination est activé, vous devez ajouter une ligne à votre fichier */etc/fstab* afin que les changements soient pris en compte au prochain redémarrage. Si cette ligne n'est pas ajoutée, ce dispositif ne sera plus activé au prochain reboot.

```
$ cat /etc/fstab
/dev/wd0a / ffs rw 1 1
/var/swap /var/swap swap sw 0 0
```

Paginer via un périphérique vnode

Il s'agit d'une solution plus définitive pour ajouter de l'espace de pagination. Afin de paginer vers un fichier de façon permanente, commencez par créer un noyau avec vnd0c comme swap. Si votre système de fichiers racine est wd0a, alors wd0b est votre ancien swap ; utilisez la ligne suivante dans le fichier de configuration du noyau (référez-vous à la section de compilation d'un nouveau noyau en cas de doute) :

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

Après cela, le fichier qui sera utilisé pour paginer devra être créé. Pour ce faire, utilisez la même commande que pour l'exemple précédent.

```
$ 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)
```

A présent que votre fichier est en place, vous devez l'ajouter dans votre fichier */etc/fstab*. Voici une ligne d'exemple pour démarrer la machine avec ce dispositif de pagination activé.

```
$ cat /etc/fstab
/dev/wd0a / ffs rw 1 1
/dev/vnd0c none swap sw 0 0
```

Votre ordinateur doit maintenant être redémarré afin que les changements effectués sur le noyau soient pris en compte. Une fois que cela est fait, il est temps de configurer le dispositif pour paginer. On utilisera la commande vnconfig(8).

```
$ sudo vnconfig -c -v vnd0 /var/swap
vnd0: 33554432 bytes on /var/swap
```

La dernière étape consiste à activer la pagination vers ce dispositif. Cela s'effectue de la même façon que dans les exemples précédents, en utilisant using swapctl(8). Puis nous vérifierons qu'il a bien été ajouté à la liste des dispositifs de pagination disponibles.

```
$ 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

Les "Soft Updates" sont basés sur une idée de Greg Ganger et Yale Patt et ont été développés pour FreeBSD par Kirk McKusick. Les Soft Updates imposent une réorganisation partielle des opérations sur le tampon permettant la suppression du code FFS, des écritures synchrones des entrées. Une augmentation des performances est ainsi réalisée lors des écritures sur disque.

L'activation des Soft Updates est effectuée par une option de montage. Lors du montage d'une partition avec l'utilitaire mount(8), vous pouvez activer les Soft Updates sur celle-ci. Voici l'exemple d'une entrée dans le fichier */etc/fstab(5)* permettant de monter la partition *sd0a* avec les Soft Updates activés.

```
/dev/sd0a / ffs rw,softdep 1 1
```

Note aux utilisateur sparc : n'activez pas Soft Updates sur les machines de type sun4 ou sun4c. Ces architectures ne supportent qu'un montant très faible de mémoire noyau et ne peuvent utiliser cette fonctionnalité. En revanche, les machines de type sun4m ne posent pas de problème.

14.6 - Comment se déroule le processus de démarrage d'OpenBSD/i386 ?

Le processus de démarrage d'un système OpenBSD/i386 est loin d'être simple et une bonne compréhension de celui-ci peut être utile afin de diagnostiquer un problème lorsque les choses ne fonctionnent pas. Ce processus comporte quatre étapes clefs :

1. ***Master Boot Record (MBR) :*** Le "Master Boot Record" (enregistrement de démarrage principal) est le premier secteur (512 bytes) du disque. Il contient la table de partition primaire et un petit programme permettant de charger le "Partition Boot Record" (PBR - enregistrement de démarrage des partitions). Notez que dans certains environnements, le terme "MBR" se réfère uniquement à la portion de code comprise dans le premier bloc du disque et non pas au bloc entier (incluant la table de partition). Il est primordial de comprendre le sens d'"initialiser le MBR" ; la terminologie sous OpenBSD sous-entend la réécriture complète du secteur MBR et pas seulement son code, comme cela peut être le cas avec certains systèmes. Vous n'en aurez l'utilité que très rarement. Pour ce faire, utilisez l'option "-u" de la commande fdisk(8) ("`fdisk -u wd0`").

   Bien qu'un MBR soit inclut avec OpenBSD, vous n'êtes pas obligé de l'utiliser puisque virtuellement, n'importe quel MBR est capable de le faire démarrer. Le MBR se manipule avec l'aide du programme fdisk(8) qui est utilisé pour éditer la table de

partition mais également pour installer le code MBR sur le disque.

Le MBR d'OpenBSD s'annonce avec le message suivant :

```
Using drive 0, partition 3.
```

montrant le disque ainsi que la partition d'où il s'apprête à charger le PBR. On peut également voir un point (".") en fin de ligne, ce qui signifie que la machine est capable d'utiliser la conversion LBA ("Logical Block Adressing") au démarrage. Si la machine avait été incapable d'une telle conversion, le point aurait été remplacé par un point-virgule (";"), indiquant une conversion CHS ("Cylinder-Head-Sector") :

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

Notez que le point ou le point-virgule en fin de ligne peut servir d'indicateur du "nouveau" MBR OpenBSD introduit à partir de la version 3.5.

2. ***Partition Boot Record (PBR) :*** Le "Partition Boot Record" (enregistrement de démarrage de partition), aussi appelé PBR ou biosboot(8) (d'après le nom du fichier contenant le code) représente le premier secteur physique de la partition OpenBSD du disque. Le PBR est le "first-stage boot loader" (chargeur de démarrage de niveau un) d'OpenBSD. Il est exécuté par le code MBR et se charge de lancer le "second- stage boot loader" (chargeur de démarrage de niveau deux), boot(8). Comme le MBR, le PBR est constitué d'une toute petite quantité de code et de données d'une taille totale de 512 bytes. Ce n'est pas suffisant pour avoir une application reconnaissant un système de fichiers, donc, plutôt que de demander au PBR de localiser `/boot` sur le disque, la localisation de `/boot` accessible par le BIOS est codée physiquement dans le PBR lors de l'installation.

   Le PBR est installé par installboot, décrit dans plus en détails dans la suite de ce document. Le PBR s'annonce avec le message suivant :

   ```
   Loading...
   ```

   affichant un point pour chaque bloc qu'il essaye de charger. Comme précédemment, le PBR montre s'il utilise une conversion LBA ou CHS pour s'exécuter. Dans le cas de CHS, il affichera un message suivi d'un point- virgule :

   ```
   Loading;...
   ```

   L'ancien biosboot(8) (avant la version 3.5) affichait le message "`reading boot...`".
3. ***Second Stage Boot Loader, `/boot` :*** `/boot` est chargé par le PBR et se charge d'accéder au système de fichiers OpenBSD par le BIOS de la machine afin de localiser et de lancer le noyau. boot(8) permet également de passer plusieurs options et informations au noyau.

   boot(8) est un programme intéractif. Après son exécution, il tente de localiser et lire `/etc/boot.conf` si celui-ci existe (ce qui n'est pas le cas dans une installation par défaut) puis lance les commandes qui y sont indiquées. A moins que le fichier `/etc/boot.conf` ne l'en empêche, boot(8) offre à l'utilisateur une invite de commandes similaire à celle-ci :

   ```
   probing: pc0 com0 com1 apm mem[636k 190M a20=on]
   disk: fd0 hd0+
   >> OpenBSD/i386 BOOT 2.10
   boot>
   ```

   Par défaut, l'utilisateur a cinq secondes pour entrer une commande avant le démarrage du noyau, `bsd`, à partir de la partition racine du premier disque dur, ce qui représente le comportement par défaut. Le "second- stage boot loader" détecte (et examine) votre matériel grâce au BIOS (puisque le noyau OpenBSD n'est pas encore chargé). Précédemment on peut voir qu'il a détecté :
   - **pc0** - Clavier et affichage vidéo d'un système i386 standard.
   - **com0, com1** - Deux ports série.

- **apm** - BIOS compatible APM ("Advanced Power Management" - gestion de courant évoluée).
- **636k 190M** - Le montant de mémoire conventionnelle (sous 1M) et étendue (au-dessus d'1M) trouvée.
- **fd0 hd0+** - Les disques accessibles par le BIOS, dans ce cas, un lecteur de disquettes et un disque dur.

Le signe '+' après "hd0" signifie que le BIOS a indiqué à `/boot` que ce disque peut être accessible par LBA. Lors d'une première installation, vous verrez souvent un '*' après un disque dur ; ceci signifie que ce disque ne possède pas de label OpenBSD valide.

4. ***Noyau : `/bsd`*:** il s'agit de la finalité du processus de démarrage, avoir le noyau OpenBSD chargé en mémoire RAM et fonctionnant correctement. Une fois le noyau exécuté, OpenBSD peut accéder directement au matériel sans passer par le BIOS.

Ainsi, le tout début du processus de démarrage pourrait ressembler à ceci :

```
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 2.10
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       <- Noyau
        The Regents of the University of California.  All rights reserved.
Copyright (c) 1995-2006 OpenBSD.  All rights reserved.  http://www.OpenBSD.org

OpenBSD 3.9 (GENERIC) #617: Thu Mar  2 02:26:48 MST 2006
   ...
```

Ce qui peut mal se passer

- **Bad/invalid/incompatible MBR :** habituellement, un disque déjà utilisé possède un code MBR, mais s'il est neuf ou déplacé à partir d'une plate-forme différente ET que vous ne répondez pas "Yes" à la question "Use entire disk" lors de l'installation, vous pourriez vous retrouver avec un disque sans MBR valide, le rendant ainsi non-bootable même si sa table de partition est correcte.

  Vous pouvez installer le MBR OpenBSD sur votre disque dur à l'aide du programme fdisk. Démarrez à l'aide du medium d'installation; choisissez "Shell" pour vous retrouver en ligne de commandes :

  ```
  # fdisk -u wd0
  ```

  Vous pouvez également installer un MBR particulier avec fdisk :

  ```
  # fdisk -u -f /usr/mdec/mbr wd0
  ```

  ce qui installera le fichier `/usr/mdec/mbr`en tant que MBR de votre système. Dans une installation standard d'OpenBSD, ce fichier représente le MBR par défaut codé dans fdisk, mais n'importe quel autre MBR pourrait ici être spécifié.
- **Invalid `/boot` location installed in PBR :** lorsqu'installboot(8) installe le PBR, il inscrit l'offset et le numéro du bloc de l'inode de `/boot` dans le celui-ci. Ainsi, supprimer puis remplacer `/boot` sans relancer installboot(8) empêchera votre système de redémarrer puisque le PBR chargera ce qui se trouve à l'ancien inode spécifié, ce qui ne sera certainement plus le "second-stage boot loader" désiré ! Puisque que `/boot` est accédé au travers d'appels BIOS, les anciennes versions de PBR étaient dépendantes de la conversion BIOS des disques. Si vous avez altéré la géométrie (à savoir, pris un disque d'un ordinateur utilisant la conversion CHS pour le mettre dans une machine utilisant LBA, ou simplement changé l'option de conversion dans votre BIOS), il *apparaîtra au niveau du BIOS* comme étant déplacé vers une autre location (un bloc numérique différent doit être accédé afin de récupérer les mêmes informations du disque) et vous devrez donc relancer installboot(8) avant que le système ne puisse être redémarré. Le nouveau PBR (OpenBSD 3.5 et plus) est beaucoup plus tolérant dans les changements de conversion.

Le PBR étant très petit, son éventail de messages d'erreur est limité et quelque peu complexe. La plupart des messages ressemblent à :

- **ERR R** - Le BIOS a retourné une erreur en tentant de lire un bloc à partir du disque. Habituellement, ceci est assez explicite : votre disque n'était pas lisible.
- **ERR M** - Un nombre magic(5) invalide a été lu dans l'entête du "second-stage bootloader". Ceci signifie généralement que ce qui a été lu n'était PAS `/boot`, à savoir qu'installboot(8) n'a pas été correctement exécuté, que le fichier /boot a été altéré ou encore que vous avez excédé la capacité de votre BIOS à lire les disques de grande capacité.

Les autres messages d'erreur sont expliqués en détail dans le manuel de biosboot(8). Pour plus d'informations sur le processus de démarrage pour i386, référez-vous à :

- boot_i386(8)
- http://www.ata-atapi.com/hiw.htm - Documents "How it Works" (comment cela fonctionne) de Hale Landis.

14.7 - Quels sont les problèmes liés aux disques de grande capacité sous OpenBSD ?

OpenBSD supporte les systèmes de fichiers jusqu'à 2^{31}-1, soit 2,147,483,647 secteurs, et puisque chaque secteur équivaut à 512 bytes, cela correspond à un peu moins de 1T.

Il existe aussi une taille limite de 1T pour le disque physique bien que sous *certaines* conditions, vous ne rencontrerez pas de problèmes avant 2T, mais ceci n'est **pas** garanti.

Bien entendu, les capacités du système de fichiers et celles d'un matériel défini sont deux choses bien différentes. Un nouveau disque dur IDE de 250G ne fonctionnera pas sur des interfaces trop anciennes (d'une capacité standard de 137G maximum), certains adaptateurs SCSI très anciens peuvent avoir des problèmes avec des lecteurs modernes et des BIOS un peu trop vieux peuvent se figer lorsqu'ils tombent sur un disque dur de trop grande capacité. De fait, vous devez respecter les possibilités de votre matériel.

Taille des partitions et limitations au niveau de la localisation

Malheureusement, les fonctionnalités du système d'exploitation ne sont accessibles qu'une fois celui-ci chargé en mémoire. Le processus de démarrage devra se contenter de la ROM de lancement du système (bien plus limitée).

Pour cette raison, le fichier /bsd (le noyau) doit se trouver dans l'espace disque adressable par la ROM. Cela signifie que sur certains systèmes i386, la partition racine doit se situer dans les premiers 504M, mais les ordinateurs plus récents peuvent avoir des limites de l'ordre de 2G, 8G, 32G, 128G ou plus. Il est important de noter que de nombreux ordinateurs récents supportant un adressage au delà de 128G sont en fait limités par le BIOS et ne peuvent démarrer que sur les 128 premiers G. Vous pouvez utiliser ces systèmes avec de gros disques durs, mais la partition racine devra être placée dans les premiers 128G.

Notez qu'il est toujours possible d'installer un disque dur de 40G sur un vieux 486 et d'y placer OpenBSD sur une seule et énorme partition ; vous pourriez penser avoir réussi à violer la règle précédente. Cependant, un beau jour, celle-ci risque de revenir vous hanter de manière peu plaisante :

- Vous installez tout sur une partition / de 40G. Cela fonctionne car le système de base et ses fichiers (/bsd inclut) se situent dans les premiers 504M.
- Vous utilisez le système et finissez par avoir pus de 504M de données dessus.
- Vous mettez à jour, compilez un nouveau noyau, au autre, et vous copiez votre nouveau /bsd sur l'ancien.
- Vous redémarrez.
- C'est alors que vous recevez un message tel que "ERR M" ou rencontrez des problèmes lors du démarrage.

Pourquoi ? Simplement parce que vous avez "écrasé" l'ancien /bsd par le nouveau, ce fichier n'a pas été écrit au-dessus de l'autre, mais a été relocalisé vers un autre endroit sur le disque, probablement au-delà des 504 premiers M supportés par le bios. Le chargeur de démarrage se voit alors dans l'impossibilité d'atteindre /bsd et le système se fige.

Afin de permettre le lancement d'OpenBSD, les chargeurs de démarrage (biosboot(8) et `/boot` dans le cas d'un i386) ainsi que le noyau (`/bsd`) doivent se situer à l'intérieur de l'espace d'adressage supporté par la ROM de boot et par leurs propres capacités. Afin de ne jamais rencontrer de problèmes, la règle est simple :

la partition racine dans son ensemble doit se situer dans l'espace d'adressage supporté par le BIOS (ou la ROM de démarrage).

Certains utilisateurs n'utilisant pas d'i386 peuvent penser qu'ils sont immunisés contre ce problème, cependant, de nombreuses plates-formes possèdent une certaine limite fixée par la ROM de démarrage concernant la taille du disque. Trouver cette limite peut être compliquée.

Et c'est une autre raison pour partitionner votre disque dur, plutôt que d'utiliser une seule grosse partition.

fsck(8) : impératifs de durée et de mémoire

Les autres points à prendre en considération avec les gros systèmes de fichiers sont le temps et la mémoire nécessaires à l'utilisation de fsck(8) après un crash ou à une coupure de courant. On ne peut pas créer un système de fichiers de 120G sur un disque et espérer utiliser fsck(1) avec succès après un crash. Une règle simple à appliquer est de considérer que le système doit posséder au moins 1M de mémoire libre pour chaque 1G d'espace disque afin d'utiliser fsck sans problème. Le temps nécessaire à l'utilisation de fsck peut devenir un problème au fur et à mesure que le système de fichiers grossit.

14.8 - Installation des blocs de démarrage ("Bootblocks") - spécifique i386

Les anciennes versions de MS-DOS n'étaient capables d'interagir qu'avec les disques dont la géométrie était de 1024 cylindres ou moins. Puisque de nos jours tous les disques ont plus de 1024 cylindres, la plupart des BIOS SCSI (qui sont présents sur les cartes contrôleur SCSI) et IDE (qui sont inclus avec le reste du BIOS du PC) possèdent une option qui "traduit" la véritable géométrie du disque en une géométrie compatible 'MS-DOS'. Cependant, tous les BIOS ne "traduisent" pas cette géométrie de la même manière. Si vous changez votre BIOS (en installant une nouvelle carte mère ou un nouveau contrôleur SCSI) et que le nouveau utilise une "traduction" de géométrie différente, vous ne pourrez pas lancer le chargeur de démarrage de niveau 2 ("second-stage boot loader") et donc le noyau. Cela est du au fait que le chargeur de démarrage de niveau 1 ("first-stage boot loader") contient la liste des blocs comprenant /boot formatée selon l'ancienne géométrie "traduite". Si vous utilisez des disques IDE et effectuez des changements dans la configuration de votre BIOS, vous pouvez (sans le savoir) changer aussi la traduction de la géométrie (la plupart des BIOS proposent 3 types de traductions différentes).Pour réparer votre bloc de démarrage afin que vous puissiez booter normalement, insérez une disquette dans votre lecteur (ou utilisez un CD-ROM bootable) et à l'invite de commandes, tapez "b hd0a:/bsd" afin de le forcer à démarrer sur le premier disque (et non sur la disquette). Votre machine devrait se lancer normalement. Vous devez à présent mettre à jour le chargeur de premier niveau pour qu'il voit la nouvelle géométrie (et récrive le bloc de démarrage en conséquence).
Notre exemple assume que votre disque de démarrage est sd0 (mais pour de l'IDE, ce pourrait être wd0, etc) :

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

Si une nouvelle version des blocs de démarrage est nécessaire, vous devrez la compiler vous même. Pour ce faire, tapez simplement :

```
# cd /sys/arch/i386/stand/
# make && make install
# cd /usr/mdec; cp ./boot /boot
# ./installboot /boot biosboot sd0 (ou quelque soit le périphérique
désignant votre disque dur)
```

14.9 - Se préparer au désastre : faire une sauvegarde vers une bande et effectuer une restauration

Introduction :

Si vous envisagez d'administrer ce que l'on pourrait appeler un serveur de production, il est préférable d'avoir une sauvegarde au cas où l'un de vos disques durs tomberait en panne.

Cette documentation vous assistera dans l'utilisation des utilitaires standards dump(8)/ restore(8) fournis avec OpenBSD. Un autre utilitaire de sauvegarde plus avancé nommé "Amanda" est également disponible via les ports afin de sauvegarder plusieurs serveurs vers un unique lecteur de bandes. Dans la plupart des environnements, dump(8) / restore(8) suffit. Cependant, si vous avez besoin de sauvegarder plusieurs machines, il peut être utile de se documenter sur Amanda.

Les périphériques utilisés en exemple dans ce document se réfèrent à une configuration utilisant des disques et lecteurs de bandes SCSI. En environnement de production, les disques SCSI sont recommandés en place des disques IDE pour la façon dont ils gèrent les blocs défectueux. Ceci ne signifie pas pour autant que ce document est sans intérêt pour les possesseurs de disques IDE ou d'autres types de lecteurs de bandes, les noms de vos périphériques changeront légèrement. Par exemple, sd0a deviendra wd0a dans un système à base d'IDE.

Sauvegarder vers une bande :

Sauvegarder sur bande demande de savoir exactement où vos systèmes de fichiers sont montés. Vous pouvez le déterminer en utilisant la commande mount(8) dans un terminal. Vous devriez obtenir une sortie similaire à celle-ci :

```
# mount
/dev/sd0a on / type ffs (local)
/dev/sd0h on /usr type ffs (local)
```

Dans cet exemple, le système de fichiers racine (/) réside physiquement sur sd0a, ce qui désigne la partition a du disque SCSI fixe 0. Le système de fichiers /usr réside sur sd0h, ma partition h du disque SCSI fixe 0.

Un autre exemple de ce qu'une table de montage plus avancée pourrait être :

```
# 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)
```

Dans cet exemple plus compliqué, le système de fichiers racine (/) réside physiquement sur sd0a ; /var sur sd0d; /home sur sd0e et enfin /usr sur sd0h.

Afin de sauvegarder votre machine, vous aurez besoin de renseigner dump avec le nom exact de chaque partition fixe. Voici un exemple des commandes nécessaires pour sauvegarder la table de montage simple vue plus haut :

```
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0a
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0h
# mt -f /dev/rst0 rewind
```

Pour la table de montage plus avancée, vous utiliserez des commandes similaires à :

```
# /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
```

Vous pouvez vous référer à la page de manuel de dump(8) afin de connaître les fonctionnalités de chaque option. Voici une brève description des paramètres utilisés :

- **0** - Effectue un dump de niveau 0, sauvegarde tout.
- **a** - Tente de détecter automatiquement la taille de la bande.
- **u** - Met à jour le fichier /etc/dumpdates afin de savoir quand la dernière sauvegarde a été effectuée.
- **f** - Lecteur de bande à utiliser (ici, /dev/nrst0).

Et finalement, la partition à sauvegarder (/dev/rsd0a, etc).

La commande mt(1) est utilisée à la fin pour rembobiner la bande. Référez-vous au manuel de mt afin de connaître les options disponibles (comme eject).

Si vous n'êtes pas sûr du nom de périphérique du lecteur, utilisez dmesg pour le localiser. Un lecteur de bande peut, par exemple, apparaître ainsi dans dmesg :

```
st0 at scsibus0 targ 5 lun 0: <ARCHIVE, Python 28388-XXX, 5.28>
```

Vous avez peut-être noté que, lors d'une sauvegarde, le lecteur est accédé par son nom de périphérique "`nrst0`" au lieu de "`st0`" qui apparaît dans dmesg. Lorsque vous accédez `st0` à la place de `nrst0`, vous accédez au même lecteur bande mais en mode "raw" et en lui indiquant de ne pas rembobiner la bande à la fin du travail. Pour sauvegarder plusieurs systèmes de fichiers sur une même bande, soyez certains d'utiliser le périphérique de non-rembobinage ; dans le cas contraire (si vous utilisez `rst0`), la sauvegarde du système de fichiers précédent sera écrasée à la prochaine écriture sur la bande. Vous pouvez trouver une description plus conséquente de plusieurs lecteurs de bande dans la page de manuel de dump.

Si vous souhaitez écrire un petit script appelé "backup", il pourrait ressembler à celui-ci :

```
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
```

Si des sauvegardes nocturnes planifiées sont nécessaires, cron(8) pourra être utilisé pour lancer votre utilitaire de sauvegarde automatiquement.

Il serait également utile de préciser (sur un bout de papier) la taille que devrait avoir chaque système de fichiers. Vous pouvez utiliser "`df -h`" afin de connaître l'espace actuellement utilisé par chaque partition. En effet, ceci pourra vous aider lorsque votre disque dur tombera en panne et que vous devrez recréer la table de partition sur le nouveau disque.

Restaurer vos données aidera aussi à réduire la fragmentation. Afin d'être sûr de récupérer tous les fichiers, le meilleur moyen de sauvegarder est de redémarrer votre système en mode "single user". Les systèmes de fichiers n'ont pas besoin d'être montés pour être sauvegardés. N'oubliez pas de monter la racine (/) en mode r/w (lecture/écriture) après avoir redémarré en mode "single user" ou dump retournera une erreur lorsqu'il essayera d'écrire les "dumpdates". Au démarrage, entrez "`bsd -s`" à l'invite de commandes boot> pour

lancer le système en mode "single user".

Voir le contenu d'une sauvegarde sur bande :

Après avoir sauvegardé votre système de fichiers pour la première fois, il est bon de vérifier rapidement votre bande afin d'être certain que les données sont bien présentes.

Vous pouvez utiliser l'exemple suivant afin de voir le catalogue de fichiers présents sur une bande :

```
# /sbin/restore -tvs 1 -f /dev/rst0
```

Cette commande affichera la liste des fichiers présents sur la première partition de la bande. Si l'on se réfère à l'exemple précédent, 1 serait votre système de fichiers racine (/).

Pour afficher se qui est présent sur la deuxième partition vers un fichier, vous utiliserez une commande similaire à :

```
# /sbin/restore -tvs 2 -f /dev/rst0 > /home/me/list.txt
```

Pour une table de partition assez simple, 2 serait /usr ; si la vôtre est plus compliquée, 2 pourrait être /var/ ou un autre système de fichiers. L'ordre des séquences est similaire à celui utilisé pour sauvegarder les systèmes de fichiers sur la bande.

Restaurer à partir d'une bande :

Le scénario de l'exemple suivant pourrait être utile si votre disque tombait complètement en panne. Dans le cas ou vous ne souhaiteriez récupérer qu'un seul fichier à partir de la bande, référez-vous à la page de manuel de la commande restore et soyez attentifs aux instructions concernant le mode interactif.

Si vous avez bien tout préparé, remplacer un disque et restaurer vos données à partir d'une bande peut être une procédure très rapide. La disquette standard install/boot sous OpenBSD contient déjà l'utilitaire restore ainsi que les exécutables nécessaires au partitionnement de votre nouveau disque et permettant de le rendre bootable. Dans la plupart des cas, cette disquette et votre bande de sauvegarde la plus récente sont tout ce dont vous avez besoin pour remettre votre machine en production.

Après avoir remplacé le disque en panne, suivez les étapes suivantes afin de restaurer vos données :

- Démarrez à partir de la disquette install/boot d'OpenBSD. Au menu, choisissez Shell. Insérez la bande de votre sauvegarde la plus récente bande, en lecture seule, dans votre lecteur.
- En vous aidant de la commande fdisk(8), créez une partition primaire OpenBSD sur le nouveau disque. Par exemple :

  ```
  # fdisk -e sd0
  ```

  Référez-vous à l'Utilisation de fdisk(8) sous OpenBSD pour de plus amples informations.
- Avec la commande disklabel, recréez votre table de partition OpenBSD à l'intérieur de la partition primaire que vous venez de créer avec fdisk. Par exemple :

  ```
  # disklabel -E sd0
  ```

  (N'oubliez pas la partition d'échange, swap, référez-vous à Utilisation de disklabel pour plus d'informations)
- Utilisez la commande newfs pour créer un système de fichier vierge sur chaque partition que vous venez de créer. Par exemple :

```
# newfs /dev/rsd0a
# newfs /dev/rsd0h
```

- Monter votre nouvelle partition racine (/) sur /mnt. Par exemple :

```
# mount /dev/sd0a /mnt
```

- Allez dans le nouveau répertoire racine et démarrez la procédure de restauration. Par exemple :

```
# cd /mnt
# restore -rs 1 -f /dev/rst0
```

- Vous aurez besoin de rendre ce disque bootable, inspirez-vous la commande suivante pour y inscrire un nouveau MBR. Par exemple :

```
# fdisk -i sd0
```

- En plus d'inscrire un nouveau MBR, vous aurez besoin d'installer les blocs de démarrage afin de pouvoir lancer la machine. Voici un bref exemple :

```
# cp /usr/mdec/boot /mnt/boot
# /usr/mdec/installboot -v /mnt/boot /usr/mdec/biosboot sd0
```

- Votre nouveau système de fichiers racine devrait être suffisamment prêt pour que vous puissiez démarrer dessus et continuer de restaurer le reste de vos systèmes de fichiers. Puisque votre système d'exploitation n'est pas encore complet, soyez sûrs de redémarrer en mode single-user. A l'invite de commandes, effectuez la procédure suivante afin de démonter et redémarrer le système :

```
# umount /mnt
# halt
```

- Retirez la disquette install/boot du lecteur et redémarrez votre machine. A l'invite de commandes de boot d'OpenBSD, lancez les commandes suivantes :

```
boot> bsd -s
```

  La commande "bsd -s" ordonnera au noyau de démarrer en mode single-user qui ne nécessite que le système de fichiers racine (/).
- En admettant que vous ayez effectué correctement les étapes précédentes et que tout se soit bien passé, vous devriez vous retrouver à une invite de commandes vous demandant le chemin d'accès à un shell ou d'appuyer sur entrée. Appuyez sur entrée pour utiliser sh. Ensuite, vous devrez remonter la racine en mode r/w plutôt qu'en lecture seule. Lancez la commande suivante :

```
# mount -u -w /
```

- Une fois la racine en mode r/w, vous pourrez continuer à restaurer vos autres systèmes de fichiers. Par exemple :

```
(table de montage simple)
# mount /dev/sd0h /usr; cd /usr; restore -rs 2 -f /dev/rst0

(table de montage avancée)
# 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
```

A la place, vous pourriez utiliser "**restore rvsf**" ou juste rsf pour voir le nom des fichiers extraits de la sauvegarde.

- Pour terminer, après avoir restauré tous vos systèmes de fichiers sur le disque, redémarrez en mode multi-utilisateurs. Si tout s'est déroulé comme prévu, votre système devrait se retrouver dans l'état dans lequel il se trouvait à la date de votre sauvegarde la plus récente et être totalement fonctionnel.

14.10 - Montage des images disque sous OpenBSD

Pour monter une image disque (images ISO, images disques créées avec dd, etc) sous OpenBSD, vous devez configurer un périphérique vnd(4) . Par exemple, si vous avez une image ISO localisée dans */tmp/ISO.image* , voici les étapes nécessaires pour la monter :

```
# vnconfig svnd0 /tmp/ISO.image
# mount -t cd9660 /dev/svnd0c /mnt
```

Remarquez que, puisqu'il s'agit d'une image ISO-9660 utilisée sur les CDs et DVDs, vous devez spécifier le type *cd9660* lors du montage. Ceci est vrai peut importe le type, par exemple, vous devez utiliser *ext2fs* lors du montage d'images disque Linux.

Pour démonter une image, utilisez les commandes suivantes.

```
# umount /mnt
# vnconfig -u svnd0
```

Pour plus d'information, référez-vous à la page de manuel de vnconfig(8).

14.11 - A l'aide ! J'ai des erreurs avec IDE DMA !

Les transferts IDE DMA supportés par pciide(4) ne sont pas fiables avec plusieurs combinaisons de matériel. Jusqu'à récemment, la plupart des "principaux" systèmes d'exploitation prétendant supporter les transferts DMA sur périphériques IDE n'activaient pas cette fonctionnalité par défaut à cause de certains matériels peu fiables. A présent, beaucoup de ces machines sont utilisées sous OpenBSD.

OpenBSD est agressif et tente d'utiliser le mode DMA le plus haut qu'il puisse configurer. Ceci pourrait engendrer une corruption dans le transfert des données sur certaines configurations à cause de certaines puces ("chipsets") de cartes mères, de disques de mauvaise qualité et/ou de la friture sur les câbles. Heureusement, les modes Ultra-DMA protègent les transferts de données avec CRC afin de détecter une éventuelle corruption. Si le CRC de l'Ultra-DMA échoue, OpenBSD affichera un message d'erreur et recommencera l'opération de transfert.

```
wd2a:  aborted command, interface CRC error reading fsbn 64 of 64-79
(wd2 bn 127; cn 0 tn 2 sn 1), retrying
```

Après plusieurs échecs, OpenBSD fera descendre le mode Ultra-DMA (à priori plus fiable). Lorsqu'un mode Ultra-DMA atteint 0, le mode PIO prendra le relais.

Les erreurs UDMA sont souvent causées par des câbles endommagés ou de mauvaise qualité. C'est pourquoi les câbles devraient être les premiers suspectés si vous avez de nombreuses erreurs de DMA ou des performances dégradées. De plus, il n'est pas conseillé de placer un lecteur de CD-ROM sur la même nappe qu'un disque dur.

Si le remplacement des câbles ne résout pas le problème et qu'OpenBSD n'arrive pas à faire descendre le niveau UDMA, que votre machine se fige ou que vous obtenez trop de messages d'erreur sur la console et dans les logs, vous pouvez forcer le système à utiliser un mode DMA ou UDMA inférieur par défaut. Ceci peut être effectué avec UKC ou config(8) afin de changer les options du

périphérique wd(4).

14.13 - Options RAID avec OpenBSD

Un ensemble redondant de disques durs indépendants (RAID - "Redundant Array of Inexpensive Disks") vous offre la possibilité d'utiliser plusieurs disques afin d'améliorer les performances, la capacité et/ou la redondance qu'un disque seul ne serait pas capable d'apporter. Bien qu'une description des avantages ou des risques apportés par une configuration RAID ne soit pas le propos de cet article, il est important de noter les points suivants :

- Un RAID ne constitue pas un système de sauvegarde.
- Seul, un RAID n'éliminera pas le délais d'indisponibilité.

Si ces informations sont nouvelles pour vous, ce document n'est pas un bon point de départ pour débuter avec un RAID.

Options logicielles

OpenBSD inclut RAIDframe, une solution logicielle de RAID. La documentation est disponible ici :

- Optimisation des disques, RAID
- Page d'accueil de RAIDframe
- manuel de raidctl(8)
- manuel de raid(4)

La partition racine peut être directement configurée en miroir par OpenBSD en utilisant l'option "Autoconfiguration" de RAIDframe.

A partir d'OpenBSD 3.7-stable, le pilote ccd(4) inclut également une fonction de "mirroring". Ce système est inclu dans le noyau GENERIC et est présent dans le noyau bsd.rd sur certaines plates-formes (amd64, hppa, hppa64, i386), ce qui facilite son utilisation même s'il possède certaines limitations dans la reconstruction de l'array. Référez-vous aux manuels :

- ccd(4)
- ccdconfig(8)

Options matérielles

De nombreuses plates-formes OpenBSD supportent plusieurs contrôleurs RAID matériels. Ceux-ci varient selon les plates-formes, référez-vous à la page des plates-formes supportées (listées ici).

Une autre option accessible à plusieurs plates-formes est l'un des nombreux produits qui font apparaître plusieurs disques comme un seul disque IDE ou SCSI et sont ensuite branchés sur un contrôleur IDE ou SCSI standard. Ces périphériques peuvent virtuellement fonctionner sur n'importe quelle plateforme supportant IDE ou SCSI.

Quelques fabriquants de ce type de produits :

- Arco
- Accusys
- Maxtronic
- Infortrend

(Note : ce sont des produits que certains utilisateurs d'OpenBSD utilisent ; il ne s'agit en aucun cas d'une liste exhaustive ni de matériels

certifiés).

Options non valides

Une question récurrente sur les listes de diffusion consiste à savoir si "les contrôleurs RAID IDE ou SATA peu coûteux (tels que les contrôleurs HighPoint, Promise ou Adaptec HostRAID) sont supportés ?". La réponse est "Non". Ces cartes et puces ne sont pas de véritables contrôleurs RAID matériels mais plutôt des RAID logiciels assistés par un BIOS. Puisqu'OpenBSD supporte déjà le RAID logiciel indépendamment du type de matériel, il n'y a pas de volonté parmi les développeurs OpenBSD d'implémenter un support spécifique à ces cartes.

Presque tous ces contrôleurs "RAID" SATA ou IDE embarqués sont de type logiciel et fonctionneront sans problème comme contrôleur SATA ou IDE en utilisant le pilote IDE standard (pciide(4)), mais ne fonctionneront pas comme RAID matériel sous OpenBSD.

14.14 - Pourquoi `df(1)` me dit que j'ai plus de 100% d'espace disque utilisé ?

Certaines personnes sont parfois surprises de voir qu'elles ont un espace disque disponible *négatif* ou que plus de 100% de l'espace d'un système de fichiers est utilisé, comme on peut le voir avec df(1).

Lorsqu'un système de fichiers est créé avec newfs(8) , un certain montant d'espace disque inaccessible aux utilisateurs est réservé. Cela permet d'avoir une marge d'erreur en cas de remplissage accidentel du disque et de garder la fragmentation au minimum. Par défaut, l'espace réservé correspond à 5% de la capacité du disque ce qui signifie que si root a imprudemment rempli le disque, vous pouvez vous retrouver avec 105% de capacité disponible utilisée.

Si une valeur de 5% ne vous semble pas appropriée, vous pouvez la changer avec la commande tunefs(8).

14.15 - Récupération de partitions après une suppression du disklabel

Si vous avez une table de partitions endommagée, vous disposez de plusieurs moyens afin d'essayer de la récupérer.

Premièrement, paniquez. Vous le faites relativement souvent, et ceci devrait donc bien se passer. Cependant, ne faites rien de stupide. Paniquez à l'écart de votre machine. Ensuite, relaxez vous, et voyez si les étapes suivantes ne pourraient pas vous aider.

Une copie de votre disklabel est sauvegardée pour chaque disque dans `/var/backups` via la maintenance système journalière. A condition d'avoir encore votre partition var, vous pouvez simplement lire la sortie, et la réintégrer dans disklabel.

Dans le cas ou vous ne pouvez plus voir cette partition, il y a deux options. Corriger une assez grande partie du disque afin de la voir, ou fixer une partie suffisamment grande du disque afin de pouvoir récupérer vos données. Selon ce qui s'est passé, l'une ou l'autre de ces deux solutions serait préférable (avec des disques mourants vous voudrez en premier les données, avec des doigts mouillés vous ne pourrez avoir que le label).

Le premier bon outil dont vous avez besoin est scan_ffs(8) (remarquez l'underscore, il ne s'appelle pas "scanffs"). scan_ffs(8) regardera au travers du disque, et essaiera de trouver les partitions, en vous avertissant des informations qu'il trouve sur celles-ci). Vous pouvez utiliser ces informations pour recréer le disklabel. Si vous voulez uniquement restaurer `/var`, vous pouvez recréer la partition pour `/var`, et ensuite récupérer le label et ajouter le reste.

disklabel(8) mettra à jour à la fois la compréhension du disklabel par le noyau, et essaiera également d'écrire le label sur le disque.

Ainsi, même si la partie du disque qui contient le disklabel est illisible, vous serez en mesure de la monter avec mount(8) jusqu'au prochain redémarrage.

14.16 - Est-il possible d'accéder aux données présentes sur des systèmes de fichiers autres que FFS ?

Oui. Les systèmes de fichiers supportés sont : ext2 (Linux), ISO9660 et UDF (CD-ROM, DVD), FAT (MS-DOS et Windows), NFS, NTFS (Windows), AmigaDOS. Certains n'ont qu'un support limité, comme par exemple l'accès en lecture uniquement. Notez que le système de fichiers UFS2 de FreeBSD n'est pas supporté.

Nous allons donner un aperçu sur la façon d'utiliser un de ces systèmes de fichiers sous OpenBSD. Afin d'utiliser un système de fichiers, celui-ci doit être monté. Pour plus de détails sur les options de montage, référez-vous au manuel de mount(8) ainsi qu'à celui correspondant au système de fichiers que vous souhaitez monter. Par exemple, mount_msdos, mount_ext2fs, ...

Tout d'abord, vous devez connaître le périphérique sur lequel votre système de fichiers est présent. Cela peut être tout simplement votre disque dur, `wd0` ou `sd0`, mais ça n'est pas toujours évident. Tous les périphériques reconnus et configurés par votre système apparaissent dans la sortie de la commande dmesg(1) : un nom de périphérique, suivi par une ligne descriptive. Ainsi, mon premier CD-ROM est reconnu de la façon suivante :

```
cd0 at scsibus0 targ 0 lun 0: <COMPAQ, DVD-ROM LTD163, GQH3> SCSI0 5/cdrom removable
```

Afin d'obtenir une liste beaucoup plus courte des disques disponibles, vous pouvez utiliser sysctl(8). La commande

```
# sysctl hw.disknames
```

affichera tous les disques reconnus par votre système, par exemple :

```
hw.disknames=cd0,cd1,wd0,fd0,cd2
```

A présent, il est temps de déterminer quelles partitions sont présentes sur le périphérique et sur quelle partition le système de fichiers qui nous intéresse est présent. Nous examinerons le périphérique en utilisant disklabel(8). Le 'disklabel' comporte une liste de partitions, 16 au maximum. La partition c désigne toujours le périphérique dans son ensemble. Les partitions a-b et d-p sont utilisées par OpenBSD. Les partitions i-p peuvent être allouées automatiquement aux systèmes de fichiers d'autres systèmes d'exploitation. Dans notre cas, j'analyserai le 'disklabel' de mon disque dur qui contient un certain nombre de systèmes de fichiers.

REMARQUE : OpenBSD a été installé après les autres systèmes d'exploitation sur cette machine et pendant l'installation, le disklabel contenant les partitions pour systèmes de fichiers natifs et étrangers a été installé. Cependant, si vous ajoutez des systèmes de fichiers étrangers après l'installation du disklabel d'OpenBSD vous devrez probablement les ajouter ou les modifier ultérieurement. Cette procédure est expliquée ici.

```
# disklabel wd0

# using MBR partition 2: type A6 off 20338290 (0x1365672) size 29318625 (0x1bf5de1)
# /dev/rwd0c:
type: ESDI
disk: ESDI/IDE disk
label: ST340016A
flags:
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
```

```
total sectors: 78165360
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # microseconds
track-to-track seek: 0  # microseconds
drivedata: 0

16 partitions:
#             size        offset  fstype [fsize bsize  cpg]
  a:        408366      20338290  4.2BSD   2048 16384   16 # Cyl 20176*- 20581
  b:       1638000      20746656    swap                    # Cyl 20582 - 22206
  c:      78165360             0  unused      0     0      # Cyl     0 - 77544
  d:       4194288      22384656  4.2BSD   2048 16384   16 # Cyl 22207 - 26367
  e:        409248      26578944  4.2BSD   2048 16384   16 # Cyl 26368 - 26773
  f:      10486224      26988192  4.2BSD   2048 16384   16 # Cyl 26774 - 37176
  g:      12182499      37474416  4.2BSD   2048 16384   16 # Cyl 37177 - 49262*
  i:         64197            63 unknown                    # Cyl     0*-    63*
  j:      20274030         64260 unknown                    # Cyl    63*- 20176*
  k:       1975932      49656978   MSDOS                    # Cyl 49262*- 51223*
  l:       3919797      51632973 unknown                    # Cyl 51223*- 55111*
  m:       2939832      55552833  ext2fs                    # Cyl 55111*- 58028*
  n:       5879727      58492728  ext2fs                    # Cyl 58028*- 63861*
  o:      13783707      64372518  ext2fs                    # Cyl 63861*- 77535*
```

Comme on peut le constater, les partitions OpenBSD sont affichées en premier. Puis l'on peut voir un certain nombre de partitions ext2, une partition MSDOS ainsi que plusieurs partitions inconnues ('unknown'). Sous i386 ou amd64, vous pouvez généralement en savoir plus sur celles-ci en utilisant l'utilitaire fdisk(8). Pour le lecteur curieux, la partition i est une partition de maintenance créée par le revendeur, la partition j est une partition NTFS et et la partition l est une partition d'échange Linux ('Linux swap').

Après avoir déterminée la partition que vous souhaitez utiliser, vous pouvez enfin monter le système de fichiers qu'elle contient. La plupart des systèmes de fichiers sont supportés par de noyau GENERIC : référez-vous à son fichier de configuration situé dans le répertoire `/usr/src/sys/arch/<arch>/conf`. Cependant, certains dont le support est expérimental (comme NTFS) ne sont pas inclus dans le noyau GENERIC. Si vous souhaitez utiliser un de ces systèmes de fichiers, vous devrez recompiler votre noyau .

Une fois les informations nécessaires entre vos mains, il est temps de monter le système de fichiers. Imaginons que le répertoire `/mnt/otherfs` existe, nous l'utiliserons pour monter le système de fichiers souhaité. Dans l'exemple suivant, nous monterons un système de fichiers ext2 dans la partition m :

```
# mount -t ext2fs /dev/wd0m /mnt/otherfs
```

Si vous souhaitez utiliser ce système de fichiers régulièrement, vous gagnerez du temps en insérant la ligne suivante dans votre fichier `/etc/fstab` :

```
/dev/wd0m /mnt/otherfs ext2fs rw,noauto,nodev,nosuid 0 0
```

Notez les valeurs 0 dans les cinquièmes et sixièmes champs. Cela indique que nous ne souhaitons pas que le système de fichiers soit sauvegardé (avec dump) ni vérifié avec fsck. Généralement, il est préférable de laisser le système d'exploitation associé à ces systèmes de fichiers s'occuper de cela.

14.16.1 - Les partitions n'apparaissent pas dans mon disklabel ! Que dois-je faire ?

Si vous installez des systèmes de fichiers étrangers sur votre machine (souvent suite à l'ajout d'un système d'exploitation) après l'installation d'OpenBSD, un disklabel sera présent mais ne sera pas mis à jour automatiquement pour prendre en compte les nouveaux systèmes de fichiers. Si vous souhaitez pouvoir y accéder, vous aurez besoin d'ajouter ou de modifier ces partitions manuellement en

utilisant disklabel(8).

Pour prendre un exemple, j'ai modifié une de mes partitions ext2 existantes : en utilisant le programme fdisk de Linux, j'ai réduit la taille de la partition 'o' (voir la sortie de disklabel plus haut) à 1G. Il sera facile de la reconnaître grâce à sa position de départ (offset: 64372518) et sa taille (13783707) sur le disque. Notez que ces valeurs sont exprimées en numéros de secteurs (et pas en megabytes ou autre mesure) et qu'utiliser de tels nombres reste le moyen le plus sûr et le plus précis d'obtenir ces informations.

Avant notre changement, la partition ressemblait à ceci (en utilisant le fdisk(8) d'OpenBSD et en ne gardant que ce qui nous intéresse) :

```
# fdisk wd0
. . .
Offset: 64372455        Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
 0: 83 4007   1  1 - 4864 254 63 [    64372518:    13783707 ] Linux files*
. . .
```

Comme vous pouvez le voir, la position de départ et la taille sont exactement les mêmes que celles rapportées précédemment par disklabel(8) (ne vous méprenez pas sur par la valeur indiquée par "Offset" : il s'agit de la position de départ de la partition étendue à laquelle la partition ext2 fait partie).

Après avoir changé la taille de la partition sous Linux, elle ressemble à ceci :

```
# fdisk wd0
. . .
Offset: 64372455        Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
 0: 83 4007   1  1 - 4137 254 63 [    64372518:     2104452 ] Linux files*
. . .
```

Ceci doit être changé par disklabel(8). Vous pouvez, par exemple, utiliser `disklabel -e wd0` qui lancera l'éditeur correspondant à la variable d'environnement EDITOR (par défaut il s'agit de vi). Une fois dans l'éditeur, changez la dernière ligne du disklabel afin qu'elle corresponde à la nouvelle taille :

```
  o:       2104452      64372518  ext2fs
```

Pour terminer, enregistrez le disklabel sur le disque. A présent que le disklabel est à jour, vous devriez pouvoir monter vos partitions comme indiqué précédemment.

Vous pouvez suivre une procédure similaire afin d'ajouter de nouvelles partitions.

14.17 - Est-il possible d'utiliser un périphérique de masse ('flash memory device') sous OpenBSD ?

Théoriquement, un périphérique mémoire devrait être reconnu dès sont insertion dans la machine. Peut après l'avoir branché, un certain nombre de messages du noyau devrait apparaître sur la console. Ainsi, lorsque je branche mon périphérique USB, voici ce qui apparaît sur la console :

```
umass0 at uhub1 port 1 configuration 1 interface 0
umass0: LEXR PLUG DRIVE LEXR PLUG DRIVE, rev 1.10/0.01, addr 2
```

```
umass0: using SCSI over Bulk-Only
scsibus2 at umass0: 2 targets
sd0 at scsibus2 targ 1 lun 0: <LEXAR, DIGITAL FILM, /W1.> SCSI2 0/direct removable
sd0: 123MB, 123 cyl, 64 head, 32 sec, 512 bytes/sec, 251904 sec total
```

Ces lignes indiquent que le pilote umass(4) (périphérique de masse USB -'USB mass storage') à été rattaché au périphérique mémoire et qu'il utilise le système SCSI. Les deux dernières lignes sont les plus importantes : elles indiquent sur quel fichier de périphérique le matériel a été rattaché ainsi que le montant total d'espace de stockage. Si vous avez manqué ces lignes, vous pouvez vous y référez plus tard à l'aide de la commande dmesg(1). La géométrie CHS est fictive puisque le périphérique USB est traité comme un disque SCSI classique.

Nous allons voir deux scénarios possibles.

Le périphérique est nouveau/vide et vous souhaitez l'utiliser exclusivement avec OpenBSD.

Vous allez devoir initialiser un 'disklabel' sur le périphérique et créer au minimum une partition. Pour plus de détails, lisez Utilisation de disklabel(8) sous OpenBSD ainsi que sa page de manuel.

Dans cet exemple, je ne vais créer qu'une seule partition *a* dans laquelle je placerai un système de fichiers FFS :

```
# newfs sd0a
Warning: inode blocks/cyl group (125) >= data blocks (62) in last
    cylinder group. This implies 1984 sector(s) cannot be allocated.
/dev/rsd0a:     249856 sectors in 122 cylinders of 64 tracks, 32 sectors
        122.0MB in 1 cyl groups (122 c/g, 122.00MB/g, 15488 i/g)
super-block backups (for fsck -b #) at:
 32,
```

Montons le système de fichiers créé dans la partition *a* sur `/mnt/flashmem`. Si le point de montage n'existe pas, créez-le.

```
# mkdir /mnt/flashmem
# mount /dev/sd0a /mnt/flashmem
```

Vous avez reçu le périphérique de quelqu'un avec qui vous souhaitez échanger des données.

Il y a une grande chance que la personne en question n'utilise pas OpenBSD, ce qui signifie qu'un système de fichiers étranger doit être présent sur le périphérique. Nous allons donc devoir découvrir quelles partitions sont présentes, comme décrit dans la section précédente : Est-il possible d'accéder aux données présentes sur des systèmes de fichiers autres que FFS ?.

```
# disklabel sd0

# /dev/rsd0c:
type: SCSI
disk: SCSI disk
label: DIGITAL FILM
flags:
bytes/sector: 512
sectors/track: 32
tracks/cylinder: 64
sectors/cylinder: 2048
cylinders: 123
total sectors: 251904
```

```
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # microseconds
track-to-track seek: 0  # microseconds
drivedata: 0

16 partitions:
#             size        offset  fstype [fsize bsize  cpg]
  c:        251904             0  unused      0     0      # Cyl     0 -   122
  i:        250592            32   MSDOS                   # Cyl     0*-   122*
```

Le 'disklabel' nous apprend qu'il n'y a qu'une seule partition *i*, contenant un système de fichiers FAT créé sur une machine Windows. Comme d'habitude, la partition *c* représente l'ensemble du disque.

A présent, nous allons monter le système de fichiers présent dans la partition *i* vers `/mnt/flashmem`.

```
# mount -t msdos /dev/sd0i /mnt/flashmem
```

Vous pouvez désormais l'utiliser comme n'importe quel autre disque.

ATTENTION : Vous devriez toujours **démonter** le système de fichiers **avant de débrancher** le périphérique de masse. Si vous ne le faites, le système de fichiers pourrait être laissé dans un état inconsistant pouvant emmener de la corruption dans vos données.

Après avoir débranché le périphérique de votre machine, vous verrez à nouveau des messages du noyau sur la console :

```
umass0: at uhub1 port 1 (addr 2) disconnected
sd0 detached
scsibus2 detached
umass0 detached
```

14.18 - Optimiser les performances des disques durs

La performance des 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.

- CCD - Pilote de Disques Concaténés.
- RAID
- Soft Updates
- Taille du cache namei()

14.18.1 - 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.

14.18.2 - 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
```

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.

14.18.3 - Soft Updates

Un autre outil qui peut être utilisé pour accélérer 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 .

14.18.4 - Taille du cache namei()

Le cache de traduction nom-vers-inode (alias 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).

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

Question : "Je fais simplement "mount -u -o async /" ce qui rend un paquetage 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 mécanisme sûrement 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 recréées après un redémarrage.

[Index de la FAQ] [Section 13 - Multimédia] [Section 15 - Paquetages et Ports]


www@openbsd.org

$OpenBSD: faq14.html,v 1.43 2006/10/27 07:04:44 jufi Exp $

[FAQ Index] [To Section 14 - Disk Setup]

15 - The OpenBSD packages and ports system

Table of Contents

15.1 - Introduction

There are a lot of third party applications available which one might want to use on an OpenBSD system. To make this software easier to install and manage, plus to help it comply with OpenBSD's policy and goals, the third party software is *ported* to OpenBSD. This porting effort can involve many different things. Examples are: making the software use the standard OpenBSD directory layout (e.g. configuration files go into `/etc`), conforming to OpenBSD's shared library specifications, making the software more secure whenever possible, etc.

The end result of the porting effort are ready-to-install binary packages. The aim of the package system is to keep track of which software gets installed, so that it may at any time be updated or removed very easily. This way, no unnecessary files are left behind, and users can keep their systems clean. The package system also helps ensure nothing is deleted by accident, causing software to stop functioning properly. Another advantage is that **users rarely need to compile software from source**, as packages have already been compiled and are available and ready to be used on an OpenBSD system. In minutes, a large number of packages can be fetched and installed, with everything in the right place.

The packages and ports collection does NOT go through the same thorough security audit that is performed on the OpenBSD base system. Although we strive to keep the quality of the packages collection high, we just do not have enough human resources to ensure the same level of robustness and security. Of course security updates for various applications are committed to the ports tree as soon as possible, and corresponding package security updates are made available as explained below.

15.2 - Package Management

15.2.1 - How does it work?

Packages are the pre-compiled binaries of some of the most used third party software. Packages can be managed easily with the help of several utilities, also referred to as the pkg* tools:

- pkg_add(1) - a utility for installing and upgrading software packages.
- pkg_delete(1) - a utility for deleting previously installed software packages.
- pkg_info(1) - a utility for displaying information about software packages.
- pkg_create(1) - a utility for creating software packages.

In order to run properly, an application X may require that other applications Y and Z be installed. Application X is said to be dependent on these other applications, which is why Y and Z are called *dependencies* of X. In turn, Y may require other applications P and Q, and Z may require application R to function properly. This way, a whole *dependency tree* is formed.

Packages look like simple `.tgz` bundles. Basically they are just that, but there is one crucial difference: they contain some extra *packing information*. This information is used by pkg_add(1) for several purposes:

- Different checks: has the package already been installed or does it conflict with other installed packages or file names?
- Dependencies which are not yet present on the system, are automatically fetched and installed, before proceeding with the installation of the package.
- Information about the package(s) is recorded in a central repository, by default located in `/var/db/pkg/`. This will, among other things, prevent the dependencies of a package from being deleted before the package itself has been deleted. This helps ensure that an application cannot be accidentally broken by a careless user.

15.2.2 - Making things easy: PKG_PATH

You can make things really easy by using the `PKG_PATH` environment variable. Just point it to your favorite location, and pkg_add(1) will automatically look there for any package you specify, **and** also fetch and install the necessary dependencies of this package automatically.

A list of possible locations to fetch packages from is given in the following section.

Example 1: fetching from your CDROM, assuming you mounted it on `/mnt/cdrom`

```
$ export PKG_PATH=/mnt/cdrom/3.9/packages/`machine -a`/
```

Example 2: fetching from a nearby FTP mirror

```
$ export PKG_PATH=ftp://your.ftp.mirror/pub/OpenBSD/3.9/packages/`machine -a`/
```

It's usually a good idea to add a line similar to the above examples to your `~/.profile`. As with the classic `PATH` variable, you can specify multiple locations, separated by colons. **HOWEVER, every path in the `PKG_PATH` variable MUST end in a slash (/).** That way, pkg_add(1) can split the path correctly even if it holds URL schemes containing colons. If the first entry in `PKG_PATH` fails, the next one will be tried, and so on, until the package is found. If all entries fail, an error is produced.

Notice the use of machine(1) in the above command lines. This automatically substitutes your installed OpenBSD "application architecture", which is usually, but not always, your platform name. Of course, if you are using snapshots, you will replace "3.9" with "snapshots".

15.2.3 - Finding packages

A large collection of pre-compiled packages is available for the most common architectures. Just look for your package in one of these places:

- On one of the three CD-ROMs, depending on your architecture. The CD-ROMs carry only the most commonly used, freely distributable packages for the most commonly used platforms.
- On the FTP mirror servers. Packages are located in the `/pub/OpenBSD/3.9/packages` directory. From there, packages are broken down depending on architecture.
- In the package lists on the OpenBSD website:
  - Packages for OpenBSD 3.9
  - Packages for OpenBSD 3.8
  - Packages for OpenBSD 3.7

If you have the ports tree on your system, you can quickly find the package you are looking for as explained in Searching the ports tree.

You will notice that certain packages are available in a few different varieties, formally called **flavors**. Others are pieces of the same application which may be installed separately. They are called **subpackages**. This will be detailed further in Using flavors and subpackages but flavor basically means they are configured with different sets of options. Currently, many packages have flavors, for example: database support, support for systems without X, or network additions like SSL and IPv6. Every flavor of a package will have a different suffix in its package name. For detailed information about package names, please refer to packages-specs(7).

Note: Not all possible packages are necessarily available on the FTP servers! Some applications simply don't work on all architectures. Some applications can not be distributed via FTP (or CDROM) for licensing reasons. There may also be many possible combinations of flavors of a port, and the OpenBSD project just does not have the resources to build them all. If you need a combination which is not available, you will have to build the port from source. For more information on how to do that, read Using flavors and subpackages in the Ports section of this document.

15.2.4 - Installing new packages

To install packages, the utility pkg_add(1) is used. If you have made things easy for yourself by setting the PKG_PATH environment variable, you can just call pkg_add(1) with the package name, as in the following basic example.

```
$ sudo pkg_add -v screen-4.0.2
parsing screen-4.0.2
installed /etc/screenrc from /usr/local/share/examples/screen/screenrc | 71%
screen-4.0.2: complete
```

In this example the **-v** flag was used to give a more verbose output. This option is not needed but it is helpful for debugging and was used here to give a little more insight into what pkg_add(1) is actually doing. Notice the message mentioning */etc/screenrc*. Specifying multiple **-v** flags will produce even more verbose output.

Using pkg_add(1) in interactive mode

Since OpenBSD 3.9, pkg_add(1) has an interactive mode, which is enabled by invoking it with the **-i** flag, and which causes it to ask you questions when it cannot make decisions by itself. For example, if you don't know the version number of a package beforehand, you can try something like:

```
$ sudo pkg_add -i screen
Ambiguous: screen could be screen-4.0.2 screen-4.0.2-shm screen-4.0.2-static
Choose one package
         0: <None>
         1: screen-4.0.2
         2: screen-4.0.2-shm
         3: screen-4.0.2-static
Your choice: 1
screen-4.0.2: complete
```

For some packages, some important additional information will be given about the configuration or use of the application on an OpenBSD system. Since it is important, it will be displayed whether or not you use the **-v** flag. Consider the following example:

```
$ sudo pkg_add ghostscript-fonts-6.0p0
ghostscript-fonts-6.0p0: complete
You may wish to update your font path for /usr/local/share/ghostscript/fonts
--- ghostscript-fonts-6.0p0 -------------------
To install these fonts for X11, just make sure that the fontpath
lists the 75dpi or 100dpi bitmap fonts before the ghostscript fonts,
and make sure you have the string ":unscaled" appended to the bitmap
font's fontpath. This way, the bitmap fonts will be used if they
match, and the Type 1 versions will be used if the font needs to be
scaled. Below is the relevant section from a typical xorg.conf file.

   FontPath   "/usr/X11R6/lib/X11/fonts/misc/"
   FontPath   "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
   FontPath   "/usr/X11R6/lib/X11/fonts/100dpi/:unscaled"
   FontPath   "/usr/local/lib/X11/fonts/ghostscript/"
   FontPath   "/usr/X11R6/lib/X11/fonts/Type1/"
```

Let us now continue with an example of a package which has dependencies:

```
$ sudo pkg_add -v tin-1.6.2
parsing tin-1.6.2
Dependencies for tin-1.6.2 resolve to: gettext-0.14.5p1, libutf8-0.8, pcre-6.4p1, libiconv-1.9.2p3 (todo: libiconv-
1.9.2p3,gettext-0.14.5p1,pcre-6.4p1,libutf8-0.8)
tin-1.6.2:parsing libiconv-1.9.2p3
tin-1.6.2:libiconv-1.9.2p3: complete
tin-1.6.2:parsing gettext-0.14.5p1
Dependencies for gettext-0.14.5p1 resolve to: expat-1.95.6p1, libiconv-1.9.2p3 (todo: expat-1.95.6p1)
tin-1.6.2:parsing expat-1.95.6p1
tin-1.6.2:expat-1.95.6p1: complete
tin-1.6.2:gettext-0.14.5p1: complete
tin-1.6.2:parsing pcre-6.4p1
tin-1.6.2:pcre-6.4p1: complete
tin-1.6.2:parsing libutf8-0.8
tin-1.6.2:libutf8-0.8: complete
tin-1.6.2: complete
```

Again we added the **-v** flag to see more of what is happening. Upon investigating the packing information, dependencies are found and they are installed first. Somewhere in the middle you can see the gettext package being installed, which depends on libiconv. Before installing gettext, its packing information is examined and it is verified whether libiconv has already been installed.

It is possible to specify multiple package names on one line, which then all get installed at once, along with possible dependencies.

If for some reason you decide not to use PKG_PATH, it is also possible to specify the absolute location of a package on the command line. This absolute location may be a local path, or a URL referring to FTP, HTTP, or SCP locations. Let's consider installation via FTP in the next example:

```
$ sudo pkg_add ftp://ftp.openbsd.org/pub/OpenBSD/3.9/packages/`machine -a`/screen-4.0.2.tgz
screen-4.0.2: complete
```

In this example the **-v** flag wasn't used, so only needed messages are shown. Notice that the complete filename was entered by adding a **.tgz** suffix. You can also skip this suffix as in the previous examples since it is auto-completed by pkg_add(1).

Note: Not all architectures have the same packages available. Some ports do not work on certain architectures, and performance limits the number of packages that can be built on others.

For safety, if you are installing a package which you had installed earlier (or an older version of it) and removed, pkg_add(1) will **not** overwrite configuration files which have been modified. Instead, it will inform you about this as follows (only when using the **-v** flag, however!):

```
$ sudo pkg_add -v screen-4.0.2
parsing screen-4.0.2
The existing file /etc/screenrc has NOT been changed**                | 71%
It does NOT match the sample file /usr/local/share/examples/screen/screenrc
You may wish to update it manually
screen-4.0.2: complete
```

Sometimes you may encounter an error like the one in the following example:

```
$ sudo pkg_add xv-3.10ap4
xv-3.10ap4:jpeg-6bp3: complete
xv-3.10ap4:png-1.2.8: complete
xv-3.10ap4:tiff-3.7.3p0: complete
Can't install xv-3.10ap4: lib not found X11.9.0
Even by looking in the dependency tree:
        tiff-3.7.3p0, jpeg-6bp3, png-1.2.8
Maybe it's in a dependent package, but not tagged with @lib ?
(check with pkg_info -K -L)
If you are still running 3.6 packages, update them.
```

There is pkg_add(1) nicely installing dependencies, when all of a sudden it aborts the installation of xv. This is another safety precaution which is available since OpenBSD 3.7. The packing information bundled in the package includes information about shared libraries that the package expects to be installed, system libraries as well as third party libraries. If one of the required libraries cannot be found, the package is not installed because it would not function anyway.

To solve this type of conflict, you must find out what to install in order to get the required libraries on your system. There are several things to check:

- You may have older packages installed: an older version of the required library is present. In this case, upgrade these packages.
- Your system may be incomplete: you did not install one of the file sets, which contains the required library. Just add the required file set.
- Your system may be out of date: you have an older version of the required library. Boot the installer (as detailed in FAQ 4), and choose to (U)pgrade your complete system.

15.2.5 - Listing installed packages

You can see a list of installed packages by using the pkg_info(1) utility.

```
$ pkg_info
aterm-0.4.2         color vt102 terminal emulator with transparency support
bzip2-1.0.3         block-sorting file compressor, unencumbered
expat-1.95.6p1      XML 1.0 parser written in C
fluxbox-0.9.14      window manager based on the original Blackbox code
gettext-0.14.5p1    GNU gettext
imlib2-1.1.2p2      image manipulation library
jpeg-6bp3           IJG's JPEG compression utilities
libiconv-1.9.2p3    character set conversion library
libltdl-1.5.22p1    GNU libtool system independent dlopen wrapper
libungif-4.1.4      tools and library routines for working with GIF images
libutf8-0.8         provides UTF-8 locale support
mutt-1.4.2ip2       tty-based e-mail client
pcre-6.4p1          perl-compatible regular expression library
png-1.2.8           library for manipulating PNG images
screen-4.0.2        multi-screen window manager
tcsh-6.14.00p0      extended C-shell with many useful features
tiff-3.7.3p0        tools and library routines for working with TIFF images
tin-1.6.2           threaded NNTP and spool based UseNet newsreader
```

When given an installed package name (or a location of a package which is to be installed), pkg_info(1) will show more detailed information about that specific package.

15.2.6 - Updating installed packages

Since OpenBSD 3.7, it is possible to update existing packages by using the **-r** (= replace) switch to pkg_add(1). OpenBSD 3.8 introduced the **-u** switch to pkg_add(1), which has been turned into a true update mechanism in 3.9.

Let's say you had an older version of unzip installed before upgrading this box from OpenBSD 3.8 to 3.9. Now you can easily upgrade to the newer 3.9 package as follows:

```
$ sudo pkg_add -u unzip
unzip-5.52 (extracting): complete
unzip-5.51 (deleting): complete
unzip-5.52 (installing): complete
Clean shared items: complete
```

Invoking pkg_add(1) with the **-u** flag and no package name will just examine all installed packages for updated versions. When a package has dependencies, they are also examined for updates.

Note: The **-u** switch relies on the PKG_PATH environment variable. If it is not set, pkg_add(1) will not be able to find updates.

If you had a configuration file belonging to the old version, which you modified, it will be left untouched by default. You can, however, replace it with the default configuration file of the new version, by calling pkg_add(1) with the **-c** flag.

15.2.7 - Removing installed packages

To delete a package, simply take the proper name of the package as shown by pkg_info(1) (see Listing installed packages above) and use pkg_delete(1) to remove the package. In the example below, the screen package is being removed. Notice that on some occasions there are instructions of extra items 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
screen-4.0.2: complete
Clean shared items: complete
```

Note: Often, it is not necessary to specify the version numbers and flavors with the package name, since pkg_delete(1) will usually be able to find the full name by itself. You need to specify the complete package name (in the example, that is screen-4.0.2) only if ambiguity is possible due to multiple installed packages with the specified name. In that case pkg_delete(1) cannot know which package to delete.

For safety, pkg_delete(1) will not remove configuration files if they have been modified. Instead it will inform you about this as follows:

```
$ sudo pkg_delete screen
screen-4.0.2: complete
Clean shared items: complete
--- screen-4.0.2 -------------------
You should also remove /etc/screenrc (which was modified)
```

If you prefer to have those configuration files removed automatically, you can do so by using the **-c** flag.

15.2.8 - Security updates (-stable packages)

When serious bugs or security flaws are discovered in third party software, they are fixed in the -stable branch of the ports tree, and a selection of updated binary packages is made available.

Please refer to the stable packages page to find out about updated packages and important updates to the -stable branch. Note that updated packages are only available for the i386 platform. For other platforms, you will need to use the -stable branch of the ports tree and compile from source.

If you want to receive security announcements related to software in the packages and ports system, you can subscribe to the ports-security mailing list.

Package names are **always** changed in case of a package update, to avoid any risk of confusion between a package from the release and a bug-fixed package. This name change may be a higher version number or, in case the version number remains the same, a **pN** suffix is added, where **N+1** represents the number of times this package has been patched.

15.2.9 - Incomplete package installation or removal

In some odd cases, you may find that a package was not added or deleted completely, because of conflicts with other files. The incomplete installation is usually marked with "partial-" prepended to the package name. This can, for instance, happen when you coincidentally press CTRL+C during installation:

```
$ sudo pkg_add screen-4.0.2
screen-4.0.2: complete                                              7%
Adjusting md5 for /usr/local/info/screen.info-3 from 49fb3fe1cc3a3b0057518459811b6dac to
3b9c7811244fb9f8d83bb27d3a0f60d8
```

```
/usr/sbin/pkg_add: Installation of screen-4.0.2 failed , partial installation recorded as partial-screen-4.0.2
```

It is always a good idea to remove partial packages from your system, and to fix potential problems that lead to this failure. It is often an indication that you do not have a clean system with everything installed from packages, but possibly packages mixed up with other software installed straight from source.

15.3 - Working with ports

As mentioned in the introduction, packages are compiled from the ports tree. In this section we will explain how the ports tree works, when you should use it and how you can use it.

IMPORTANT NOTE: The ports tree is meant for advanced users. **Everyone is encouraged to use the pre-compiled binary packages**. Do NOT ask beginner questions on the mailing lists like "How can I get the ports tree working?". If you have questions about the ports tree, it is assumed that you have read the manual pages and this FAQ, and that you are able to work with it.

15.3.1 - How does it work?

The ports tree, a concept originally borrowed from FreeBSD, is a set of Makefiles, one for each third party application, for controlling

- where and how to fetch the source of the software,
- which other software it depends upon,
- how to alter the sources (if necessary),
- how to configure and build it,
- how to test it (optional),
- how to install it.

Apart from the Makefile, each port also contains at least the following:

- a PLIST or packing list, which contains instructions for package creation once the application has been built,
- a DESCR or description of the application,
- a distfile, containing distribution file checksums and size.

All this information is kept in a directory hierarchy under `/usr/ports`. This hierarchy contains three special subdirectories:

- `distfiles/` - where the ports system stores software distribution sets after downloading.
- `infrastructure/` - the main directory of the ports infrastructure, containing all necessary scripts and makefiles.
- `packages/` - contains all binary packages built by the ports system.

The other subdirectories all form different application categories, which contain the subdirectories of the actual ports. Complex ports may be organized to an even deeper level, for example if they have a core part and a set of extensions, or a stable and a snapshot version of the application. Every port directory must contain a `pkg/` subdirectory containing packing list(s) and description file(s). There may also be `patches/` and `files/` subdirectories, for source patches and additional files, respectively.

When a user issues make(1) in the subdirectory of a specific port, the system will recursively walk its dependency tree, check whether the required dependencies are installed, build and install any missing dependencies, and then continue the build of the desired port. All of the building happens inside the *working directory* that the port creates. This is either a subdirectory of the port's main directory, in which case it is recognized by its prefix "w-", or a subdirectory of `$WRKOBJDIR`, if the `WRKOBJDIR` variable has been set (see Configuration of the ports system).

Note: Ports are never directly installed on your system! They use a *fake installation directory*. Everything that gets installed there, is bundled together into a package (which is stored in the `packages/` subdirectory of the ports tree as mentioned earlier). Installing a port really means: creating a package, and then installing that package!

More information about the ports system may be found in these manual pages:

- ports(7) - describes the different stages (make targets) of port installation, the use of flavors and subpackages and some other options.
- bsd.port.mk(5) - in depth information about all the make targets, variables, the fake (installation directory) framework, etc.

15.3.2 - Fetching the ports tree

Before continuing, you must read the section about NOT mixing up your OpenBSD system and ports tree. Once you have decided which flavor of the ports tree you want, you can get the ports tree from different sources. The table below gives an overview of where you can find the different flavors, and in which form. An 'x' marks availability and '-' means it is not available through that specific source.

Source	Form	Flavor			
		-release	-stable	snapshots	-current
CD-ROM	.tar.gz	x	-	-	-
FTP mirrors	.tar.gz	x	-	x	-
AnonCVS	cvs checkout	x	x	-	x

On the CD-ROM and FTP mirrors, look for a file named `ports.tar.gz`. You want to untar this file in the `/usr` directory, which will create `/usr/ports`, and all the directories under it. For example:

```
$ cd /tmp
$ ftp ftp://ftp.openbsd.org/pub/OpenBSD/3.9/ports.tar.gz
$ cd /usr
$ sudo tar xzf /tmp/ports.tar.gz
```

The snapshots available on the FTP mirrors are generated daily from the -current ports tree. You will find the snapshots in the `pub/OpenBSD/snapshots/` directory. If you are using a snapshot of the ports tree, you should have installed a matching snapshot of OpenBSD. Make sure you keep your ports tree and your OpenBSD system in sync!

For more information about obtaining the ports tree via AnonCVS, please read the AnonCVS page which contains a list of available servers and a number of examples.

15.3.3 - Configuration of the ports system

NOTE: This section introduces some additional global settings for building applications from ports. You can skip this section, but then you will be required to perform many of the make(1) statements in the examples as root.

Because the OpenBSD project does not have the resources to fully review the source code of all software in the ports tree, you can configure the ports system to take a few safety precautions. The ports infrastructure is able to perform all building as a regular user, and perform only those steps that require superuser privileges as root. Examples are the `fake` and `install` make targets. However, because root privileges are always required at some point, the ports system will not save you when you decide to build a malicious application.

- You can set up sudo(8) and have the ports system use it for tasks requiring superuser permissions. Just add a line to `/etc/mk.conf` containing

  ```
  SUDO=/usr/bin/sudo
  ```

- You can modify the ownerships of the ports tree so that you can write there as a regular user. In this case, the regular user has been added to the wsrc group, and the underlying directories are made group writable.

  ```
  # chgrp -R wsrc /usr/ports
  # find /usr/ports -type d -exec chmod g+w {} \;
  ```

- You can have the ports system use systrace(1) by adding the following to `/etc/mk.conf`

  ```
  USE_SYSTRACE=Yes
  ```

  This enforces the build procedure to stay inside allowed directories, and prohibits writing in illegal places, thereby considerably reducing the risk of a damaged system. Note that the use of systrace(1) adds about 20% overhead in build time.

It is possible to use a read-only ports tree by separating directories that are written to during port building:

- The working directory of ports. This is controlled by the `WRKOBJDIR` variable, which specifies the directory which will contain the working directories.
- The directory containing distribution files. This is controlled by the `DISTDIR` variable.
- The directory containing newly built binary packages. This is controlled by the `PKGREPOSITORYBASE` variable.

For example, you could add the following lines to `/etc/mk.conf`

```
WRKOBJDIR=/usr/obj/ports
DISTDIR=/usr/distfiles
PKGREPOSITORYBASE=/usr/packages
```

If desired, you can also change the ownership of these directories to your local username and group, so that the ports system can create underlying working directories as a regular user.

15.3.4 - Searching the ports tree

Once you have the ports tree in place on your system, it becomes very easy to search for software. Just use `make search key="searchkey"`, as shown in the following example.

```
$ cd /usr/ports
$ make search key=rsnapshot
Port:   rsnapshot-1.2.1
Path:   net/rsnapshot
Info:   remote filesystem snapshot utility
Maint:  Sigfred Haversen
Index:  net
```

```
L-deps:
B-deps: :net/rsync
R-deps: :net/rsync
Archs:  any
```

The search result gives a nice overview of each application that is found: the port name, the path to the port, a one-line description, the port's maintainer, keywords related to the port, library/build/runtime dependencies, and architectures on which the port is known to work.

15.3.5 - Straightforward installation: a simple example

For clarity's sake, let's consider a simple example: rsnapshot. This application has one dependency: rsync.

```
$ cd /usr/ports/net/rsnapshot
$ make install
===>  Checking files for rsnapshot-1.2.1
>> rsnapshot-1.2.1.tar.gz doesn't seem to exist on this system.
>> Fetch http://www.rsnapshot.org/downloads/rsnapshot-1.2.1.tar.gz.
100% |***********************************************|   133 KB    00:00
>> Size matches for /usr/ports/distfiles/rsnapshot-1.2.1.tar.gz
>> Checksum OK for rsnapshot-1.2.1.tar.gz. (sha1)
===>  rsnapshot-1.2.1 depends on: rsync-2.6.6p0 - not found
===>  Verifying install for rsync-2.6.6p0 in net/rsync
===>  Checking files for rsync-2.6.6p0
>> rsync-2.6.6.tar.gz doesn't seem to exist on this system.
>> Fetch ftp://ftp.samba.org/pub/rsync/old-versions/rsync-2.6.6.tar.gz.
100% |***********************************************|   673 KB    00:04
>> Size matches for /usr/ports/distfiles/rsync-2.6.6.tar.gz
>> Checksum OK for rsync-2.6.6.tar.gz. (sha1)
===>  Verifying specs:  c
===>  found c.39.0
===>  Extracting for rsync-2.6.6p0
===>  Patching for rsync-2.6.6p0
===>  Configuring for rsync-2.6.6p0
  [...snip...]
===>  Building for rsync-2.6.6p0
  [...snip...]
===>  Faking installation for rsync-2.6.6p0
  [...snip...]
===>  Building package for rsync-2.6.6p0
Link to /usr/ports/packages/i386/ftp/rsync-2.6.6p0.tgz
Link to /usr/ports/packages/i386/cdrom/rsync-2.6.6p0.tgz
===>  Installing rsync-2.6.6p0 from /usr/ports/packages/i386/all/rsync-2.6.6p0.tgz
rsync-2.6.6p0: complete
===> Returning to build of rsnapshot-1.2.1
===>  rsnapshot-1.2.1 depends on: rsync-2.6.6p0 - found
===>  Extracting for rsnapshot-1.2.1
===>  Patching for rsnapshot-1.2.1
===>  Configuring for rsnapshot-1.2.1
  [...snip...]
===>  Building for rsnapshot-1.2.1
  [...snip...]
===>  Faking installation for rsnapshot-1.2.1
  [...snip...]
===>  Building package for rsnapshot-1.2.1
Link to /usr/ports/packages/i386/ftp/rsnapshot-1.2.1.tgz
Link to /usr/ports/packages/i386/cdrom/rsnapshot-1.2.1.tgz
===>  rsnapshot-1.2.1 depends on: rsync-2.6.6p0 - found
===>  Installing rsnapshot-1.2.1 from /usr/ports/packages/i386/all/rsnapshot-1.2.1.tgz
rsnapshot-1.2.1: complete
```

As you can see, the ports system is doing many things automatically. It will fetch, extract, and patch the source code, configure and build (compile) the source, install the files into a fake directory, create a package (corresponding to the packing list) and install this package onto your system (usually under `/usr/local/`). And it does this recursively **for all dependencies** of the port. Just notice the "`===> Verifying install for ...`" and "`===> Returning to build of ...`" lines in the above output, indicating the walk through the dependency tree.

If a previous version of the application you want to install, was already installed on your system, you can use **`make update`** instead of **`make install`**. This will call pkg_add(1) with the **-r** flag.

Note: Large applications will require a lot of system resources to build. Good examples are compilers like GCC 4.0 or the Java 2 Software Development Kit. If you get "out of memory" type of errors when building such a port, this usually has one of two causes:

- Your resource limits are too restrictive. Adjust them with ksh's ulimit or csh's limit command. If that doesn't help, just become root before starting the build, or use sudo(8) with the `-c` flag to run the build with the resources limited by the specified login class (refer to login.conf(5) for details about login classes).
- You simply don't have enough RAM in your machine.

15.3.6 - Cleaning up after a build

You probably want to clean the port's default working directory after you have built the package and installed it.

```
$ make clean
===>  Cleaning for rsnapshot-1.2.1
```

In addition, you can also clean the working directories of all dependencies of the port with this make target:

```
$ make clean=depends
===>  Cleaning for rsync-2.6.6p0
===>  Cleaning for rsnapshot-1.2.1
```

If you wish to remove the source distribution set(s) of the port, you would use

```
$ make clean=dist
===>  Cleaning for rsnapshot-1.2.1
===>  Dist cleaning for rsnapshot-1.2.1
```

In case you have been compiling multiple flavors of the same port, you can clear the working directories of all these flavors at once using

```
$ make clean=flavors
```

15.3.7 - Uninstalling a port's package

It is very easy to uninstall a port:

```
$ make uninstall
===> Deinstalling for rsnapshot-1.2.1
rsnapshot-1.2.1: complete
Clean shared items: complete
```

This will call pkg_delete(1) to have the corresponding package removed from your system. If desired, you can also uninstall and re-install a port's package by using

```
$ make reinstall
===>  Cleaning for rsnapshot-1.2.1
/usr/sbin/pkg_delete rsnapshot-1.2.1
rsnapshot-1.2.1: complete
Clean shared items: complete
===>  Installing rsnapshot-1.2.1 from /usr/ports/packages/i386/all/rsnapshot-1.2.1.tgz
rsnapshot-1.2.1: complete
```

If you would like to get rid of the package you just built, you can do so as follows:

```
$ make clean=package
===>  Cleaning for rsnapshot-1.2.1
rm -f /usr/ports/packages/i386/all/rsnapshot-1.2.1.tgz
```

Use "`packages`" instead of "`package`" to also delete any subpackages (see below).

15.3.8 - Using flavors and subpackages

Please do read the ports(7) manual page, which gives a good overview of this topic. There are two mechanisms to control the packaging of software according to different needs.

The first mechanism is called **flavors**. A flavor usually indicates a certain set of compilation options. For instance, some applications have a "no_x11" flavor which can be used on systems without X. Some shells have a "static" flavor, which will build a statically linked version. There are ports which have different flavors for building them with different graphical toolkits. Other examples include: support for different database formats, different networking options (SSL, IPv6, ...), different paper sizes, etc.

Summary: Most likely you will use flavors when a package has not been made available for the flavor you are looking for. In this case you will specify the desired flavor and build the port yourself.

Every flavor of a port will have its own working directory during building and every flavor will be packaged into a correspondingly named package to avoid any confusion. To see the different flavors of a certain port, you would change to its subdirectory and issue

```
$ make show=FLAVORS
```

The second mechanism is called **subpackages**. A porter may decide to create subpackages for different pieces of the same application, if they can be logically separated. You will

often see subpackages for the client part and the server part of a program. Sometimes extensive documentation is bundled in a separate subpackage because it takes up quite some disk space. Other examples are: extensive test suites which come with the software, separate modules with support for different things, etc.

Summary: You will use subpackages when you need only part of a certain application, not the whole thing. However, it is very likely that the subpackage you are looking for exists, and is ready to be fetched and installed from your nearest FTP mirror.

To list the different subpackages available for a port, use

```
$ make show=MULTI_PACKAGES
```

It is possible to select which subpackage(s) to install from within the ports tree. After some tests, this procedure will just call pkg_add(1) to install the desired subpackage(s).

```
$ env SUBPACKAGE="-server" make install
```

Note: The subpackages mechanism only handles packages, it does not modify any configuration options before building the port. For that purpose you must use flavors.

15.3.9 - Security updates

When serious bugs or security flaws are discovered in third party software, they are fixed in the **-stable** branch of the ports tree. Remember that the lifecycle is 1 year: only the current and last release are updated, as explained in FAQ 5 - OpenBSD's Flavors.

This means all you need to do is make sure you check out the correct branch of the ports tree, and build the desired software from it. You can keep your tree up-to-date with CVS, and in addition subscribe to the ports-security mailing list to receive security announcements related to software in the ports tree.

Of course, security updates reach the -current ports tree before being taken up in the -stable branch.

15.4 - FAQ

15.4.1 - I'm getting all kinds of crazy errors. I just can't seem to get this ports stuff working at all.

It is very likely that you are using a system and ports tree which are not in sync.

Sorry?

- Read EVERYTHING about OpenBSD's Flavors: -release, -stable, and -current. The short summary is as follows, but please do read the document mentioned above to get an idea about which one it is you want to use.
  - Release: What is on the CD.
  - Stable: Release, plus security and reliability enhancements.
  - Current: The development version of OpenBSD.
- Do NOT check out a -current ports tree and expect it to work on a -release or -stable system. This is one of the most common errors and you will irritate people when you ask for help about why "nothing seems to work!" **If you follow -current, you need both a -current system and a -current ports tree**. Yes, this really does mean a wonderful new port will typically not work on your "older" system -- even if that system was -current just a few weeks ago. Keep in mind that if you use X11 as part of your system, it must also follow the corresponding branch!
- Because no intrusive changes are made in -stable, it is possible to use a -stable ports tree on a -release system, and vice versa. There is no need to update all your installed packages after applying a few errata patches to your system.

Another common failure is a missing X11 installation. Even if the port you try to build has no direct dependency on X11, a subpackage of it or its dependencies may require X11 headers and libraries. Building ports on systems without X11 is not supported, so if you insist on doing so, you are on your own to figure it out. For many ports, there are, however, "no_x11" flavored packages available, which you can install without needing X11 on your system.

15.4.2 - The latest version of my Top-Favorite-Software is not available!

If you are using a release or stable version of OpenBSD, you will not find any package updates until the next release, or until security issues occur which justify an update of the port in the -stable branch, and of the corresponding package.

WARNING: DO NOT mix versions of Ports and OpenBSD!

Doing so will sooner or later (probably very soon, in fact) cause you headaches trying to solve all kinds of errors!

But hey, I am all -current here!

The ports collection is a volunteer project. Sometimes the project simply doesn't have the developer resources to keep everything up-to-date. Developers pretty much pick up what they consider interesting and can test in their environment. Your donations can make a difference for testing ports on more platforms.

Some individual ports may lag behind the mainstream versions because of this. The ports collection may have a version back of a program from January while a new version of the

program has been released by its developers in May three months ago. Often this is a conscious decision; the new version may have problems in it on OpenBSD that the maintainer is trying to solve, or that have simply made the application worse than the old version: OpenBSD may have different goals than the mainstream developers in other projects, which sometimes results in features and design or implementation choices that are undesirable from OpenBSD developers' point of view. The update may also be postponed because the new version is not considered a crucial update.

If you really need a new version of a port, you should ask the maintainer of the port to update the port (see below on how to find out who the maintainer is). If you can help with this, all the better.

15.4.3 - Why is there no package for my Top-Favorite-Software?

There are several possible reasons for this:

- On the OpenBSD CD-ROMs, there is no space to include every possible package for every possible platform. Therefore only the most used packages are included on the CDs. Additionally, some software can only be redistributed for free, this means it cannot be included on the CDs. If you cannot find a package on the CDs, try another source, such as an FTP mirror.
- Some software must simply not be redistributed in binary package form at all, according to its license. Other software is encumbered by patents and can therefore not be redistributed. If your Top-Favorite-Software falls into this category, you will need to use the port and compile from source.
- Obvious, but sometimes forgotten: there is no port of your Top-Favorite-Software. You can verify this by searching the ports tree. If there is indeed no port of your Top-Favorite-Software, then you are welcome to help.

15.4.4 - Why is there no port of my Top-Favorite-Software?

The ports collection is a volunteer project. Active port development is done by a limited number of people, in their spare time. These people usually make new ports only for software they use directly or are interested in.

You can help. Consider creating your own port. There is some documentation available on this: Building an OpenBSD Port. Read it, and read it again. Especially the part about *maintaining* your port. Then try making a new port, and test it carefully and step by step. If finally it works OK for you, submit it to the ports mailing list at ports@openbsd.org. Chances are good you will get some feedback and testing from other people. If the testing is successful, your port will be considered to be taken up in the ports tree.

15.4.5 - Why is my Top-Favorite-Software not part of the base system?

Because OpenBSD is supposed to be a small stand-alone UNIX-like operating system, we need to draw a line as to what to include. Generally, for an application to be included in the base system:

- It must meet the high quality standards, laid out in the goals of the OpenBSD project.
- Its license must not be too restrictive and must be compatible with the BSD license.
- It must not be too large, in order to keep the size of the base system acceptable.

Further answers to this question are also found in FAQ 1.

15.4.6 - What should I use: packages or ports?

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 as well. 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 779kB package file. In contrast, building CVSup on the same machine from source is a huge task, requiring many tools and bootstrapping a compiler, taking 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 much easier, and definitely the recommended way of adding applications to an OpenBSD system.

15.4.7 - How do I tweak these ports to have maximum performance?

OpenBSD is about stability and security. Just like the GENERIC kernel is the default and the only supported kernel, the ports team makes sure the ports work and are stable. If you want to switch on all kinds of compiler options, you are on your own. Please do not ask questions on the mailing lists such as why it does not work, when you tried to switch on a few hidden knobs to make it work faster. In general, all this tweaking is not necessary for more than 99% of users, and it is very likely to be a complete waste of time, for you, the user, as well as for the developers who read about your "problems" when in reality there are none.

15.4.8 I submitted a new port/an update weeks ago. Why isn't it committed?

The ports team has very limited resources and no committer was able to look at your port/update in time. As frustrating as it may be, just ignore this fact. Take care of your port, send updates and eventually someone might take care of it. It has happened before that people suddenly have some free time to spend on committing ports or their interests shift to new areas and suddenly your old submission becomes interesting, if it is remembered.

15.5 - Reporting problems

If you have trouble with an existing port, please send e-mail to the port maintainer. To see who is the maintainer of the port, type, for example:

```
$ cd /usr/ports/archivers/unzip
$ make show=MAINTAINER
```

Alternatively, if there is no maintainer, or you can't reach him/her, send e-mail to the OpenBSD ports mailing list, ports@openbsd.org. Please do NOT use the misc@openbsd.org mailing list for questions about ports.

In any case please provide:

- Your OpenBSD version including any patches you may have applied. The kernel version is given by: `sysctl -n kern.version`
- The version of your ports tree: if the file `/usr/ports/CVS/Tag` exists, provide its contents. If this file is absent, you are using the -current ports tree.
- A complete description of the problem. Don't be afraid to provide details. Mention all the steps you followed before the problem occurred. Is the problem reproducible? The more information you provide, the more likely you will get help.

For ports which do not build correctly, a complete build transcript is almost always required. You can use the portslogger script, found in `/usr/ports/infrastructure/build`, for this. A sample run of portslogger might be:

```
$ mkdir ~/portslogs
$ cd /usr/ports/archivers/unzip
$ make clean install 2>&1 | /usr/ports/infrastructure/build/portslogger \
          ~/portslogs
```

After this, you should have a logfile of the build in your ~/portslogs directory that you can send to the port maintainer. Also, make sure you are not using any special options in your build, for example in `/etc/mk.conf`.

Alternatively, you can

- Use script(1) to create a complete build transcript. Do not remove the configure information.
- Attach the output of pkg_info(1) if it seems even remotely relevant.
- gcc(1) internal compiler errors ask you to report the bug to the gcc mailinglist. It does save time if you follow their direction, and provide at least the various files produced by `gcc -save-temps`.

15.6 - Helping us

There are many ways you can help. They are listed below, by increasing order of difficulty.

- Report problems as you experience them.
- You can systematically test ports and report breakages, or suggest improvements. Just have a look at the Port Testing Guide.
- Test the updates to ports which are posted to the ports mailing list.
- Send updates or patches to a port's maintainer, or to the ports mailing list if the port has no maintainer. Generally this is highly appreciated, unless your patches will cause developers to waste time rather than save time.
- Create new ports. If you are really eager and want to know everything about porting applications to OpenBSD, a good starting point is Building an OpenBSD Port.

Note: For all creation of new ports and subsequent testing, or for testing port updates, you **must run a -current system!** In general, this is not a desirable environment because of its continuously evolving nature, so proceed only if you are sure about committing yourself to following -current.

[FAQ Index] [To Section 14 - Disk Setup]

$OpenBSD: faq15.html,v 1.29 2006/08/11 20:28:59 steven Exp $