Module n°1

ADMINISTRATION SYSTEME

Auteur : Labo-Linux
Version 0.1 – 24 octobre 2003
Nombre de pages : 126

Paris Academy of Computer Science SUPINFO Paris Ecole Supérieure d'Informatique

Ecole Supérieure d'Informatique de Paris
23. rue Château Landon 75010 – PARIS
www.supinfo.com

Table des matières

1. Introduction

Pendant longtemps les systèmes issus de la famille des Unix étaient considérés comme réservés à une élite de scientifique ou d'étudiants bidouilleurs, en effet, trop de matériel non supporté, un look définitivement dépassé, et une convivialité presque inexistante.

Encore faut-il avoir réussi à installer un système de cette famille, car le processus n'était pas non plus abordable.

Cependant tout cela a beaucoup changé, Linux peut aujourd'hui tenir tête haute à tous ses concurrents et n'à plus à rougir de son look "vieillot" ou de ses problèmes de compatibilité avec le matériel standard.

Mais malgré toutes ces évolutions, Linux a su garder (est c'est l'une de ses grande force) l'héritage des tous premiers systèmes Unix.

Né de la volonté de Ken Thompson et Dennis Ritchie tous deux de Bell Labs, de faire un système d'exploitation à la fois multitâches et multi utilisateurs, UNIX a vu le jour dans les années 1970 (1969), alors que le projet MULTICS auquel ils avaient participé au début, piétine.

Dans la philosophie qui a accompagné la création d'UNIX, il était question de concevoir de petits programmes ne faisant qu'un nombre limité de tâches, mais le faisant bien. S'est posé ensuite rapidement le moyen de les combiner, d'où la naissance des "pipes" et redirections.

Pour ces utilitaires de l'époque, il était fastidieux de réécrire les programmes en assembleur lors de l'achat de nouvelles machines. C'est dans ce but qu'est né le B puis le C qui offrait plus d'avantages.

Depuis cette période, une quantité astronomique d'Unix ont vu le jour dont, parmi eux des Unix gratuits, comme celui de l'université de Berkeley dont sont dérivés les célèbres OpenBSD, NetBSD, FreeBSD, TrustedBSD, BSDi...

Pour ce qui est de GNU\Linux, il a vu le jour en 1991. Le noyau (cœur du système) a été developpé par un étudiant: **Linus Torvalds**. Il a très rapidement été adapté aux outils GNU (librairie C, compilateur, shell, …) qui fournissent le reste des outils qui font de Linux un système d'exploitation : GNU\Linux.

La plupart des Unix conventionnels sont vendus avec les machines (architectures) dont ils sont propriétaires, la force de Linux reside dans le fait qu'il peut tourner sur des architectures proprietaires, ou bien sur un vieux 486 avec 4Mo de ram.

Parallèlement à cela s'est développé une philosophie nouvelle, pour le partage des connaissances et ce, protégé par une licence qui garantirait l'ouverture des fichiers sources et la possibilité de les modifier (c'était tout le contraire de la philosophie informatique de l'époque) sans devoir des droits à qui que se soit.

C'est la naissance de la GPL ("General Public License") et de la FSF ("Free Software Foundation") tous deux fondés par Richard Stallman.

L'ouverture des sources est en partie à l'origine de l'engouement des logiciels libres, mais est surtout à l'origine d'un support exceptionnel maintenu par des centaines de milliers de passionnés.

Nous allons vous initier à ce monde qui peut paraître un peu hardu au départ, mais dévoile rapidement tous ses charmes.

Si vous vous retrouvez nez à nez avec des erreurs absolument inacceptables, que vous pensez que tout ou une partie de ce cours n'est pas bon, envoyez un email à labo-linux@supinfo.com

Bien sûr nous restons ouvert à toute remarque constructive ou non.

2. Appréhender le système

Malgré une interface graphique très efficace, il est plus que conseillé de bien maîtriser le mode texte. Vous verrez qu'une fois quelques habitudes prises vous ne pourrez plus vous en passer car vous gagnerez énormement en efficatité.

Nous allons donc dans un premier temps nous consacrer à l'utilisation du mode texte.

Remarque : Les terminaux et émulations de terminaux
Tout ce qui va être dit dans cette section est valable aussi bien dans un terminal mode texte (tty1-6) que dans les terminaux graphiques (xterm, Eterm, etc...)

2.1. Les terminaux

En général, lorsqu'on arrive sur un système Unix, la première chose que l'on voit est une invite de commande demandant d'entrer un nom d'utilisateur et un mot de passe. C'est ce que l'on appelle un terminal.

Dans des temps désormais immémoriaux, il y avait un terminal par utilisateur connecté à un serveur au moins équivalent à un 386 SX 33. Maintenant, vous pouvez avoir plusieurs terminaux virtuels sur une machine, permettant ainsi de faire plusieurs choses en même temps. En général on y accède par la combinaison de touche "Alt+F*n*" où F*n* est une touche de fonction.

Nous n'allons pas détailler dans ce qui va suivre comment faire, mais il est possible d'avoir un terminal sur le port série, sur le réseau, ou avec des "émulateurs" de terminaux...
Imaginez que par ce biais, vous pouvez administrer un supercalculateur avec un simple 386.

2.2. Le shell

Le shell est un interpréteur de commandes. C'est lui qui va nous permettre d'exécuter les différents utilitaires que nous allons utiliser dans ce qui va suivre. Une fois connecté au système, vous interagissez avec lui. Il existe beaucoup de shell dont bash (faisant partie de la norme POSIX), zsh, ksh, csh, tcsh, ash...

L'intérêt d'avoir un shell plutôt qu'un autre dépend des fonctionnalités offertes par chacun, et donc de l'utilisation que l'on peut en avoir. Il est intéressant de savoir qu'il dispose d'une syntaxe utile pour la génération de scripts.

Bash (Bourne Again Shell) : offre l'édition de la ligne de commande et le rappel des commandes précédentes. C'est actuellement le shell le plus utilisé car il est très puissant.

Csh (C Shell) : développé à Berkeley, compatible avec le shell Bourne. Pas d'édition de la ligne de commande et historique des commandes.

Ksh (Korn Shell) : offre l'édition de la ligne de commande (touches compatibles Emacs).

Sh : le shell original, pas d'édition de la ligne de commande.

Tcsh : C Shell amélioré.

http://www.zsh.org
http://www.gnu.org/software/bash/bash.html
http://www.kornshell.com
http://www.tcsh.org

2.3. Où trouver de l'aide

Voici une des nombreuses force des systèmes UNIX : la gestion de l'aide. Elle est disponible à la fois sur le système dans des versions plus ou moins détaillées et bien sûr sur Internet.

2.3.1. Au niveau des commandes utilisées

Pratiquement chaque commande dispose d'une aide indiquant son fonctionnement général ainsi qu'un résumer des diverses options qu'il est possible de lui passer. Elle est accessible par les options "-h" ou "--help".

Exemple :

```
$ man --help
man, version 1.5h
usage: man [-adfhktwW] [section] [-M path] [-P pager] [-S list]
[-m system] [-p string] name ...
a : find all matching entries
c : do not use cat file
d : print gobs of debugging information
D : as for -d, but also display the pages
f : same as whatis(1)
h : print this help message
k : same as apropos(1)
K : search for a string in all pages
t : use troff to format pages for printing
w : print location of man page(s) that would be displayed
(if no name given: print directories that would be searched)
W : as for -w, but display filenames only
C file : use file' as configuration file
M path : set search path for manual pages to path'
P pager : use program pager' to display pages
S list : colon separated section list
m system : search for alternate system's man pages
p string : string tells which preprocessors to run
e - [n]eqn(1) p - pic(1) t - tbl(1)
g - grap(1) r - refer(1) v - vgrind(1)
```

2.3.2. Les pages de manuel ou man pages

Elles sont organisées en catégories classées de 1 à 8 dont voici la représentation :

- Commandes utilisateurs (Ex : “man”)
- Appels systèmes (Ex : “socket”)
- Fonctions de la libc (Ex : “printf”)
- Organisation de certains fichiers et protocoles (Ex : “nfs”)
- Jeux et utilitaires sympas sur le système
- Divers (Ex : “fortune”)
- Commandes d’administration

Nous pouvons constater, que les manuels font non seulement référence à des utilitaires et commandes, mais aussi à des fonctions de la libc (librairie C), voir même à d’autres bibliothèques de programmation et des appels systèmes.

C’est un aspect très pratique lorsqu’on développe, pour obtenir les prototypes des fonctions, leurs valeurs de retour, le détail de ce qu’elles font...

L’utilisation des manuels est très simple :

```
man fonction_ou_commande
```

La première chose à faire afin de bien se servir des manuels à notre disposition est **man man**. La disposition des informations au sein des manuels restera la même d’une commande à une autre, et d’une fonction à une autre.

Il peut arriver que plusieurs manuels utilisent le même nom pour y accéder, il suffit alors de préciser en paramètre de la commande `man` la section à rechercher. C’est le cas notamment de la fonction “socket” disponible à la fois dans la section 2 et la section 7 des manuels. Il suffit alors de faire soit man 2 socket ou man 7 socket pour avoir l’aide souhaitée.

Bien sûr cette démarche n’est faisable que lorsqu’on connaît le nom des commandes ou des fonctions.

Comment faire lorsque ce n’est pas le cas ?

Tout simplement en faisant :

```
man -k mot clé
apropos mot clé
whatis mot clé
```

Vous obtiendrez ainsi la liste des manuels contenant le mot clé désigné :

```
$ man -k socket
```

```
QSocketNotifier (3qt) - Support for socket callbacks
accept (2) - accept a connection on a socket
bind (2) - bind a name to a socket
connect (2) - initiate a connection on a socket
fuser (1) - identify processes using files or sockets
getpeername (1) - get information about this or that end of the socket's
connection
getsockname (2) - get socket name
getsockopt, setsockopt (2) - get and set options on sockets
listen (2) - listen for connections on a socket
netpipes (1) - a package to manipulate BSD TCP/IP stream sockets
perlipc (1) - Perl interprocess communication (signals, fifos, pipes, safe
subprocesses, recv, recvfrom, recvmsg (2) - receive a message from a socket
send, sendto, sendmsg (2) - send a message from a socket
sockdown (1) - shutdown(2) a socket
socket (2) - create an endpoint for communication
socket (n) - Open a TCP network connection
socketcall (2) - socket system calls
socketpair (2) - create a pair of connected sockets
socklist (8) - display list of open sockets
```

```
$ whatis socket
socket (2) - create an endpoint for communication
socket (n) - Open a TCP network connection
```

Lorsque vous êtes dans un "man", diverses commandes sont accessibles dont voici les plus intéressantes :

- <ESP> : Avance dans le manuel d'une page entière.
- <ENT> : Avance dans le manuel d'une ligne.
- <b> : Recule dans le manuel d'une page
- / *mot* ou *regexp* : Recherche l'occurrence d'un mot ou d'une expression régulière.
- <q> : Quitte le manuel

Pour avoir la liste complète des commandes accessibles dans un manuel, allez voir dans le manuel de "less" qui est une des commandes que "man" utilise pour éditer le fichier d'aide.

2.3.3. Les fichiers "info"

Il existe encore une autre méthode pour accéder à une aide plus détaillée que les "man pages" : ce sont les "info files".

Ils se trouvent généralement dans les répertoires "/usr/info", "/usr/share/info" ou "/usr/local/info" et sont accessibles par le biais de la commande `info nom fichier`.

De même que pour "*man*" il est possible de retrouver dans quels fichiers "info" trouver les informations souhaitées en utilisant :

```
info --apropos mot-clé
```

```
$ info -k socket
"(libc)Socket Addresses" -- address of socket
"(libc)Socket Addresses" -- binding a socket address
```

```
"(libc)Byte Order" -- byte order conversion, for socket
"(libc)Closing a Socket" -- closing a socket
"(libc)Socket Concepts" -- communication style (of a socket)
"(libc)Connecting" -- connecting a socket
"(libc)Creating a Socket" -- creating a socket
"(libc)Socket Pairs" -- creating a socket pair
"(libc)Socket Concepts" -- loss of data on sockets
"(libc)Socket Addresses" -- name of socket
"(libc)Socket Concepts" -- namespace (of socket)
"(libc)Creating a Socket" -- opening a socket
 […]
"(libc)Setting Address" -- sys/socket.h <13>
"(libc)Address Formats" -- sys/socket.h <14>
"(libc)Communication Styles" -- sys/socket.h
```

Le déplacement au sein d'un fichier info est diffèrent de celui d'un "man".
Les fichiers info se composent de noeuds (titres de parties) et de pages relatives à ces noeuds.

Voici une petite liste de commandes utiles pour se déplacer dans les fichiers info :

- **n** : Se déplacer au noeud suivant.
- **p** : Se déplacer au noeud précédent.
- **u** : Remonter d'un noeud
- **Entrée** : sur un titre avec une étoile, se déplace dans ce noeud
- **Espace** : Descend d'une page dans la page courante

2.3.4. Ressources sur le "Net"

Les "HOW-TO" :

Cette fois nous entrons dans le monde plus spécifique de GNU/Linux. Les "**How-to**" sont des guides et didacticiels sur des domaines bien particuliers (Ex : Comment enregistrer des CD sous Linux).

Ils sont disponibles sous plusieurs formats et sont accessibles sur le site dédié qui est le site officiel du projet LDP ("Linux Documentation Project").

Vous retrouverez notamment sur ce site les "man-pages" à jour traduits dans diverses langues, pour ceux qui n'aiment pas la langue de Shakespeare.

Le "Handbook" de FreeBSD :

Pour ceux qui utilisent FreeBSD, sur leur site dans la section "Handbook", vous trouverez, un peu à la manière des "How-To" des didacticiels sur comment faire telles ou telles tâches.

Documentation pour NetBSD :
NetBSD, dont le but essentiel est d'être portable sur un maximum d'architectures dispose également d'une bonne documentation sur leur site dans la partie "documentation".

3. Les commandes de bases

Nous allons lister ici, une suite de commandes indispensables à l'utilisation de votre machine sous UNIX :

3.1. Opérations sur les fichiers et répertoires

cd *rep*	Se déplace vers le répertoire *rep*
ls *rep*	Liste le contenu d'un répertoire (*-R* pour lister les autres répertoires rencontrés).
cp source destination	Copie un fichier *source* vers un fichier *destination* (*-R* pour un répertoire).
mv source destination	Bouge le contenu d'un fichier ou répertoire *source* vers un fichier ou répertoire de *destination* (utilisé aussi pour renommer un fichier ou répertoire).
mkdir *rep* :	Crée un répertoire.
rm fichier :	Efface un fichier (*-r* pour effacer un répertoire en entier).
ln source destination	Crée un lien *destination* qui va pointer sur *source* (pour un lien symbolique, il suffit d'ajouter l'option "-s")
touch fichier ou repertoire	Met à jour la date de modification du fichier, ou crée un fichier vide si le fichier n'existait pas.

3.2. Lecture de fichier

cat fichier	Lit le contenu du fichier.
more fichier	Lit le contenu d'un fichier page par page. (Il doit lire l'intégralité du fichier avant de l'afficher).
less fichier	Comme "more" sauf qu'il n'est pas obligé de lire l'intégralité du fichier.
tail fichier	N'affiche que les dernières lignes d'un fichier (d'autres options permettent de spécifier combien de lignes afficher).
head fichier	Comme *tail*, mais affiche les N premières lignes d'un fichier (N=10 par défaut).
find répertoire options	Permet de rechercher des fichiers suivant une quantité incroyable de critères.

grep ”chaine” fichier	Recherche l’occurence d’une chaine de caractères ”chaine” dans un ou plusieurs fichiers.

Il y en a bien d’autres, mais à ce stade, il n’est pas encore nécessaire de les voir. Nous les découvrirons au fur et à mesure des thèmes qui vont suivre.

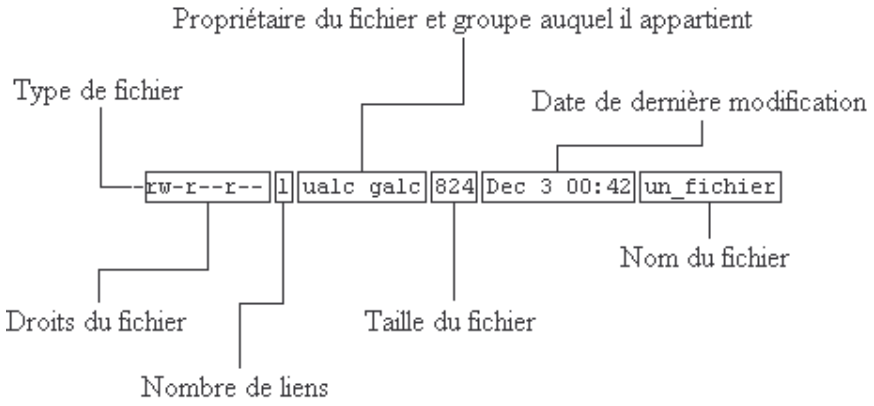
Lorsqu’on liste le contenu d’un répertoire, la différence entre un fichier ne saute pas aux yeux. Pour ce faire, nous allons ajouter l’option *-l* à la commande *ls*.

```
$ cd /tmp
$ mkdir un_repertoire
$ touch un_fichier
$ ln -s un_fichier un_lien

$ ls
un_fichier un_lien un_repertoire

$ ls -l
total 4
-rw-r--r-- 1 ualc galc 0 Dec 3 00:42 un_fichier
lrwxrwxrwx 1 ualc galc 10 Dec 3 00:42 un_lien -> un_fichier
drwxr-xr-x 2 ualc galc 4096 Dec 3 00:42 un_repertoire
```

Il suffit de regarder la partie à l’extrême gauche du terminal. Un fichier ordinaire est symbolisé par un “**-**”, un répertoire par un “**d**” et un lien symbolique par un “**l**”. Pour les autres champs :



3.3. Noms des fichiers et globbing

3.3.1. Noms des fichiers

Les noms de fichiers sont ”case sensitives”, c’est à dire qu’ils tiennent compte des majuscules et des minuscules. Le nom d’un fichier peut contenir jusqu’à 255 caractères.

Pour accéder à un fichier comportant des espaces dans son nom, on doit placer le nom du fichier entre guillemets (ou utiliser des backslash avant l'espace (espace est considéré comme un caractère "spécial") :

```
[mathieu@escaflowne bases unix]$ vi "ceci est un test.txt"
[mathieu@escaflowne bases_unix]$ vi ceci\ est\ un\ test.txt
```

Notons que les fichiers ou répertoires débutant par un "." sont cachés, c'est-à-dire qu'ils ne seront pas visibles avec la commande *ls*. Si on veut voir les fichiers cachés, on passe en argument "-a" à ls :

```
[mathieu@escaflowne ftp]$ ls
Kismet-dump             ether2                  tools
WEP.ppt                 font.tar.gz             tools-prismdump_20001122.tgz
clairtcp.dump           if_wi.c
test.c                  kismet.sh

[mathieu@escaflowne ftp]$ ls -a
.                       .profile                ether2
..                      .rhosts                 font.tar.gz
.cshrc                  .shrc                   if_wi.c
.login                  Kismet.dump             kismet.sh
.login_conf             WEP.ppt                 tools
.mail_aliases           clairtcp.dump           tools-prismdump_20001122.tgz
.mailrc test.c
```

3.3.2. Le globbing

Le globbing est l'utilisation de caractères jokers comme "*", "?".

- **:** correspond à aucun ou plusieurs caractères
- **?** : correspond à un caractère.
- **[a-z]** : correspond à un ensemble de caractères
- **[^ a-z]** : correspond à tous les caractères sauf ceux de cet ensemble

Exemple :

```
[mathieu@battousai Who Made Who]$ ls
ACDC - Who Made Who - Chase the ace.mp3
ACDC - Who Made Who - D.T.mp3
ACDC - Who Made Who - For those about to rock ( we salute you ).mp3
ACDC - Who Made Who - Hells bells.mp3
ACDC - Who Made Who - Ride on.mp3
ACDC - Who Made Who - Shake your foundations.mp3
ACDC - Who Made Who - Sink the pink.mp3
ACDC - Who Made Who - Who Made Who.mp3
ACDC - Who Made Who - You shook me all night long.mp3
bla.html
bonjour.txt
test.c
```

On affiche tous les mp3 :

```
[mathieu@battousai Who Made Who]$ ls *.mp3
ACDC - Who Made Who - Chase the ace.mp3
ACDC - Who Made Who - D.T.mp3
```

```
ACDC - Who Made Who - For those about to rock ( we salute you ).mp3
ACDC - Who Made Who - Hells bells.mp3
ACDC - Who Made Who - Ride on.mp3
ACDC - Who Made Who - Shake your foundations.mp3
ACDC - Who Made Who - Sink the pink.mp3
ACDC - Who Made Who - Who Made Who.mp3
ACDC - Who Made Who - You shook me all night long.mp3
```

On affiche tout sauf les fichiers contenant la chaîne ”mp3”.

```
[mathieu@battousai Who Made Who]$ ls *[^mp3]
bla.html bonjour.txt test.c
```

Exemple avec ”?” :

```
[mathieu@battousai Who Made Who]$ touch tOst.c
[mathieu@battousai Who Made Who]$ ls t?st.c
tOst.c test.c
```

Dernier exemple avec ”*” :

```
[mathieu@battousai Who Made Who]$ ls
ACDC - Who Made Who - Chase the ace.mp3
ACDC - Who Made Who - D.T.mp3
ACDC - Who Made Who - For those about to rock ( we salute you ).mp3
ACDC - Who Made Who - Hells bells.mp3
ACDC - Who Made Who - Ride on.mp3
ACDC - Who Made Who - Shake your foundations.mp3
ACDC - Who Made Who - Sink the pink.mp3
ACDC - Who Made Who - Who Made Who.mp3
ACDC - Who Made Who - You shook me all night long.mp3
bla.html
bonjour.txt
tOst.c
test.c
```

Là on va effacer tous les fichiers comportant ”the” dans le nom de fichier revient à taper :

```
rm ”ACDC - Who Made Who - Chase the ace.mp3” ”ACDC - Who Made Who - Sink
the pink.mp3”
```

```
[mathieu@battousai Who Made Who]$ rm *the*
ACDC - Who Made Who - Chase the ace.mp3
ACDC - Who Made Who - Sink the pink.mp3
```

4. Gestion des utilisateurs et groupes

4.1. Les utilisateurs

Unix est un système multi utilisateurs. C'est à dire que plusieurs personnes peuvent l'utiliser de façon simultanée (dans le cas de configurations en réseau).

Pour le système, un utilisateur n'est pas obligatoirement une personne physique.
Un utilisateur peut détenir des fichiers, exécuter des programmes ou encore déclencher automatiquement des fonctions systèmes.

Par exemple, un utilisateur peut être créé dans le seul but de détenir des fichiers publics. On parle alors de *pseudo utilisateur*.

Un utilisateur possède un *nom d'utilisateur* appelé aussi *login* lui permettant d'établir une connexion. Ce login est associé à un mot de passe personnel.

Les utilisateurs sont identifiés par le système grâce à un *UID* (*identifiant d'utilisateur*) unique. Cet identifiant est une valeur numérique.

4.1.1. Le fichier /etc/passwd

Les informations sur les utilisateurs sont regroupées dans le fichier */etc/passwd*. On y trouve :

- Les logins des utilisateurs
- Les mots de passe cryptés des utilisateurs
- Leur UID
- Leur GID principal
- Les noms complets des utilisateurs
- Le répertoire principal de chaque utilisateur
- Le shell de chaque utilisateur

Ce fichier assure la rémanence des informations concernant les utilisateurs du système.
Sur les anciens systèmes, il était utilisé pour générer une base de données.
La syntaxe de ce fichier est la suivante :

```
nom:mot-de-passe-codé:UID:GID:info-utilisateur:répertoire-principal:shell
```

nom
: Le nom de l'utilisateur, son *login*.

mot de passe codé
: Le mot de passe crypté de l'utilisateur. On utilise la commande *passwd* pour entrer le mot de passe voulu lors de la création du compte ou lors de la modification de son mot de passe.

On peut également remplir ce champ avec le caractère "*****" qui signifie que l'on ne peut pas utiliser ce login pour se connecter au système. De la même manière, si ce champ contient "**!!**" cela signifie que le compte est désactivé. Cette particularité est très pratique lorsque la seule vocation de l'utilisateur est de posséder des fichiers.

Il est également possible que ce champ soit vide. Au quel cas, aucun mot de passe ne sera requis pour se connecter au système avec cette identité (pas sécurisé).

UID

L'identifiant de l'utilisateur. Le super utilisateur (*root*) possède toujours l'UID 0. Par convention, les UID inférieurs à 100 sont réservés aux comptes systèmes. Cette règle n'est pas une obligation et peut varier suivant les distributions.

GID

Le groupe principal de l'utilisateur.

info utilisateur

Le nom complet de l'utilisateur. Il peut être utilisé pour stocker différentes informations séparées par des virgules. Cependant, la confidentialité de ces données n'est pas assurée.

répertoire principal

Le répertoire personnel de l'utilisateur typiquement : "/home/*login*".

shell

Le nom du shell utilisé pour interpréter les commandes de l'utilisateur. Si ce champ est vide, c'est le système qui sélectionne le shell de connexion par défaut.

4.1.2. Les fichiers de mot de passe ombre (shadow)

La présence des mots de passe (sous forme crypté) dans le fichier **"/etc/passwd"** pose des problèmes de sécurité : ce fichier est accessible en lecture par tout le monde ce qui permet à certaines commandes de convertir les noms d'utilisateur en UID et vice-versa.

Pour pallier à ce problème, on utilise un fichier de mot de passe ombre, typiquement **"/etc/shadow"**. Ce fichier n'est accessible que par le super utilisateur.

La plupart des systèmes Unix et Linux supportent cette fonctionnalité. Sous Linux, où les fichiers de mots de passe ombre sont optionnels, il faut installer le logiciel shadow (package **"shadowutils"**) pour l'activer.

Le fichier **"/etc/shadow"** se charge également de conserver les informations relatives au vieillissement des mots de passe.

Il est vivement conseillé d'utiliser les fichiers de mot de passe ombre à chaque fois que c'est possible.

Lorsque cette fonctionnalité est utilisée, le champ réservé au mot de passe est complété avec la valeur "**x**" ou " **!**" suivant le système dans le fichier **"/etc/passwd".**

Dans le cas d'un système comme Linux où cette fonctionnalité est optionnelle, on utilise la commande ***pwconv*** pour créer et mettre à jour le fichier de mot de passe ombre à partir du fichier **"/etc/passwd".**

Cette commande procède de la façon suivante :

1. Le programme récupère toutes les informations nécessaires et empêche temporairement toute modification les concernant durant son exécution.

2. Si le fichier **"/etc/shadow"** n'existe pas, il est créé.

3. Les mots de passe cryptés dans **"/etc/passwd"** sont déplacés dans le fichier **"/etc/shadow".**

4. Le champ réservé au mot de passe de chaque utilisateur est complété par le caractère "**x**" dans le fichier **"/etc/passwd".**

5. Les comptes ne figurant pas dans le fichier **"/etc/passwd"** sont supprimés de **"/etc/shadow".**

Pour revenir à une configuration sans mot de passe ombre, on utilise la commande ***pwunconv*** qui réalise le processus inverse.

Bien que possible, l'ajout d'un grand nombre d'utilisateurs s'avère fastidieux, voire dangereux dans le cas d'une erreur de saisie, en travaillant directement dans le fichier de configuration.

Des commandes ont donc été créées pour simplifier l'ajout d'utilisateur. Elles effectuent également des vérifications, ce qui limite les erreurs de saisie.

4.1.3. Ajouter un utilisateur

La commande utilisée pour créer un nouvel utilisateur ou pour mettre à jour ses informations est *useradd*. La syntaxe de cette commande est la suivante :

```
useradd [options] login
```

Les options qui s'appliquent à cette commande sont les suivantes :

-c commentaire
: Le champ de commentaires de l'utilisateur. Généralement son nom complet.

-d répertoire personnel
: C'est le répertoire personnel de l'utilisateur. Le comportement par défaut est de concaténer le nom de *login* au chemin du répertoire par défaut.

Généralement ”/home/*login*”.

-e date d’expiration
C’est la date à laquelle l’utilisateur sera désactivé. Le format de cette date est *AAAA-MM-JJ*.

-f nombre de jours inactifs
C’est le nombre de jours suivant l’expiration du mot de passe après lequel le compte est désactivé. La valeur 0 permet de désactiver le compte dès que le mot de passe expire. La valeur -1 permet de désactiver cette caractéristique. La valeur par défaut est -1.

-g groupe initial
Le nom du groupe ou le numéro du groupe de connexion initial de l’utilisateur. Le nom ou le numéro du groupe doivent exister. Le numéro de groupe par défaut est 1.

-G listes des groupes
Une liste de groupes supplémentaires auxquels fait partie l’utilisateur. Chaque groupe est séparé du suivant par un virgule, sans espace entre les deux. Le comportement par défaut pour l’utilisateur est de n’appartenir qu’au groupe initial.

-m
Le répertoire personnel de l’utilisateur sera créé s’il n’existe pas déjà. Les fichiers et les répertoires contenus dans le répertoire squelette (”/etc/skel”) seront copiés dans le répertoire personnel si l’option **-k** est employée. L’option **-k** n’est valide qu’en conjonction avec l’option **-m**. Le comportement par défaut est de ne pas créer le répertoire personnel.

-M
Le répertoire personnel de l’utilisateur ne sera pas créé, même si les réglages globaux au système présents dans ”/etc/login.defs” consistent en la création de répertoires personnels.

-n
Un groupe de même nom que l’utilisateur ajouté au système sera créé par défaut. Cette option désactive ce comportement.

-r
Cette option est utilisée pour créer un compte système, c’est à dire un utilisateur avec un UID plus petit que la valeur de l’UID MIN définie dans ”/etc/login.defs”. Si vous souhaitez créer un répertoire personnel pour le nouvel utilisateur, il faut utiliser l’option **-m**.

-p mot de passe crypté
Le mot de passe crypté. Le comportement par défaut est de désactiver le compte.

-s shell
Le nom du shell de connexion de l’utilisateur. Le comportement par défaut est de laisser ce champ vide

-u *uid*
La valeur numérique de l'UID de l'utilisateur. Cette valeur doit être unique, à moins que l'option **-o** ne soit utilisée.

```
$ useradd -D
GROUP=100
HOME=/home/
INACTIV=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

Ceci affiche les valeurs actuelles par défaut. Elles peuvent être modifiées avec les options suivantes :

-b répertoire personnel
Le préfixe de chemin initial pour un nouveau répertoire personnel d'utilisateur.

-e date d'expiration
La date à laquelle le compte utilisateur sera désactivé.

-f inactivité
Le nombre de jours après l'expiration d'un mot de passe avant que le compte ne soit désactivé.

-g groupe par défaut
Le nom de groupe ou l'ID du groupe initial d'un nouvel utilisateur. Le nom et l'ID de groupe doivent exister.

-s shell
Le nom du shell de connexion d'un nouvel utilisateur.

Par exemple : on veut créer un utilisateur korlaz qui a comme commentaire «Le plus fort de tous» qui a comme shell tcsh, qui a un répertoire personnel avec le contenu du répertoire squelette /etc/skel et qui appartient au groupe zik, final et dev. Il faudra aussi que le compte se désactive dès que le mot de passe expirera (rien que ça ;-)) :

```
useradd -c «Le plus fort de tous» -f 0 -G zik,final,dev -m -k /etc/skel -s
/bin/tcsh korlaz
```

4.1.4. Supprimer un utilisateur

La suppression d'un utilisateur de fait avec la commande *userdel*. La syntaxe de cette commande est la suivante :

```
userdel [option] login
```

La seule option est **-r** qui permet d'effacer le répertoire personnel de l'utilisateur.

```
userdel -r toto
```

4.1.5. Changer le mot de passe d'un utilisateur

Chaque utilisateur (au sens physique) devrait normalement posséder un mot de passe même si ce n'est pas obligatoire. Pour changer le mot de passe d'un utilisateur, on utilise la commande *passwd*. Sa syntaxe est la suivante :

```
passwd [options] login
```

Cette commande est généralement utilisée sans option.

```
passwd toto
```

Ceci permet de changer le mot de passe de l'utilisateur toto

Les options qui s'appliquent à cette commande sont les suivantes :

-k
Indique que seul le mot de passe doit être mis à jour sans toucher aux propriétés d'expirations.

-l
Permet de verrouiller le compte spécifié en préfixant le mot de passe crypté par le caractère "!". Seul l'utilisateur root peut utiliser cette option.

--stdin
Le mot de passe doit être lu à partir de l'entrée standard qui peut être un tube.

-u
Déverrouille le mot de passe du compte. Seul l'utilisateur root peut utiliser cette option.

-d
Supprime le mot de passe d'un compte. Le champ réservé au mot de passe crypté sera supprimé dans le fichier de configuration. Seul l'utilisateur root peut utiliser cette option.

-S
Affiche des informations sur le statut du mot de passe pour un compte donné. Seul l'utilisateur root peut utiliser cette option.

4.1.6. Afficher des informations sur les utilisateurs

Pour connaître l'identité de l'utilisateur courant (bien que cela soit affiché dans la majorité des prompts par défaut) on utilise la commande :

```
$ whoami
```

Elle affiche le login de l'utilisateur courant.

Les commandes ***who***, ***users*** et ***w*** permettent de connaître les utilisateurs actuellement connectés sur la machine.

4.2. Les groupes

Un groupe est un ensemble d'utilisateurs qui partagent la même fonction. Par exemple, plusieurs personnes travaillant sur le même projet, peuvent partager certains fichiers.

Tout comme les utilisateurs, les groupes possèdent un identifiant unique : un *GID* (*identifiant de groupe*). Chaque utilisateur possède son propre groupe. Il peut être ajouté à d'autres afin de paramétrer ses droits d'accès aux fichiers et aux ressources du système.

4.2.1. Le fichier /etc/group

Tout comme le fichier "/etc/passwd", le fichier "/etc/group" assure la rémanence des informations concernant les groupes. La syntaxe de ce fichier est la suivante :

```
groupe:*:GID:utilisateurs
```

groupe
Le nom du groupe.

*****
La présence de ce champ est lié aux anciennes versions d'Unix et n'est plus utilisé. Il peut rester vide ou contenir les caractères "*" ou "x".

GID
C'est l'identifiant du groupe représenté par une valeur numérique unique.

utilisateurs
C'est la liste des utilisateurs membres de ce groupe. Les membres sont séparés par des virgules sans espace.

Exemple :

```
dev:x:230:pierre,romain,jerome
```

4.2.2. Afficher des informations sur les groupes

Pour connaître les groupes auxquels appartient un utilisateur, on utilise la commande *groups*. Sans argument, elle affiche les groupes de l'utilisateur courant.
Pour connaître les groupes d'un utilisateur particulier, il suffit de passer son login en argument de la commande :

```
$ groups
root wheel disk adm sys daemon bin
```

```
$ groups toto
toto users fileshare
```

Sur de très anciens systèmes V, il n'était pas possible d'activer plusieurs groupes simultanément pour le même utilisateur.

La commande *id* permet de connaître les groupes actifs :

```
$ id
uid=0(root) gid=0(root) groupes=0(root),
10(wheel),6(disk),4(adm),3(sys),2(daemon),1(bin)
```

Pour changer le groupe actif sur un tel système, on utilise la commande *newgrp*. Lorsqu'elle est utilisée sans argument, elle active le groupe principal de l'utilisateur (le groupe qui figure dans "/etc/passwd").

4.2.3. Créer un nouveau groupe

La création d'un nouveau groupe se fait grâce à la commande :

```
groupadd nom_du_groupe
```

Elle crée un nouveau groupe qu'elle ajoute au fichier "/etc/group". Les options de la commande *groupadd* sont les suivantes :

-g *gid*
: Permet de choisir la valeur numérique du GID du nouveau groupe. Cet identifiant doit être unique.

-r
: Cette option permet d'ajouter un groupe système (dont le GID est inférieur à 500).

-f
: Permet de stopper la commande lorsque le groupe ou le GID du nouveau groupe existe déjà.

4.2.4. Suppression d'un groupe

Pour supprimer un groupe, on utilise la commande :

```
groupdel nom_du_groupe
```

On ne peut pas supprimer le groupe primaire d'un utilisateur.

4.2.5. Modifier les groupes secondaires d'un compte

La modification des groupes secondaires d'un utilisateur se fait avec la commande ***usermod.*** ***usermod*** est similaire à ***useradd*** et supporte les mêmes options.

Exemple :

```
usermod -G toto,users,fileshare,dev toto
```

Ceci permet d'ajouter l'utilisateur toto dans les groupes toto, users, fileshare et dev

4.3. Changer d'identité

Le changement d'identité est assuré par la commande *su*.

Exemples :

```
su toto
```

Vous permet d'agir en tant qu'utilisateur *toto*. Cela signifie que vous avez les droits de l'utilisateur *toto*.

```
su - toto
```

Vous permet de vous connecter en tant qu'utilisateur *toto*. Notez la présence du – qui permet d'utiliser l'environnement de l'utilisateur *toto*.

L'option **-c** vous permet de lancer une seule commande avec les droits du nouvel utilisateur :

```
su -c "mount /mnt/cdrom"
```

Lance la commande ***mount /mnt/cdrom*** en tant que *root* qui est le login par défaut lorsque celui-ci n'est pas spécifié.

Dans tout les cas, le système vous invite à saisir le mot de passe correspondant.

4.4. Interfaces de gestion des utilisateurs et des groupes

De nombreuses interfaces comme *userdrake* (pour la distribution Mandrake) permettent la gestion des utilisateurs en mode graphique.

Cependant, les possibilités offertes par ce type d'outil sont bien souvent plus limitées que les programmes en ligne de commande et ne vous permettent pas de créer des scripts pour automatiser ces taches.