FORMATUX - Support de cours GNU/Linux

Administration CentOS 6

Version 1.0
17-04-2017

Table des matières

Préface

Table des matières

GNU/Linux est un **système d'exploitation** libre fonctionnant sur la base d'un **noyau Linux**, également appelé **kernel Linux**.

Linux est une implémentation libre du système **UNIX** et respecte les spécifications **POSIX**.

GNU/Linux est généralement distribué dans un ensemble cohérent de logiciels, assemblés autour du noyau Linux et prêt à être installé. Cet ensemble porte le nom de "**Distribution**".

- La plus ancienne des distributions est la distribution **Slackware**.
- Les plus connues et utilisées sont les distributions **Debian**, **Redhat** et **Arch**, et servent de base pour d'autres distributions comme **Ubuntu**, **CentOS**, **Fedora**, **Mageia** ou **Manjaro**.

Chaque distribution présente des particularités et peut être développée pour répondre à des besoins très précis :

- services d'infrastructure ;
- pare-feu ;
- serveur multimédia ;
- serveur de stockage ;
- etc.

La distribution présentée dans ces pages est la CentOS, qui est le pendant gratuit de la distribution RedHat. La distribution CentOS est particulièrement adaptée pour un usage sur des serveurs d'entreprises.

1. Crédits

Ce support de cours a été rédigé par les formateurs :

- Patrick Finet ;
- Antoine Le Morvan ;
- Xavier Sauvignon ;

2. Licence

3. Gestion des versions

Tableau 1. Historique des versions du document

Version	Date	Observations
1.0	Avril 2017	Version initiale.

1

Introduction

1.1. Qu’est-ce qu’un système d’exploitation ?

Linux est un **système d’exploitation**.

Un système d’exploitation est un **ensemble de programmes permettant la gestion des ressources disponibles d’un ordinateur**.

Parmi cette gestion des ressources, le système d’exploitation est amené à :

- Gérer la mémoire physique ou virtuelle.
    - La **mémoire physique** est composée des barrettes de mémoires vives et de la mémoire cache du processeur, qui sert pour l’exécution des programmes.
    - La **mémoire virtuelle** est un emplacement sur le disque dur (la partition **swap**) qui permet de décharger la mémoire physique et de sauvegarder l'état en cours du système durant l’arrêt électrique de l’ordinateur (hybernation du système).
- Intercepter les **accès aux périphériques**. Les logiciels ne sont que très rarement autorisés à accéder directement au matériel (à l’exception des cartes graphiques pour des besoins très spécifiques).
- Offrir aux applications une **gestion correcte des tâches**. Le système d’exploitation est responsable de l’ordonnancement des processus pour l’occupation du processeur.
- **Protéger les fichiers** contre tout accès non autorisé.

- **Collecter les informations** sur les programmes utilisés ou en cours d'utilisation.

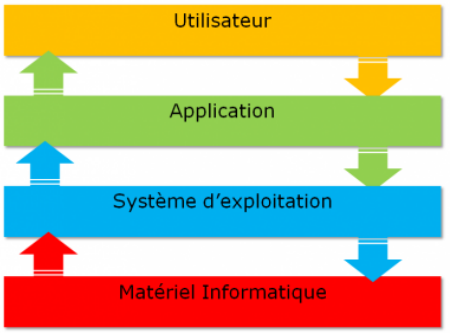


Figure 1.1. Fonctionnement d'un système d'exploitation

1.2. Généralités UNIX - GNU/Linux

1.2.1. Historique

UNIX

- De **1964 à 1968** : MULTICS (MULTiplexed Information and Computing Service) est développé pour le compte du MIT, des laboratoires Bell Labs (AT&T) et de General Electric.
- **1969** : Après le retrait de Bell (1969) puis de General Electric du projet, deux développeurs (Ken Thompson et Dennis Ritchie), rejoints plus tard par Brian Kernighan, jugeant MULTICS trop complexe, lancent le développement d'UNIX (UNiplexed Information and Computing Service). À l'origine développé en assembleur, les concepteurs d'UNIX ont développé le langage B puis le langage C (1971) et totalement réécrit UNIX. Ayant été développé en 1970, la date de référence des systèmes Unix/Linux est toujours fixée au 01 janvier 1970.

Le langage C fait toujours partie des langages de programmation les plus populaires aujourd'hui ! Langage de bas niveau, proche du matériel, il permet l'adaptation du système d'exploitation à toute architecture machine disposant d'un compilateur C.

Unix est un système d'exploitation ouvert et évolutif ayant joué un rôle primordial dans l'histoire de l'informatique. Il a servi de base pour de nombreux autres systèmes : Linux, BSD, Mac OSX, etc.

Unix est toujours d'actualité (HP-UX, AIX, Solaris, etc.)

Minix

- **1987** : Minix. A.S. Tanenbaum développe MINIX, un UNIX simplifié, pour enseigner les systèmes d'exploitation de façon simple. M. Tanenbaum rend disponible les sources de son système d'exploitation.

Linux

Figure 1.2. Linus Torvald, créateur du noyau Linux

- **1991** : Linux. Un étudiant finlandais, Linus Torvalds, crée un système d'exploitation dédié à son ordinateur personnel et le nomme Linux. Il publie sa première version 0.02, sur le forum de discussion Usenet et d'autres développeurs viennent ainsi l'aider à améliorer son système. Le terme Linux est un jeu de mot entre le prénom du fondateur, Linus, et Unix.
- **1993** : La distribution Debian est créée. Debian est une distribution non commerciale a gestion associative. À l'origine développée pour une utilisation sur des serveurs, elle est particulièrement bien adaptée à ce rôle, mais elle se veut être un système universel et donc utilisable également sur un ordinateur personnel. Debian est utilisée comme base pour de nombreuses autres distributions, comme Mint ou Ubuntu.
- **1994** : La distribution commerciale RedHat est créee par la société RedHat, qui est aujourd'hui le premier distributeur du système d'exploitation GNU/Linux.

RedHat soutient la version communautaire Fedora et depuis peu la distribution libre CentOS.

- **1997** : L'environnement de bureau KDE est créé. Il est basé sur la bibliothèque de composants Qt et sur le langage de développement C++.
- **1999** : L'environnement de bureau Gnome est créé. Il est quant à lui basé sur la bibliothèque de composants GTK+.
- **2002** : La distribution Arch est créée. Sa particularité est d'être diffusée en Rolling Release (mise à jour en continue).
- **2004** : Ubuntu est créée par la société Canonical (Mark Shuttleworth). Elle se base sur Debian, mais regroupe des logiciels libres et privateurs.

1.2.2. Parts de marché

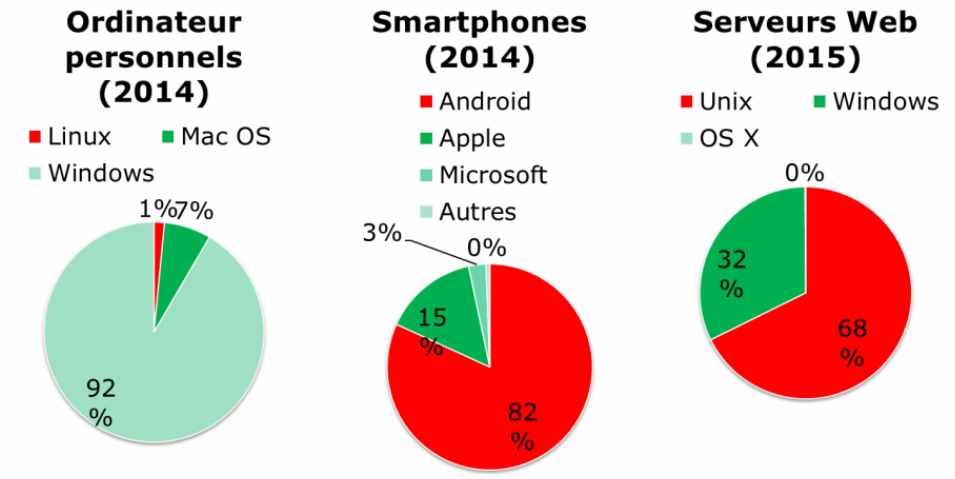


Figure 1.3. Les parts de marché de Linux

Linux est finalement encore peu connu du grand public, alors que ce dernier l'utilise régulièrement. En effet, Linux se cache dans les **smartphones**, les **téléviseurs**, les **box internet**, etc. Presque **70% des pages web** servies dans le monde le sont par un serveur Linux ou Unix !

Linux équipe un peu plus d'**1,5% des ordinateurs personnels** mais plus de **82% des smartphones**. **Android** étant un système d'exploitation dont le kernel est un Linux.

1.2.3. Architecture

- Le noyau (ou kernel) est le premier composant logiciel.
    - Il est le cœur du système UNIX.
    - C'est lui qui gère les ressources matérielles du système.
    - Les autres composants logiciels passent obligatoirement par lui pour accéder au matériel.
- Le Shell est un utilitaire qui interprète les commandes de l'utilisateur et assure leur exécution.
    - Principaux shell : Bourne shell, C shell, Korn shell et Bourne Again shell (bash).
- Les applications regroupent les programmes utilisateurs comme :
    - le navigateur internet ;
    - le traitement de texte ;
    - …

Multitâche

Linux fait partie de la famille des systèmes d'exploitation à temps partagé. Il partage le temps d'utilisation processus entre plusieurs programmes, passant de l'un à l'autre de façon transparente pour l'utilisateur. Cela implique :

- éxécution simultanée de plusieurs programmes ;
- distribution du temps CPU par l'ordonnanceur ;
- réduction des problèmes dus à une application défaillante ;
- diminution des performances lorsqu'il y a trop de programmes lancés.

Multiutilisateur

La finalité de Multics était de permettre à plusieurs utilisateurs de travailler à partir de plusieurs terminaux (écran et clavier) sur un seul ordinateur (très coûteux à l'époque). Linux étant un descendant de ce système d'exploitation, il a gardé cette capacité à pouvoir fonctionner avec plusieurs utilisateurs simultanément

et en toute indépendance, chacun ayant son compte utilisateur, son espace de mémoire et ses droits d'accès aux fichiers et aux logiciels.

Multiprocesseur

Linux est capable de travailler avec des ordinateurs multiprocesseurs ou avec des processeurs multicœurs.

Multiplateforme

Linux est écrit en langage de haut niveau pouvant s'adapter à différents types de plateformes lors de la compilation. Il fonctionne donc sur :

- les ordinateurs des particuliers (le PC ou l'ordinateur portable) ;
- les serveurs (données, applications,…) ;
- les ordinateurs portables (les smartphones ou les tablettes) ;
- les systèmes embarqués (ordinateur de voiture) ;
- les éléments actifs des réseaux (routeurs, commutateurs) ;
- les appareils ménagers (téléviseurs, réfrigérateurs,…).

Ouvert

Linux se base sur des standards reconnus (posix[1], TCP/IP, NFS, Samba …) permettant de partager des données et des services avec d'autres systèmes d'applications.

1.2.4. La philosophie Unix

- Tout est fichier.
- Portabilité.
- Ne faire qu'une seule chose et la faire bien.
- KISS : Keep It Simple and Stupid.
- "Unix est simple, il faut juste être un génie pour comprendre sa simplicité" (*Dennis Ritchie*)

[1] http://fr.wikipedia.org/wiki/POSIX

- “Unix est convivial. Cependant Unix ne précise pas vraiment avec qui.” (*Steven King*)

1.3. Les distributions GNU/LINUX

Une distribution Linux est un **ensemble cohérent de logiciels** assemblés autour du noyau Linux et prêt à être installé. Il existe des distributions **associatives ou communautaires** (Debian, CentOS) ou **commerciales** (RedHat, Ubuntu).

Chaque distribution propose un ou plusieurs **environnements de bureau**, fournit un ensemble de logiciels pré-installés et une logithèque de logiciels supplémentaires. Des options de configuration (options du noyau ou des services par exemple) sont propres à chacune.

Ce principe permet d’avoir des distributions orientées **débutants** (Ubuntu, Linux Mint …) ou d’une approche plus complexe (Gentoo, Arch), destinées à faire du **serveur** (Debian, Red Hat, …) ou dédiées à des **postes de travail**.

1.3.1. Les environnements de bureaux

Les environnements graphiques sont nombreux : **Gnome**, **Kde**, Lxde, Xfce, etc. Il y en a pour tous les goûts, et leurs **ergonomies** n’ont pas à rougir de ce que l’on peut retrouver sur les systèmes Microsoft ou Apple !

Alors pourquoi si peu d’engouement pour Linux, alors qu’il n**'existe pas (ou presque pas) de virus pour ce système** ? Parce que tous les éditeurs (Adobe) ou constructeur (NVidia) ne jouent pas le jeu du libre et ne fournissent pas de version de leurs logiciels ou de leurs drivers pour GNU/Linux. Trop peu de jeux également sont (mais plus pour longtemps) distribués sous Linux.

La donne changera-t-elle avec l’arrivée de la steam-box qui fonctionne elle aussi sous Linux ?

Figure 1.4. L'environnement de bureau Gnome

L'environnement de bureau **Gnome 3** n'utilise plus le concept de Bureau mais celui de Gnome Shell (à ne pas confondre avec le shell de la ligne de commande). Il sert à la fois de bureau, de tableau de bord, de zone de notification et de sélecteur de fenêtre. L'environnement de bureau Gnome se base sur la bibliothèque de compostants GTK+.

Figure 1.5. L'environnement de bureau Kde

L'environnement de bureau **KDE** se base sur la bibliothèque de composants **QT**.

Il est traditionnellement plus conseillé aux utilisateurs venant d'un monde Windows.

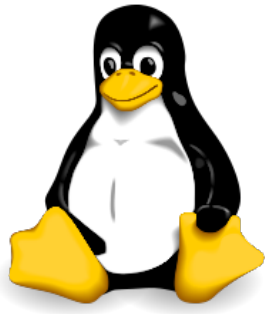


Figure 1.6. Tux, la mascotte Linux

1.3.2. Libre / Open source

Un utilisateur de système d'exploitation Microsoft ou Mac doit s'affranchir d'une licence d'utilisation du système d'exploitation. Cette licence a un coût, même s'il est généralement transparent (le prix de la licence étant inclus dans le prix de l'ordinateur).

Dans le monde GNU/Linux, le mouvement du Libre permet de fournir des distributions gratuites.

Libre ne veut pas dire gratuit !

Open source : les codes sources sont disponibles, il est donc possible de les consulter, de les modifier et de le diffuser.

Licence GPL (General Public License)

Cette licence garantit à l'auteur d'un logiciel sa propriété intellectuelle, mais autorise la modification, la rediffusion ou la revente de logiciels par des tiers, sous condition que les codes sources soient fournis avec le logiciel. La licence GPL est la licence issue du projet GNU (Gnu is Not Unix), projet déterminant dans la création de Linux.

Elle implique :

- la liberté d'exécuter le programme, pour tous les usages ;
- la liberté d'étudier le fonctionnement du programme et de l'adapter aux besoins ;
- la liberté de redistribuer des copies ;
- la liberté d'améliorer le programme et de publier vos améliorations, pour en faire profiter toute la communauté.

Par contre, même des produits sous licences GPL peuvent être payants. Ce n'est pas le produit en lui-même mais la garantie qu'une équipe de développeurs continue à travailler dessus pour le faire évoluer et dépanner les erreurs, voire fournir un soutien aux utilisateurs.

1.4. Les domaines d'emploi

Une distribution Linux excelle pour :

- **Un serveur** : HTTP, messagerie, groupware, partage de fichiers, etc.
- **La sécurité** : Passerelle, parefeu, routeur, proxy, etc.
- **Ordinateur central** : Banques, assurances, industrie, etc.
- **Système embarqué** : Routeurs, Box Internet, SmartTV, etc.

Linux est un choix adapté pour l'hébergement de base de données ou de sites web, ou comme serveur de messagerie, DNS, parefeu, firewall. Bref Linux peut à peu près tout faire, ce qui explique la quantité de distributions spécifiques.

1.5. Shell

1.5.1. Généralités

Le shell, interface de commandes en français, permet aux utilisateurs d'envoyer des ordres au système d'exploitation. Il est moins visible aujourd'hui, depuis la mise en place des interfaces graphiques, mais reste un moyen privilégié sur les systèmes Linux qui ne possèdent pas tous des interfaces graphiques et dont les services ne possèdent pas toujours une interface de réglage.

Il offre un véritable langage de programmation comprenant les structures classiques : boucles, alternatives et les constituants courants : variables, passage

de paramètres, sous-programmes. Il permet donc la création de scripts pour automatiser certaines actions (sauvegardes, création d'utilisateurs, surveillance du système,…).

Il existe plusieurs types de Shell disponibles et configurables sur une plateforme ou selon le choix préférentiel de l'utilisateur :

- sh, le shell aux normes POSIX ;
- csh, shell orienté commandes en C ;
- bash, Bourne Again Shell, shell de Linux.
- etc, …

1.6. Fonctionnalités

- Exécution de commandes (vérifie la commande passée et l'exécute) ;
- Redirections Entrées/Sorties (renvoi des données dans un fichier au lieu de l'inscrire sur l'écran) ;
- Processus de connexion (gère la connexion de l'utilisateur) ;
- Langage de programmation interprété (permettant la création de scripts) ;
- Variables d'environnement (accès aux informations propres au système en cours de fonctionnement).

1.6.1. Principe

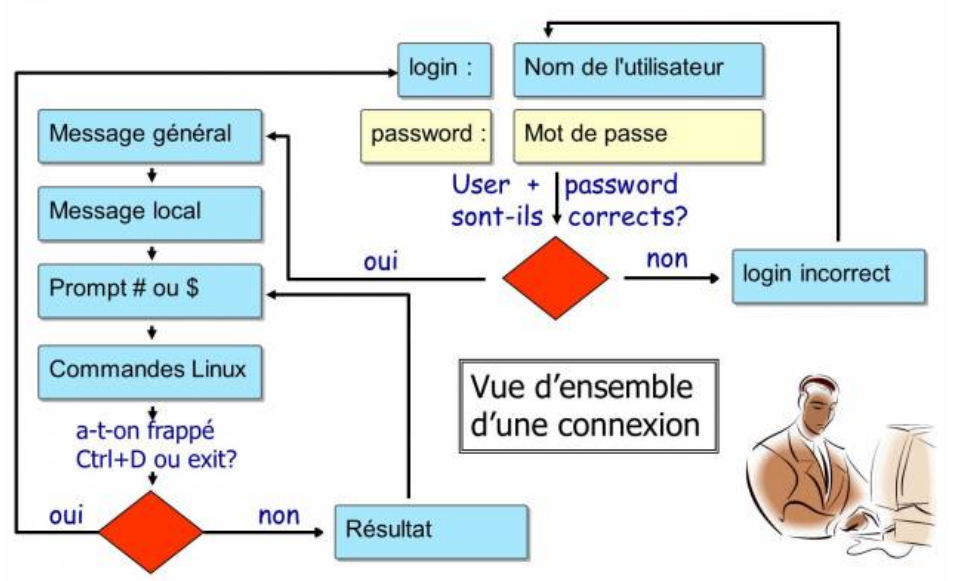


Figure 1.7. Principe de fonctionnement du SHELL

2

Commandes pour utilisateurs Linux

L'objectif de ce chapitre est d'apprendre aux futurs administrateurs à :

- se **déplacer** dans l'arborescence du système ;
- **créer** un fichier texte, **afficher** son contenu et le **modifier** ;
- utiliser les commandes les plus utiles de Linux.

2.1. Généralités

Les systèmes Linux actuels possèdent des utilitaires graphiques dédiés au travail d'un administrateur. Toutefois, il est important d'être capable d'utiliser l'interface en mode ligne de commandes et cela pour plusieurs raisons :

- La majorité des commandes du système sont communes à toutes les distributions Linux, ce qui n'est pas le cas des outils graphiques.
- Il peut arriver que le système ne démarre plus correctement mais qu'un shell de secours reste accessible.
- L'administration à distance se fait en ligne de commandes avec un terminal SSH.
- Afin de préserver les ressources du serveur, l'interface graphique n'est soit pas installée, soit lancée à la demande.
- L'administration se fait par des scripts.

L'apprentissage de ces commandes permet à l'administrateur de se connecter à un terminal Linux, de gérer ses ressources, ses fichiers, d'identifier la station, le terminal et les utilisateurs connectés, etc.

2.1.1. Les utilisateurs

L'utilisateur du système Linux est défini (dans le fichier /etc/passwd) par :

- un **nom de connexion**, plus communément appelé « login », ne contenant pas d'espace ;
- un identifiant numérique : **UID** (User Identifier) ;
- un identifiant de groupe : **GID** (Group Identifier) ;
- un **mot de passe**, qui sera chiffré avant d'être stocké ;
- un **interpréteur de commandes**, un Shell, qui peut être différent d'un utilisateur à l'autre ;
- un **répertoire de connexion**, le « home directory » ;
- un **prompt de connexion,** qui sera symbolisé par un **#** pour les administrateurs et un $ pour les autres utilisateurs.

En fonction de la politique de sécurité mise en œuvre sur le système, le mot de passe devra comporter un certain nombre de caractères et respecter des exigences de complexité.

Parmi les Shells existants, le **Bourne Again Shell** (/bin/bash) est le Shell le plus fréquemment utilisé. Il est affecté par défaut aux nouveaux utilisateurs. Pour diverses raisons, des utilisateurs avancés de Linux choisiront des Shells alternatifs parmi le Ksh, le Csh, etc.

Le répertoire de connexion de l'utilisateur est par convention stocké dans le répertoire /home du poste de travail. Il contiendra les données personnelles de l'utilisateur. Par défaut, à la connexion, le répertoire de connexion est sélectionné comme répertoire courant.

Une installation type poste de travail (avec interface graphique) démarre cette interface sur le terminal 1. Linux étant multi-utilisateurs, il est possible de connecter plusieurs utilisateurs plusieurs fois, sur des **terminaux physiques** (TTY) ou **virtuels** (PTS) différents. Les terminaux virtuels sont disponibles au sein d'un environnement graphique. Un utilisateur bascule d'un terminal physique à l'autre à l'aide des touches **Alt**+**Fx** depuis la ligne de commande ou à l'aide des touches **Ctrl**+**Alt**+**Fx**.

2.1.2. Le Shell

Une fois que l'utilisateur est connecté sur une console, le Shell affiche le prompt. Il se comporte ensuite comme une boucle infinie, à chaque saisie d'instruction :

- affichage du prompt ;
- lecture de la commande ;
- analyse de la syntaxe ;
- substitution des caractères spéciaux ;
- exécution de la commande ;
- affichage du prompt ;
- etc.

La séquence de touche **Ctrl**+**C** permet d'interrompre une commande en cours d'exécution.

L'utilisation d'une commande respecte généralement cette séquence :

Séquence d'une commande.

```
commande [option(s)] [arguments(s)]
```

Le nom de la commande est **toujours en minuscules**.

Un espace sépare chaque élément.

Les **options courtes** commencent par un tiret (-l), alors que les **options longues** commencent par deux tirets (--list). Un double tiret (--) indique la fin de la liste d'options. Il est possible de regrouper certaines options courtes :

Options courtes.

```
[root]# ls –l –i –a
```

est équivalent à :

Regroupement d'options.

```
[root]# ls -lia
```

Il peut bien entendu y avoir plusieurs arguments après une option :

Arguments d'une commande.

```
[root]# ls -lia /etc /home /var
```

Dans la littérature, le terme « option » est équivalent au terme « paramètre », plus utilisé dans le domaine de la programmation. Le côté optionnel d'une option ou d'un argument est symbolisé en le mettant entre crochets **[** et **]**. Lorsque plusieurs options sont possibles, une barre verticale appelée « pipe » les sépare [a**|**e**|**i].

2.2. Les commandes générales

2.2.1. Les commandes man et whatis

Il est impossible pour un administrateur, quel que soit son niveau, de connaître toutes les commandes et options dans les moindres détails. Une commande a été spécialement conçue pour accéder en ligne de commande à un ensemble d'aides, sous forme d'un manuel : la commande man (« le man est ton ami »).

Ce manuel est divisé en 8 sections, regroupant les informations par thème, la section par défaut étant la section 1 :

1. Commande utilisateur ;
2. Appels système ;
3. Fonctions de bibliothèque C ;
4. Périphériques et fichiers spéciaux ;
5. Formats de fichiers ;
6. Jeux ;
7. Divers ;
8. Outils d'administration système et démons.

Des informations sur chaque section sont accessibles en saisissant man x intro, x indiquant le numéro de section.

La commande :

Syntaxe de la commande man.

```
[root]# man passwd
```

informera l'administrateur sur la commande passwd, ses options, etc. Alors qu'un :

Syntaxe de la commande man avec section.

```
[root]# man 5 passwd
```

l'informera sur les fichiers en relations avec la commande.

Toutes les pages du manuel ne sont pas traduites de l'anglais. Elles sont toutefois généralement très précises et fournissent toutes les informations utiles. La syntaxe utilisée et le découpage peut dérouter l'administrateur débutant, mais avec de la pratique, l'administrateur y retrouvera rapidement l'information qu'il recherche.

La navigation dans le manuel se fait avec les flèches « Haut » et « Bas ». Le manuel se quitte en appuyant sur la touche « q ».

La commande **whatis** permet de faire une recherche par mot clef au sein des pages de manuel :

Syntaxe de la commande whatis.

```
[root]# whatis clear
```

2.2.2. La commande shutdown

La commande **shutdown** permet de **stopper électriquement**, immédiatement ou après un certain laps de temps, un serveur Linux.

Syntaxe de la commande shutdown.

```
[root]# shutdown [-h] [-r] heure [message]
```

L'heure d'arrêt est à indiquer au format **hh:mm** pour une heure précise, ou **+mm** pour un délai en minutes.

Pour forcer un arrêt immédiat, le mot « **now** » remplacera l'heure. Dans ce cas, le message optionnel n'est pas envoyé aux autres utilisateurs du système.

Exemples

Exemples de la commande shutdown.

```
[root]# shutdown -h 0:30 "Arrêt du serveur à 0h30"
[root]# shutdown -r +5
```

Options

Tableau 2.1. Options de la commande shutdown

Options	Observations
-h	Arrête le système électriquement
-r	Redémarre le système

2.2.3. La commande history

La commande **history** permet d'afficher l'historique des commandes qui ont été saisies par l'utilisateur.

Les commandes sont mémorisées dans le fichier **.bash_history** du répertoire de connexion de l'utilisateur.

Exemple de commande history.

```
[root]# history
  147 man ls
  148 man history
```

Tableau 2.2. Options de la commande history

Options	Commentaires
-w	L'option –w permet d'y copier l'historique de la session en cours.
-c	L'option –c effacera l'historique de la session en cours (mais pas le contenu du fichier .bash_history).

Manipuler l'historique

Pour manipuler l'historique, des commandes permettent depuis le prompt de :

Touches	Fonction
!!	Rappeler la dernière commande passée.
!n	Rappeler la commande par son numéro dans la liste.
!string	Rappeler la commande la plus récente commençant par la chaîne de caractères.
[↑]	Remonter l'historique des commandes.
[↓]	Redescendre l'historique des commandes.

L'auto-complétion

L'auto-complétion est également d'une aide précieuse.

- Elle permet de compléter les commandes, les chemins saisis ou les noms de fichiers.
- Un appui sur la touche **[TAB]** complète la saisie dans le cas d'une seule solution.
- Sinon, il faudra faire un deuxième appui pour obtenir la liste des possibilités.

Si un double appui sur la touche [TAB] ne provoque aucune réaction de la part du système, c'est qu'il n'existe aucune solution à la complétion en cours.

2.3. Affichage et identification

2.3.1. La commande clear

La commande clear permet d'effacer le contenu de l'écran du terminal. En réalité, pour être plus précis, elle permet de décaler l'affichage de sorte que le prompt se retrouve en haut de l'écran sur la première ligne.

Dans un terminal, l'affichage sera définitivement masqué tandis que dans une interface graphique, un ascenseur permettra de remonter dans l'historique du terminal virtuel.

2.3.2. La commande echo

La commande echo permet d'afficher une chaîne de caractères.

Cette commande est plus particulièrement utilisée dans les scripts d'administration pour informer l'utilisateur pendant l'exécution.

L'option –n permet de ne pas revenir à la ligne après avoir affiché le texte (ce qui est le comportement par défaut de la commande).

Pour diverses raisons, le développeur du script peut être amené à utiliser des séquences spéciales (commençant par un caractère \). Dans ce cas, l'option –e sera stipulée, permettant l'interprétation des séquences.

Parmi les séquences fréquemment utilisées, nous citerons :

Tableau 2.3. Séquences spéciales de la commande echo

Séquence	Résultat
\a	Émet un bip sonore
\b	Retour en arrière
\n	Ajoute un saut de ligne
\t	Ajoute une tabulation horizontale
\v	Ajoute une tabulation verticale

2.3.3. La commande date

La commande date permet d'afficher la date et l'heure. La commande respecte la syntaxe suivante :

Syntaxe de la commande date.

```
[root]# date [-d AAAAMMJJ] [format]
```

Exemples :

```
[root]# date
mer. Avril 17 16:46:53 CEST 2013
[root]# date –d 20150729 +%j
210
```

Dans ce dernier exemple, l'option –d affiche une date donnée. L'option +%j formate cette date pour n'afficher que le quantième.

Attention : Le format d'une date peut changer suivant la valeur de la langue définie dans la variable d'environnement $LANG.

L'affichage de la date peut suivre les formats suivants :

Tableau 2.4. Formats de la commande date

Option	Format
+%A	Nom complet du jour
+%B	Nom complet du mois
+%c	Affichage complet de la date
+%d	Numéro du jour
+%F	Date au format AAAA-MM-JJ
+%G	Année
+%H	Heure
+%j	Quantième du jour
+%m	Numéro du mois
+%M	Minute
+%R	Heure au format hh:mm
+%s	Secondes depuis le 1er janvier 1970
+%T	Heure au format hh:mm:ss
+%u	Jour de la semaine (1 pour lundi)
+%V	Numéro de la semaine
+%x	Date au format JJ/MM/AAAA

La commande date permet également de modifier la date et l'heure système. Dans ce cas, l'option –s sera utilisée.

```
[root]# date –s "2013-04-17 10:19"
jeu. Avril 17 10:19:00 CEST 2013
```

Le format à respecter pour l'argument suivant l'option –s est celui-ci :

```
date -s "[AA]AA-MM-JJ hh:mm:[ss]"
```

2.3.4. Les commandes id, who et whoami

La commande id affiche le nom de l'utilisateur courant et ses groupes ou ceux d'un utilisateur, si le login de celui-ci est fourni comme argument.

```
[root]# id util1
uid=501(util1) gid=501(group1) groups=501(group1),502(group2)
```

Les options –g, -G, -n et –u affiche respectivement le GID du groupe principal, les GID des groupes secondaires, les noms au lieu des identifiants numériques et l'UID de l'utilisateur.

La commande whoami affiche le login de l'utilisateur courant.

La commande who seule affiche le nom des utilisateurs connectés :

```
[root]# who
root tty1 2014-09-15 10:30
root pts/0 2014-09-15 10:31
```

Linux étant multi-utilisateurs, il est probable que plusieurs sessions soient ouvertes sur la même station, que ce soit physiquement ou à travers le réseau. Il est intéressant de savoir quels utilisateurs sont connectés, ne serait-ce que pour communiquer avec eux par l'envoi de messages.

- tty : représente un terminal.
- pts/ : représente une console virtuelle sous environnement graphique.

L'option « -r » affiche en plus le niveau d'exécution (voir chapitre « démarrage »).

2.4. Arborescence de fichiers

Sous Linux, l'arborescence des fichiers se présente sous la forme d'un arbre inversé, appelé **arborescence hiérarchique unique**, dont la racine est le répertoire « / ».

Le **répertoire courant** est le répertoire où se trouve l'utilisateur.

Le **répertoire de connexion** est le répertoire de travail associé à l'utilisateur. Les répertoires de connexion sont, en standard, stockés dans le répertoire **/home**.

À la connexion de l'utilisateur, le répertoire courant est le répertoire de connexion.

Un **chemin absolu** référence un fichier depuis la racine en parcourant l'arborescence complète jusqu'au niveau du fichier :

- /home/groupeA/alice/monfichier

Le **chemin relatif** référence ce même fichier en parcourant l'arborescence complète depuis le répertoire courant :

- ../alice/monfichier

Dans l'exemple précèdent, les "**..**" font référence au répertoire parent du répertoire actuel.

Un répertoire, même s'il est vide, contiendra obligatoirement au minimum **deux références** :

- « **.** » : référence sur lui-même.
- « **..** » : référence le répertoire parent du répertoire actuel.

Un chemin relatif peut ainsi commencer par « ./ » ou par « ../ ». Lorsque le chemin relatif fait référence à un sous dossier ou à un fichier du répertoire courant, alors le « ./ » est souvent omis. Mentionner le premier « ./ » de l'arborescence ne sera réellement requis que pour lancer un fichier exécutable.

Les erreurs dans les chemins peuvent être la cause de nombreux problèmes : création de dossier ou de fichiers aux mauvais endroits, suppressions involontaires, etc. Il est donc fortement recommandé d'utiliser l'auto-complétion (cf. 2.2) lors des saisies de chemin.

Figure 2.1. Notre arborescence exemple

Dans l'exemple ci-dessus, nous cherchons à donner l'emplacement du fichier monfichier depuis le répertoire de bob.

- Par un **chemin absolu**, le répertoire courant importe peu. Nous commençons par la racine, pour descendre successivement dans les répertoires "home", "groupeA", "alice" et enfin le fichier "monfichier" : */home/groupeA/alice/monfichier*.
- Par un **chemin relatif**, notre point de départ étant le répertoire courant "bob", nous remontons d'un niveau par ".." (soit dans le répertoire groupeA), puis nous descendons dans le répertoire "alice", et enfin le fichier "monfichier" : *../alice/monfichier*.

2.4.1. La commande pwd

La commande pwd (Print Working Directory) affiche le chemin absolu du répertoire courant.

```
[root]# pwd
/root/instructeur
```

Pour se déplacer à l'aide d'un chemin relatif, il faut impérativement connaître son positionnement dans l'arborescence.

Selon le shell, le prompt peut également afficher le nom du répertoire courant.

2.4.2. La commande cd

La commande cd (Change Directory) permet de changer le répertoire courant, autrement dit, de se déplacer dans l'arborescence.

```
[root]# cd /tmp
[root]# pwd
/tmp
[root]# cd ../
[root]# pwd
/
[root]# cd
[root]# pwd
/root
```

Comme vous pouvez le constater dans le dernier exemple ci-dessus, la commande cd sans argument permet de repositionner le répertoire courant sur le répertoire de connexion (home directory).

2.4.3. La commande ls

La commande ls affiche le contenu d'un répertoire.

Syntaxe de la commande ls.

```
ls [-a] [-i] [-l] [repertoire1] [repertoire2] […]
```

Exemple :

```
[root]# ls /home
. .. STAGE
```

Options

Les options principales de la commande ls sont :

Tableau 2.5. Options principales de la commande ls

Option	Information
-a	Affiche tous les fichiers, même ceux cachés. Les fichiers cachés sous Linux sont ceux qui commencent par un ".".
-i	Affiche les numéros d'inode.
-l	Affiche sous forme de liste verticale la liste des fichiers avec des informations supplémentaires formatées par colonnes.

La commande ls offre toutefois de très nombreuses options (voir le man) :

Tableau 2.6. Options complémentaires de la commande ls

Option	Information
-d	Affiche les informations d'un répertoire au lieu de lister son contenu.
-g	Affiche les UID et GID plutôt que les noms des propriétaires.

Option	Information
-h	Affiche les tailles de fichiers dans le format le plus adapté (octet, kilo-octet, méga-octet, giga-octet, …). h pour Human Readable.
-s	Affiche la taille en octets (sauf si option k).
-A	Affiche tous les fichiers du répertoire sauf “.” et “..”.
-R	Affiche récursivement le contenu des sous répertoires.
-F	Affiche le type des fichiers. Imprime un / pour un répertoire, * pour les exécutables, @ pour un lien symbolique, et rien pour un fichier texte.

Description des colonnes

```
[root]# ls -lia /root
78489 drwxr-xr-x 4 root root 4096 25 oct. 08:10 STAGE
```

Tableau 2.7. Description des colonnes du résultat généré par la commande ls

Valeur	Information.
78489	Numéro d’inode.
drwxr-xr-x	Type de fichier (d) et droits (rwxr-xr-x).
4	Nombre de sous-répertoires (“.” et “..” inclus). Pour un fichier de type lien physique : nombre de liens physiques.
root	Utilisateur propriétaire.
root	Groupe propriétaire.
4096	Taille en octets.
25 oct. 08:10	Date de dernière modification.
STAGE	Nom du fichier (ou du répertoire).

Alias

Des alias sont fréquemment positionnés au sein des distributions courantes.

C’est le cas de l’alias ll :

Alias de la commande ls -l.

```
alias ll='ls -l --color=auto'
```

Utilisations avancées

- Lister les fichiers de **/etc** par ordre de dernière modification :

```
[root]# ls -ltr /etc
total 1332
-rw-r--r--.  1 root root    662 29 aout   2007 logrotate.conf
-rw-r--r--.  1 root root    272 17 nov.   2009 mailcap
-rw-------.  1 root root    122 12 janv.  2010 securetty
...
-rw-r--r--.  2 root root     85 18 nov.  17:04 resolv.conf
-rw-r--r--.  1 root root     44 18 nov.  17:04 adjtime
-rw-r--r--.  1 root root    283 18 nov.  17:05 mtab
```

- Lister les fichiers de **/var** plus gros qu'un mega-octet mais moins qu'un giga-octets :

```
[root]# ls -Rlh /var | grep [0-9]M
...
-rw-r--r--. 1 apache apache 1,2M 10 nov.  13:02 XB RiyazBdIt.ttf
-rw-r--r--. 1 apache apache 1,2M 10 nov.  13:02 XB RiyazBd.ttf
-rw-r--r--. 1 apache apache 1,1M 10 nov.  13:02 XB RiyazIt.ttf
...
```

- Afficher les droits sur un dossier :

Pour connaître les droits sur un dossier, dans notre exemple /etc, la commande suivante ne conviendrait pas :

```
[root]# ls -l /etc
total 1332
-rw-r--r--.  1 root root     44 18 nov.  17:04 adjtime
-rw-r--r--.  1 root root   1512 12 janv.  2010 aliases
-rw-r--r--.  1 root root  12288 17 nov.  17:41 aliases.db
drwxr-xr-x.  2 root root   4096 17 nov.  17:48 alternatives
...
```

puisque cette dernière liste par défaut le contenu du dossier et non le contenant.

Pour ce faire, il faut utiliser l'option -d :

```
[root]# ls -ld /etc
drwxr-xr-x. 69 root root 4096 18 nov.  17:05 /etc
```

- Lister les fichiers par taille :

```
[root]# ls -lhS
```

- Afficher la date de modification au format "timestamp" :

```
[root]# ls -l --time-style="+%Y-%m-%d $newline%m-%d %H:%M"
total 12378
dr-xr-xr-x. 2 root root 4096 2014-11-23 11-23 03:13 bin
dr-xr-xr-x. 5 root root 1024 2014-11-23 11-23 05:29 boot
```

- Ajouter le "trailing slash" à la fin des dossiers :

Par défaut, la commande ls n'affiche pas le dernier slash d'un dossier.

Dans certains cas, comme pour des scripts par exemple, il est utile de les afficher :

```
[root]# ls -dF /etc
/etc/
```

2.4.4. La commande mkdir

La commande mkdir crée un répertoire ou une arborescence de répertoire.

Syntaxe de la commande mkdir.

```
mkdir [-p] repertoire [repertoire] […]
```

Exemple :

```
[root]# mkdir /home/STAGE/travail
```

Le répertoire « STAGE » devra exister pour créer le répertoire « travail ».

Sinon, l'option « –p » devra être utilisée. L'option « –p » crée les répertoires parents s'ils n'existent pas.

Il est vivement déconseillé de donner des noms de commandes UNIX comme noms de répertoires ou fichiers.

2.4.5. La commande touch

La commande touch modifie l'horodatage d'un fichier ou crée un fichier vide si le fichier n'existe pas.

Syntaxe de la commande touch.

```
touch [-t date] fichier
```

Exemple :

```
[root]# touch /home/STAGE/fichier
```

Option	Information
-t date	Modifie la date de dernière modification du fichier avec la date précisée. Date au format : [AAAA]MMJJhhmm[ss]

La commande touch est utilisée en priorité pour créer un fichier vide, mais elle peut avoir un intérêt dans le cadre de sauvegarde incrémentale ou différentielle. En effet, le fait d'exécuter un touch sur un fichier aura pour seul effet de forcer sa sauvegarde lors de la sauvegarde suivante.

2.4.6. La commande rmdir

La commande rmdir supprime un répertoire vide.

Exemple :

```
[root]# rmdir /home/STAGE/travail
```

Option	Information
-p	Supprime le ou les répertoire(s) parent(s) à la condition qu'ils soient vides.

Pour supprimer à la fois un répertoire non-vide et son contenu, il faudra utiliser la commande rm.

2.4.7. La commande rm

La commande rm supprime un fichier ou un répertoire.

Syntaxe de la commande rm.

```
rm [-f] [-r] fichier [fichier] […]
```

ATTENTION !!! Toute suppression de fichier ou de répertoire est définitive.

Tableau 2.8. Options de la commande rm

Options	Information
-f	Ne demande pas de confirmation de la suppression.
-i	Demande de confirmation de la suppression.
-r	Supprime récursivement les sous-répertoires.

La commande rm en elle-même ne demande pas de confirmation lors de la suppression de fichiers. Ce comportement est propre à la distribution RedHat/CentOS.

La commande rm est ici un alias de la commande rm -i. Ne soyez pas surpris sur une autre distribution, type Debian par exemple, de ne pas obtenir de demande de confirmation.

La suppression d'un dossier à l'aide de la commande rm, que ce dossier soit vide ou non, nécessitera l'ajout de l'option –r.

La fin des options est signalée au shell par un double tiret "--".

Dans l'exemple :

```
[root]# >-dur-dur # Creer un fichier vide appelé -dur-dur
[root]# rm -f -- -dur-dur
```

Le nom du fichier -dur-dur commence par un "-". Sans l'usage du "--" le shell aurait interprété le "-d" de "-dur-dur" comme une option.

2.4.8. La commande mv

La commande mv déplace et renomme un fichier.

Syntaxe de la commande mv.

```
mv fichier [fichier …] destination
```

Exemples :

```
[root]# mv /home/fic1 /home/fic2
[root]# mv /home/fic1 /home/fic2 /tmp
```

Tableau 2.9. Options de la commande mv

Options	Information
-f	Ne demande pas de confirmation si écrasement du fichier de destination.
-i	Demande de confirmation si écrasement du fichier de destination (par défaut).

Cas concrets

```
[root]# mv /home/fic1 /home/fic2
```

Permet de renommer "fic1" en "fic2", si "fic2" existe déjà, il sera remplacé par "fic1".

```
[root]# mv /home/fic1 /home/fic2 /tmp
```

Permet de déplacer "fic1" et "fic2" dans le répertoire "/tmp".

```
[root]# mv fic1 /repexiste/fic2
```

« fic1 » est déplacé dans « /repexiste » et renommé « fic2 ».

```
[root]# mv fic1 fic2
```

« fic1 » est renommé « fic2 ».

```
[root]# mv fic1 /repexiste
```

Si le répertoire de destination existe, « fic1 » est déplacé dans « /repexiste ».

```
[root]# mv fic1 /repexistepas
```

Si le répertoire de destination n'existe pas, « fic1 » est renommé « repexistepas » à la racine.

2.4.9. La commande cp

La commande cp copie un fichier.

Syntaxe de la commande cp.

```
cp fichier [fichier …] destination
```

Exemple :

```
[root]# cp -r /home/STAGE /tmp
```

Tableau 2.10. Options de la commande cp

Options	Information
-i	Demande de confirmation si écrasement (par défaut).
-f	Ne demande pas de confirmation si écrasement du fichier de destination.
-p	Conserve le propriétaire, les permissions et l'horodatage du fichier copié.

Options	Information
-r	Copie un répertoire avec ses fichiers et sous-répertoires.

Cas concrets

```
[root]# cp fic1 /repexiste/fic2
```

« fic1 » est copié dans « /repexiste » sous le nom « fic2 ».

```
[root]# cp fic1 fic2
```

« fic1 » est copié sous le nom « fic2 » dans ce répertoire.

```
[root]# cp fic1 /repexiste
```

Si le répertoire de destination existe, « fic1 » est copié dans « /repexiste ».

```
[root]# cp fic1 /repexistepas
```

Si le répertoire de destination n'existe pas, « fic1 » est copié sous le nom « repexistepas ».

2.5. Visualisation

2.5.1. La commande file

La commande file affiche le type d'un fichier.

Syntaxe de la commande file.

```
file fichier [fichiers]
```

Exemple :

```
[root]# file /etc/passwd /etc
/etc/passwd: ASCII text
/etc:  directory
```

2.5.2. La commande more

La commande more affiche le contenu d'un ou de plusieurs fichiers écran par écran.

Syntaxe de la commande more.

```
more fichier [fichiers]
```

Exemple :

```
[root]# more /etc/passwd
root:x:0:0:root:/root:/bin/bash
...
```

En utilisant la touche [ENTREE], le déplacement se fait ligne par ligne. En utilisant la touche [ESPACE], le déplacement se fait page par page.

2.5.3. La commande less

La commande less affiche le contenu d'un ou de plusieurs fichiers. La commande less est interactive et possède des commandes d'utilisation qui lui sont propres.

Syntaxe de la commande less.

```
less fichiers [fichiers]
```

Les commandes propres à less sont :

Tableau 2.11. Commandes internes à less

Commande	Action
h	Aide.
Flèches	Monter, descendre d'une ligne ou pour aller à droite ou à gauche.
Entrée	Descendre d'une ligne.
Espace	Descendre d'une page.
PgAR ou PgAV	Monter ou descendre d'une page.

Commande	Action
Pos1 ou Fin	Se placer en début de fichier ou en fin de fichier.
/texte	Rechercher le texte.
q	Quitter la commande less.

2.5.4. Les commandes cat et tac

La commande cat

La commande cat concatène (mettre bout à bout) le contenu de plusieurs fichiers et affiche le résultat sur la sortie standard.

Syntaxe de la commande cat.

```
cat fichier [fichiers]
```

Exemple 1 - Afficher le contenu d'un fichier vers la sortie standard :

```
[root]# cat /etc/passwd
```

Exemple 2 - Afficher le contenu de plusieurs fichiers vers la sortie standard :

```
[root]# cat /etc/passwd /etc/group
```

Exemple 3 - Afficher le contenu de plusieurs fichiers et rediriger la sortie standard :

```
[root]# cat /etc/passwd /etc/group > utilisateursEtGroupes.txt
```

Exemple 4 - Afficher la numérotation des lignes :

```
[root]# cat -n /etc/passwd
 1 root:x:0:0:root:/root:/bin/bash
 2 bin:x:1:1:bin:/bin:/sbin/nologin
...
```

Exemple 5 - Affiche la numérotation des lignes non vides :

```
[root]# cat -b /etc/openldap/ldap.conf
```

```
1 #
2 # LDAP Defaults
3 #

4 # See ldap.conf(5) for details
5 # This file should be world readable but not world writable
```

La commande tac

La commande tac fait quasiment l'inverse de la commande cat. Elle affiche le contenu d'un fichier en commencant par la fin (ce qui est particulièrement intéressant pour la lecture des logs !).

Exemple : Afficher un fichier de logs en affichant en premier la dernière ligne :

```
[root]# tac /var/log/messages | less
```

2.5.5. La commande head

La commande head affiche le début d'un fichier.

Syntaxe de la commande head.

```
head [-n x] fichier
```

Tableau 2.12. Options de la commande head

Option	Observation
-n x	Affiche les x premières lignes du fichier

Par défaut (sans l'option –n), la commande head affichera les 10 premières lignes du fichier.

2.5.6. La commande tail

La commande tail affiche la fin d'un fichier.

Syntaxe de la commande tail.

```
tail [-f] [-n x] fichier
```

Tableau 2.13. Options de la commande tail

Option	Observation
-n x	Affiche les x dernières lignes du fichier
-f	Affiche les modifications du fichier en temps réel

Exemple :

```
[root]# tail -n 3 /etc/passwd
sshd:x:74:74:Privilege-separeted sshd:/var/empty /sshd:/sbin/nologin
tcpdump::x:72:72::/:/sbin/nologin
user1:x:500:500:grp1:/home/user1:/bin/bash
```

Avec l'option -f, la commande tail ne rend pas la main et s'exécute tant que l'utilisateur ne l'interrompt pas par la séquence [CTRL] + [C]. Cette option est très fréquemment utilisée pour suivre les fichiers journaux (les logs) en temps réel.

Sans l'option -n, la commande tail affiche les 10 dernières lignes du fichier.

2.5.7. La commande sort

La commande sort trie les lignes d'un fichier.

Elle permet d'ordonner, ranger dans un ordre donné, le résultat d'une commande ou le contenu d'un fichier, selon un ordre numérique, alphabétique, par ordre de grandeur (Ko, Mo, Go) ou dans l'ordre inverse.

Syntaxe de la commande sort.

```
sort [-kx] [-n] [-o fichier] [-ty] fichier
```

Exemple :

```
[root]# sort -k3 -t: -n /etc/passwd
root:x:0:0:root:/root:/bin/bash
adm:x:3:4:adm:/var/adm/:/sbin/nologin
```

Tableau 2.14. Options de la commande sort

Option	Observation
-kx	Précise la colonne x sur laquelle se fera le tri

Option	Observation
-n	Demande un tri numérique
-o fichier	Enregistre le tri dans le fichier précisé
-ty	Précise le caractère séparateur de champs y
-r	Inverse l'ordre du résultat

La commande sort ne trie le fichier qu'à l'affichage écran. Le fichier n'est pas modifié par le tri. Pour enregistrer le tri, il faut utiliser l'option –o ou une redirection de sortie >.

Par défaut, le tri des nombres se fait selon leur caractère. Ainsi, "110" sera avant "20", qui sera lui-même avant "3". Il faut préciser l'option –n pour que les blocs caractères numériques soient bien triés par leur valeur.

Inverser l'ordre des résultats

La commande sort permet d'inverser l'ordre des résultats, avec l'option -r :

```
[root]# sort -k3 -t: -n -r /etc/passwd
root:x:0:0:root:/root:/bin/bash
adm:x:3:4:adm:/var/adm/:/sbin/nologin
```

rangera cette fois-ci le contenu du fichier /etc/passwd du plus grand uid au plus petit.

Mélanger les valeurs

La commande sort permet également de mélanger les valeurs avec l'option -R :

```
[root]# sort -R /etc/passwd
```

Trier des adresses IP

Un administrateur système est rapidement confronté au traitement des adresses IP issues des logs de ses services comme SMTP, VSFTP ou Apache. Ces adresses sont typiquement extraites avec la commande cut.

Voici un exemple avec le fichier client-dns.txt :

```
192.168.1.10
192.168.1.200
5.1.150.146
208.128.150.98
208.128.150.99
```

```
[root]# sort -nr client-dns.txt
208.128.150.99
208.128.150.98
192.168.1.200
192.168.1.10
5.1.150.146
```

Trier des tailles de fichiers

Sort sait reconnaître les tailles de fichiers, issues de commande comme ls avec l'option -h.

Voici un exemple avec le fichier taille.txt :

```
1,7G
18M
69K
2,4M
1,2M
4,2G
6M
124M
12,4M
4G
```

```
[root]# sort -hr taille.txt
4,2G
4G
1,7G
124M
18M
12,4M
6M
2,4M
1,2M
```

```
69K
```

2.5.8. La commande wc

La commande wc compte le nombre de lignes, mots ou octets d'un fichier.

Syntaxe de la commande wc.

```
wc [-l] [-m] [-w] fichier [fichiers]
```

Tableau 2.15. Options de la commande wc

Option	Observation
-c	Compte le nombre d'octets.
-m	Compte le nombre de caractères.
-l	Compte le nombre de lignes.
-w	Compte le nombre de mots.

2.6. Recherche

2.6.1. La commande find

La commande find recherche l'emplacement d'un fichier.

Syntaxe de la commande find.

```
find repertoire [-name nom] [-type type] [-user login] [-date date]
```

Les options de la commande find étant très nombreuses, il est préférable de se référer au man.

Si le répertoire de recherche n'est pas précisé, la commande find cherchera à partir du répertoire courant.

Tableau 2.16. Options de la commande find

Option	Observation
-perm permissions	Recherche des fichiers selon leurs permissions.

Option	Observation
-size taille	Recherche des fichiers selon leur taille.

L'option -exec

Il est possible d'utiliser l'option –exec pour exécuter une commande à chaque ligne de résultat :

```
[root]# find /tmp –name *.log –exec rm –f {} \;
```

La commande précédente recherche tous les fichiers du répertoire /tmp nommés *.log et les supprime.

Comprendre l'option -exec

Dans l'exemple ci-dessus, la commande find va construire une chaîne de caractères représentant la commande à exécuter.

Si la commande find trouve trois fichiers nommés log1.log, log2.log et log3.log, alors la commande find va construire la chaîne en remplaçant dans la chaîne "rm -f {} \;" les accolades par un des résultats de la recherche, et cela autant de fois qu'il y a de résultats.

Ce qui nous donnera :

```
rm -f /tmp/log1 ; rm -f /tmp/log2 ; rm -f /tmp/log3 ;
```

Le caractère ";" est un caractère spécial du Shell qui doit être protégé par un "\" pour éviter son interprétation trop tôt par la commande find (et non plus dans le exec).

2.6.2. La commande whereis

La commande whereis recherche des fichiers liés à une commande.

Syntaxe de la commande whereis.

```
whereis [-b] [-m] [-s] commande
```

Exemple :

```
[root]# whereis -b ls
ls: /bin/ls
```

Tableau 2.17. Options de la commande whereis

Option	Observation
-b	Ne recherche que le fichier binaire.
-m	Ne recherche que les pages de manuel.
-s	Ne recherche que les fichiers sources.

2.6.3. La commande grep

La commande grep recherche une chaîne de caractères dans un fichier.

Syntaxe de la commande grep.

```
grep [-w] [-i] [-v] "chaîne" fichier
```

Exemple :

```
[root]# grep -w "root:" /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

Tableau 2.18. Options de la commande grep

Option	Observation
-i	Ignore la casse de la chaîne de caractères recherchée.
-v	Inverse le résultat de la recherche.
-w	Recherche exactement la chaîne de caractères précisée.

La commande grep retourne la ligne complète comprenant la chaîne de caractères recherchée.

- Le caractère spécial ^ permet de rechercher une chaîne de caractères placée en début de ligne.
- Le caractère spécial $ permet de rechercher une chaîne de caractères placée en fin de ligne.

```
[root]# grep -w "^root" /etc/passwd
```

Cette commande est très puissante et il est fortement conseillé de consulter son manuel. Elle a de nombreux dérivés,

Recherche récursive

Il est possible de rechercher une chaîne de caractères dans une arborescence de fichiers avec l'option -R.

```
[root]# grep -R "Virtual" /etc/httpd
```

2.6.4. Les méta-caractères

Les méta-caractères se substituent à un ou plusieurs caractères (voire à une absence de caractère) lors d'une recherche.

Ils sont combinables.

Le caractère * remplace une chaîne composée de plusieurs caractères quelconques. Le caractère * peut également représenter une absence de caractère.

```
[root]# find /home -name test*
/home/test
/home/test1
/home/test11
/home/tests
/home/test362
```

Les méta-caractères permettent des recherches plus complexes en remplaçant tout ou partie d'un mot. Il suffit de remplacer les inconnues par ces caractères spéciaux.

Le caractère "?" remplace un unique caractère, quel qu'il soit.

```
[root]# find /home -name test?
/home/test1
```

```
/home/tests
```

Les crochets “[ ]” permettent de spécifier les valeurs que peut prendre un unique caractère.

```
[root]# find /home -name test[123]*
/home/test1
/home/test11
/home/test362
```

Il ne faut pas confondre les méta-caractères du shell et ceux des expressions régulières. La commande grep utilise les méta-caractères des expressions régulières.

2.7. Redirections et tubes

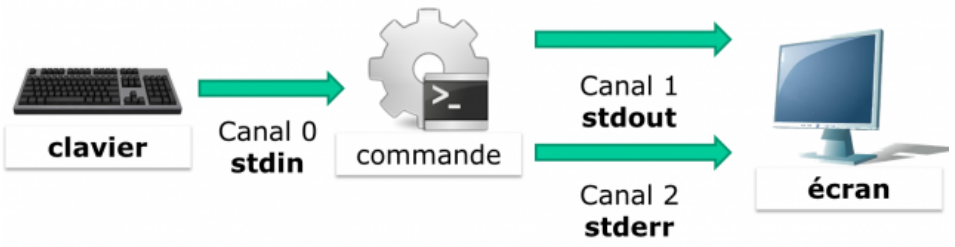
2.7.1. L’entrée et les sorties standards

Sur les systèmes UNIX et Linux, les flux standards sont aux nombres de trois. Ils permettent aux programmes, via la bibliothèque stdio.h de faire entrer ou sortir des informations.

Ces flux sont appelés canal X ou descripteur X de fichier.

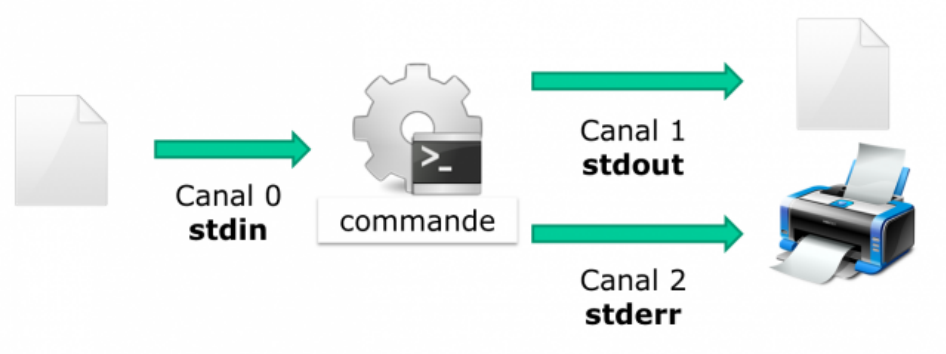
Par défaut :

- le clavier est le périphérique d’entrée pour le canal 0, appelé stdin ;
- l'écran est le périphérique de sortie pour les canaux 1 et 2, appelés stdout et stderr.



stderr reçoit les flux d’erreurs renvoyés par une commande. Les autres flux sont dirigés vers stdout.

Ces flux pointent vers des fichiers périphériques, mais comme tout est fichier sous UNIX, les flux d'entrées/sorties peuvent facilement être détournés vers d'autres fichiers. Ce principe fait toute la force du shell.



La redirection d'entrée

Il est possible de rediriger le flux d'entrée depuis un autre fichier avec le caractère inférieur "<" ou "<<". La commande lira le fichier au lieu du clavier :

```
[root]# ftp -in serverftp << cdes-ftp.txt
```

Seules les commandes demandant une saisie au clavier pourront gérer la redirection d'entrée.

La redirection d'entrée peut également être utilisée pour simuler une interactivité avec l'utilisateur. La commande lira le flux d'entrée jusqu'à rencontrer le mot clef défini après la redirection d'entrée.

Cette fonctionnalité est utilisée pour scripter des commandes interactives :

```
[root]# ftp -in serverftp << FIN
user alice password
put fichier
bye
FIN
```

Le mot clef FIN peut être remplacé par n'importe quel mot.

```
[root]# ftp -in serverftp << STOP
user alice password
put fichier
bye
STOP
```

Le shell quitte la commande ftp lorsqu'il reçoit une ligne ne contenant que le mot clef.

La redirection de l'entrée standard est peu utilisée car la plupart des commandes acceptent un nom de fichier en argument.

La commande wc pourrait s'utiliser ainsi :

```
[root]# wc -l .bash_profile
27 .bash_profile # le nombre de lignes est suivi du nom du fichier
[root]# wc -l < .bash_profile
27 # le nombre de lignes est seul
```

Les redirections de sortie

Les sorties standards peuvent être redirigées vers d'autres fichiers grâce aux caractères ">" ou ">>".

La redirection simple ">" écrase le contenu du fichier de sortie :

```
[root]# date +%F > fic_date
```

alors que la redirection double ">>" ajoute (concatène) au contenu du fichier de sortie.

```
[root]# date +%F >> fic_date
```

Dans les deux cas, le fichier est automatiquement créé lorsqu'il n'existe pas.

La sortie d'erreur standard peut être également redirigée vers un autre fichier. Cette fois-ci, il faudra préciser le numéro du canal (qui peut être omis pour les canaux 0 et 1) :

```
[root]# ls -R / 2> fic_erreurs
[root]# ls -R / 2>> fic_erreurs
```

Exemples

Redirection de 2 sorties vers 2 fichiers :

```
[root]# ls -R / >> fic_ok 2>> fic_nok
```

Redirection des 2 sorties vers un fichier unique :

```
[root]# ls -R / >> fic_log 2>&1
```

Redirection de **stderr** vers un "puit sans fond" (/dev/null) :

```
[root]# ls -R / 2>> /dev/null
```

Redirection vers la ou les consoles actives :

```
[root]# ls -R / 2>> /dev/console
```

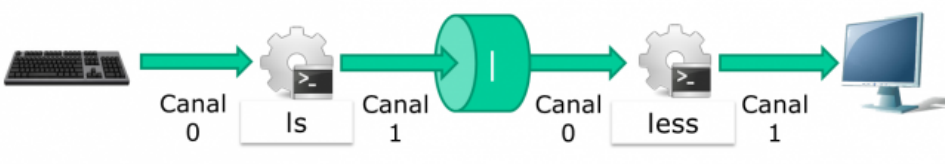
L'exemple précédent utilisant une redirection vers **/dev/console** est utilisé lors du TP sur les tâches planifiées (crontab).

Lorsque les 2 flux de sortie sont redirigés, aucune information n'est affichée à l'écran. Pour utiliser à la fois la redirection de sortie et conserver l'affichage, il faudra utiliser la commande tee.

2.7.2. Les tubes (pipe)

Un tube (pipe en anglais) est un mécanisme permettant de relier la sortie standard d'une première commande vers l'entrée standard d'une seconde.

Cette communication est monodirectionnelle et se fait grâce au symbole |. Le symbole pipe "|" est obtenu en appuyant simultanément sur les touches **AltGR+6**.



Toutes les données envoyées par la commande à gauche du tube à travers le canal de sortie standard sont envoyées au canal d'entrée standard de la commande placée à droite.

Les commandes particulièrement utilisées après un pipe sont des filtres.

Exemples

```
# N'afficher que le début :
[root]# ls -lia / | head

# N'afficher que la fin :
[root]# ls -lia / | tail

# Trier le résultat
[root]# ls -lia / | sort

# Compter le nombre de mots / caractères
[root]# ls -lia / | wc

# Chercher une chaîne de caractères dans le résultat :
[root]# ls -lia / | grep fichier
```

2.8. Points particuliers

2.8.1. La commande tee

La commande tee permet de rediriger la sortie standard d'une commande vers un fichier tout en maintenant l'affichage à l'écran.

Elle est combinée avec le pipe "|" pour recevoir en entrée la sortie de la commande à rediriger.

```
[root]# ls -lia / | tee fic
```

L'option -a permet d'ajouter au fichier au lieu de l'écraser.

2.8.2. Les commandes alias et unalias

Utiliser les alias est un moyen pour demander au shell de se souvenir d'une commande particulière avec ses options et lui donner un nom.

Par exemple :

```
[root]# ll
```

remplacera la commande :

```
root]# ls -l
```

La commande alias liste les alias de la session en cours. Des alias sont positionnés par défaut sur les distributions linux. Ici, les alias d'un serveur centos :

```
[root]# alias
alias
alias cp='cp -i'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mv='mv -i'
alias rm='rm -i'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot
 --show-tilde'
```

Les alias ne sont définis que de façon temporaire, le temps de la session utilisateur.

Pour une utilisation permanente, il faut les créer dans le fichier :

- .bashrc du répertoire de connexion de l'utilisateur ;
- /etc/profile.d/alias.sh pour tous les utilisateurs.

Une attention particulière doit être portée lors de l'usage d'alias qui peuvent potentiellement s'avérer dangereux !

> Par exemple, un alias mis en place à l'insu de l'administrateur :
>
> alias cd='rm -Rf'

La commande unalias permet de supprimer les alias.

```
[root]# unalias ll

# Pour supprimer tous les alias :
root]# unalias -a
```

Alias et fonctions utiles

grep

Colorise le résultat de la commande grep :

```
alias grep='grep --color=auto'
```

mcd

Il est fréquent de créer un dossier puis de se déplacer dedans :

```
mcd() { mkdir -p "$1"; cd "$1"; }
```

cls

Se déplacer dans un dossier et lister son contenu :

```
cls() { cd "$1"; ls; }
```

backup

Créer une copie de sauvegarde d'un fichier :

```
backup() { cp "$1"{,.bak}; }
```

extract

Extrait tout type d'archive :

```
extract () {
  if [ -f $1 ] ; then
    case $1 in
      *.tar.bz2) tar xjf $1 ;;
      *.tar.gz) tar xzf $1 ;;
      *.bz2) bunzip2 $1 ;;
      *.rar) unrar e $1 ;;
      *.gz) gunzip $1 ;;
      *.tar) tar xf $1 ;;
      *.tbz2) tar xjf $1 ;;
      *.tgz) tar xzf $1 ;;
      *.zip) unzip $1 ;;
      *.Z) uncompress $1 ;;
      *.7z) 7z x $1 ;;
      *)
        echo "'$1' cannot be extracted via extract()" ;;
    esac
  else
    echo "'$1' is not a valid file"
  fi
}
```

cmount

```
alias cmount="mount | column -t"

[root]# cmount
/dev/simfs  on  /                                          type  simfs
      (rw,relatime,usrquota,grpquota)
proc        on  /proc                                      type  proc
      (rw,relatime)
sysfs       on  /sys                                       type  sysfs
      (rw,relatime)
none        on  /dev                                       type
 devtmpfs     (rw,relatime,mode=755)
none        on  /dev/pts                                   type  devpts
      (rw,relatime,mode=600,ptmxmode=000)
none        on  /dev/shm                                   type  tmpfs
      (rw,relatime)
```

```
none        on  /proc/sys/fs/binfmt_misc                    type
 binfmt_misc  (rw,relatime)
```

2.8.3. Le caractère ;

Le caractère '**;**' chaîne les commandes.

Les commandes s'exécuteront toutes séquentiellement dans l'ordre de saisie une fois que l'utilisateur aura appuyé sur [ENTREE].

```
[root]# ls /; cd /home; ls -lia; cd /
```

3

La gestion des utilisateurs

3.1. Généralités

Chaque utilisateur est membre d'au moins un groupe : **c'est son groupe principal**.

Plusieurs utilisateurs peuvent faire partie d'un même groupe.

Les utilisateurs peuvent appartenir à d'autres groupes. Ces utilisateurs sont invités dans ces **groupes secondaires**.

Chaque utilisateur possède un groupe principal et peut être invité dans un ou plusieurs groupes secondaires.

Les groupes et utilisateurs se gèrent par leur identifiant numérique unique GID et UID.

Les fichiers de configuration se trouvent dans "/etc".

- **UID** : User IDentifier. Identifiant unique d'utilisateur.
- **GID** : Group IDentifier. Identifiant unique de groupe.

Il est recommandé d'utiliser les commandes d'administration au lieu de modifier manuellement les fichiers.

3.2. Gestion des groupes

Fichiers modifiés, ajout de lignes :

- /etc/group
- /etc/gshadow

3.2.1. Commande groupadd

La commande groupadd permet d'ajouter un groupe au système.

Syntaxe de la commande groupadd.

```
groupadd [-f] [-g GID] groupe
```

Exemple :

```
[root]# groupadd -g 512 GroupeB
```

Tableau 3.1. Options de la commande groupadd

Option	Description
-g GID	GID du groupe à créer.
-f	Le système choisit un GID si celui précisé par l'option -g existe déjà.
-r	Crée un groupe système avec un GID compris entre **SYS_GID_MIN** et **SYS_GID_MAX** définies dans **/etc/login.defs**.

Règles de nommage des groupes :

- Pas d'accents, ni caractères spéciaux ;
- Différents du nom d'un utilisateur ou fichier système existant.

3.2.2. Commande groupmod

La commande groupmod permet de modifier un groupe existant sur le système.

Syntaxe de la commande groupmod.

```
groupmod [-g GID] [-n nom] groupe
```

Exemple :

```
[root]# groupmod -g 516 GroupeP
[root]# groupmod -n GroupeC GroupeB
```

Tableau 3.2. Options de la commande groupmod

Option	Description
-g GID	Nouveau GID du groupe à modifier.
-n nom	Nouveau nom.

Il est possible de modifier le nom d'un groupe, son GID ou les deux simultanément.

Après modification, les fichiers appartenant au groupe ont un GID inconnu. Il faut leur réattribuer le nouveau GID.

```
[root]# find / -gid 502 -exec chgrp 516 {} \;
```

3.2.3. Commande groupdel

La commande groupdel permet de supprimer un groupe existant sur le système.

Syntaxe de la commande groupdel.

```
groupdel groupe
```

Exemple :

```
[root]# groupdel GroupeC
```

Pour être supprimé, un groupe ne doit plus contenir d'utilisateurs.

La suppression du dernier utilisateur d'un groupe éponyme entrainera la suppression de ce groupe par le système.

Chaque groupe possède un GID unique. Un groupe peut être dupliqué. Par convention, les GID des groupes systèmes vont de 0 (root) à 499.

Un utilisateur faisant obligatoirement partie d'un groupe, il est nécessaire de créer les groupes avant d'ajouter les utilisateurs. Par conséquent, un groupe peut ne pas avoir de membres.

3.2.4. Fichier /etc/group

Ce fichier contient les informations de groupes (séparées par ' : ').

```
[root]# tail -1 /etc/group
GroupeP:x:516:stagiaire
   1    2  3          4
```

- 1 : Nom du groupe.
- 2 : Mot de passe (x si défini dans /etc/gshadow).
- 3 : GID.
- 4 : Membres invités (séparés par des virgules, ne contient pas les membres principaux).

Chaque ligne du fichier /etc/group correspond à un groupe. Les utilisateurs dont ce groupe est leur groupe principal ne sont pas listés à ce niveau.

Cette information d'appartenance est en fait déjà fournie par le fichier /etc/passwd…

Fichier /etc/gshadow

Ce fichier contient les informations de sécurité sur les groupes (séparées par ' : ').

```
[root]# grep GroupeA /etc/gshadow
GroupeA:$6$2,9,v...SBn160:alain:stagiaire
   1           2            3      4
```

- 1 : Nom du groupe.
- 2 : Mot de passe chiffré.
- 3 : Administrateur du groupe.

- 4 : Membres invités (séparés par des virgules, ne contient pas les membres principaux).

Pour chaque ligne du fichier /etc/group doit correspondre une ligne du fichier /etc/gshadow.

Un **!** au niveau du mot de passe indique que celui-ci est bloqué. Ainsi aucun utilisateur ne peut utiliser le mot de passe pour accéder au groupe (sachant que les membres du groupe n'en ont pas besoin).

3.3. Gestion des utilisateurs

3.3.1. Définition

Un utilisateur se définit comme suit dans le fichier **/etc/passwd** :

1. Login ;
2. Mot de passe ;
3. UID ;
4. GID du groupe principal ;
5. Commentaire ;
6. Répertoire de connexion ;
7. Interpréteur de commandes (/bin/bash, /bin/nologin,…).

Il existe trois types d'utilisateurs :

- **root** : Administrateur du système ;
- **utilisateur système** : Utilisé par le système pour la gestion des droits d'accès des applications ;
- **utilisateur ordinaire** : Autre compte permettant de se connecter au système.

Fichiers modifiés, ajout de lignes :

- /etc/passwd
- /etc/shadow

3.3.2. Commande useradd

La commande useradd permet d'ajouter un utilisateur.

Syntaxe de la commande useradd.

```
useradd [-u UID] [-g GID] [-d répertoire] [-s shell] login
```

Exemple :

```
[root]# useradd -u 1000 -g 513 -d /home/GroupeC/carine carine
```

Tableau 3.3. Options de la commande useradd

Option	Description
-u UID	UID de l'utilisateur à créer.
-g GID	GID du groupe principal.
-d répertoire	Répertoire de connexion.
-s shell	Interpréteur de commandes.
-c	Ajoute un commentaire.
-U	Ajoute l'utilisateur à un groupe portant le même nom créé simultanément.
-M	Ne crée pas le répertoire de connexion.

À la création, le compte ne possède pas de mot de passe et est verrouillé. Il faut assigner un mot de passe pour déverrouiller le compte.

Règles de nommage des comptes :

- Pas d'accents, de majuscules ni caractères spéciaux ;
- Différents du nom d'un groupe ou fichier système existant ;
- Définir les options -u, -g, -d et -s à la création.

L'arborescence du répertoire de connexion doit être créée à l'exception du dernier répertoire. Le dernier répertoire est

> créé par la commande useradd qui en profite pour y copier les fichiers du "skel".

Un utilisateur peut faire partie de plusieurs groupes en plus de son groupe principal.

Pour les groupes secondaires, il faut utiliser l'option -G.

Exemple :

```
[root]# useradd -u 500 -g GroupeA -G GroupeP,GroupeC albert
```

Valeur par défaut de création d'utilisateur.

Modification du fichier **/etc/default/useradd**.

```
useradd -D [-b répertoire] [-g groupe] [-s shell]
```

Exemple :

```
[root]# useradd -D -g 500 -b /home -s /bin/bash
```

Tableau 3.4. Options de la commande useradd pour modifier les valeurs par défaut

Option	Description
-D	Définit les valeurs par défaut de création d'utilisateur.
-b répertoire	Définit le répertoire de connexion par défaut.
-g groupe	Définit le groupe par défaut.
-s shell	Définit le shell par défaut.
-f	Nombre de jours suivant l'expiration du mot de passe avant que le compte ne soit désactivé.
-e	Date à laquelle le compte sera désactivé.

3.3.3. Commande usermod

La commande usermod permet de modifier un utilisateur.

Syntaxe de la commande usermod.

```
usermod [-u UID] [-g GID] [-d répertoire] [-m] login
```

Exemple :

```
[root]# usermod -u 544 carine
```

Options identiques à la commande useradd.

Tableau 3.5. Options de la commande usermod

Option	Description
-m	Associé à l'option -d, déplace le contenu de l'ancien répertoire de connexion vers le nouveau.
-l login	Nouveau nom.
-e AAAA-MM-JJ	Date d'expiration du compte.
-L	Verrouille le compte.
-U	Déverrouille le compte.
-a	Empêche la suppression de l'utilisateur d'un groupe secondaire lors de l'ajout dans un autre groupe secondaire.
-G	Précise plusieurs groupes secondaires lors de l'ajout.

Avec la commande **usermod**, le verrouillage d'un compte se traduit par l'ajout de ! devant le mot de passe dans le fichier **/etc/shadow**.

Pour être modifié un utilisateur doit être déconnecté et ne pas avoir de processus en cours.

Après modification de l'identifiant, les fichiers appartenant à l'utilisateur ont un UID inconnu. Il faut leur réattribuer le nouvel UID.

```
[root]# find / -uid 1000 -exec chown 544: {} \;
```

Il est possible d'inviter un utilisateur dans un ou plusieurs groupes secondaires avec les options -a et -G.

Exemple :

```
[root]# usermod -aG GroupeP,GroupeC albert
```

La commande **usermod** agit en modification et non en ajout.

Pour un utilisateur invité dans un groupe par l'intermédiaire de cette commande et déjà positionné comme invité dans d'autres groupes secondaires, il faudra indiquer dans la commande de gestion de groupe tous les groupes dont il fait partie sinon il disparaîtra de ceux-ci.

L'option -a empêche ce problème.

Exemples :

- Invite albert dans le groupe GroupeP

```
[root]# usermod -G GroupeP albert
```

- Invite albert dans le groupe GroupeG, mais le supprime de la liste des invités de GroupeP.

```
[root]# usermod -G GroupeG albert
```

- Donc soit :

```
[root]# usermod -G GroupeP,GroupeG albert
```

- Soit :

```
[root]# usermod -aG GroupeG albert
```

3.3.4. Commande userdel

La commande userdel permet de supprimer le compte d'un utilisateur.

Syntaxe de la commande userdel.

```
[root]# userdel -r carine
```

Tableau 3.6. Options de la commande userdel

Option	Description
-r	Supprime le répertoire de connexion et les fichiers contenus.

Pour être supprimé, un utilisateur doit être déconnecté et ne pas avoir de processus en cours.

userdel supprime la ligne de l'utilisateur dans les fichiers /etc/passwd et /etc/gshadow

Chaque utilisateur possède un UID unique. Par convention, les UID des utilisateurs 'système' vont de 0 (root) à 499.

Un utilisateur est obligatoirement membre d'un groupe. Il est donc nécessaire de créer les groupes avant d'ajouter les utilisateurs.

3.3.5. Fichier /etc/passwd

Ce fichier contient les informations des utilisateurs (séparées par ' : ').

```
[root]# head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
 1   2 3 4   5    6       7
```

- 1 : Login.
- 2 : Mot de passe (x si défini dans /etc/shadow).
- 3 : UID.
- 4 : GID du groupe principal.
- 5 : Commentaire.
- 6 : Répertoire de connexion.
- 7 : Interpréteur de commandes.

3.3.6. Fichier /etc/shadow

Ce fichier contient les informations de sécurité des utilisateurs (séparées par ' : ').

```
[root]# tail -1 /etc/shadow
root:$6$...:15399:0:99999:7:::
   1    2     3   4   5   6,7,8,9
```

- 1 : Login.
- 2 : Mot de passe chiffré.
- 3 : Date du dernier changement.
- 4 : Durée de vie minimale du mot de passe.
- 5 : Durée de vie maximale du mot de passe.
- 6 : Nombre de jours avant avertissement.
- 7 : Délai avant désactivation du compte après expiration.
- 8 : Délai d'expiration du compte.
- 9 : Réservé pour une utilisation future.

Pour chaque ligne du fichier /etc/passwd doit correspondre une ligne du fichier /etc/shadow.

3.4. Propriétaires des fichiers

Tous les fichiers appartiennent forcément à un utilisateur et à un groupe.

Le groupe principal de l'utilisateur qui crée le fichier est, par défaut, le groupe propriétaire du fichier.

3.4.1. Commandes de modifications :

Commande chown

La commande chown permet de modifier les propriétaires d'un fichier.

Syntaxe de la commande chown.

```
chown [-R] [-v] login[:groupe] fichier
```

Exemples :

```
[root]# chown root fichier
[root]# chown albert:GroupeA fichier
```

Tableau 3.7. Options de la commande chown

Option	Description
-R	Modifie les propriétaires du répertoire et de son contenu.
-v	Affiche les modifications exécutées.

Pour ne modifier que l'utilisateur propriétaire :

```
[root]# chown albert fichier
```

Pour ne modifier que le groupe propriétaire :

```
[root]# chown :GroupeA fichier
```

Modification de l'utilisateur et du groupe propriétaire :

```
[root]# chown albert:GroupeA fichier
```

Dans l'exemple suivant le groupe attribué sera le groupe principal de l'utilisateur précisé.

```
[root]# chown albert: fichier
```

3.4.2. Commande chgrp

La commande chgrp permet de modifier le groupe propriétaire d'un fichier.

Syntaxe de la commande chgrp.

```
chgrp [-R] [-v] groupe fichier
```

Exemple :

```
[root]# chgrp groupe1 fichier
```

Tableau 3.8. Options de la commande chgrp

Option	Description
-R	Modifie les groupes propriétaires du répertoire et de son contenu (récursivité).
-v	Affiche les modifications exécutées.

Il est possible d'appliquer à un fichier un propriétaire et un groupe propriétaire en prenant comme référence ceux d'un autre fichier :

```
chown [options] --reference=RRFILE FILE
```

Par exemple :

```
chown --reference=/etc/groups /etc/passwd
```

3.5. Gestion des invités

3.5.1. Commande gpasswd

La commande gpasswd permet de gérer un groupe.

Syntaxe de la commande gpasswd.

```
gpasswd [-a login] [-A login] [-d login] [-M login] groupe
```

Exemples :

```
[root]# gpasswd -A alain GroupeA
[alain]$ gpasswd -a patrick GroupeA
```

Tableau 3.9. Options de la commande gpasswd

Option	Description
-a login	Ajoute l'utilisateur au groupe.
-A login	Définit l'administrateur du groupe.
-d login	Retire l'utilisateur du groupe.
-M login	Définit la liste exhaustive des invités.

La commande gpasswd -M agit en modification et non en ajout.

```
# gpasswd GroupeA
New Password :
Re-enter new password :
```

3.5.2. Commande id

La commande id affiche les noms des groupes d'un utilisateur.

Syntaxe de la commande id.

```
id login
```

Exemple :

```
[root]# id alain
uid=500(alain) gid=500(GroupeA) groupes=500(GroupeA),516(GroupeP)
```

3.5.3. Commande newgrp

La commande newgrp permet d'utiliser temporairement un groupe secondaire pour la création de fichiers.

Syntaxe de la commande newgrp.

```
newgrp [groupesecondaire]
```

Exemple :

```
[alain]$ newgrp GroupeB
```

Après utilisation de cette commande, les fichiers seront créés avec le GID de son groupe secondaire.

La commande newgrp sans paramètre réaffecte le groupe principal.

3.6. Sécurisation

3.6.1. Commande passwd

La commande passwd permet de gérer un mot de passe.

Syntaxe de la commande passwd.

```
passwd [-d] [-l] [-S] [-u] [login]
```

Exemples :

```
[root]# passwd -l albert
[root]# passwd -n 60 -x 90 -w 80 -i 10 patrick
```

Tableau 3.10. Options de la commande passwd

Option	Description
-d	Supprime le mot de passe.
-l	Verrouille le compte.
-S	Affiche le statut du compte.
-u	Déverrouille le compte.
-e	Fait expirer le mot de passe.
-n jours	Durée de vie minimale du mot de passe.
-x jours	Durée de vie maximale du mot de passe.
-w jours	Délai d'avertissement avant expiration.
-i jours	Délai avant désactivation lorsque le mot de passe expire.

Avec la commande passwd, le verrouillage d'un compte se traduit par l'ajout de !! devant le mot de passe dans le fichier /etc/shadow.

L'utilisation de la commande usermod -U ne supprime qu'un seul des !. Le compte reste donc verrouillé.

Cette commande est accessible aux utilisateurs pour modifier leur mot de passe (l'ancien mot de passe est demandé).

L'administrateur peut modifier les mots de passe de tous les utilisateurs sans restriction.

Exemple :

- Alain change son mot de passe :

```
[alain]$ passwd
```

- root change le mot de passe d'alain :

```
[root]# passwd alain
```

La commande passwd est accessible aux utilisateurs pour modifier leur mot de passe (l'ancien mot de passe est demandé). L'administrateur peut modifier les mots de passe de tous les utilisateurs sans restriction.

Ils devront se soumettre aux restrictions de sécurité.

Lors d'une gestion des comptes utilisateurs par script shell, il peut être utile de définir un mot de passe par défaut après avoir créé l'utilisateur.

Ceci peut se faire en passant le mot de passe à la commande passwd.

Exemple :

```
[root]# echo "azerty,1" | passwd --stdin philippe
```

Le mot de passe est saisi en clair, passwd se charge de le chiffrer.

3.7. Commande chage

La commande chage permet de gérer la stratégie de compte.

Syntaxe de la commande chage.

```
chage [-d date] [-E date] [-I jours] [-l] [-m jours] [-M jours] [-W
 jours] [login]
```

Exemple :

```
[root]# chage -m 60 -M 90 -W 80 -I 10 alain
```

Tableau 3.11. Options de la commande chage

Option	Description
-I jours	Délai avant désactivation, mot de passe expiré (i majuscule).
-l	Affiche le détail de la stratégie (l minuscule).
-m jours	Durée de vie minimale du mot de passe.
-M jours	Durée de vie maximale du mot de passe.
-d AAA-MM-JJ	Dernière modification du mot de passe.
-E AAA-MM-JJ	Date d'expiration du compte.
-W jours	Délai d'avertissement avant expiration.

La commande chage propose également un mode interactif.

L'option -d force la modification du mot de passe à la connexion.

Exemples :

```
[root]# chage philippe
[root]# chage -d 0 philippe
```

En l'absence d'utilisateur précisé, la commande concernera l'utilisateur qui la saisit.

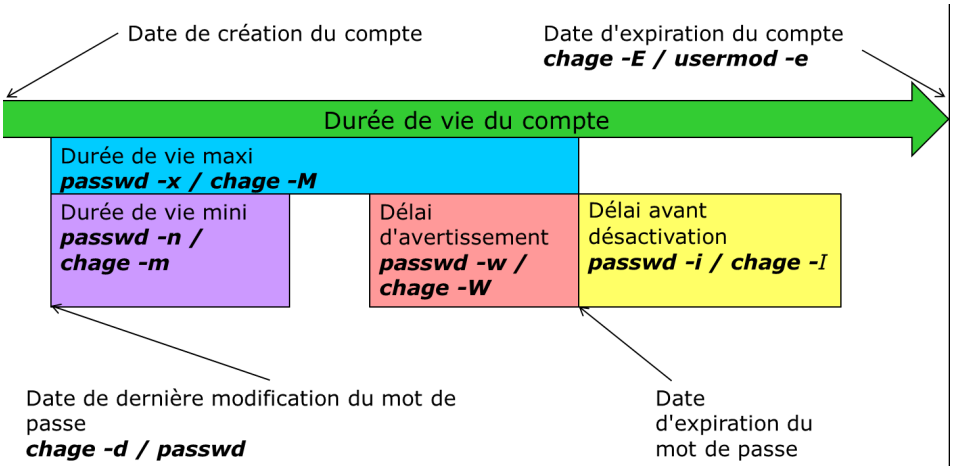


Figure 3.1. Gestion des comptes utilisateurs avec chage

3.8. Gestion avancée

Fichiers de configuration :

- /etc/default/useradd
- /etc/login.defs
- /etc/skel

L'édition du fichier /etc/default/useradd se fait grâce à la commande useradd.

Les autres fichiers sont à modifier avec un éditeur de texte.

3.8.1. Fichier /etc/default/useradd

Ce fichier contient le paramétrage des données par défaut.

Lors de la création d'un utilisateur, si les options ne sont pas précisées, le système utilise les valeurs par défaut définies dans /etc/default/useradd.

Ce fichier est modifié par la commande **useradd -D** (**useradd -D** saisie sans autre option affiche le contenu du fichier **/etc/default/useradd**).

Tableau 3.12. Contenu du fichier /etc/default/useradd

Valeur	Commentaire
GROUP	Groupe par défaut.
HOME	Chemin dans lequel le répertoire de connexion du nom de l'utilisateur sera créé.
INACTIVE	Nombre de jours suivant l'expiration du mot de passe avant que le compte ne soit désactivé.
EXPIRE	Date d'expiration du compte.
SHELL	Interpréteur de commandes.
SKEL	Répertoire squelette du répertoire de connexion.
CREATE_MAIL_SPOOL	Création de la boîte aux lettres dans /var/spool/mail.

Sans l'option **-g**, la commande **useradd** crée un groupe du nom de l'utilisateur et l'y place.

Pour que la commande **useradd** récupère la valeur du champ **GROUP** du fichier **/etc/default/useradd**, il faut préciser l'option **-N**.

Exemple :

```
[root]# useradd -u 501 -N GroupeA
```

3.8.2. Fichier /etc/login.defs

Ce fichier contient de nombreux paramètres par défaut utiles aux commandes de création ou de modification d'utilisateurs. Ces informations sont regroupées par paragraphe en fonction de leur utilisation :

- Boites aux lettres ;
- Mots de passe ;
- UID et GID ;
- Umask ;
- Connexions ;

- Terminaux.

3.8.3. Fichier /etc/skel

Lors de la création d'un utilisateur, son répertoire personnel et ses fichiers d'environnement sont créés.

Ces fichiers sont copiés automatiquement à partir du répertoire /etc/skel.

- .bash_logout
- .bash_profile
- .bashrc

Tous les fichiers et répertoires placés dans ce répertoire seront copiés dans l'arborescence des utilistateurs lors de leur création.

3.9. Changement d'identité

3.9.1. Commande su

La commande su permet de modifier l'identité de l'utilisateur connecté.

Syntaxe de la commade su.

```
su [-] [-c commande] [login]
```

Exemples :

```
[root]# su - alain
[albert]$ su -c "passwd alain"
```

Tableau 3.13. Options de la commande su

Option	Description
-	Charge l'environnement complet de l'utilisateur.
-c commande	Exécute la commande sous l'identité de l'utilisateur.

Si le login n'est pas spécifié, ce sera root.

Les utilisateurs standards devront taper le mot de passe de la nouvelle identité.

Il y a création de couches successives. Pour passer d'un utilisateur à un autre, il faut d'abord taper la commande exit pour reprendre son identité puis la commande su pour prendre une autre identité.

Chargement du profil

root endosse alain avec su :

```
...
/home/GroupeA/alain/bash_rc
/etc/bashrc
...
```

root endosse alain avec su - :

```
...
/home/GroupeA/alain/bash_profile
/home/GroupeA/alain/bash_rc
/etc/bashrc
...
```

Un utilisateur peut endosser temporairement (pour une autre commande ou une session entière) l'identité d'un autre compte.

Si aucun utilisateur n'est précisé, la commande concernera root su -.

Il est nécessaire de connaitre le mot de passe de l'utilisateur dont l'identité est endossé sauf si c'est root qui exécute la commande.

Un administrateur peut ainsi travailler sur un compte utilisateur standard et n'utiliser les droits du compte root que ponctuellement.

4

Système de fichiers

4.1. Partitionnement

Le partitionnement va permettre l'installation de plusieurs systèmes d'exploitation car il est impossible d'en faire cohabiter plusieurs sur un même lecteur logique. Le partitionnement permet également de cloisonner des données (sécurité, optimisation d'accès, …).

Le découpage du disque physique en volumes partitionnés est inscrit dans la table des partitions stockée dans le premier secteur du disque (MBR : Master Boot Record).

Un même disque physique peut être découpé en 4 partitions maximum :

- **Primaire** (ou principale)
- **Étendue**

Il ne peut y avoir qu'une seule partition étendue par disque physique. Afin de bénéficier de lecteur supplémentaire, la partition étendue peut être découpée en partitions logiques

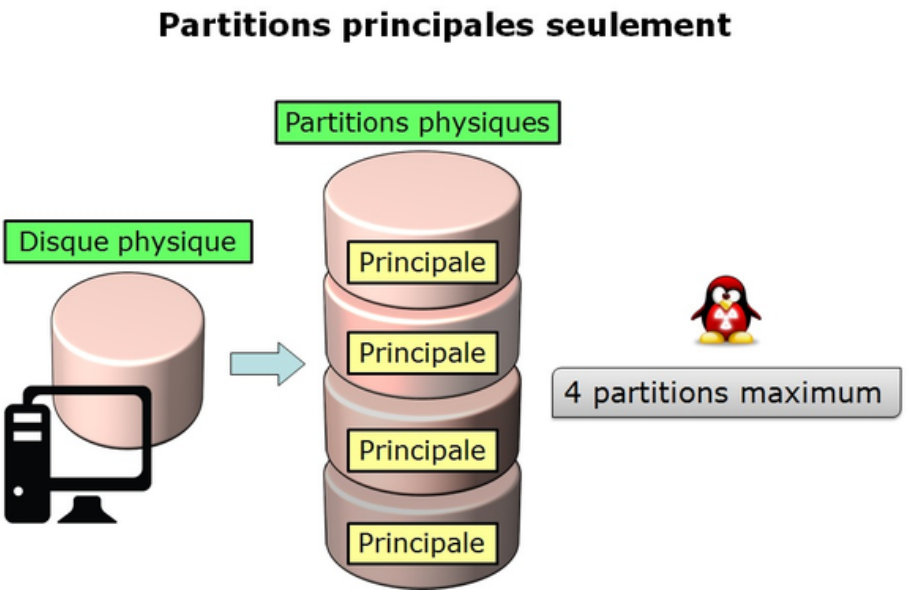


Figure 4.1. Découpage en quatre partitions principales seulement

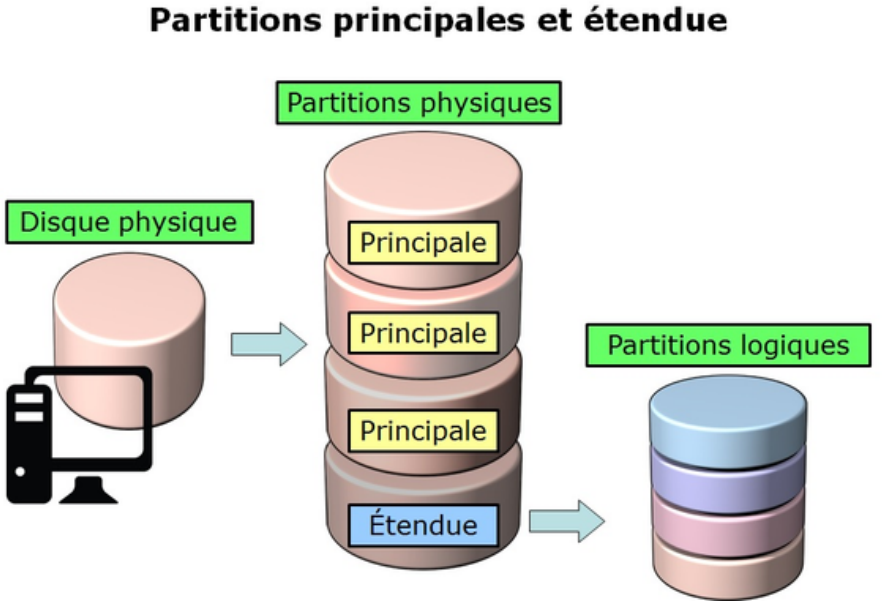


Figure 4.2. Découpage en trois partitions principales et une partition étendue

Les devices, ou périphériques, sont les fichiers identifiant les différents matériels détectés par la carte mère. Ces fichiers sont stockés sans **/dev**. Le service qui détecte les nouveaux périphériques et leur donne des noms s'appelle "udev".

Ils sont identifiés en fonction de leur type.

Les périphériques de stockage se nomment **hd** pour les disques durs IDE et **sd** pour les autres supports. Vient ensuite une lettre qui commence par **a** pour le premier périphérique, puis **b**, **c**, …

Enfin nous allons trouver un chiffre qui définit le volume partitionné : **1** pour la première partition primaire, …

Attention, la partition étendue, qui ne supporte pas de système de fichier, porte quand même un numéro.

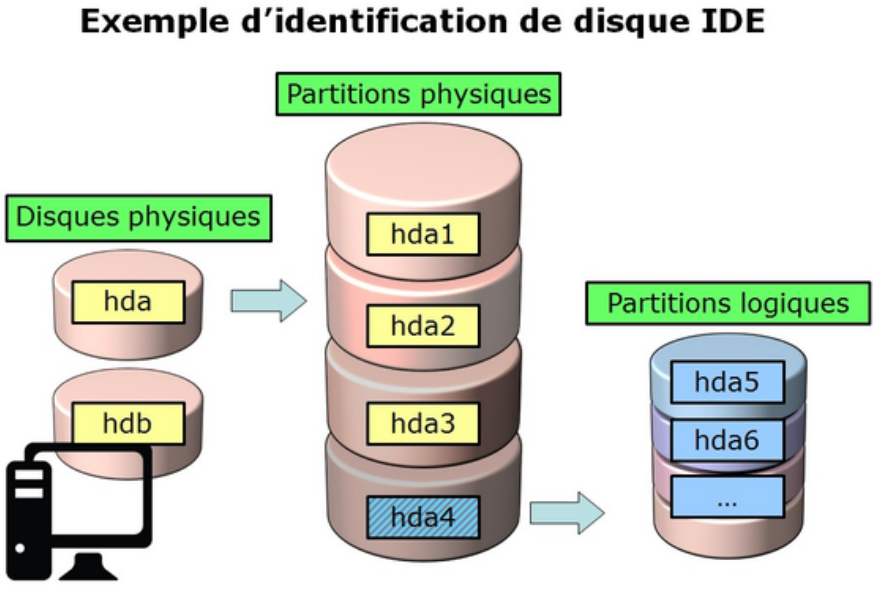


Figure 4.3. Identification des partitions

Il existe deux commandes permettant le partitionnement d'un disque : **fdisk** et **cfdisk**. Ces deux commandes possèdent un menu interactif. **cfdisk** étant plus fiable et mieux optimisée, il est préférable de l'utiliser.

La seule raison d'utiliser **fdisk** est lorsqu'on souhaite lister tous les périphériques logiques avec l'option **-l**.

```
[root]# fdisk -l
[root]# fdisk -l /dev/sdc
[root]# fdisk -l /dev/sdc2
```

4.1.1. Commande cfdisk

La commande cfdisk permet de gérer les partitions

Syntaxe de la commande cfdisk.

```
cfdisk device
```

Exemple :

```
[root]# cfdisk /dev/sda
             cfdisk (util-linux-ng 2.17.2)
               Unité disque : /dev/sda
          Taille: 10737418240 octets, 10.7 Go
   Têtes: 255 Secteurs par piste: 63 Cylindres: 1305
    Nom  Fanions  Part Type Sys.Fic  Étiq. Taille
    ------------------------------------------------
                      ...
[aide] [nouvelle] [afficher] [quitter] [unités] [ecrire]
```

La préparation, sans LVM, du support physique passe par cinq étapes :

- Mise en place du disque physique ;
- Partitionnement des volumes (découpage géographique du disque, possibilité d'installer plusieurs systèmes, …) ;
- Création des systèmes de fichiers (permet au système d'exploitation de gérer les fichiers, l'arborescence, les droits, …) ;
- Montage des systèmes de fichiers (inscription du système de fichiers dans l'arborescence) ;
- Gérer l'accès aux utilisateurs.

4.2. LVM

Logical **V**olume **M**anager

La gestion de volume crée une couche abstraite sur un stockage physique offrant des avantages par rapport à l'utilisation directe du stockage physique :

- Capacité du disque plus flexible ;
- Déplacement des données en ligne ;
- Disques en mode "stripe" (découpage) ;
- Volumes miroirs (recopie) ;
- Instantanés de volumes (snapshot).

L'inconvénient est que si un des volumes physiques devient HS, alors c'est l'ensemble des volumes logiques qui utilisent ce volume physique qui sont perdus. Il faudra utiliser LVM sur des disques raid.

Le LVM est disponible sous linux à partir de la version 2.4 du noyau.

LVM est uniquement géré par le système d'exploitation. Par conséquent le BIOS a besoin d'au moins une partition sans LVM pour démarrer.

4.2.1. Les groupes de volume

Les volumes physiques **PV** (issus des partitions) sont combinés en des groupes de volumes **VG**. Chaque **VG** représente un espace disque pouvant être découpé en volumes logiques **LV**. **L'extension** est la plus petite unité d'espace de taille fixe pouvant être allouée.

- **PE** : Physical Extension
- **LE** : Logical Extension

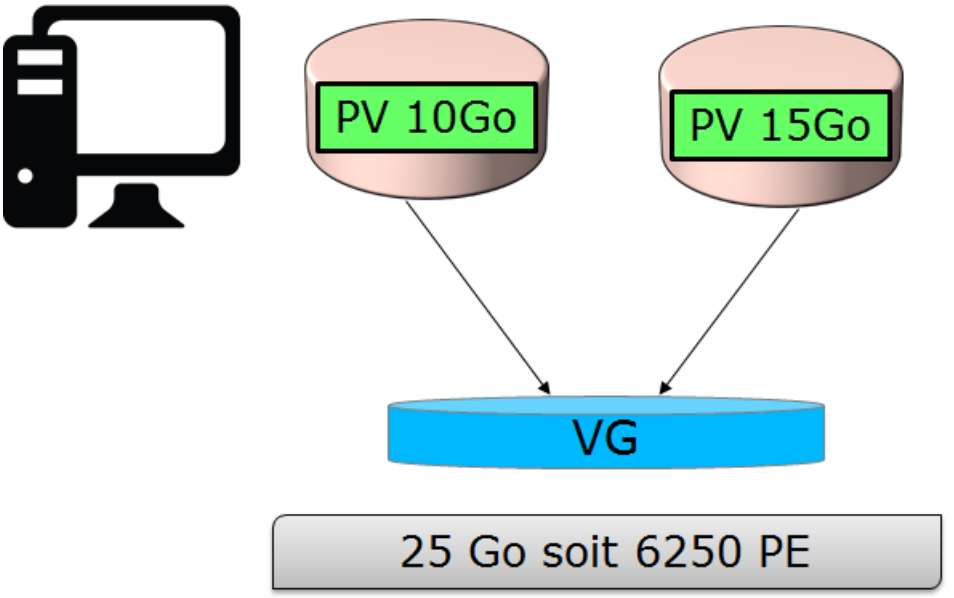


Figure 4.4. Groupe de volumes, taille de PE égale à 4Mo

4.2.2. Les volumes logiques

Un groupe de volume **VG** est divisé en volumes logiques **LV** offrant différents modes de fonctionnement :

- Volumes linéaires ;
- Volumes en mode stripe ;
- Volumes en miroirs.

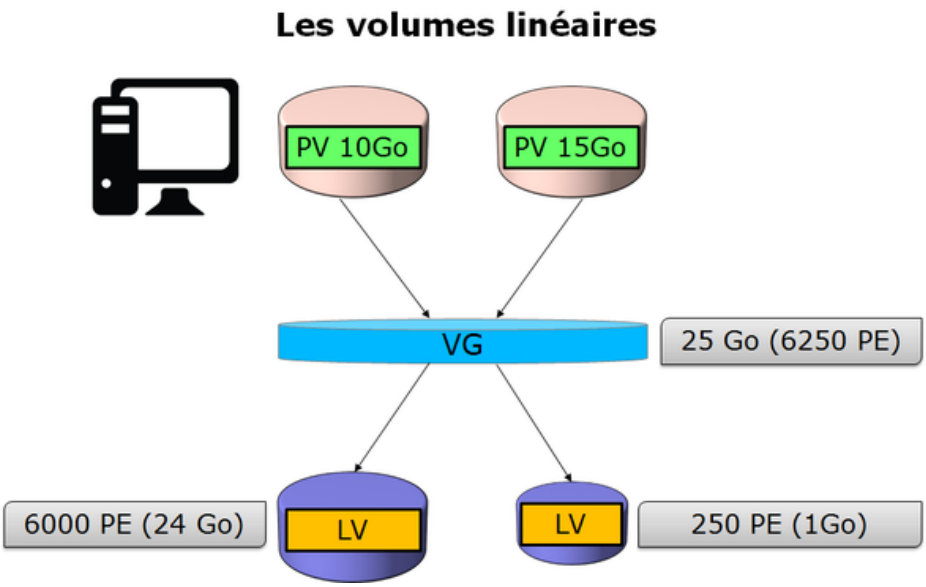


Figure 4.5. Volumes linéaires

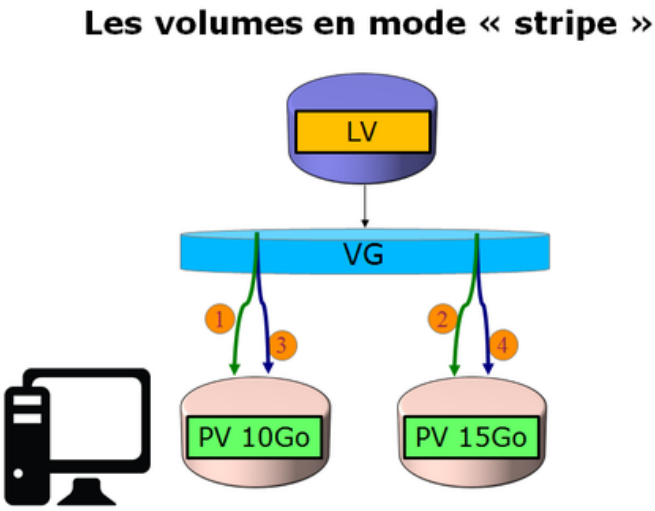


Figure 4.6. Volumes en mode stripe

Le “striping” améliore les performances en écrivant des données sur un nombre prédéterminé de volumes physiques avec une technique de round-robin.

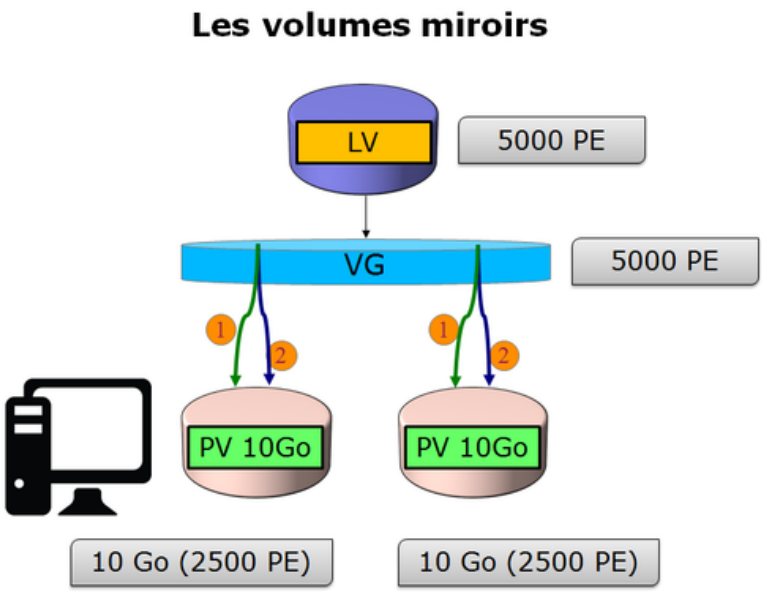


Figure 4.7. Volumes en mirroirs

4.2.3. Commandes LVM pour la gestion des volumes

Commande pvcreate

La commande **pvcreate** permet de créer des volumes physiques. Elle transforme des partition Linux en volumes physiques.

Syntaxe de la commande pvcreate.

```
pvcreate [-options] partition
```

Exemple :

```
[root]# pvcreate /dev/hdb1
pvcreate -- physical volume « /dev/hdb1 » successfuly created
```

Tableau 4.1. Option de la commande pvcreate

Option	Description
-f	Force la création du volume (disque déjà transformé en volume physique).

Commande vgcreate

La commande **vgcreate** permet de créer des groupes de volumes. Elle regroupe un ou plusieurs volumes physiques dans un groupe de volumes.

Syntaxe de la commande vgcreate.

```
vgcreate volume physical_volume [PV...]
```

Exemple :

```
[root]# vgcreate volume1 /dev/hdb1
…
vgcreate – volume group « volume1 » successfuly created and activated
```

Commande lvcreate

La commande **lvcreate** permet de créer des volumes logiques. Le système de fichiers est ensuite créé sur ces volumes logiques.

Syntaxe de la commande lvcreate.

```
lvcreate -L taille [-n nom] nom_VG
```

Exemple :

```
[root]# lvcreate –L 600M –n VolLog1 volume1
lvcreate -- logical volume « /dev/volume1/VolLog1 » successfuly created
```

Tableau 4.2. Options de la commande lvcreate

Option	Description
-L taile	Taille du volume logique en K, M ou G
-n nom	Nom du LV. Fichier spécial créé dans **/dev/nom_volume** portant ce nom

4.2.4. Commandes LVM pour visualiser les informations concernant les volumes

Commande pvdisplay

La commande **pvdisplay** permet de visualiser les informations concernant les volumes physiques.

Syntaxe de la commande pvdisplay.

```
pvdisplay /dev/nom_PV
```

Exemple :

```
[root]# pvdisplay /dev/nom_PV
```

Commande vgdisplay

La commande **vgdisplay** permet de visualiser les informations concernant les groupes de volumes.

Syntaxe de la commande vgdisplay.

```
vgdisplay nom_VG
```

Exemple :

```
[root]# vgdisplay volume1
```

Commande lvdisplay

La commande **lvdisplay** permet de visualiser les informations concernant les volumes logiques.

Syntaxe de la commande lvdisplay.

```
lvdisplay /dev/nom_VG/nom_LV
```

Exemple :

```
[root]# lvdisplay /dev/volume1/VolLog1
```

4.2.5. Préparation du support physique

La préparation avec LVM du support physique se décompose comme suit :

- Mise en place du disque physique
- Partitionnement des volumes
- **Volume physique LVM**
- **Groupes de volumes LVM**
- **Volumes logiques LVM**
- Création des systèmes de fichiers
- Montage des systèmes de fichiers
- Gérer l'accès aux utilisateurs

4.3. Structure d'un système de fichiers

Un système de fichiers **SF** peut se nommer système de gestion de fichiers **SGF** mais également file system **FS**.

Un système de fichiers est en charge des actions suivantes :

- Sécuriser les droits d'accès et de modification des fichiers ;
- Manipuler des fichiers : créer, lire, modifier et supprimer ;
- Localiser les fichiers sur le disque ;
- Gérer l'espace mémoire.

Le système d'exploitation Linux est capable d'exploiter différents systèmes de fichiers (ext2, ext3, ext4, FAT16, FAT32, NTFS, HFS, BtrFS, JFS, XFS, …).

4.3.1. Commande mkfs

La commande mkfs permet de créer un système de fichiers Linux.

Syntaxe de la commande mkfs.

```
mkfs [-t fstype] filesys
```

Exemple :

```
[root]# mkfs -t ext4 /dev/sda1
```

Tableau 4.3. Option de la commande mkfs

Option	Description
-t	Indique le type de système de fichiers à utiliser

Sans système de fichiers il n'est pas possible d'utiliser l'espace disque.

Chaque système de fichiers possède une structure qui est identique sur chaque partition. Un Bloc de Boot et un Super Bloc initialisés par le système puis une Table des Inodes et une Zone de Données initialisées par l'administrateur.

La seule exception est concernant la partition **swap**.

4.3.2. Bloc de boot

Il occupe le premier bloc sur le disque et est présent sur toutes les partitions. Il contient le programme assurant le démarrage et l'initialisation du système et n'est donc renseigné que pour la partition de démarrage.

4.3.3. Super bloc

La taille de sa table est définie à la création. Il est présent sur chaque partition et contient les éléments nécessaires à l'exploitation de celle-ci.

Il décrit le Système de Fichiers :

- Nom du Volume Logique ;
- Nom du Système de Fichiers ;
- Type du Système de Fichiers ;

- État du Système de Fichiers ;
- Taille du Système de Fichiers ;
- Nombre de blocs libres ;
- Pointeur sur le début de la liste des blocs libres ;
- Taille de la liste des inodes ;
- Nombre et la liste des inodes libres.

Une copie est chargée en mémoire centrale dès l'initialisation du système. Cette copie est mise à jour dès modification et le système la sauvegarde périodiquement (commande sync). Lorsque le système s'arrête, il recopie également cette table en mémoire vers son bloc.

4.3.4. Table des inodes

La taille de la table des inodes est définie à sa création et est stockée sur la partition. Elle se compose d'enregistrements, appelés inodes, correspondant aux fichiers créés. Chaque enregistrement contient les adresses des blocs de données constituant le fichier.

Un numéro d'inode est unique au sein d'un système de fichiers.

Une copie est chargée en mémoire centrale dès l'initialisation du système. Cette copie est mise à jour dès modification et le système la sauvegarde périodiquement (commande sync). Lorsque le système s'arrête, il recopie également cette table en mémoire vers son bloc. Un fichier est géré par son numéro d'inode.

La taille de la table des inodes détermine le nombre maximum de fichiers que peut contenir le SF.

Informations présentes dans la table des inodes :

- Numéro d'inode ;
- Type de fichier et permissions d'accès ;
- Numéro d'identification du propriétaire ;

- Numéro d'identification du groupe propriétaire ;
- Nombre de liens sur ce fichier ;
- Taille du fichier en octets ;
- Date du dernier accès au fichier ;
- Date de la dernière modification du fichier ;
- Date de la dernière modification de l'inode (= création) ;
- Tableau de plusieurs pointeurs (table de blocs) sur les blocs logiques contenant les morceaux du fichier.

4.3.5. Zone de données

Sa taille correspond au reste de l'espace disponible de la partition. Cette zone contient les catalogues correspondant à chaque répertoire ainsi que les blocs de données correspondant aux contenus des fichiers.

Afin de garantir la cohérence du système de fichiers, une image du super-bloc et de la table des inodes est chargée en mémoire (RAM) lors du chargement du système d'exploitation afin que toutes les opérations d'E/S se fassent à travers ces tables du système. Lorsque l'utilisateur crée ou modifie des fichiers, c'est en premier lieu cette image mémoire qui est actualisée. Le système d'exploitation doit donc régulièrement actualiser le super-bloc du disque logique (commande sync).

Ces tables sont inscrites sur le disque dur lors de l'arrêt du système.

En cas d'arrêt brutal, le système de fichiers peut perdre sa cohérence et provoquer des pertes de données.

4.3.6. Réparation du système de fichiers

Il est possible de vérifier la cohérence d'un système de fichiers à l'aide de la commande **fsck**.

En cas d'erreurs, des solutions sont proposées afin de réparer les incohérences. Après réparation, les fichiers restant sans entrées dans la table des inodes sont rattachés au dossier **/lost+found** du lecteur logique.

Commande fsck

fsck est un outil en mode console de contrôle d'intégrité et réparation pour les systèmes de fichiers Linux.

Syntaxe de la commande fsck.

```
fsck [-sACVRTNP] [ -t fstype ] filesys
```

Exemple :

```
[root]# fsck /dev/sda1
```

Pour vérifier la partition racine, il est possible de créer un fichier forcefsck et de redémarrer ou de faire un shutdown avec l'option -F.

```
[root]# touch /forcefsck
[root]# reboot
ou
[root]# shutdown –r -F now
```

La partition devant être vérifiée doit impérativement être démontée.

4.4. Organisation d'un système de fichiers

Par définition, un Système de Fichiers est une structure arborescentre de répertoires construite à partir d'un répertoire racine (un périphérique logique ne peut contenir qu'un seul système de fichiers).

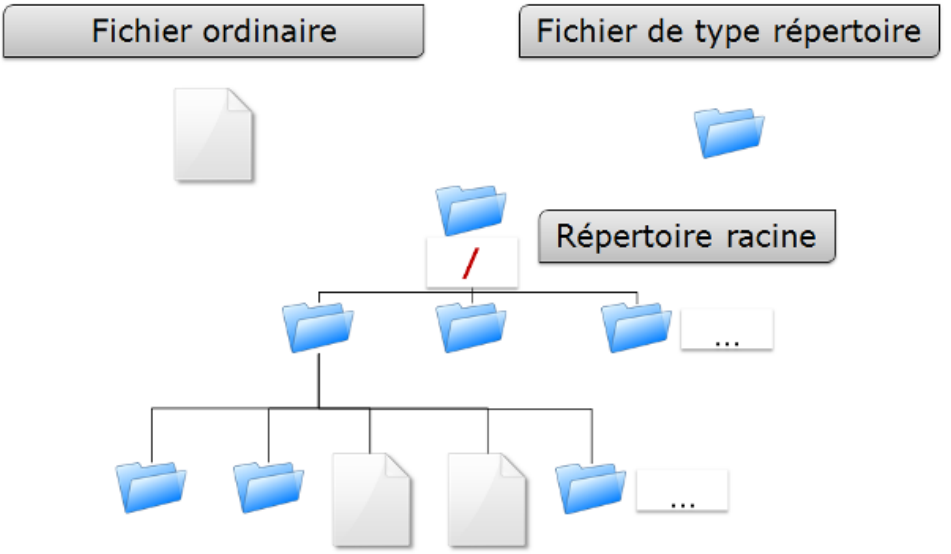


Figure 4.8. Organisation du système de fichiers

Sous Linux, tout est fichier.

Document texte, répertoire, binaire, partition, resssource réseau, écran, clavier, noyau Unix, programme utilisateur, …

Linux répond à la norme FHS (Filesystems Hierarchy Standard) qui définit le nom des dossiers et leurs rôles.

Tableau 4.4. Organisation standard du système de fichiers

Répertoire	Observation	Abréviation
/	Contient les répertoires spéciaux	
/boot	Fichiers relatifs au démarrage du système	
/sbin	Commandes indispensables au démarrage système	system binaries
/bin	Exécutables des commandes de base du système	binaries
/usr/bin	Commandes d'administration système	
/lib	Librairies partagées et modules du noyau	libraries

Répertoire	Observation	Abréviation
/usr	Tout ce qui n'est pas nécessaire au fonctionnement minimal du système	UNIX System Resources
/mnt	Pour le montage de SF temporaires	mount
/media	Pour le montage de médias amovibles	
/root	Répertoire de connexion de l'administrateur	
/home	Données utilisateurs	
/tmp	Fichiers temporaires	temporary
/dev	Fichiers spéciaux des périphériques	device
/etc	Fichiers de configuration et de scripts	editable text configuration
/opt	Spécifiques aux applications installées	optional
/proc	Système de fichiers virtuel représentant les différents processus	processes
/var	Fichiers variables divers	variables

Montage, démontage…quelques affirmations :

- Pour effectuer un montage ou démontage, au niveau de l'arborescence, il ne faut pas se trouver sous le point de montage.
- Le montage sur un répertoire non vide n'efface pas le contenu. Il est seulement masqué.
- Seul l'administrateur peut effectuer des montages.
- Les points de montage devant être montés automatiquement au démarrage doivent être inscrit dans **/etc/fstab**.

4.4.1. Le fichier /etc/fstab

Ce fichier est lu au démarrage du système et contient les montages à effectuer. Chaque système de fichiers à monter est décrit sur une seule ligne, les champs étant séparés pas des espaces ou des tabultations.

Les lignes sont lues séquentiellement (fsck, mount, umount).

Structure du fichier /etc/fstab.

```
/dev/mapper/VolGroup-lv_root   /          ext4    defaults         1   1
UUID=46….92                    /boot      ext4    defaults         1   2
/dev/mapper/VolGroup-lv_swap   swap       swap    defaults         0   0
tmpfs                          /dev/shm   tmpfs   defaults         0   0
devpts                         /dev/pts   devpts  gid=5,mode=620   0   0
sysfs                          /sys       sysfs   defaults         0   0
proc                           /proc      proc    defaults         0   0

  1                              2          3         4              5   6
```

Champ	Description
1	Périphérique du système de fichiers (/dev/sda1, UUID=…, …)
2	Nom du point de montage, **chemin absolu** (excepté **swap**)
3	Type de système de fichiers (ext4, swap, …)
4	Options particulières pour le montage (defaults, ro, …)
5	Active ou non la gestion des sauvegardes (0:non sauvegardé, 1:sauvegardé)
6	Ordre de vérification lors du contrôle du SF par la commande **fsck** (0:pas de contrôle, 1:prioritaire, 2:non prioritaire)

La commande **mount -a** permet de prendre en compte les nouveaux montages sans redémarrage. Ils sont ensuite inscrits dans le fichier **/etc/mtab** qui contient les montages actuels.

Seuls les points de montages inscrits dans **/etc/fstab** seront montés au redémarrage.

Il est possible de faire une copie du fichier **/etc/mtab** ou de copier son contenu vers **/etc/fstab**.

4.4.2. Commandes de gestion des montages

Commande mount

La commande **mount** permet de monter et de visualiser les lecteurs logiques dans l'arborescence.

Syntaxe de la commande mount.

```
mount [-option] [device] [directory]
```

Exemple :

```
[root]# mount /dev/sda7 /home
```

Tableau 4.5. Options de la commande mount

Option	Description
-n	Monte sans écrire dans /etc/fstab
-t	Indique le type de système de fichiers à utiliser
-a	Monte tous les systèmes de fichiers mentionnés dans /etc/fstab
-r	Monte le système de fichiers en lecture seule (équivalent **-o ro**)
-w	Monte le système de fichiers en lecture/écriture, par défaut (équivalent **-o rw**)
-o	Argument suivi d'une liste d'option(s) séparée(s) par des virgules (remount, ro, …)

La commande **mount** seule permet de visualiser tous les systèmes de fichiers montés.

Commande umount

La commande **umount** permet de démonter les lecteurs logiques.

Syntaxe de la commande umount.

```
umount [-option] [device] [directory]
```

Exemple :

```
[root]# umount /home
[root]# umount /dev/sda7
```

Tableau 4.6. Options de la commande umount

Option	Description
-n	Démonte sans écrire dans /etc/fstab
-r	Si le démontage échoue, remonte en lecture seule
-f	Force le démontage
-a	Démonte tous les systèmes de fichiers mentionnés dans /etc/fstab

Pour le démontage, il ne faut pas rester en dessous du point de montage. Sinon, le message d'erreur suivant s'affiche : **"device is busy"**.

4.5. Types de fichiers

Comme dans tout système, afin de pouvoir se retrouver dans l'arborescence et la gestion des fichiers, il est important de respecter des règles de nommage des fichiers.

- Les fichiers sont codés sur 255 caractères ;
- Tous les caractères ASCII sont utilisatbles ;
- Les majuscules et minuscules sont différenciées ;
- Pas de notion d'extension.

Les groupes de mots séparés par des espaces doivent être encadrès par des guillements :

```
[root]# mkdir "repertoire travail"
```

Le . sert seulement à cacher un fichier quand il débute le nom.

Sous Linux, la notion d'extension n'existe pas. Cependant, elle peut être utilisée mais fait alors partie intégrante du nom du fichier.

Exemples de conventions d'extension :

- .c : fichier source en langage C ;
- .h : fichier d'entête C et Fortran ;
- .o : fichier objet en langage C ;
- .tar : fichier de données archivées avec l'utilitaire tar ;
- .cpio : fichier de données archivées avec l'utilitaire cpio ;
- .gz : fichier de données compressées avec l'utilitaire gzip ;
- .html : page web.

4.5.1. Détails du nom d'un fichier

```
[root]# ls -liah /usr/bin/passwd
18 -rwxr-xr-x. 1 root root 26K 22 févr. 2012 /usr/bin/passwd
1  2    3      4  5    6    7       8               9
```

Champ	Description
1	Numéro d'inode
2	Type de fichiers
3	Droits d'accès
4	Nombre de liens (ordinaire) ou sous-répertoires (répertoires)
5	Nom du propriétaire
6	Nom du groupe
7	Taille (octet, kilo, méga)
8	Date de la dernière mise à jour

Champ	Description
9	Nom du fichier

4.5.2. Différents types de fichiers

On retrouve sur un système les types de fichiers suivants :

- Ordinaires (textes, binaires, …) ;
- Répertoires ;
- Spéciaux (imprimantes, écrans, …) ;
- Liens ;
- Communications (tubes et socket).

Fichiers ordinaires

Ce sont des fichiers textes, programmes (sources), exécutables (après compilation) ou fichiers de données (binaires, ASCII) et multimédias.

```
[root]# ls -l fichier
-rwxr-xr-x   1   root   root   26   nov   31   15:21 fichier
```

Le tiret **-** au début du groupe de droits indique qu'il s'agit d'un fichier de type ordinaire.

Fichiers répertoires

Les fichiers de type répertoire contiennent des références à d'autres fichiers.

Par défaut dans chaque répertoire sont présents **.** et **..** .

Le **.** représente la position dans l'arborescence.

Le **..** représente le père de la position courante.

```
[root]# ls -l repertoire
drwxr-xr-x   1   root   root   26   nov   31   15:21 repertoire
```

La lettre **d** au début du groupe de droits indique qu'il s'agit d'un fichier de type répertoire.

Fichiers spéciaux

Afin de communiquer avec les périphériques (disques durs, imprimantes, …), Linux utilise des fichiers d'interface appelés fichiers spéciaux (device file ou special file). Ils permettent donc d'identifier les périphériques.

Ces fichiers sont particuliers car ils ne contiennent pas de données mais spécifient le mode d'accès pour communiquer avec le périphérique.

Ils sont déclinés en deux modes :

- mode **bloc** ;
- mode **caractère**.

Le fichier spécial **mode bloc** permet en utilisant les buffers système de transférer des données vers le périphérique.

```
[root]# ls -l /dev/sda
brw-------   1   root  root  8, 0 jan 1 1970 /dev/sda
```

La lettre **b** au début du groupe de droits indique qu'il s'agit d'un fichier spécial bloc.

Le fichier spécial **mode caractère** est utilisé pour transférer des données vers le périphérique sous forme de flux un caractère à la fois sans utiliser de buffer. Ce sont les périphériques comme l'imprimante, l'écran ou les bandes DAT, …

La sortie standard est l'écran.

```
[root]# ls -l /dev/tty0
crw-------   1   root  root  8, 0 jan 1 1970 /dev/tty0
```

La lettre **c** au début du groupe de droits indique qu'il s'agit d'un fichier spécial caractère.

Fichiers de communication

Il s'agit des fichiers tubes (pipes) et des fichiers sockets.

Les fichiers tubes passent les informations entre processus par FIFO (first in first out). Un processus écrit de informations transitoires dans un fichier *pipe* et un autre les lit. Après lecture, les informations ne sont plus accessibles.

Les fichiers sockets permettent la communication bidirectionnelle interprocessus (sur système local ou distant). Ils utilisent un inode du système de fichiers.

Fichiers liens

Ces fichiers donnent la possibilité de donner plusieurs noms logiques à un même fichier physique. Un nouveau point d'accès au fichier est par conséquent créé.

On distingue deux types de fichiers lien :

- Les liens physiques ;
- Les liens symboliques.

Le lien physique

Le fichier lien et le fichier source ont le même numéro d'inode et le compteur de lien est incrémenté. Il est impossible de lier des répertoires et des fichiers de système de fichiers différents.

Si le fichier source est détruit, le compteur est décrémenté et le fichier lien accède toujours au fichier.

Commande ln

La commande **ln** permet de créer des liens

```
[root]# ls -li lettre
666 -rwxr--r-- 1 root root … lettre

[root]# ln /home/paul/lettre /home/jack/lire

[root]# ls -li /home/*/l*
666 -rwxr--r-- 2 root root … lettre
```

```
666 -rwxr--r-- 2 root root … lire
```

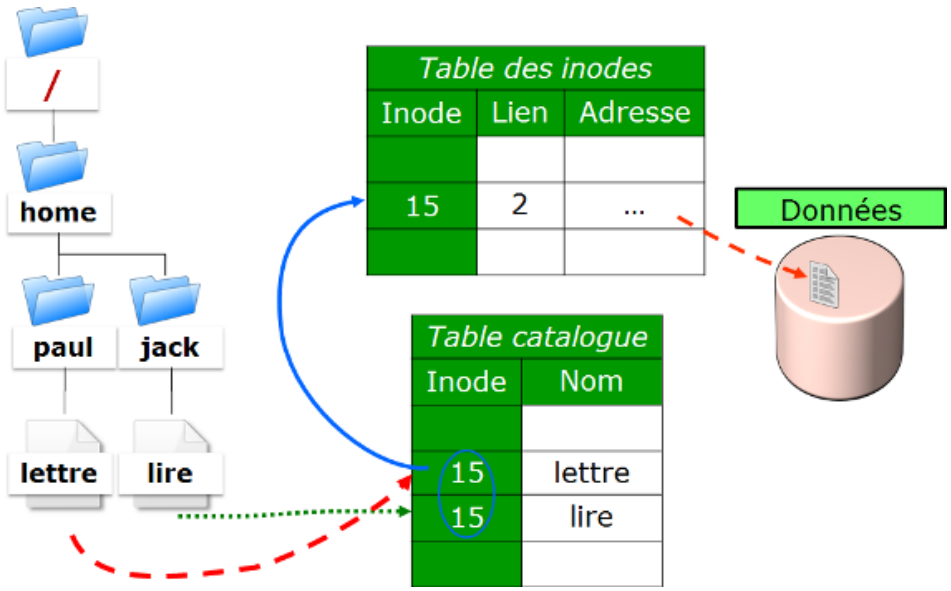


Figure 4.9. Représentation d'un lien physique

Le lien symbolique

Contrairement au lien physique, le lien symbolique implique la création d'un nouvel **inode**. Au niveau du lien symbolique, seul un chemin d'accès est stocké dans la table des inodes.

Le fichier créé ne contient qu'une indication sur le chemin permettant d'atteindre le fichier. Cette notion n'a plus les limitations des liens physiques et il est désormais possible de lier des répertoires et des fichiers appartenant à des systèmes de fichiers différents.

Si le fichier source est détruit, le fichier lien ne peut plus accéder au fichier.

```
[root]# ls -li lettre
666 -rwxr--r--- 1 root root … lettre

[root]# ln -s /home/paul/lettre /tmp/lire
```

```
[root]# ls -li /home/paul/lettre /tmp/lire
666 -rwxr--r--- 1 root root … lettre
678 lrwxrwxrwx 1 root root … lire -> lettre
```

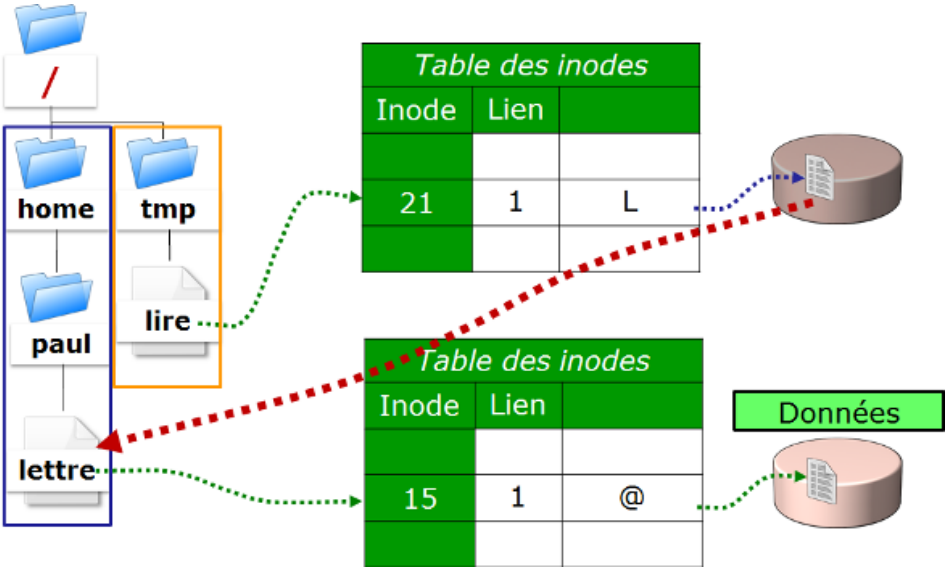


Figure 4.10. Représentation d'un lien symbolique

4.6. Attributs des fichiers

Linux est système d'exploitation multi-utilisateurs où l'accès aux fichiers est contrôlé.

Ces contrôles sont fonctions :

- des permissions d'accès au fichier ;
- des utilisateurs (ugo).

La commande **ls -l** permet afficher les attributs.

Il existe 4 droits d'accès aux fichiers :

- **r**ead (lecture) ;
- **w**rite (écriture) ;
- e**x**ecution (exécution) ;
- **-** aucun droit.

Les droits associés aux fichiers diffèrent de ceux associés aux répertoires (voir ci-dessous).

Les types d'utilisateurs associés aux droits d'accès des fichiers sont :

- **u**ser (propriétaire) ;
- **g**roup (groupe propriétaire) ;
- **o**thers (les autres) ;

Dans certaines commandes, il est possible de désigner tout le monde avec **a** (all).

a = ugo

4.6.1. Droits associés aux fichiers ordinaires

- **r**ead : Permet la lecture d'un fichier (cat, less, …) et autorise la copie (cp, …).
- **w**rite : Aurotise la modification du contenu du fichier (cat, », vim, …).
- e**x**ecute : Considère le fichier comme une commande (binaire, script).
- **-** : Aucune permission.

Déplacer ou renommer un fichier dépend des droits du répertoire cible. Supprimer un fichier dépend des droits du répertoire parent.

4.6.2. Droits associés aux répertoires

- **r**ead : Permet la lecture du contenu d'un répertoire (ls -R).
- **w**rite : Aurotise la modification du contenu d'un répertoire (touch) et permet la **création et suppression de fichiers** si la permission **x** est activée.
- e**x**ecute : Permet de descendre dans le répertoire (cd).
- **-** : Aucun droit.

4.6.3. Gestion des attributs

L'affichage des droits se fait à l'aide de la commande **ls -l**

```
[root]# ls -l /tmp/fichier
-rwxrw-r-x  1  root  sys  ... /tmp/fichier
  1  2  3       4     5
```

Champ	Description
1	Permissions du propriétaire (**u**ser), ici **rwx**
2	Permissions du groupe propriétaire (**g**roup), ici **rw-**
3	Permissions des autres utilisateurs (**o**thers), ici **r-x**
4	Propriétaire du fichier
5	Groupe propriétaire du fichier

Les permissions s'appliquent sur **u**ser, **g**roup et **o**ther (**ugo**) en fonction du propriétaire et du groupe.

Par défaut, le propriétaire d'un fichier est celui qui le crée. Le groupe du fichier est le groupe du propriétaire qui a créé le fichier. Les autres sont ceux qui ne sont pas concernés par les cas précédents.

La modification des attributs s'effectue à l'aide de la commande **chmod**

Seuls l'administrateur et le propriétaire d'un fichier peuvent modifier les droits d'un fichier.

Commande chmod

La commande **chmod** permet de modifier les autorisations d'accès à un fichier.

```
chmod [option] mode fichier
```

L'indication de mode peut être une représentation octale (ex : 744) ou une représentation symbolique ([ugoa][+=-][rwxst]).

Plusieurs opérations symboliques peuvent être séparées par des virgules

Exemple :

```
[root]# chmod -R u+rwx,g+wx,o-r /tmp/fichier1
[root]# chmod g=x,o-r /tmp/fichier2
[root]# chmod -R o=r /tmp/fichier3

[root]# ls -l /tmp/fic*
-rwxrwx--- 1 root root … /tmp/fichier1
-rwx--x--- 1 root root … /tmp/fichier2
-rwx--xr-- 1 root root … /tmp/fichier3
```

```
[root]# chmod 741 /tmp/fichier1
[root]# chmod -R 744 /tmp/fichier2

[root]# ls -l /tmp/fic*
-rwxr----x 1 root root … /tmp/fichier1
-rwxr--r-- 1 root root … /tmp/fichier2
```

Option	Observation
-R	Modifier récursivement les autorisations des répertoires et de leurs contenus

Il existe deux méthodes pour effectuer les changements de droits :

- La méthode **octale** ;
- La méthode **symbolique**.

Les droits des fichiers et des répertoires ne sont pas dissociés. Pour certaines opérations, il faudra connaître les droits du répertoire contenant le fichier.

Un fichier protégé en écriture peut être supprimé par un autre utilisateur dans la mesure où les droits du répertoire qui le contient autorisent cet utilisateur à effectuer cette opération.

Principe de la méthode octale

Chaque droit possède une valeur.

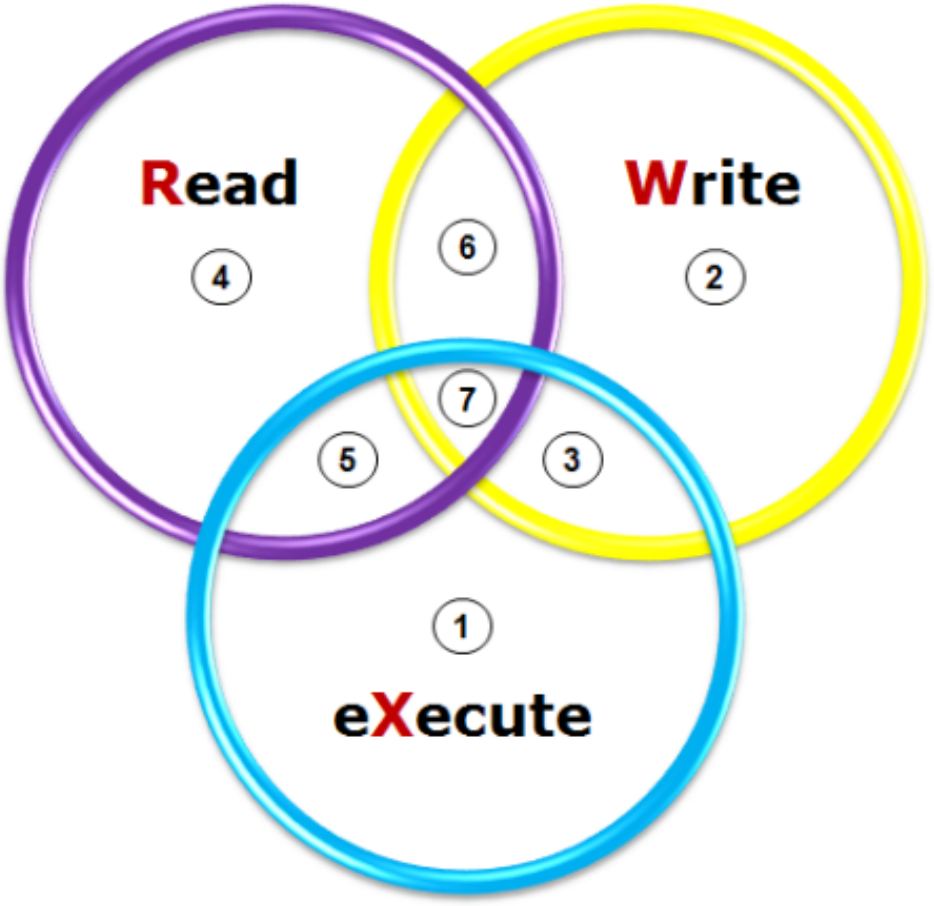


Figure 4.11. Méthode octale

```
[root]# ls -l /tmp/fichier
-rwxrwxrwx  1  root  root  ... /tmp/fichier
```

Figure 4.12. Droits 777

```
[root]# chmod 741 /tmp/fichier
-rwxr----x  1  root  root  ... /tmp/fichier
```

Figure 4.13. Droits 741

Principe de la méthode symbolique

Cette méthode peut être considérée comme une association “littérale” entre un type d’utilisateur, un opérateur et des droits.

Figure 4.14. Méthode symbolique

```
[root]# chmod u+rwx,g+wx,o-r /tmp/fichier
[root]# chmod g=x,o-r /tmp/fichier
[root]# chmod o=r /tmp/fichier
```

```
[root]# ls -l /tmp/fichier
----r--r-- 1 root root … /tmp/fichier

[root]# chmod u+rwx,g+wx,o-r /tmp/fichier
```

```
[root]# ls -l /tmp/fichier
-rwxrwx--- 1 root root … /tmp/fichier
```

4.6.4. Les droits particuliers

En complément des droits fondamentaux (rwx), il existe les droits particuliers :

- set-user-ID
- set-group-ID (SGID)
- sticky-bit (SUID)

Comme pour les droits fondamentaux, les droits particuliers possèdent chacun une valeur. Celle-ci se place avant l'ensemble de droits **ugo**.

suid	sgid	sticky -bit	user			group			other		
s	s	t	r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1	4	2	1

Figure 4.15. Les droits particuliers

S, S et **T** en majuscules **si le droit n'existe pas**.

Le Sticky-bit

Une des particularités des droits sous Linux est que le droit d'écrire sur un répertoire permet également de supprimer **tous** les fichiers, propriétaire ou non.

Le sticky-bit positionné sur le répertoire ne permettra aux utilisateurs d'effacer que les fichiers dont ils sont propriétaires.

La mise en place du sitcky-bit peut s'effectuer comme ci-dessous :

Méthode octale :

```
[root]# chmod 1777 repertoire
```

Méthode symbolique :

```
[root]# chmod o+t repertoire
```

```
[root]# ls -l
drwxrwxrwt … repertoire
```

SUID et SGID

Ces droits permettent d'exécuter une commande suivant les droits positionnés sur la commande et non plus suivant les droits de l'utilisateur.

La commande s'exécute avec l'identité du propriétaire (**suid**) ou du groupe (**sgid**) de la commande.

L'identité de l'utilisateur demandant l'exécution de la commande n'est plus prise en compte.

Il s'agit d'une possibilité supplémentaire de droits d'accès attribués à un utilisateur lorsqu'il est nécessaire qu'il dispose des mêmes droits que ceux du propriétaire d'un fichier ou ceux du groupe concerné.

En effet, un utilisateur peut avoir à exécuter un programme (en général un utilitaire système) mais ne pas avoir les droits d'accès nécessaires. En positionnant les droits adéquats ("**s**" au niveau du propriétaire et/ou au niveau du groupe), l'utilisateur du programme possède, pour le temps d'exécution de celui-ci, l'identité du propriétaire (ou celle du groupe) du programme.

Exemple :

Le fichier **/usr/bin/passwd** est un fichier exécutable (une commande) qui porte un **SUID**.

Lorsque l'utilisateur bob va le lancer, ce dernier devra accéder au fichier **/etc/shadow**, or les droits sur ce fichier ne permettent pas à bob d'y accéder.

Ayant un **SUID** cette commande sera exécutée avec l'UID de root et le GID de root. Ce dernier étant le propriétaire du fichier **/etc/shadow**, il aura les droits en lecture.

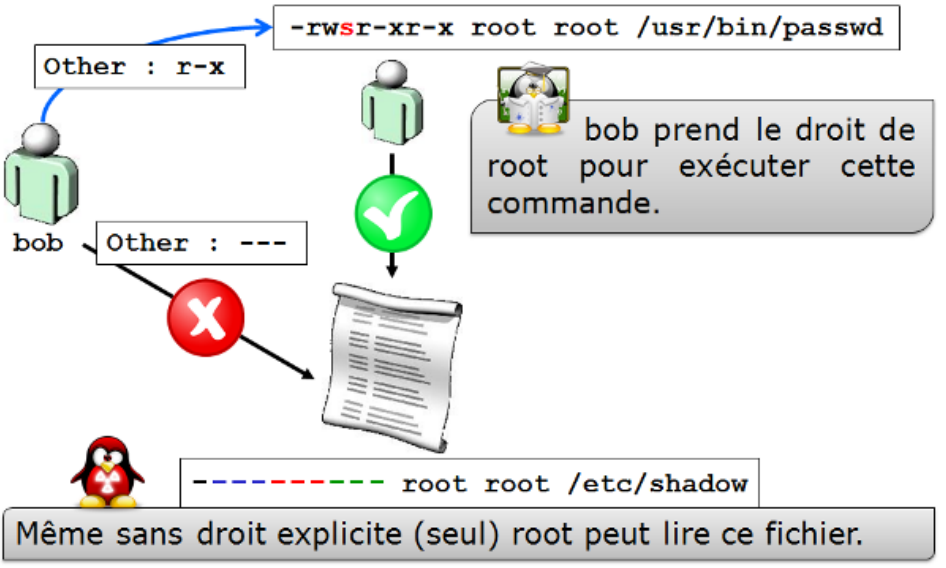


Figure 4.16. Fonctionnement du SUID

La mise en place du SUID et SGID peut s'effectuer comme ci-dessous :

Méthode octale :

```
[root]# chmod 4777 commande1
[root]# chmod 2777 commande2
```

Méthode symbolique :

```
[root]# chmod u+s commande1
[root]# chmod g+s commande2
```

```
[root]# ls -l
-rwsrwxrwx … commande1
-rwxrwsrwx … commande2
```

4.7. Droits par défaut et masque

Lors de sa création, un fichier ou un répertoire possède déjà des permissions.

- Pour un répertoire : **rwxr-xr-x** soit **755**

- Pour un fichier : **rw-r–r–** soit **644**

Ce comportement est défini par le **masque par défaut**.

Le principe est d'enlever la valeur définit du masque aux droits maximums.

Pour un répertoire :

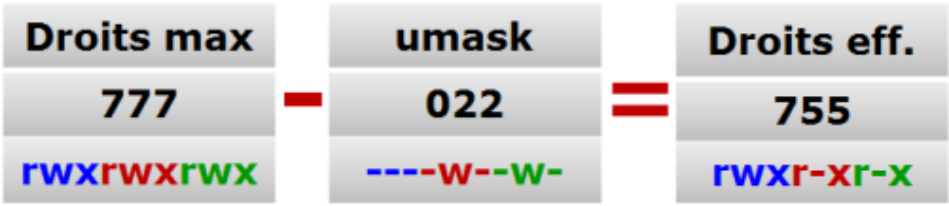


Figure 4.17. Droits par défaut d'un répertoire

Pour un fichier, les droits d'exécution sont retirés :

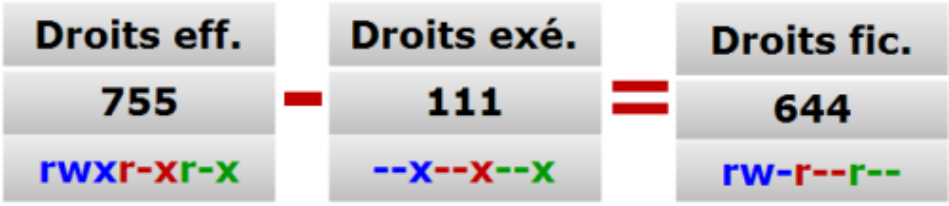


Figure 4.18. Droits par défaut d'un fichier

4.7.1. Commande umask

La commande **umask** permet d'afficher et de modifier le masque.

```
umask [option] [mode]
```

Exemple :

```
[root]# umask
0033
[root]# umask 025
[root]# umask
0025
```

Tableau 4.7. Options de la commande umask

Option	Description
-S	Affichage symbolique

umask n’affecte pas les fichiers existants.

umask modifie le masque jusqu'à la déconnexion. Pour garder la valeur, il faut modifier les fichiers de profile suivants :

Pour tous les utilisateurs :

- /etc/profile
- /etc/bashrc

Pour un utilisateur en particulier :

- ~/.bashrc

5

Gestion des processus

5.1. Généralités

Un système d'exploitation se compose de processus. Ces derniers,sont exécutés dans un ordre bien précis et observent des liens de parenté entre eux. On distingue deux catégories de processus, ceux axés sur l'environnement utilisateur et ceux sur l'environnement matériel.

Lorsqu'un programme s'exécute, le système va créer un processus en plaçant les données et le code du programme en mémoire et en créant **une pile d'exécution**. Un processus est donc une instance d'un programme auquel est associé un environnement processeur (Compteur Ordinal, registres, etc.) et un environnement mémoire.

Chaque processus dispose :

- d'un **PID** : Process IDentifiant, identifiant unique de processus ;
- d'un **PPID** : Parent Process IDentifiant, identifiant unique de processus parent.

Par filiations successives, le processus **init** est le père de tous les processus.

- Un processus est toujours créé par un processus père ;
- Un processus père peut avoir plusieurs processus fils.

Il existe une relation père / fils entre les processus, un processus fils est le résultat de l'appel système de la primitive fork() par le processus père qui duplique son propre code pour créer un fils. Le PID du fils est renvoyé au processus père pour qu'il puisse dialoguer avec. Chaque fils possède l'identifiant de son père, le **PPID**.

Le numéro PID représente le processus au moment de son exécution. À la fin de celui-ci, le numéro est de nouveau disponible pour un autre processus. Exécuter plusieurs fois la même commande produira à chaque fois un PID différent.

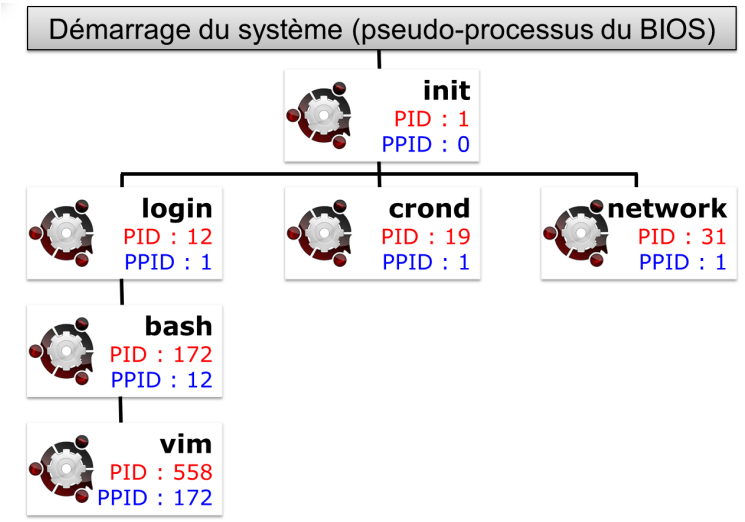


Figure 5.1. Filiation des processus

Les processus ne sont pas à confondre avec les threads. Chaque processus possède son propre contexte mémoire (ressources et espace d'adressage) alors que les threads issus d'un même processus partagent ce même contexte.

5.2. Visualisation des processus

La commande **ps** affiche l'état des processus en cours.

Syntaxe de la commande ps.

```
ps [-e] [-f] [-u login]
```

Exemple :

```
[root]# ps -fu root
```

Tableau 5.1. Options principales de la commande ps

Option	Description
-e	Affiche tous les processus.
-f	Affiche des informations supplémentaires.
-u login	Affiche les processus de l'utilisateur.

Quelques options supplémentaires :

Tableau 5.2. Options supplémentaires de la commande ps

Option	Description
-g	Affiche les processus du groupe.
-t tty	Affiche les processus exécutés à partir du terminal.
-p PID	Affiche les informations du processus.
-H	Affiche les informations sous forme d'arborescence.
-l	Affiche des informations supplémentaires.

Sans option précisée, la comme ps n'affiche que les processus exécutés à partir du terminal courant.

Le résultat est affiché par colonnes :

```
[root]# ps -ef
UID  PID PPID C STIME  TTY TIME      CMD
root 1   0    0 Jan01  ?   00:00/03  /sbin/init
```

Tableau 5.3. Descriptions des colonnes du résultat de la commande ps -ef

Colonne	Description
UID	Utilisateur propriétaire.
PID	Identifiant du processus.
PPID	Identifiant du processus parent.
C	Priorité du processus.
STIME	Date et heure d'exécution.
TTY	Terminal d'exécution.

Colonne	Description
TIME	Durée de traitement.
CMD	Commande exécutée.

5.3. Types de processus

Le processus utilisateur :

- il est démarré depuis un terminal associé à un utilisateur ;
- il accède aux ressources via des requêtes ou des démons.

Le processus système (démon) :

- il est démarré par le système ;
- il n'est associé à aucun terminal et son propriétaire est un utilisateur système (souvent root) ;
- il est chargé lors du démarrage, il réside en mémoire et est en attente d'un appel ;
- il est généralement identifié par la lettre "d" associé au nom du processus.

Les processus système sont donc appelés démons de l'anglais **daemon** (Disk And Execution MONitor).

5.4. Permissions et droits

À l'exécution d'une commande, les identifiants de l'utilisateur sont transmis au processus créé.

Par défaut, les **UID** et **GID** effectifs (du processus) sont donc identiques aux **UID et GID réels** (les UID et GID de l'utilisateur qui a exécuté la commande).

Lorsqu'un **SUID** (et/ou un **SGID**) est positionné sur une commande, les UID (et/ou GID) effectifs deviennent ceux du propriétaire (et/ou du groupe propriétaire) de la commande et non plus celui de l'utilisateur ou du groupe utilisateur qui a lancé la commande. **UID effectifs et réels** sont donc **différents**.

À chaque accès à un fichier, le système vérifie les droits du processus en fonction de ses identifiants effectifs.

5.5. Gestion des processus

Un processus ne peut pas être exécuté indéfiniment, cela se ferait au détriment des autres processus en cours et empêcherait de faire du multitâche.

Le temps de traitement total disponible est donc divisé en petites plages et chaque processus (doté d'une priorité) accède au processeur de manière séquencée. Le processus prendra plusieurs états au cours de sa vie parmi les états :

- prêt : attend la disponibilité du processus ;
- en exécution : accède au processeur ;
- suspendu : en attente d'une E/S (entrée/sortie) ;
- arrêté : en attente d'un signal d'un autre processus ;
- zombie : demande de destruction ;
- mort : le père du processus tue son fils.

Le séquencement de fin de processus est le suivant :

1. Fermeture des fichiers ouverts ;
2. Libération de la mémoire utilisée ;
3. Envoi d'un signal au processus père et aux processus fils.

Lorsqu'un processus meurt, ses processus fils sont dits orphelins. Ils sont alors adoptés par le processus **init** qui se chargera de les détruire.

5.5.1. La priorité d'un processus

Le processeur travaille en temps partagé, chaque processus occupe un quantum de temps du processeur.

Les processus sont classés par priorité dont la valeur varie de -20 (la priorité la plus élevée) à +20 (la priorité la plus basse).

La priorité par défaut d'un processus est 0.

5.5.2. Modes de fonctionnements

Les processus peuvent fonctionner de deux manières :

- **synchrone** : l'utilisateur perd l'accès au shell durant l'exécution de la commande. L'invite de commande réapparait à la fin de l'exécution du processus.
- **asynchrone** : le traitement du processus se fait en arrière-plan, l'invite de commande est réaffichée immédiatement.

Les contraintes du mode asynchrone :

- la commande ou le script ne doit pas attendre de saisie au clavier ;
- la commande ou le script ne doit pas retourner de résultat à l'écran ;
- quitter le shell termine le processus.

5.6. Les commandes de gestion des processus

5.6.1. La commande kill

La commande kill envoie un signal d'arrêt à un processus.

Syntaxe de la commande kill.

```
kill [-signal] PID
```

Exemple :

```
[root]# kill -9 1664
```

Tableau 5.4. Codes numériques des principaux signaux d'arrêt des processus

Code	Signal	Description
2	SIGINT	Interruption du processus (CTRL + D)
9	SIGKILL	Terminaison immédiate du processus
15	SIGTERM	Terminaison propre du processus
18	SIGCONT	Reprise du processus
19	SIGSTOP	Suspension du processus

Les signaux sont les moyens de communication entre les processus. La commande kill permet d'envoyer un signal à un processus.

La liste complète des signaux pris en compte par la commande kill est disponible en tapant la commande :

```
[root]# man 7 signal
```

5.6.2. La commande nohup

La commande **nohup** permet de lancer un processus indépendant d'une connexion.

Syntaxe de la commande nohup.

```
nohup commande
```

Exemple :

```
[root]# nohup MonProgramme.sh 0</dev/null &
```

Nohup ignore le signal SIGHUP envoyé lors de la déconnexion d'un utilisateur.

Nohup gère les sorties et erreur standards, mais pas l'entrée standard, d'où la redirection de cette entrée vers / dev/null.

[CTRL] + [Z]

En appuyant simultanément sur les touches **CTRL**+**Z**, le processus synchrone est temporairement suspendu. L'accès au prompt est rendu après affichage du numéro du processus venant d'être suspendu.

5.6.3. Instruction &

L'instruction **&** exécute la commande en mode asynchrone (la commande est alors appelée job) et affiche le numéro de job. L'accès au prompt est ensuite rendu.

Exemple :

```
[root]# time ls -lR / > list.ls 2> /dev/null &
```

```
[1] 15430
[root]#
```

Le numéro de **job** est obtenu lors de la mise en tâche de fond et est affiché entre crochets, suivi du numéro de PID.

5.6.4. Les commandes fg et bg

La commande **fg** place le processus au premier plan :

```
[root]# time ls -lR / > list.ls 2>/dev/null &
[root]# fg 1
time ls -lR / > list.ls 2/dev/null
```

tandis que la commande **bg** le place à l'arrière plan :

```
[CTRL]+[Z]
^Z
[1]+ Stopped
[root]# bg 1
[1] 15430
[root]#
```

Qu'il ait été mis à l'arrière plan lors de sa création grâce à l'argument **&** ou plus tard avec les touches **CTRL+Z**, un processus peut être ramené au premier plan grâce à la commande **fg** et à son numéro de job.

5.6.5. La commande jobs

La commande jobs affiche la liste des processus tournant en tâche de fond et précise leur numéro de job.

Exemple :

```
[root]# jobs
[1]- Running    sleep 1000
[2]+ Running    find / > arbo.txt
```

Les colonnes représentent :

1. le numéro de job ;
2. ordre de passage des processus
   - un **+** : le processus est le prochain processus à s'exécuter par défaut avec fg ou bg ;
   - un **-** : le processus est le prochain processus qui prendra le + ;
3. Running (en cours de traitement) ou Stopped (processus suspendu).
4. la commande

5.6.6. Les commandes nice/renice

La commande nice permet d'exécuter une commande en précisant sa priorité.

Syntaxe de la commande nice.

```
nice priorité commande
```

Exemple :

```
[root]# nice -n+15 find / -name "fichier"
```

Contrairement à root, un utilisateur standard ne peut que réduire la priorité d'un processus. Seules les valeurs entre +0 et +19 seront acceptées.

La commande **renice** permet de modifier la priorité d'un processus en cours d'exécution.

Syntaxe de la commande renice.

```
renice priorité [-g GID] [-p PID] [-u UID]
```

Exemple :

```
[root]# renice +15 -p 1664
```

Tableau 5.5. Options principales de la commande renice

Option	Description
-g	GID du groupe propriétaire du processus.

Option	Description
-p	PID du processus.
-u	UID du propriétaire du processus.

La commande **renice** agit sur des processus déjà en cours d'exécution. Il est donc possible de modifier la priorité d'un processus précis, mais aussi de plusieurs processus appartenant à un utilisateur ou à un groupe.

5.6.7. La commande top

La commande top affiche les processus et leur consommation en ressources.

```
[root]# top
PID  USER PR NI ... %CPU %MEM   TIME+    COMMAND
2514 root 20 0       15    5.5 0:01.14   top
```

Tableau 5.6. Descriptions des colonnes du résultat de la commande top

Colonne	Description
PID	Identifiant du processus.
USER	Utilisateur propriétaire.
PR	Priorité du processus.
NI	Valeur du nice.
%CPU	Charge du processeur.
%MEM	Charge de la mémoire.
TIME+	Temps d'utilisation processeur.
COMMAND	Commande exécutée.

La commande **top** permet de contrôler les processus en temps réel et en mode interactif.

6

Sauvegardes et restaurations

La sauvegarde va permettre de répondre à un besoin de conservation et de restauration des données de manière sûre et efficace.

La sauvegarde permet de se protéger des éléments suivants :

- **Destruction** : Volontaire ou involontaire. Humaine ou technique. Virus, …
- **Suppression** : Volontaire ou involontaire. Humaine ou technique. Virus, …
- **Intégrité** : Données devenues inutilisables.

Aucun système n'est infaillible, aucun humain n'est infaillible, aussi pour éviter de perdre des données, il faut les sauvegarder pour être en mesure de les restaurer suite à un problème.

Les supports de sauvegarde sont conservés dans une autre pièce (voire bâtiment) que le serveur afin qu'un sinistre ne vienne pas détruire le serveur et les sauvegardes.

De plus, l'administrateur devra régulièrement vérifier que les supports soient toujours lisibles.

6.1. Généralités

Il existe deux principes, la **sauvegarde** et l'**archive**.

- L'archive détruit la source d'information après l'opération.
- La sauvegarde conserve la source d'information après l'opération.

Ces opérations consistent à enregistrer des informations dans un fichier, sur un périphérique ou un support (bandes, disques, …).

6.1.1. La démarche

La sauvegarde nécessite de l'administrateur du système beaucoup de discipline et une grande rigueur. Il est nécessaire de se poser les questions suivantes :

- Quel est le support approprié ?
- Que faut-il sauvegarder ?
- Nombre d'exemplaires ?
- Durée de la sauvegarde ?
- Méthode ?
- Fréquence ?
- Automatique ou manuelle ?
- Où la stocker ?
- Délai de conservation ?

6.1.2. Méthodes de sauvegardes

- **Complète** : un ou plusieurs **systèmes de fichiers** sont sauvegardés (noyau, données, utilitaires, …).
- **Partielle** : un ou plusieurs **fichiers** sont sauvegardés (configurations, répertoires, …).
- **Incrémentale** : uniquement les **fichiers modifiés** depuis la dernière sauvegarde sont sauvegardés.

6.1.3. Périodicité

- **Ponctuelle** : à un instant donné (avant une mise à jour du système, …).
- **Périodique** : Journalière, hebdomadaire, mensuelle, …

Avant une modification du système, il peut être utile de faire une sauvegarde. Cependant, il ne sert à rien de

sauvegarder tous les jours des données qui ne sont modifiées que tous les mois.

6.1.4. Méthodes de restauration

En fonction des utilitaires disponibles, il sera possible d'effectuer plusieurs types de restaurations.

- **Restauration complète** : arborescences, …
- **Restauration sélective** : partie d'arborescences, fichiers, …

Il est possible de restaurer la totalité d'une sauvegarde mais il est également possible d'en restaurer uniquement une partie. Toutefois, lors de la restauration d'un répertoire, les fichiers créés après la sauvegarde ne sont pas supprimés.

Pour récupérer un répertoire tel qu'il était au moment de la sauvegarde il convient d'en supprimer complètement le contenu avant de lancer la restauration.

6.1.5. Les outils

Il existe de nombreux utilitaires pour réaliser les sauvegardes.

- **outils éditeurs** ;
- **outils graphiques** ;
- **outils mode de commande** : **tar**, **cpio**, pax, dd, dump, …

Les commandes que nous verrons ici sont tar et cpio.

- tar :
  - simple d'utilisation ;
  - permet l'ajout de fichiers à une sauvegarde existante.
- cpio :
  - conserve les propriétaires ;
  - groupes, dates et droits ;
  - saute les fichiers endommagés ;

- système de fichiers complet.

Ces commandes sauvegardent dans un format propriétaire et standardisé.

6.1.6. Convention de nommage

L'emploi d'une convention de nommage permet de cibler rapidement le contenu d'un fichier de sauvegarde et d'éviter ainsi des restaurations hasardeuses.

- nom du répertoire ;
- utilitaire employé ;
- options utilisées ;
- date.

Le nom de la sauvegarde doit être un nom explicite.

La notion d'extension sous Unix n'existe pas.

6.1.7. Contenu d'une sauvegarde

Une sauvegarde contient généralement les éléments suivants :

- le fichier ;
- le nom ;
- le propriétaire ;
- la taille ;
- les permissions ;
- date d'accès.

Le numéro d'inode est absent.

6.1.8. Modes de stockage

Deux modes de stockage se distinguent :

- fichier sur le disque ;
- périphérique.

6.2. Tape ArchiveR - tar

La commande tar permet la sauvegarde sur plusieurs supports successifs (options multi-volumes).

Il est possible d'extraire tout ou partie d'une sauvegarde.

Tar sauvegarde implicitement en mode relatif même si le chemin des informations à sauvegarder est mentionné en mode absolu.

6.2.1. Consignes de restauration

Il faut se poser les bonnes questions

- quoi : Partielle ou complète ;
- où : Lieu où les données seront restaurées ;
- comment : Absolu ou relatif.

Avant une restauration, il faut prendre le temps de la réflexion et déterminer la méthode la mieux adaptée afin d'éviter toutes erreurs.

Les restaurations s'effectuent généralement après un problème qui doit être résolu rapidement. Une mauvaise restauration peut dans certains cas aggraver la situation.

6.2.2. La sauvegarde avec tar

L'utilitaire par défaut pour créer des archives dans les systèmes UNIX est la commande tar. Ces archives peuvent être compressées avec une compression gzip ou bzip.

Tar permet d'extraire aussi bien un seul fichier ou un répertoire d'une archive, visualiser son contenu ou valider son intégrité, etc.

Créer une archive

Créer une archive non-compressée s'effectue avec les clefs cvf :

Syntaxe de la commande tar pour créer une archive.

```
tar c[vf] [support] [fichiers(s)]
```

Exemple :

```
[root]# tar cvf /sauvegardes/home.133.tar
```

Tableau 6.1. Clefs principales de la commande tar

Clef	Description
c	Crée une sauvegarde.
v	Affiche le nom des fichiers traités.
f	Permet d'indiquer le nom de la sauvegarde (support).

Il n'y a pas de tiret '-' devant les clefs de tar !

Créer une sauvegarde en mode absolu

Syntaxe de la commande tar pour créer une archive en mode absolu.

```
tar c[vf]P [support] [fichiers(s)]
```

Exemple :

```
[root]# tar cvfP /sauvegardes/home.133.P.tar /home
```

Clef	Description
P	Créer une sauvegarde en mode absolu.

Avec la clef **P**, le chemin des fichiers à sauvegarder doit être renseigné en **absolu**. Si les deux conditions (clef **P** et chemin **absolu**) ne sont pas indiquées, la sauvegarde est en mode relatif.

Créer une archive compressée avec gzip

Créer une archive compressée en gzip s'effectue avec les clefs cvzf :

```
[root]# tar cvzf archive.tar.gz dirname/
```

Clef	Description
z	Compresse l'archive en gzip.

L'extension .tgz est une extension équivalente à .tar.gz

Conserver les clefs 'cvf' ('tvf' ou 'xvf') inchangée pour toutes les manipulations d'archives et simplement ajouter à la fin des clefs celle de compression simplifie la compréhension de la commande (par exemple 'cvfz' ou 'cvfj', etc.).

Créer une archive compressée avec bzip

Créer une archive compressée en bzip s'effectue avec les clefs cvfj :

```
[root]# tar cvfj archive.tar.bz2 dirname/
```

Clef	Description
j	Compresse l'archive en bzip2.

Les extensions .tbz et .tb2 sont des extensions équivalentes à .tar.bz2

gzip vs bzip2

bzip2 nécessite plus de temps pour compresser ou décompresser que gzip mais offre des ratios de compression supérieurs.

Extraire (untar) une archive

Extraire une archive *.tar s'effectue avec les clefs xvf :

```
[root]# tar xvf /sauvegardes/etc.133.tar etc/exports
[root]# tar xvfj /sauvegardes/home.133.tar.bz2
[root]# tar xvfP /sauvegardes/etc.133.P.tar
```

Se placer au bon endroit.

Vérifier le contenu de la sauvegarde.

Clef	Description
x	Extrait des fichiers de l'archive, compressée ou non.

Extraire une archive tar-gzippée (*.tar.gz) s'effectue avec les clefs xvfz

```
[root]# tar xvfz archive.tar.gz
```

Extraire une archive tar-bzippée (*.tar.bz2) s'effectue avec les clefs xvfj

```
[root]# tar xvfj archive.tar.bz2
```

Lister le contenu d'une archive

Visualiser le contenu d'une archive sans l'extraire s'effectue avec les clefs tvf :

```
[root]# tar tvf archive.tar
[root]# tar tvfz archive.tar.gz
[root]# tar tvfj archive.tar.bz2
```

Lorsque le nombre de fichiers contenus dans une archive devient important, il est possible de passer à la commande less le résultat de la commande tar par un pipe ou en utilisant directement la commande less :

```
[root]# tar tvf archive.tar | less
[root]# less archive.tar
```

Extraire uniquement un fichier d'une archive .tar, tar.gz ou tar.bz2

Pour extraire un fichier spécifique d'une archive tar, spécifier le nom du fichier à la fin de la commande tar xvf.

```
[root]# tar xvf archive.tar /path/to/file
```

La commande précédente permet de n'extraire que le fichier file de l'archive archive.tar.

```
[root]# tar xvfz archive.tar.gz /path/to/file
[root]# tar xvfj archive.tar.bz2 /path/to/file
```

Extraire uniquement un dossier d'une archive tar, tar.gz, tar.bz2

Pour n'extraire qu'un seul répertoire (ses sous-répertoires et fichiers inclus) d'une archive, spécifier le nom du répertoire à la fin de la commande tar xvf.

```
[root] tar xvf archive.tar /path/to/dir/
```

Pour extraire plusieurs répertoires, spécifier chacun des noms les uns à la suite des autres :

```
[root] tar xvf archive_file.tar /path/to/dir1/ /path/to/dir2/
[root] tar xvfz archive_file.tar.gz /path/to/dir1/ /path/to/dir2/
[root] tar xvfj archive_file.tar.bz2 /path/to/dir1/ /path/to/dir2/
```

Extraire un groupe de fichiers d'une archive tar, tar.gz, tar.bz2 grâce à des expressions régulières (regex)

Spécifier une regex pour extraire les fichiers correspondants au pattern spécifié.

Par exemple, pour extraire tous les fichiers avec l'extension .conf :

```
[root] tar xvf archive_file.tar --wildcards '*.conf'
```

Clefs :

- --wildcards *.conf correspond aux fichiers avec l'extension .conf.

Ajouter un fichier ou un répertoire à une archive existante

Il est possible d'ajouter des fichiers à une archive existante avec la clef r.

Par exemple, pour ajouter un fichier :

```
[root]# tar rvf archive.tar filetoadd
```

Le fichier filetoadd sera ajouté à l'archive tar existante. Ajouter un répertoire est similaire :

```
[root]# tar rvf archive_name.tar dirtoadd
```

Il n'est pas possible d'ajouter des fichiers ou des dossiers à une archive compressée.

```
[root]# tar rvfz archive.tgz filetoadd
tar: Cannot update compressed archives
Try `tar --help' or `tar --usage' for more
 information.
```

Vérifier l'intégrité d'une archive

L'intégrité d'une archive peut être testée avec la clef W au moment de sa création :

```
[root]# tar cvfW file_name.tar dir/
```

La clef W permet également de comparer le contenu d'une archive par rapport au système de fichiers :

```
[root]# tar tvfW file_name.tar
Verify 1/file1
1/file1: Mod time differs
1/file1: Size differs
Verify 1/file2
Verify 1/file3
```

La vérification avec la clef W ne peut pas être effectuée avec une archive compressée. Il faut utiliser la clef d :

```
[root]# tar dfz file_name.tgz
[root]# tar dfj file_name.tar.bz2
```

Estimer la taille d'une archive

La commande suivante estime la taille d'un fichier tar en KB avant de la créer :

```
[root]# tar cf - /directory/to/archive/ | wc -c
20480
[root]# tar czf - /directory/to/archive/ | wc -c
508
[root]# tar cjf - /directory/to/archive/ | wc -c
428
```

Ajout d'éléments à une sauvegarde existante

Syntaxe de la commande tar pour ajouter un élément à une sauvegarde existante.

```
tar {r|A}[clé(s)] [support] [fichiers(s)]
```

Exemple :

```
[root]# tar rvf /sauvegardes/home.133.tar /etc/passwd
```

Clef	Description
r	Ajoute un ou plusieurs fichiers à la fin d'une sauvegarde sur support à accès direct (disque dur).
A	Ajoute un ou plusieurs fichiers à la fin d'une sauvegarde sur support à accès séquentiel (bande).

Si la sauvegarde a été réalisée en mode relatif, ajoutez des fichiers en mode relatif. Si la sauvegarde a été réalisée en mode absolu, ajoutez des fichiers en mode absolu. En mélangeant les modes, vous risquez d'avoir des soucis au moment de la restauration.

Lire le contenu d'une sauvegarde

Syntaxe de la commande tar pour lire le contenu d'une sauvegarde.

```
tar t[clé(s)] [support]
```

Exemple :

```
[root]# tar tvf /sauvegardes/home.133.tar
[root]# tar tvfj /sauvegardes/home.133.tar.bz2
```

Clef	Description
t	Affiche le contenu d'une sauvegarde (compressée ou non).

Toujours vérifier le contenu d'une sauvegarde.

Tableau 6.2. Convention d'écriture de la commande Tar

Clés	Fichiers	Suffixe
cvf	home	home.tar
cvfP	/etc	etc.P.tar
cvfz	usr	usr.tar.gz
cvfj	usr	usr.tar.bz2
cvfPz	/home	home.P.tar.gz
cvfPj	/home	home.P.tar.bz2

6.3. CoPy Input Output - cpio

La commande cpio permet la sauvegarde sur plusieurs supports successifs sans indiquer d'options.

Il est possible d'extraire tout ou partie d'une sauvegarde.

cpio ne permet pas de sauvegarder directement une arborescence. L'arborescence ou fichiers sont donc transmis sous forme de liste à cpio.

Il n'y a aucune option, comme pour la commande tar, permettant de sauvegarder et de compresser en même temps. Cela s'effectue donc en deux temps : la sauvegarde puis la compression.

Pour effectuer une sauvegarde avec cpio, il faut préciser une liste des fichiers à sauvegarder.

Cette liste est fourni avec les commandes find, ls ou cat.

- find : parcourt une arborescence, récursif ou non ;
- ls : liste un répertoire, récursif ou non ;
- cat : lit un fichier contenant les arborescences ou fichiers à sauvegarder.

ls ne peut pas être utilisé avec -l (détails) ou -R (récursif).

Il faut une liste simple de noms.

6.3.1. Créer une sauvegarde

Syntaxe de la commande cpio.

```
[cde de fichiers |] cpio {-o| --create} [-options] [<fic-liste]
 [>support]
```

Exemple :

```
[root]# find /etc | cpio -ov > /sauvegardes/etc.cpio
```

Le résultat de la commande **find** est envoyé en entrée de la commande **cpio** par l'intermédiare du signe "|" (**AltGr**+**6**). Ici, la commande find /etc renvoie une liste de fichiers correspondant au contenu du répertoire /etc (en récursif) à la commande cpio qui en effectue la sauvegarde. Ne surtout pas oublier le signe > lors de la sauvegarde.

Tableau 6.3. Options principales de la commande cpio

Options	Description
-o	Crée une sauvegarde (output).
-v	Affiche le nom des fichiers traités.
-F	Désigne la sauvegarde à modifier (support).

Sauvegarde vers un support :

```
[root]# find /etc | cpio -ov > /dev/rmt0
```

Le support peut être de plusieurs types :

- /dev/rmt0 : lecteur de bande ;
- /dev/sda5 : une partition.

6.3.2. Type de sauvegarde

- Sauvegarde avec chemin relatif

```
[root]# cd /
[root]# find etc | cpio -o > /sauvegardes/etc.cpio
```

- Sauvegarde avec chemin absolu

```
[root]# find /etc | cpio -o > /sauvegardes/etc.A.cpio
```

Si le chemin indiqué au niveau de la commande "find" est en **absolu** alors la sauvegarde sera réalisée en absolu.

> Si le chemin indiqué au niveau de la commande “find” est en **relatif** alors la sauvegarde sera réalisée en relatif.

6.3.3. Ajouter à une sauvegarde

Syntaxe de la commande cpio pour ajouter un contenu.

```
[cde de fichiers |] cpio {-o| --create} -A [-options] [<fic-liste] {F|
>support}
```

Exemple :

```
[root]# find /etc/shadow | cpio -o -AF FicSyst.A.cpio
```

L’ajout de fichiers n’est possible que sur un support à accès direct.

Option	Description
-A	Ajoute un ou plusieurs fichiers à une sauvegarde sur disque.
-F	Désigne la sauvegarde à modifier.

6.3.4. Compresser une sauvegarde

- Sauvegarder **puis** compresser

```
[root]# find /etc | cpio  -o > etc.A.cpio
[root]# gzip /sauvegardes/etc.A.cpio
[root]# ls /sauvegardes/etc.A.cpio*
/sauvegardes/etc.A.cpio.gz
```

- Sauvegarder **et** compresser

```
[root]# find /etc | cpio –o | gzip > /sauvegardes/etc.A.cpio.gz
```

Il n’y a aucune option, comme pour la commande tar, permettant de sauvegarder et de compresser en même temps. Cela s’effectue donc en deux temps : la sauvegarde puis la compression.

La syntaxe de la première méthode est plus facile à comprendre et à retenir, car elle s'effectue en deux temps.

Pour la première méthode, le fichier de sauvegarde est automatiquement renommé par l'utilitaire gzip qui rajoute .gz à la fin du nom de ce fichier. De même l'utilitaire bzip2 rajoute automatiquement .bz2.

6.3.5. Lire le contenu d'une sauvegarde

Syntaxe de la commande cpio pour lire le contenu d'une sauvegarde cpio.

```
cpio -t [-options] [<fic-liste]
```

Exemple :

```
[root]# cpio -tv </sauvegardes/etc.152.cpio | less
```

Options	Description
-t	Lit une sauvegarde.
-v	Affiche les attributs des fichiers.

Après avoir réalisé une sauvegarde, il faut lire son contenu pour être certain qu'il n'y a pas eu d'erreur.

De la même façon, avant d'effectuer une restauration, il faut lire le contenu de la sauvegarde qui va être utilisée.

6.3.6. Restaurer une sauvegarde

Syntaxe de la commande cpio pour restaurer une sauvegarde.

```
cpio {-i| --extract} [-E fichier] [-options] [<support]
```

Exemple :

```
[root]#cpio -iv </sauvegardes/etc.152.cpio | less
```

Options	Description
-i	Restauration complète d'une sauvegarde .

Options	Description
-E fichier	Restaure uniquement les fichiers dont le nom est contenu dans fichier.
-d	Reconstruit l'arborescence manquante.
-u	Remplace tous les fichiers même s'ils existent.
--no-absolute-filenames	Permet de restaurer une archive effectuée en mode absolu de manière relative.

Par défaut, au moment de la restauration, les fichiers sur le disque dont la date de dernière modification est plus récente ou égale à la date de la sauvegarde ne sont pas restaurés (afin d'éviter d'écraser des informations récentes par des informations plus anciennes).

L'option -u permet au contraire de restaurer d'anciennes versions des fichiers.

Exemples :

- Restauration en absolu d'une sauvegarde absolue :

```
[root]# cpio -iv <home.A.cpio
```

- Restauration en absolu sur une arborescence existante :

```
[root]# cpio -iuv <home.A.cpio
```

L'option "u" permet d'écraser des fichiers existants à l'endroit où s'effectue la restauration. * Restauration en relatif d'une sauvegarde absolue :

```
[root]# cpio -iv --no-absolute-filenames <home.A.cpio
```

L'option longue "--no-absolute-filenames" permet une restauration en mode relatif. En effet le "/" en début de chemin est enlevé.

- Restauration en relatif d'une sauvegarde relative :

```
[root]# cpio -iv <etc.cpio
```

- Restauration en absolu du fichier « passwd » :

```
echo "/etc/passwd" > tmp;cpio -iuE tmp <etc.A.cpio; rm -f tmp
```

6.4. Utilitaires de compression - décompression

Le fait d'utiliser la compression au moment d'une sauvegarde peut présenter un certain nombre d'inconvénients :

- Allonge le temps de la sauvegarde ainsi que celui de la restauration.
- Rend impossible l'ajout de fichiers à cette sauvegarde.

Il vaut donc mieux effectuer une sauvegarde et la compresser qu'effectuer la compression lors de la sauvegarde.

6.4.1. Compresser avec gzip

Syntaxte de la commande gzip.

```
gzip [options] [fichier ...]
```

Exemple :

```
[root]# gzip usr.tar
[root]# ls
usr.tar.gz
```

Le fichier reçoit l'extension .gz.

Il conserve les mêmes droits et les mêmes dates de dernier accès et de modification.

6.4.2. Compresser avec bunzip2

Syntaxe de la commande bzip2.

```
bzip2 [options] [fichier ...]
```

Exemple :

```
[root]# bzip2 usr.cpio
[root]# ls
```

```
usr.cpio.bz2
```

Le nom du fichier reçoit l'extension .bz2.

La compression par "bzip2" est meilleure que celle par "gzip" mais dure plus longtemps.

6.4.3. Décompresser avec gunzip

Syntaxe de la commande gunzip.

```
gunzip [options] [fichier ...]
```

Exemple :

```
[root]# gunzip usr.tar.gz
[root]# ls
usr.tar
```

Le nom du fichier est tronqué par gunzip et se voit enlever l'extension .gz .

Gunzip décompresse également les fichiers portant les extensions suivantes :

- .z ;
- -z ;
- _z.

6.4.4. Décompresser avec bunzip2

Syntaxe de la commande bzip2.

```
bzip2 [options] [fichier ...]
```

Exemple :

```
[root]# bunzip2 usr.cpio.bz2
[root]# ls
usr.cpio
```

Le nom du fichier est tronqué par bunzip2 et se voit enlever l'extension .bz2 .

bunzip2 décompresse également le fichier portant les extensions suivantes :

- -bz ;
- .tbz2 ;
- tbz.

Glossaire

BASH
Bourne Again SHell

BIOS
Basic Input Output System

CIDR
Classless Inter-Domain Routing

Daemon
Disk And Execution MONitor

DHCP
Dynamic Host Control Protocol

DNS
Domain Name Service

FQDN
Fully Qualified Domain Name

LAN
Local Area Network

NTP
Network Time Protocol

nsswitch
Name Service Switch

POSIX
Portable Operating System Interface

POST
Power On Self Test

SHELL
En français "coquille". Á traduire par "interface système".

SMB
Server Message Block

SMTP
Simple Mail Tranfer Protocol

SSH
Secure SHell

TLS
Transport Layer Security, un protocole de cryptographie pour sécuriser les communications IP.

TTY
teletypewriter, qui se traduit téléscripteur. C'est la console physique.

UEFI
Unified Extensible Firmware Interface

Index

U

V

W