FreeBSD 7.0

Le guide complet du FreeBSD

par Michael W. Lucas

PEARSON

INTRODUCTION

Bienvenue dans *FreeBSD 7.0* ! Ce livre est un passage obligé pour tout administrateur système voulant construire, configurer et gérer des serveurs FreeBSD. Il sera également utile à ceux qui veulent employer FreeBSD en tant que station de travail, serveur, ferme de systèmes sans disque dur, et autres usages. Lorsque vous terminerez la lecture de ce livre, vous devriez être en mesure d'utiliser FreeBSD pour fournir des services réseau. Vous devriez également avoir appris comment gérer, patcher et maintenir vos systèmes FreeBSD et vous devriez être capable de maîtriser les notions de base des réseaux, de la sécurité des systèmes et de la gestion des logiciels. Nous étudierons FreeBSD version 7, qui est la version recommandée pour un usage en production au moment de la publication de ce livre. Ceci dit, la majeure partie du contenu de ce livre s'applique aussi bien aux anciennes qu'aux futures versions.

Qu'est-ce que FreeBSD ?

FreeBSD est un système d'exploitation de type Unix librement disponible, largement utilisé par des fournisseurs d'accès à Internet, dans des solutions tout-en-un et des systèmes embarqués et partout où la fiabilité par rapport à un matériel informatique est primordiale. FreeBSD est miraculeusement apparu sur Internet un jour de la semaine dernière, complètement formé, directement extrait du cerveau mutant de son héroïque créateur et de sa noble intelligence. Je plaisante, la vérité est bien plus impressionnante. FreeBSD est le résultat de presque trois décennies de développement continu, de recherche et de raffinement. L'histoire de FreeBSD commence en 1979, avec BSD.

BSD : le grand-père de FreeBSD

Il y a des années, AT&T a eu besoin d'un grand nombre de logiciels spécialisés et spécialement écrits pour ses opérations commerciales. Il ne fut pas autorisé à concourir dans l'industrie du logiciel et ne put donc vendre son logiciel. Au lieu de cela, AT&T délivra des licences pour divers morceaux de logiciels ainsi que leurs codes sources à des universités, à des prix très très bas. Les universités purent économiser de l'argent en utilisant ces logiciels plutôt que leurs équivalents commerciaux aux licences coûteuses. Les étudiants eurent accès à ces technologies géniales et purent lire leurs codes sources afin de voir comment tout fonctionnait. En retour, AT&T s'était montré, avait gagné un peu d'argent de poche et une génération d'informaticiens s'était fait les dents sur les technologies d'AT&T. Tout le monde y avait gagné. Le logiciel le plus connu distribué sous cette licence est Unix.

Comparé aux systèmes d'exploitation modernes, Unix avait beaucoup de problèmes. Mais des milliers d'étudiants avaient accès à son code source et des centaines de professeurs avaient besoin de projets intéressants pour leurs étudiants. Si un programme se comportait de façon étrange ou si le système d'exploitation lui-même avait un problème, les personnes qui vivaient quotidiennement avec le système avaient les outils et la motivation pour le réparer. Grâce à leurs efforts, Unix a été amélioré rapidement et beaucoup de fonctionnalités que nous tenons à présent pour acquises furent créées à ce moment-là. Les étudiants ajoutèrent la possibilité de contrôler les processus en cours, également connue sous le nom de *job control*. Le système de fichiers Unix S51K faisant pleurer les administrateurs système comme des petits enfants, ils le remplacèrent par le système de fichiers Fast File System, dont les fonctionnalités se propagèrent dans tous les systèmes de fichiers modernes. Beaucoup de petits programmes très utiles furent écrits au fil des années, remplaçant au fur et à mesure des parties complètes d'Unix.

Le groupe de recherche en informatique (*Computer Science Research Group*, CSRG) de l'université de Berkeley, en Californie, participa à ces améliorations et centralisa les améliorations du code d'Unix. Le CSRG collecta les modifications provenant d'autres universités, les évalua, les mit en paquetages, et distribua gratuitement cette compilation à tous les possesseurs d'une licence AT&T Unix valide. Le CSRG signa également un contrat avec l'Agence des projets de recherches avancées pour la Défense américaine (*Defense Advanced Research Projects Agency*, DARPA) afin d'implémenter diverses fonctionnalités au sein d'Unix, comme TCP/IP. La collection de logiciels résultants commença à se faire connaître sous le nom de distribution logicielle de Berkeley (*Berkeley Software Distribution*, BSD).

Les utilisateurs de BSD prirent ces logiciels et les améliorèrent encore, puis rendirent leurs ajouts à BSD. Aujourd'hui, nous considérons ceci comme un fonctionnement plutôt standard pour un projet open source, mais en 1979 c'était révolutionnaire ! BSD fut plutôt un succès ; si vous regardez la déclaration de copyright sur un ancien système BSD, vous verrez ceci :

```
Copyright 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
The Regents of the University of California. All rights reserved.
```

Oui, quinze années de travail – toute une vie dans le monde du développement de logiciels. Combien d'autres morceaux de logiciels sont non seulement toujours utilisés, mais également en développement actif, quinze ans après leur naissance ?

En fait, il y a eu tellement d'ajouts et d'améliorations apportés à BSD que le CSRG finit par admettre qu'au fil des années la quasi-totalité du code de l'Unix original avait été remplacée par du code créé par le CSRG et ses contributeurs. Vous devrez bien chercher pour trouver un code original d'AT&T.

En fin de compte, le CSRG baissa les bras et il devint clair que le projet BSD allait se terminer. Après quelques disputes d'ordre politique au sein de l'université de Californie, le code BSD fut publié en 1992 à l'intention du grand public sous une licence qui se fit connaître sous le nom de licence BSD.

La licence BSD

Le code BSD est disponible pour tous sous des termes qui constituent probablement la licence la plus libérale de l'histoire du développement de logiciels. Cette licence peut être résumée comme suit :

- Ne prétendez pas l'avoir écrit.
- Ne nous rendez pas responsables si cela ne fonctionne pas.
- N'utilisez pas notre nom pour promouvoir vos produits.

Cela signifie que vous pouvez faire presque tout ce que vous voulez avec le code BSD (la licence originale exigeait que les utilisateurs soient mis au courant qu'un logiciel inclut du code sous licence BSD, mais cette exigence fut plus tard abandonnée). Il n'y a même pas obligation de partager vos modifications avec les auteurs initiaux ! Les personnes sont libres de prendre BSD et d'y inclure des produits propriétaires, des produits open source ou des produits libres ; elles peuvent même l'imprimer sur des cartes perforées et recouvrir le gazon avec. Vous voulez copier 10 000 CD de BSD et les distribuer à vos amis ? Faites-vous plaisir. Au lieu d'un *copyright*, la licence BSD est souvent prise comme un *copycenter*, comme dans *Amenez ceci dans un centre de copie et faites-en quelques copies pour vous-même*. Sans surprise, des compagnies comme Sun Microsystems sautèrent sur l'occasion : c'était gratuit, cela fonctionnait, et énormément de jeunes diplômés possédaient de l'expérience avec cette technologie. Une société, BSDi, fut créée spécifiquement pour profiter de l'Unix BSD.

La bagarre entre AT&T, le CSRG et BSDi

Chez AT&T, le travail sur Unix continuait, même si le CSRG traçait son joyeux chemin. AT&T prit des morceaux de la distribution BSD Unix et les intégra dans son Unix, puis délivra à nouveau des licences pour le tout aux universités qui avaient fourni ces améliorations. Cela fonctionna bien pour AT&T jusqu'à ce que la société soit séparée en plusieurs morceaux et que les compagnies résultant de cette séparation soient autorisées à entrer dans l'industrie du logiciel. AT&T avait un atout particulièrement intéressant : un système d'exploitation haut de gamme ayant été intensivement débogué par des milliers de personnes. Ce système d'exploitation avait beaucoup de fonctionnalités utiles, comme toute une variété de petites mais puissantes commandes, un système de fichiers moderne, le job control, et TCP/IP. AT&T créa une filiale, Unix Systems Laboratories (USL), qui commença à vendre joyeusement et au prix fort Unix aux entreprises, tout en maintenant les relations universitaires qui avaient contribué à l'élaboration d'un système d'exploitation aussi avancé.

La première sortie publique du code BSD de Berkeley, en 1992, fut assez mal accueillie par USL, qui intenta presque immédiatement un procès à l'université et aux entreprises de logiciels qui en avaient profité, notamment BSDi. L'université de Californie affirma que le CSRG avait compilé BSD avec l'aide de milliers de contributeurs tiers non liés à AT&T et qu'il relevait de la propriété intellectuelle du CSRG d'en disposer à sa guise.

Cette action en justice poussa beaucoup de personnes à prendre une copie de BSD pour voir ce qu'il en était vraiment de cette histoire, tandis que d'autres commencèrent à construire des projets en se basant sur BSD. Parmi les produits résultants, il y eut 386BSD, qui fut finalement utilisé en tant que cœur de FreeBSD 1.0.

En 1994, après deux années de bagarre judiciaire, les avocats de l'université de Californie prouvèrent que la majeure partie de l'AT&T Unix venait de BSD, et pas le contraire. Pire, AT&T avait violé la licence BSD en supprimant le copyright du CSRG des fichiers assimilés (seule une entreprise très spéciale peut violer la licence la plus libérale du monde !). Seulement une demi-douzaine de fichiers étaient sources de contentieux et, pour résoudre ces problèmes en suspens, USL donna certains d'entre eux à BSD tout en en gardant certains en tant que parties propriétaires.

Une fois l'affaire passée, une nouvelle version de BSD Unix fut dévoilée au monde sous le nom de BSD 4.4-Lite. BSD 4.4-Lite2, une mise à jour ultérieure, est le grand-père de l'actuel FreeBSD, ainsi que de toutes les autres variantes de BSD en usage de nos jours.

La naissance de FreeBSD

L'un des premiers résultats de BSD fut 386BSD, une version de BSD conçue pour fonctionner sur le peu coûteux processeur 386[1]. Le projet 386BSD porta avec succès BSD vers le processeur 386 d'Intel, puis stagna. Après une période pendant laquelle le projet fut négligé, un groupe d'utilisateurs de 386BSD décida de partir de son côté et créa FreeBSD de façon à pouvoir garder le système d'exploitation à jour. À peu près au même moment, plusieurs autres groupes démarrèrent leurs propres branches de 386BSD ; il n'en reste aujourd'hui que NetBSD.

386BSD et FreeBSD 1 dérivaient du BSD sorti en 1992, qui avait provoqué le courroux d'AT&T. À la suite de l'action en justice, on demanda à tous les utilisateurs du BSD original de baser tout nouveau travail sur BSD 4.4-Lite2. BSD 4.4-Lite2 n'était pas un système d'exploitation complet – en particulier les quelques fichiers que AT&T retenait comme étant propriétaires étaient vitaux aux fonctions du système. (si ces fichiers n'avaient pas été vitaux, AT&T n'en aurait pas fait tout un fromage !). L'équipe de développement de FreeBSD travailla frénétiquement pour remplacer ces fichiers manquants et FreeBSD 2.0 fut rapidement publié par la suite. Le développement n'a depuis lors jamais cessé.

Aujourd'hui, FreeBSD est utilisé sur Internet par quelques-unes des sociétés les plus vitales et les plus visibles de l'Internet. Yahoo! repose quasi entièrement sur FreeBSD. IBM, Nokia, Juniper, NetApp et beaucoup d'autres entreprises de matériel informatique utilisent FreeBSD dans des systèmes embarqués et vous ne l'auriez jamais su si on ne vous l'avait pas dit. Le fait est que si une entreprise a besoin d'une grosse bande passante sur Internet, elle utilise probablement FreeBSD ou l'un des

1. À l'époque, plusieurs milliers de dollars pour un ordinateur n'était pas cher payé. Les jeunes voyous d'aujourd'hui n'ont aucune idée de la chance qu'ils ont.

systèmes de la famille BSD. Comme le smog ou les araignées, FreeBSD est partout autour de vous ; si vous ne le voyez pas, c'est tout simplement parce qu'il fonctionne. La clé de la fiabilité de FreeBSD est l'équipe de développement et la communauté des utilisateurs, ce qui, en fait, revient au même.

Le développement de FreeBSD

On dit toujours que gérer des programmeurs revient à élever des chats. Malgré le fait que l'équipe de développement de FreeBSD soit éparpillée dans le monde entier et que ses membres parlent des dizaines de langues différentes, ils travaillent ensemble en tant que partie de l'équipe de FreeBSD. Ils ressemblent donc plus à une troupe de lions qu'à une collection de chats d'appartement. Au contraire de certains autres projets, tout le développement de FreeBSD se fait en public. Trois groupes de personnes sont responsables du progrès de FreeBSD : les committers, les contributeurs, et les utilisateurs.

Les committers

FreeBSD dispose d'environ 500 développeurs, ou committers. Les *committers* ont accès en lecture et écriture au dépôt maître du code source de FreeBSD et peuvent développer, déboguer ou améliorer n'importe quel morceau du système (le terme *committer* vient de leur capacité à "commettre" des modifications dans le code source). Étant donné qu'ils peuvent casser le système de manière subtile ou évidente, les committers ont une lourde responsabilité. Ils doivent garder FreeBSD en état de fonctionnement ou, au moins, ne pas le casser lorsqu'ils ajoutent de nouvelles fonctionnalités et évaluent des patches provenant de contributeurs. La plupart de ces développeurs sont volontaires ; seule une poignée d'entre eux est vraiment payée pour faire ce minutieux travail et la plupart de ces personnes ne sont rémunérées que si leur action est en relation avec un autre travail. Par exemple, Intel emploie un committer pour s'assurer que FreeBSD supporte correctement ses cartes réseau. FreeBSD tient un haut rang parmi les poids lourds d'Internet ; donc Intel a besoin que ses cartes fonctionnent sous FreeBSD.

Pour vous brancher directement sur la ruche de développement de FreeBSD, inscrivez-vous à la liste de diffusion **FreeBSD-hackers@FreeBSD.org**, qui contient la plupart des discussions techniques. Une partie des discussions techniques est divisée entre plusieurs listes de diffusion plus spécialisées – par exemple, les détails les plus subtils de l'implémentation du réseau sont discutés dans **FreeBSD-net@FreeBSD.org**.

Régulièrement, au bout de quelques années, l'équipe des committers élit un petit nombre de ses membres pour servir en tant qu'équipe cœur, ou *Core*. Le travail du Core est à la fois vital, sous-estimé et mal compris. Core est théoriquement responsable de la gestion générale de FreeBSD mais, en pratique il ne gère pas beaucoup plus que les disputes personnelles ou les conflits procéduriers entre committers. Core approuve également les nouveaux committers et délègue des responsabilités concernant de gros morceaux de FreeBSD à des individus ou des groupes. Par exemple, il délègue l'autorité sur le système de ports et de paquetages à l'équipe de gestion des ports. Core ne définit pas la direction de l'architecture de FreeBSD et ne dicte pas non plus de processus ou de procédures ; c'est aux committers de le

faire et ils doivent être massivement d'accord. Néanmoins, Core est là pour suggérer, persuader et inspirer.

Core prend également en charge les aspects les plus pénibles du management. Certaines des fonctions clés de la gestion des entreprises sont la supervision, la motivation et la gestion des problèmes entre les personnes. La supervision est effectuée par les millions d'utilisateurs qui se plaignent lourdement lorsque quelque chose se casse ou se comporte de manière inexplicable, et les committers de FreeBSD se motivent eux-mêmes. La partie vraiment désagréable du management consiste à calmer les querelles entre les personnes, et c'est exactement la fonction à laquelle Core consacre le plus de temps. Le prestige lié au fait de pouvoir dire "Je fais partie de Core" est une récompense insuffisante pour compenser l'obligation de gérer une dispute entre deux développeurs talentueux qui se tapent mutuellement sur les nerfs.

Les contributeurs

En plus de l'équipe des committers, FreeBSD bénéficie de l'apport de milliers de contributeurs. Les *contributeurs* n'ont pas à se soucier de ne pas casser le dépôt principal du code source du système d'exploitation ; ils soumettent simplement des patches aux committers pour inspection. Les committers évaluent les soumissions des contributeurs et décident de ce qui doit être accepté et de ce qui doit être refusé. Un contributeur qui soumet beaucoup de patches de grande qualité finit souvent par être recruté pour faire partie des committers.

Par exemple, j'ai passé plusieurs années à contribuer à FreeBSD, chaque fois qu'une envie pressante me prenait. Chaque fois que je pense avoir gâché ma vie, je vais voir le site Web de FreeBSD et j'y retrouve mes travaux qui ont été acceptés par les committers et distribués à des milliers de personnes. Après avoir proposé la première édition de ce livre à l'éditeur, j'ai passé mon temps libre à soumettre des patches à la FAQ de FreeBSD. Finalement, des membres du projet de documentation de FreeBSD m'ont approché et m'ont demandé de devenir committer. En récompense, j'ai reçu une adresse e-mail et l'opportunité de m'humilier devant des milliers de personnes, démontrant une fois de plus qu'aucune bonne action ne reste impunie.

Si je n'avais jamais fait de contribution, je serais resté un simple utilisateur. Il n'y a aucune honte à cela non plus.

Les utilisateurs

Les *utilisateurs* sont les personnes qui se servent de FreeBSD. Il est impossible d'estimer de manière réaliste le nombre d'utilisateurs de FreeBSD, bien que des organisations telles que le BSDstats Projet (**http://www.bsdstats.org**) tentent de le faire. Après tout, vous pouvez très bien télécharger gratuitement la totalité de FreeBSD et ne jamais vous enregistrer, ni mettre à jour votre système, ni envoyer un e-mail sur une liste de diffusion. Des entreprises comme Netcraft estiment qu'entre 5 et 15 % des ordinateurs connectés à Internet sont basés sur BSD. Si vous retirez tous les Windows des ordinateurs professionnels, ce pourcentage risque d'augmenter considérablement.

Comme FreeBSD est largement le plus populaire des BSD open source, ce nombre n'est pas inconcevable. En outre, un serveur FreeBSD pouvant gérer des centaines

de milliers de domaines Internet, un nombre disproportionné de sites utilisent donc FreeBSD comme système d'exploitation sous-jacent. Cela signifie qu'il y a actuellement des centaines de milliers, si ce n'est des millions, d'administrateurs système FreeBSD dans le monde.

Les autres BSD

FreeBSD est peut-être le BSD le plus connu, mais il n'est pas pour autant le seul. BSD 4.4-Lite2 a engendré plusieurs projets différents, chacun avec ses propres buts et spécificités. Ces projets ont à leur tour eu une descendance, dont plusieurs représentants s'épanouissent aujourd'hui.

NetBSD

NetBSD ressemble à FreeBSD sur de nombreux points et ils partagent des développeurs et des codes. Le but principal de NetBSD est de fournir un système d'exploitation sécurisé et fiable pouvant être porté vers n'importe quelle plateforme matérielle avec un minimum d'effort. En tant que tel, NetBSD tourne sur des VAX, des appareils PocketPC et des serveurs haut de gamme SPARC et Alpha. Je fais tourner NetBSD sur mon ordinateur de poche HP Jordana[1].

OpenBSD

OpenBSD s'est séparé de NetBSD en 1996 avec l'objectif de devenir le BSD le plus sécurisé. OpenBSD a été le premier à supporter l'accélération cryptographique matérielle et ses développeurs sont fiers à juste titre que leur installation par défaut soit largement immunisée contre les exploits à distance et ce depuis de nombreuses années. L'équipe OpenBSD a contribué à la publication de plusieurs logiciels de grande valeur, le plus remarquable étant la suite OpenSSH utilisée aujourd'hui par à peu près tous les systèmes d'exploitation et tous les vendeurs de matériel informatique.

Mac OS X

Apple utilise depuis le début de larges portions de FreeBSD dans son système OS X. Si vous êtes à la recherche d'un système d'exploitation stable avec un joli visage et un cœur puissant, Mac OS X est sans aucun doute fait pour vous. Alors que FreeBSD fait un excellent bureau pour un professionnel de l'informatique, je ne le mettrais pas face à ma grand-mère : je mettrais certainement Mac OS X et j'irais même jusqu'à penser que j'ai fait le bon choix. Mais Mac OS X inclut beaucoup de choses qui ne sont pas du tout nécessaires pour un serveur Internet et il ne fonctionne que sur du matériel Apple ; je ne le recommanderai donc pas pour un petit serveur bon marché à usage général.

1. Si jamais vous devez prouver que vous êtes le super geek de la mort qui tue, l'installation d'Unix sur votre assistant personnel suffira certainement.

En outre, le code voyage dans les deux sens. FreeBSD a ainis intégré du code originellement développé pour Mac OS X. Bien que vous n'ayez pas accès au code source de l'interface utilisateur de Mac OS X, vous pouvez lire le code source de son cœur BSD et de son noyau Mach : Apple les a publiés tous les deux sous le nom de code *Darwin*.

Les enfants de FreeBSD

Plusieurs projets ou produits sont issus de FreeBSD. FreeNAS, qui a été récompensé par ses pairs, transforme un système x86 en serveur de fichiers réseau au moyen d'un simple menu. FreeSBIE est un CD-ROM de démarrage qui vous permet de faire tourner FreeBSD sans l'installer. Le projet m0n0wall est également un CD-ROM démarrable, mais il transforme votre système en firewall avec une jolie interface Web pour le gérer. PC-BSD ajoute un joli visage à FreeBSD, tentant ainsi de le rendre utilisable par ma grand-mère. D'autres projets comme ceux-ci apparaissent de temps à autre ; bien que certains ne soient pas couronnés de succès, je suis certain qu'au moment de la sortie de ce livre, nous aurons un ou deux autres solides membres dans ce groupe.

Les autres Unix

Plusieurs autres systèmes d'exploitation dérivent ou émulent d'une façon ou d'une autre l'Unix initial. Cette liste n'est en aucun cas exhaustive, mais elle met l'accent sur les points principaux.

Solaris/OpenSolaris

L'Unix le plus connu est Solaris de Sun Microsystems et sa descendance, OpenSolaris. Solaris fonctionne sur le matériel haut de gamme qui supporte des douzaines de processeurs et des tonnes de disques. Solaris, spécialement dans ses premières versions, a de fortes racines BSD. Beaucoup d'applications de niveau professionnel fonctionnent avec Solaris. Il tourne principalement sur la plate-forme matérielle SPARC construite par Sun, ce qui permet à Sun de supporter des fonctionnalités intéressantes comme la mémoire et les cartes échangeables à chaud. OpenSolaris cible néanmoins de plus en plus d'offres matérielles.

Pourquoi "de type Unix" ?

Vous aurez remarqué que l'on dit que FreeBSD, Linux et les autres sont *de type Unix* au lieu de *Unix*. Le terme *Unix* est une marque déposée de The Open Group. Pour qu'un système d'exploitation ait le droit de s'appeler Unix, le fabricant doit prouver que cet OS respecte la version courante de la Single Unix Specification. FreeBSD reste généralement dans le standard, mais les tests continus et les recertifications coûtent de l'argent, argent que le Projet FreeBSD n'a pas à dépenser. La certification Unix exige également que l'on signe un papier déclarant non seulement qu'on est responsable de la conformité de FreeBSD vis-à-vis de la Single Unix Specification, mais également qu'on réparera toute déviation par rapport au standard qui pourrait être trouvée à l'avenir. Le modèle de développement de FreeBSD rend ceci encore plus difficile – des bogues sont découverts et les déviations sont corrigées, mais personne ne peut signer un papier garantissant une compatibilité à 100 % par rapport au standard.

AIX

Un autre concurrent d'Unix est le système d'IBM, AIX. Le principal atout d'AIX est son système de fichiers journalisés, qui enregistre au fur et à mesure toutes les transactions des disques durs et permet donc un rétablissement rapide des données après un crash. Ce fut également l'Unix standard d'IBM et de tout ce qui était appuyé par IBM, durant de nombreuses années. AIX repose largement sur BSD.

Linux

Linux est un proche cousin d'Unix, écrit à partir de rien. Linux est semblable à FreeBSD sur de nombreux points, bien que FreeBSD ait un héritage bien plus ancien et se trouve être plus adapté à une utilisation commerciale que Linux. En effet, l'une des exigences de Linux est que tout utilisateur qui le distribue doit rendre ses modifications disponibles pour les utilisateurs finaux, alors que BSD n'a pas ce genre de restriction. Bien entendu, un fan de Linux dirait : "FreeBSD est plus vulnérable à une exploitation commerciale"… Les développeurs de Linux croient au principe du partage réciproque, tandis que les développeurs de BSD offrent leur travail sans condition ni obligation. Tout cela dépend donc de votre point de vue sur cette question.

Beaucoup de nouveaux utilisateurs de systèmes Unix croient à un conflit entre les camps BSD et Linux. Mais, si vous regardez un peu plus attentivement, vous verrez que la plupart des développeurs de ces systèmes d'exploitation communiquent et collaborent de manière libre et ouverte. Seuls les extrémistes parmi les développeurs et les utilisateurs provoquent des frictions, un peu comme les hooligans des clubs de football ou les fans des différentes séries *Star Trek*.

IRIX, HP/UX…

Parmi les autres Unix, citons IRIX de Silicon Graphics, un Unix solide pour applications graphiques et HP/UX de Hewlett-Packard, populaire dans les grandes entreprises. Une recherche rapide sur le Web fait apparaître beaucoup de concurrents plus petits, comme Tru64 Unix et UnixWare du suicidaire SCO Group. Vous trouverez également quelques vieux systèmes laissés pour compte, comme A/UX d'Apple, et Xenix de Microsoft (au temps où les dinosaures regardaient nerveusement le ciel et où mon père chassait le mammouth pour les rituels tribaux, Microsoft possédait une licence lui permettant de vendre un Unix). Beaucoup d'applications haut de gamme sont conçues pour fonctionner de manière optimisée sur une version particulière d'Unix. Tous les Unix modernes ont retenu les leçons de ces systèmes d'exploitation plus anciens et les Unix et systèmes de type Unix d'aujourd'hui sont remarquablement proches.

Les points forts de FreeBSD

Avec tout ça, qu'est-ce qui rend FreeBSD unique ?

Portabilité

Le but du FreeBSD Project est de fournir un système d'exploitation librement redistribuable, stable et sécurisé, fonctionnant sur le matériel informatique le plus susceptible d'être utilisé par le public. Cela comprend de nos jours les systèmes compatibles Intel x86, comme le 486, les différents Pentium, AMD, et ainsi de suite, sans oublier l'architecture amd64 d'AMD (copiée par Intel en tant que EM64T). Les systèmes x86 plus anciens ne fonctionnent plus directement avec les nouvelles versions de FreeBSD, mais la plupart ne fonctionnent plus depuis longtemps ou ne sont pas prêts de changer de système d'exploitation de sitôt.

La plate-forme ARM utilisée sur les matériels embarqués est une nouvelle addition à FreeBSD et elle est bien supportée sur certaines cartes spécifiques. FreeBSD supporte également les systèmes SPARC de Sun et les Itanium (IA64) d'Intel tout autant que le processeur PowerPC utilisé encore récemment par Apple. Bien que ces autres plates-formes ne soient pas pour autant considérées comme des parents pauvres, elles ne reçoivent pas le même niveau d'attention que les architectures x86 et amd64.

Puissance

FreeBSD fonctionnant parfaitement sur du matériel 386, il tourne à merveille sur les ordinateurs modernes. C'est plutôt bien d'avoir un système d'exploitation qui ne demande pas un Pentium III et un demi-giga de RAM pour simplement lancer l'interface utilisateur. Vous pouvez donc dédier votre matériel à l'accomplissement d'un vrai travail au lieu de tâches dont vous n'avez rien à faire. Si vous choisissez de lancer une jolie interface graphique avec toutes sortes de bidules tournoyants et de sonorités fantaisistes, FreeBSD vous conviendra ; par contre, il ne vous pénalisera pas si vous ne voulez pas de tout cela. FreeBSD conviendra également pour les matériels les plus récents à *n* processeurs.

Gestion simplifiée des logiciels

FreeBSD simplifie également la gestion des logiciels grâce à son arborescence des ports. Traditionnellement, lancer des logiciels dans un système de type Unix nécessitait une certaine compétence. Le système des ports facilite considérablement tout ceci en automatisant et en documentant les processus d'installation, de désinstallation et de mise à jour pour des milliers de paquetages logiciels.

Processus de mise à jour optimisé

Au contraire d'autres systèmes d'exploitation qui exigent des procédures de mise à jour risquées et douloureuses, le processus de mise à jour de FreeBSD construit un système d'exploitation optimisé pour votre matériel et vos applications. Ceci permet à FreeBSD d'utiliser toutes les fonctionnalités supportées par votre matériel, au lieu de se contenter du plus petit dénominateur commun. Si vous changez de matériel, vous pouvez reconstruire votre système d'exploitation pour mieux gérer ce matériel particulier. Des constructeurs comme Sun et Apple font exactement la même chose, mais ils contrôlent à la fois le matériel et le logiciel ; FreeBSD réussit le même tour de force mais sur tous les types de matériels commercialisés.

Système de fichiers avancé

Un *système de fichiers* est la manière dont l'information est stockée sur le disque physique – c'est ce qui transforme le fichier *Mon CV* en une série de zéros et de uns sur un disque dur. FreeBSD accepte des systèmes de fichiers très sophistiqués et peut supporter un pétaoctet (soit un millier de milliers de gigaoctets). Son système de fichiers par défaut est hautement résistant aux problèmes et il lit et écrit extrêmement rapidement. Le système de fichiers BSD est suffisamment avancé pour que des vendeurs d'Unix commerciaux l'utilisent comme base pour leur propre système de fichiers.

Qui devrait utiliser FreeBSD ?

Bien que FreeBSD puisse être utilisé en tant que machine de bureau ou de développement, son histoire montre un fort parti pris pour le support de services Web, e-mail, fichiers. FreeBSD est plus célèbre pour sa force en tant que serveur Internet et il est un excellent choix comme plate-forme sous-jacente pour n'importe quel service réseau. Si des sociétés aussi importantes que Yahoo! comptent sur lui pour produire des services fiables, c'est qu'il fonctionnera aussi bien pour vous.

Si vous songez à utiliser FreeBSD (ou n'importe quel Unix) sur votre machine de bureau, vous devrez comprendre comment votre ordinateur fonctionne. FreeBSD n'est pas le meilleur choix si vous avez besoin de la simplicité du tout-en-un-clic. Si c'est là votre but, prenez un Mac et vous pourrez profiter de la puissance d'Unix lorsque vous en avez besoin, et ne pas vous en soucier le reste du temps. Par contre, si vous voulez apprendre FreeBSD, la meilleure solution est de l'utiliser sur votre machine de bureau, comme nous le verrons plus tard.

Qui devrait utiliser un autre BSD ?

NetBSD et OpenBSD sont les compétiteurs les plus proches de FreeBSD. Au contraire de ce qui peut se passer dans le monde commercial, cette compétition est surtout amicale. FreeBSD, NetBSD et OpenBSD partagent librement du code et des développeurs ; certaines personnes vont jusqu'à maintenir les mêmes sous-systèmes dans plusieurs systèmes d'exploitation.

Si vous voulez utiliser un matériel vieux ou exotique, NetBSD est un bon choix. Pendant des années, j'ai utilisé NetBSD sur une antique station de travail SGI en tant que serveur de fichiers et serveur de noms de domaine (DNS). Cela a fonctionné correctement jusqu'à ce que le matériel finisse par rendre l'âme dans un dernier souffle de fumée.

OpenBSD a implémenté une impressionnante variété de fonctionnalités liées à la sécurité. Beaucoup de ces outils finissent par être intégrés au sein de FreeBSD, mais cela prend des mois ou des années. Si vous êtes réellement concerné par la sécurité et que vous n'avez pas besoin d'un support sophistiqué pour les machines multiprocesseur, jetez un coup d'œil à OpenBSD.

Si vous testez simplement pour voir ce qui se passe, n'importe quel BSD conviendra !

Qui devrait utiliser un système propriétaire ?

Les systèmes d'exploitation tels que Solaris, Windows, AIX et autres systèmes de ce genre sont toujours assez populaires, malgré les systèmes d'exploitation open source qui rongent leurs parts de marché. Les entreprises haut de gamme sont fortement liées à ces systèmes d'exploitation. Bien les choses évoluent lentement, dans de tels environnements, vous êtes probablement coincé avec ces systèmes d'exploitation commerciaux. Mais glisser une machine d'occasion sous FreeBSD pour s'occuper de services de base comme le monitoring ou le serveur de fichiers du département peut vous rendre la vie plus facile à un meilleur coût. D'ailleurs, Yahoo! et NetApp ont construit tout leur business en utilisant FreeBSD à la place de systèmes d'exploitation commerciaux.

Bien entendu, si le logiciel dont vous avez besoin ne fonctionne que sous un système d'exploitation propriétaire, le choix est plutôt clair. Mais demandez toujours à l'éditeur si une version FreeBSD est disponible ; vous pourriez être agréablement surpris.

Comment lire ce livre

Beaucoup de livres d'informatique sont assez lourds et épais pour assommer un bœuf, pourvu que vous ayez la force de les lever suffisamment haut. De plus, ils ont soit une portée encyclopédique, soit ils sont si lourdement détaillés qu'ils sont en fait difficiles à lire. Avez-vous réellement besoin de vous référer à une copie d'écran où l'on vous dit de cliquer sur "OK" ou encore sur "Accepter l'accord de licence" ? Et à quand remonte la dernière fois où vous vous êtes assis pour lire une encyclopédie ?

Cet ouvrage est légèrement différent. Il est conçu pour être lu en une fois, du début à la fin. Vous pouvez passer des paragraphes si vous le souhaitez, mais chaque chapitre s'appuie sur ce qui a été vu avant. Bien que ce ne soit pas un petit livre, il est moins épais que beaucoup d'ouvrages d'informatique. Une fois lu en entier, il constitue une bonne référence.

Si vous achetez souvent des livres d'informatique, n'hésitez pas à appliquer la bonne vieille méthode du "lire un chapitre à la fois pour mieux apprendre". Je ne vais pas vous dorloter ; si vous avez pris ce livre, c'est que vous avez des neurones à connecter ou que vous êtes de passage chez quelqu'un qui l'a (dans ce dernier cas, espérons que votre hôte soit assez malin pour éloigner ce livre avant que vous en sachiez assez pour devenir dangereux).

Que devez-vous savoir ?

Ce livre est destiné à un administrateur Unix novice. Il y a vingt ans, l'administrateur Unix moyen possédait de l'expérience dans la programmation noyau et travaillait à son diplôme de master en informatique. Il y a dix ans, il était toujours un utilisateur avancé d'Unix, possédant de véritables connaissances en programmation et au moins une licence en informatique. De nos jours, les systèmes d'exploitation de type Unix sont librement disponibles et sont moins chers que la nourriture ; même un enfant de 12 ans peut lancer Unix, lire le code source et en apprendre suffisamment pour intimider des plus âgés que lui. Je n'attends donc pas de vous que vous en sachiez beaucoup sur Unix avant de continuer.

Pour utiliser ce livre de façon optimale, certaines tâches basiques telles que changer de répertoire, lister les fichiers d'un répertoire et vous logger avec un nom d'utilisateur et un mot de passe doivent vous être familières. Si vous n'êtes pas habitué aux commandes de base et du shell Unix, je vous recommande de débuter par un livre comme le *Manuel d'administration du système UNIX* publié par Evi Nemeth et ses amis (Prentice Hall PTR, 2006). Afin de rendre les choses plus aisées pour les administrateurs Unix les plus novices, j'inclurai les commandes exactes requises pour produire les résultats souhaités. Si vous apprenez mieux par l'exemple, vous devriez trouver ici tout ce dont vous avez besoin.

Vous aurez également besoin de vous y connaître un peu en matériel informatique – pas énormément, mais un minimum. Il est utile par exemple de savoir reconnaître un câble IDE, un câble SCSI, un câble SATA. Votre niveau de connaissance dépend du matériel que vous utilisez, mais si vous êtes suffisamment intéressé pour avoir pris ce livre et l'avoir lu jusque-là, c'est que vous en savez probablement déjà assez.

Pour l'administrateur Unix novice

Si vous êtes nouveau sous Unix, la meilleure façon d'apprendre est de manger votre propre nourriture pour chien. Non, cela ne veut pas dire dîner avec Max. Je vous explique : si vous étiez à la tête d'une entreprise de nourriture pour chien, vous auriez l'ambition de faire un produit que votre chien mangerait avec plaisir. Si votre chien fait la tête devant votre dernière recette, vous avez un problème. Le point important à retenir est que si vous travaillez avec un outil ou créez quelque chose, alors vous avez tout intérêt à l'utiliser. Ceci s'applique également aux systèmes d'exploitation de type Unix et notamment FreeBSD.

FreeBSD sur une machine de bureau

Si vous voulez sérieusement apprendre à vous servir de FreeBSD, je vous suggère de supprimer le système d'exploitation de votre principal ordinateur et de le passer sous FreeBSD. Oui, je sais, à présent cette pâtée ne semble pas si mauvaise. Mais l'apprentissage d'un système d'exploitation est identique à l'apprentissage d'une langue ; une immersion totale est le moyen le plus rapide et le plus efficace pour apprendre. C'est ce que j'ai fait et aujourd'hui je peux utiliser un système de type Unix pour faire tout ce que je veux. En fait, ce livre a été entièrement écrit sur un ordinateur portable sous FreeBSD, en utilisant l'éditeur de texte open source XEmacs et la suite OpenOffice.org. J'utilise également FreeBSD pour regarder des films, copier et écouter des MP3, gérer mes comptes bancaires, traiter mes e-mails, et surfer sur le Web. Au moment où j'écris ces lignes, j'ai une douzaine de démons animés qui se baladent au-dessus des fenêtres de mon bureau et je fais parfois une petite pause pour les zapper avec ma souris. Si ceci n'est pas une stupide blague de bureau, je ne sais pas ce qui en est une[1].

1. Dans la première édition de ce livre, j'ai négligé de mentionner comment réaliser une stupide astuce de bureau identique, ce qui généra plus de questions par e-mail que n'importe quel autre sujet de l'ouvrage. C'est une erreur que je ne ferai pas deux fois !

De nos jours, beaucoup d'administrateurs Unix viennent d'une culture Windows. Ils travaillaient d'arrache-pied dans leur petit monde, lorsque la direction a fait une descente et a dit avant de disparaître dans un souffle : "Vous pouvez vous occuper d'un système de plus, n'est-ce pas ? Je suis heureux de l'apprendre ! Au fait, voici une machine Unix." Après que le tout nouvel administrateur Unix ait décidé de ne pas s'ouvrir les veines, ni celles du chef, ni de démarrer une nouvelle et excitante carrière de technicien d'autopsie de baleine, il tâte timidement le système du doigt. Il apprend que `ls` est comme `dir` et que `cd` est le même sur les deux plates-formes. Ce qu'il ne peut pas apprendre, en venant de cette culture, c'est la manière dont une machine Unix pense. Unix ne s'adaptera pas à vous, c'est à vous de vous y adapter. Windows et OS X exigent des adaptations identiques, mais ils le cachent derrière une façade scintillante. En gardant cela à l'esprit, prenons un peu de temps pour savoir comment réfléchir à propos d'Unix.

Comment réfléchir à propos d'Unix

De nos jours, la plupart des systèmes Unix sont fournis par défaut avec de jolies interfaces utilisateur graphiques, mais celles-ci ne sont que du tape-à-l'œil. Le vrai travail se fait en ligne de commande, quel que soit le nombre de soi-disant outils pour le cacher. La ligne de commande est en fait l'une des forces d'Unix, et c'est de là que vient sa souplesse sans égal.

La philosophie sous-jacente tient dans *beaucoup de petits outils, chacun accomplissant une seule tâche mais le faisant bien.* Le répertoire des programmes locaux de mon ordinateur portable (`/usr/local/bin/`) possède 662 programmes. J'ai installé le moindre d'entre eux, directement ou indirectement. La plupart sont de simples petits programmes qui ne font qu'une seule chose, à quelques rares exceptions telles que la suite bureautique. Cet éventail de petits outils rend Unix extrêmement flexible et adaptable. Beaucoup de paquetages de logiciels commerciaux essaient de tout faire ; ils finissent par offrir toutes sortes de fonctionnalités mais avec des performances médiocres dans leurs fonctions essentielles. Souvenez-vous, à une certaine époque, il fallait impérativement être programmeur pour utiliser un système Unix. Les programmeurs ne voient aucun inconvénient à construire leurs propres outils. Le concept Unix des canaux a encouragé cela.

Canaux de communication

Les personnes habituées aux environnements graphiques comme ceux de Windows et Mac OS X ont probablement du mal à saisir la manière dont Unix gère les entrées et les sorties. Elles sont habituées à cliquer sur quelque chose et à voir un message OK, une erreur, rien, ou encore (toujours trop souvent) un bel écran bleu aux lettres vives et high-tech expliquant dans une langue curieuse pourquoi le système s'est crashé. Unix fait les choses de manière légèrement différente.

Les programmes Unix possèdent trois canaux de communication : l'entrée standard, la sortie standard, et l'erreur standard. Une fois que vous aurez compris comment chacun d'eux fonctionne, vous serez sur la bonne voie pour appréhender le système dans sa globalité.

L'*entrée standard* est la source d'information. Lorsque vous êtes à la console en train de taper une commande, l'entrée standard c'est les données qui proviennent du clavier. Si un programme écoute sur le réseau, l'entrée standard est le réseau.

Beaucoup de programmes peuvent réarranger l'entrée standard pour accepter des données provenant du réseau, d'un fichier, d'un autre programme, du clavier, ou de toute autre source.

La *sortie standard* est l'endroit où la sortie du programme s'affiche. C'est fréquemment la console (l'écran). Les programmes réseau retournent habituellement leurs données vers le réseau. Des programmes peuvent renvoyer leurs données vers un fichier, un autre programme, sur le réseau, ou vers n'importe quelle ressource disponible dans l'ordinateur.

Enfin, l'*erreur standard* est l'endroit où le programme envoie ses messages d'erreur. Généralement, les programmes en mode console retournent leurs erreurs à la console ; d'autres enregistrent les erreurs dans un fichier. Si vous mettez un programme en place de manière incorrecte, il pourrait tout simplement supprimer toutes ses informations d'erreurs.

Ces trois canaux de communication peuvent être arrangés arbitrairement, un concept qui est peut-être le plus grand obstacle pour les nouveaux utilisateurs et administrateurs Unix. Par exemple, si vous n'aimez pas que les erreurs apparaissent sur le terminal, vous pouvez les rediriger vers un fichier. Si vous ne voulez pas taper de manière répétitive un gros morceau d'information dans une commande, vous pouvez le mettre dans un fichier (de façon à pouvoir le réutiliser) et vider le fichier dans l'entrée standard de la commande. Ou même, encore mieux, vous pouvez lancer une commande pour produirfe cette information et la mettre dans un fichier, ou simplement canaliser (envoyer) la sortie de la première commande directement dans la seconde, sans même vous ennuyer avec un fichier.

Les petits programmes, les canaux et la ligne de commande

En poussant cette logique à l'extrême, ces canaux d'entrée/sortie et la diversité des outils semblent terriblement puissants. Lorsque j'ai vu un administrateur système taper quelque chose comme ce qui suit pendant ma session de formation initiale sur Unix, j'ai sérieusement songé à changer de carrière.

```
tail -f /var/log/messages | grep -v popper | grep -v named &
```

D'incompréhensibles lignes de texte commencèrent à défiler sur l'écran. Pire encore, mon mentor continuait à taper pendant que ce charabia se déversait ! Si vous venez d'un environnement informatique tout-en-un-clic, une longue chaîne de commandes comme celle-ci est vraiment intimidante. Que signifient tous ces mots marrants ? Et le & ? Vous voulez m'apprendre *quoi* ?

Considérez l'apprentissage de l'usage de la ligne de commande comme l'apprentissage d'une langue. Lorsque nous apprenons une langue, nous commençons par des mots simples. À mesure que notre vocabulaire augmente, nous apprenons également la manière d'enchaîner les mots. Nous apprenons que, placés dans un certain ordre, les mots ont un sens et que dans un ordre différent ils n'ont aucun sens. Vous ne parliez pas correctement à 3 ans : arrêtez de pratiquer et vous reviendrez à ce niveau.

Des programmes plus simples et plus petits alliés à des canaux de communication fournissent une flexibilité quasi illimitée. N'avez-vous jamais rêvé d'utiliser dans un programme une fonction provenant d'un autre programme ? En utilisant plusieurs programmes plus petits et en arrangeant les entrées et les sorties selon

vos désirs, vous pouvez faire en sorte qu'un système Unix se comporte comme vous voulez. Finalement, vous vous sentirez véritablement paralysé si vous ne pouvez tout simplement pas passer la sortie d'une commande au travers de `| sort -rnk 6 | less`[1].

Tout est fichier

Vous ne pouvez pas être dans Unix depuis longtemps sans jamais avoir entendu que "tout est fichier". Les programmes, les informations sur les comptes et la configuration du système sont tous stockés dans des fichiers. Unix n'a pas de registre à la Windows ; si vous sauvegardez les fichiers, vous possédez le système entier.

Mieux, le système identifie le matériel en tant que fichiers ! Votre lecteur de CD-ROM est un fichier `/dev/acd0`. Les cartes réseau apparaissent en tant que fichiers dans `/dev/net`. Même les périphériques virtuels comme les sniffeurs de paquets et les partitions sur les disques durs sont des fichiers.

Lorsque vous avez un problème, gardez cela en mémoire. Tout est fichier ou se trouve dans un fichier, quelque part sur votre système. Tout ce que vous avez à faire est de le trouver !

Sommaire de ce livre

FreeBSD 7.0 est composé des chapitres suivants :

Chapitre 1 : Obtenir plus d'aide

Ce chapitre présente les ressources et informations que le FreeBSD Project et ses adeptes fournissent aux utilisateurs. Aucun livre ne peut tout couvrir, mais savoir comment utiliser le grand nombre de ressources de FreeBSD sur Internet aidera à combler les éventuels vides que vous pourriez trouver ici.

Chapitre 2 : Installer FreeBSD

Ce chapitre vous donne une vue d'ensemble de l'installation de FreeBSD et propose quelques conseils pour une installation optimale.

Chapitre 3 : Start me up ! Le processus de démarrage

Ce chapitre détaille le processus de démarrage de FreeBSD et vous apprend comment faire démarrer, stopper et redémarrer votre système dans différentes configurations.

Chapitre 4 : Lisez ceci avant de faire plus de dégâts !

Ici nous discutons de la manière de sauvegarder vos données à la fois au niveau du système global et fichier par fichier et comment rendre vos modifications aisément réversibles.

1. Cette horrible chose prend la sortie de la dernière commande, la trie en ordre inverse selon le contenu de la sixième colonne et l'affiche page par page. Si vous avez des milliers de lignes de sortie et si vous voulez savoir quelles entrées ont la plus haute valeur dans la sixième colonne, voici comment faire. Si vous avez beaucoup de temps, vous pouvez envoyer la sortie dans un tableur et les tripoter avec des commandes tout aussi obscures, pendant un long moment.

Chapitre 5 : Jouer avec le noyau

Ce chapitre décrit la configuration du noyau de FreeBSD. Au contraire de certains autres systèmes d'exploitation, on attend de vous que vous personnalisiez le noyau de FreeBSD afin de le façonner en fonction de vos propres besoins. Ceci vous donne une formidable flexibilité et vous permet d'optimiser le potentiel de votre matériel.

Chapitre 6 : Le réseau

Ici nous parlerons du réseau et de la manière dont il fonctionne sous FreeBSD.

Chapitre 7 : Sécuriser votre système

Ce chapitre vous apprend comment rendre votre ordinateur résistant aux attaques et aux intrusions.

Chapitre 8 : Disques et systèmes de fichiers

Ce chapitre couvre certains détails du travail avec des disques durs sous FreeBSD, le support d'autres systèmes de fichiers et de quelques systèmes de fichiers réseau.

Chapitre 9 : Fonctionnalités de sécurité avancées

Nous discutons ici de certaines des fonctionnalités de sécurité les plus intéressantes que l'on trouve dans FreeBSD.

Chapitre 10 : Exploration de *`/etc`*

Ce chapitre décrit les nombreux fichiers de configuration de FreeBSD et la façon dont ils fonctionnent.

Chapitre 11 : Rendre votre système utile

Ici, nous décrivons le système de ports et de paquetages qu'utilise FreeBSD pour gérer les logiciels supplémentaires.

Chapitre 12 : Gestion avancée des logiciels

Ce chapitre traite certains points un peu subtils à propos du lancement de logiciels dans un système FreeBSD.

Chapitre 13 : Mettre à jour FreeBSD

Ce chapitre vous apprend à utiliser le processus de mise à jour de FreeBSD. Ce système de mise à jour est l'un des plus doux et des plus remarquables de tous les systèmes d'exploitation.

Chapitre 14 : La carte routière d'Internet : DNS

Ce chapitre décrit le protocole DNS et vous apprend comment l'installer et le dépanner.

Chapitre 15 : Les petits services système

Ici nous discutons de certains des petits programmes que vous devrez gérer afin d'utiliser FreeBSD proprement.

Chapitre 16 : Le spam, les vers et les virus (plus les e-mails, si vous insistez)

Ce chapitre décrit comment mettre en place un système de courrier électronique sous FreeBSD pour délivrer des messages de manière fiable tout en repoussant le spam et les virus.

Chapitre 17 : Services Web et FTP

Ce chapitre vous apprend comment mettre en place et sécuriser ces deux services Internet vitaux.

Chapitre 18 : Astuces de disques avec GEOM

Ce chapitre couvre certaines des jolies techniques que FreeBSD supporte pour faire des miroirs de disques, exporter des disques au travers du réseau et, d'une manière générale, vous faire passer du bon temps à protéger et manipuler vos données.

Chapitre 19 : Performances et supervision système

Ce chapitre couvre certains des outils de test de performance et de dépannage de FreeBSD et montre la manière d'interpréter les résultats. Nous verrons également la journalisation système et l'implémentation du protocole SNMP de FreeBSD.

Chapitre 20 : En marge de FreeBSD

Ce chapitre vous apprend certaines des figures de style les plus intéressantes que vous pouvez réaliser avec FreeBSD, comme faire tourner des systèmes sans disque ou avec de minuscules disques, ainsi que des configurations redondantes et à tolérance de panne.

Chapitre 21 : Les paniques du système (et de l'administrateur système) et les crashs

Ce chapitre vous apprend ce qu'il faut faire dans ces rares occasions où un système FreeBSD tombe en panne, comment déboguer les problèmes, et comment créer un rapport de problème utile.

Annexe : Quelques MIB *sysctl* intéressants

Cette annexe fournit les informations de base concernant certaines options de personnalisation du noyau mises à votre disposition.

Allez, maintenant ça suffit pour ces données d'introduction. Allons-y !