

Le mythe BSD

Il existe toute sorte d'idées, d'objections et de mythes à propos des BSD. En voici quelques uns :

Matériel

"BSD ne supporte pas assez de matériel." Est-ce que Linux supporte plus de matériel que BSD ? Probablement, mais est-ce important ? Uniquement si vous avez ce matériel. Est-ce que Linux supporte plus de matériel que Windows ? Dans ce domaine, on peut aussi taquiner MacOS... BSD supporte tout le matériel commun nécessaire pour un serveur ou une station de travail. Il y a des oublis, mais cela se comble à chaque nouvelle version, comme tout système. Tous les BSDs ont une liste de matériels supportés que vous pouvez vérifier. Vous pouvez également utiliser un LiveCD.

Compatibilité des programmes

"Mais Linux a plus de logiciels que BSD !" C'est faux. Si une application est écrite de manière portable, il y a 98% de chances qu'elle se compile correctement sur un système conforme POSIX. FreeBSD et Debian ont la plus grosse collection de programmes open-source. De même, tous les BSDs ont une couche d'émulation Linux qui fournit une compatibilité binaire.

Popularité

"Mais Linux est plus populaire." Et alors ? Windows est également plus populaire. Oublions ça ! Souvent, l'argument de la popularité signifie quelque chose comme "On trouve plus facilement de support". Mais il existe beaucoup d'endroits et de personnes qui fournissent un support commercial, sans compter l'énorme communauté répartie sur les diverses listes de diffusions.

Usage

"BSD est difficile à utiliser, plus avancé, plus compliqué, moins orienté utilisateurs." C'est certainement différent de Windows ou Linux, mais sur quels points ? Dans beaucoup de cas, on va le trouver plus logique et simple, principalement dans son effort d'uniformité et de clarté. BSD n'est pas plus dur, il demande juste un regard un peu différent.

Élitisme

"Les utilisateurs BSD sont un groupe d'élitistes bornés." Désolé, les personnes de BSD ne sont que de simples femmes et hommes comme tout le monde. Ils aiment Unix, BSD et Beastie.

Pourquoi utiliser BSD ?

Dois-je utiliser BSD ou Linux ? C'est une question assez difficile. Voici quelques points pour vous y aider :

1. Ça marche. Quand votre système est installé, un ensemble d'outil est déjà présent.
2. "Si ce n'est pas cassé, ne le change pas !": Si vous utilisez déjà un système open-source et que vous en êtes satisfait, il n'y a probablement pas de bonnes raisons de le changer.
3. Les systèmes BSD ont notamment de meilleures performances que Linux, mais ce n'est pas flagrant. Dans beaucoup de cas, cela sera minime, voire invisible. Il se peut même qu'elles soient en dessous d'un Linux.
4. En général, les systèmes BSD ont une meilleure réputation en ce qui concerne la stabilité, qui résulte d'un code de base bien plus mature.
5. La licence BSD, moins restrictive, peut être plus attractive que la GPL.
6. BSD peut exécuter la plupart des binaires Linux, alors, alors que le contraire n'est pas possible.

Liens et sites importants :

Tous les BSDs disposent d'une excellente documentation :

<http://www.FreeBSD.org/>

<http://www.NetBSD.org/>

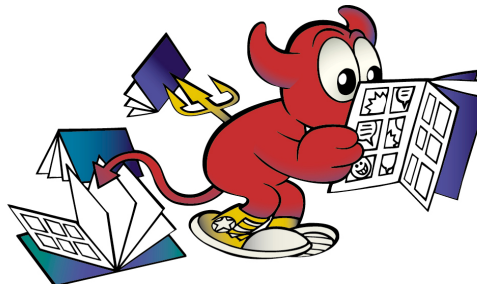
<http://www.OpenBSD.org/>

<http://www.DragonFlyBSD.org/>

http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/linux-comparison/

<http://sites.inka.de/mips/unix/bsdlinux.html>

http://www.FreeBSD.org/doc/fr_FR.ISO8859-1/articles/explaining-bsd/index.html



BSD vs Linux

Une courte comparaison

Une des questions les plus fréquemment posées est : En quoi BSD est différent de Linux ? Nous allons essayer de répondre brièvement. Nous admettons préférer utiliser BSD, mais c'est personnel, et basé sur nos expériences. Sans rentrer dans les détails, nous allons essayer d'expliquer les différences en les vulgarisant. Ce flyer est principalement basé sur le travail de Matthew D. Fuller, Greg Lehey et Dru Lavigne.

Qu'est-ce qu'Unix ?

Unix est un système d'exploitation développé à l'origine dans les années 60 au Bell Labs par Ken Thompson et Dennis Ritchie. Unix est sûrement le système d'exploitation qui a le plus influencé l'informatique moderne. Chaque dispositif informatique universel que vous trouverez, et beaucoup de dispositifs informatiques particuliers, utiliseront des idées, des concepts et souvent du code de quelque chose dans l'arbre de famille d'Unix.

Lorsque l'on utilise le terme *Unix*, c'est dans la majorité des cas de manière générique et non pour désigner le SE spécifique du nom d'UNIX™. Sa forme générique signifie plutôt "tout système d'exploitation qui, sous sa forme, son exécution, son interface ainsi que sa style, ressemble en partie au système Unix". Cela comprends tous les BSDs, Linux, Solaris, AIX, HP/UX, et peut-être même une centaine d'autres !



Qu'est-ce que Linux ?

Linux veut dire plusieurs choses. C'est un *noyau*, à l'origine écrit par Linus Torvalds. Linux est également un terme pour désigner une famille de systèmes d'exploitation. On se fiche des débats comme "*Linux n'est pas vraiment un système d'exploitation, c'est juste un noyau*", "*On devrait l'appeler GNU/Linux*", etc... Quand on dit *Linux*, ça signifie Red Hat, Slackware, Debian, Gentoo et les centaines autres distributions basées sur le noyau Linux avec des *espaces utilisateurs* plus ou moins similaires, qui reposent sur des variantes des outils GNU.

Qu'est-ce que BSD ?

BSD vient de *Berkeley Software Distribution*. A l'origine, il a été conçu à l'Université de Californie, Berkeley (CSRG). L'ensemble du code fût immédiatement disponible sous licence BSD, dont la philosophie peut se traduire par "*Faites tout ce qui vous plaira avec notre code, n'oubliez simplement pas de nous citer*". Plus tard, le projet 386BSD débuta et le fit tourner sur les plate-formes Intel. Quand le projet 386BSD déclina, deux principaux groupes se formèrent : FreeBSD et NetBSD. En 1995 OpenBSD se sépara de NetBSD et en 2003 DragonFly BSD se créa à partir de FreeBSD.

Quand nous disons *BSD*, nous parlons de l'organisation générale et de l'approche du système. Précisément, il existe principalement 4 systèmes BSD :

- **FreeBSD** est conçu pour être robuste et efficace sous des architectures Intel et AMD, que ce soit en serveurs ou en station bureautique.
- **NetBSD** est connu pour tourner sur toutes les plate-formes possibles. Son but est d'être l'OS le plus portable de la planète.
- **OpenBSD** se focalise prioritairement sur la sécurité et les points sensibles. L'intégration sécuritaire, l'audit, la cryptographie intégrée et ses dérivées sont ses buts principaux.
- **DragonFly BSD** se fixe comme objectif de fournir une infrastructure SMP, facile à comprendre, à maintenir ainsi qu'à développer.

Mais attention : *tous* les BSD se préoccupent de la sécurité. *Tous* les BSD se préoccupent des performances. Des parties de code sont également échangées entre les groupes. Pour finir, beaucoup de développeurs travaillent sur un même projet pour veiller à sa qualité. Philosophiquement, tous les BSDs sont très similaires, à l'opposé de la méthodologie Linux. Et de toutes façons, c'est de philosophie que parle principalement ce flyer.

Le système de base

Le concept du système de base peut être difficile à comprendre, puisque justement, l'idée même n'existe pas dans le monde de Linux.

Linux, au début, n'était qu'un noyau. Mais un noyau seul n'est pas très utile. On a besoin d'un *espace utilisateur* pour le faire fonctionner. Linux a toujours été une réunion d'outils : un noyau par-ci, un ls par-là, vim, perl, gzip, tar, et un paquet d'autres outils à cet endroit. Linux n'a jamais fait de distinction entre le *système de base* et les *utilitaires additionnels*. Le *système entier* est constitué d'*utilitaires additionnels*.

A l'opposé, BSD a toujours eu un modèle de développement centralisé. BSD n'utilise pas GNU ls ou GNU libc, il utilise BSDs ls and BSDs libc, qui sont des descendants directs du ls et libc qui étaient dans les mises à jour CSRG-distribuées par BSD. Le système est une seule et même pièce, et non une multitude de briques. Cependant le serveur graphique X n'est pas un composant de base de FreeBSD. C'est un paquetage additionnel tout comme les autres applications X telles que KDE, Gnome, Mozilla. La principale différence est l'endroit où ils sont développés. NetBSD et OpenBSD ont quant à eux leur propre implémentation de X dans le système, afin de mieux intégrer leurs propres drivers d'affichages. Ils utilisent tout deux des noyaux fortement modifiés et adaptés, rendant difficile l'implémentation de la couche graphique dans des paquetages séparés.

Le système complet est développé de manière unie. Il existe cependant des parties du système de base comme Sendmail, Bind, SSH ou d'autres qui sont développés par d'autres équipes. Il y a même des paquets comme GCC qui seront immédiatement reconnaissables pour tout linuxien. Mais ces derniers sont spécialement traités, avant d'être intégrés et modelés au reste du système. En réalité beaucoup de ceux-ci sont fait pour s'intégrer à BSD : Bind et Sendmail étaient à l'origine développés en tant que partie de BSD, avant d'être disponibles plus tard, séparément.

La principale raison de cette intégration d'outils externes dans la base est qu'ils concernent le fonctionnement basique d'un système que l'on aimerait avoir par défaut. GCC est les binutils font partis du système de base parce que... eh bien parce qu'ils sont requis pour *construire* ce propre système. Gnome et KDE ne sont pas intégrés dans le système, et ne le seront probablement jamais, puisqu'ils ne sont pas requis pour avoir un système opérationnel et déployable.

La base est uniquement là pour vous permettre d'obtenir un système fonctionnel, vous permettant de le mettre à jour et d'y ajouter d'autres applications. Ensuite, c'est à vous installez ce dont vous avez besoin, suivant le rôle précis de la machine. Ainsi, chaque BSD définit sa propre base. NetBSD et OpenBSD, par exemple, ont des critères plus larges sur l'inclusion dans le système de base (Apache est dans la base d'OpenBSD) que FreeBSD.

Ports/pkgsrc contre RPMs/paquets

Enfin, il y a la seconde partie : les programmes étant des utilitaires additionnels. Dans le monde BSD, on les trouve dans l'*arbre des ports* (FreeBSD et OpenBSD) ou les *pkgsrc* (NetBSD et DragonFly BSD).

Traditionnellement, quand vous avez besoin d'un programme, la première chose à faire est de le compiler. Et dans ces cas-là, on a souvent bien plus d'un souci. Le système voudrait des entêtes différents. Parfois même, on doit réécrire certaines parties du code pour mieux l'intégrer. En d'autres termes, on doit le *porter* vers notre système d'exploitation. Le principe du système de port est d'effectuer automatiquement ce travail de *portage* à notre place. Cette fonctionnalité automatise la construction et l'installation, tout en permettant par la suite une gestion de paquets (pour des choses comme la désinstallation).

La majorité des linuxiens installent des paquets binaires, alors que sous BSD, la compilation depuis les sources est plus appréciée. Cela résulte en partie des outils fournis : le système de ports englobe tout un concept de compilation depuis les sources, avec la possibilité de construire et d'installer des paquets par la suite. Sous Linux, les systèmes comme RPM ou APT sont axés sur l'installation de paquets binaires, avec la construction depuis les sources en dernier recours. Maintenant, il y a des avantages aux binaires pré-compilés : un gain de temps (énorme), ainsi qu'un plus faible espace disque par rapport à la compilation. Mais il y a également des avantages pour la construction depuis les sources, comme un meilleur contrôle des bibliothèques installées, ainsi qu'une meilleure optimisation. Vous pouvez installer des paquets binaires sur Linux ou BSD. Vous pouvez également compiler des sources sur les deux.

Des problèmes peuvent arriver, évidemment. Peut-être qu'une dépendance aura disparu d'un serveur, rendant impossible son installation ? Peut-être qu'une nouvelle version d'un programme-tiers viendra en casser un autre ? Mais disons que le problème du "*je veux A, qui dépend de B qui est absent*" est quand même plus spécifique aux systèmes décentralisés comme RPM.