MEC 1310 : TI en Génie Mécanique

LE SYTÈME D'EXPLOITATION LINUX

RICARDO CAMARERO
Département de génie mécanique
École Polytechnique de Montréal

Janvier 2015

ÉCOLE
POLYTECHNIQUE
MONTRÉAL

Table des matières

Chapitre 1

Les systèmes d'exploitation

Un système d'exploitation est un environnement logiciel qui facilite l'utilisation des ressources d'un ordinateur. Il permet à un usager de réaliser les tâches courantes telles que l'accès, le traitement et la sauvegarde des données ; la communication (navigation sur l'internet, la lecture et rédaction de documents) ; l'exécution d'applications diverses ; la développement de programmes par le biais de compilateurs, etc.

Dans la réalisation des ces fonctions, il doit :

- cacher les complexités et les particularités du matériel en présentant à l'utilisateur une machine virtuelle simplifiée, définie par quelques concepts universels ;
- offrir un certain nombre de services élémentaires vis-à-vis le matériel ;
- gérer l'accès des usagers et la sécurité.

L'implantation des fonctionalités est faite spécifiquement pour chaque processeur pour en tirer le meilleur parti. L'architecture d'un système d'exploitation comprend trois couches :

Matériel : pilotes pour réaliser les échanges avec les périphériques et la gestion élémentaire de la mémoire vive ;

Information : système de fichiers (navigation, entrée/sorties) et la gestion des tâches et de la mémoire ;

Services : éditeurs, compilateurs, archivage....

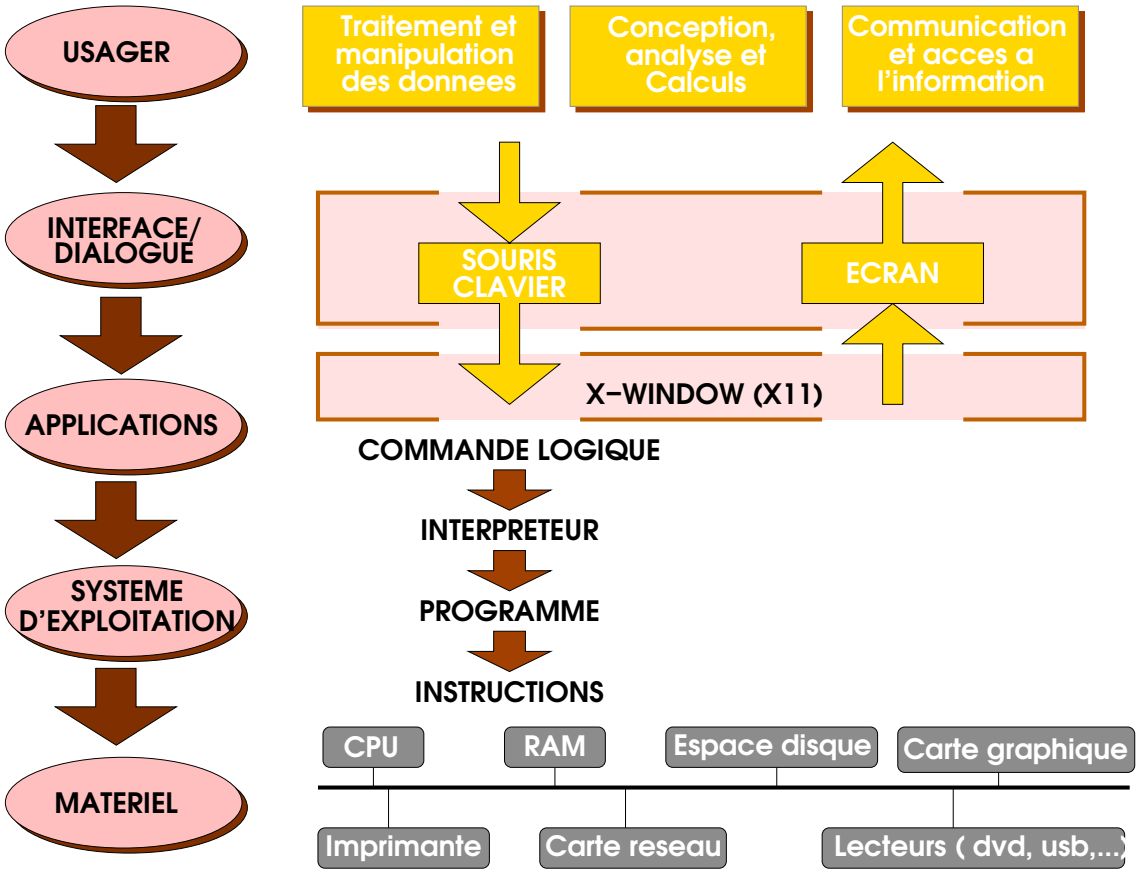

FIGURE 1.1 – Modèle conceptuel d'un système d'exploitation

Chapitre 2

Le sytème Unix

Unix est un système d'exploitation moderne bien adapté au travail technique et scientifique. Il offre un environnement performant permettant de réaliser les tâches courantes telles que la programmation dans divers languages informatiques ; l'utilisation d'applications scientifiques (Modélisation géométrique, calcul numérique....) et des applications bureautiques (base de données, tableurs, traitement de texte, graphisme...) ; la gestion et l'archivage de données et fichiers ; la production et la consultation de documents électroniques et audio-visuels (gravure de CD et DVD) ; la navigation sur l'Internet....

Unix/Linux est libre de tout droit d'auteur. Quiconque a le droit d'en faire des copies, de le modifier et de le distribuer.

2.1 Historique

Unix a été développé initialement comme système propriétaire dans les laboratoires Bell de la société AT&T. Comme son monopole dans le secteur de la téléphonie lui interdisait l'accès au marché de l'informatique, elle dû offrir un libre accès au code source de Unix. Conçu à l'origine comme système de fichiers, ses fonctionalités et qualités se sont affirmées au point où il est devenu rapidement un système de référence. Deux versions ont dominé le marché, la version de AT&T dite *System V* et celle de l'Université Berkeley, *BSD*. Depuis, des variantes sont apparues chez les constructeurs de matériel informatique, par exemple, AIX chez IBM, Solaris chez SUN, ULTRIX chez Digital Equipment, HP-UX chez Hewlett-Packard.....adaptées à leur processeurs. En parallèle avec ces versions propriétaires, il s'est développé des systèmes semblables sans aucun

droit de license et accessibles gratuitement, appelés logiciels de distribution libre.

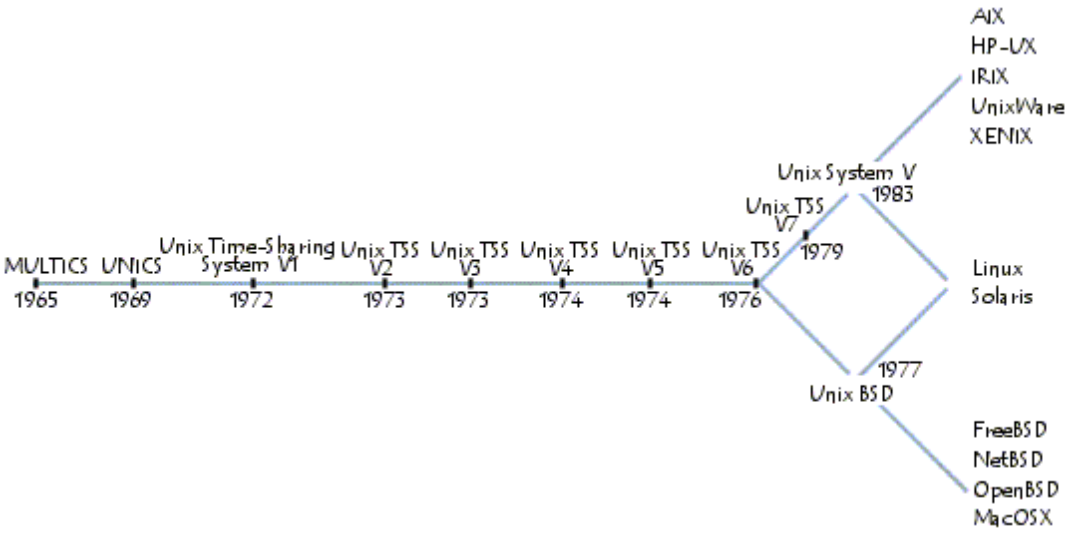

FIGURE 2.1 – Évolution du sytème UNIX

La différence avec les systèmes propriétaires est la libre disposition du code source. De plus, les auteurs doivent publier le code source avec les modifications qu'ils y ont apporté. Personne ne peut revendiquer la propriété du code source. Avec l'apparition des micro ordinateurs, la version LINUX a été développée par Linux Torwald dans cet esprit, avec des récentes distributions illustrées à la Fig. 2.2

Outre sa disponibilité, la souplesse et l'efficacité d'Unix a engendré de nombreuses variantes dans tous les domaines d'applications, notament en robotique, contrôle et surtout dans le domaine de la réseautique et des serveurs Internet. Trente années de développement ouvert lui confèrent une fiabilité et stabilité inégalées. Adapté et disponible sur la plupart des processeurs, Linux devient une norme de facto chez la plupart des constructeurs informatiques.

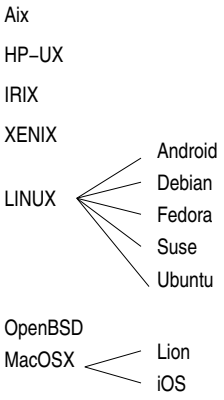

FIGURE 2.2 – Distributions du système Linux

2.2 l'architecture

L'architecture du système UNIX/LINUX est structurée en plusieurs couches. La première couche comprend les pilotes qui réalisent les échanges avec les divers périphériques ainsi que la gestion élémentaire de la mémoire. La deuxième couche fournit le système de fichiers et le système de gestion des tâches, incluant, respectivement, la gestion des entrée/sorties et de la mémoire. Finalement, la troisième couche est composée des différents outils de traitement. La première couche est proche du matériel et est réalisée en assembleur. Ensemble, les deux premières couches constituent le noyau qui est généralement écrit en assembleur et en C, et varie selon le matériel.

Unix est un système multi-tâches et multi-utilisateurs où à un instant donné plusieurs tâches s'exécutent logiquement en parallèle et où plusieurs utilisateurs peuvent travailler simultanement. Ceci est possible par une gestion de l'unité centrale selon le principe de temps partagé où les tâches reçoivent à tour de rôle une tranche de temps pour s'exécuter.

En plus de ce noyau de base, Linux comprend le système d'exploitation réseau qui est à l'origine du développement de TCP/IP, avec tous les programmes nécessaires pour s'interfacer avec l'infrastructure réseau.

Les sytèmes Unix/Linux s'installent indifférement en mode poste de travail individuel ou serveur.

2.3 Les interfaces

L'utilisateur accède aux ressources de l'ordinateur au travers d'un système de fenêtrage appellé x-window (ne pas confondre avec windows) qui offre une ligne de commande ou une interface graphique (GUI). Dans le premier cas, on utilise le clavier comme mode d'entrée en tapant la commande qui est envoyée à un interpréteur qui exécute un programme correspondant. Dans le mode interface graphique, l'usager saisi, par le biais d'un icône, la "commande" et la transmet également à l'interpréteur. Dans les deux cas, on utilise le langage de commande appellé le "shell" qui appelle des programmes sous formes d'instructions machines qui lancent les processus appropriés.

L'interpréteur de commandes est ainsi chargé de faire l'intermédiaire entre le système d'exploitation et l'utilisateur. Son rôle consiste ainsi à lire la ligne de commande, interpréter sa signification, exécuter la commande, puis retourner le résultat sur les sorties.

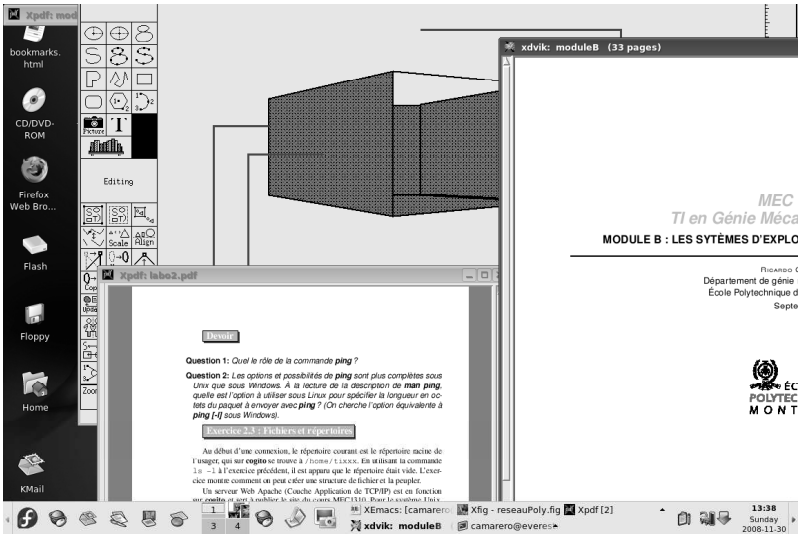

FIGURE 2.3 – Système de fenêtrage x-window sous Linux

Le shell est un fichier exécutable chargé d'interpréter les commandes, de les transmettre au système et de retourner le résultat. Il existe plusieurs shells, les plus courants étant *ssh*, *csh* (C Shell), *Tcsh* (Tenex C shell), *sh* (Bourne shell), etc. Chaque utilisateur possède un interpréteur de commande par défaut, qui sera lancé à l'ouverture d'une invite de commande. Il est toutefois possible de changer de shell dans une session en exécutant tout simplement le fichier exécutable correspondant. La plupart des distributions LINUX installent le shell *bash* (Bourne Again Shell) par défaut.

Lors de l'exécution d'une commande, un processus est créé. Par défaut, lorsque l'on exécute un programme, les données sont donc lues à partir du clavier et le programme envoie sa sortie et ses erreurs sur l'écran. Il est toutefois possible de lire les données à partir de n'importe quel périphérique d'entrée ou fichier, et d'envoyer la sortie sur un périphérique d'affichage, un fichier, etc.

Un système de documentation en ligne appellé les *pages de manuel* (en anglais *man pages*) décrit toutes les commandes disponibles. On tape sur la ligne de commande :

man *nom-de-la commande*

Pour plus d'information voir la Section 2.6.

2.4 Les processus

En se connectant à un ordinateur sous UNIX, un usager ouvre une *session* au cours de laquelle il émet des commandes ou exécute des programmes ou lance des applications. Ces actions engendrent des tâches appellées des *processus*. Un processus est une activité autonome qui exécute un programme qui fait appel au noyau. C'est est une entité dynamique qui a :

— un début (son lancement par le système d'exploitation),

— des actions consistant en une suite d'instructions, communiquant avec d'autres processus, et en créant d'autres,

— et une fin.

Ainsi, lors du démarrage d'une session, le système d'exploitation crée un premier processus qui consiste à exécuter l'interpréteur du langage de commande qui reçoit, interprète et exécute les commandes en provenance directement de l'usager, des programmes ou des applications lancés par l'usager. Ces commandes peuvent être lancées en mode "commande" dans une fenêtre ou en mode "graphique" par une interface appropriée.

Un processus comprend trois organes de communication avec son environnement (voir Figure 2.4) : les données sont lues dans le fichier *standard d'entrée*, les résultats sont écrits dans le fichier *standard de sortie*, et les erreurs sont acheminées vers le fichier *standard d'erreur*.

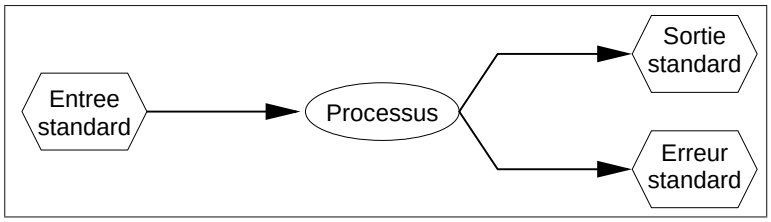

FIGURE 2.4 – Schéma d'un processus élémentaire

Par défaut, lors de la connexion, le fichier standard d'entrée est le clavier, les fichiers standard de sortie et d'erreur sont l'écran du terminal, tel qu'illustré à la Figure 2.5.

On peut rediriger les entrées-sorties d'un processus en connectant les fichiers standards vers d'autres périphériques ou bien d'autres fichiers sur disque. La Figure 2.6 illustre un processus qui prend ses entrées au clavier, et achemine ses sorties vers un fichier, et les messages d'erreur vers le terminal.

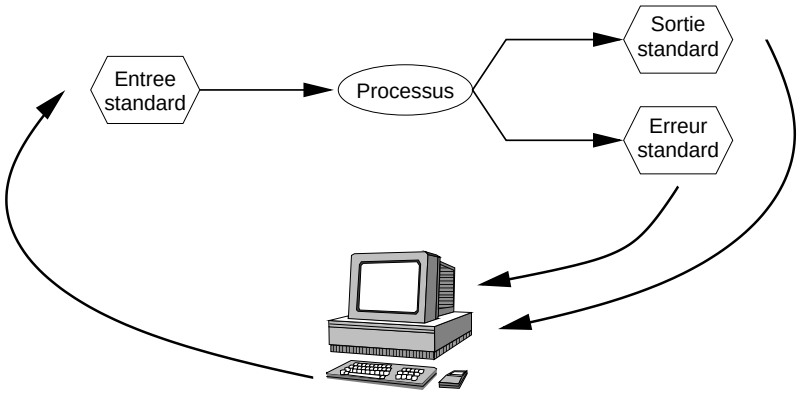

FIGURE 2.5 – Schéma d'un processus standard

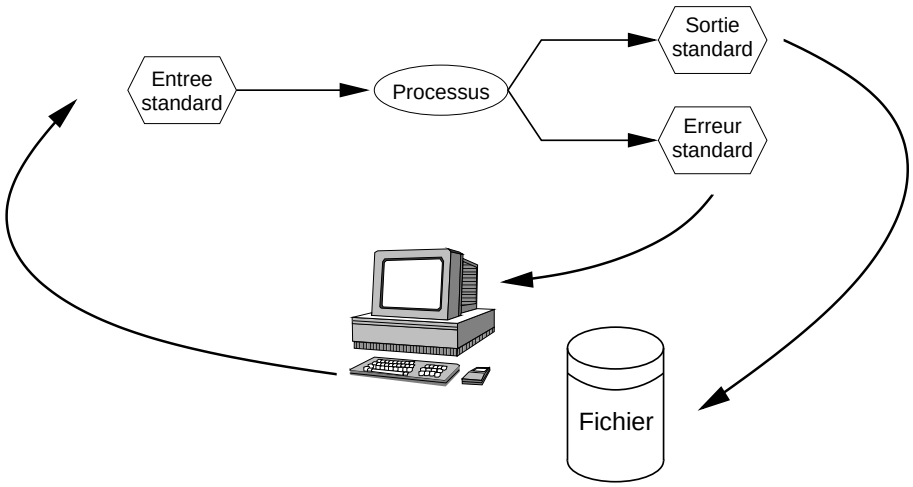

FIGURE 2.6 – Schéma d'un processus avec entrées-sorties redirigées

2.5 Le système de fichiers

Sous UNIX, le concept de lecture et d'écriture a été généralisé de sorte que écrire sur un fichier, un périphérique, sur une prise réseau ou bien l'entrée d'un processus se fait de façon identique. L'unification des entrées-sorties est réalisée par le système de fichiers au niveau de la deuxième couche (voir Section 2.2 qui traite toutes ces entités comme des fichiers, et les différences sont prises en compte seulement au niveau de la première couche du noyau.

Dans ce contexte, un *fichier* est une suite non structurée de caractères avec deux modes d'accès :

— *séquentiel*, pour accéder au caractère suivant,

— *direct*, pour accéder au caractère de numéro donné.

Les divers types de fichiers qui sont reconnus par le système sont :

— les *fichiers ordinaires*, que l'usager utilise pour stocker ses programmes ou données ;

— les *répertoires*, qui sont des listes de fichiers et servent à regrouper ces derniers pour les identifier de façon simple et dans la logique de l'usager ;

— les *liens (symboliques)*, qui sont des fichiers contenant des références à d'autres fichiers ;

— les *périphériques à accès par caractères*, comme un terminal ;

— les *périphériques à accès par blocs*, comme un disque ou autre support de stockage permanent ;

— les *prises réseaux (ou ports)*, qui permettent l'accès à distance à d'autres fichiers via un des nombreux protocoles TCP/IP ;

— les *tubes*, qui sont les organes de communications entre deux processus ;

Même si ces "fichiers" représentent des support physiques distincts, ils sont utilisés de façon transparente et unique par l'usager ou les programmes par l'entremise du système de fichiers.

Les caractéristiques d'un fichier sont regroupées dans un tableau appellé *i-noeud* qui contient les informations suivantes :

— le type de fichier ;

— sa taille, en octets ;

— l'identification du propriétaire ;

— les droits d'accès au fichier ;

— trois dates : la date de création, la date de la dernière modification et la date de la dernière consultation ;

— un compteur de référence sur le i-noeud ;

— la liste des blocs contenant l'information sur le disque.

La commande **ls** (voir Section 2.7) donne accès à ces diverses informations : avec la date et la taille, on peut gérer les différentes versions d'un document et synchroniser les copies résidant sur des ordinateurs distincts ; les permissions montrent les droits d'accès qui peuvent être éventuellement modifiés. Dans sa forme la plus simple, **ls** affiche le contenu du répertoire :

```
[tixxx@Montvallin amodule]$ ls
aExercise.pdf  aLecon.pdf  livre.aux  livre.pdf  notes.tex
aLecon.aux     aLecon.ps   livre.dvi  livre.ps
aLecon.dvi     aLecon.tex  livre.log  livre.tex
aLecon.log     figures     livre.out  livre.toc
```

Avec l'option *ls -i*, on obtient le numéro du *i-noeud* :

```
[tixxx@Montvallin amodule]$ ls -i
6177144 aExercise.pdf  6177125 aLecon.tex  6177123 livre.pdf
6177141 aLecon.aux     6177070 figures     6177068 livre.ps
6177142 aLecon.dvi     6177118 livre.aux   6177066 livre.tex
6177140 aLecon.log     6177065 livre.dvi   6177048 livre.toc
6177145 aLecon.pdf     6177052 livre.log   6177122 notes.tex
6177143 aLecon.ps      6177069 livre.out
```

L'option **ls -l** donne des informations encore plus complètes :

```
[tixxx@Montvallin amodule]$ ls -l
total 4612
drwxr-xr-x    2 tixxx    mec1310      4096 Jan 12  2007 aviation
-rw-r--r--    1 tixxx    mec1310       320 Nov  6  2006 Cours.sql
-rw-r--r--    1 tixxx    mec1310       532 Mar 22  2006 exemple1.php
-rwxrwxrwx    1 tixxx    mec1310      1009 Mar 22  2006 exemple2.html
-rw-r--r--    1 tixxx    mec1310      1203 Mar 22  2006 exemple2.php
-rwxr-xr-x    1 tixxx    mec1310      1110 Mar 22  2006 exemple3.html
-rw-r--r--    1 tixxx    mec1310       862 Mar 26  2006 exemple3.php
-rw-r--r--    1 tixxx    mec1310      2340 Mar 26  2006 exemple4.php
-rw-r--r--    1 tixxx    mec1310      1313 Mar 26  2006 exercice1.php
-rwxr-xr-x    1 tixxx    mec1310       735 Mar 26  2006 exercice4b.html
-rwxr-xr-x    1 tixxx    mec1310      2070 Mar 26  2006 formulaire4b.php
-rw-r--r--    1 tixxx    mec1310       389 Nov  6  2006 Horaires.sql
-rw-r--r--    1 tixxx    mec1310       662 Nov  6  2006 INS_Etudiants.sql
drwxr-xr-x    2 tixxx    mec1310      4096 Nov  1 14:28 laboSGBD
[tixxx@Montvallin amodule]$
```

La figure 2.7 illustre comment sur certains systèmes un code de couleurs permet d'identifier les divers types de fichiers.

FIGURE 2.7 – Caractéristiques des fichiers

Un fichier est identifié de façon unique, à l'externe (c-à-d par l'usager) par un nom, et à l'interne (c-à-d par le sytème) par son i-noeud. À l'aide d'un répertoire, le système établi la correspondance entre ces deux désignations : le nom symbolique (identificateur, nom donné par l'usager) d'un fichier et la localisation physique des enregistrements contenant le fichier (i-noeud). Le répertoire est essentiellement un tableau qui associe la désignation externe à la désignation interne, tel qu'illustré ci-dessous :

nom-du-fichier *(externe)*	⇒	**i-noeud** *(interne)*
fichier1	⇒	25
fichier2	⇒	29
...	⇒	...

Comme un répertoire est un fichier, il possède également une référence externe et une référence interne, et de plus, il peut contenir des fichiers qui sont eux-même des répertoires. Donc, pour désigner un fichier, on donne :

nom-du-répertoire/*nom-du-fichier*

Cela donne lieu à une arborescence dans laquelle les noeuds sont les répertoires et les fichiers sont les feuilles. Un fichier dans un tel environnement a deux repérage uniques : son numéro de i-noeud ou la liste des répertoires menant à son nom. On appelle cette liste *le chemin* (*path* en anglais). L'origine de cet arbre est la racine, designée par "/" (nom symbolique), et avec son i-noeud (interne) égal à 2. Ce fichier racine est le seul connu du système et ce mécanisme suffit à retrouver tous fichiers dans le système. Afin de permettre une navigation dans les deux sens, tout répertoire comprend par défaut deux entrées ou fichiers particuliers qui sont le répertoire parent, nommé "**..**" et le fichier (répertoire) courant, nommé "**.**", tel qu'illustré à la Figure 2.8.

La désignation interne du fichier *fich29* est le numéro i-noeud 25, tandis que la désignation absolue est *bin/local/dir49/fich29*. Ce modèle de représentation interne du système de fichiers ne sera pas utilisé en pratique, et sera remplacé par une représentation conceptuelle plus conviviale pour l'usager sur laquelle repose les outils de navigation d'usage courant tels que **ls** ou **cd**. La représentation de cette arborescence est illustrée à la Figure 2.9. Ce schéma montre la structure d'un système de fichiers où les répertoires partagés par tous les usagers sont sous le répertoire racine, " / ", et les répertoires personnels des différents usagers sont sous le répertoire *usagers* ou *home*. Un rectangle indique un répertoire tandis qu'un oval indique une feuille ou un fichier.

En plus des désignations interne et absolue, les langages de commande ou "shell" utilisent pour faciliter la navigation dans le système de fichiers, la désignation relative. D'abord on défini le concept de répertoire courant qui est le répertoire dans lequel travaille un usager à un moment donné d'une session. Les fichiers de ce répertoire sont identifiés simplement par leur nom (identificateur externe). Une désignation relative débute à partir du répertoire courant plutôt que de la racine, et ne commence pas par " / " mais plutôt par le nom d'un fichier du

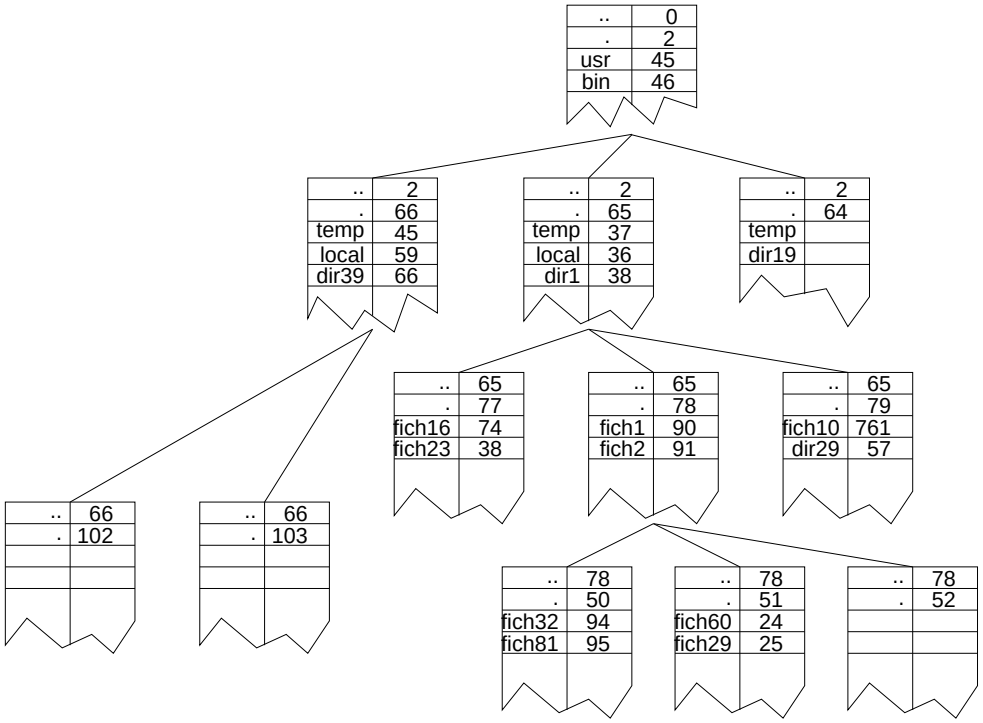

FIGURE 2.8 – Représentation interne de l'arborescence d'un système de fichiers

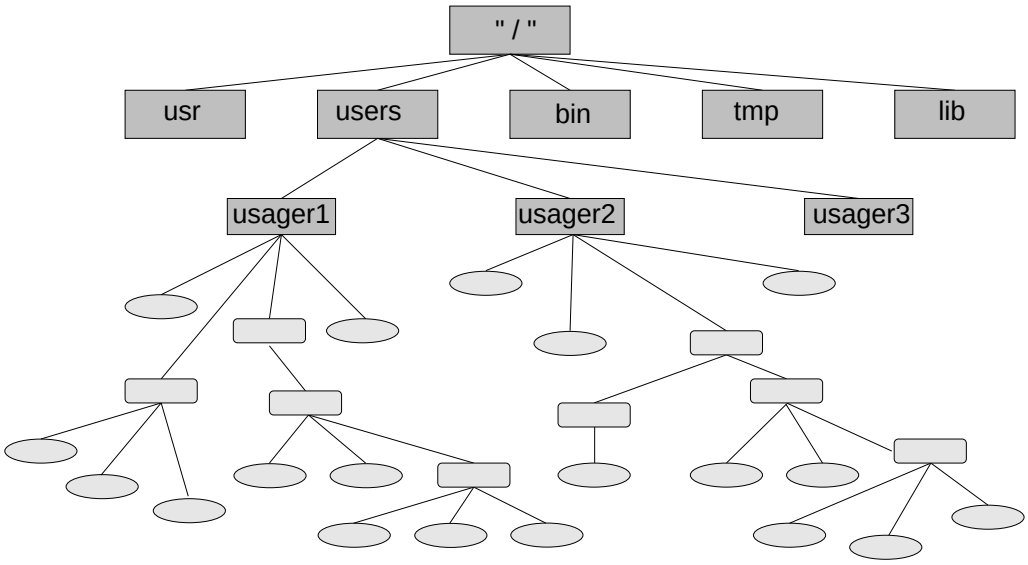

FIGURE 2.9 – Modèle conceptuel de l'arborescence d'un système de fichiers

répertoire courant. En particulier un nom relatif peut utiliser les fichiers "**.**" et "**..**" pour remonter ou descendre dans l'arborescence.

2.6 Le langage de commande

Après la phase de connexion, le "shell" ou l'interpréteur de commande se met en attente et l'indique par l'invite "**$**" [1] qui apparaît sur le terminal (ou fenêtre active). L'usager peut faire appel aux services du système en tapant des commandes au terminal qui seront "interprétées" par le "shell" et exécutées.

Une commande comprend son nom, des options et des arguments, séparés par un séparateur (généralement un espace) :

cmd [options] [arguments]

Les options sont facultatives et précédées par un tiret "**-**" et composées d'un seul caractère. On note que Linux fait la différence entre les majuscules et les minuscules.

On accède au système de documentation en ligne avec commande :

man nom-de-cmd

qui donne une description d'une commande donnée en argument. Comme le texte est généralement long, on utilise les flèches ***PgUp*** et ***PgDn*** pour faire défiler le texte, puis on tape la lettre***q*** pour sortir du mode **man**. Pour obtenir de l'information sur la commande **man** elle-même, on tape **man man** :

1. Ceci dépendra de l'ìnstalation

```
[tixxx@everest figures]$ man man
MAN(1)                      Commandes utilisateur                      MAN(1)

NOM
       man - formate et affiche les pages du manuel en ligne
       manpath - determine le chemin de recherche des pages pour lâĂŹutilisateur

SYNOPSIS
       man [-acdfhktwW] [-m  systeme]  [-p  chaine]  [-C  fichier_config]  [-M
       chemin] [-P visualiseur] [-S liste_sections] [section] nom ...

DESCRIPTION
       man formate  et  affiche les  pages  du  manuel  en ligne.  Cette  ver-
       sion reconnait les variables dâĂŹenvironnement MANPATH et (MAN)PAGER,  de
       sorte que vous pouvez avoir  vos propres jeux de manuels  personnels et
       employer le programme de votre choix pour les  visualiser.  Si  section
       est  specifiee,  man ne  recherchera  que  dans  cette section  partic-
       uliere.  Vous  pouvez egalement  indiquer  lâĂŹordre  de  recherche  des
       sections  desirees  ainsi  que  les  preprocesseurs a  utiliser pour
       traiter les  fichiers sources, par  des options de  la ligne  de  com-
       mandes  ou  des variables dâĂŹenvironnement. Si nom contient le caractere
       /, il sera dâĂŹabord considere en tant que nom de  fichier,  vous  pouvez
       donc faire : man ./toto.5 ou meme man /truc/machin/bidule.1.gz.

OPTIONS
.....................................................
.....................................................
.....................................................
```

Par exemple, la commande "*who*" affiche des informations sur les usagers connectés au système :

who affiche les usagers connectés ;

```
[tixxx@everest figures]$ who
tixxx :0           Aug 14 12:38
tixxx pts/1        Aug 14 12:38
tixxx pts/2        Aug 14 12:39
tixxx pts/3        Aug 14 12:39
tixxx pts/4        Aug 14 12:39
[tixxx@everest figures]$
```

who -u affiche les usagers connectés, avec des informations additionnelles ;

```
[tixxx@Montvallin Amodule]$ who -u
tixxx  :0           Aug 14 09:37   ?          3704
tixxx  pts/1        Aug 14 09:37 00:06        3815
tixxx  pts/2        Aug 14 09:37 00:05        3893
tixxx  pts/3        Aug 14 09:37   .          3899
tixxx  pts/4        Aug 14 09:37 00:05        3900
tixxx  pts/5        Aug 14 09:37 00:04        3919
[tixxx@Montvallin Amodule]$
```

who -H affiche les usagers connectés, avec des entêtes pour identifier les informations ;

```
[tixxx@Montvallin Amodule]$ who -H
NOM      LIGNE        HEURE        COMMENTAIRE
tixxx  :0           Aug 14 09:37
tixxx  pts/1        Aug 14 09:37
tixxx  pts/2        Aug 14 09:37
tixxx  pts/3        Aug 14 09:37
tixxx  pts/4        Aug 14 09:37
tixxx  pts/5        Aug 14 09:37
[tixxx@Montvallin Amodule]$
```

On peut combiner plusieurs options, tel qu'illustré ci-dessous :

```
[tixxx@Montvallin Amodule]$ who -Hu
NOM      LIGNE        HEURE        OSIF          PID COMMENTAIRE
tixxx  :0           Aug 14 09:37   ?          3704
tixxx  pts/1        Aug 14 09:37 00:07        3815
tixxx  pts/2        Aug 14 09:37 00:07        3893
tixxx  pts/3        Aug 14 09:37   .          3899
tixxx  pts/4        Aug 14 09:37 00:07        3900
tixxx  pts/5        Aug 14 09:37 00:06        3919
[tixxx@Montvallin Amodule]$
```

ou bien ajouter l'argument "who am i" :

```
[tixxx@Montvallin Amodule]$ who am i
tixxx  pts/3        Aug 14 09:37
[tixxx@Montvallin Amodule]$
```

Il est habituellement préférable, lorsqu'on obtient un nouveau compte, de changer le mot de passe tout de suite. La commande est *passwd* et demande l'ancien et le nouveau mot-de-passe :

```
[tixxx@Montvallin Amodule]$ passwd
Changing password for user tixxx.
Changing password for tixxx
(current) UNIX password:
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[tixxx@Montvallin Amodule]$
```

2.7 La navigation dans un sytème de fichiers

Le langage de commande ou "shell" dispose de plusieurs commandes qui permettent à l'usager de se déplacer dans l'arborescence d'un système de fichiers. La commande :

pwd (*print working directory*) affiche le nom absolu du répertoire courant. Utile pour s'y retrouver après plusieurs manipulations ;

```
[tixxx@Montvallin Amodule]$ pwd
/home/tixxx/TI/cours/Amodule
[tixxx@Montvallin Amodule]$
```

cd (change directory) déplace le répertoire courant. Sans paramètre, **cd** positionne le répertoire courant au répertoire personnel de l'usager, indiqué par " ~ " :

```
[tixxx@Montvallin Amodule]$ cd
[tixxx@Montvallin ~]$
```

Avec un nom absolu :

```
[tixxx@Montvallin ~]$ cd TI/cours/Amodule
[tixxx@Montvallin Amodule]$
```

ou relatif, le répertoire courant est positionné au répertoire désigné :

```
[tixxx@Montvallin Amodule]$ cd
[tixxx@Montvallin ~]$ cd TI/cours/Amodule
[tixxx@Montvallin Amodule]$ cd ../../../
[tixxx@Montvallin ~]$
```

mkdir (make directory) créé un répertoire dans le répertoire courant ;

rmdir (remove directory) efface un répertoire à condition qu'il soit vide ;

ls affiche le contenu du répertoire ;

La commande *ls* comprend plusieurs options :

ls -l pour chaque fichier, affiche toutes les informations suivantes :

```
[tixxx@Montvallin Amodule]$ ls -l
total 5868
drwxrwxr-x  2 tixxx mec1310    4096 aout 14 10:04 figures
-rw-rw-r--  1 tixxx mec1310    5361 aout 14 10:08 livre.aux
-rw-rw-r--  1 tixxx mec1310   94596 aout 14 10:08 livre.dvi
-rw-rw-r--  1 tixxx mec1310   24720 aout 14 10:08 livre.log
-rw-rw-r--  1 tixxx mec1310    1473 aout 14 10:08 livre.out
-rw-rw-r--  1 tixxx mec1310  455132 aout 13 14:44 livre.pdf
-rw-rw-r--  1 tixxx mec1310 5093835 aout 14 10:08 livre.ps
-rwxr-xr-x  1 tixxx mec1310  117278 aout 14 10:08 livre.tex
-rwxr-xr-x  1 tixxx mec1310  115404 aout 14  2006 livre.tex~
-rw-rw-r--  1 tixxx mec1310    1806 aout 14 10:08 livre.toc
```

Les informations affichées sont les suivantes :

- le type de fichier, indiqué par un tiret "-" pour un fichier ordinaire, un "d" pour un répertoire ou un "l" pour un lien symbolique ;
- les droits d'accès pour le propriétaire, le groupe, et les autres utilisateurs ; les droits sont indiqués par **r** pour la lecture, **w** pour l'écriture, **x** pour l'exécution ou un tiret "-" pour l'absence de droits ; (voir la Section 2.9
- le nom du propriétaire et du groupe ;
- la taille du fichier ;
- la date de la dernière modification ;
- le nom du fichier.

ls -t l'affichage du contenu est trié par la date de la dernière modification, plutôt que par ordre alphabétique ;

```
[tixxx@Montvallin Amodule]$ ls -lt
total 5868
-rwxr-xr-x  1 tixxx mec1310  115404 aout 14  2006 livre.tex~
-rw-rw-r--  1 tixxx mec1310 5093943 aout 14 10:09 livre.ps
-rw-rw-r--  1 tixxx mec1310   25092 aout 14 10:09 livre.log
-rw-rw-r--  1 tixxx mec1310   94620 aout 14 10:09 livre.dvi
-rw-rw-r--  1 tixxx mec1310    1473 aout 14 10:09 livre.out
-rw-rw-r--  1 tixxx mec1310    1806 aout 14 10:09 livre.toc
-rw-rw-r--  1 tixxx mec1310    5361 aout 14 10:09 livre.aux
-rwxr-xr-x  1 tixxx mec1310  117269 aout 14 10:09 livre.tex
drwxrwxr-x  2 tixxx mec1310    4096 aout 14 10:04 figures
-rw-rw-r--  1 tixxx mec1310  455132 aout 13 14:44 livre.pdf
[tixxx@Montvallin Amodule]$
```

ls -r le tri est dans l'ordre inverse :

```
[tixxx@Montvallin Amodule]$ ls -lr
total 5868
-rw-rw-r--  1 tixxx mec1310    1806 aout 14 10:19 livre.toc
-rwxr-xr-x  1 tixxx mec1310  115404 aout 14  2006 livre.tex~
-rwxr-xr-x  1 tixxx mec1310  117354 aout 14 10:19 livre.tex
-rw-rw-r--  1 tixxx mec1310 5096123 aout 14 10:19 livre.ps
-rw-rw-r--  1 tixxx mec1310  455132 aout 13 14:44 livre.pdf
-rw-rw-r--  1 tixxx mec1310    1473 aout 14 10:19 livre.out
-rw-rw-r--  1 tixxx mec1310   24878 aout 14 10:19 livre.log
-rw-rw-r--  1 tixxx mec1310   93060 aout 14 10:19 livre.dvi
-rw-rw-r--  1 tixxx mec1310    5361 aout 14 10:19 livre.aux
drwxrwxr-x  2 tixxx mec1310    4096 aout 14 10:04 figures
[tixxx@Montvallin Amodule]$
```

2.8 Manipulation des fichiers

Dans cette section, on présente les commandes qui servent à manipuler les fichiers, leur contenu et leurs attributs.

ls Nous avons vu cette commande dans la section précedente dans le cadre du contenu d'un répertoire. Utilisée avec le nom d'un fichier comme argument, cette commande donne toutes les information relative à un fichier :

ls -l nom-de-fichier

Par exemple,

```
tixxx@everest Amodule]$ ls -l livre.tex
-rwxr-xr-x  1 tixxx mec1310 111284 Aug 14 06:28 livre.tex
[tixxx@everest Amodule]$
```

cp (*copy*)] Permet de faire des copies d'un fichier. La syntaxe est la suivante :

cp source copie

Le système créé un nouveau fichier *copie* (s'il n'existe pas) avec le contenu de *source*. La source peut être un fichier dans le répertoire courant ou non. Si ce n'est pas le cas, il faudra alors spécifier le chemin pour s'y rendre.

Si l'argument *copie*, la destination, est un répertoire, alors un nouveau fichier est créé dans ce répertoire qui aura le même nom que dans le répertoire d'origine.

Il est possible d'utiliser l'astérisque (*) pour travailler avec plusieurs fichiers. Le tableau suivant illustre quelques possibilités.

* sélectionne tous les fichiers du répertoire ;

***.jpg** sélectionne tous les fichiers du répertoire se terminant avec l'extension **.jpg** ;

moduleA* sélectionne tous les fichiers du répertoire débutant par **moduleA** ;

Lecture sélectionne tous les fichiers du répertoire contenant la chaîne de caractères **Lecture**.

mv (*move*) permet de renomer ou déplacer un fichier :

mv ancien nouveau

Si les deux arguments sont des noms de fichiers, alors le résultat est un changement de nom. Si l'argument **nouveau** est le nom d'un répertoire, alors le fichier **ancien** est déplacé vers le répertoire **nouveau**.

rm (remove) efface un fichier ;

rm nom-du-fichier

***text** efface tous les fichiers se terminant par **text** (voir toutes variantes de l'utilisation de (*) ;

-r (récursif) entraine une suppression récursive de tous les fichiers si l'argument **nom-du-fichier** est un répertoire ;

-i (intéractif) Avec cette option, le système afiche le nom de chaque fichier avec un point d'interrogation. En répondant *yes*, le fichier est effectivement détruit. Tout autre caractère annule la commande. Option utile pour éviter des dégats !

-v les détails de l'action sont affichés à l'écran.

more Affiche le contenu d'un fichier sur la sortie standard (généralement l'écran) :

more nom-de-fichier

Cette commande comprend plusieurs options qui contrôlent la disposition du contenu à l'écran.(Voir le détail avec *man more*)

less semblable à *more* avec une meilleure fonctionnalité pour parcourir le contenu dans les deux sens.

2.9 Les droits d'accès

Linux est un système multi-usagers. Il permet aux usagers d'un système de partager des fichiers. Trois catégories d'usagers peuvent avoir des droits d'accès à un fichier : le propriétaire, les membres du groupe et les autres utilisateurs du système. Pour chacune de ces catégorie, on défini trois types de droits d'accès qui varient selon le type de fichier.

Pour un répertoire :

r (read) donne droit à consulter le contenu du répertoire, c-à-d la commande *ls* ;

w (write) donne droit à écrire dans un répertoire, c-à-d ajouter des fichiers, modifier les noms des fichiers et effacer des fichiers ;

x droit de traverser un répertoire (plus restrictif que *r* car on ne peut même pas voir le contenu du répertoire).

Pour un fichier ordinaire :

r (read) donne droit à lire le fichier ;

w (write) donne droit à modifier le contenu du fichier ;

x (exécuter) droit d'exécuter un fichier exécutable.

Pour contrôler l'accès à son système de fichier, un usager (le propriétaire) dispose de certains outils. Ces droits sont placés dans le i-noeud, donc les valeurs courantes sont accessibles par la commande *ls* :

ls -l nom-d-fichier

```
[tixxx@everest Amodule]$ ls -li livre.tex
5488779 -rwxr-xr-x  1 tixxx mec1310 113218 Aug 14 13:25 livre.tex
[tixxx@everest Amodule]$
```

Dans cet exemple, le propriétaire (tixxx) a les droits de lecture, écriture et d'exécution. Les membres du groupe (mec1310) et les autres usagers du système peuvent lire et exécuter, sans droits de modification.

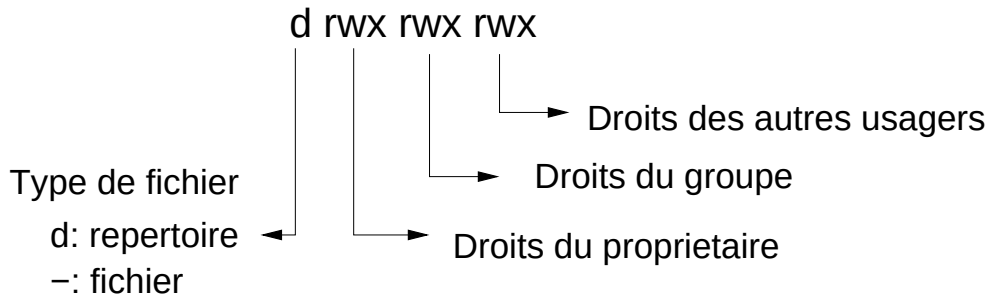

FIGURE 2.10 – Convention pour les droits d'accès

Les droits sont émis et modifiés par la commande *chmod* :

chmod mode fichier

Le paramètre *mode* peut prendre deux formes ; la forme absolue ou la forme symbolique. La première se base sur la représentation en trois chiffres octaux des paramètres des permissions. Cette approche est abstraite et on lui préfère la forme symbolique qui se présente de la façon suivante :

[qui]position[droit]

qui ce paramètre représente le type d'usager ; le propriétaire, le groupe, les autres ou tous, et prendra les valeurs **u**, **g**, **o** ou **a**, respectivement. Il indique à qui on donne les droits qui suivent dans la commande.

position est un caractère qui indique si l'on ajoute ou retire un droit. Il prend les valeurs suivantes : **+** pour donner un droit, **-** pour retirer un droit et **=** pour donner un droit et supprimer les autres permissions.

droit précise le droit : **r** pour lire, **w** pour écrire et **x** pour exécuter.

```
[tixxx@Montvallin Amodule]$ ls -l livre.tex
-rwxr-xr-x  1 tixxx mec1310 118664 aout 14 13:24 livre.tex

[tixxx@Montvallin Amodule]$ chmod g-r livre.tex

[tixxx@Montvallin Amodule]$ ls -l livre.tex
-rwx--xr-x  1 tixxx mec1310 118664 aout 14 13:24 livre.tex

[tixxx@Montvallin Amodule]$ chmod go-rwx livre.tex

[tixxx@Montvallin Amodule]$ ls -l livre.tex
-rwx------  1 tixxx mec1310 118664 aout 14 13:24 livre.tex
[tixxx@Montvallin Amodule]$
```

Dans cet exemple. on affiche d'abord les droits courants avec la commande **ls -l**. Ensuite, on retire au groupe le droit de lecture, **g-r**, et on vérifie le résultat. Finalement, on retire au groupe et autres usagers tous les droits, **go-rwx**.

2.10 Les répertoires

mkdir (make directory) créé un répertoire dans le répertoire courant ;

rmdir (remove directory) efface un répertoire à condition qu'il soit vide ;

2.11 Résumé des commandes

who affiche les usagers connectés.

passwd change le mot-de-passe.

pwd (*print working directory*) affiche le nom absolu du répertoire courant.

cd (*change directory*) déplace le répertoire courant.

mkdir (*make directory*) créé un répertoire dans le répertoire courant.

rmdir efface un répertoire à condition qu'il soit vide.

ls affiche le contenu du répertoire.

rm (*remove*) efface un fichier.

cp (*copy*) permet de faire des copies d'un fichier.

mv (*move*) permet de renommer un fichier.

more affiche le contenu d'un fichier sur la sortie standard (généralement l'écran).

less semblable à *more* avec une meilleure fonctionnalité.

Chapitre 3

Le monde Linux

3.1 Les distributions

UNIX

LINUX

Ubuntu

SuSe

Red Hat

Mandriva

Fedora

Debian

Mac OS

Mac OS X Snow Leopard

3.2 Les navigateurs

3.3 La bureautique

3.4 Audio-visuel

3.5 Graphisme

3.6 Le courrier électronique