Université de Corse
Faculté des Sciences et Techniques

Le système GNU/Linux

Notes de cours

F. Bernardi, 2003

bernardi@univ-corse.fr

http://spe.univ-corse.fr/bernardiweb/cours.htm

Introduction au système

La licence GPL

Principes et commandes de base

Périphériques de stockage et systèmes de fichiers

Processus de démarrage et connexion au système

Gestion des processus

Le noyau Linux

Table des matières

1 Introduction au système

1.1 Brève chronologie des systèmes Unix

1979 : AT&T annonce la commercialisation de son Unix, Unix System V. En réaction, l'Université de Berkerley développe sa propore version, BSD Unix qui est à la base de niombreux systèmes actuels.

1983 : mise en vente par AT&T de son Unix System V.

1987 : diffusion de X-Window développé au MIT et de Unix BSD 4.3

1990 : coopération AT&T et Sun Microsystems : version V.4. Diffusion du langage Perl.

1992 : sortie de Solaris, l'Unix de Sun dérivé du System V.

1.2 Le système GNU/Linux

L'initiateur du projet Linux est le Finlandais Linus Torvalds dont le but initial était de créer un Unix pour les processeurs i386 et qui a utilisé l'Usenet pour diffuser son travail. Par la suite, il a décidé de s'associer avec le projet GNU (GNU is Not Unix) ce qui a donné naissance au système GNU/Linux composé du noyau Linux et de la suite logicielle des outils GNU.

Les caractéristiques générales (vues plus en détail par la suite pour la plupart) du système sont les suivantes :

– code source disponible (Licence GPL, General Public License) ;
– distributions (packaging) multiples ;
– système multi-tâches et multi-utilisateurs ;
– multi-plateformes (Intel x86, SPARC,...) ;
– gestion du multi-processeur (option SMP, Symetric Multi-Processing) ;
– compatible POSIX ;
– gestion de consoles virtuelles ;
– support d'un grand nombre de systèmes de fichiers ;
– implémentation complète de la pile TCP/IP ;
– services réseaux : SLIP, PPP, NFS, SMB,... ;
– interface graphique : X-Window.

Un système GNU/Linux peut être vue selon trois couches complémentaires : une couche « physique » comprenant les périphériques et le BIOS, une couche « système » comprenant le noyau et les processus et une couche « interface » comprenant le shell et l'interface graphique basée sur X-Window. Les communications entre ces couches sont très réglementées par le noyau selon la Figure 1.

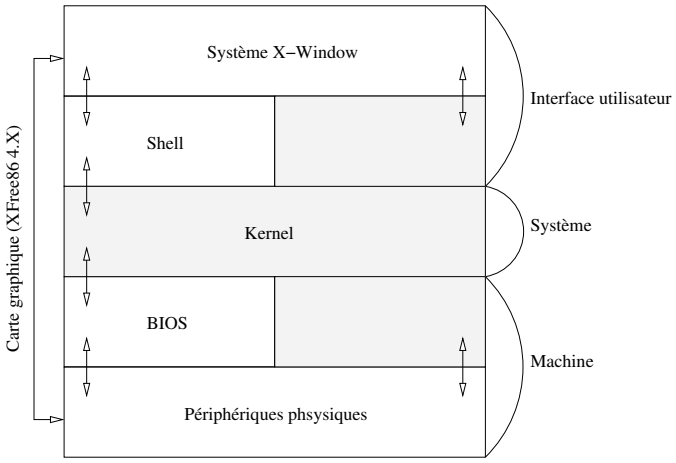

FIG. 1 – Architecture d'un système GNU/Linux

L'un des éléments essentiels à l'interaction avec les utilsateurs est le shell qui est un interpréteur de commandes qui lit et exécute les commandes de l'utilisateur. Il propose un contrôle des processus et permet de gérer les redirections en entrée et en sortie. Il propose en général un véritable langage de programmation et l'on peut en trouver en fait plusieurs types de shells disponibles : `bash`, `csh`, `ksh`, `tcsh`,...

Le système X-Window est l'interface graphique standard des systèmes Unix. Il repose sur un processus particulier appelé « serveur X » et utilise un gestionnaire de fenêtre dont un grand nombre sont actuellement disponibles (KDE, Gnome, WindowMaker,...). Une des principales caractéristiques de ce système est la possibilité pour l'utilisateur de déporter des affichages graphiques à travers un réseau.

2 La licence GPL et les distributions

Le noyau Linux est distribué sous la licence GPL (General Public License) developpée dans le cadre du projet GNU par la Free Software Foundation. Cette licence définit les modalités de la distribution du noyau et de ses modifications. « Free » est à prendre ici dans le sens « Libre » et non pas « Gratuit ».

À l'origine, il était possible de deistribuer et de modifier librement le système, mais pas d'en tirer profit par sa distribution ou son utilisation. Par contre, la GPL autorise à vendre ses programmes et à en tirer des bénéfices. Chacun est en droit de distribuer ces mêmes programmes sans aucune restriction.

Un logiciel libre sous licence GPL est différent d'un logiciel en domaine public : un logiciel en domaine public n'appartient à personne et donc à tout le monde, tandis qu'un programme sous GPL reste la propriété de son auteur (lois internationales sur la propriété). De la même manière, un programme sous GPL est différent d'un « shareware » dans le sens où l'auteur d'un shareware conserve la propriété du logiciel, mais sollicite les utilisateurs pour lui verser une contribution. Les logiciels sous GPL sont librement diffusables sans sollicitation de ce genre.

La GPL autorise les utilisateurs à modifier les programmes puis à distribuer leur propre version. Obligation est faite de distribuer sous licence GPL tout programme conçu de cette manière. Ceci implique par exemple, qu'il est interdit à toute société de récupérer Linux, de le modifier puis de le revendre sous une licence plus restrictive. Tout programme dérivé du code source de Linux doit être mis sous GPL.

Le distribution et l'utilisation d'un programme GPL sont gratuits, mais la GPL permet d'en tirer des bénéfices. Par contre, modifier les modalités de la licence lors de lavente d'un tel programme est interdit. L'acquisition d'un programme GPL, quel que soit le revendeur, donne le droit de le diffuser gratuitement ou par un prix choisi par la suite.

Une distribution est un « packaging » de logiciels sous GPL et souvent d'autres sous un autre type de licence. Elles proposent souvent une « hotline », un support technique à l'installation et à la maintenance, des manuels utilisateurs pré-imprimés et des interfaces de configuration personnelles. Le prix est lié aux frais de fabrication, mais ces distributions sont soumises aux restrictions volontaires de la GPL, à savoir préciser la licence et fournir le code source des logiciels sous GPL.

3 Principes et commandes de base

3.1 Arborescence du système

Le système possède une structure en arborescence dont la racine est notée « / ». Chacun des répertoires sous cette racine a un nom standardisé et un contenu précis (Table 1).

/	Racine du système, contient les répertoires principaux
/bin	Commandes essentielles communes à tous les utilisateurs
/boot	Fichiers de démarrage du système, contient le noyau
/dev	Points d'entrée des périphériques
/etc	Fichiers de configuration
/home	Répertoires personnels des utilisateurs
/root	Répertoire personnel de l'administrateur
/usr	Hiérarchie secondaire, applications, bibliothèques partagées
/var	Fichiers traces du système (logs)
/proc	Système de fichiers virtuel, informations en temps réel

TAB. 1 – Répertoires particuliers d'un système GNU/Linux

3.2 Commandes essentielles

Il existe un très grand nombre de commandes sous Unix. Un des principes fondateurs de l'approche Unix est d'ailleurs qu'une commande ne doit exécuter qu'une seule fonction, mais de manière exhaustive. Ces commandes sont exécutées depuis l'invite du shell ou un terminal dans le cas de X-Window et sont de la forme : `nom_commande [-options] <cible1> <cible2>`...

La Table 2 donne les commandes de base disponibles.

ls	Lister le contenu d'un répertoire
rm	Supprimer un fichier
cp	Copier un fichier
ln	Créer un lien sur un fichier
man	Afficher l'aide d'une commande
mv	Déplacer un fichier
cd	Changer de répertoire
mkdir	Créer un répertoire
pwd	Afficher le répertoire courant
cat	Afficher le contenu d'un fichier texte
file	Afficher le type d'un fichier
find	Rechercher un fichier dans l'arborescence

TAB. 2 – Commandes de base d'un système GNU/Linux

3.3 Gestion des utilisateurs

Sous Linux, un utilisateur est une personne physique ou virtuelle possédant des droits d'accès au système, un groupe d'utilisateur et éventuellement un répertoire personnel. L'utilisateur root est l'adminstrateur du système. Il possède tous les droits sur le système, les fichiers et les utilisateurs.

L'identification d'un utilisateur s'effectue par l'intermédiaire de son nom (« login ») et de son mot de passe. Tous les utilisateurs sont référencés dans les fichiers `/etc/passwd` ou `/etc/shadow`, tandis que `/etc/group` contient les références aux différents groupes. Le répertoire personnel d'un utilisateur est en général `/home/<login>`, et il peut changer son mot de passe par la commande passwd.

Les commandes `adduser`, `deluser` et `usermod` sont utilisables uniquement par root et permettent respectivement d'ajouter, de supprimer et de modifier un compte utilisateur.

Le fichier `/etc/passwd` contient toutes les informations sur les comptes utilisateurs du système (Figure 2). Seul root a le droit d'écriture dessus et chacun des utilisateurs est référencé par un ligne donnant :

– son login ;
– son mot de passe crypté ;
– son numéro d'identification sur le système (UID),
– son numéro de groupe initial (GID) ;
– un commentaire (son nom complet en général) ;
– son répertoire personnel de base ;
– son shell par défaut.

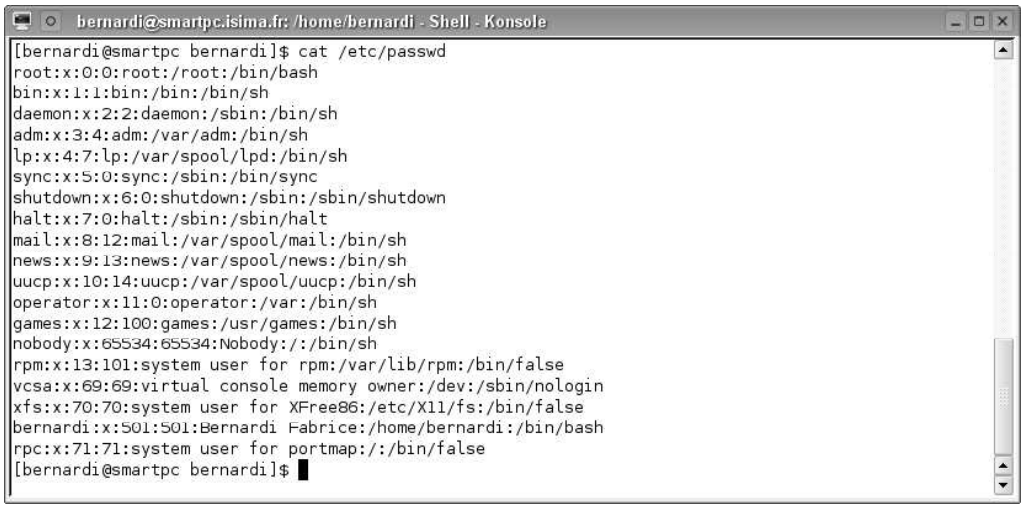

FIG. 2 – Le fichier `/etc/passwd`

3.4 Droits d'accès sur les fichiers

Il existe trois types de permissions pour un utilisateur sur un fichier : la lecture (Read), l'écriture (Write) et l'exécution (eXecute). À la création d'un nouveau fichier, des droits par défaut sont utilisés. Par ailleurs, trois catégories d'utilisateurs sont définies pour un même fichier : le propriétaire (User), les membres du groupe (Group) et les autres utilisateurs (Other). Chacun des fichiers est associé à un propriétaire et à un groupe.

La commande pour afficher les droits des fichiers est `ls -l` (Figure 3).

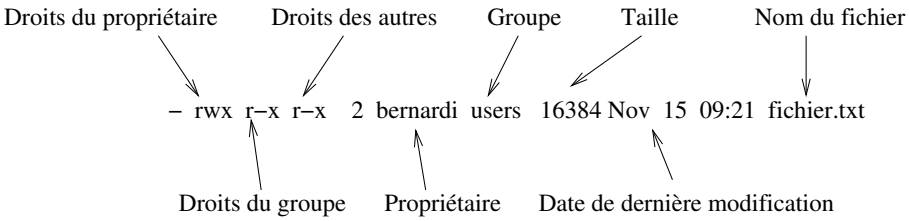

FIG. 3 – Droits d'accès affichés pas la commande ls -l

Les commandes pour la gestion des accès sont chown pour changer le propriétaire, chgrp pour changer le groupe et chown pour les différents droits :

– `chmod + fichier.txt` : positionne l'accès en lecture ;
– `chmod -r fichier.txt` : enlève l'accès en lecture ;
– `chmod o-x fichier.txt` : enlève l'accès en exécution pour les autres ;
– `chmod go+w fichier.txt` : ajoute le droit en écriture pour le groupe et les autres.

4 Périphériques de stockage et systèmes de fichiers

4.1 Rappels sur les disques durs

Un disque dur est composé de plateaux reliés à un moteur central et l'on trouve des têtes de lecture de part et d'autre de chacun des plateaux. Sur chaque plateau, on trouve des pistes cylindriques découpées en secteurs et l'adressage d'un secteur est une référence au cylindre, à la tête de lecture utilisée, à la piste, et enfin au secteur.

Il existe un ensemble de secteurs particuliers appelé le MBR, très petit en général et partagé en deux parties :

– la table des partitions qui est une zone d'informations sur l'emplacement, la taille et les propriétés des partitions ;
– le système d'amorçage chargé de lancer le système d'exploitation.

Le secteur d'amorçage d'une partition est installé sur le premier secteur d'une partition et est executé par le programme d'amorçage du MBR. Un tel secteur est partagé en deux parties :

– des données de propriétés ;
– un programme d'amorçage.

4.2 Partitions

À l'installation, un disque dur n'est ni partitionné, ni formaté. Partitionner signifie définir des espaces réservés sur le disque, et formater signifie préparer la partition à recevoir des informations.

Une partition est définie par son type, son emplacement de début de partition et sa taille (ou son emplacement de fin de partition). Un partitionnement est réversible (non physique). Une seule partition peut être activée sur un ordinateur : l'activation indique où le BIOS doit aller chercher le noyau du système d'exploitation pour le démarrage.

Il existe trois sorte de partitions :

– partition principale : au maximum quatre, pas de minimum, accepte tout type de système de fichiers
– partition étendue : ne peut contenir que des partitions logiques, ne peut pas recevoir de système de fichiers, ne peut exister que s'il existe une partition principale
– partition logique : contenue dans une partition étendue, pas de limitation en nombre, accepte tout type de système de fichiers.

Les partitions permettent la possibilité d'installer plusieurs systèmes d'exploitation sur une même machine, la sauvegarde des donnes en cas de réinstallation ou de plantage ainsi qu'une séparation logique des répertoires (système, utilisateur, fichiers journaux,...).

4.3 Périphériques de stockage et montage

On accède aux partitions par des pointeurs stockés dans `/dev`. Ces pointeurs sont de la forme « `/dev/PPLN` » avec « PP » le type de bus (« `hd` » pour les disques IDE, « `sc` » pour les disques SCSI, « `fd` » pour les disquettes), « L » la lettre du périphérique concerné et « N » le numéro de partition concernée (1 à 4 pour une partition principale ou étendue, > 4 pour les partitions logiques).

Exemples :

– `/dev/hda` : périphérique maître du bus primaire IDE ;
– `/dev/hdd` : périphérique esclave du bus secondaire IDE ;
– `/dev/hda1` : première partition primaire du périphérique maître du bus primaire IDE ;
– `/dev/hda7` : troisième partition logique du périphérique maître du bus primaire IDE.

Monter un système de fichiers signifie l'associer à un répertoire sur le système. Par exemple, le système de fichiers racine est monté sur « / ». Un système de fichiers n'est pas accessible tant qu'il n'est pas monté.

La commande pour monter un système de fichiers est la commande mount :

`mount -t <système_de_fichiers> <périphérique> <répertoire>`

Exemple : `mount -t reiserfs /dev/hda3 /home`

Les points de montage usuels du système sont décrits dans le fichier `/etc/fstab` (Figure 4).

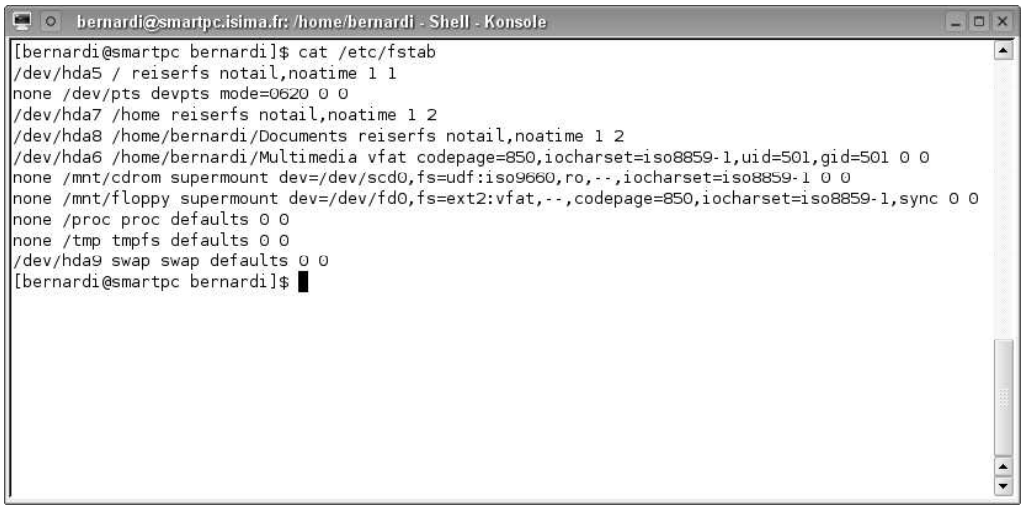

FIG. 4 – Le fichier `/etc/fstab`

4.4 Systèmes de fichiers « classiques »

L'arborescence d'un système est différente du système de fichiers de ce même système.

On appelle arborescence la structure logique des fichiers et répertoires, telle qu'elle apparaît à l'utilisateur final. On appelle système de fichiers l'agencement logique et structuré des données sur un média.

Linux supporte un grand nombre de systèmes de fichiers, et notamment ceux provenant du monde Windows (FAT16, FAT32, NTFS en lecture seule actuellement).

4.4.1 Le système de fichiers de « Swap »

Ce type de système de fichiers est un peu particulier : il est uniquement dédié à l'utilisation de la mémoire virtuelle et sert de mémoire complémentaire quand la RAM est saturée. Le Swap est utilisé intensivement par le système pour la décharge des logiciels en mémoire.

4.4.2 Le système de fichiers FAT (File Allocation Table)

C'est l'un des plus simples que l'on puisse trouver. Il repose sur une table d'allocation des fichiers stockée au niveau supérieur de la partition. Il possède quatre limitations principales :

- le répertoire doit racine doit être stocké à un emplacement fixe sur la partition ;
- la mise à jour de la table d'allocation est très laborieuse pour le système ;
- il n'existe aucune organisation de la structure des répertoires, d'où une fragmentation importante ;
- la taille des clusters est variable en fonction de la taille des partitions

4.4.3 Le système de fichiers NTFS (New Technology File System)

Dans NTFS, l'organisation des fichiers est optimisée sur la partition, d'où une très faible fragmentation. NTFS respecte la norme POSIX.1 et permet une bonne gestion des disques de grande taille. Il prend également en charge le modèle de sécurité des systèmes Windows NT, 2000 et XP.

4.4.4 Le système de fichiers ext2fs

C'est le système de fichiers par défaut de GNU/Linux. Il définit quatre catégories de fichiers :

– les fichiers normaux (textes, exécutables) ;
– les fichiers répertoires ;
– les fichiers spéciaux, contenus dans le répertoire /dev, qui sont des points d'accès aux périphériques ;
– les fichiers liens symboliques qui font référence à d'autres fichiers.

Ses principales caractéristiques sont les suivantes :

– répertoires séparés par des / ;
– sensible à la casse (aaa <> AAA) ;
– fichiers cachés débutant par un « . ».
– espaces et noms longs acceptés ;
– défragementation quasi-inutile.

Une partition ext2fs comporte un secteur d'amorçage et un ensemble de groupes de secteurs de même taille. Le secteur d'amorçage est le premier sur la partition, est noté secteur 0 et a une taille de 1 Ko. Chacun des groupes de secteurs comprend six parties :

– un super bloc, contenant les informations de structure de la partition ;
– la liste des descripteurs de groupe permettant de localiser sur le disque les informations de chaque groupe ;
– la table d'allocation des blocs du groupe ;
– la table d'allocation des inodes ;
– la table d'allocation des inodes du groupe ;
– les blocs de données.

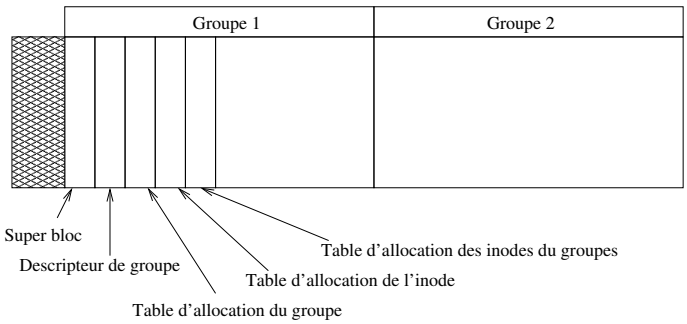

FIG. 5 – Les groupes d'ext2fs

Le super bloc et les descripteurs de groupes sont répétés au début de chacun de chacun des groupes. Ces groupes ont pour but de regrouper les informations reliées entre elles et ont une taille de 8192 secteurs.

Pour chacun des fichiers créés sur un disque, un inode est associé. Il contient :

– l'identification du propriétaire ;
– la taille en octets ;
– les heures de modification, création ou accès ;
– le nombre de liens associés ;
– le type ;
– les droits d'accès.

4.5 Sytèmes de fichiers journalisés

Pour accélerer les opérations d'E/S, le noyau Linux stocke temporairement dans la mémoire les données à inscrire sur le disque. Un problème peut survenir lors de l'interruption brutale du système et il pourrait alors exister une incohérence du système tout entier avec, par exemple, un nouveau fichier qui n'avait pas encore été effacé sur le disque dur, mais dont les inodes et le bloc de données figurent encore sur le disque dur.

Un système de fichiers journalisé est un système qui réagit aux erreurs. L'intégrité des données est assurée au moyen de mises à jour de fichiers journaux qui sont écrits avant que les blocs disques soient eux-mêmes mis à jour. Lors d'une défaillance du système, un fichier exhaustif relatif au système de fichiers, assure que le système est restauré.

4.5.1 Les B-Trees

L'idée de base pour améliorer les performances d'un système de fichiers sous Unix est d'éviter l'utilisation de listes chaînées et d'utiliser des arbres équilibrés B-Trees (Balanced Trees).

Une structure dite B+Trees est utilisée de manière traditionnelle en bases de données. Elle permet de classer toutes les clés aux feuilles de l'arbre. Les feuilles doivent être reliées entre elles. Les noeuds et feuilles peuvent être de tailles différentes. Il n'y a jamais besoin de modifier le père lorsqu'une clé est effacée.

4.5.2 Le système de fichiers ReiserFS

ReiserFS est un système de fichiers libre aui implémente les B+Trees. Ces objets sont les structures utilisées pour conserver l'information relative aux fichiers (permissions, etc...).

ReiserFS définit les noeuds non-formatés qui sont des blocs logiques sans formats définis et utilisés pour stocker des données sur les fichiers. Il définit également les Direct Items qui sont les données du fichier lui-même. Ces Items sont de taille variable et stockés dans les feuilles de l'arbre et quelque fois même avec d'autres si la place est suffisante. ReiserFS utilise la technique du « Tail Packing » pour ranger les queues de secteurs dans le minimum d'espace possible. Les Directs Items sont d'ailleurs prévus pour conserver les données de petits fichiers et même les queues de fichiers.

ReiserFS utilise par défaut une taille de bloc fixe de 4 Ko et donc pénalise les opérations d'E/S sur les gros fichiers.

4.5.3 Le système de fichiers ext3fs

Ext3fs est compatible avec ext2fs. C'est en fait un système ext2 avec un journal. Il n'est donc qu'un système de fichiers journalisé partiel et souffre donc des limitations d'E/S.

Avantages principaux d'ext3fs :

– Journalisation et maintient de la cohérence des données non seulement pour les descripteurs de fichiers mais aussi pour les données elles-même ;
– les partitions ext3 ont la même structure que ext2, ce qui rend facile la sauvegarde et le sauvetage de l'ancien système.

Ext3 réserve un des inodes spéciaux d'ext2 pour stocker le journal, et il est possible d'avoir plusieurs systèmes de fichiers partageant le même journal.

5 Processus de démarrage et connexion au système

5.1 Démarrage du système

Quand l'ordinateur est démarré, le premier élément a être lancé est le BIOS qui va choisir un disque de démarrage (disque dur ou disquette) et va lire le tout premier secteur.

Dans le cas d'un disque dur, ce secteur est le Master Boot Record (MBR). Ce secteur de démarrage contient un petit programme qui tient sur un seul secteur et dont la responsabilité est de lancer le système d'exploitation. Pour un disque dur, le code contenu dans le secteur de démarrage va examiner la table des partitions, identifier la partition active, lire le secteur de démarrage depuis cette partition, et l'exécuter.

Il existe plusieurs systèmes de démarrage pour les systèmes Unix. Les plus connus pour Linux sont LILO (LInux LOader) et GRUB (GRand Unified Bootloader). Ces systèmes sont chargés de démarrer le noyau et d'initialiser le système en suivant les étapes suivantes :

- le noyau Linux est décompressé ;
- le noyau Linux initialise les périphériques (disque dur, carte réseau,...) et configure automatiquement les pilotes ;
- le noyau monte le système de fichiers racine qui est monté en lecture seule de manière à pouvoir vérifier les erreurs qu'il pourrait contenir ;
- le noyau lance le programme init qui permet de lancer les processus de base. Init va lire le fichier `/etc/inittab`, lancer le script `/etc/rc.d/rc.sysinit`, lancer les processus contenus dans `/etc/rc.d/rc<n>` et finalement lancer le script `/etc/rc.d/rc.local` ;
- init passe alors en mode multiutilisateurs et lance un getty pour chaque console virtuelle, getty étant un programme permettant le login au système ;
- finalement, le système de fichiers racine est remonté en lecture-écriture et les processus des services sont lancés

5.2 Arrêt du système

Il est très important de suivre une procédure d'arrêt coorecte pour les systèmes Linux. En cas de procédure incorrecte, le système de fichiers sera probablement inutilisable au prochain reboot. Ce risque provient essentiellement du fait que Linux possède un cache disque qui n'écrit pas les données toutes à la fois, mais seulement par intervalles.

La commande pour arrêter le système est shutdown. Il est possible de spécifier une date d'arrêt du système :

- `shutdown -h now` arrête le système immédiatement ;
- `shutdown -h 10` arrête le système dans 10 minutes ;
- `shutdown -h 18 :30` arrête le système à 18h30.

La commande shutdown permet également de redémarrer le système en utilisant l'option -r au lieu et place du -h.

Une fois que cette a été lancée, tous les systèmes de fichiers sauf la racine sont démontés, les processus utilisateurs sont tués, les services sont arrêtés et le système stoppé.

5.3 Le processus `init`

Quand le noyau est chargé, il lance le processus `init`. Ce processus est le père de tous les processus ultérieurs et possède PID 1.

Quand il démarre, init va lire le fichier /etc/inittab qui contient des instructions sous la forme suivante :

```
id : runlevels :action :processus
```

- `id` : identifie la ligne dans le fichier ;
- `runlevels` : représente les niveaux de démarrage auxquels se rapportent la ligne considérée ;
- `action` : représente la manière de lancer le processus. Ses valeurs principales sont « respawn » (pour relancer automatiquement le processus une fois celui-ci terminé), « wait » (pour qu'init attende la fin du processus avant de continuer), « boot » (qui définit que le processus doit être lancé au moment du démarrage du système) et « ctrlaltdel » (qui spécifie quel processus est lancé si la séquence de touches Ctrl-Alt-Del est pressée).

Par exemple : `1 :2345 :respawn :/sbin/getty 9600 tty1` permet de lancer un processus de connexion sur la première console virtuelle.

5.4 Runlevels (ou niveaux de démarrage)

Un runlevel est l'état dans lequel se trouve init et le système entier. Ils définissent quels sont les services qui tournent et sont identifiés par des chiffres. Il n'existe pas de standard de numérotation, mais en général, le niveau 0 permet d'arrêter la machine, le 1 permet de démarrer en mode mono-utilisateur, et les niveaux de 2 à 6 sont laissés à la disposition de l'utilisateur.

Quand `init` démarre, il recherche dans `/etc/inittab` une ligne dont l'action est initdefault qui permet de spécifier le runlevel de démarrage. Il est cependant possible de démarrer selon un runlevel particulier en le passant par exemple comme argument à LILO, et de changer de runlevel en cours d'utilisation en utilisant l'une des deux commandes `init <n>` ou `telinit <n>`.

5.5 Login au système

`Init` lance tout d'abord des processus `getty` qui écoutent les terminaux de connexion et attendent l'interaction d'un utilisateur. Quand un utilisateur est détecté, `getty` lance le programme `login` qui permet de vérifier le mot de passe. Si ce mot de passe est correct, `login` lance alors le shell par défaut configuré par l'utilisateur. Dès que l'utilisateur de déconnecte, `init` relance un nouveau `getty`.

Le programme `login` permet l'identification de l'utilisateur et tient à jour un listing des personnes connectées dans le fichier `/var/run/utmp`. La base de donnée des utilisateurs est contenu dans le fichier `/etc/passwd` qui spécifie par exemple le shell de login de chacun des utilisateurs.

6 Gestion des processus

6.1 Notion de processus

On appelle processus l'exécution d'un programme à un instant donné. Dans le cas des systèmes Unix, un processus constitue un objet sur le système et est rangé sur le disque sous la forme d'un fichier. L'ensemble de ces fichiers fournissent l'image de l'état d'avancement d'un programme à un moment donné.

A chacun des processus est associé un bloc de contrôle constitué de la suite des intstructions du programme et d'un contexte d'exécution (registres, piles, système d'E/S, liens avec les utilisateurs). L'ordonnanceur (scheduler) est le module du noyau chargé d'allouer le temps processeur à chacun des processus. Sa tâche principale est de gérer les conflits et d'assurer à chacun des processus sa part de temps processeur. Il existe pour cela plusieurs stratégies possibles (partage de temps, round robin,...).

Sous Unix System V, il existe deux classes d'ordonnancement de processus :

– la classe des processus en partage de temps (classe TS, Time Share), dans laquelle la priorité est dynamique ;
– la classe des processus en temps réel (classe RT, Real Time), dans laquelle la priorité est donnée au temps de réponse.

La distinction entre ces deux classes s'effectue par le mécanisme de partage du temps processeur.

Il existe également deux types de processus :

– les processus systèmes appelés les démons (daemons). Ils ne sont sous le contrôle d'aucun terminal et leur rôle est d'assurer un certain nombre de services généraux accessibles à tous les utilisateurs. Leur propriétaire est root en général, et leur répertoire de travail est la racine /, de manière à permettre le démontage des systèmes de fichiers sans avoir à les interrompre. Les démons sont en général lancés au démarrage du système.
– les processus utilisateurs. Ils sont dédiés à l'exécution d'un tâche particulière et ont une durée de vie limitée. Ils sont par ailleurs généralement liés à un terminal.

Les processus peuvent s'exécuter en modes user (le processus ne peut lancer que des programmes utilisateurs) ou kernel (le processus ne peut exécuter que des instructions du noyau). Un processus n'utilisant plus aucune ressource passe dans l'état zombie. Cet état permet deux choses essentielles sui sont la prise en compte de la terminaison d'un processus par son père, et la récupération par son père d'informations sur la manière dont celui-ci s'est terminé. Cette information prend la forme d'un code de retour (exit status) qui est renvoyé par tout processus qui se termine.

6.2 Création des processus

La gestion des processus sous Unix permet la création d'un nouveau processus de deux manières différentes :

– l'appel système fork permet de créer un nnouveau processus à l'intérieur d'un même programme ;
– l'appel système execve permet de lancer un nouveau programme.

Sous Linux, les propriétés d'un processus se divisent en trois groupes, le PID, l'environnement et le contexte.

Le PID (Process IDentifier) est un identificateur unique du processus. Il est utilisé pour référencer le processus au niveau du noyau.

L'environnement du processus est hérité depuis ses parents et est composé de deux vecteurs, le vecteur des arguments donnant la ligne de commande utilisée pour lancer le processus et le vecteur d'environnement donnant les valeurs des variables d'environnement associées.

Le contexte d'un processus donne l'état d'un processus s'exécutant à n'importe quel instant. Le scheduler du noyau utilise le scheduling context du processus pour arrêter ou redémarrer le processus.

6.3 Le système de fichiers /proc

Le système de fichiers `/proc` est monté sur la racine mais n'existe pas physiquement sur le disque. C'est en fait un pseudo-système de fichiers, tenu à jour par le noyau et définissant un sous-répertoire pour chacun des processus.

Il comporte également quatre fichiers importants :

– `cpuinfo` qui contient les informations sur le processeur ;
– `filesystems` qui contient les systèmes de fichiers supportés par le système ;
– `meminfo` qui contient des informations sur la mémoire du système ;
– `kcore` qui est l'image de la mémoire physique du système.

Les informations sur un processus particulier se trouvent dans un sous-répertoire de `/proc` portant pour nom le PID de ce processus. Dans chacun de ces répertoires, on trouve les fichiers suivants :

– `cmdline` : la ligne de commande par laquelle le processus a été lancé ;
– `cwm` : le répertoire courant du processus ;
– `environ` : l'environnement du processus ;
– `fd` : liens vers les fichiers ouverts par le processus ;
– `maps` : plages mémoire allouées par le processus.

7 Le noyau Linux

Le noyau (kernel) Linux est l'élément essentiel du système. Il permet le contrôle à bas niveau du matériel ainsi que cinq fonctions essentielles :

– le démarrage du système ;
– la gestion des processus : le noyau permet le contrôle de tous les processus du système et s'assure que chacun d'entre eux dispose d'une partie raisonnable de mémoire et de cycles processeurs ;

– la communication inter-processus (IPC, Inter Process Communication) : le noyau contrôle la communication entre les processus en utilisant un système de mémoire partagée, de tubes (pipes) et de tubes nommés. Toute cette gestion est conforme au standard POSIX.
– l'interaction avec le matériel : le noyau fournit aux processus un accès aux périphériques matériels. Il dispose d'instructions générales pour des périphériques standards (disque dur par exemple) et permet également l'utilisatio de pilotes spécifiques.
– le système de fichiers virtuel (VFS, Virtual FileSystem) : Linux dispose d'une interface spéciale du noyau appelée VFS qui lui permet de supporter plusieurs systèmes de fichiers. Cette interface permet de traiter les systèmes de fichier de manière unifiée, et apparaît donc comme un pilote de périphérique, mais pour les systèmes de fichiers.

7.1 Fonctionnement du noyau

Le noyau Linux est dit monolithique et uniforme. Un noyau monolithique fonctionne comme un seul grand noyau et non pas à l'aide de composants distincts. Un noyau est uniforme quand chaque fonction est exécutée de la même manière à chaque fois qu'elle est lancée.

Le noyau est cependant très modulaire et conserve toutes ses fonctions dans la même adresse mémoire, les fonctions communiquant alors grâce à cette adresse. Pour garantir la stabilité du système, il place chacun des pilotes chargés dans une « bulle » virtuelle de sorte que les erreurs ne puissent pas se propager.

Toutes les fonctions du noyau sont localisées dans des éléments de code modulaires qui peuvent être soit intégrés au noyau compilé, soit compilés séparément en tant que modules. Dans ce dernier cas, ces modules pourront être chargés et déchargés dynamiquement.

7.2 Utilisation des modules

La commande permettant de connaître la version du noyau est `uname -a` et les modules sont stockés dans le répertoire `/lib/modules/<nom_kernel>`.

La commande permettant de lister l'ensemble des modules chargés est `lsmod`. Chacun des modules est associé à un champ appelé used : tant que le champ used est à 0, le module reste hors de la mémoire et dès qu'il passe à 1, il est chargé en mémoire.

Les commandes permettant de décharger et de charger les modules sont respectivement `rmmod` et `insmod`. On peut également utiliser les commandes `modprobe -r` et `modprobe -i`.

Il est possible de spécifier quels modules doivent être chargés au démarrage en les listant dans le fichier `/etc/modules`.

7.3 IPC

Les processus communiquent avec le kernel mais également entre eux grâce aux mécanismes IPC : les signaux, les pipes et les IPC System V.

7.3.1 Les signaux

Les signaux sont utilisé pour envoyer des évènements asynchrones à un ou plusieurs processus. Ils peuvent être générés soir à l'aide du clavier (Ctrl-C, Ctrl-Z, Ctrl-D,...), soit par la commande `kill`, soit par une condition d'erreur.

Les signaux sont également utilisés pour les communications inter-processus. La liste de tous les signaux disponibles sur un système est accessible par la commande `kill -l`.Tous les signaux sont associés à des nombres entiers : 1 (SIGHUP) pour le kill par défaut ou 9 (SIGKILL) qui ne peut être ignoré par n'importe quel processus (l'autre signal auquel les processus sont obligés de répondre et le SIGSTOP, signal 19).

N'importe quel processus ne peut pas envoyer des signaux à chacun des autres processus. Seuls le kernel et le super utilisateur peuvent le faire.. Les processus normaux ne peuvent envoyer des signaux qu'à des processus possédant les mêmes uid et gid ou apparetnant au même groupe de processus.

7.3.2 Les pipes

Les shells classiques autorisent les redirections. Par exemple, `ls | pr` associe la sortie de la commande ls à l'entrée de la commande pr. Les pipes sont des flux de d'octets unidirectionnels qui connectent la sortie standard d'un processus à l'entrée standard d'un autre processus. Aucun des deux processus n'a « conscience » de cette redirection et ils se comportent tous deux normalement.

Sous Linux, un pipe est implémenté en utilisant deux structures de données de type fichiers qui pointent chacune vers un même inode VFS temporaire qui lui-même pointe vers une page physique en mémoire.

Linux supporte également les pips nommés, connus également sous le nom de pipes FIFOs (First In, First Out). Les premières données inscrites dans le pipe sont les premières lues. À la différence des pipes classiques, les FIFOs ne sont pas des objets temporaires. Ce sont au contraire des entités du système de fichiers et peuvent être créés en utilisant la commande mkfifo. Les processus sont libres d'utiliser un pipe FIFO tant qu'ils possèdent les droits nécessaires sur celui-ci.

7.3.3 Les IPC system V

Linux supporte également trois types de mécanismes de communication interprocessus apparus initialement dans Unix System V en 1983. Ce sont les files de messages, les sémaphores et la mémoire partagée. Ces mécanismes partagent des méthodes d'autenficiation communes et les processus ne peuvent accéder à ces ressources qu'en passant un identificateur unique au noyau par l'intermédiaire d'appels systèmes.

Les files de messages permettent à un ou plusieurs processus d'écrire des messages qui seront lus par un ou plusieurs autres. Le noyau maintient une liste de files de messages, le vecteur `msgque` ; chacun des éléments de ce vecteur pointe vers une structure `msqid_ds` décrivant complètement la file. Quand une file est créée, une nouvelle structure `msqid_ds` est allouée et insérée dans le vecteur. À chaque fois qu'un processus essaie d'écrire un messsage dans la file d'écriture, ses identificateurs d'utilisateur et de groupe sont comparés avec le mode courant de cette file en utilisant la structure `ipc_perm`. Si le processus peut écrire dans la file d'écriture, le message est copié de la plage d'adresse du processus dans une structure `msg` et placée à la fin de la file.

Dans leur forme la plus simple, une sémaphore est une adresse mémoire dont la valeur peut être testée et positionnée par un ou plusieurs processus. Ces deux opérations sont des opérations ininterrompables et atomiques : une fois démarrée, rien ne peut les arrêter, ce qui signifie qu'un processus voulant modifier la valeur tandis qu'un autre processus la teste devra attendre la fin de ce dernier. Les sémaphores sont souvent utilisées pour créer des « régions critiques » définissant des zones de code crititque qu'un seul processus à la fois ne peut exécuter.

La mémoire partagée permet à un ou plusieurs processus de communiquer par l'intermédiaire d'une mémoire apparaissant dans toutes leurs espaces d'adressage virtuels. Les pages de mémoire virtuelle sont referencées par des entrées dans chacune des tables de partage des processus. Comme pour tous les IPC System V, l'accès à ce type de mémoire est contrôlée par des clés et des vérifications de droits d'accès. Une fois que la mémoire est partagée, il n'existe aucun contrôle sur la manière dont les processus l'utilise.