


**Université de Sfax**  
Institut Supérieur d'Informatique et de Multimédia de Sfax  
2009 / 2010

**Cours : Systèmes d'exploitation évolués**

**Le système d'Exploitation Linux**


Mohamed Ben Halima



- But
  - Présentation de Linux
  - Apprentissage de base pour bien démarrer avec Linux
- Application et expérimentation
  - Machines en double boot Windows/Linux
  - VirtualBox

2009/2010

Systèmes d'Exploitation Evolués




**Plan**

1. Historique
2. Présentation de Linux
3. Structure de Linux
4. Le Principes de fonctionnement de Linux
5. Le Shell Linux
6. Éléments d'administration de Linux
7. Les qualités du système Linux
8. Caractéristiques générales du noyau
9. Système de Gestion de Fichiers
10. Processus dans Linux
11. Implémentation des processus sous Linux
12. La gestion de la mémoire sous Linux
13. Entrées/Sorties dans Linux

2009/2010

Systèmes d'Exploitation Evolués




**1. Historique (1)**

**1969 – 1979 : les premiers pas universitaires**

- Été 1969 : Ken Thompson, aux BELL Laboratories, écrit la version expérimentale d'Linux : système de fichiers exploité dans un environnement mono-utilisateur, multi-tâche, le tout étant écrit en assembleur.
- 1ère justification officielle : traitement de texte pour secrétariat. Puis : étude des principes de programmation, de réseaux et de langages.

2009/2010

Systèmes d'Exploitation Evolués



## 1. Historique (2)

- Eté 1973 : réécriture du noyau et des utilitaires d'Linux en C.
- En 1974 distribution d'Linux aux Universités (Berkeley et Columbia notamment). Il se compose alors :
  - d'un système de fichiers modulaire et simple,
  - d'une interface unifiée vers les périphériques par l'intermédiaire du système de fichiers,
  - du multi-tâche,
  - et d'un interpréteur de commandes flexible et interchangeable.

2009/2010

Systèmes d'Exploitation Evolués

5



## 1. Historique (3)

### 1979 – 1984 : les premiers pas commerciaux

- En 1979, avec la version 7, Linux se développe commercialement

### 1984 – 1993 ... : la standardisation

- En 1984 le Système V.2 est adopté comme standard,
- En 1985 AT&T publie SVID (System V Interface Definition) qui définit l'interface d'application du Système V.2 et non pas son implémentation,
- En 1986, le Système V.3 apporte les Streams, les bibliothèques partagées et RFS (Remote File Sharing),
- En 1993, X/Open lance le COSE (Common Open Software Environment). Il s'agit d'accords entre constructeurs pour le développement d'applications dans un environnement commun.

2009/2010

Systèmes d'Exploitation Evolués

6



## 1. Historique (4)

### 1991 - ... : Linux, le renouveau d'Linux

- LINUX est une implantation libre des spécifications POSIX avec des extensions System V (AT&T) et BSD (Berkeley),
- En 1991, Linus B. Torvalds (Helsinki) utilise [MINIX](#),
- Août 1991 : 1ère version de LINUX 0.01. C'est une réécriture de MINIX, avec des ajouts de nouvelles fonctionnalités et la diffusion des sources sur « Internet » ,

2009/2010

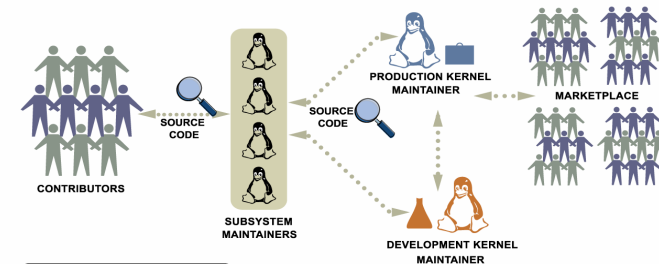
Systèmes d'Exploitation Evolués

7



## 2. Présentation de Linux

### Comment est maintenu le projet Linux ?



2009/2010

Systèmes d'Exploitation Evolués

8



© 2003 Open Source Development Labs

## 2. Présentation de Linux

### Développement du noyau Linux

Si au début de son histoire le développement du noyau Linux était assuré par des développeurs bénévoles, les principaux contributeurs sont aujourd'hui un ensemble d'entreprises, souvent concurrentes, comme Red Hat, Novell, IBM ou Intel.

La licence du noyau Linux est la licence publique générale GNU. Cette licence est libre, ce qui permet d'utiliser, copier et modifier le code source selon ses envies ou ses besoins. Ainsi, quiconque a les connaissances nécessaires peut participer aux tests et à l'évolution du noyau.



## 2. Présentation de Linux

### Mode de numérotation

Les numéros de version du noyau sont composés de trois chiffres : le premier est le numéro majeur, le second le numéro mineur.

Avant l'apparition des versions 2.6.x, les numéros mineurs pairs indiquaient une version stable et les numéros mineurs impairs une version de développement. Ainsi, les versions 2.2, 2.4 sont stables, les versions 2.3 et 2.5 sont des versions de développement.

Depuis la version 2.6 du noyau, ce modèle de numérotation stable/développement a été abandonné et il n'y a donc plus de signification particulière aux numéros mineurs pairs ou impairs. Le troisième chiffre indique une révision, ce qui correspond à des corrections de bogues, de sécurité ou un ajout de fonctionnalité.

Exemple 2.2.26, 2.4.30, 2.6.11 ou 2.6.32



## 2. Présentation de Linux

### Le projet GNU



Principe de base : le libre accès au code source accélère le progrès en matière d'informatique car l'innovation dépend de la diffusion du code source

La liberté au sens GNU est définie selon quatre principes (le copyleft GPL) :

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



## 2. Présentation de Linux

### La licence GPL

La licence GPL (General Public licence)  
[www.gnu.org/copyleft/gpl.html](http://www.gnu.org/copyleft/gpl.html)

- Autorise l'utilisateur à copier et distribuer à volonté le logiciel qu'elle protège, pourvu qu'il n'interdise pas à ses pairs de le faire aussi,
- Requiert aussi que tout dérivé d'un travail placé sous sa protection soit lui aussi protégé par elle,
- Quand la GPL évoque les logiciels libre, elle traite de liberté et non de gratuité (un logiciel GPL peut être vendu),
- Remarque : en anglais « free » mélange gratuité et liberté.



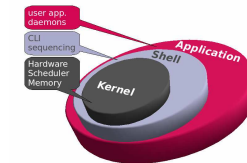
## 2. Présentation de Linux Distributions Linux

Une distribution est un noyau auquel des logiciels ont été ajoutés  
Possibilités de créer des distributions dédiées à un usage particulier



## 3. Structure de Linux (1)

- Le noyau gère les tâches de base du système :
  - L'initialisation du système
  - La gestion des ressources
  - La gestion des processus
  - La gestion des fichiers
  - La gestion des Entrées/Sorties



- L'utilisateur communique avec le noyau par l'intermédiaire d'un SHELL. Les Shells sont aussi des langages de commandes et de programmation.

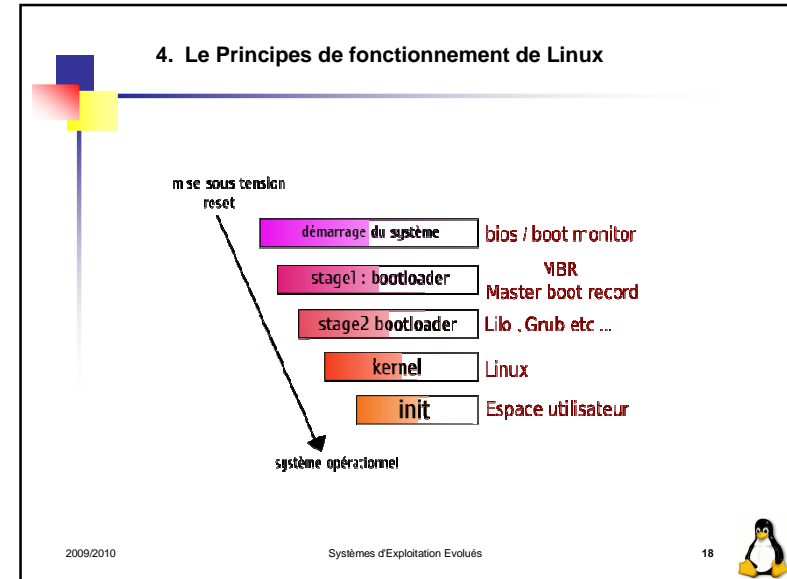
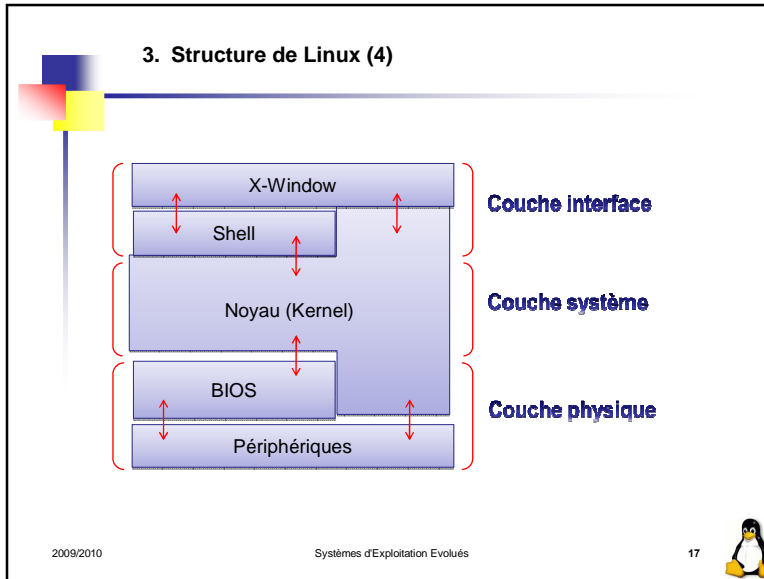
## 3. Structure de Linux (2)

- Les utilitaires sont des outils d'interfaçage avec le système, de programmation et de communication.

- Les shells les plus connus sont :
  - BOURNE SHELL (sh)
  - KORN-SHELL (ksh)
  - C-SHELL
  - TC-SHELL
  - BOURNE AGAIN SHELL (bash)

## 3. Structure de Linux (3)

- Multi-tâches, multi-usagers depuis le début
- Le système Linux initial était aussi préoccupé par les limitation du matériel
- Distinction entre:
  - programmes du système
  - noyau
    - tout ce qu'il y a entre l'interface des appels de système et le matériel
    - fournit dans une seule couche un grand nombre de fonctionnalités
      - système fichiers, ordonnancement UCT, gestion mémoire...
- Plus modulaire et protégé que MS-DOS



- ### 4. Le Principes de fonctionnement de Linux
- **Boot et lancement du noyau**
  - **Processus *init***
  - **Services et démons**
  - **Les *runlevels***
  - **Scripts de lancement des services**
- 2009/2010 Systèmes d'Exploitation Evolués 19

- ### 4. Le Principes de fonctionnement de Linux Boot et lancement du noyau
- Lancement du système : boot et chargement du noyau**
- Au *boot* le BIOS exécute le MBR (Master Boot Record) situé sur le premier secteur (512 octets) du support bootable choisi (disque, CD, clef USB, ...)
  - Le MBR :
    - scanne le disque pour trouver la partition bootable (flag)
    - lance le *boot loader* (chargeur de démarrage) du secteur de boot (premier secteur) de la partition bootable
  - Le *bootloader* :
    - charge le *noyau* en mémoire et l'exécute
    - charge le *ramdisk* `initrd.img` en mémoire
  - 2 bootloaders possibles:
    - LILO (Linux LOader)
    - GRUB (GRand Unified Bootloader)
- 2009/2010 Systèmes d'Exploitation Evolués 20

#### 4. Le Principes de fonctionnement de Linux Processus *init*

Lancement du système : boot -> init

- Une fois le noyau chargé en mémoire, il lance le premier processus :  
/bin/init
- *init* est le père de tous les autres processus qui seront créés par l'appel `system fork()`
- *init* lit le fichier `/etc/inittab` pour savoir :
  - quel est le fichier à exécuter pour continuer le chargement du système
  - quel est le *runlevel* (niveau d'exécution) par défaut
  - comment lancer les services pour un *runlevel* donné
  - ...



#### 4. Le Principes de fonctionnement de Linux Services et démons

Lancement du système : boot -> init -> modules/services

- Après le chargement du noyau, le script correspondant à `sysinit` dans fichier `inittab` est chargé :
  - Mandriva : `/etc/rc.d/rc.sysinit`
  - Debian : `/etc/rc.d/rcS` lance les scripts `/etc/rcS.d/S??*`
- Ce script d'initialisation est chargé de 2 tâches fondamentales :
  - charger les modules dans le noyau (gestion des périphériques)
  - démarrer les services en exécutant les processus «Deferred Auxiliary Executive Monitor» (*daemons*) correspondant, en français : démons



#### 4. Le Principes de fonctionnement de Linux Les *runlevels*

Lancement du système : boot -> init -> services

- Le mécanisme de démarrage des services est caractéristique d'une distribution (incompatibilités entre distributions) :
  - Mandriva, Debian, RedHat, ... mécanisme dérivé d'«Unix System V»
  - Slackware, FreeBSD, NetBSD, ... mécanisme dérivée d'«Unix BSD»
- le répertoire `/etc/init.d` contient tous les scripts de gestion des services installés (1 service <-> 1 ou plusieurs démon(s))
- les lignes `«/etc/rc.d/rc x»` du fichier `/etc/inittab` déterminent le lancement des scripts pour le *runlevel* *x*
- Le *runlevel* de l'action `initdefault` est lancé par la ligne correspondante

```
id:5:initdefault:
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
...
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```



#### 4. Le Principes de fonctionnement de Linux Les *runlevels*

Lancement du système : boot -> init -> services

- Le *runlevel* (numéro de 0 à 6) fixe le répertoire de démarrage des services :
  - Mandriva -> répertoires `/etc/rc.d/rc[0-6].d`
  - Debian -> répertoires `/etc/rc[0-6].d`
- `rcX.d` : contient des liens symboliques vers les scripts de gestion des services qui sont dans le répertoire :
  - `/etc/rc.d/init.d` (Mandriva, + lien symbolique vers `/etc/init.d`)
  - `/etc/init.d` (Debian)
- Les liens sont formés selon la syntaxe : `[S/K]XX<nom_du_script>`
  - *S* lance le script avec l'argument `start` (démarrage du service)
  - *K* lance le script avec l'argument `stop` (arrêt du service)
  - *XX* est un rang qui fixe l'ordre dans lequel les scripts sont lancés



#### 4. Le Principes de fonctionnement de Linux Les Scripts de lancement des services

Lancement du système : boot -> init -> services

- Utilitaires en mode console :
  - Debian : update-rc.d
  - Mandriva : chkconfig, service
- le script /etc/rc.local peut contenir des personnalisations locales qui seront lancées à la fin du processus init

- Pour démarrer un service sous mandrake, on peut taper :

```
service <nom_du_service> start
```

ou encore :

```
/etc/rc.d/init.d/<script_correspondant_au_service> action
```

**action** : start | stop | restart | status | ...

2009/2010

Systèmes d'Exploitation Evolués

25



#### 5. Le Shell Linux(1)

- Le Shell est un programme (application), qui assure l'interface entre les différents programmes et la machine
- Interprète les commandes
  - ✓ Gère les I/O utilisateur sur le terminal
  - ✓ Mémorise le set-up de l'environnement de l'utilisateur dans le fichier .profile
- Les utilisateurs communiquent avec *sh*
- Commandes internes : gérées dans le shel (set, unset)
- Commandes externes : exécutées en tant que programmes (ls, grep, sort, ps)

2009/2010

Systèmes d'Exploitation Evolués

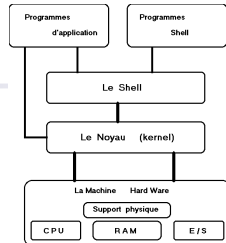
26



#### 5. Le Shell Linux(2)

##### Propriétés du Shell Linux

- Interactive,
- Exécution des programmes en arrière plan (non interactifs),
- Redirection des E/S,
- Connexion de pgms ensembles à travers les tubes (pipes)
- A un langage simple pour écrire des scripts,



2009/2010

Systèmes d'Exploitation Evolués

27



#### 5. Le Shell Linux(3)

##### A quoi cela sert-il ?

- Administrer.
- Tuer des processus récalcitrants (Indisciplinés).
- Sur une machine multiprocesseur, avoir une idée de son occupation.
- Automatiser des traitements massifs ou un même programme doit être lancé des dizaines de fois.
- Savoir se connecter à une machine Linux distante pour y prendre ou mettre des fichiers ainsi qu'y lancer des programmes.

2009/2010

Systèmes d'Exploitation Evolués

28



## 5. Le Shell Linux(4)

### Syntaxe d'une commande Shell

#### Syntaxe :

```
command [ -options ] [ arguments ]
```

#### Exemples :

```
Contexte  (root@machine1:~#) pwd (Commande à exécuter)
Résultat de la commande (/home/root)
root@machine1:~#
```



## 6. Éléments d'administration de Linux

- Les 2 modes d'administration
- Notion de « fichier spécial »
- Nommage des périphériques de boot
- Partitionnement des disques
- Formatage disque et *filesystem*
- Le « montage » des périphériques
- Les gestionnaires de paquets (rpm et Debian)
- Configuration du *bootloader* (LILO)



## 6. Éléments d'administration de Linux Les 2 modes d'administration

Linux supporte 2 modes d'administration :

« À la main » :

- Édition (manuelle) des fichiers de configuration
- Utilisation (manuelle) des commandes d'administration
- Utilisation (manuelle) des gestionnaires de paquets RPM ou DEBIAN
- Édition de scripts de commande (langage : shell, perl, awk, ...)

Avec des logiciels d'administration (graphique ou mode caractère) :

- Qui manipulent les fichiers de configuration
- Qui utilisent des commandes d'administration standard ou spécifiques
- *linuxconf*, *webmin*, *DrakConf*, ...



## 6. Éléments d'administration de Linux Notion de « fichier Spécial »

Principe :

- Sous Unix tout est fichier
- => tous les périphériques sont représentés par un fichier spécial dans le répertoire */dev*
- disques, clavier, souris, carte son, ports d'E/S, ...

```
Carte son  -> brw-rw---- 1 jlc audio  14,  3 sep 28 11:36 dsp
              lrwxrwxrwx  1 root root   3 sep 28 11:34 dvd -> hdc
              ...
              brw-rw---- 1 jlc floppy  2,  0 sep 28 11:34 fd0
              brw-rw---- 1 jlc floppy  2,  1 sep 28 11:34 fd1
              ...
Disque dur  -> brw-rw---- 1 root disk   3,  0 sep 28 11:33 hda
              brw-rw---- 1 root root   3,  1 sep 28 11:33 hda1
              brw-rw---- 1 root root   3,  2 sep 28 11:33 hda2
              brw-rw---- 1 root root   3,  5 sep 28 11:33 hda5
              brw-rw---- 1 root root   3,  6 sep 28 11:33 hda6
Lect. DVD  -> brw-rw---- 1 jlc cdrom  22,  0 sep 28 11:34 hdc
Lect. CD   -> brw-rw---- 1 jlc cdrom  22, 64 sep 28 11:34 hdd
              ...
```





## 6. Éléments d'administration de Linux Notion de «fichier Spécial»

### Principe :

- Sous Unix tout est fichier

=> tous les périphériques

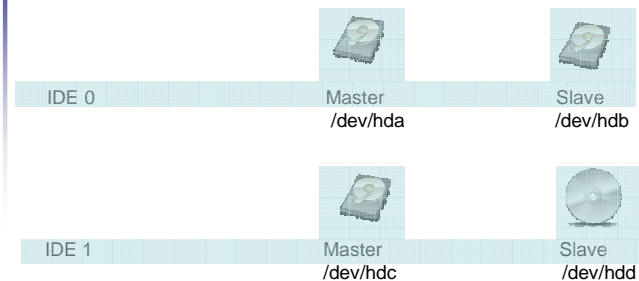
- disques
- clavier
- souris
- carte son
- ports d'E/S
- sockets réseau
- mémoire ...

...sont représentés par un fichier spécial dans le répertoire /dev

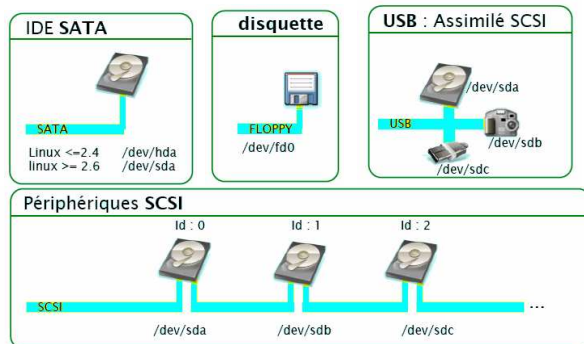


## 6. Éléments d'administration de Linux Appellation des périphériques

### Périphériques IDE



## 6. Éléments d'administration de Linux Appellation des périphériques

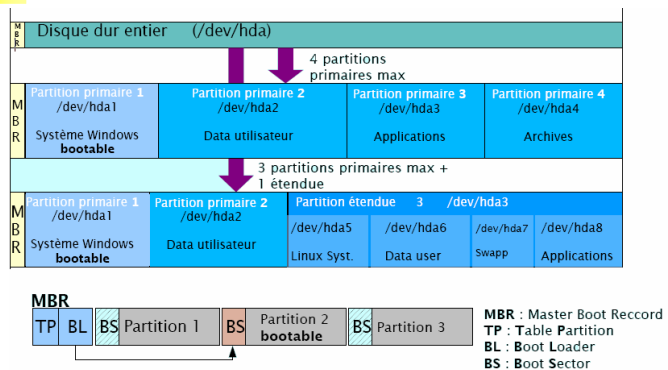


## 6. Éléments d'administration de Linux Partitionnement des disques

- La plupart des systèmes d'exploitation « correctement » installés utilisent un disque avec plusieurs partitions:
  - partition « système » (fichiers systèmes, fichiers de configuration ...)
  - partition « utilisateurs » (données des utilisateurs)
  - ...
- Exploitation plus sécurisée
  - on peut formater une partition indépendamment des autres
  - on peut utiliser une partition en lecture seule
  - ...
- partitionnement statique => planifier le partitionnement
  - on ne peut pas modifier simplement un partitionnement statique
  - partitionner est une opération « low level », risquée !!
  - ...
- Pour bénéficier des avantages du partitionnement dynamique il faut passer à des solutions de type RAID (Redondant Array of Independent Disks) ou LVM (Logical Volume Manager)



## 6. Éléments d'administration de Linux Partitionnement des disques



2009/2010

Systèmes d'Exploitation Evolués

37



## 6. Éléments d'administration de Linux Partitionnement/Filesystem

Partitionnement et formatage du disque dur :

- Formatage « bas niveau » (physique, en usine)
- Partitionnement (à l'installation de l'OS)
  - *fdisk*, *PartitionMagic* (DOS)
  - *fdisk*, *parted*, *partman* (Linux) ..
  - à l'installation de Linux (menu caractère, menu graphique)
- « Formatage » « haut niveau » (logique, dépend de l'OS et du FileSystem cible)
  - *format* (Windows : créer un filesystem FAT ou NTFS)
  - *mkfs* (Unix : créer un filesystem Ext2, Ext3, FAT, ...)

Exemple : `mkfs -t ext2 /dev/hda1`  
`mkfs -t fat /dev/fd0`

Système de fichiers journalisés (ext3): plus robuste aux pannes secteurs

2009/2010

Systèmes d'Exploitation Evolués

38



## 6. Éléments d'administration de Linux Partitionnement/Filesystem

### Arborescence du système

#### / : Racine du système

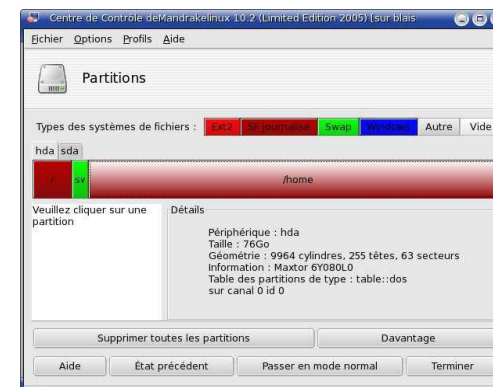
- etc/ : fichiers de configuration
- bin/ : programmes et commandes de base
- boot/ : noyau du système
- tmp/ : fichiers temporaires créés et utilisés par le système
- var/ : divers fichiers relatifs au système d'exploitation (logs, files, cache, ...)
- usr/ : programmes utilisateurs, bibliothèques, fichiers partagés, ...
- home/ : répertoires personnels des utilisateurs
  - toto/
  - jean/
- mnt/ : communément utilisé pour les divers "montages" de disques ou partitions
  - cdrom/
  - windows/

39



## 6. Éléments d'administration de Linux Partitionnement/Filesystem

Installation Mandriva / outil DrakConf



2009/2010

40



## 6. Éléments d'administration de Linux Filesystem

Système d'exploitation	Types de système de fichiers supportés
Dos	FAT16
Windows 95	FAT16
Windows 95 OSR2	FAT16, FAT32
Windows 98	FAT16, FAT32
Windows NT4	FAT, NTFS version4
Windows 2000/XP	FAT, FAT16, FAT32, NTFS (versions 4 et 5)
Linux	Ext2, Ext3, ReiserFS Linux Swap, FAT,NTFS
MacOS	HFS, MFS
SGI IRIX	XFS
FreeBSD	UFS
Sun Solaris	UFS
IBM AIX	JFS

2009/2010

Systèmes d'Exploitation Evolués

41



## 6. Éléments d'administration de Linux Montage des périphériques

L'opération de montage des périphériques :

- DOS et Windows utilisent la notion d'unité logique pour fournir un accès aux ressources de stockage ( A: -> floppy, C: -> disque dur, ... E: -> lecteur CD)
  - Tous les Unix utilisent la notion de montage :
    - un périphérique est associé à un point de montage (répertoire) par une « opération de montage » (*commande mount*)
    - la commande mount peut être utilisée « à la main »
- exemple :
- ```
mount /dev/hda1 /
mount /dev/sda1 /mnt/removable
```
- -tous les périphériques montés bénéficient du « cache disque »

2009/2010

Systèmes d'Exploitation Evolués

42



## 6. Éléments d'administration de Linux Montage des périphériques

- Avant d'extraire un périphérique amovible (disquette, clef USB, etc.), nous DEVONS le démonter (*umount*), pour synchroniser les écritures (vidage du cache disk)
- Tous les montages permanents sont indiqués dans le fichier */etc/fstab*

La commande *df* affiche la liste des périphériques montés + propriétés :

```
[root@jot ~]# df
Sys. de fich.      Tail. Occ. Disp. %Occ. Monté sur
/dev/hda1         9,9G  4,4G  5,0G  47% /
/dev/hdb2         69G   40G  20G  68% /home
/dev/hda6         9,7G 1010M  8,5G  11% /opt
none             253M   81M  172M  32% /tmp
/dev/hda5         9,9G 496M  8,5G   6% /var
[root@jot ~]#
```

2009/2010

Systèmes d'Exploitation Evolués

43



## 7) Les qualités du système Linux

1. Code source facile à lire et à modifier ; disponible commercialement.
2. Interface utilisateur simple ; non-conviviale mais très puissante.
3. Le système est construit sur un petit nombre de primitives de base ; de nombreuses combinaisons possibles entre programmes.
4. Les fichiers ne sont pas structurés au niveau des données, ce qui favorise une utilisation simple.
5. Toutes les interfaces avec les périphériques sont unifiées (système de fichier).
6. Le programmeur n'a jamais à se soucier de l'architecture de la machine sur laquelle il travaille.
7. C'est un système disponible sur de nombreuses machines, allant du super-calculateur au microordinateur (PC).
8. Les utilitaires et programmes proposés en standard sont très nombreux.

2009/2010

Systèmes d'Exploitation Evolués

44



## 8) Caractéristiques générales du noyau (1)

```

graph TD
    A[PROGRAMMES UTILISATEURS] -- Appels Systèmes --> B[NOYAU LINUX]
    B --> C[GESTIONNAIRE de PERIPHERIQUES]
    C --> D[PERIPHERIQUES]
    
```

### 1. Multi-tâche / multi-utilisateur

- Plusieurs utilisateurs peuvent travailler en même temps ; chaque utilisateur peut effectuer une ou plusieurs tâches en même temps.
- Une tâche ou un processus = programme s'exécutant dans un environnement spécifique.
- Les tâches sont protégées; certaines peuvent communiquer, c-à-d échanger ou partager des données, se synchroniser dans leur exécution ou le partage de ressources. Certaines tâches peuvent être « temps réel ».

2009/2010 Systèmes d'Exploitation Evolués 45

## 8) Caractéristiques générales du noyau (2)

### 2. Système de fichiers arborescent

Arborescence unique de fichiers, même avec plusieurs périphériques (disques) de stockage.

### 3. Entrée/Sorties compatible fichiers, périphériques et processus

- Les périphériques sont manipulés comme des fichiers ordinaires.
- Les canaux de communication entre les processus (pipe) s'utilisent avec les mêmes appels systèmes que ceux destinés à la manipulation des fichiers.

2009/2010 Systèmes d'Exploitation Evolués 46

## 8. Caractéristiques générales du noyau (3)

### Réduction du noyau système

- Linux comprend un noyau (kernel) et des utilitaires. Irremplaçable par l'utilisateur.
- Le noyau gère les processus, les ressources (mémoires, périphériques ...) et les fichiers.
- Tout autre traitement doit être pris en charge par des utilitaires ; c'est le cas de l'interprète de commande (sh, csh, ksh, tcsh ...).

### Interface au noyau

- L'interface entre le noyau Linux et les périphériques est assurée par les **gestionnaires de périphériques** (devices driver)
- L'interface entre le noyau Linux et les programmes est assurée par un ensemble d'**appels systèmes**

2009/2010 Systèmes d'Exploitation Evolués 47

## 9. Système de Gestion de Fichiers (1)


- Le système de gestion de fichiers est un outil de manipulation des fichiers et de la structure d'arborescence des fichiers sur disque et a aussi le rôle sous Linux de conserver toutes les informations dont la pérennité est importante pour le système
- Il permet de plus une utilisation facile des fichiers et gère de façon transparente les différents problèmes d'accès aux supports de masse
- Ce principe est différent de celui employé par les systèmes MS-DOS et Windows, pour lesquels chaque volume (disque) possède une racine spécifique repérée par une lettre (A:\, C:\, etc).

2009/2010 Systèmes d'Exploitation Evolués 48

## 9. Système de Gestion de Fichiers (2)

### Le concept de fichier

- L'unité logique de base du S.G.F. le fichier
- Le contenu est entièrement défini par le créateur
- Sur Linux les fichiers ne sont pas typés
- Matérialisé par une inode et des blocs du disque

2009/2010 Systèmes d'Exploitation Evolués 49 

## 9. Système de Gestion de Fichiers (3)

- fichiers -


- Affichage des caractéristiques: `ls -l`

```

-rw-r--r-- 1 user01 ISIMS 58K 16 Jul 09:19 tp1.tex
  
```

Diagram explaining the fields of the `ls -l` output:

- `-rw-r--r--`: permissions (type: `-`, user: `rw`, group: `r--`, others: `r--`)
- `1`: nb liens
- `user01`: propriétaire
- `ISIMS`: groupe
- `58K`: taille
- `16 Jul 09:19`: date
- `tp1.tex`: nom

2009/2010 Systèmes d'Exploitation Evolués 50 


## 9. Système de Gestion de Fichiers (4)

- fichiers -

- Changer les permissions: `chmod`

`chmod <classe op perm, ...>|nnn <fic>`


|                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |   |   |   |   |   |   |      |   |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---|---|---|---|---|------|---|
| <p>– classe:</p> <ul style="list-style-type: none"> <li>u : user</li> <li>g : group</li> <li>o : others</li> <li>a : all</li> </ul> <p>– op:</p> <ul style="list-style-type: none"> <li>= : affectation</li> <li>- : suppr.</li> <li>+ : ajout</li> </ul> <p>– perm:</p> <ul style="list-style-type: none"> <li>r : lecture</li> <li>w : écriture</li> <li>x : exécution</li> </ul> | <p>– chaque perm = 1 valeur:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>r</td><td>4</td></tr> <tr><td>w</td><td>2</td></tr> <tr><td>x</td><td>1</td></tr> <tr><td>rien</td><td>0</td></tr> </table> <p>– déf. des permissions (par addition) pour chaque classe</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>exemples:</p> <pre> chmod u=rwx,g=rx,o=r tp1.tex chmod a+x script.sh chmod 755 script.sh </pre> </div> | r | 4 | w | 2 | x | 1 | rien | 0 |
| r                                                                                                                                                                                                                                                                                                                                                                                   | 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |   |   |   |   |   |      |   |
| w                                                                                                                                                                                                                                                                                                                                                                                   | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |   |   |   |   |   |      |   |
| x                                                                                                                                                                                                                                                                                                                                                                                   | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |   |   |   |   |   |      |   |
| rien                                                                                                                                                                                                                                                                                                                                                                                | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |   |   |   |   |   |      |   |

2009/2010 Systèmes d'Exploitation Evolués 51 

## 9. Système de Gestion de Fichiers (5)

- liens -

- Liens physiques
  - `ln <nom_fic> <nouveau_nom_fic>`
  - permet de donner plusieurs noms à un fichier
  - pas pour les répertoires
  - ne traverse pas les partitions
  - un fic est détruit quand TOUS ses liens physiques sont supprimés (≠ raccourcis)
- Liens symboliques
  - `ln -s <nom_fic> <nouveau_nom_fic>`
  - crée un **raccourci**
  - traverse les partitions
  - fonctionne aussi pour les répertoires
- Lister les liens d'un fichier: `ls -l <nom_fic>`

2009/2010 Systèmes d'Exploitation Evolués 52 

## 9. Système de Gestion de Fichiers (6)

### Les inodes

- L'inode est le centre de tous les échanges entre le disque et la mémoire,
- L'inode est la structure qui contient toutes les informations sur un fichier donné à l'exception de sa référence, dans l'arborescence.

Les informations stockées dans une inode disque sont :

- utilisateur propriétaire
- groupe propriétaire
- type de fichier
- droits d'accès
- date de dernier accès
- date de dernière modification
- date de dernière modification de l'inode
- taille du fichier
- adresses des blocs disque contenant le fichier.

2009/2010

Systèmes d'Exploitation Evolués

53



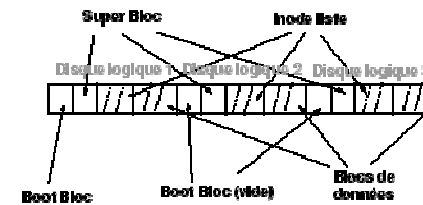
## 9. Système de Gestion de Fichiers (7)

### Organisation des disques

Structure du système de fichiers sur un disque logique



Plusieurs disques logiques sur un disque physique



2009/2010

54



## 9. Système de Gestion de Fichiers (8)

### Organisation des disques

#### Boot bloc

utilisé au chargement du système.

#### Super Bloc

il contient toutes les informations générales sur le disque logique.

#### Inode list

Table des inodes.

#### blocs de données

chaînés à la création du disque (mkfs).

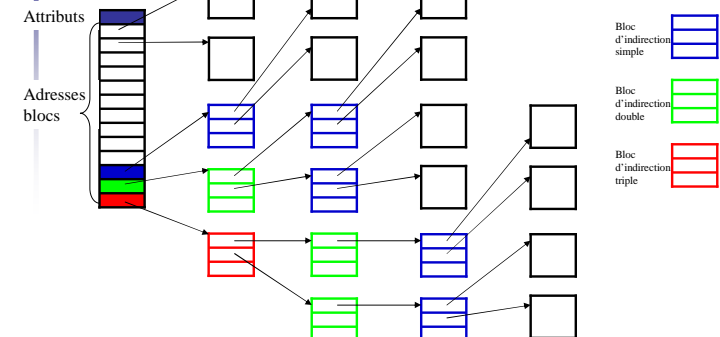
2009/2010

Systèmes d'Exploitation Evolués

55



## 9. Système de Gestion de Fichiers (9)



2009/2010

Systèmes d'Exploitation Evolués

56



## 9. Système de Gestion de Fichiers (10)

### Adressage des blocs dans les inodes

Le système d'adressage des blocs dans les inodes (Système V) consiste en 13 adresses de blocs :

- Les 10 premières adresses sont des adresses qui pointent directement sur les blocs de données du fichier
- Les autres sont des adresses indirectes vers des blocs de données contenant des adresses

Les inodes sont stockés en mémoire tant que le fichier est ouvert



## 9. Système de Gestion de Fichiers (11)

Capacité de la structure d'index : numéro de bloc sur 32 bits (4 octets), et bloc de 1 KO.

On peut donc mettre  $p = 256$  numéros de blocs dans un bloc.

- blocs directs : 10 blocs,
- bloc indirect\_1 : 256 blocs,
- bloc indirect\_2 :  $256^2$  blocs,
- bloc indirect\_3 :  $256^3$  blocs.

Nombre maximum de blocs dans un fichier :  $10 + 256 + 256^2 + 256^3$

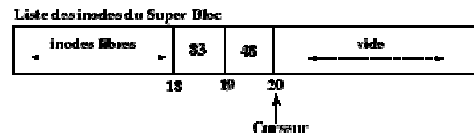
Taille maximale d'un fichier ~ 16 Go !!!



## 9. Système de Gestion de Fichiers (12)

### Allocation des inodes d'un disque

L'allocation des inodes est réalisée en recherchant dans la zone des inodes du disque un inode libre. Pour accélérer cette recherche : un tampon d'inodes libres est géré dans le SuperBloc, de plus l'indice du premier inode libre est gardé en référence dans le SuperBloc afin de redémarrer la recherche qu'à partir du premier inode réellement libre.



Inodes libres dans le SuperBloc

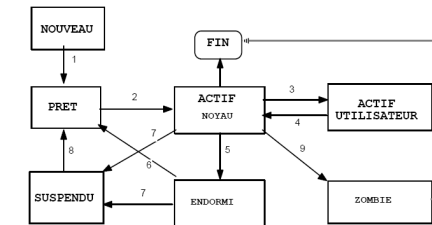


## 10. Processus dans Linux (1)

Processus = objet dynamique qui représente un programme en cours d'exécution et son contexte

Caractéristiques:

- identification (pid)
- identification du proc. parent (ppid)
- propriétaire
- priorité
- ...



## 10. Processus dans Linux (2)

- 1: le processus créé par fork a acquis les ressources nécessaires à son exécution
- 2 : le processus vient d'être élu par l'ordonnanceur
- 3 : le processus revient d'un appel système ou d'une interruption
- 4 : le processus a réalisé un appel système ou une interruption est survenue
- 5 : le processus se met en attente d'un événement (libération de ressource, terminaison de processus par wait). Il ne consomme pas de temps UC
- 6 : l'événement attendu par le processus s'est produit
- 7 : conséquence d'un signal particulier
- 8 : réveil du processus par le signal de continuation SIGCONT
- 9 : le processus s'achève par exit, mais son père n'a pas pris connaissance de sa terminaison. Il ne consomme pas de temps UC et ne mobilise que la ressource table des processus

2009/2010

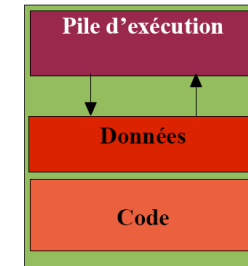
Systèmes d'Exploitation Evolués

61



## 10. Processus dans Linux (3)

Un processus comporte du **code machine exécutable**, une **zone mémoire** (données allouées par le processus) et une **pile** (pour les variables locales des fonctions et la gestion des sous-programmes)



2009/2010

Systèmes d'Exploitation Evolués

62



## 10. Processus dans Linux (4)

Les processus sont composés d'un espace de travail en mémoire formé de 3 segments :

- Le code correspond aux instructions, en langage d'assemblage, du programme à exécuter.
- La zone de données contient les variables globales ou statiques du programme ainsi que les allocations dynamiques de mémoire.
- les appels de fonctions, avec leurs paramètres et leurs variables locales, viennent s'empiler sur la pile.

Les zones de pile et de données ont des frontières mobiles qui croissent en sens inverse lors de l'exécution du programme.

2009/2010

Systèmes d'Exploitation Evolués

63



## 10. Processus dans Linux (5)

Le noyau maintient une table pour gérer l'ensemble des processus. Cette table contient la liste de tous les processus avec des informations concernant chaque processus.

Le nombre des emplacements dans cette table des processus est limité pour chaque système et pour chaque utilisateur.

2009/2010

Systèmes d'Exploitation Evolués

64





## 10. Processus dans Linux (6)

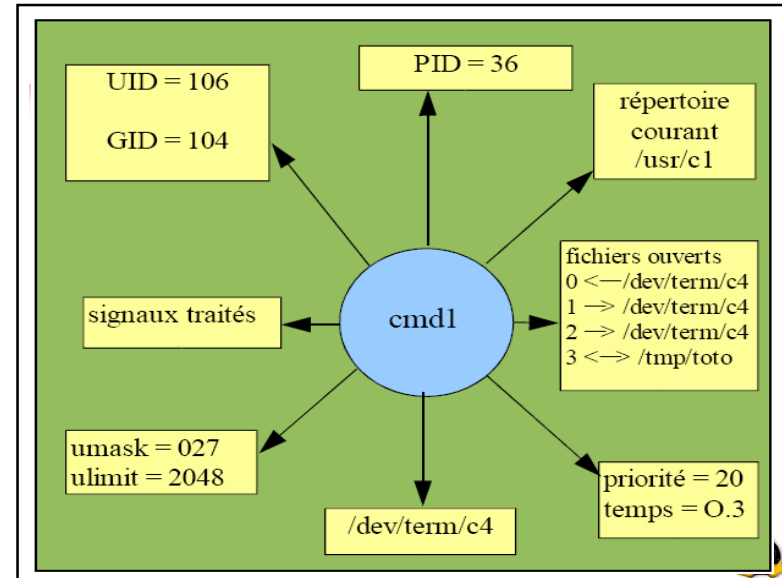
L'environnement d'un processus encore appelé *contexte*, comprend entre autre :

- Un numéro d'identification unique appelé **PID** (Process Identifier) ;
- Le numéro d'identification de l'utilisateur qui a lancé ce processus, appelé **UID** (User Identifier), et le numéro du groupe auquel appartient cet utilisateur, appelé **GID** (Group Identifier) ;
- Le **répertoire courant** ;
- Les **fichiers ouverts** par ce processus ;
- Le **masque** de création de fichier, appelé *umask* ;
- La **taille maximale** des fichiers que ce processus peut créer, appelée *ulimit* ;
- La **priorité** ;
- Les temps d'exécution ;
- Le **terminal de contrôle**, c'est-à-dire le terminal à partir duquel la commande a été lancée.

2009/2010

Systèmes d'Exploitation Evolués

65



## 10. Processus dans Linux (8)

### Création de processus

Pour chaque commande lancée (sauf les commandes internes), le shell crée automatiquement un nouveau processus.

Il y a donc 2 processus. Le premier, appelé **processus père**, exécute le programme Shell, et le deuxième, appelé **processus fils**, exécute la commande.

Le fils **hérite** de tout l'environnement du père, sauf bien sûr du **PID**, du **PPID** et des temps d'exécution.

Le **PPID** est le **PID** du processus père.

2009/2010

Systèmes d'Exploitation Evolués

67



## 10. Processus dans Linux (9)

### Création de processus

Sous Linux la création de processus est réalisée par l'appel système :

```
int fork(void)
```

Tous les processus sauf le processus d'identification 1 (init) sont créés par un appel fork

Le processus qui appelle le fork est appelé processus père. Le nouveau processus est appelé processus fils.

Tout processus a un seul processus père. Tout processus peut avoir zéro ou plusieurs processus fils.

2009/2010

Systèmes d'Exploitation Evolués

68



## 10. Processus dans Linux (10)

### Création de processus

Voyons ce qu'il se passe lorsque qu'un shell exécute la commande

`compress toto`

qui demande la compression du fichier nommé toto :

1. Le shell se duplique (fork); on a alors deux processus shell identiques.
2. Le shell père se met en attente de la fin du fils (wait).
3. Le shell fils remplace son exécutable par celui de la commande `compress`;
4. La commande `compress` s'exécute et compacte le fichier `toto`; lorsqu'elle termine, le processus fils disparaît.
5. Le père est alors réactivé, et affiche le prompt suivant.

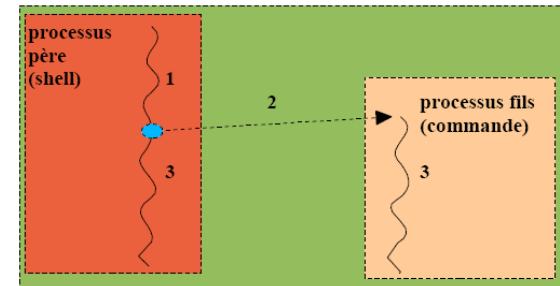
2009/2010

Systèmes d'Exploitation Evolués

69



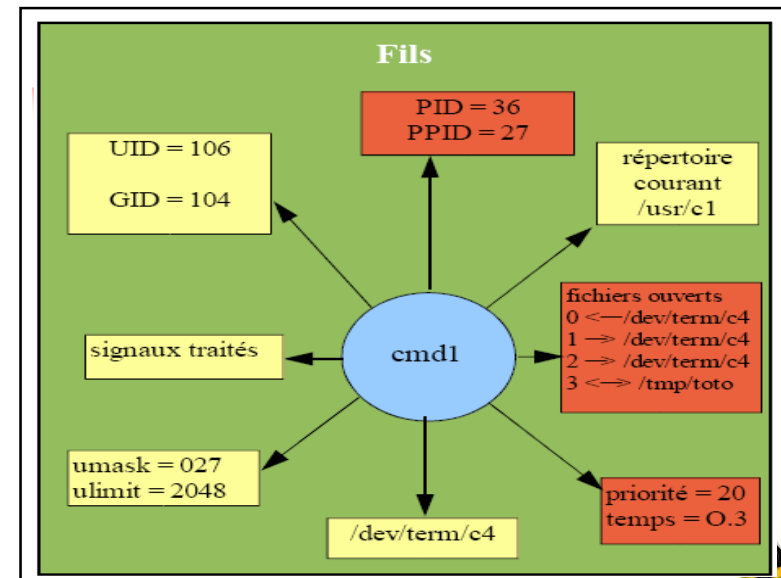
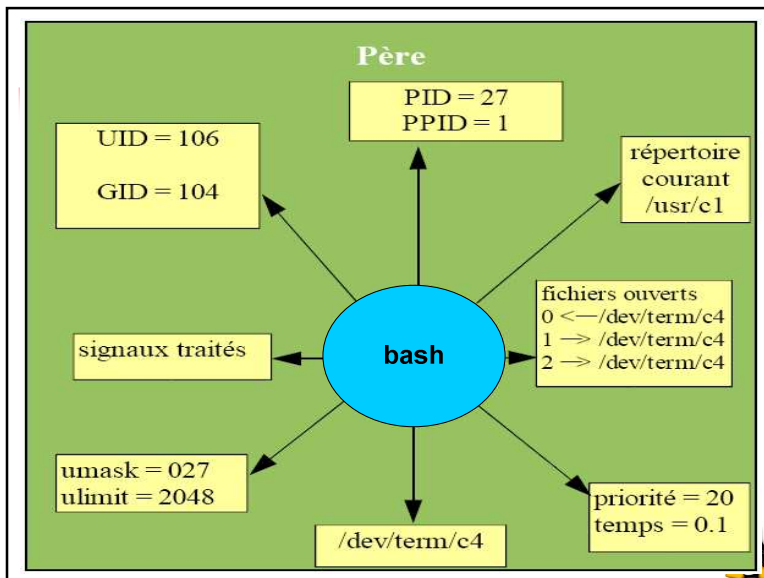
## 10. Processus dans Linux (11)



2009/2010

Systèmes d'Exploitation Evolués

70



## 10. Processus dans Linux (14)

On utilisera cette solution (processus lancés en parallèle) par exemple pour lancer un traitement très long, et continuer à travailler en même temps. Dans ce cas, on dit que le père a lancé un fils en *tâche de fond (background)* ou encore en *mode asynchrone*.

Pour lancer une commande en tâche de fond, il faut faire suivre cette commande par le caractère '&'

Une autre solution consiste à placer le processus père en attente jusqu'à ce que le processus fils soit terminé.

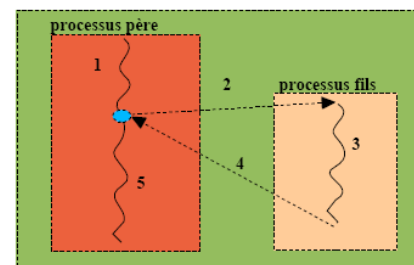
2009/2010

Systèmes d'Exploitation Evolués

73



## 10. Processus dans Linux (15)



Pour lancer une commande en plaçant le père en attente, il suffit de taper cette commande :

```
$ cmd1
... résultat de la commande cmd1
$
```

Ce mode est donc le mode par défaut dans le shell.

2009/2010

Systèmes d'Exploitation Evolués

74



## 10. Processus dans Linux (16)

### Arborescence de processus

Tous les processus sont créés à partir d'un processus père, existant déjà.

Le premier processus est un peu spécial. Il est créé lorsque le système est initialisé. Il s'appelle "*init*", a le **PID 0** et n'est associé à aucun terminal. Son travail consiste à créer de nouveaux processus.

2009/2010

Systèmes d'Exploitation Evolués

75



## 10. Processus dans Linux (17)

### Arborescence de processus

Le processus "*init*" crée 2 sortes de processus :

- ❑ des **démons**, c'est-à-dire des processus qui ne sont rattachés à aucun terminal, qui sont **endormis** la plupart du temps, mais qui se **réveillent** de temps en temps pour effectuer une tâche précise (par exemple la gestion des imprimantes).
- ❑ des **processus interactifs**, associés aux lignes d'entrées/sorties sur lesquelles sont rattachés des terminaux. Autrement dit des processus vous permettant de vous connecter.

2009/2010

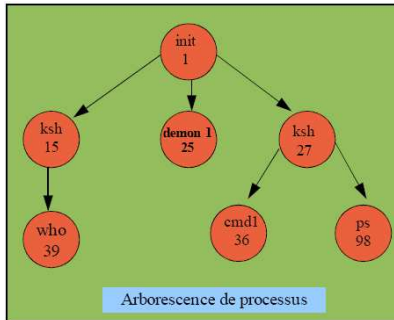
Systèmes d'Exploitation Evolués

76



## 10. Processus dans Linux (18)

### Arborescence de processus



Pour visualiser les processus que vous avez lancé, tapez la Commande : **ps**

2009/2010

Systèmes d'Exploitation Evolués

77



## 10. Processus dans Linux (19)

### Infos retournées par ps:

| PID  | TT  | STAT | TIME    | COMMAND |
|------|-----|------|---------|---------|
| 3899 | p1  | S    | 0:00.08 | -zsh    |
| 4743 | p1  | S+   | 0:00.14 | emacs   |
| 4180 | std | S    | 0:00.04 | -zsh    |

numéro de processus

terminal associé

état du processus:

temps CPU utilisé  
commande exécutée

|    |                      |
|----|----------------------|
| R  | actif                |
| T  | bloqué               |
| P  | en attente de page   |
| D  | en attente de disque |
| S  | endormi              |
| IW | swappé               |
| Z  | tué                  |

2009/2010

Systèmes d'Exploitation Evolués

78



## 10. Processus dans Linux (20)

### Communication entre processus

Les processus Linux communiquent avec deux mécanismes :

1. En passant des messages à travers les tubes (pipelines)

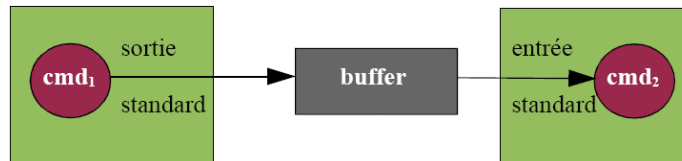


illustration de "**cmd1 | cmd2**"

2009/2010

Systèmes d'Exploitation Evolués

79



## 10. Processus dans Linux (21)

### Communication entre processus

Le système assure la synchronisation de l'ensemble dans le sens où :

- il bloque le processus lecteur du tube lorsque le tube est vide en attendant qu'il se remplisse (s'il y a encore des processus écrivains);
- il bloque (éventuellement) le processus écrivain lorsque le tube est plein (si le lecteur est plus lent que l'écrivain et que le volume des résultats à écrire dans le tube est important).

2009/2010

Systèmes d'Exploitation Evolués

80



## 10. Processus dans Linux (22)

### Communication entre processus

#### 2. En utilisant les interruptions logicielles

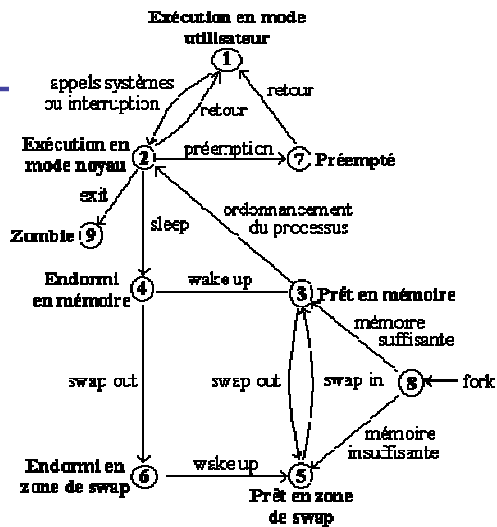
- Un processus envoie un **signal** à un autre;
- Le processus dit au système quoi faire : ignore le signal, prendre le signal, ou bien laisser le signal terminer le processus (par défaut);
- Si le signal est pris, le processus doit indiquer le processus en charge du signal (signal handler);



## 10. Processus dans Linux (23)

### Communication entre processus

- Le signal handler commence à exécuter, et lorsqu'il termine le contrôle retourne au processus qui a pris le signal;
- Un processus peut envoyer un signal uniquement au membre de son groupe (parent, frères, enfants et descendants);
- Le même signal peut être envoyé à tous les membres du groupe du processus.



## 10. Processus dans Linux (25)

### Listes des états d'un processus

1. le processus s'exécute en **mode utilisateur**.
2. le processus s'exécute en **mode noyau**.
3. le processus ne s'exécute pas mais est **éligible** (prêt à s'exécuter) .
4. le processus est **endormi en mémoire centrale** .
5. le processus est prêt mais le swappeur doit le transférer en mémoire centrale pour le rendre éligible.
6. le processus est **endormi en zone de swap** (sur disque par exemple).
7. le processus passe du mode noyau au mode utilisateur mais est préempté et a effectué un changement de contexte pour élire un autre processus.



## 10. Processus dans Linux (26)

### Listes des états d'un processus

8. naissance d'un processus, ce processus n'est pas encore prêt et n'est pas endormi, c'est l'état initial de tous processus sauf le *swappeur*.

9. *zombie* le processus vient de réaliser un exit, il apparaît uniquement dans la table des processus où il est conservé le temps pour son processus père de récupérer le code de retour et d'autres informations de gestion (coût de l'exécution sous forme de temps, et d'utilisation des ressources ).

➔ L'état *zombie* est l'état final des processus, les processus restent dans cet état jusqu'à ce que leur père lise leur valeur de retour (exit status).



## 10. Processus dans Linux (27)

### Liste des signaux Linux/Linux

| Numéro | Nom     | Signification                      |
|--------|---------|------------------------------------|
| 1      | SIGHUP  | Fin de session                     |
| 2      | SIGINT  | Interruption                       |
| 3      | SIGQUIT | Instruction                        |
| 4      | SIGILL  | Instruction illégale               |
| 5      | SIGTRAP | Trace                              |
| 6      | SIGABRT | Instruction IOT ou abort           |
| 7      | SIGEMT  | Instruction EMT                    |
| 8      | SIGFPE  | Exception arithmétique             |
| 9      | SIGKILL | Tuer un processus                  |
| 10     | SIGBUS  | Bus error                          |
| 11     | SIGSEGV | Violation de mémoire               |
| 12     | SIGSYS  | Erreur appel système               |
| 13     | SIGPIPE | Ecriture dans un pipe sans lecteur |
| 14     | SIGALRM | Alarme de l'horloge                |
| 15     | SIGTERM | Signal de terminaison              |



## 11. Implémentation des processus sous Linux (1)

Chaque processus a deux parties :

- Une composante utilisateur qui exécute le programme,
- Une partie noyau.

Le noyau maintient deux structures de données fondamentales relatives aux processus :

- La table de processus : réside en permanence en mémoire et contient des informations nécessaires à tous les processus (même ceux qui ne sont pas en mémoire)
- La structure utilisateur : swappée ou paginée quand le processus associé n'est pas en mémoire (pour économiser la mémoire).



## 11. Implémentation des processus sous Linux (2)

La table de processus contient les informations suivantes :

- Les paramètres d'ordonnement :
  - Priorité du processus, temps CPU récemment consommé, temps récemment passé à dormir.
  - Servent à déterminer le prochain processus à exécuter.
- L'image mémoire :
  - Pointeurs sur les segments de codes (text), de données et de pile du programme et sur les tables des pages (si la pagination existe).
  - Si le segment de page est partagé, le pointeur adresse la table partagée.
  - Si le processus n'est pas en mémoire, on trouve des informations permettant d'en retrouver les éléments sur disque



## 11. Implémentation des processus sous Linux (3)

### ▪ Les signaux :

- Des masques indiquent quels signaux sont ignorés, attrapés, temporairement bloqués et en instance d'émission par le processus.

### ▪ Le reste :

- l'état courant du processus, l'événement éventuel sur lequel il attend, le temps au bout duquel l'horloge de l'alarme aura terminé, le PID du processus, le PID de son parent, le UID et le GID.



## 11. Implémentation des processus sous Linux (4)

La structure utilisateur contient les informations suivantes (nécessaire uniquement quand le processus est en cours d'exécution):

### ▪ Les registres :

- quand une interruption matérielle (trap) arrive au noyau, les registres sont sauvegardés.

### ▪ L'état de l'appel système :

- l'information concernant l'appel système en cours, y compris ses paramètres et le résultat.

### ▪ La table de descripteur de fichiers :

- quand un appel système utilisant un descripteur de fichier survient, ce descripteur sert d'indice dans cette table pour trouver l'i-node relative au fichier concerné.



## 11. Implémentation des processus sous Linux (5)

### ▪ L'accounting :

- Un pointeur sur une table contenant le temps CPU en mode utilisateur et en mode noyau consommé par le processus.
- Certains systèmes définissent une limite pour le temps CPU consommable par un processus, la taille maximale de sa pile, le nombre de cadre de pages qu'il peut utiliser, etc.

### ▪ La pile noyau :

- la pile utilisé par le processus lorsqu'il s'exécute en mode noyau.



## 11. Implémentation des processus sous Linux (6)

### L'ordonnancement sous Linux

Il s'agit d'un algorithme d'ordonnancement à deux niveaux :

### ▪ L'algorithme de haut niveau (**swapper**) :

- déplace le processus entre mémoire et disque afin de permettre à tous d'avoir une chance d'être exécuter;
- utilise plusieurs files d'attente, dont chacune correspond à un niveau de priorité;
- un processus s'exécutant en mode utilisateur a une priorité positive;
- un processus s'exécutant en mode noyau a une priorité négative;
- seuls les processus en mémoire et prêts à l'exécution sont placés dans ces files (un processus bloqué sera enlevé de sa file);



## 11. Implémentation des processus sous Linux (7)

### L'ordonnancement sous Linux

- L'algorithme de haut niveau (suite) :
  - un processus sélectionné sera exécuté pendant au plus un quantum ou se bloquer avant.
    - ✓ dans le premier cas, il est replacé en fin de file et l'ordonnanceur recommence.
  - les processus de même classe de priorité partagent la CPU suivant un algorithme de type tourniquet (Round Robin);
  - quand un processus utilise son quantum, il sera déplacé vers la fin de sa file;
  - chaque seconde, la priorité de chaque processus est mise à jour à l'aide de la formule :

$$\text{priorité} = \text{CPU-usage} + \text{nice} + \text{base}$$



## 11. Implémentation des processus sous Linux (8)

### L'ordonnancement sous Linux

- L'algorithme de bas niveau :
  - sélectionne le processus à exécuter dans l'ensemble des processus en mémoire et prêts à l'exécution;
  - quand il s'exécute, il consulte les files en partant de la plus prioritaire jusqu'à en trouver une non vide;
  - le premier processus de cette file est sélectionné et démarre



## 12. La gestion de la mémoire sous Linux (1)

Tout processus Linux est associé à un espace d'adressage en trois zones :

- Le segment de texte : instructions exécutables du programme (lecture seule)
- Le segment de données (initialisées ou non initialisées BBS) : variables, vecteurs, etc. peut être changé, peut changer de taille (appel bib. C : malloc)
- La pile (stack): contient a début de l'exécution du programme toutes les variables de l'environnement (shell) et la commande tapée pour lancer le programme



## 12. La gestion de la mémoire sous Linux (2)

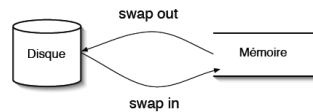
- Pour partager un programme (éditeur de texte) entre plusieurs utilisateurs, quelques version Linux ont la notion de segment de texte partagé (shared text segment)
- Data (Le segment de données ) et stack ne sont jamais partagées
- Version avant 3BSD (Berkley Software Distribution) : Swapping
- Version après 3BSD : Paging





## 12. La gestion de la mémoire sous Linux (3)

### Fonctionnement du Swapper



**Swap out** quand manque d'espace dans la mémoire

- fork, brk, ou débordement de pile
- choix d'une victime :
  1.  $\exists$  processus bloqués : critère **C** = prio + temps résidence  
victime = le plus grand **C**
  2. sinon : parmi les non bloqués, victime = plus grand **C**

2009/2010

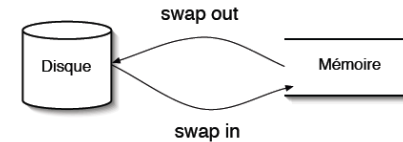
Systèmes d'Exploitation Evolués

97



## 12. La gestion de la mémoire sous Linux (4)

### Fonctionnement du Swapper



**Swap in** : toutes les 4-5 secondes

- Swapper cherche un processus prêt sur le disque
- critère de choix : temps le plus long sur le disque
- éventuellement swap out d'un autre processus

2009/2010

Systèmes d'Exploitation Evolués

98



## 12. La gestion de la mémoire sous Linux (5)

### Fonctionnement du Swapper

- Swapper : swap in (toutes les 4-5 secs) + swap out (à la demande)
  - Changement de comportement quand :
    - ✓ plus de processus swappés
    - ✓ trop de processus en mémoire (**Thrashing** : S'il y a trop de défauts de page, un processus peut passer plus de temps en attente de pagination qu'en exécution )
- min. de 2 sec. en mémoire avant de swapper un processus

2009/2010

Systèmes d'Exploitation Evolués

99



## 12. La gestion de la mémoire sous Linux (6)

### Fonction de pagination

- Toutes les 250 msec. le **daemon** (démon) est activé
- Algorithme :
  - ✓ nombre de cadres libres  $\geq$  lotsfree? (paramètre du système, typiquement le 1/4 de la mémoire)
    - si oui : sleep
    - sinon : transférer des pages sur le disque
  - but** : avoir des pages libres constamment
- Algorithme de traitement de défaut basé : sur "seconde chance" (ou "Clock Algorithm")

2009/2010

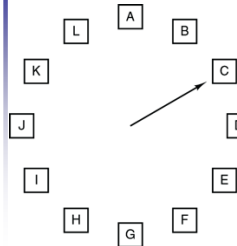
Systèmes d'Exploitation Evolués

100



## 12. La gestion de la mémoire sous Linux (7)

### Clock Algorithm classique



Quand un défaut de page se produit, la page pointée est testée. L'action entreprise dépend de la valeur du bit R :  
R = 0 : retirer la page  
R = 1 : mettre R à 0 et avancer le pointeur

⊗ les 2 passes prennent trop de temps

Les cadres sont examinés selon une liste circulaire (contour de l'horloge) :

- passage 1 : lorsqu'une page est pointée par l'aiguille de l'horloge, son bit d'usage est mis à 0 (cleared)
- passage 2 : une page non accédée aura toujours son bit d'usage 0 et donc sera libérée (mise sur la liste des cadres libres sans être effacée)

2009/2010

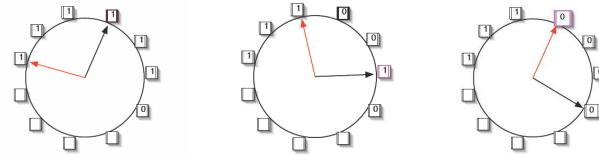
Systèmes d'Exploitation Evolués

101



## 12. La gestion de la mémoire sous Linux (8)

### Two-Handed Clock Algorithm



- garde 2 aiguilles,
- au début, met à 0 le bit d'usage du cadre pointé par l'aiguille en avant,
- puis vérifie le bit d'usage du cadre pointé par l'aiguille en arrière,
- avance les deux aiguilles

2009/2010

Systèmes d'Exploitation Evolués

102



## 12. La gestion de la mémoire sous Linux (10)

### Two-Handed Clock Algorithm

- ❑ Chaque fois que le démon est réveillé, les 2 aiguilles sont avancées moins d'un tour de l'horloge (selon le nombre de cadre à libérer pour atteindre *lotsfree*),
- ❑ Si les 2 aiguilles sont gardées proches l'une de l'autre, uniquement les pages les plus utilisées auront la chance d'être accédées entre le passage de l'aiguille avant et de l'aiguille arrière,
- ❑ Si les deux aiguilles sont gardées à 359° à part, on retrouve le Clock Algorithm classique

2009/2010

Systèmes d'Exploitation Evolués

103



## 12. La gestion de la mémoire sous Linux (11)

### Thrashing

Si le **taux de défaut** de pages est très grand : appel au **swapper**

- si ∃ des processus idle depuis  $\geq 20$ sec.  
**swap out le moins actif** récemment (max. idle time)
- sinon parmi **les 4 plus gros processus** (taille mémoire)  
**swap out le plus vieux** (en mémoire)
- si nécessaire plusieurs processus peuvent être swappés

2009/2010

Systèmes d'Exploitation Evolués

104



### 13. Entrées/Sorties dans Linux (1)

- Intégrées en tant que fichiers spéciaux (special files) dans le répertoire de fichiers (file system)
- Ayant un chemin et accédés comme tout autre fichier ( /dev/tty1, /dev/fd0, /dev/hda1, etc.)
- Deux types de périphériques (fichiers spéciaux) :
  - ✓ périphériques blocs: séquence de blocs à accès aléatoires (les disques et les bandes)
  - ✓ périphériques caractère : séquences de caractères (terminaux, imprimantes)
- Chaque fichier est décrit par un inode et est accessible indifféremment des autre à travers l'interface du SGF.

