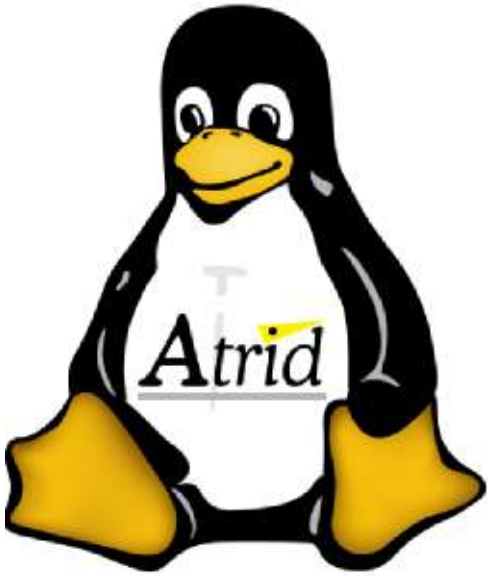
COURS LINUX - INSTALLATION ET ADMINISTRATION



ATRID

COURS LINUX - INSTALLATION ET ADMINISTRATION

par ATRID

Historique des versions

Version 1.0 du25/08/1998
Version initiale
Version 1.1 du04/12/1998
Ajout outils graphiques
Version 1.2 du19/02/1999

Corrections présentation
Version 1.3 du09/07/1999
Corrections, adaptation Mandrake 6.0
Version 1.4 du16/11/1999
Relecture Thierry STOEHR
Version 1.5 du22/05/2000
Passage en LateX et corrections
Version 1.6 du18/10/2000
Passage en SGML et corrections

Table des matières

Introduction **7**
1. Présentation de Linux 7
2. Caractéristiques techniques de Linux 7
3. Linux et les distributions 7
4. De ce support de cours 8
5. Copyrights 8
1. Installation **9**
2. Documentations **10**
2.1. Documentation en ligne 10
2.1.1. Le manuel 10
2.1.2. Texinfo 11
2.1.3. Les HOWTO 11
2.1.4. Les documentations 11
2.1.5. Les guides du Linux Documentation Project (LDP) : 11
2.2. Autres sources d'information 11
2.2.1. Les sites Web 11
2.2.2. les forums 12
2.2.3. Les livres 12
3. Organisation des fichiers **13**
3.1. Le répertoire `/etc` 13
3.2. Le répertoire `/home` 14
3.3. Le répertoire `/usr` 14
3.4. Le répertoire `/var` 14
4. Les commandes Linux **16**
4.1. Principes 16
4.2. Manipulation des fichiers 16
4.3. Filtre en mode texte 17
4.4. Gestion des processus 18
4.5. Commandes diverses 18
4.6. L'interpréteur de commandes 19
4.6.1. Le lancement 19
4.6.2. La ligne de commandes 20
4.6.3. Les variables de l'interpréteur 21
4.6.4. Un éditeur de texte : vi 22
4.6.5. La programmation 24
4.6.5.1. Le premier programme 24
4.6.5.2. Les variables prédéfinies 24
4.6.5.3. L'exécution conditionnelle 25
4.6.5.4. Les boucles 27
4.6.5.5. Les fonctions 28
4.6.6. Gestion de processus 29

5. Utilisateurs, processus et fichiers **30**
5.1. Utilisateurs 30
5.2. Processus 31
5.3. Fichiers 32
5.3.1. Types de fichiers 32
5.3.2. Attributs des fichiers 32
5.3.3. Droits d'accès 34
6. Démarrage et arrêt **36**
6.1. Le démarrage du noyau 36
6.2. Le processus init 36
6.2.1. Le fichier `/etc/inittab` 36
6.2.2. Le répertoire `/etc/rc.d` 38
6.3. L'arrêt du système 39
7. Gestion des périphériques **40**
7.1. Les fichiers spéciaux 40
7.2. Les disques durs 41
7.3. Imprimantes 43
7.3.1. Les commandes d'impression 43
7.3.2. Le fichier `/etc/printcap` 44
7.3.3. Les filtres 45
7.4. Sauvegardes 45
7.4.1. Les commandes **dump** et **restore** 46
7.4.2. La commande **tar** 46
7.4.3. La commande **cpio** 47
7.4.4. La commande **dd** 48
7.4.5. La commande **mt** 48
7.4.6. Recherche avec **find** 49
8. Automatisation **51**
8.1. Le fichier crontab 51
8.2. La commande **crontab** 51
8.3. La commande **at** 52
9. Le noyau **53**
9.1. La génération d'un noyau 53
9.2. LILO 54
9.3. Les modules 55
10. Suivi et traces **56**
10.1. L'utilisation du disque 56
10.1.1. Les commandes de suivi 56
10.1.2. Gestion des quotas 56
10.2. Les processus et la mémoire 57
10.3. Les fichiers de trace 58
10.4. Le système de fichiers /proc 59

11. Réseaux 61
11.1. Configuration 61
11.2. Les commandes 62
11.3. Les fichiers 62
11.4. Applications réseau 63
11.4.1. DNS 63
11.4.2. SaMBa 63
11.4.3. SENDMAIL 63
11.4.4. NFS 63
11.4.5. NIS 64
12. Xwindow 65
13. Sécurité 66
13.1. Généralités 66
13.2. Les services systèmes 66
13.3. Les firewalls 66
13.4. Les outils 66
13.4.1. SATAN 67
13.4.2. TRIPWIRE 67
13.4.3. SAINT 67
13.4.4. COPS 67
14. Outils graphiques d'administration 68
14.1. Les outils Red Hat 68
14.2. Linuxconf 68
14.3. Webmin 68
15. Gestion des packages 70
15.1. Fonctionnement des packages RPM 70
15.2. Commande RPM 70
15.3. Outils graphiques 71
15.4. Commande rpmfind 72
16. En cas de problèmes sur le système 73
16.1. Mode Single User 73
16.2. Disquette de rescue 73
16.3. Demolinux 73
16.4. Réparer son système 74
16.4.1. Systèmes de fichiers endommagés 74
16.4.2. Mot de passe root oublié 74
16.4.3. Restaurer une sauvegarde 74

Introduction

1. Présentation de Linux

Linux est un système d'exploitation libre, réalisant un sous-ensemble de la norme POSIX. Initialement, le terme "Linux" désigne uniquement le noyau. Par extension, on donne également ce nom aux distributions basées sur ce noyau et un ensemble d'outils du projet GNU.

Linux est un système Unix, mais n'utilise aucun code propriétaire et est fourni sous licence GPL, ce qui implique la disponibilité des sources. Comme tout autre système Unix, il est multi-tâches, multi-utilisateurs. Il est également extrêmement portable, puisqu'il est aujourd'hui disponible officiellement sur des processeurs Intel (i386 à Pentium III), Alpha, Motorola (680x0 et PowerPC), Sparc, StrongArm, Mips. Sans compter les portages opérationnels ou en cours sur PalmPilot, Itanium (ex-Merced), PA-Risc, Crusoe ...

Les version du noyau Linux sont de la forme "x.y.z". Si "y" est un nombre pair, la version est stable et seules des corrections d'anomalies y sont en général appliquée lors de l'incrément de "z". Si "y" est un nombre impair, il s'agit d'une version de développement et donc instable.

2. Caractéristiques techniques de Linux

Linux dispose des caractéristiques suivantes :

- Multi-tâches : exécute plusieurs programmes en parallèle.
- Multi-utilisateurs : plusieurs utilisateurs peuvent travailler simultanément sur la même machine.
- Intéropérable : supporte les systèmes de fichiers : System V, BSD, Sun, MS-DOS, VFAT, NTFS, Mac, HPFS, EFS, ISO9660 et les protocoles réseau TCP/IP v4 et v6, Appletalk, Netware (client et serveur), SMB (client et serveur).
- Conforme aux standards : Posix, avec les extensions Système V et BSD. Prise en charge des binaires COFF et ELF. Compatibilité binaire avec SCO, SVR3/4 par le module iBCS2.

3. Linux et les distributions

Le terme Linux ne désigne que le noyau, qui n'est pas suffisant pour obtenir un système d'exploitation fonctionnel. Il faut y ajouter les fonctions de base (manipulation de fichiers, d'utilisateurs...) ainsi que les logiciels qui vont assurer certains services (serveur web, serveur de messagerie...). Afin de simplifier l'installation de ces différents outils, de nombreux éditeurs ont mis en place des distributions, contenant à la fois le noyau Linux et tous les outils nécessaires à son fonctionnement.

Les distributions diffèrent par le choix des logiciels proposés (en dehors dans grands standards incontournables), de la configuration par défaut, du format de packages logiciels, et éventuellement par l'ajout d'outils propres à la distribution.

Parmi les principales distributions GNU/Linux, on peut citer :

- RedHat (http://www.redhat.com)
- Debian (http://www.debian.org)
- Mandrake (http://www.linux-mandrake.com)
- Slackware (http://www.slackware.org)
- Suse (http://www.suse.com)

4. De ce support de cours

Les informations données ici sont basées sur les distributions Redhat6.0 et Mandrake 6.0 de Linux. Les autres distributions de Linux partagent la plupart des concepts évoqués ici, mais peuvent apporter des modifications mineures concernant la localisation des programmes ou les outils mis en oeuvre.

Linux étant un système vivant, il y a beaucoup de chances que les informations données ici changent au cours des prochaines évolutions. Néanmoins, la plupart des concepts présentés viennent du monde Unix et sont standardisés voire normalisés par les normes Posix.

5. Copyrights

- Linux est une marque déposée de Linus Torvalds.
- Windows, Windows NT, Windows 95 sont~ des marques déposées de Microsoft Corporation.
- Linux-Mandrake est une marque déposée de la société MandrakeSoft.
- Red Hat est une marque déposée de la société Red Hat Software.
- UNIX est une marque déposée de The Open Group.

Chapitre 1. Installation

L'installation de la distribution Mandrake 6.0 s'effectue en suivant la procédure décrite dans le guide d'installation. Il est possible de sélectionner différentes méthodes d'installation :

- le cédérom local connecté à la machine
- par NFS en se connectant à une image du cédérom exportée par une machine distante
- à partir d'un serveur FTP
- par une image exportée via SaMBa
- à partir de fichiers copiés sur le disque dur local

Les points importants à vérifier sont :

- le découpage du disque dur en partitions (voir le chapitre Gestion des périphériques)
- les périphériques installés dans la machine ; il faut avoir les informations de configuration de ces périphériques (adresse, IRQ).
- les paramètres de connexion au réseau

Il est important de réaliser une disquette de démarrage qui sera utilisée en cas de problème sur le disque dur.

Chapitre 2. Documentations

2.1. Documentation en ligne

Il existe plusieurs types d'aide en ligne :

- le manuel de référence
- les "HOWTO"
- les documents du "Linux Documentation Project", alias LDP
- les fichiers de documentation des paquetages

2.1.1. Le manuel

Le manuel en ligne est accessible par la commande **man**. Il est organisé en différentes sections :

Section	Intitulé
1	les commandes utilisateur
2	les appels système
3	les bibliothèques de programmation
4	les fichiers spéciaux
5	les formats de fichiers
6	les jeux
7	divers
8	les commandes d'administration
9	le noyau

La syntaxe de la commande **man** est :

```
man {[}options{]} {[}section{]} commande
```

Par défaut, les sections sont parcourues dans l'ordre des numéros. Il est possible de préciser le numéro de section pour résoudre les problèmes de synonymes :

ex : **man kill** et **man 2 kill**

La variable d'environnement MANPATH donne le chemin de recherche des pages de manuel. Par exemple :

```
MANPATH=/usr/man:/usr/local/man
```

Il existe une base de mots clés pour la recherche dans les pages de manuel. Cette base est construite par la commande **/usr/sbin/makewhatis**. Les commandes **apropos** et **whatis** permettent d'effectuer des recherches dans cette base de données.

2.1.2. Texinfo

C'est un projet de la FSF (Free Software Foundation) pour remplacer les pages de manuel classiques. La base de données d'informations est répartie et accessible par la commande **info**. Cette commande permet de visualiser en mode interactif des documents hypertexte. Le système est auto renseigné, il suffit de lancer info et de naviguer dans cet environnement pour avoir l'aide en ligne.

2.1.3. Les HOWTO

Les HOWTO sont des documents expliquant comment effectuer une tâche donnée comme : "Installer Linux sur le même disque que Windows NT". Ils sont disponibles dans plusieurs formats (texte, Postscript, HTML) et sont installés dans le répertoire /usr/doc/HOWTO. Ce sont les premiers documents à consulter lorsqu'on recherche une information pouvant s'exprimer par " Comment faire pour ? ". Il existe des traductions françaises de ces documents sur (http ://www.freenix.fr/linux/HOWTO/)

2.1.4. Les documentations

A partir du répertoire /usr/doc, on trouve les documentations des paquetages installés. Le contenu est assez inégal et dépend du concepteur du paquetage. C'est néanmoins une ressource importante en terme de documentation.

2.1.5. Les guides du Linux Documentation Project (LDP) :

- "Installation and getting started guide"
- "The Linux kernel hackers' guide"
- "The Linux Kernel"
- "The Linux Programmers guide"
- "The Linux Kernel Module Programming Guide"
- "The Linux Network Administrators' Guide"
- "The Linux Users Guide"
- "The Linux System Administrators' Guide"

Comme pour les HOWTO, certains de ces guides sont traduits en français.

2.2. Autres sources d'information

2.2.1. Les sites Web

Certains sites web proposent de la documentation en ligne ou téléchargeable sur le système Linux :

- Linux Center : (http://linux-center.org/fr/)
- Unix Guru Universe : (http://www.ugu.com/)
- LDP : (http://sunsite.unc.edu/mdw/linux.html/)
- Le guide du ROOTard : (http://www.freenix.fr/linux/Guide)

D'autres sites proposent des nouvelles concernant Linux et les logiciels libres en général. Ce sont de bonnes sources pour se tenir informé des nouvelles versions de logiciels :

- Freshmeat : (http://www.freshmeat.net)
- Da Linux French Page : (http://linuxfr.org/)

2.2.2. les forums

Un grand nombre de forums existe traitant en totalité ou partie de Linux. Les plus importants et les plus fréquentés sont :

- fr.comp.os.linux.configuration
- fr.comp.os.linux.annonce
- fr.comp.os.linux.moderated
- comp.os.linux.answers
- comp.os.linux.security
- comp.os.linux.setup

Les forums disposent de leur moteur de recherche : dejanews, disponible sur http ://www.deja.com/usenet, qui permet d'effectuer une recherche sur la totalité des forums, ou sur certains d'entre eux uniquement grâce aux fonctions *power search* (http ://www.deja.com/home_ps.shtml).

2.2.3. Les livres

Les ouvrages de la collection O'Reilly sont généralement bien écrits et de bon niveau technique car réalisés par des spécialistes du domaine concerné : (http ://www.editions-oreilly.fr/)

Chapitre 3. Organisation des fichiers

Ce chapitre présente l'organisation du système de fichiers et le rôle de chacun des répertoires

Il n'existe pas de norme d'organisation du système de fichiers, mais un standard est à peu près suivi par les différentes distributions de Linux.

L'organisation traditionnelle du répertoire racine est décrite dans le tableau suivante :

Répertoire	contient
/bin	les fichiers exécutables nécessaires à l'initialisation
/boot	le noyau et les fichiers de démarrage
/dev	les fichiers spéciaux
/etc	les fichiers de configuration du système et certains scripts
/home	la base des répertoires utilisateurs
/lib	les librairies système et les modules
/lost+found	le stockage des fichiers retrouvés par fsck
/mnt	les points d'ancrage des systèmes extérieurs
/proc	un système de fichiers virtuels permettant l'accès aux variables du noyau
/root	le répertoire de base du super utilisateur
/sbin	les fichiers exécutables pour l'administration du système
/tmp	les fichiers temporaires
/usr	les programmes, les librairies et les fichiers accessibles pour l'utilisateur
/var	les données variables liées à la machine (spool, traces)

Le répertoire de base `/` s'appelle : répertoire racine (*root*) par analogie avec la racine d'un arbre représentant le système de fichiers.

Il n'est pas standard d'ajouter des répertoires au niveau de la racine.

Ce système de fichiers peut résider sur différentes partitions, différents supports physiques ou sur d'autres machines sur le réseau. Ce découpage est complètement transparent pour les utilisateurs du système de fichiers. Les différentes parties peuvent être connectées au démarrage du système ou à la demande, en cours d'utilisation.

3.1. Le répertoire `/etc`

Ce répertoire contient les fichiers de configuration du système. On trouve les sous-répertoires suivants :

Répertoire	Contient
./X11	les fichiers de configuration de Xwindow
./rc.d	les scripts de démarrage du système
./logrotate.d	les fichiers de configuration de logrotate pour les paquetages

Répertoire	Contient
./cron	les tâches à effectuer à la périodicité donnée (daily, hourly, monthly, weekly)
./skel	les fichiers à recopier dans le répertoire d'un nouvel utilisateur
./sysconfig	les fichiers de configuration des périphériques

3.2. Le répertoire `/home`

Le répertoire `/home` contient les répertoires des utilisateurs logés sur cette machine. Chaque utilisateur possède son propre répertoire, mais il est possible de créer des structures de groupes de travail en organisant les sous-répertoires.

3.3. Le répertoire `/usr`

Le répertoire `/usr` contient de nombreux sous-répertoires. On retrouve presque la même organisation que sous la racine, mais le contenu est destiné aux utilisateurs plus qu'au système lui-même.

La structure de `/usr` est la suivante :

Répertoire	Contient
./X11R6	la hiérarchie des fichiers Xwindow (version 11 révision 6)
./bin	les commandes du système
./doc	les documentations en ligne
./etc	la configuration des commandes utilisateurs
./games	les jeux
./include	les fichiers de définition des librairies pour la programmation en C
./lib	les librairies non système
./local	la hiérarchie des fichiers propres à cette installation
./man	les fichiers des pages du manuel en ligne
./sbin	les commandes d'administration non nécessaires au démarrage
./share	les fichiers de configuration partagés
./src	les sources du système et des applications

Xwindow est l'environnement graphique d'Unix. Il est très flexible et configurable et possède des fonctionnalités rarement vues dans d'autres environnements. On retrouve, dans ce répertoire, une hiérarchie de fichiers ressemblant à celle de la racine, mais dédiée à l'environnement Xwindow.

`/usr/local` doit contenir les outils installés en dehors du contexte de la distribution. On retrouve une hiérarchie complète semblable à `/usr`.

3.4. Le répertoire `/var`

Le répertoire `/var` contient les données variables du système, c'est-à-dire les fichiers propres à l'installation réalisée sur cette machine.

Répertoire	Contient
./catman	les fichiers d'aide mis en forme
./lib	quelques fichiers de configuration
./lock	les fichiers de verrous des applications
./log	les fichiers d'enregistrement des traces
./run	les fichiers contenant les "pid" des processus du système
./spool	les fichiers du spooler et de la messagerie

Le répertoire `catman` contient les pages de manuel mises en forme pour un accès plus rapide lors d'une deuxième utilisation.

Le répertoire `log` contient les fichiers de trace de fonctionnement du système. Une grande partie du travail d'administration consiste à suivre les enregistrements afin de détecter les mauvais fonctionnements. Le programme logrotate permet de conserver un historique des fichiers. Il existe des outils de gestion des fichiers de trace pour permettre, entre autres, la détection des intrusions sur le système.

Le répertoire `spool` contient des sous-répertoires de gestion des spoolers d'impression (`lpd`), de courriers (`mail`), de forums (`news`), etc. Ces sous-répertoires peuvent contenir, momentanément, des fichiers de taille importante.

Chapitre 4. Les commandes Linux

4.1. Principes

Utiliser Linux consiste à se connecter sur la machine et à passer les commandes nécessaires pour accomplir la tâche que l'on s'est fixée. Il existe deux environnements de travail différents sous Linux :

- le mode texte, où l'on saisit les commandes au clavier et où le résultat s'affiche sous forme de lignes de texte sur un écran
- l'environnement Xwindow ; il s'agit d'un environnement multi-fenêtres en réseau avec des applications commandées par une souris.

Nous allons voir les commandes utilisateur principales de Linux (il en existe plusieurs centaines). Les commandes sont des programmes écrits avec n'importe quel langage informatique pour résoudre un problème donné. Le système est évolutif et il est facile d'ajouter de nouvelles commandes. Toutefois, avant d'écrire une commande, il faut vérifier qu'il n'existe pas quelqu'un ayant déjà effectué ce travail par une recherche sur le système ou sur le Web. Une commande Linux est toujours composée de la même façon :

```
nom_commande {[}options{]} liste_de_paramètres
```

Les options sont généralement composées du signe '-' et d'un caractère ou d'un mot. Elles modifient, ou précisent, le fonctionnement de la commande. La liste des paramètres permet de spécifier la cible de la commande. Les options et les paramètres sont séparés par des caractères blancs (espace ou tabulation). Tous les programmes Linux sont lancés avec trois fichiers ouverts : l'entrée standard (stdin), la sortie standard (stdout) et la sortie des erreurs (stderr). Par défaut, l'entrée standard est le clavier et les sorties sont dirigées sur l'écran. Il est possible de rediriger l'entrée ou la sortie vers un fichier ou vers une autre commande. Ceci permet d'enchaîner des traitements sur un flot de données.

signe	fonction
<	permet de rediriger l'entrée standard depuis un fichier
>	permet de rediriger la sortie standard vers un fichier
2>	permet de rediriger la sortie des erreurs vers un fichier
2>&1	permet de rediriger la sortie des erreurs vers la sortie standard
|	permet de rediriger la sortie standard dans l'entrée d'une autre commande

Sachant que les périphériques sont vus par Linux comme des fichiers (par l'intermédiaire des fichiers spéciaux), cela permet de créer des filtres puissants pouvant gérer des fichiers ou des entrées / sorties.

4.2. Manipulation des fichiers

Commande	Description
pwd	affiche le chemin du répertoire courant
cd	change le répertoire courant (commande interne du shell)
basename	extrait le nom du fichier d'un chemin complet
chmod	modifie les droits d'un fichier
chgrp	change le groupe propriétaire du fichier
chown	change l'utilisateur propriétaire du fichier
cp	copie de fichiers
ln	affiche la liste des fichiers d'un répertoire
mkdir	crée des répertoires
mknod	crée un nom de fichier temporaire unique
rm	détruit des fichiers
rmdir	détruit des répertoires
mv	déplace des fichiers
touch	met à jour les dates d'accès des fichiers
df	affiche la place disque disponible
du	donne la place disque utilisée par le répertoire courant
file	donne le type de fichier
mtools	ensemble d'outils pour la gestion des disquettes MS-DOS
mdir	affiche la liste des fichiers d'une disquette MS-DOS

4.3. Filtre en mode texte

Commande	Description
cat	concatène les fichiers sur la sortie standard
sort	trie les lignes de texte en entrée
sed	effectue des modifications sur les lignes de texte
more	affiche à l'écran l'entrée standard en mode page par page
less	affiche à l'écran l'entrée standard en mode page par page avec des possibilités de retour en arrière
cut	permet d'isoler des colonnes dans un fichier
expand	transforme les tabulations en espaces
head	affiche les n premières lignes

Commande	Description
join	permet de joindre les lignes de deux fichiers en fonction d'un champ commun
paste	concatène les lignes des fichiers
tail	affiche les n dernières lignes du fichier
tac	concatène les fichiers en inversant l'ordre des lignes
uniq	élimine les doublons d'un fichier trié
rev	inverse l'ordre des lignes d'un fichier
pr	formate un fichier pour l'impression
tee	écrit l'entrée standard sur la sortie standard et dans un fichier
tr	remplace ou efface des caractères

4.4. Gestion des processus

Commande	Description
ps	affiche la liste des processus
kill	envoie un signal à un processus
nice	fixe la priorité d'un processus

4.5. Commandes diverses

Commande	Description
true	ne fait rien sans erreur
false	ne fait rien, avec une erreur
echo	affiche une ligne de texte
date	affiche et modifie la date courante
gzip	compresse les fichiers
sleep	attend pendant le temps spécifié
grep	recherche de chaînes de caractères dans des fichiers
find	recherche des fichiers sur le système
bc	calculatrice en mode texte
cal	affiche le calendrier
clear	efface l'écran
csplit	découpe un fichier en sections
diff	permet d'obtenir la différence entre deux fichiers

Commande	Description
factor	recherche les facteurs premiers d'un nombre
hexdump	affiche le contenu d'un fichier en hexadécimal
id	affiche l'identification d'un utilisateur
passwd	permet de changer le mot de passe
wc	compte les caractères, les mots et les lignes en entrée
whereis	permet de trouver l'emplacement d'une commande
who	affiche la liste des utilisateurs présents

4.6. L'interpréteur de commandes

Le shell est l'interpréteur de commandes en mode texte du système Linux lancé à chaque ouverture de session. Ce n'est pas le seul moyen de passer des commandes, il existe un environnement graphique avec des outils interactifs, mais c'est le plus pratique pour l'administrateur.

Le shell permet :

- l'exécution des commandes
- la redirection des entrées et des sorties
- la substitution des noms de fichiers
- la gestion des variables d'environnement
- la possibilité de réaliser des scripts

Il existe de nombreux shells avec des caractéristiques différentes ; celui livré en standard dans les distributions Linux est bash (Bourne Again SHell). Les différences se trouvent surtout dans la syntaxe du langage de scripts et dans la gestion des variables d'environnement.

4.6.1. Le lancement

En fonction du mode de lancement, l'interpréteur de commandes utilise différents fichiers d'initialisation.

On distingue deux modes de démarrage de l'interpréteur de commandes, qui peut être un shell de démarrage (login) ou un shell interactif.

Pour le shell de démarrage, l'ordre de lecture des fichiers de démarrage est :

- le fichier `/etc/profile`
- le fichier `$HOME/.bash_profile` s'il existe, sinon le fichier `$HOME/.bash_login` sinon le fichier `$HOME/.profile`

Lorsqu'il se termine, il exécute les commandes du fichier `$HOME/.bash_logout` s'il existe.

Pour un shell interactif, le fichier `$HOME/.bashrc` est exécuté.

4.6.2. La ligne de commandes

Le shell interprète les commandes en gérant des caractères spéciaux. La signification de ceux-ci est indiquée dans cette table :

Caractères	Signification
nouvelle ligne (0xa)	fin de la ligne de commande
espaces	les caractères blancs (espaces et tabulations) séparent les arguments de la ligne de commande
‘ ou “	permettent de bloquer l’interprétation des caractères spéciaux
\	caractère d’échappement
&	la commande sera exécutée en arrière plan
< > << >> | ‘	redirection des entrées et des sorties
* ? [ ] [^]	caractères de substitution des noms de fichiers
$	départ d’un nom de variable
;	séparateur de commandes

Les caractères de substitution des noms de fichiers sont interprétés par le shell pour générer les listes de noms de fichiers correspondant aux critères :

Caractères	Signification
*	zéro ou plusieurs caractères
?	un caractère
[ ]	n’importe lequel des caractères entre les crochets
[^]	n’importe quel caractère n’apparaissant pas dans les crochets

Les caractères d’échappement permettent d’enlever la signification spéciale des caractères lorsqu’ils doivent être utilisés comme paramètres comme indiqué dans la table suivante :

Caractères	Signification
‘	guillemet simple : permet d’ignorer tous les caractères spéciaux entre les guillemets
“	guillemet double : comme ci-dessus sauf pour les caractères ‘, $, \.
\	permet d’ignorer le caractère spécial suivant

La redirection des entrées et des sorties consiste à envoyer l’entrée, ou la sortie d’une commande depuis, ou vers un fichier ou une autre commande. Cette fonction est gérée par le shell grâce au fonctionnement interne du

système pour la gestion des descripteurs de fichiers dans les programmes.

Les possibilités sont :

Commande	Signification
commande > fichier	envoie la sortie standard de la commande dans le fichier
commande < fichier	lit l'entrée standard à partir du fichier
commande >> fichier	envoie la sortie standard de la commande dans le fichier en ajoutant les lignes à la fin du fichier
commande << libellé	lit l'entrée standard jusqu'au libellé
commande1 | commande2	envoie la sortie standard de commande1 dans l'entrée standard de commande2
commande n>&m	envoie la sortie du file descripteur n dans le descripteur de fichier m

4.6.3. Les variables de l'interpréteur

L'interpréteur de commandes gère des variables dans lesquelles l'utilisateur enregistre des valeurs pour pouvoir les utiliser par la suite. Dans l'environnement de chaque utilisateur, il existe quelques variables définies par le système (tout cela est configurable) et utilisées par le shell lui-même ou par les processus en exécution. On trouve entre autres :

- le chemin du répertoire de base de l'utilisateur : `$HOME`
- la liste de recherche des commandes : `$PATH`
- l'invite de l'interpréteur : `$PS1`
- le nom de l'utilisateur : `$USER`
- le type de shell : `$SHELL`

La commande **set** permet de visualiser les variables et leurs valeurs.

La commande **expr** permet d'effectuer des opérations arithmétiques entières sur les variables.

La commande **export** permet de faire connaître la variable aux processus fils du shell.

On initialise une variable en affectant une valeur :

```
bash$ VARIABLE='valeur de la variable'
```

On accède au contenu de la variable en précédant son nom du signe '$' :

```
bash$ echo $VARIABLE
valeur de la variable
```

Il existe des méthodes d'accès plus sophistiquées permettant une substitution de valeur comme le montre le tableau ci-après :

Méthode	Résultat
${variable:-valeur{	donne la valeur de la variable si celle-ci n'est pas nulle, sinon donne la valeur de substitution
${variable:=valeur{	comme ci-dessus et la valeur de substitution est affectée à la variable
${variable:?message{	donne la valeur de la variable si celle-ci n'est pas nulle, sinon affiche le message de substitution
${variable:+valeur{	donne la valeur de substitution si la variable n'est pas nulle, sinon ne donne rien

4.6.4. Un éditeur de texte : vi

Vi est un éditeur de texte en mode écran qu'il faut absolument connaître en tant qu'administrateur car c'est souvent le seul disponible sur les machines. Il est peu convivial mais extrêmement puissant de part la rapidité de ses commandes.

Le lancement de l'éditeur s'effectue par :

```
vi [fichier]
```

L'éditeur fonctionne avec deux modes principaux :

- un mode commande pour se déplacer dans le texte et passer les commandes de type ex.
- un mode saisie de texte dont on sort en appuyant sur <ESC>.

Les commandes de déplacement sur le texte sont nombreuses :

Commande	Fonction
h	déplace le curseur d'un caractère vers la gauche
l	déplace le curseur d'un caractère vers la droite
k	déplace le curseur d'une ligne vers le haut
j	déplace le curseur d'une ligne vers le bas
w	déplace le curseur au début du mot suivant
b	déplace le curseur au début du mot précédent
W	déplace le curseur au début du mot suivant séparé par des blancs
B	déplace le curseur au début du mot précédent séparé par des blancs
e	déplace le curseur à la fin du mot

Commande	Fonction
E	déplace le curseur à la fin du mot séparé par des blancs
<CTRL>-f	affiche la page suivante
<CTRL>-b	affiche la page précédente
<CTRL>-d	affiche la demi-page suivante
<CTRL>-u	affiche la demi-page précédente

Voici les quelques commandes nécessaires pour saisir et modifier du texte :

Commande	Fonction
i	passe en mode insertion avant le caractère courant
a	passe en mode ajout après le caractère courant
o	ajoute une ligne après la ligne courante et passe en insertion en début de ligne
O	ajoute une ligne avant la ligne courante et passe en insertion en début de ligne

Les commandes d'effacement de texte placent le texte détruit dans un tampon en mémoire. Ceci permet d'effectuer des annulations de modifications ainsi que des déplacements de texte.

Commande	Fonction
x	détruit le caractère courant
dd	détruit la ligne courante
y	place le caractère courant dans le tampon
yy ou Y	copie la ligne entière dans le tampon
P	copie le contenu du tampon avant le curseur
p	copie le contenu du tampon après le curseur
u	annule la dernière modification apportée au texte

Les commandes de manipulation de fichiers permettent la lecture, l'écriture et la concaténation de fichiers :

Commande	Fonction
:w	écrit dans le fichier en cours
:w nom	écrit dans le fichier “ nom ”
:q	quitte l'éditeur
:wq	sauve et quitte
:e nom	lit le fichier “ nom ”
:e #	lit le fichier précédent
:n	lit le fichier suivant sur la ligne de commande

4.6.5. La programmation

L'écriture de programmes d'enchaînement de commandes, les scripts, permet à l'administrateur de simplifier la réalisation de tâches répétitives. Un script est un fichier texte contenant les commandes à exécuter ainsi que, éventuellement, des structures de contrôle, des boucles, des assignations de variables, des fonctions et des commentaires.

4.6.5.1. Le premier programme

Le script donné ci-dessous permet de sacrifier à la coutume du programme " print Hello " que l'on doit écrire dans tout nouveau langage :

```
#!/bin/sh
# fichier : bonjour.sh
# Affiche un salut a l'utilisateur
echo "Bonjour $USER"
```

Les premières lignes sont des lignes de commentaires qui doivent commencer par un caractère "#" en première colonne. La première ligne permet de fournir au noyau la commande à lancer pour exécuter le script. Il faut rendre le fichier texte exécutable à l'aide de la commande **chmod**.

```
bash$ chmod +x bonjour.sh
bash$ ./bonjour.sh
Bonjour sandra
bash$
```

La commande **set -x** est utile pour la mise au point des programmes car elle permet de visualiser les commandes exécutées au fur et à mesure de l'avancement.

4.6.5.2. Les variables prédéfinies

Il existe plusieurs variables prédéfinies utilisables dans les scripts :

Variable	Fonction
$$	l'identificateur du processus courant (pid)
$?	la valeur de sortie de la commande précédente
$!	l'identificateur du processus fils
$0	le nom du script
$1 à $9	les neuf premiers arguments du script
$#	le nombre d'arguments
$*	tous les arguments à partir de `$1 séparés par un espace`

Les variables `$1` à `$9` contiennent les neuf premiers arguments du script. Pour accéder aux autres, il faut utiliser

la commande **shift**. Celle-ci décale les arguments d'un cran vers la gauche : `$2` est mis dans `$1`, `$3` dans `$2`, etc. La variable `$9` est initialisée avec le dixième argument.

4.6.5.3. L'exécution conditionnelle

L'exécution conditionnelle est gérée par la structure :

```
if commande
then
        autres commandes
fi
```

if teste le code de sortie de la commande en paramètre. Si celui-ci vaut 0 (vrai), le script exécute les commandes entre then et fi.

Il est possible d'avoir des structures du type :

```
if commande
then
         commandes
else
         commandes
fi
```

ou :

```
if commande
then
        commandes
elif commande
        commandes
fi
```

Il existe deux commandes utiles pour les tests, ce sont true qui retourne toujours 0, donc vrai, et false qui retourne toujours 1, donc faux.

Pour permettre des tests complexes mettant en ?uvre plusieurs commandes, il faut utiliser les opérateurs && et || qui effectuent un ET logique et un OU logique sur le résultat des commandes. Ces opérateurs sont utilisés de la façon suivante :

```
commande1 && commande2
commande1 || commande2
```

Dans le cas du ET logique, la deuxième commande ne sera exécutée que si la première réussit. Dans le cas du OU logique, la deuxième commande ne sera exécutée que si la première échoue.

La commande **if** n'effectue pas de tests autres que le résultat de la commande en paramètre. Pour effectuer ces tests, il existe une commande **test** qui peut aussi s'écrire **[**. Cette commande évalue ces arguments et retourne vrai ou faux en fonction du résultat de cette évaluation.

Les options les plus courantes sont :

Expression	Vraie si
-e fichier	le fichier existe
-f fichier	le fichier existe et est un fichier standard
-d fichier	le fichier existe et est un répertoire
-r fichier	le fichier est lisible
-w fichier	le fichier est accessible en écriture
-x fichier	le fichier est exécutable
fichier1 -nt fichier2	le fichier 1 est plus récent que le fichier 2
-z chaîne	la chaîne a une longueur nulle
chaîne1 = chaîne2	la chaîne 1 est égale à la chaîne 2
chaîne1 != chaîne2	la chaîne 1 est différente de la chaîne 2
chaîne	la chaîne n'est pas nulle
entier1 -eq entier2	l'entier entier1 est égal à l'entier entier2
entier1 -ne entier2	l'entier entier1 n'est pas égal à l'entier entier2
entier1 -lt entier2	l'entier entier1 est inférieur à l'entier entier
entier1 -gt entier2	l'entier entier1 est supérieur à l'entier entier2
entier1 -le entier2	l'entier entier1 est inférieur ou égal à l'entier entier2
entier1 -ge entier2	l'entier entier1 est supérieur ou égal à l'entier entier2
expr1 -a expr2	les deux expressions sont vraies
expr1 -o expr2	l'une des deux expressions est vraie
! expr	l'expression est fausse

Par exemple on peut avoir :

```
if [ $# -lt 2 ]
then
        echo "Nombre d'arguments trop petit"
fi
```

Qui affichera le message d'erreur si le nombre d'arguments passé est inférieur à deux.

L'autre commande d'exécution conditionnelle est la commande **case** qui permet de comparer une valeur à plusieurs expressions et d'exécuter les commandes lorsqu'il y a égalité. La syntaxe de cette commande est :

```
case valeur in
expression1) commande
                        autre commande ;;
expression2) commande
                        autre commande ;;
esac
```

Si une expression correspond à la valeur, les commandes sont exécutées et `$ ?` est mis à 0. Sinon `$ ?` est mis à 1.

Par exemple, le code suivant teste la réponse d'un utilisateur :

```
echo -n "Votre reponse :"
read REPONSE
case $REPONSE in
O* | o*) REPONSE="OUI" ;;
N* | n*) REPONSE="NON" ;;
*) REPONSE="PEUT-ETRE !" ;;
esac
```

4.6.5.4. Les boucles

L'interpréteur de commandes gère trois types de constructions pour les boucles :

```
while commande
do
commandes
done
ou
until commande
do
commandes
done
```

ou

```
for variable in liste_de_valeurs
do
commandes
done
```

Pour les deux premières constructions, le fonctionnement est similaire à la commande **if**. Pour la boucle **for**, la variable prendra successivement toutes les valeurs de la liste.

Les commandes **break** et continue permettent de modifier le fonctionnement des boucles. La commande **break** est utilisée sous deux formes :

break : sort de la boucle courante

break n : n étant un entier supérieur à 0, permet de sortir de n boucles imbriquées.

La commande **continue** permet de passer à l'itération suivante sans exécuter les commandes situées après.

Redirection et structures de contrôle

Il est possible de rediriger la sortie ou l'entrée d'une structure de contrôle globalement.

Par exemple, le programme suivant lit les lignes à partir d'un fichier dont le nom est dans la variable $fichier et imprime celles dont la longueur est supérieure à deux caractères dans un fichier temporaire.

```
while read ligne
do
        longueur=`echo $ligne | wc -c`
        if [ $longueur -lt 2 ]
        then
                continue
        fi
        echo $ligne
done < $fichier > tmp.$$
```

4.6.5.5. Les fonctions

Le shell permet de définir des fonctions appelables par le script. Cette méthode est plus efficace que d'appeler des scripts externes car la fonction est exécutée dans le même processus.

La syntaxe de définition d'une fonction est :

```
fonction()
{
        commandes
{
```

et elle est appelée par :

```
fonction [paramètres]
```

Les fonctions reçoivent les paramètres d'appel dans les variables `$1` à `$9` et `$*`. La valeur de `$0` reste inchangée.

Il est possible de déclarer des variables locales à une fonction en utilisant la commande **local** avant la déclaration de la variable. Dans ce cas, la modification de la variable locale ne modifie pas une variable globale portant le même nom.

Par exemple :

```
MaFonction()
{
        local maVariable="coucou"
        echo $maVariable
{
maVariable="Bonjour"
echo $maVariable
MaFonction
echo $maVariable
```

La commande **return** permet de spécifier le code de retour de la fonction. S'il n'est pas spécifié, la variable `$?` est initialisée au code de fin de la dernière commande de la fonction.

Le shell supporte les appels récursifs de fonctions. Comme dans tout langage récursif, il faut faire très attention aux effets de bord et aux variables non déclarées en local.

4.6.6. Gestion de processus

L'interpréteur de commandes permet l'exécution de processus en tâche de fond. Ceux-ci sont lancés en ajoutant le signe & à la fin de la ligne de commande. La variable `$!` reçoit l'identificateur du processus fils.

La commande **wait** permet d'attendre la fin de l'exécution d'un processus fils. Il est possible de spécifier en paramètre l'identificateur du processus fils que l'on désire attendre. La valeur de retour de wait est le code de sortie du processus fils.

La commande **trap** permet de gérer les signaux envoyés au script et d'exécuter une commande sur réception d'un signal. La syntaxe de la commande est :

```
trap commande signaux
```

Comme pour un programme en langage C, il existe des signaux qui ne sont pas gérables.

Chapitre 5. Utilisateurs, processus et fichiers

Un processus est un programme en exécution. Pour tout travail, Linux utilise des processus, le noyau n'étant là que pour gérer ces processus et leur permettre d'interagir avec le monde extérieur. Dès son lancement, le noyau lance un premier processus (init) qui est l'ancêtre de tous les processus lancés sur le système par la suite (appels systèmes **fork** et **exec**). Le système maintient une table des processus en fonctionnement avec tous les attributs qui les caractérisent.

Toute l'information maintenue par le système est enregistrée dans des fichiers organisés dans une structure hiérarchique de répertoires. Comme pour les processus, chaque fichier possède des attributs qui peuvent être modifiés par les utilisateurs par l'intermédiaire des commandes du système.

Linux étant un système multi-utilisateurs, la plupart des attributs des processus et des fichiers déterminent les droits des utilisateurs à les utiliser ou à les modifier. Ces droits sont associés à une identification de chaque utilisateur pour le système.

5.1. Utilisateurs

Chaque utilisateur qui utilise le système doit être connu de celui-ci par un nom et, éventuellement, un mot de passe. Un utilisateur doit appartenir à un ou plusieurs groupes d'utilisateurs pour être autorisé à utiliser le système. Il existe plusieurs méthodes d'identification et de contrôle des utilisateurs, nous ne parlerons ici que de la méthode la plus simple mettant en oeuvre les fichiers `/etc/passwd` et `/etc/group`.

Les utilisateurs et les groupes sont repérés dans le système par des numéros : uid pour le numéro d'utilisateur (User IDentifier) et gid pour le numéro de groupe (Group IDentifier). Le numéro est unique pour un utilisateur ou un groupe donné.

L'identification d'un utilisateur s'effectue dans le fichier `/etc/passwd`. Ce fichier est constitué de lignes décrivant tous les utilisateurs connus de la machine.

Par exemple, un utilisateur est représenté par une ligne du type :

```
sandra:*:500:500:Sandra BULLOCK:/home/sandra:/bin/bash
```

Cette ligne comporte les champs suivants séparés par le caractère ' :' :

- nom de l'utilisateur pour l'identification sur le système (logging)
- mot de passe crypté
- numéro d'utilisateur (uid)
- numéro de groupe d'utilisateur par défaut (gid)
- nom complet ; ce champ peut contenir de nombreuses informations dont le format est dépendant de l'outil qui les interprète
- répertoire de départ de l'utilisateur
- programme à lancer au démarrage, généralement un interpréteur de commande (shell)

Il est possible de définir plusieurs utilisateurs ayant le même uid. Ceux-ci auront les mêmes droits d'accès, mais pourront bénéficier de mots de passe, de répertoires de départ ou d'interpréteur de commandes différents.

Le fichier de déclaration des groupes `/etc/group` contient une ligne par groupe dans un format similaire au fichier `/etc/passwd`.

Une ligne de ce fichier comporte les champs suivants, séparés par des caractères ' :' :

- nom du groupe
- mot de passe du groupe
- numéro du groupe (gid)
- liste des utilisateurs appartenant au groupe séparés par des virgules

Par exemple :

```
actrices:*:400:sandra,meg,michelle
```

La commande **id** permet d'afficher les numéros d'utilisateur et de groupes d'un utilisateur. Sans argument, elle affiche les numéros de l'utilisateur courant. Par exemple :

```
bash$ id
uid=500 (sandra) gid=500 (sandra) groups=500(sandra),100(users), 400(actrices)
bash$ id meg
uid=501 (meg) gid=500 (meg) groups=500(meg),100(users),400 (actrices)
```

Le premier groupe donné dans la liste est le groupe courant avec lequel sont effectués les tests de droits d'accès et les créations de fichiers et de processus.

La commande **newgrp** permet à un utilisateur de changer son groupe courant. Par exemple :

```
bash$ newgrp actrices
bash$ id
uid=500 (sandra) gid=400 (actrices) groups=500(sandra),100(users), 400(actrices)
```

La commande **useradd** permet d'ajouter un utilisateur sans avoir à éditer les fichiers. Les paramètres de la ligne de commande permettent de fixer les différents attributs. Cette commande crée le répertoire de travail et copie les fichiers de base dans celui-ci. Les commandes **usermod** et **userdel** permettent respectivement de modifier un compte utilisateur et de le détruire.

De la même façon, les commandes **groupadd**, **groupmod** et **groupdel** permettent de créer, modifier et détruire les groupes d'utilisateurs.

5.2. Processus

Tous les processus (sauf le premier init) sont créés par un autre processus par les appels systèmes **fork** et éventuellement **exec**. Cela introduit une hiérarchie de parenté entre les processus, le processus créé étant l'enfant du processus l'ayant créé (processus parent). Le processus fils hérite de nombreux attributs de son processus père mais peut les modifier lors de son exécution.

Pour chaque processus en exécution, Linux maintient un grand nombre d'informations dont les paramètres d'identification (uid et gid) pour les vérifications de droits d'accès.

Ces paramètres sont :

- uid et gid réels : c'est l'identification de l'utilisateur exécutant le processus. Ils sont utilisés pour la comptabilisation des temps.
- uid et gid effectifs : ce sont ces paramètres qui déterminent ce que le processus peut faire. Dans la plupart des cas, ils sont identiques aux identificateurs réels mais peuvent être forcés à une valeur donnée par modification des attributs du fichier programme.
- Linux utilise un autre couple uid et gid pour le test d'accès aux fichiers. Normalement, ces tests sont effectués avec les uid et gid effectifs, mais des programmes comme le serveur NFS utilisent ces paramètres pour éviter des trous de sécurité potentiels.

5.3. Fichiers

Sous Linux, toutes les informations sont enregistrées et lues dans des fichiers. Les répertoires sont des fichiers d'un type particulier et les périphériques sont accédés par l'intermédiaire de fichiers spéciaux.

5.3.1. Types de fichiers

Les différents types de fichiers sous Linux sont :

- les fichiers normaux : ce sont des collections d'octets. Il n'existe pas de différence entre les fichiers texte et les fichiers binaires.
- les répertoires : ce sont des fichiers contenant les noms des fichiers et leurs numéros d'inode.
- les liens symboliques : permettent de présenter une image d'un fichier sous un autre nom ou à un autre endroit sans dupliquer les données.
- les fichiers spéciaux en mode bloc : sont les portes sur les périphériques fonctionnant par blocs de données (ex : disques).
- les fichiers spéciaux en mode caractère : sont les portes vers les périphériques fournissant ou consommant les données octet par octet.
- les tubes nommés "fifo" : permettent à deux processus sans relation de parenté de s'échanger des données comme par un tube.

5.3.2. Attributs des fichiers

Pour afficher les attributs principaux des fichiers, il faut utiliser l'option `'-l'` de la commande **ls** :

```
-rw-r--r--    2    root   root   6656   Apr 15  1998   fichier
prw-r--r--    1    root   root    0     Apr 15  1998   fifo
brw-r--r--    1    root   root    0     Apr 15  1998   bloc
crw-r--r--    1    root   root    0     Apr 15  1998   caracteres
drwxr-xr-x    1     root   root   1024   Nov 12 19:42   répertoire
```

Cet affichage présente beaucoup d'informations :

- le premier caractère donne le type du fichier :
  - `'-'` pour un fichier normal
  - `'p'` pour un fifo
  - `'b'` pour un fichier spécial en mode bloc
  - `'c'` pour un fichier spécial en mode caractère
  - `'d'` pour un répertoire
- les neuf caractères suivants donnent les droits d'accès (voir plus loin)
- le champ suivant donne le nombre de liens sur le fichier
- on trouve ensuite le nom du propriétaire et du groupe du fichier
- le champ suivant donne la taille en octets du fichier
- la date de la dernière modification est indiquée selon deux formats :
  - avec l'année pour les fichiers vieux de plus de 6 mois ou de plus d'une heure dans le futur
  - avec l'heure pour les autres cas
  - enfin, le nom du fichier.

La commande **stat** permet d'afficher plus d'informations sur un fichier.

```
bash$ stat file1
  File: "file1"
  Size: 3562         Filetype: Regular File
  Mode: (0777/-rwxrwxrwx)    Uid: ( 500/ sandra)  Gid: ( 500/ sandra)
Device:  8,0   Inode: 2043      Links: 1
Access: Wed Nov 18 18:52:42 1997(00000.00:26:18)
Modify: Wed Nov 18 18:52:42 1997(00000.00:26:18)
Change: Wed Nov 18 18:52:59 1997(00000.00:26:01)
```

En plus des attributs déjà vus, cette commande affiche :

- le type du fichier en toutes lettres
- les droits d'accès en numérique (voir plus loin)
- la référence du périphérique physique (device)
- le numéro d'inode
- les dates de dernier accès, dernière modification et dernier changement.

5.3.3. Droits d'accès

Les neuf caractères donnant les droits d'accès s'interprètent par groupe de trois :

- le premier groupe de trois caractères donne les droits pour le propriétaire
- le deuxième groupe de trois caractères donne les droits pour les utilisateurs du groupe
- le dernier groupe donne les droits pour les autres utilisateurs

Dans un groupe, la signification des caractères est donnée, dans l'ordre, par :

- 'r' pour autoriser la lecture, '-' pour l'interdire
- 'w' pour autoriser l'écriture, '-' pour l'interdire
- 'x' pour autoriser l'exécution, '-' pour l'interdire

Par exemple, un fichier avec les droits `rwxr-x--x` peut être :

- lu, écrit et exécuté par le propriétaire
- lu et exécuté par les membres du groupe
- exécuté par les autres

La signification des droits est différente en fonction du type de fichier.

Pour un fichier :

- `'r'` permet de lire le contenu du fichier
- `'w'` permet de modifier le contenu du fichier
- `'x'` permet d'exécuter un programme

Pour un répertoire :

- `'r'` permet d'afficher la liste des fichiers
- `'w'` permet de créer et de détruire des fichiers
- `'x'` permet d'accéder aux fichiers ou d'en faire le répertoire courant

Il existe deux symboles supplémentaires, `'s'` et `'t'`, pouvant prendre la place du 'x' dans la liste des droits. Ces symboles signifient :

- `'s'` : dans le cas d'un fichier exécutable, celui-ci sera exécuté avec les droits du propriétaire ou du groupe en fonction de la place du symbole. Dans le cas d'un répertoire, tous les fichiers créés dans ce répertoire appartiendront au même groupe que celui du répertoire en question.
- `'t'` (*sticky bit*) : pour les fichiers exécutables, demande de garder le code en mémoire après l'exécution. Pour les répertoires, permet de limiter la destruction des fichiers au propriétaire du répertoire, du fichier ou au super utilisateur

Ces droits d'accès peuvent s'exprimer sous forme d'un nombre en octal composé en fonction des masques de bits donnés dans le tableau ci-après :

Symbole	Valeur numérique	
r	400	utilisateur
	40	groupe
	4	autres
w	200	utilisateur
	20	groupe
	2	autres
x	100	utilisateur
	10	groupe
	1	autres
s	4000	utilisateur
	2000	groupe
t	1000	

La commande **chmod** permet de modifier les droits d'un ou plusieurs fichiers en utilisant le mode numérique ou le mode littéral.

La commande **umask** permet de fixer les droits qui seront enlevés par défaut pour la création de nouveaux fichiers. Cette commande est généralement lancée une fois lors de l'ouverture de session, elle est intégrée à l'interpréteur de commandes et sa syntaxe varie en fonction de celui que l'on utilise. Il est conseillé de positionner le masque en numérique pour s'affranchir des différences syntaxiques.

Les commandes **chown** et **chgrp** permettent de changer, respectivement le propriétaire et le groupe d'un fichier.

Chapitre 6. Démarrage et arrêt

Nous allons décrire la procédure de démarrage et d'arrêt d'un système Linux sur un PC. Cette procédure se décompose en plusieurs parties :

- le BIOS charge le programme de démarrage
- celui-ci lit et démarre le noyau
- le noyau lance le processus init
- ce dernier initialise les autres processus.

Il est possible de démarrer LINUX à partir de différents média et à partir de différents programmes de démarrage. Le chargeur standard de Linux est LILO (LInux LOader) et il peut être installé :

- sur le secteur de démarrage d'une disquette
- sur le secteur de démarrage du disque (MBR, Master Boot Record)
- sur le secteur de démarrage de la partition système

Dans le dernier cas, il doit être activé par le chargeur principal (*primary loader*)

6.1. Le démarrage du noyau

En fonction de sa configuration, le BIOS va chercher le programme de démarrage (*primary loader*) sur le premier secteur d'une disquette ou d'un disque dur (MBR *Master Boot Record*) et le charge en mémoire. Ensuite le programme de démarrage de second niveau (*secondary loader*) est chargé puis invoqué. Sur l'écran, le programme de démarrage affiche l'invite `LILO :`. Le premier `L` est écrit avant de charger le deuxième programme et le `I` est écrit avant de l'exécuter. Le deuxième `L` signifie que le chargeur secondaire s'exécute mais qu'il y a une erreur d'accès au disque. Dès que le `O` s'affiche, le système peut être démarré.

A l'invite de LILO, l'utilisateur peut entrer des commandes pour spécifier le noyau à démarrer, des paramètres pour un périphérique ou pour démarrer en mono-utilisateur (`single`).

Dès que le noyau est chargé et décompressé, il prend la main et s'initialise. Il réserve de la mémoire pour son utilisation, initialise les périphériques dont il a besoin, attache la zone d'échange (*swap*) et monte la racine du système de fichier.

Le compte-rendu de l'initialisation du système est donné dans le fichier `/var/log/dmesg`.

6.2. Le processus init

Ce programme est le premier processus lancé par le noyau. Il est chargé de démarrer les processus systèmes et d'en relancer certains lorsqu'ils se terminent, et ce durant la totalité du fonctionnement du système.

Sa configuration s'effectue dans le fichier `/etc/inittab`.

6.2.1. Le fichier `/etc/inittab`

Ce fichier contient des lignes respectant le format suivant :

```
code:niveau:action:commande
```

Le champ code contient une séquence de un à quatre caractères (deux pour compatibilité) unique pour identifier la ligne dans le fichier.

Le champ niveau donne le niveau d'exécution pour lequel cette ligne doit être prise en compte. La notion de niveau d'exécution permet de spécifier des configurations d'exécution différentes. Un standard existe et est résumé dans la table ci-dessous. Il est possible de spécifier plusieurs niveaux lorsque la commande associée doit être lancée à différents niveaux d'exécution.

Niveau	Description
0	Arrêt de la machine
1	mode mono-utilisateur, seul le super-utilisateur peut se connecter
2	mode multi-utilisateurs, avec peu de services réseaux
3	mode multi-utilisateurs, avec tous les services réseaux (mode par défaut)
4	définissable par l'utilisateur
5	démarrage de X11 au boot
6	redémarrage de la machine

Le champ action définit la manière d'exécuter la commande du champ commande.

Le tableau ci-après présente les actions les plus courantes :

Action	Description
respawn	relance la commande lorsqu'elle se termine
wait	attend la fin de la commande avant de continuer
once	la commande est exécutée une fois
boot	la commande est exécutée au démarrage du système (le champ niveau est ignoré)
bootwait	comme ci-dessus avec attente
off	ne rien faire (permet de conserver la ligne pour une utilisation future)
initdefault	permet de spécifier le niveau d'exécution par défaut
sysinit	la commande est exécutée au démarrage avant celles des directives boot et bootwait
ctrlaltdel	la commande est exécutée lorsque l'utilisateur tape les trois caractères <CTRL>-<ALT>-<SUPPR> sur le clavier
powerfail	la commande est exécutée lorsque le processus init reçoit le signal SIGPWR (défaut d'alimentation)

6.2.2. Le répertoire `/etc/rc.d`

Ce répertoire contient les scripts utilisés pour l'initialisation du système. Ils sont prévus pour démarrer les différents services et processus et effectuer quelques vérifications de configuration.

La table suivante présente les différents scripts :

Niveau	Description
rc.sysinit	exécuté une fois au démarrage pour initialiser le système
rc	script de gestion du niveau d'exécution. Il le reçoit en paramètre.
rc.local	script utilisé pour les initialisations particulières à la machine
init.d	répertoire contenant les scripts d'initialisation des sous-systèmes
rc0.d, rc1.d, rc2.d, rc3.d, rc4.d, rc5.d et rc6.d	répertoires contenant des liens sur les scripts du répertoire init.d devant être lancés à un niveau d'exécution particulier

Le fichier `rc.sysinit` réalise les opérations suivantes :

- initialise la variable `PATH` pour les autres scripts
- active la partition de swap
- initialise le nom du système (hostname)
- vérifie l'intégrité du système de fichiers
- démarre la gestion des quotas
- initialise le "Plug and Play"
- prépare la gestion des modules
- initialise l'horloge système
- détruit les fichiers de verrouillage

Le fichier `rc` exécute les scripts du répertoire `rcN.d` où `N` correspond au niveau d'exécution. Ces scripts sont des liens symboliques sur les fichiers de démarrage des sous-systèmes du répertoire `init.d`. Le lien reprend le nom du fichier d'origine précédé de la lettre `S` et d'un nombre pour les scripts de démarrage ou de `K` et d'un nombre pour les scripts d'arrêt du sous-système. La valeur numérique permet de spécifier l'ordre d'exécution des scripts.

Les scripts du répertoire `init.d`. peuvent être appelés par la suite pour forcer un arrêt ou un démarrage d'un sous-système. La syntaxe utilisée est :

```
/etc/rc.d/init.d/script [start | stop | restart]
```

La plupart des démons du système enregistrent leur numéro de processus dans un fichier du répertoire `/var/run`.

Après l'exécution des scripts, le processus init lance les processus en mode `respawn`. Ce sont, en général, les programmes d'invitation à l'ouverture de session. Ces programmes doivent être relancés dès qu'une session se termine.

6.3. L'arrêt du système

Le système Linux doit être arrêté par une commande système et non en coupant l'alimentation secteur directement. Un arrêt brutal du système peut causer des pertes de données dues aux applications en fonctionnement.

La procédure d'arrêt permet :

- d'avertir les utilisateurs que le système doit être arrêté (commencer quelques jours avant)
- de demander aux applications de s'arrêter et de fermer les connexions et les fichiers ouverts
- de passer le système en mode mono-utilisateur
- de vider les tampons mémoire du cache disque

Le système garde une trace du fait qu'il est démarré pour permettre une vérification d'intégrité dans le cas d'un arrêt brutal.

La commande d'arrêt du système est la commande **shutdown** qui permet, selon les options utilisées :

- de donner l'heure de l'arrêt (*`now`*, *`hh:mm`*, *`+minutes`*)
- de donner le mode arrêt (arrêt ou redémarrage)

Chapitre 7. Gestion des périphériques

Les périphériques sont gérés par le système par grâce à des pilotes intégrés au noyau soit de façon fixe, soit sous forme de modules. Les pilotes sont accessibles par les fichiers spéciaux contenus dans le répertoire `/dev`. A chaque périphérique physique du système est attribué un ou plusieurs fichiers spéciaux.

7.1. Les fichiers spéciaux

Le répertoire `/dev` contient un nombre important de fichiers :

Fichier spécial	Description
mem	accès à la mémoire physique
kmem	accès à la mémoire du noyau
null	périphérique vide
port	accès aux ports d'entrées/sorties
mouse	gestion de la souris (peut être un lien sur le fichier effectif)
tty0 à tty__	les terminaux virtuels (de 0 à 63)
ttyS0 à ttyS_	les ports séries
pty[p-s][0-9a-f]	pseudos terminaux maîtres
tty[p-s][0-9a-f]	pseudos terminaux esclaves
lp0, lp1, lp2	ports parallèles
js0 et js1	port joystick
fd___	les lecteurs de disquettes (fd0 est le lecteur standard)
hd__	les disques durs et les cédéroms IDE
sd__	les disques durs SCSI
scd__	les cédéroms SCSI
st__ et nst__	les lecteurs de bandes SCSI

La commande **ls -l** permet d'afficher les attributs importants d'un fichier spécial.

```
brw-rw----   1 root     disk    3, 0 May  5  1998 /dev/hda
```

Le premier caractère est `b` pour les périphériques en mode bloc et `c` pour les périphériques en mode caractères.

La taille du fichier étant par définition nulle, elle est remplacée dans l'affichage par deux entiers donnant des informations sur le pilote et sur le périphérique :

- le premier est appelé "major number"; c'est un index dans la table des pilotes de périphériques
- le deuxième est appelé "minor number" et permet de spécifier au pilote, les caractéristique du périphérique

connecté.

La commande **/dev/MAKEDEV** permet de créer les fichiers spéciaux standard. Elle est exécutée en donnant en paramètre le nom du périphérique pour lequel on veut un fichier spécial. Ce nom doit appartenir à la liste des noms gérés par cette commande.

La commande **mknod** permet de créer un fichier spécial en précisant tous ses attributs. La syntaxe est :

```
mknod nom type major minor
```

7.2. Les disques durs

Les disques durs sont virtuellement découpés en zones distinctes appelées "partitions". Il peut y avoir jusqu'à quatre partitions primaires sur un disque dur. Si plus de partitions sont nécessaires, il faut utiliser une des partitions primaires pour créer une partition étendue qui contiendra des partitions logiques. L'outil de gestion des partitions est **fdisk**.

Un découpage minimum consiste à définir une partition pour le système de fichiers et une autre pour la zone d'échange (swap).

Le disque complet est vu par un fichier spécial (ex : `/dev/hda`) et chaque partition est vue par un fichier indépendant (ex : `/dev/hda1`, `/dev/hda2`, etc.).

La commande **mkfs** permet de créer un système de fichiers sur une partition. Cette commande permet aussi d'effectuer une vérification de la surface du disque.

Linux utilise un système de fichiers virtuel qui lui permet de se connecter physiquement à des partitions contenant des systèmes de fichiers provenant d'autres systèmes d'exploitation.

Système de fichiers	Commentaire
ext2	le système de fichiers de Linux
minix	le système de fichiers de Minix (historique)
MS-DOS	FAT 16 et 32
vfat	Windows 95
ISO9660	cédérom
UFS	Système de fichiers des DVD
NTFS	Windows NT (béta-test)
SMB	Serveur Lan Manager, Windows NT, WfW
HPFS	OS/2
NFS	Network File System
AFS	Andrew File System
proc	accès au noyau

Le découpage du disque et du système de fichiers sur plusieurs partitions est conseillé pour plusieurs raisons :

- certains répertoires peuvent contenir des gros fichiers (`/tmp`, `/var/spool`)
- il peut être intéressant de protéger un système de fichiers en écriture
- séparer le système des fichiers utilisateurs
- il est plus facile de sauvegarder des petites partitions

Exemple de découpage :

Point de montage	Taille
/	60 Mo
/usr	400 Mo (avec les sources)
/usr/local	200 Mo
/tmp	60 Mo
/var	60 Mo
/home	dépend du système
swap	64 Mo

Il peut être intéressant de prévoir une partition "poubelle" à partir de laquelle on peut effectuer des liens symboliques sur le reste du système de fichiers à l'aide de la commande **ln**.

La connexion des systèmes de fichiers s'effectue avec la commande **mount**. Cette commande permet d'associer un fichier spécial à un répertoire dans le système de fichiers en spécifiant le type de système sur la partition en cours de connexion.

La syntaxe est :

```
mount [-t type] fichier_spécial répertoire
```

La commande **umount** permet de terminer la connexion.

```
umount [répertoire | fichier_special]
```

Le fichier `/etc/fstab` contient une description des connexions devant être effectuées au démarrage du système. Ce fichier est composé de lignes dont la syntaxe est :

```
fichier_spécial repertoire      type    options [val1] [val2]
```

La signification des différents champs de cette ligne est la suivante :

- le fichier spécial pointe sur le périphérique à connecter. Il peut être remplacé par une adresse de machine dans le cas d'une connexion par réseau

- le répertoire donne le point de connexion
- le type de système de fichiers est spécifié
- les options permettent de donner les directives de connexion (noauto, ro, users, etc.)
- la valeur val1 est utilisée par la commande **dump**
- la valeur val2 donne l'ordre d'examen des systèmes de fichiers pour **fsck**

La commande **fsck** est utilisée pour vérifier et réparer l'intégrité du système de fichiers. Elle est exécutée automatiquement au démarrage du système avant de monter le système de fichiers. Un test est effectué pour savoir si le système a été arrêté correctement et, dans le cas contraire, pour lancer la vérification.

La vérification porte principalement sur :

- les blocs utilisés mais marqués comme libres ou vice-versa
- les nombres de liens incorrects
- la taille des fichiers
- les inodes libres (non raccrochés à un répertoire)

7.3. Imprimantes

Le système d'impression de Linux est basé sur un spooler, c'est-à-dire un processus du système qui gère les queues d'impression. Les applications n'utilisent pas directement le fichier spécial d'accès à l'imprimante mais passent par un programme qui enregistre le travail dans la queue d'impression. La version du spooler la plus utilisée est le système BSD. Il existe d'autres systèmes, comme LPRng ou des systèmes commerciaux, mais ils ne sont pas présentés ici.

Au démarrage du système, le démon lpd est lancé. Lorsqu'il démarre, ce démon lit le fichier de configuration `/etc/printcap`, imprime les fichiers en attente et se met à l'écoute de nouvelle requêtes. Il crée un nouveau processus pour chaque nouvelle requête.

Le démon lpd peut gérer des imprimantes connectées physiquement à la machine (imprimantes locales) ou des imprimantes accessibles par le réseau grâce à un autre démon lpd distant ou, par le logiciel SaMBa, sur une machine Windows (NT ou 95/98).

7.3.1. Les commandes d'impression

La commande **lpr** permet de soumettre un travail au démon lpd. On peut l'utiliser en mode filtre (lecture sur l'entrée standard) ou en lui passant les noms des fichiers comme paramètres.

La syntaxe est la suivante :

```
lpr [-Pnom] [-#nombre] [options] [fichiers ?]
```

L'option `-P` permet de spécifier la file d'impression que l'on veut utiliser. Si cette option n'est pas positionnée, la commande utilise la variable d'environnement `PRINTER`. Si celle-ci n'existe pas, lpr utilise la file `lp`.

L'option `-#` permet de spécifier un nombre de copies pour chaque fichier.

La commande **lpq** permet de visualiser le contenu de la file d'impression.

La commande **lprm** permet de supprimer un travail de la file d'impression.

L'utilisateur root gère le système d'impression avec la commande **lpc**. Cette commande fonctionne à l'aide de sous-commandes passées en arguments et peut être utilisée en interactif lorsqu'elle est lancée sans paramètre.

Les principales sous-commandes sont :

Commande	Fonction
help	donne de l'aide en ligne
abort {all | nom {	tue le démon immédiatement
clean { all | nom {	efface les fichiers temporaires
disable { all | nom {	invalide la file d'impression ; empêche lpr d'enregistrer de nouveaux fichiers
enable { all | nom {	valide la file d'impression ; permet à lpr d'enregistrer des nouveaux fichiers
restart { all | nom {	redémarre le démon
status { all | nom {	affiche l'état de la file d'impression et du démon
start { all | nom {	valide la file d'impression et démarre un démon
stop { all | nom {	termine le travail en cours et arrête le démon ; invalide la file d'impression
down { all | nom {	invalide la file d'impression ; enregistre un message dans le fichier d'état de l'imprimante
up { all | nom {	le contraire de down

Si la machine Linux doit être serveur lpd, il faut ajouter les autorisations d'accès dans le fichier `/etc/hosts.lpd` (ou `/etc/hosts.equiv`).

7.3.2. Le fichier `/etc/printcap`

Le fichier `/etc/printcap` contient la description des imprimantes gérées par le démon lpd. Les lignes commençant par le caractère "#" sont des lignes de commentaire. Une imprimante est décrite par une ligne composée de champs séparés par le caractère " :". Par exemple :

```
laser5P|lp:\
        :sd=/var/spool/lpd/laser5P:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp1:\
        :if=/var/spool/lpd/laser5P/filter:
```

Le premier champ de la ligne contient la dénomination de l'imprimante. Il peut y avoir plusieurs synonymes séparés par des caractères "|".

Il faut créer un sous-répertoire dans `/var/spool/lpd` qui porte le nom de l'imprimante et initialiser un champ de la ligne avec la variable `sd`.

Les principales variables d'une ligne de description sont :

Commande	Fonction
sd	donne le répertoire de stockage des fichiers
lp	le fichier spécial si l'imprimante est locale
lf	nom du fichier de compte rendu
if	filtre d'entrée
mx	taille maximum du fichier (mx#0 = taille illimitée)
sh	supprime l'édition des pages de garde
rp	nom d'une imprimante distante
rm	nom de la machine pour l'imprimante distante

7.3.3. Les filtres

Le démon lpd utilise des filtres pour effectuer l'impression. Les filtres sont sélectionnés en fonction des paramètres passés à la commande **lpr**. Si aucune spécification n'est donnée, le filtre indiqué dans la variable `if` du fichier `/etc/printcap` sera utilisé.

Le programme servant de filtre reçoit les paramètres d'impression en arguments d'appel. Les données à imprimer sont lues sur l'entrée standard, les données traitées sont envoyées sur la sortie standard. Les principaux paramètres sont :

- le nom de l'utilisateur
- le nom de la machine source
- le nom du fichier de comptage
- la largeur et la longueur de la page en nombre de caractères

Il existe des ensembles de filtres pour les imprimantes les plus courantes comme celui fourni avec la distribution RedHat. Ce dernier est associé à un outil de configuration printtool qui facilite la déclaration des imprimantes.

7.4. Sauvegardes

C'est la tâche la plus importante dévolue à l'administrateur du système. Il doit mettre en place une stratégie de sauvegarde automatique et vérifier régulièrement que les fichiers sont bien sauvegardés et que les média sont encore valables.

Linux propose en standard plusieurs commandes de gestion de sauvegarde :

7.4.1. Les commandes dump et restore

La commande **dump** fonctionne avec une gestion de niveaux de sauvegarde. Le niveau 0 effectue une sauvegarde totale alors que les niveaux 1 à 9 ne sauvegardent que les fichiers modifiés depuis la dernière sauvegarde d'un niveau inférieur.

La syntaxe de la commande est :

```
dump [options [arguments] ] fichier
```

Les options principales de **dump** sont :

Option	Signification
0 à 9	niveau de sauvegarde
a fichier	nom du fichier d'archivage du contenu de la sauvegarde
f fichier	nom du fichier destination
u	mise à jour le fichier /etc/dumpdates

La commande **restore** permet d'extraire des fichiers d'une sauvegarde réalisée par **dump**. Il est possible de sélectionner un fichier, un répertoire entier ou l'intégralité du système de fichiers.

La syntaxe de la commande est :

```
restore options [fichiers]
```

Les options principales de restore sont :

option	signification
i	mode interactif
r	extrait la totalité de la sauvegarde
t	liste la totalité des fichiers de la sauvegarde
x	extrait les fichiers spécifiés
f fichier	nom du fichier contenant la sauvegarde

7.4.2. La commande tar

La commande **tar** de Linux est basée sur la version GNU. La syntaxe de la commande est :

```
tar options fichiers
```

Les options principales de cette commande sont :

option	signification
c	crée une nouvelle sauvegarde
r	ajoute les fichiers à la fin de la sauvegarde
u	ajoute les fichiers plus récents
x	extrait les fichiers de la sauvegarde
t	liste les fichiers de la sauvegarde
f fichier	nom du fichier contenant la sauvegarde
--verify	effectue une vérification après écriture
--files-from	inclut les fichiers et répertoires dont les noms sont donnés dans le fichier en argument
--exclude-from fichier	n'inclut pas les fichiers dont les noms sont donnés dans le fichier en argument
v	mode bavard
p	conserve les chemins en absolu
z	utilise gzip ou gunzip pour la compression

Cette commande est très utilisée pour fournir des ensembles de fichiers en un seul paquetage compressé. Par exemple :

```
tar cvzf fichier.tar.gz repertoire
```

Cette commande effectue une sauvegarde de tous les fichiers du répertoire dans le fichier `fichier.tar.gz` en compressant les données.

7.4.3. La commande cpio

La commande **cpio** permet de copier des fichiers dans un fichier archive. Elle fonctionne selon trois modes :

- en sortie, elle copie, sur la sortie standard, les fichiers dont les noms sont donnés sur l'entrée standard (voir la commande **find**)
- en entrée, elle lit un fichier archive sur l'entrée standard et restitue les fichiers en fonction des options données
- en copie, elle copie les fichiers d'un répertoire vers un autre sans passer par un fichier archive

Les options de cette commande sont :

Option	Signification
o	mode sortie
i	mode entrée

Option	Signification
p	mode copie
v	bavard
B	passe la taille des blocs à 5120 octets au lieu de 512
d	crée les répertoires si besoin
t	affiche les noms des fichiers sans les restituer

7.4.4. La commande dd

La commande **dd** effectue des copies binaires entre deux fichiers en effectuant, optionnellement, des conversions de format.

Les options de cette commande sont :

Option	Signification
if=nom	lit le fichier "nom" au lieu de l'entrée standard
of=nom	écrit le fichier "nom" au lieu de la sortie standard
bs=nombre	utilise la taille de bloc donnée (défaut = 512 octets)
ibs=nombre	utilise la taille de bloc donnée pour l'entrée (défaut = 512 octets)
obs=nombre	utilise la taille de bloc donnée pour la sortie (défaut = 512 octets)
skip=nombre	saute le nombre de blocs en entrée avant de commencer la copie
count=nombre	copie le nombre de blocs donné
conv=ascii	convertit de l'EBCDIC en ASCII
conv=lcase	passe toutes les majuscules en minuscules
conv=swab	inverse deux à deux les octets

Cette commande est très utile pour lire et écrire des périphériques en mode caractère. Elle permet aussi de déterminer la taille des blocs d'une bande magnétique :

```
dd if=/dev/rmt of=/tmp/bloc ibs=64k count=1
```

Le fichier `/tmp/bloc` aura la taille d'un bloc d'enregistrement sur la bande.

7.4.5. La commande mt

La commande **mt** permet de gérer les lecteurs de bandes magnétiques (DAT). Sa syntaxe est :

```
mt [-f fichier] commande [paramètre]
```

Les principales commandes sont :

Option	Signification
rewind	rembobine la bande
erase	efface la bande
fsf nombre	saute le nombre de fichiers spécifiés
bsf	rembobine du nombre de fichiers spécifié
offline	éjecte la bande

7.4.6. Recherche avec find

Un administrateur système a souvent besoin d'effectuer des recherches de fichiers en fonction de différents critères :

- la taille des fichiers
- un fichier particulier
- les dates de modifications
- le propriétaire ou le groupe

Ces recherches peuvent être effectuées à l'aide de la commande **find**. Celle-ci possède une importante liste d'options pour configurer les paramètres de la recherche. La syntaxe de base est la suivante :

```
find [chemins ?] [expression]
```

où :

- chemins est une liste de répertoires à parcourir (par défaut le répertoire courant)
- expression est une liste d'options, de tests, d'opérateurs et d'actions

Le tableau ci-après présente un extrait des options de cette commande :

Option	Signification
-follow	suit les liens symboliques
-maxdepth n	limite la profondeur de descente des répertoires
-mindepth n	commence à n niveaux en dessous des répertoires donnés
-mount	ne parcourt pas les autres systèmes de fichiers connectés

Le tableau ci-après présente un extrait des tests possibles pour la sélection des fichiers. Pour ces tests, lorsqu'une valeur `n` est demandée en paramètre, il est possible de spécifier `+n` pour les valeurs supérieures à n, `-n` pour les valeurs inférieures à n et `n` pour les valeurs égales à n.

Tests	Signification
-name expr	les fichiers dont le nom correspond à l'expression régulière
-mtime n	les fichiers modifiés pendant les n derniers jours
-nouser	les fichiers dont l'uid ne correspond à aucun utilisateur
-nogroup	les fichiers dont le gid ne correspond à aucun groupe
-size n	les fichiers dont la taille est égale à n
-type t	les fichiers de type spécifié

En plus des options de recherche et des tests, find permet de spécifier des actions à effectuer sur les fichiers trouvés :

Actions	Signification
-exec commande	permet d'effectuer une commande avec en paramètre le nom du fichier
-print	affiche le nom du fichier
-ok commande	permet d'effectuer une commande avec le nom du fichier en paramètre, avec une confirmation utilisateur

Chapitre 8. Automatisation

Linux fournit un mécanisme à l'administrateur système pour automatiser les tâches répétitives. Le programme cron est utilisé pour planifier ces tâches et pour fournir un compte rendu d'exécution éventuel. La commande **crontab** est utilisée pour définir les tâches que doit exécuter **cron**.

Le démon est lancé au démarrage du système et regarde, toutes les minutes, dans le fichier `/etc/crontab` et le répertoire `/var/spool/cron` s'il y a des tâches à exécuter.

8.1. Le fichier crontab

Un fichier crontab contient des instructions pour le démon cron. Chaque utilisateur possède son propre fichier crontab dans le répertoire `/var/spool/cron`. Il n'est pas éditable directement mais doit être modifié à l'aide de la commande **crontab**.

Les lignes de ce fichier contiennent soit des initialisations de variables d'environnement, soit des commandes pour le démon **cron**.

Les variables `$LOGNAME`, `$HOME`, `$SHELL` sont initialisées automatiquement.

Une ligne de commande de **cron** est constituée :

- de cinq champs de définition de temps
- d'un identificateur d'utilisateur (uniquement pour les fichiers crontab du système)
- d'une commande à exécuter

Les champs de définition des temps représentent dans l'ordre :

- les minutes (valeurs de 0 à 59)
- les heures (valeurs de 00 à 24)
- les jours (valeurs de 1 à 31)
- les mois (valeurs de 1 à 12)
- le jour de la semaine (valeurs de 0 à 7, Dimanche est le 0 et le 7, ou sun, mon, tue, ?)

Chacun de ces champs peut prendre une des valeurs suivantes :

- le caractère "*" qui signifie "n'importe quand"
- une valeur numérique
- une liste de valeurs séparées par des virgules
- deux valeurs séparées par un tiret pour définir un intervalle

Le résultat de la commande doit être envoyé dans un fichier car cron ne dispose d'aucun terminal pour afficher le résultat. Si des sorties sont générées, cron les envoie par courrier à l'utilisateur.

8.2. La commande crontab

La commande **crontab** permet de modifier la liste des commandes **cron** d'un utilisateur donné. Elle peut être utilisée de deux façons :

- soit en passant un fichier en paramètre : crontab fichier
- soit en mode interactif : crontab -e

La première méthode remplace le fichier crontab de l'utilisateur par le contenu du fichier donné en paramètre. C'est la méthode utilisée dans un shell script.

La deuxième méthode permet de modifier une ligne donnée du fichier crontab existant sans remettre en cause les autres lignes de commande.

8.3. La commande at

Cette commande permet de demander l'exécution d'un travail à une date donnée. Elle lit la liste des commandes à exécuter sur son entrée standard. Si les commandes génèrent des sorties, celles-ci sont envoyées sous forme d'un mail. Les spécifications de date sont très nombreuses et offrent de nombreuses possibilités.

La commande **atq** permet de visualiser la liste des travaux en attente.

Chapitre 9. Le noyau

Le noyau (*kernel*) du système Linux est responsable du bon fonctionnement de l'ensemble du système. C'est lui qui a la responsabilité :

- de l'ordonnancement des processus avec la gestion des priorités
- de la gestion de la mémoire
- de la gestion des interruptions et des entrées / sorties
- de la sécurité
- de la communication entre processus (IPC)

Le fichier contenant le noyau se trouve dans le répertoire `/boot` et s'appelle `vmlinuz`. Il s'agit d'un fichier compressé qui sera chargé au démarrage du système. Ce fichier est créé par la procédure de génération du noyau.

Une des forces de Linux est de gérer ses sous-systèmes sous forme de modules chargeables pendant l'exécution du noyau. Cela permet de créer un noyau plus petit qui consomme moins de ressources mémoire inutiles lorsqu'un périphérique n'est pas utilisé.

9.1. La génération d'un noyau

Un administrateur système peut être amené à générer un nouveau noyau pour différentes raisons :

- ajouter des périphériques non supportés dans la version actuelle
- créer un noyau plus petit en enlevant des sous-systèmes non utilisés
- mettre à jour son système vers une version plus récente

La première étape consiste à installer les sources du noyau dans le répertoire `/usr/src/linux`. Avec les sources, on trouve un répertoire `Documentations` dans lequel sont placées des informations utiles pour la génération du noyau.

L'étape suivante consiste à configurer les sous-systèmes composant le noyau. Cela consiste à définir pour chaque élément s'il doit être inclus directement dans le noyau, s'il doit apparaître sous forme de module ou s'il ne doit pas être présent. Pour chaque option, un fichier d'aide permet de prendre la bonne décision. Cette configuration peut être effectuée de trois façons :

- **make config** : cette commande déroule un grand nombre de questions sur la console.
- **make menuconfig** : il s'agit d'une interface plus conviviale sous forme de menus en mode texte
- **make xconfig** : il s'agit d'une application fonctionnant dans l'environnement Xwindow.

Ensuite il faut valider les dépendances entre fichiers à l'aide de la commande **make dep** puis, s'assurer que les anciens fichiers sont bien détruits en tapant **make clean**. Ces deux opérations ne sont pas très longues et ne doivent pas être omises.

La compilation effective du noyau est lancée par la commande **make zImage** qui place le fichier généré dans le répertoire `arch/i386/boot`. Une alternative intéressante consiste à compiler une image en plaçant le résultat sur une disquette à l'aide de la commande **make zdisk**. Cette possibilité permet de tester le nouveau noyau sans détruire celui se trouvant sur le disque dur.

L'installation du nouveau noyau s'effectue par la commande **make zlilo** à condition d'avoir configuré préalablement le fichier `/etc/lilo.conf`.

La compilation et l'installation des modules du nouveau noyau s'effectuent au moyen des commandes **make modules** et **make modules_install**.

Une fois ces opérations réalisées, le nouveau noyau est prêt à être testé, ce qui nécessite de redémarrer la machine avec la commande **reboot**.

9.2. LILO

La commande **lilo** permet de générer et d'installer le programme de démarrage du noyau : LILO (*LInux LOader*). Cette commande utilise un fichier de configuration, `/etc/lilo.conf`, dans lequel sont donnés les paramètres d'installation. Le but de cette commande est de faire connaître au programme de démarrage l'adresse exacte, en termes de secteurs sur le disque, du fichier image contenant le noyau.

Le programme LILO peut aussi démarrer d'autres systèmes d'exploitation installés sur des partitions différentes.

Voici un exemple de fichier de configuration :

```
boot = /dev/hda
delay = 30

# linux
image = /boot/vmlinuz
        root = /dev/hda1
        label = linux
# dos
other = /dev/hda4
        table = /dev/hda
        label = dos
```

Cet exemple installe le chargeur LILO sur le secteur de démarrage principal du disque dur (MBR) et autorise un délai de 3 secondes pour choisir le système à démarrer. Si l'utilisateur ne prend pas la main dans le temps imparti, c'est le premier système qui sera choisi. A moins de taper sur la touche **tab**, ce qui donne les différentes possibilités pour démarrer.

Les déclarations pour le système Linux sont :

- image : donne le nom du fichier contenant l'image du noyau
- root : la partition sur laquelle de trouve la racine du système de fichiers
- label : le nom utilisé pour sélectionné le système à démarrer en interactif

Les déclarations pour le système DOS sont :

- other : donne la partition sur laquelle se trouve le système
- table : donne le nom du disque qui contient la table des partitions
- label : le nom utilisé pour sélectionner le système à démarrer en interactif

Il est possible de passer des arguments au noyau pour configurer des périphériques particuliers. Ceci peut être fait dans le fichier `/etc/lilo.conf` ou directement sur la ligne de commande de LILO.

9.3. Les modules

Les fichiers contenant le code des modules chargeables sont dans le répertoire `/lib/modules`. Lorsque l'on veut utiliser un périphérique dont le pilote est sous forme de module, il faut installer le module dans le noyau. Ceci est effectué grâce à la commande **insmod**. Celle-ci prend en paramètre le nom du module et les arguments propres au module.

Par exemple, pour installer une carte réseau 3c509 :

```
insmod 3c509 irq=5
```

Si un module a besoin d'autres modules pour fonctionner, il faut les installer les uns après les autres.

Une autre méthode utilise les commandes **depmod** et **modprobe**. La commande **depmod** est exécutée au démarrage pour construire une table de dépendance entre modules. La commande **modprobe** utilise cette table pour charger les modules dont dépend celui que l'on veut installer.

Pour la même carte :

```
modprobe 3c509 irq=5
```

Le fichier `/etc/conf.modules` est utilisé pour fixer les paramètres des modules couramment utilisés. Il permet aussi de spécifier des alias pour les noms demandés par les commandes systèmes.

Par exemple, la ligne :

```
alias eth0 3c509
```

permettra le chargement du module **3c509** à l'initialisation de l'interface réseau eth0.

Le démon kerneld est utilisé pour automatiser le chargement des modules. Lorsque le noyau a besoin d'un module, il le demande à kerneld. Celui-ci lance **modprobe** pour installer le module. Ce démon permet aussi d'enlever les modules qui ne sont plus utilisés.

Chapitre 10. Suivi et traces

Une part importante du travail d'un administrateur système consiste à maintenir les machines dont il a la responsabilité en ordre de marche. Linux met à disposition des commandes et des fichiers d'enregistrement de traces pour faciliter le suivi du fonctionnement du système.

10.1. L'utilisation du disque

10.1.1. Les commandes de suivi

La commande **df** permet de visualiser la place disponible sur les différents systèmes de fichiers raccordés à la machine. Cette commande affiche :

- le nom du fichier spécial
- la taille totale (en nombre de blocs)
- l'espace utilisé
- l'espace libre
- le pourcentage de place utilisée
- le point de montage

L'option `-i` permet d'afficher les mêmes valeurs pour les inodes.

La commande **du** permet d'afficher la place disque utilisée par un ensemble de fichiers. Pour chaque fichier, du affiche la place occupée en nombre de blocs et parcourt les sous-répertoires de façon récursive. L'option -s permet d'afficher la somme des tailles des fichiers pour un répertoire.

10.1.2. Gestion des quotas

Linux permet d'affecter des quotas d'utilisation du disque pour un utilisateur ou un groupe d'utilisateurs sur un système de fichiers donné. Cela permet de limiter l'utilisation de la place disque ou des inodes en fonction de deux limites :

- une limite douce qui peut être dépassée pendant un délai donné
- une limite dure qui ne peut pas être dépassée

La gestion des quotas est démarrée dans le fichier `rc.sysinit` grâce à la commande **quotaon**.

Pour configurer la gestion des quotas, il faut ajouter dans le fichier `/etc/fstab` les mots `usrquota` ou `grpquota` au niveau des options de montage :

```
/dev/hda2   /home   ext2   defaults,usrquota,grpquota  1 1
```

La définition des valeurs des quotas est effectuée à l'aide de la commande **edquota**. Cette commande affiche les informations courantes pour l'utilisateur demandé :

```
Quotas for user sandra:
/dev/sda5: blocks in use: 43552, limits (soft = 50000, hard = 60000)
    inodes in use: 907, limits (soft = 0, hard = 0)
```

Il est possible de modifier les champs soft et hard pour les limiter en nombre de blocs et en inodes. Une valeur à zéro annule la gestion des quotas pour cet utilisateur.

La commande **edquota -t** permet de fixer la période de grâce pour la limite douce.

La commande **repquota -a** permet de visualiser les valeurs courantes pour l'ensemble des utilisateurs.

10.2. Les processus et la mémoire

La commande **ps** permet de visualiser la liste des processus en cours d'exécution. Les différentes options sont :

Option	Signification
a	montre les processus de tous les utilisateurs
x	montre les processus non attachés à un terminal
l	format long
m	affiche les informations sur la mémoire

Les principales valeurs affichées par cette commande sont :

Option	Signification
PID	l'identificateur de processus
TTY	le terminal de contrôle
STAT	l'état du processus
SIZE	affiche la taille du processus (texte + données + pile)
RSS	taille du processus en mémoire
TIME	le temps d'exécution cumulé

La commande **top** affiche la liste des processus s'exécutant ainsi que plusieurs variables système. Cet affichage est rafraîchi périodiquement avec un intervalle configurable. Les processus sont triés par ordre décroissant d'utilisation du CPU.

Les valeurs affichées représentent :

- le temps depuis le démarrage de la machine
- la charge moyenne

- le nombre de processus
- le taux de charge du CPU
- l'utilisation de la mémoire
- l'utilisation de la zone d'échange

La commande **free** permet de visualiser l'utilisation de la mémoire. Elle affiche les valeurs pour :

- la mémoire disponible
- la mémoire utilisée
- la mémoire libre
- la mémoire partagée
- le tampon de cache disque
- l'utilisation de la zone d'échange

10.3. Les fichiers de trace

Le système Linux possède un système centralisé d'enregistrement de traces utilisable par tous les processus système ainsi que par le noyau. Ce système est composé d'un démon (syslogd), d'une API, d'une commande pour les scripts (**logger**) et de fichiers d'enregistrement (`/var/log/messages`, etc.).

Le démon syslogd est configuré par le fichier `/etc/syslog.conf`. Celui-ci contient les règles d'enregistrement exprimées sous la forme :

```
dispositif.niveau       action
```

Le dispositif donne le nom du sous-système qui a émis le message. Par exemple, tous les programmes de gestion de courrier doivent utiliser le dispositif mail. Le niveau donne la gravité du message.

Le dispositif est l'un de ces mots : `auth` (security), `auth-priv`, `cron`, `daemon`, `kern`, `lpr`, `mail`, `news`, `syslog`, `user`, `uucp` et `local0` à `local7`. Le niveau est donné par un mot dans la liste suivante (classée par ordre ascendant) : `debug`, `info`, `notice`, `warning` (warn), `err` (error), `crit`, `alert`, `emerg` (panic).

L'action est une généralisation du fichier d'enregistrement. Elle peut être :

- un fichier standard
- un tube nommé (fifo)
- un terminal
- une machine distante
- une liste d'utilisateurs
- tous les utilisateurs connectés

Le message enregistré contient la date, le nom de la machine, le nom du programme, son identificateur de processus (PID) suivi du message proprement dit.

Les commandes **last** et **ac** permettent de visualiser les informations d'ouverture et de fermeture de sessions enregistrées dans le fichier `/var/log/wtmp`.

10.4. Le système de fichiers /proc

Le système de fichiers `/proc` donne l'accès, par l'intermédiaire de pseudos fichiers, aux variables internes du noyau.

A la racine de ce système de fichiers, on trouve un répertoire par processus en exécution ayant comme nom l'identificateur du processus. Dans chacun de ces répertoires, on a :

- cmdline : donne la ligne de commande
- cwd : c'est un lien sur le répertoire courant du processus
- environ : contient l'environnement du processus
- exe : c'est un lien sur le fichier exécutable
- fd : répertoire contenant les liens sur les fichiers ouverts
- maps : liste des zones mémoire du processus
- mem : contenu de l'espace d'adressage du processus
- root : lien sur le répertoire racine du processus
- stat, statm, status : informations sur l'état du processus

Les principaux fichiers sont :

- devices : liste des pilotes avec les major numbers
- dma : liste des canaux dma utilisés
- interrupts : liste des interruptions utilisées
- ioports : liste des ports utilisés
- loadavg : charge moyenne du système
- meminfo : état de la mémoire
- modules : liste des modules chargés
- stat : informations sur le système
- net : répertoire contenant les informations sur le réseau
- scsi : répertoire contenant les informations sur les périphériques scsi
- sys : répertoire contenant des variables du noyau

Les informations retournées par le pseudo fichier stat sont :

- cpu : temps passé dans les états user, nice, system et idle en 1/100 de seconde
- disk : pour les quatre premiers disques, donne le résumé des opérations effectuées
- disk_rio : lecture
- disk_wio : écriture
- disk_rblk : lecture blocs
- disk_wblk : écriture blocs
- page : pages lues et écrites
- swap : compte des échanges en lecture et en écriture
- intr : total des interruptions depuis le démarrage
- ctxt : nombre de changements de contexte
- btime : heure du démarrage
- processes : dernier PID utilisé

Chapitre 11. Réseaux

Linux supporte de nombreux types de matériels réseau et en particulier les interfaces pour :

- FDDI
- relais de trames
- RNIS
- PLIP, SLIP, PPP
- X.25
- Ethernet

11.1. Configuration

La configuration du réseau sous Linux peut être effectuée lors de l'installation ou a posteriori en modifiant les fichiers concernés.

Le fichier `/etc/sysconfig/network` contient les variables :

- `NETWORKING` : initialisée à "yes" pour valider l'utilisation du réseau
- `FORWARD_IPV4` : initialisée à "no" pour empêcher le transfert automatique des paquets
- `HOSTNAME` : contient le nom complet de la machine
- `GATEWAYDEV` : interface d'accès à la passerelle
- `GATEWAY` : adresse IP de la passerelle

Ce fichier est utilisé dans les scripts d'initialisation du réseau pour positionner les variables d'environnement.

Pour chaque interface, il faut construire un fichier `ifcfg-<nom>` où nom est remplacé par le nom de l'interface utilisée :

- `ethN` pour la Nième interface réseau
- `dummyN` pour la Nième fausse interface

Ce fichier contient les variables suivantes :

- `DEVICE` : nom du périphérique
- `ONBOOT` : initialisée à "yes" pour valider l'interface au démarrage
- `BROADCAST` : contient l'adresse IP de diffusion
- `NETWORK` : contient l'adresse IP du réseau
- `NETMASK` : contient le masque du réseau
- `IPADDR` : contient l'adresse IP de l'interface

Ce fichier est utilisé en paramètre des scripts **ifup** et **ifdown** d'initialisation de l'interface. L'initialisation du réseau au démarrage du système est effectuée par le script **/etc/rc.d/init.d/network**.

11.2. Les commandes

La commande **hostname** permet d'afficher le nom de la machine ainsi que le nom de domaine.

La commande **ifconfig** permet de visualiser ou de configurer les interfaces. Lorsqu'elle est exécutée sans argument, elle affiche la configuration actuelle avec, pour chaque interface, les informations suivantes :

- adresse Ethernet
- adresse IP
- adresse de diffusion
- masque de réseau
- le nombre de paquets reçus et transmis

La commande **netstat** permet d'afficher :

- les connexions réseau actives et en attente
- les tables de routage
- des statistiques sur l'interface
- les tables de translation d'adresse

La commande **arp** permet de visualiser et d'agir sur la table de résolution d'adresses Ethernet.

La commande **route** permet de visualiser et de modifier la table de routage de la machine.

Les commandes **ping** et **traceroute** permettent de valider le chemin réseau jusqu'à un hôte donné. La commande **traceroute** donne la liste des routeurs par où a transité la demande.

11.3. Les fichiers

Le fichier `/etc/hosts` contient une liste d'adresses IP associées aux noms des machines. C'est le moyen le plus simple d'effectuer la résolution de noms pour un petit réseau.

Le fichier `/etc/host.conf` spécifie le mode de résolution des noms de machines. Il contient les lignes suivantes :

Option	Description
order	spécifie l'ordre d'utilisation des différents moyens de résolution de noms : hosts : fichier `/etc/hosts` bind : serveur de noms DNS nis : Network Information Service

Option	Description
nospoof	validé par la valeur on, cela permet de détecter les tentatives d'usurpation d'adresse IP.
alert	validé par la valeur on, cela permet d'enregistrer, via syslog, les tentatives d'usurpation d'adresse IP.
multi	validé par la valeur on, cela permet d'affecter plusieurs adresses IP au même hôte dans le fichier `/etc/hosts`
trim	permet d'enlever le nom de domaine en argument avant d'effectuer une recherche dans le fichier `/etc/hosts`

Le fichier `/etc/resolv.conf` permet de configurer la partie DNS de la résolution de noms. Ce fichier contient :

Option	Description
domain	spécifie le nom de domaine de la machine
nameserver	donne une adresse IP d'un serveur de nom ; il est possible de spécifier trois serveurs de noms
search	liste les noms de domaines à chercher

Le fichier `/etc/services` donne la liste des services TCP et UDP supportés par la machine. Il associe le nom du service au numéro de port et au protocole.

Le fichier `/etc/inetd.conf` permet de configurer le démon inetd qui est le super serveur de Linux. Ce démon est à l'écoute des demandes de connexion et gère le lancement des autres serveurs (telnet, ftp, etc.).

11.4. Applications réseau

11.4.1. DNS

Une machine Linux peut être configurée en serveur de noms DNS. Elle pourra répondre aux requêtes des autres machines du réseau pour la résolution des noms en adresse IP.

11.4.2. SaMBa

SaMBa est l'émulation d'un serveur LAN MANAGER® et permet de fournir des disques et des imprimantes partagés à des PC sous Windows®. La partie cliente existe et permet à une machine Linux de se connecter à un disque ou une imprimante partagés.

11.4.3. SENDMAIL

Sendmail est un logiciel de transport de courrier électronique. Il gère l'envoi et la réception du courrier en fonction des caractéristiques des adresses données. Il s'occupe du routage et de la modification éventuelle des adresses pour permettre au message d'arriver à destination. Sendmail peut gérer des listes de diffusion de courrier.

11.4.4. NFS

Le système NFS permet de partager des disques et des imprimantes à travers le réseau. Il existe deux parties dans NFS, la partie serveur qui consiste à exporter une partie de son système de fichiers vers les machines clientes et la partie cliente qui consiste à attacher les systèmes de fichiers comme s'ils faisaient partie du système local.

11.4.5. NIS

NIS est une base de données qui permet de diffuser et de contrôler les fichiers d'administration importants. La gestion s'effectue sur un domaine possédant un nom unique sur le réseau.

Chapitre 12. Xwindow

Le système de fenêtres Xwindow a été développé, à l'origine, par le MIT. La version actuelle utilisée sous Linux, Xfree86, est une reprise de ce projet par un groupe de programmeurs en vue de faire une version libre de cet outil.

Ce système fonctionne en mode client-serveur ; il est composé d'une part, d'un serveur X qui effectue la gestion des périphériques de la machine (écran, clavier et souris) et d'autre part de programmes clients qui utilisent les services de ce serveur pour interagir avec un utilisateur. Le client peut adresser un serveur distant par le réseau et donc, afficher et saisir des informations sur une machine distante.

La configuration de Xwindow consiste à décrire les périphériques au programme serveur, celui-ci étant sélectionné en fonction de la carte graphique utilisée. La configuration est effectuée dans le fichier `/etc/X11/XF86Config` soit à l'aide d'un éditeur, soit par l'utilisation du programme **xf86config**.

Le fichier `XF86Config` est constitué de plusieurs sections :

- Files : chemin d'accès aux fichiers supplémentaires
- Module : spécifie les modules optionnels à charger
- ServerFlags : permet de spécifier des options de fonctionnement du serveur
- Keyboard : définition du clavier
- Pointer : définition du système de pointage (souris)
- Monitor : spécification des caractéristiques du moniteur
- Device : description de la carte graphique
- Screen : propriétés de l'affichage
- XInput : spécifications des périphériques d'entrées optionnels (joystick, tablette, etc)

Pour pouvoir configurer correctement le serveur Xwindow, il faut connaître précisément les caractéristiques du matériel utilisé. Cependant, la plupart des distributions Linux proposent des outils supplémentaires de configuration de X ; on pourra mentionner **Xconfigurator** disponibles sur les distributions RedHat.

La configuration de Xwindow au niveau utilisateur s'effectue dans le script .xinitrc. Ce script contient les commandes à exécuter à l'ouverture de la session Xwindow. Il se termine généralement par une ligne **exec commande** de façon à ce que l'arrêt de ce client (généralement le gestionnaire de fenêtre) stoppe le serveur X.

Chapitre 13. Sécurité

Ce chapitre présente les procédures et les outils permettant de maintenir l'intégrité de la machine afin qu'elle réalise son travail sans interruption.

13.1. Généralités

La première étape est de veiller à la sécurité physique de la machine en fonction de la disponibilité demandée. L'utilisation d'un onduleur et d'une salle fermée avec un accès limité permet de garantir que le système ne sera pas arrêté. De même, la'accès à la console doit être limité aux personnes de confiance, et tout les protections doivent être mises en oeuvre pour éviter que quelqu'un de mal intentionné puisse démarrer le serveur avec une disquette ou un cd de démarrage.

La deuxième étape est de protéger le système des actions des utilisateurs locaux. Il faut créer les comptes utilisateurs avec les droits minimum et limiter l'accès aux fichiers système. Les mots de passe donnés pour l'accès au système ne doivent pas être triviaux ni faciles à découvrir.

La troisième étape consiste à définir une politique d'accès du réseau vers la machine et de la machine vers le réseau. D'une manière générale, seuls les services réseaux nécessaires au fonctionnement doivent être démarrés, et autant que possible des restrictions d'accès à ces services doivent être mis en place (limitation à certaines adresses IP et accès par login/mot de passe...).

13.2. Les services systèmes

L'accès aux services réseau du système peut être protégé par l'utilisation du TCPWrapper tcpd. Ce programme permet de spécifier la liste des machines ayant le droit d'utiliser un service donné et ce, service par service. Les accès sont enregistrés dans un fichier de trace `/var/log/secure`.

En standard, le fichier `/etc/inetd.conf` est configuré pour utiliser ce programme. Il faut saisir les règles dans les fichiers `/etc/hosts.allow` et `/etc/hosts.deny`.

Certains services, comme NFS, NIS, X11 et SaMBa, ne sont pas sécurisés et ne doivent donc pas être utilisés sur une machine "à risque".

13.3. Les firewalls

Un firewall est un dispositif isolant un réseau considéré comme "sûr" d'un réseau en principe "hostile". Le cas le plus fréquent est l'accès à Internet, mais cela permet d'isoler aussi différents départements d'une même société. Ce dispositif est constitué de matériels, routeurs et ordinateurs, et de logiciels pour la partie active et la configuration.

Linux possède un système de filtrage de paquets intégré au noyau qui permet de réaliser un routeur filtrant en entrée d'un réseau. Il existe aussi différents programmes permettant de réaliser des mandataires pour les services réseaux les plus courants. On peut citer le Firewall Toolkit de la société TIS, Socks, Squid.

13.4. Les outils

13.4.1. SATAN

C'est l'outil le plus connu pour la recherche des problèmes de sécurité à partir du réseau. Il peut parcourir toutes les machines d'un réseau et tester les potentialités d'intrusion à partir des services systèmes.

13.4.2. TRIPWIRE

C'est un outil de vérification d'intégrité de fichiers. Il compare un ensemble de fichiers et de répertoires aux données collectées lors d'un précédent passage. Il génère un rapport sur toutes les modifications détectées sur la taille des fichiers, la destruction ou la création de nouveaux fichiers.

13.4.3. SAINT

SAINT (Security Administrator's Integrated Network Tool) est un outil permettant l'analyse d'un réseau de machines et établissant un rapport sur les trous de sécurité potentiels. Il gère quatre niveaux de sévérité et produit un rapport en HTML.

13.4.4. COPS

C'est un système qui vérifie la présence de problèmes de sécurité sur un système de fichiers. Il vérifie les permissions des fichiers système et génère un rapport sur les modifications à apporter. Il vérifie aussi la pertinence des mots de passe grâce à un outil d'analyse par dictionnaire.

Chapitre 14. Outils graphiques d'administration

14.1. Les outils Red Hat

Les outils graphiques d'administration système livrés avec la distribution Red Hat sont lancés à partir d'un panneau de contrôle affiché à l'exécution de la commande **control-panel**.

Les outils disponibles permettent :

- la gestion de l'arrêt et du démarrage des sous-systèmes de Linux
- la gestion de la date et de l'heure
- la gestion des imprimantes locales ou distantes
- la configuration des interfaces réseau et des tables de routage
- la configuration des modems
- la gestion des modules
- la gestion des paquetages
- le lancement de linuxconf

14.2. Linuxconf

Linuxconf est un outil d'administration système possédant plusieurs interfaces opérateur. Il peut être utilisé en mode ligne de commande, à travers un navigateur Web ou avec une interface graphique.

Il permet de gérer :

- le réseau (DNS, routage, NFS, etc.)
- Apache
- SaMBa
- les utilisateurs
- le système de fichiers
- les niveaux de démarrage

Cet outil est livré en standard dans la distribution RedHat.

14.3. Webmin

Webmin est une interface Web pour l'administration d'un système Linux. Elle est composée d'un serveur Web simple écrit en Perl et de nombreux modules permettant la configuration du système et de quelques logiciels serveurs.

L'installation de cet outil est très facile et ne nécessite aucune compilation. Un simple script de mise en service permet de saisir les paramètres nécessaires.

Il se lance à partir d'un navigateur internet avec l'URL suivante : (http ://nom_de_la_machine :10000) (par exemple, (http ://localhost :10000) sur une machine locale). Le login et le mot de passe demandés la première fois sont ceux de l'administrateur (utilisateur "root"). Mais il est possible de configurer des utilisateurs Webmin en leur donnant les droits pour administrer quelques modules.

Webmin, dans sa version 0.82, est composé de cinq onglets :

- Webmin : permet de configurer "webmin" et créer des utilisateurs "webmin" avec des droits d'administration plus ou moins restreints.
- Système : permet de gérer les paramètres systèmes de Linux, à savoir : cron, NFS, utilisateurs Unix, montage des unités disques, quotas disques, ajout/suppression de programmes, démons, processus et les pages du manuel.
- Serveurs : permet de configurer les logiciels serveurs suivants : Sendmail, Samba, MySQL, PostgreSQL, Apache, Squid, Bind, Wuftp, DHCP, Inet, Postfix, PPP et Majordomo.
- Matériel : permet de configurer le "hardware" : imprimante, disque dur, réseau, horloge, Lilo et les disques "RAID".
- Autres : contient un gestionnaire de fichier ainsi qu'un "telnet" programmés en JAVA ; cet onglet exige donc un navigateur compatible JAVA.

Mise en garde

La version en cours, au moment ou sont écrites ces lignes, est la 0.82. Il s'agit donc encore d'une version "béta" même si elle est déjà opérationnelle. Par conséquent, il est plus que recommandé d'effectuer une sauvegarde des fichiers de configuration avant de se lancer avec Webmin ! Il arrive souvent (c'est le cas notamment avec Samba) que Webmin supprime les commentaires d'aide d'un fichier de configuration ce qui rend difficile sa relecture au cas ou l'on serait amener à le modifier "à la main".

Chapitre 15. Gestion des packages

Les packages logiciels sur les distributions Redhat et dérivées sont au format RPM (Redhat Package Manager). La gestion de ces packages logiciels s'effectue en ligne de commande en utilisant la commande **RPM**, mais d'autres outils, dont certains graphiques, sont disponibles.

Les packages sont reconnaissables par leur extension `.rpm` ou plus exactement `architecture.rpm`. Pour les plates-formes de type PC, l'extension est `i386.rpm`. Ces packages sont constitués d'une archive au format cpio de l'application compilée et des fichiers associés (configuration, documentation...). On trouve également des packages source (indépendants de la plate-forme) devant être compilés avant d'être installés ; ces packages sources possèdent l'extension `.src.rpm` ou éventuellement `.srpm`.

15.1. Fonctionnement des packages RPM

L'avantage de l'installation de logiciels sous forme de RPM, outre l'absence de compilation, est une gestion centralisée des logiciels installés, simplifiant ainsi grandement les actions de suppression ou de mise à jour. Toutes les informations concernant les packages installés sont stockées dans des fichiers au format Berkeley DB se trouvant dans le répertoire `/var/lib/rpm`. Chaque package fournit des informations quand à ses dépendances et ses conflits éventuels avec d'autres packages. Ces informations sont stockées dans les fichiers de ce répertoire. Ces fichiers ne servent qu'à la commande **rpm** mais sont vitaux pour son fonctionnement.

En cas de corruption ou de suppression d'un fichier, il est possible de les reconstruire en utilisant la commande **rpm --rebuilddb**.

15.2. Commande RPM

La commande **rpm** est l'outil de base pour la gestion des packages ; elle permet notamment l'installation, la suppression, la compilation, la vérification des packages.

Les principales options de la commande **rpm** sont les suivantes :

```
rpm -Uvh package.i386.rpm
```

Met à jour (ou installe) le package "package" en mode verbeux (`-v`) avec affichage de la progression (`-h`).

```
rpm -e package
```

Désinstalle le package "package". Ici l'extension n'apparaît pas puisque celle-ci se réfère à un nom de fichier. Une fois le package installé, on s'y réfère sans l'extension.

```
rpm -q package
```

Recherche la version installée du package "package". Le nom du package fourni doit être le nom exact. Si le nom n'est pas connu exactement, on peut utiliser l'option `-a` pour afficher la liste de tous les packages installés, et éventuellement utiliser un grep pour trouver ce que l'on cherche.

```
rpm -qa | grep pack
```

Pour trouver de quel package provient un fichier sur le disque on utilisera :

```
rpm -qf /chemin/vers/le/fichier
```

Inversement, si l'on veut connaître la liste des fichiers installés par un package, on utilisera, respectivement pour un package installé et pour un package non installé :

```
rpm -ql package
rpm -qlp package.i386.rpm
```

La commande

```
rpm -qi package
```

affichera les informations concernant le package, à savoir notamment son auteur, sa date de création, sa licence ainsi qu'un résumé de ses fonctionnalités

Enfin, la commande

```
rpm -V package
```

vérifiera l'intégrité des fichiers présents sur la machine par rapport aux fichiers initiaux.

15.3. Outils graphiques

Quelques outils graphiques sont disponibles pour la gestion des packages RPM. On citera notamment :

- kpackage fonctionnant dans l'environnement Kde, non fourni en standard dans les distributions RedHat. Ce logiciel sait également travailler avec les packages d'autres distributions (Debian, Slackware...)
- gnorpm est un logiciel fourni avec l'environnement graphique gnome. Comme son nom l'indique, il permet la gestion des packages au format rpm.
- up2date est un outil standard de la distribution RedHat (version 6.1 et supérieure) permettant la mise à jour de la distribution. Cet utilitaire va rechercher sur le serveur ftp de redhat les packages mis à jour pour la distribution. Il présente ensuite une page web permettant de sélectionner les packages à mettre à jour. Cette application ne permet pas d'installer des packages hors distribution.

15.4. Commande rpmfind

La commande **rpmfind** permet de chercher un logiciel sous forme de paquetage rpm. Son fichier de configuration se situe dans le répertoire /etc et se nomme "rpmfind.conf". Dans ce fichier se trouve la liste des sites Internet de recherche des paquetages.

La syntaxe de la commande est la suivante : **rpmfind [options] noms** Voici la liste des options :

Option	Effets
-h	Liste toutes les options valables
-v	verbose
--auto	Télécharge le paquetage sans demander confirmation
--source	Recherche le paquetage sous forme de source et non sous forme 0
--latest	Recherche la version la plus récente du paquetage
-s serveur	Demande de rechercher le paquetage sur le site spécifié
--upgrade	Recherche la mise à jour d'un paquetage et propose également la mise à jour de toutes ses dépendances
--apropos	Scanne tous les sites et affiche les infos à propos du paquetage par site

Quelques sections du fichier de configuration "rpmfind.conf" méritent une attention particulière.

Les sections suivantes permettent de pouvoir faire fonctionner **rpmfind** lorsque l'on se trouve derrière un proxy :

```
; your HTTP proxy/cache if any : http ://myhttpproxy/
 httpProxy=http ://nom_du_proxy :3128/

; your FTP proxy/cache if any : ftp ://myftpproxy/
 ftpProxy=http ://nom_du_proxy :3128/
```

"3128" est le numéro de port du proxy (ici, dans le cas de l'utilisation du serveur proxy SQUID).

La section suivante permet de définir le répertoire où sont stockés les paquetages téléchargés :

```
; where to save RPMs
 downloadDir=/tmp
```

Chapitre 16. En cas de problèmes sur le système

Il peut arriver que le système soit dans un état suffisamment instable pour qu'il ne soit plus possible de pouvoir booter correctement. Les causes principales peuvent être des problèmes sur les systèmes de fichiers (physiques ou logiques), ou des problèmes liés au noyau ou aux modules. Pour restaurer le système plusieurs moyens sont disponibles.

16.1. Mode Single User

Le mode Single User est un mode particulier de fonctionnement du système correspondant au niveau d'exécution 1 (runlevel 1) dans lequel seuls les services minimum sont lancés. Il est souvent appelé mode de maintenance. Dans ce runlevel, un shell root est lancé automatiquement au démarrage, et il n'est possible de travailler que depuis la console car les services réseaux ne sont pas démarrés. Pour démarrer en mode Single User, il suffit de taper à l'invite de démarrage de Lilo :

```
LILO : linux single
```

16.2. Disquette de rescue

La plupart des distributions propose de réaliser une disquette de démarrage lors de l'installation. Cette disquette peut s'avérer très utile en cas de problèmes sur le serveur ; il est donc fortement conseillé de la réaliser lors de l'installation. Il est toutefois possible d'en créer une après l'installation, grâce à la commande **mkbootdisk**. Pour réaliser une disquette de démarrage sur le lecteur de disquette par défaut (`/dev/fd0`), la commande est la suivante :

```
mkbootdisk version_noyau
```

Pour générer une disquette pour la version courante du noyau, la commande à taper sera la suivante :

```
mkbootdisk `uname -r`
```

Une fois cette disquette réalisée, il suffit de l'insérer dans le lecteur de disquettes et de (re)démarrer la machine. Si le Bios est correctement configuré, la machine va booter sur la disquette de sauvegarde et offrir un shell root, qui permettra de dépanner la machine.

16.3. Demolinux

Demolinux est une distribution Linux fournie sur un CD fonctionnant sans installation. L'avantage par rapport à une disquette de restauration, est que Demolinux propose un environnement graphique complet avec la plupart des outils courants.

Demolinux détecte automatiquement toutes les partitions disponibles sur le(s) disque(s) et les monte automatiquement. Il est donc trivial avec Demolinux de corriger des problèmes sur les systèmes de fichiers ou y copier des fichiers manquants.

Demolinux est disponible sur (http ://www.demolinux.org/)

16.4. Réparer son système

Une fois le système de restauration démarré, il est possible de réparer le système. Les problèmes les plus courants sont les suivants :

16.4.1. Systèmes de fichiers endommagés

Pour réparer un système de fichiers endommagés, il est nécessaire d'identifier en premier lieu la partition avec la commande :

```
fdisk -l /dev/hda
```

qui affichera la liste des partitions du premier disque IDE. Dans le cas d'un disque SCSI, il faudra utiliser le fichier spécial `/dev/sda`.

Une fois identifiée la partition endommagée, il faut la réparer avec la commande (si la partition en question est la première du premier disque IDE) :

```
fsck /dev/hda1
```

16.4.2. Mot de passe root oublié

Dans ce cas, le mieux est de booter en mode Single User puisque l'on obtient un shell en tant que root. Il est possible ensuite d'utiliser la commande **passwd** qui va demander d'entrer le nouveau mot de passe.

16.4.3. Restaurer une sauvegarde

Si la sauvegarde a été effectuée sur l'intégralité de l'arborescence, en utilisant un **tar** sur un lecteur de bande, les étapes pour la restauration sont les suivantes :

1. identifier la partition racine à l'aide de fdisk :

   ```
   fdisk -l /dev/xxx
   ```

2. créer un point de montage dans l'arborescence du système de secours :

   ```
   mkdir /mnt/racine
   ```

3. monter la partition racine

```
mount -t ext2 /dev/xxx /mnt/racine
```

4. se placer sur le point de montage :

```
cd /mnt/racine
```

5. restaurer la sauvegarde :

```
tar xvf /dev/st0
```

Cette procédure est à adapter à la situation : si la sauvegarde ne concerne qu'un répertoire particulier (`/home` par exemple), il faudra se positionner dans le répertoire (`/mnt/racine/home` par exemple) correspondant afin de restaurer la sauvegarde. A l'inverse, si l'on veut ne restaurer qu'une partie de l'arborescence, on la spécifiera à la commande tar :

```
tar xvf /dev/st0 home/
```