

Systèmes d'exploitation (Operating Systems)

Le cas Linux

Pr.YAKINE



½ Élément de module : Système exploitation Linux

1. Introduction : Linux ou Unix ?
2. Initiation au shell
3. Le système de fichiers Linux
4. Commandes de base
5. Les Filtres
6. Les redirections < > >>
7. Les métacaractères / motifs d'englobement /jokers
8. Les tubes
9. Les liens
10. Gestion des processus

Linux ou Unix ?

L'*histoire*...

Linux est le petit fils de Unix

Voir « *Unix history* »

Unix ou GNU/Linux ?

- 1979 : Première version d'Unix commercialisée
 - ◆ Unix Système V
 - ◆ Puis Unix BSD par l'université de Berkeley
- 1992 : Sortie de Solaris
 - ◆ Dérivée de Unix Système V
 - ◆ L'Unix de Sun
- Unix est un système payant et non libre
- L'alternative est le projet **GNU/Linux**
 - ◆ GNU pour (GNU is Not Unix)
 - ◆ Linux crée à l'origine par le finlandais Linus Torvald
 - ◆ C'est un système sous licence **GPL** (General Public Licence)
 - ◆ Il existe beaucoup de « **distributions** » Linux
 - Debian, Red Hat, Mandriva, Ubuntu...



Caractéristiques de GNU/linux

- Efficace.
 - ◆ Contrairement à des systèmes beaucoup plus répandus, il n'utilise pour ses besoins propres que très peu de ressources. Les logiciels que vous utilisez pour votre travail disposent donc de beaucoup plus de puissance pour fonctionner.
- Fiable.
 - ◆ Une machine sous Linux fonctionne 24h/24 si besoin sans se plaindre (si le matériel est prévu pour, en particulier au niveau thermique).
- Robuste.
 - ◆ Une erreur d'un utilisateur ou un ``plantage'' éventuel d'une application n'affectent pas le reste du système. D'autre part, il est exceptionnel de devoir l'arrêter : la quasi-totalité des opérations de configuration, mise au point, etc..., ne nécessitent pas l'arrêt du système.
- Linux est conforme à la norme POSIX et aux standards du marché, en particulier de l'Internet.
 - ◆ Cela signifie que qu'un logiciel conçu pour un autre système de la même famille (Solaris de SUN, Digital Unix, AIX d'IBM, SCO Unix...) peut être rapidement porté sous Linux et vice-versa, ce qui assure une protection de l'investissement logiciel en cas d'obligation de changement de système.

Caractéristiques de GNU/linux

- Linux est donc un **SE performant** donnant satisfaction aussi bien sur des machines anciennes ou bas de gamme que sur des machines puissantes très sollicitées ou devant remplir des fonctions importantes.

Pour quelle plate-forme matérielle ?

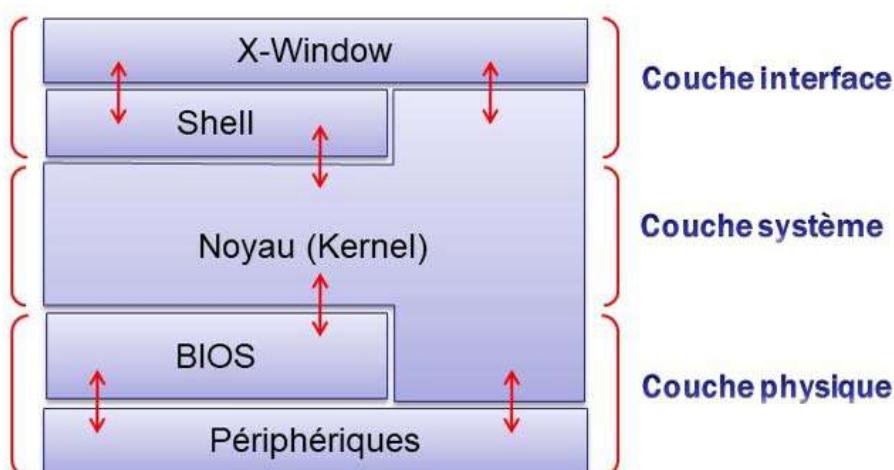
- Linux existe pour la plus majeure partie des plates-formes matérielles :
 - ◆ Pour l'architecture Intel i386 et AMD des ordinateurs PC
 - ◆ Sun Sparc
 - ◆ ARM
 - ◆ RISC
 - ◆ ...
- Le noyau prend en charge une grande variété de périphériques matériels
 - ◆ Cartes de communication Wifi, Bluetooth, Zigbee, ...
 - ◆ Support des cartes vidéos orienté « chipset »

Noyau et distribution

- Linux est architecturé autour d'un noyau
 - ◆ Ce noyau est appelé « Kernel »
 - ◆ Il contient toutes les fonctions de base d'un OS
 - Accès aux périphériques matériels standards
 - Disque dur, carte graphique, ...
 - Accès aux périphériques spécifiques
 - A l'aide de pilotes
 - Gère les processus et la communication entre les processus
 - Gère la mémoire et les fichiers.
- Linux est un système multitâche préemptif
 - ◆ Le noyau gère l'exécution de chaque processus
 - Le processus peut être interrompu à tout moment.

Architecture GNU/Linux

- Divisée en 3 couches distinctes
 - ◆ La couche physique : Périphériques et BIOS
 - ◆ La couche système : Gérée par le noyau
 - ◆ La couche interface : le Shell et/ou le système X-Window



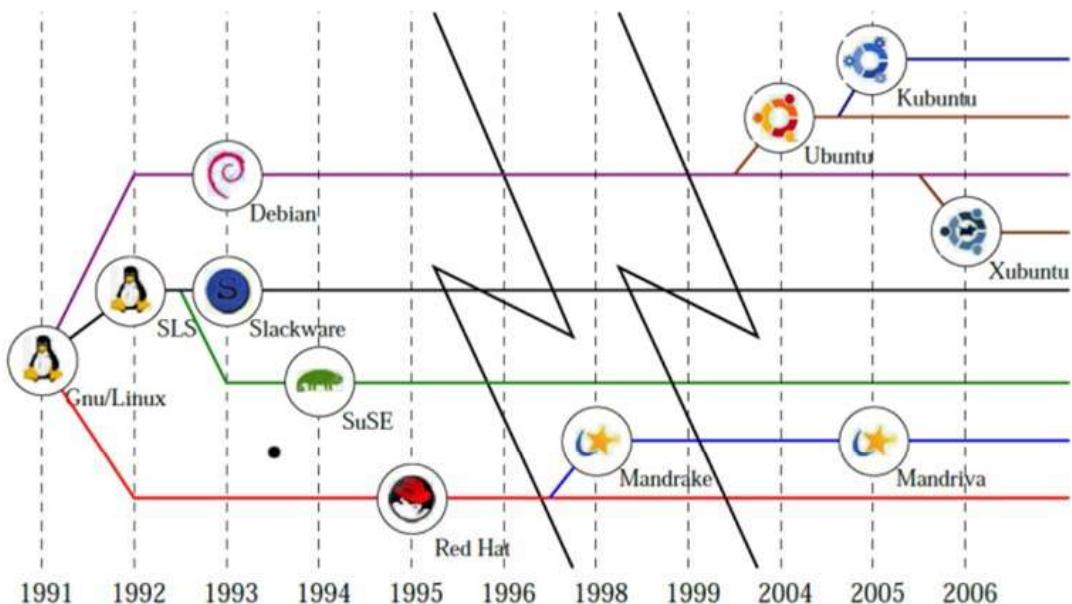
Linux sous licence GPL

- La licence GPL : General Public Licence
 - ◆ Concerne les modalités de distribution du noyau
 - ◆ Le code source est ouvert (Open Source)
 - ◆ Chacun peut le modifier et le revendre
 - Le code source modifié doit alors rester sous licence GPL
- Le noyau Linux est sous licence GPL
- Une distribution Linux est un ensemble noyau + logiciels sous licence GPL ou mixte
- L'utilisation et la copie de Linux sont gratuites
 - ◆ Certaines distributions contiennent des logiciels non GPL
 - ◆ Exemple de distrib. 100% GPL : Debian
 - ◆ Exemple de distrib. Mixte : Red Hat
- Le noyau reste entièrement sous GPL

Les distributions Linux

- Une distribution est constituée :
 - ◆ Du noyau Linux
 - ◆ De « packages » contenant des logiciels additionnels
- Certaines distributions sont spécifiques à un domaine particulier
 - ◆ Sécurité des réseaux (IPCOP, PFsense)
 - ◆ Piratage (Backtrack)
 - ◆ Systèmes embarqués ou temps réel (LynxOS)
 - ◆ Pour les enfants (DoudouLinux)
 - ◆ Pédagogique (EduBuntu)
- Pour les distributions généralistes, la liste est longue
 - ◆ Debian, Red Hat ou Cent OS, Mandriva, Suse
- Il existe des distributions basées sur d'autres distrib.
 - ◆ Exemple : Ubuntu basée sur une Debian

Distributions linux



La gestion des utilisateurs

- Linux est un OS multi-utilisateur
 - ◆ Chaque utilisateur dispose de son environnement de travail
 - Un répertoire « home »
 - Un bureau (Si X-Window)



Que faut-il pour réussir sous Linux ?

- Maîtriser le shell
 - ◆ Un serveur sous Linux n'a pas toujours d'environnement X-Window (KDE ou Gnome)
 - ◆ Les commandes de base permettent beaucoup de choses
 - ◆ Il faut savoir utiliser le manuel des commandes (man)
- Bien connaître le système de fichiers
 - ◆ Sous Linux « tout est fichier » (même les périphériques et les processus)
 - ◆ Configurer un logiciel passe souvent par l'édition d'un simple fichier texte
- Savoir lire les fichiers de journalisation
 - ◆ Lorsqu'un logiciel ne fonctionne pas, il laisse des traces.
 - ◆ Fichiers « log » stockés dans le « file system »
- **LINUX est sensible à la casse (case sensitive)**
 - ◆ Majuscules et minuscules sont interprétés différemment

Initiation au SHELL

Shell, c'est quoi exactement ?

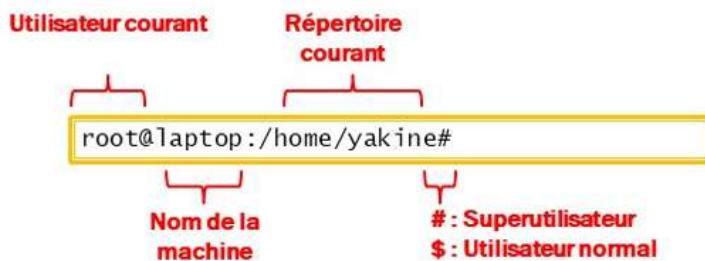
- Il s'agit d'une interface texte entre l'utilisateur et le système
 - ◆ Tout se fait au clavier
 - ◆ Pas de clic de souris
- L'utilisateur tape des commandes qui sont exécutées par le système
 - ◆ Le shell est donc un « interpréteur de commandes »
 - ◆ Chaque commande a une syntaxe particulière
 - ◆ Il existe des milliers de commandes différentes
- Les commandes peuvent aussi provenir d'un fichier
 - ◆ Le fichier contient les commandes à exécuter
 - ◆ L'utilisateur appelle le fichier plutôt que de taper toutes les commandes
 - Utile pour les tâches répétitives
- Le shell reste le moyen le plus efficace pour contrôler le système
 - ◆ C'est aussi le plus utilisé sous Linux/Unix

Les différents types de shell

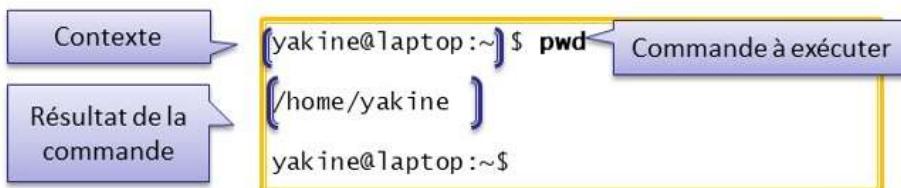
- Il existe plusieurs types de shell
 - ◆ Bourne shell
 - ◆ Korn Shell
 - ◆ Bash (Bourne again shell)
 - ◆ Tcsh (Terminal C shell)
 - ◆ L'interpréteur de commande MS-DOS (Sous Windows)
 - ◆ PowerShell (Windows 2008 server)
- Sous Linux, on peut choisir son shell
 - ◆ Le shell bash domine le marché actuellement

Présentation de la ligne de commande (CLI)

- La ligne de commande se présente sous forme de texte ayant la signification suivante :



- Il suffit alors de taper une commande pour qu'elle soit exécutée
- Une commande peut être appelée :
 - En tapant son nom puis des arguments ou paramètres
 - Exemple permettant d'afficher le répertoire courant.



Syntaxe d'une commande

- Chaque commande a une syntaxe particulière
 - Elle est composée d'**options** et de **paramètres (arguments)**
 - Les paramètres permettent de fournir les données nécessaires à l'exécution de la commande
 - Ils sont souvent obligatoires
 - Les options permettent d'offrir des fonctionnalités supplémentaires qui s'adaptent à des besoins spécifiques
- Si vous entrez une commande en utilisant la syntaxe correcte, Linux exécute la commande. Sinon, vous recevez un message qui signifie qu'il ne peut pas l'interpréter.

```
commande [-OPTION] argument1 argument2 ...
```

Syntaxe d'une commande

- Exemples de syntaxe (SYNOPSIS de la commande « cp »)



- Les options sont précédées de signe « - » :

```
cp -r /home/yakine/rep rep_copie
```

Exemple de la commande « ls »

- « ls » liste les fichiers d'un répertoire donné
- SYNOPSIS :
- Quelques options utiles (Attention à la casse) :
 - ◆ a : Liste les fichiers cachés
 - ◆ l : Listing long (Plus d'infos sur les fichiers)
 - ◆ S : Classement par taille de fichiers
 - ◆ t : Classement par date de modification
- Exemples :
 - ◆ Listing long de /etc avec répertoires cachés et fichiers plus gros en 1^{er}
 - ◆ Listing long avec fichiers plus récents en 1^{er} :

```
ls -al /etc
```

```
ls -lt
```

man : Aidez-moi s.v.p

- La commande « man » (Manual) permet d'obtenir de l'aide sur la syntaxe d'une commande
 - ◆ Très souvent en Anglais et rarement en français
 - ◆ L'aide est en générale très complète
 - Parfois difficile de trouver l'information manquante
- Une page de manuel se décompose en plusieurs parties distinctes
 - ◆ NAME : Nom et description rapide de la commande
 - ◆ SYNOPSIS : Syntaxe(s) de la commande
 - ◆ DESCRIPTION : Description complète de la commande
 - ◆ OPTIONS : La description des options
 - ◆ AUTHOR : Un mot sur l'auteur
 - ◆ BUGS : bugs connus
 - ◆ SEE ALSO : Autres commandes connexes à consulter également
 - ◆ ... (Dépend des commandes)

man : Aidez-moi s.v.p (2)

- Syntaxe :
- `man commande`
- Exemple : Obtenir l'aide en ligne pour la commande « cp »

```
~# man cp
```

CP(1)

User Commands

CP(1)

NAME

`cp - copy files and directories`

SYNOPSIS

```
cp [OPTION]... [-T] SOURCE DEST
cp [OPTION]... SOURCE... DIRECTORY
cp [OPTION]... -t DIRECTORY SOURCE...
```

DESCRIPTION

`Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.`

`Mandatory arguments to long options are mandatory for short options too...`

Le système de fichiers LINUX

Système de Gestion de Fichiers

- Organisation logique (vue de l'utilisateur)
 - ◆ Structure arborescente
 - Répertoires, fichiers
- Organisation physique (vue du concepteur)
 - ◆ i-nœud (inode)

Organisation logique du SGF

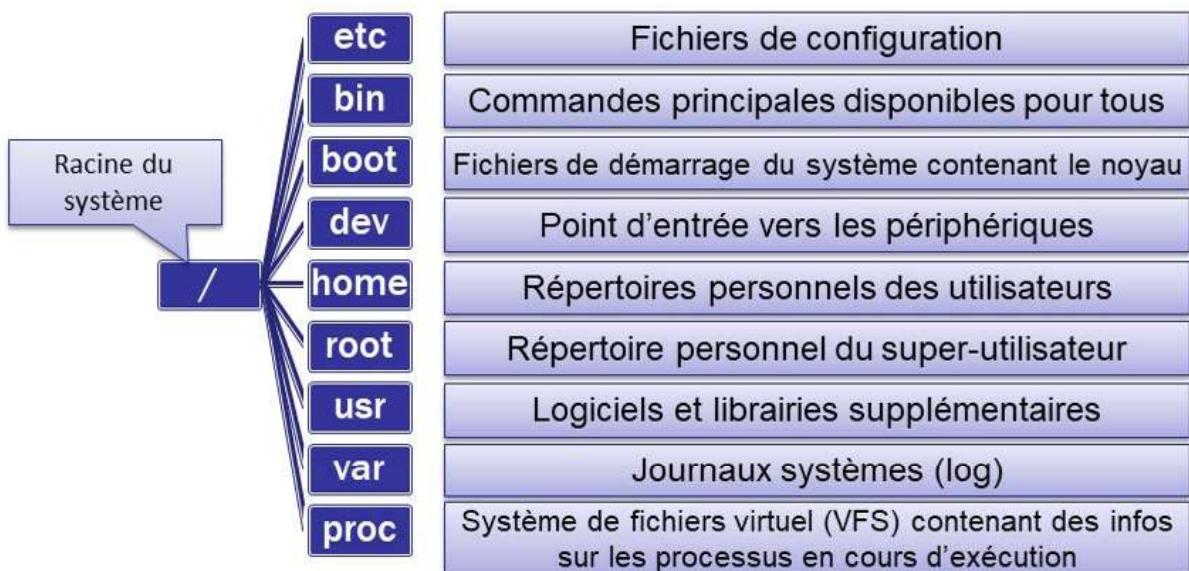
- Répertoire
 - ◆ nœud contenant
 - Des fichiers
 - Des sous répertoires
 - ◆ Répertoire racine nommé : /
 - ◆ Répertoire père : ..
 - ◆ Répertoire courant : .

Organisation logique du SGF

- Dans le cas de Linux, chaque type de fichiers est désigné par un symbole :
 - «-» les fichiers ordinaires,
 - «d » les répertoires,
 - «b » les périphériques blocs, et «c » les périphériques caractère,
 - «p » les tubes nommés (named pipe).
 - «l» les liens symboliques

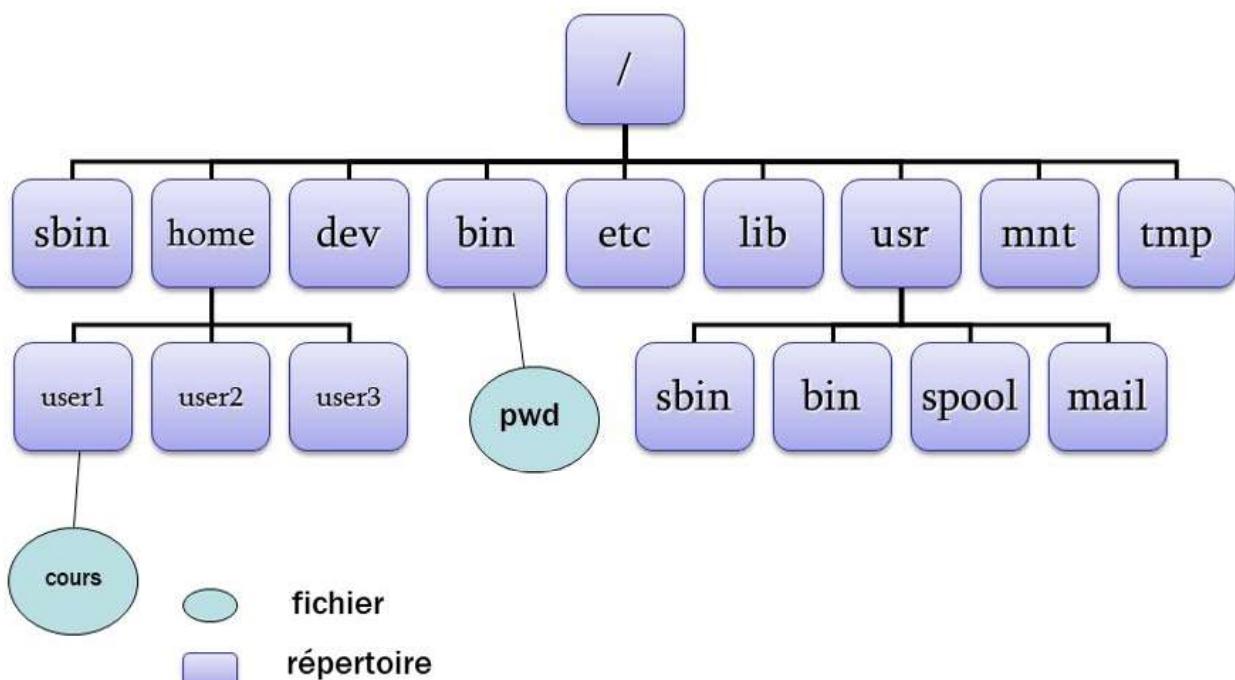
Arborescence Linux

- Voici l'arborescence typique d'un système Linux :



Organisation logique du SGF

- Voici l'arborescence typique d'un système Linux :



Les symboles associés à l'arborescence

- Différents symboles sont utilisés pour désigner des répertoires
 - ◆ Le « . » : Répertoire courant
 - ◆ Le « .. » : Répertoire parent
 - ◆ Le « ~ » : Répertoire personnel de l'utilisateur courant
- La commande « cd » permet de changer de répertoire
- La commande « ls » permet de lister un répertoire
- La commande « pwd » permet de connaître le répertoire courant

Exemples :



Comment se repérer dans le système de fichiers ?

- La ligne de commande donne des informations :



- Autre exemple



Chemin relatif et absolu

- Il existe 2 méthodes pour spécifier un chemin dans le système de fichiers
 - ◆ Chemin absolu : Débute à la racine du système (« / »)
 - Exemple : /etc/firefox
 - ◆ Chemin relatif : Dépend du répertoire courant
 - Le répertoire Courant est celui dans lequel on se situe
 - Pas de signe « / »
 - Exemple : Documents/rapports

Chemin relatif et absolu

- Exemples:

relatif	absolu
root@laptop:/home/ali# cd Documents	root@laptop:/home/ali# cd /home/ali/Documents
root@laptop:/home/sara# cd Documents root@laptop:/home/sara/Documents/#	? root@laptop:/home/sara# cd /home/ali/Documents root@laptop:/home/ali/Documents/#
root@laptop:/etc/apache# cd .. root@laptop:/etc/#	? root@laptop:/etc/apache# cd /etc root@laptop:/etc/#
ali@laptop:~ \$ cd Docs ali@laptop:~/Docs\$? sara@laptop:/etc\$ cd /home/ali/Docs sara@laptop :/home/ali/Docs \$

Exercices

- Dans quel répertoire je suis situé ?

```
sara@laptop:~/Documents/rep1$
```

◆ Réponse :

- Par quel chemin relatif équivalent peut-on remplacer celui-ci ?

```
ali@laptop:/home$ cd /home/sara/linux/
```

◆ Réponse :

- Que m'indiquera le résultat de la commande suivante ?

```
sara@laptop:~/Documents/java$ pwd
```

◆ Réponse :

- Commande la plus courte possible pour revenir dans le répertoire rep2, situé à la racine de mon répertoire « home » ?

```
ali@laptop:~/rep1/sousrep1$
```

◆ Réponse :

Les droits des fichiers et répertoires

- Linux est un système multi-utilisateurs
 - ◆ Plusieurs utilisateurs se partagent l'espace disque,
 - ◆ Les fichiers et répertoires d'un utilisateur ne doivent pas être accessibles par les autres,
 - ◆ Les fichiers de configuration du système doivent être protégés.
- Nécessité de spécifier des droits pour chaque fichier/répertoire
 - ◆ Plusieurs types de droits : Lecture (R), écriture (W), exécution (X)
 - ◆ Ces droits s'appliquent pour 3 groupes d'utilisateurs :
 - Le propriétaire (user) du fichier
 - Le groupe (group) propriétaire (Tous les utilisateurs membre du groupe)
 - Les autres (others). Désigne tous les utilisateurs non membres des 2 précédents
- Les droits sont responsables d'un grand nombre d'erreurs de configuration

Droits : Différence entre fichiers et répertoires

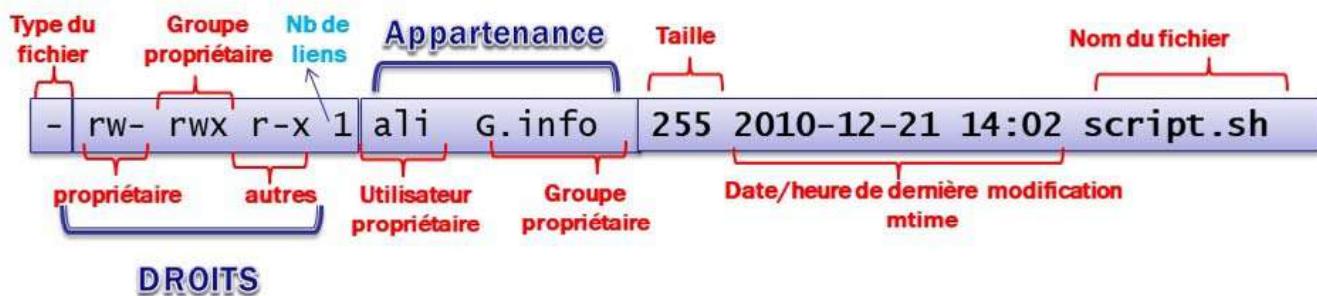
- Nous avons vu qu'il existe 3 types de droits : r, w et x
- Ces droits n'ont pas la même signification pour un fichier que pour un répertoire
- Pour un fichier :
 - ◆ r : Lecture (afficher)
 - ◆ w : Ecriture (modification)
 - ◆ x : Exécution (exécution d'un script)
- Pour un répertoire
 - ◆ r : Lire le contenu, lister les fichiers (avec ls par exemple)
 - ◆ w : Modifier le contenu, créer et supprimer des fichiers (avec les commandes « cp », « mv », « rm »)
 - ◆ x : d'accéder aux fichiers du répertoire. Mais aussi de naviguer dans les sous-répertoires (avec « cd »)
 - ◆ En général, lorsque le droit w est accordé, le droit x l'est aussi

Les droits sur les fichiers et répertoires

- La commande « ls -l » permet d'afficher les droits qui s'appliquent

```
root@laptop:/home/ali/Documents# ls -l
total 20
-rw-r--r-- 1 ali G.info      0 2011-12-21 14:42 projet.txt
-rw-rw-r-x 1 ali G.info  7406 2011-12-21 14:44 rapport.ods
-rw-rw-r-- 1 ali G.info  7363 2011-12-21 14:44 rapport-activite.odt
-rw-rwxr-x 1 ali G.info 255 2011-09-15 14:02 script.sh
```

- Signification des différents champs



Propriétaire : Lecture, écriture
 Groupe : Lecture, écriture et exécution
 Autres : Lecture et exécution

Modifier les droits avec « chmod »

- La commande « chmod » permet de modifier les droits :

 - 2 syntaxes différentes

```
chmod [OPTION]... MODE [,MODE]... FILE...
```

```
chmod [OPTION]... OCTAL-MODE FILE...
```

 - Mode symbolique :

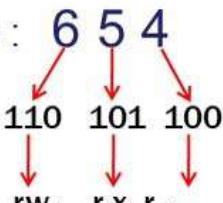
 - Basé sur des symboles (ugoa) et des opérateurs (+,-,=)
 - u** (user), **g** (group), **o** (others), **a** (all users)
 - +** (Ajouter le droit), **-** (Retirer le droit), **=** (Ajouter le droit et retirer tous les autres)
 - Exemple (Ajoute le droit d'exécution au propriétaire) :

```
chmod u+x rapport.txt
```

 - Mode octal :

 - Basé sur des nombres de 0 à 7
 - A chaque bit de la traduction binaire correspond un droit
 - Exemple (rw- rw- r--) : `chmod 664 rapport.txt`

Mode octal de « chmod »

- Les droits sont représentés par un nombre octal (Base 8)
 - De 0 à 7
- La représentation binaire (base 2) donne le détail des droits
 - Exemple : **6 5 4**


Propriétaire	: Lecture, écriture
Groupe	: Lecture et exécution
Autres	: Lecture seulement
- Ce mode permet de modifier tous les droits en même temps
 - A utiliser avec précaution
 - Très efficace pour s'assurer que tous les fichiers ont les mêmes droits
 - Utilisé pour sécuriser les accès des utilisateurs aux fichiers

Exemples d'utilisation de « chmod »

- Retirer le droit d'écriture au propriétaire et au groupe
- Positionner les droits en « rwx r-x --- »
- Ajouter le droit de lecture aux autres de tous les fichiers de sara
- Donner tous les droits à tout le monde sur le fichier secret de sara (déconseillé !)

Exercices

- Quel(s) utilisateur(s) pourra(ont) se déplacer dans le répertoire suivant ?

```
drwxr-x--- 26 sara ENSA 4096 2011-12-21 16:11 rep
```

- Qui pourra créer de nouveaux fichiers dans ce répertoire ?

```
drwxr-xrwx 26 ali ENSA 4096 2011-12-21 16:11 bilans
```

- Soit le fichier suivant :

```
-rwxr--r-- 26 ali G.info 25140 2011-12-21 16:11 rapport.txt
```

- ♦ Situé dans le répertoire suivant :

```
drwxrwxrwx 26 ali G.info 4096 2011-12-21 16:11 rapports
```

- ♦ Qui pourra effacer ce fichier ?

Remarques sur les droits

- Le droit « w » accordé à un répertoire permet :
 - ◆ D'y effacer des fichiers quels que soient le propriétaire et les droits qui s'appliquent à ces fichiers
 - ◆ Quand il est donné à un groupe, n'importe quel utilisateur de ce groupe peut supprimer des fichiers (**dangereux !**)
- Les droits ne s'appliquent pas au « super-utilisateur »
 - ◆ Il a tous les droits sur tout le système de fichiers
 - ◆ C'est une très grande responsabilité puisque sous Linux tout ou presque repose sur les fichiers
 - La tendance évolue vers une utilisation très modérée voire interdite du compte « root »
- Le droit « x » accordé à un répertoire est un préalable indispensable pour exercer des droits sur les fichiers contenus
- L'utilisateur qui crée un fichier en devient le propriétaire.

Gestion de l'espace de stockage

Consulter le manuel des commandes

df

du

Commandes de base

Manipulation des fichiers et répertoires

- **ls (list)**
 - ◆ Affiche le contenu du répertoire courant ou de celui passé en paramètre
 - ◆ SYNOPSIS : **ls [OPTION]... [FILE]...**
 - ◆ Options

-l	affiche les informations complètes des fichiers et sous répertoires
-a	affiche les fichiers cachés
-R	affichage récursif
-i	affiche le descripteur des fichiers (i-nod)

Manipulation des fichiers et répertoires

- **pwd** (print working directory)
 - ◆ SYNOPSIS : `pwd`
 - ◆ Affiche le chemin du répertoire courant

- **cd** (change the working directory)
 - ◆ SYNOPSIS : `cd chemin`
 - ◆ Se déplace vers le répertoire identifié par 'chemin'
 - ◆ Exemples


```
$ cd /home/ali/doc
$ cd ../ali/doc
```

Manipulation des fichiers et répertoires

- **mkdir** (make directory)
 - ◆ Crée un nouveau répertoire
 - Dans le répertoire courant
 - Exemple


```
$ mkdir cours
```
 - Dans le chemin indiqué en paramètre
 - Exemple


```
$ mkdir /home/ali/cours
$ mkdir ../ali/cours
```

Manipulation des fichiers et répertoires

- **rmdir** repertoire (remove directory)
 - ◆ Détruit un répertoire vide
 - ◆ Exemple
 - \$ `rmdir /home/yakine/temp`

Manipulation des fichiers et répertoires

- **rm** fichier (rm remove)
 - ◆ Détruit un fichier ou un répertoire non vide
 - ◆ Options
 - -r : la commande détruit de manière récursive toute la sous arborescence du répertoire
 - -i : demande la confirmation avant de supprimer le fichier
 - ◆ Exemples
 - \$ `rm -r rep/`
 - \$ `rm -i /home/sara/linux.pdf`

Manipulation des fichiers et répertoires

- **cp (copy)**

- ◆ La commande « cp » copie des fichiers source vers une ou plusieurs destinations

```
cp [OPTION]... SOURCE... DIRECTORY
```

- **mv (move)**

- ◆ La commande « mv » déplace ou renomme une source vers une destination.

```
mv [OPTION]... SOURCE... DIRECTORY
```

- **Exemples**

- \$ cp fichier docs/fichier1
- \$ cp fichier docs
- \$ mv fichier fichcopie

Manipulation des fichiers et répertoires

- **cat fichier [fichier,...]**

- ◆ Concatène et affiche sur la sortie standard le(s) fichier(s) en paramètre

- ◆ [Exemple](#)

- \$ cat fichier1
- \$ cat fichier1 fichier2

Les filtres

Les filtres

- Définition
 - ◆ Un filtre est une commande qui lit les données sur l'entrée standard, les traite et les écrit sur la sortie standard.

Les filtres

▪ more fichier

- ◆ Affiche le contenu du fichier page par page
- ◆ Utilisée pour les fichiers longs (contenant plusieurs pages)
 - Q : quitte la commande
 - Return : saute de ligne
 - Espace : saute de page
- ◆ Exemple
 - \$ more lettre

Les filtres

▪ head [-c nchar -n nline] fichier

- ◆ Affiche le début du fichier
 - Par défaut les dix premières lignes
 - -c nchar : affiche les nchar premiers caractères du fichiers
 - -n nline : affiche les nline premières lignes du fichier
- ◆ Exemple
 - \$ head lettre
 - \$head -c 280 lettre
 - \$ head -n 5 lettre

Les filtres

- **tail [-c nchar -n nline] fichier**
 - ◆ Affiche la fin du fichier
 - Par défaut les dix dernières lignes
 - -n nchar : affiche les nchar derniers caractères du fichiers
 - -n nline : affiche les nline dernières lignes du fichier
 - ◆ Exemple
 - `$ tail lettre`
 - `$tail -n 6 lettre`
 - `$ tail -c 6 lettre`

Les filtres

- **wc [-lwc] fichier**
 - ◆ Compte le nombre de
 - `-l` : lignes
 - `-w` : mots
 - `-c` : caractères
 - du fichier
 - ◆ Par défaut les trois
 - ◆ Exemple

```
$ wc lettre
```

Chercher avec: find

- Recherche multicritères
 - ◆ Par la date, la taille, le nom, ...
- Syntaxe :


```
find [-H] [-L] [-P] [path...] [expression]
```

 - ◆ « path » : Chemin où chercher
 - ◆ « expression » : Expression permettant de définir des critères de recherche
- L'expression est construite autour d'options et de tests
 - ◆ Exemple d'option : « maxdepth » limite la profondeur de recherche
 - ◆ Exemple de test : « name » recherche par le nom du fichier
- Rechercher les vidéos mpeg dont la taille est supérieure à 10Mo



La commande : « grep »

- Commande puissante permettant de filtrer les lignes d'une sortie de texte
 - ◆ Affiche uniquement les lignes correspondant aux critères de filtre
 - ◆ Très utilisée pour rechercher à l'intérieur des fichiers
- Syntaxes :

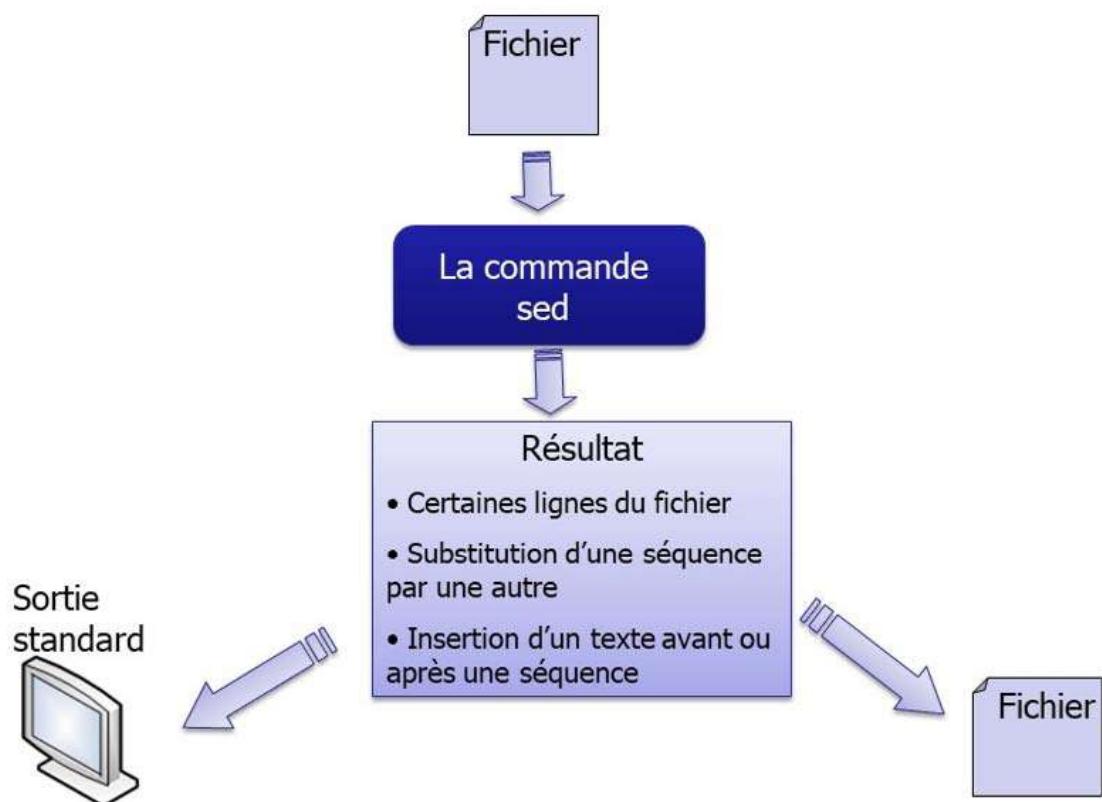

```
grep [OPTIONS] Motif [FILE...]
```

 - ◆ « File » : Fichier ou répertoire où est débutée la recherche
- Options intéressantes
 - ◆ -r : Permet de recherche dans les sous-répertoires (Peut-être long)
 - ◆ -n : Connaître le n° de la ligne et donc la position de l'occurrence trouvée dans le fichier
- Exemple
 - ◆ `$ grep linux /home/yakine/cours.pdf`

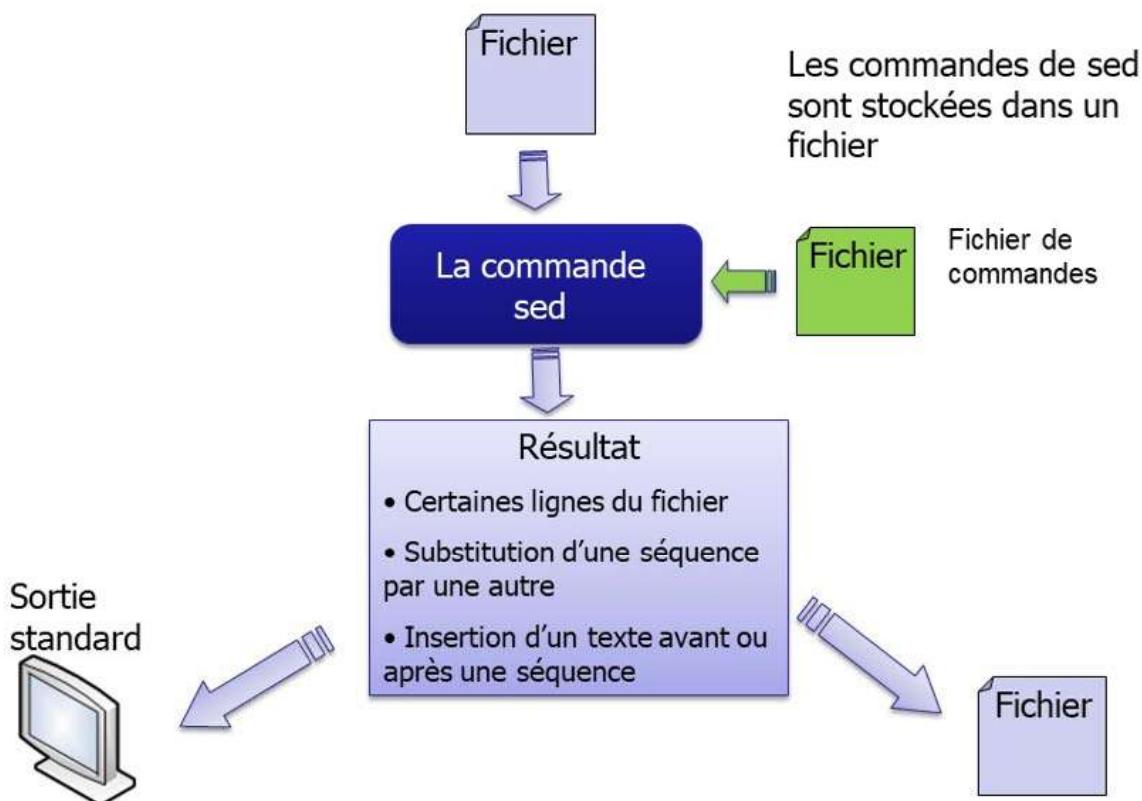
Commande sed

- stream editor
 - ◆ Editeur de texte non interactif
 - ◆ En utilisant les expressions régulières permet de
 - Sélectionner une partie d'un fichier
 - Appliquer des transformations sur un fichier
 - Afficher le résultat sur la sortie standard

La commande sed



La commande sed



La commande sed

■ Syntaxe

- ◆ `sed [-n] [-e commande] [-f fichier_commandes] [fichier]`
 - `-n` écrit seulement les lignes contenant une séquence
 - `-e` spécifie les commandes à appliquer au fichier, `-e` est optionnelle en cas d'une seule commande
 - `-f` les commandes sont lues à partir d'un fichier

La commande sed

■ Selection

- ◆ `sed -n "/motif/p" fichier`

- Affiche sur la sortie standard les lignes du fichier contenant motif

Exemple :

`sed -n "/linux/p" fichier` : affiche les lignes contenant 'linux'

- ◆ `sed -n "/motif/w fich_resultat" fichier`

Exemple :

`sed -n "/linux/w resultat" fichier` : Les lignes contenant 'linux' seront insérées dans le fichier 'resultat'

La commande sed

■ Substitution

- ◆ `sed -e "s/motif/remplacement/" fichier`

- Remplace 'motif' par 'remplacement' dans fichier

Exemples:

`sed -e "s/Linux/Unix/" fichier`

remplace la première occurrence 'Linux' par 'Unix'

`sed -e "s/Linux/Unix/gw resultat"` fichier

remplace toutes les occurrences 'Linux' par 'Linux' et écrit le résultat dans "resultat"

La commande sed

- Sélection

- ◆ Exemple

Afficher les lignes contenant linux.

```
user@ubuntu:~$ cat unixtxt
Linux est un SE Open source
il existe plusieurs distributions de linux
Linux est inspiré d'Unix, Unix est propriétaire.

user@ubuntu:~$ sed -n '/linux/p' unixtxt
il existe plusieurs distributions de linux
user@ubuntu:~$
```

Linux est SE Open source
 Il existe plusieurs distributions de linux
 Linux est inspiré d'Unix, Unix est propriétaire.

Fichier unixtxt

La commande sed

- Substitution

- ◆ Exemple

Remplacer linux par
 LINUX

```
user@ubuntu:~$ cat unixtxt
Linux est un SE Open source
il existe plusieurs distributions de linux
Linux est inspiré d'Unix, Unix est propriétaire.

user@ubuntu:~$ sed -e 's/linux/LINUX/p' unixtxt
Linux est un SE Open source
il existe plusieurs distributions de LINUX
il existe plusieurs distributions de LINUX
Linux est inspiré d'Unix, Unix est propriétaire.

user@ubuntu:~$
```

La commande sed

- Substitution
 - ◆ Exemple

Remplacer Unix par unix

Rajouter l'option 'g' pour que toutes les occurrences du motif soient remplacées

```
user@ubuntu:~$ cat unixtxt
Linux est un SE Open source
il existe plusieurs distributions de linux
Linux est inspiré d'Unix, Unix est propriétaire.

user@ubuntu:~$ sed -e 's/Unix/unix/gp' unixtxt
Linux est un SE Open source
il existe plusieurs distributions de linux
Linux est inspiré d'unix, unix est propriétaire.
Linux est inspiré d'unix, unix est propriétaire.

user@ubuntu:~$
```

La commande sed

- Suppression
 - ◆ sed "n1,n2d" fichier
 - Supprime les lignes entre n1 et n2 du fichier
 - ◆ sed "/motif/d" fichier
 - Supprime les lignes du fichier contenant motif
 - Exemples
 - sed "/linux/d" fichier : supprime les lignes contenant 'linux'
 - ◆ sed '/motif!/d' fichier
 - Supprime toutes les lignes sauf celles contenant 'linux'

La commande sed

- Suppression

- ◆ Exemple 1

Suppression des lignes
contenant linux

```
user@ubuntu:~$ cat unixtxt
Linux est un SE Open source
il existe plusieurs distributions de linux
Linux est inspiré d'Unix, Unix est propriétaire.

user@ubuntu:~$ sed '/linux/d' unixtxt
Linux est un SE Open source
Linux est inspiré d'Unix, Unix est propriétaire.

user@ubuntu:~$
```

La commande sed

- Suppression

- ◆ Exemple 2

Suppression de toutes
les lignes sauf la 2 et la
3

```
user@ubuntu:~$ cat unixtxt
Linux est un SE Open source
il existe plusieurs distributions de linux
Linux est inspiré d'Unix, Unix est propriétaire.

user@ubuntu:~$ sed '2,3!d' unixtxt
il existe plusieurs distributions de linux
Linux est inspiré d'Unix, Unix est propriétaire.

user@ubuntu:~$
```

La commande sed

- Suppression
 - ◆ Exemple 3

Suppression des lignes
ne pas contenant Unix

```
user@ubuntu:~$ cat unixtxt
Linux est un SE Open source
il existe plusieurs distributions de linux
Linux est inspiré d'Unix, Unix est propriétaire.

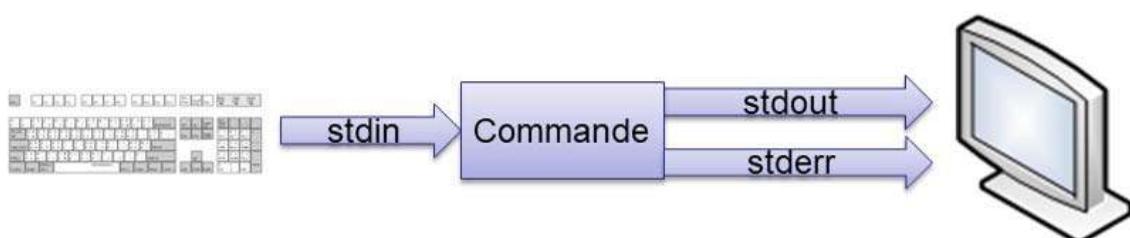
user@ubuntu:~$ sed '/Unix/!d' unixtxt
Linux est inspiré d'Unix, Unix est propriétaire.
user@ubuntu:~$
```

Les redirections

Les entrées/sorties

- Chaque commande dispose d'entrées/sorties :
 - ◆ Comme l'écran (sortie) ou le clavier (entrée)
- 3 types différents:
 - ◆ L'entrée standard définie par le symbole « **stdin** » et le descripteur **0**
 - Provenant du clavier par défaut
 - ◆ La sortie standard « **stdout** » et le descripteur **1**
 - Dirigée vers l'écran par défaut
 - ◆ La sortie d'erreurs « **stderr** » et le descripteur **2**
 - Egalement redirigée vers l'écran par défaut
- Possibilité de modifier le comportement par défaut
 - ◆ En utilisant les redirections d'entrées/sorties

Les entrées/sorties



Capacité de rediriger les entrées/sorties d'une commande
 « **stdout** » ou « **stderr** » vers un fichier plutôt qu'à l'écran
 « **stdin** » depuis un fichier plutôt que le clavier

Redirections des entrées/sorties

- Utilisation des opérateurs suivants :

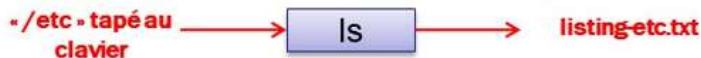
- ◆ > : Redirection de la sortie vers un fichier
- ◆ >> : Redirection de la sortie à la fin du fichier (concaténation)
- ◆ < : Redirection de l'entrée depuis un fichier

- Exemples:

Exemple de redirection de la sortie vers un fichier :

```
ls -l /etc > listing-etc.txt
```

- ◆ Le listing est écrit dans le fichier « listing-etc.txt »



Redirection de l'entrée de la commande « wc » depuis un fichier

- ◆ Compte le nb de lignes du fichier « listing-etc.txt »

```
wc -l < listing-etc.txt
```

Redirections (2)

- 2 syntaxes supplémentaires possibles pour les redirections

```
n>&m ou  
n>fichier
```

- ◆ n : Numéro du descripteur à rediriger
- ◆ m : Numéro du descripteur vers lequel on va rediriger
- ◆ fichier : Fichier vers lequel s'effectuera la redirection

- Rappel : 0 (entrée standard), 1 (sortie standard), 2 (sortie erreur)

- Exemple : Rediriger la sortie d'erreur vers un fichier

- ◆ Les messages d'erreurs seront écrits dans « erreurs.log »

```
cp /home/ali /home/sara 2>erreurs.log
```

- Exemple : Rediriger la sortie standard vers la sortie d'erreur

```
cp /home/ali /home/sara 1>&2
```

- Exemple : Rediriger stdout ET stderr vers un fichier

```
commande > fichier.txt 2>&1
```

Les tubes (pipes)

Les tubes (pipes)

- Il s'agit de rediriger la sortie d'une commande vers l'entrée d'une autre avec l'opérateur « | »



- Élaborer des commandes complexes en une seule ligne

- Exemple : Filtrer le résultat de la commande « ls » avec « grep »

```
ls -l /etc | grep 'mp3'
```

- On obtient la liste des fichiers contenant « mp3 »



```

user@laptop:~$ ls -l /etc | grep 'mp3'
-rw-r--r-- 1 user user 0 2011-08-27 15:16 morceau1.mp3
-rw-r--r-- 1 user user 0 2011-08-27 15:16 morceau2.mp3
-rw-r--r-- 1 user user 0 2011-08-27 15:16 morceau3.mp3
-rw-r--r-- 1 user user 0 2011-08-27 15:16 morceau4.mp3
  
```

Exécution conditionnelle de commandes

- Pour exécuter une seule commande, rien de plus simple
 - ◆ Taper son nom au clavier
- Pour exécuter plusieurs commandes à la suite
 - ◆ L'exemple suivant crée un répertoire, s'y déplace et crée un fichier


```
mkdir toto ; cd toto ; touch alire.txt
```
- L'exécution conditionnelle de commandes est possible
 - ◆ Les commandes s'exécutent les unes après les autres sous condition
 - ◆ Utilisation des opérateurs && et ||
 - L'exemple suivant exécute commande1 et commande2 seulement si le résultat renvoyé par commande1 est égal à 0


```
commande1 && commande2
```
 - Même chose mais si le résultat de commande1 est différent de 0


```
commande1 || commande2
```

Ca marche ou ça marche pas ?

- Une commande renvoie toujours une valeur
 - ◆ 0 si la commande s'est exécutée correctement
 - ◆ 1 ou différent de zéro dans le cas contraire
- Exemple
 - ◆ La variable « \$? » correspond à la valeur renvoyée par la dernière commande exécutée (Donc la commande « cd »)


```
# cd rep
Bash : cd rep : Le répertoire n'existe pas
# echo $?
1
```

- Cette valeur peut-être exploitée dans un script pour connaître le résultat d'une commande avant d'exécuter la suite

Exercices

- Copier le fichier « /etc/syslog.conf » vers son répertoire home
[REDACTED]
- A l'aide de la commande « cat », filtrer les lignes contenant le mot « internet » dans le fichier « toto.txt »
[REDACTED]
- Renommer le répertoire « rapports » vers « rapport2007 » et stocker les messages d'erreurs dans un fichier « mv-rapport.log »
[REDACTED]
- Créer le fichier « rap2009 dans le répertoire « rapports » seulement s'il existe
[REDACTED]

Méta-caractères motifs d'englobement

Méta-carctère	Signification
?	Remplace un seul caractère
[]	Un des caractères entre les crochets
*	Répétition
^	Début de ligne ou négation entre []
\$	Fin de ligne
-	Intervalle de ... à entre []

Caractères spéciaux

- Certains caractères ont une signification particulière
 - ◆ Interprétés par le shell
- Astérisque ou étoile : *
 - ◆ Interprété comme toute suite de caractères alphanumérique
 - ◆ Exemple : Effacer tous les fichiers commençant par « rapport »

```
rm rapport*
```
- Point d'interrogation : ?
 - ◆ Interprété comme un seul caractère alphanumérique
 - ◆ Exemple : Effacer certains fichiers commençant par « rapport?.doc »

```
rm rapport?.doc
```

 - « rapport1.doc » sera effacé mais pas « rapport12.doc »
- Point virgule ; ;
 - ◆ Séparateur de commandes

```
cp bilan.txt bilan2007.txt ; rm bilan.txt
```

Caractères spéciaux

■ Les crochets : []

- ♦ Remplace un caractère choisi parmi ceux énumérés entre les crochets
- ♦ Exemple : Effacer les fichiers dont la 1^{ère} lettre est « a » ou « b » et se terminant par « .txt »

```
rm [ab]*.txt
```

- « args1.txt » et « bilan.txt » seront effacés mais pas « comment.txt »

- ♦ Exemple : Effacer les fichiers numérotés de 10 à 29
 - « rapport12.txt » mais pas « rapport3.txt »

■ L'espace

```
rm rapport[12][0-9].txt
```

- ♦ Utilisé comme séparateur de paramètres pour une commande
- ♦ Exemple : Effacement de 2 fichiers passés en paramètres

```
rm rapport.doc rapport2011.txt
```

Caractères spéciaux

■ L'antislash : \

- ♦ Inhibe le caractère spécial suivant
- ♦ Exemple : Effacer un fichier contenant le caractère spécial espace

```
rm rapport\ .txt
```

■ Autres caractères spéciaux : !, ^, \$, <, >, |

- ♦ ^ : Exprime la négation
- ♦ \$: Utilisé pour les variables dans les scripts
- ♦ ! : Utilisé pour accéder à l'historique des commandes (! suivi du numéro de la commande dans l'historique. Voir la commande « history »)
- ♦ <, > : Redirections
- ♦ | : pipe (tube)

Chaînes de caractères

- Il existe plusieurs délimiteurs de chaînes de caractères
 - ◆ Apostrophe (simple quote) : 'texte'
 - Le texte n'est pas du tout interprété
 - ◆ Guillemets (double quotes) : "texte"
 - Seuls les caractères \, \$ et ` sont interprétés
 - Utilisé pour du texte qui contient des variables ou des caractères spéciaux
 - ◆ Anti-quotes : `texte`
 - Le texte est interprété comme une commande à exécuter. Le résultat de cette commande sera substitué
 - Utilisé dans le but d'exploiter le résultat d'une commande

Chaînes de caractères : Exemples

- Exemples
 - ◆ Rechercher la chaîne « toto » dans tous les fichiers du répertoire « /home/ali »

root@laptop:~# find /home/ali -type f -exec grep -i 'toto' {} \;
 - ◆ Rechercher les fichiers contenant la date d'aujourd'hui
 - 1) Obtenir la date d'aujourd'hui au format JJMMMAA


```
root@laptop:~# date +%d%m%Y
28082008
```
 - 2) Exploiter ce résultat pour rechercher cette date dans les fichiers

root@laptop:~# find / -type f -exec grep -i '28082008' {} \;
 - ◆ Créer un fichier « alire.txt » dans le répertoire home de l'utilisateur
 - La variable \$HOME sera remplacée par sa valeur


```
root@laptop:~# echo 'alire' > $HOME/alire.txt
```

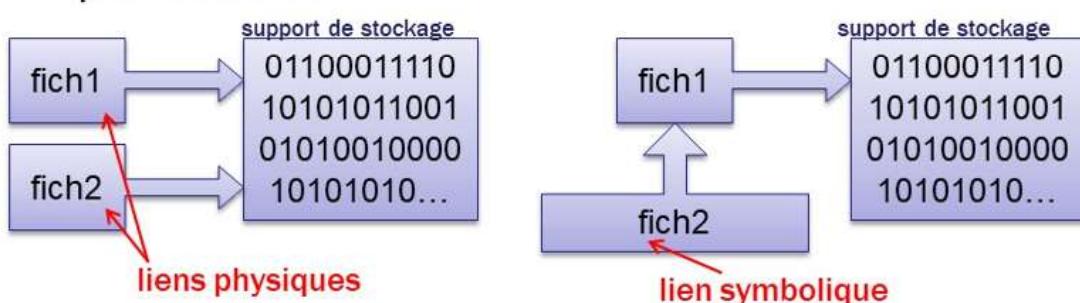
Exercices

- Déplacer tous les fichiers commençant par « bilan » vers le répertoire « bilans » qu'il faut créer avant
[Yellow Box]
- Afficher la liste des fichiers dont le nom contient « bilan2006 » et « bilan2007 »
[Yellow Box]
- Afficher la liste des fichiers dont le nom commence par une majuscule suivie d'une minuscule suivie d'un chiffre.
[Yellow Box]
- Créer un fichier qui sera nommé « backup_JJMMAA.dat » avec la date du jour
[Yellow Box]

Les liens
F62 LIENS

Les liens

- Un lien est un type spécial de fichier qui fait référence à un autre fichier
- Axe central du fonctionnement de Linux, le lien permet :
 - ◆ De créer des raccourcis vers des fichiers existants
 - La compatibilité des logiciels entre les distributions Linux est assurée par les liens
 - ◆ D'éviter de stocker plusieurs fois le même fichier dans des répertoires différents
- Un petit dessin :



Les liens symboliques

- Le lien symbolique est une référence vers un fichier cible
 - ◆ Lorsque le fichier cible est effacé, le lien est rompu
 - ◆ Lorsque le lien est effacé, le fichier cible n'est pas effacé
- Exemple :

```
ali@laptop:~/Documents$ ls -l
total 8
lrwxrwxrwx 1 ali G.info1 29 2011-08-25 14:23 ip -> /proc/sys/net/ipv4/ip_forward
drwxr-xr-x 3 ali G.info1 4096 2011-06-02 14:20 software
drwxr-xr-x 3 ali G.info1 4096 2011-07-29 15:54 vmware-tools
```

Nom du lien
Indique que c'est un lien
Emplacement du vrai fichier

- La commande « `ln` » avec l'option « `-s` » est utilisée pour créer un lien symbolique

```
ali@laptop:~/Documents$ ln -s /proc/sys/net/ipv4/ip_forward ip
```

Cible (Target)

Nom du lien (link name)

Les liens physiques

- Un lien physique est associé à un emplacement sur le support de stockage
 - ◆ 2 liens peuvent être associés au même « inode »
 - ◆ Similaire à la notion de « pointeurs » du langage C
 - ◆ Deux liens physiques sont considérés comme 2 fichiers indépendants
 - Même si leur contenu est au même emplacement sur le support
 - ◆ Le lien physique est vu comme un fichier régulier
- Créer un lien physique avec la commande « ln » :

```
ali@laptop:~/Documents$ ln /home/ali/Documents/rapport2007-2008.doc rap0708
ali@laptop:~/Documents$ ls -il
total 176
470930 -rw-r--r-- 2 ali G.info1 84091 2008-08-25 14:48 rap0708
470930 -rw-r--r-- 2 ali G.info1 84091 2008-08-25 14:48 rapport-annee2007_2008.doc
```

L' « inode » est identique. Il s'agit bien de liens physiques

Nombre de liens vers cet inode. C'est un indice permettant de supposer qu'il s'agit d'un lien

nom fichier

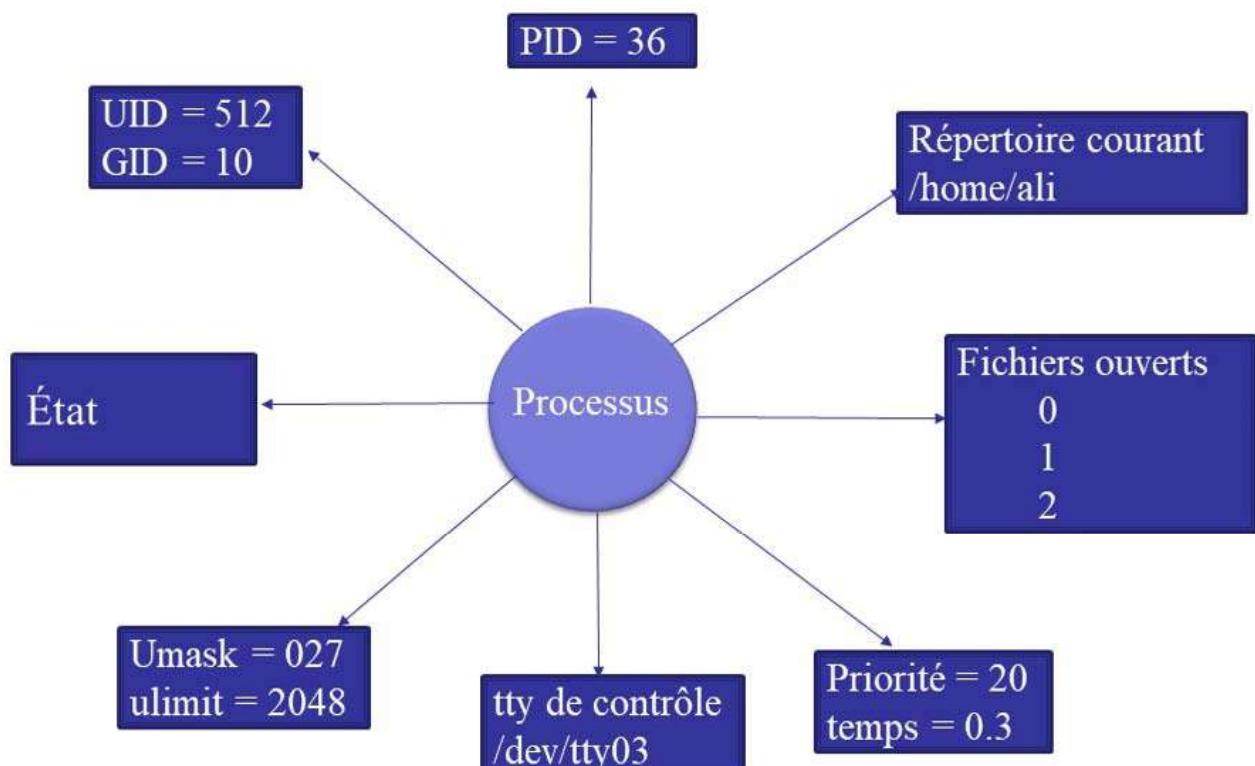
cible

Gestion des processus

Environnement d'un processus

- PID: l'identificateur du processus
- UID: l'identificateur de l'utilisateur propriétaire du processus
- GID: le groupe auquel appartient cet utilisateur
- État du processus : élu, prêt ,bloqué
- Ulimit: taille maximale des fichiers créés par ce processus
- Priorité
- Temps d'exécution
- Terminal de contrôle ou de rattachement
- Les fichiers ouverts par ce processus
- ...

Environnement d'un processus



Ordonnancement

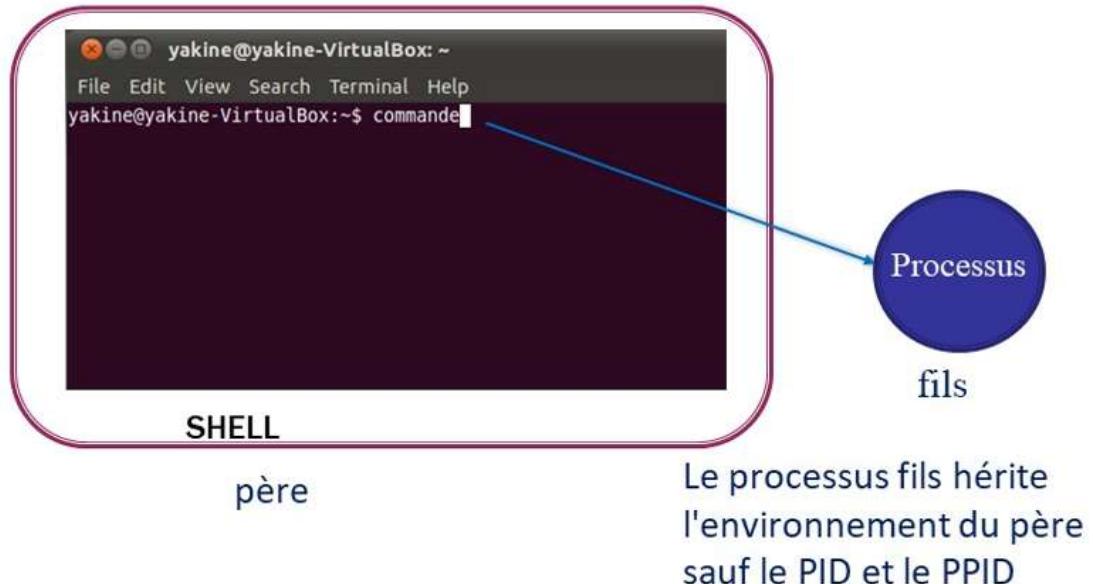
- Le noyau Linux va affecter au processeur le processus en cours dont la priorité est la plus importante.
- Lorsque ce processus est en attente ou si un processus de priorité plus importante apparaît, le noyau libère l'unité centrale du processus en cours et charge le nouveau processus de haute priorité.
- L'ordonnanceur veillera à ce que le processeur haute priorité ne monopolise pas l'unité centrale, le noyau Linux va modifier dans le temps les priorités des processus en attente et en cours afin de permettre à l'ensemble des processus d'être exécuté.

Création d'un processus

- Un processus peut être créé :
 - ◆ Initialisation du système
 - Init : le processus qui initialise le système
 - Init à pid=1
 - Processus systèmes (deamons)
 - Shell : le processus qui permet de lancer les commandes
 - ...
 - ◆ Exécution d'un appel système de création de processus par un processus en cours d'exécution
 - Sous UNIX/Linux, le seul appel système permettant de créer un nouveau processus est "fork" (TP5)
 - ◆ Directement en lançant une commande

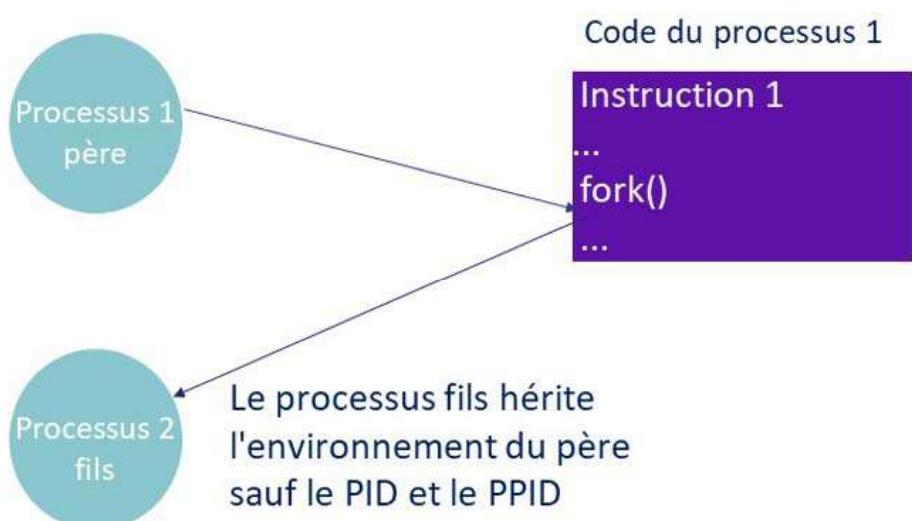
Création d'un processus

- Directement par une commande

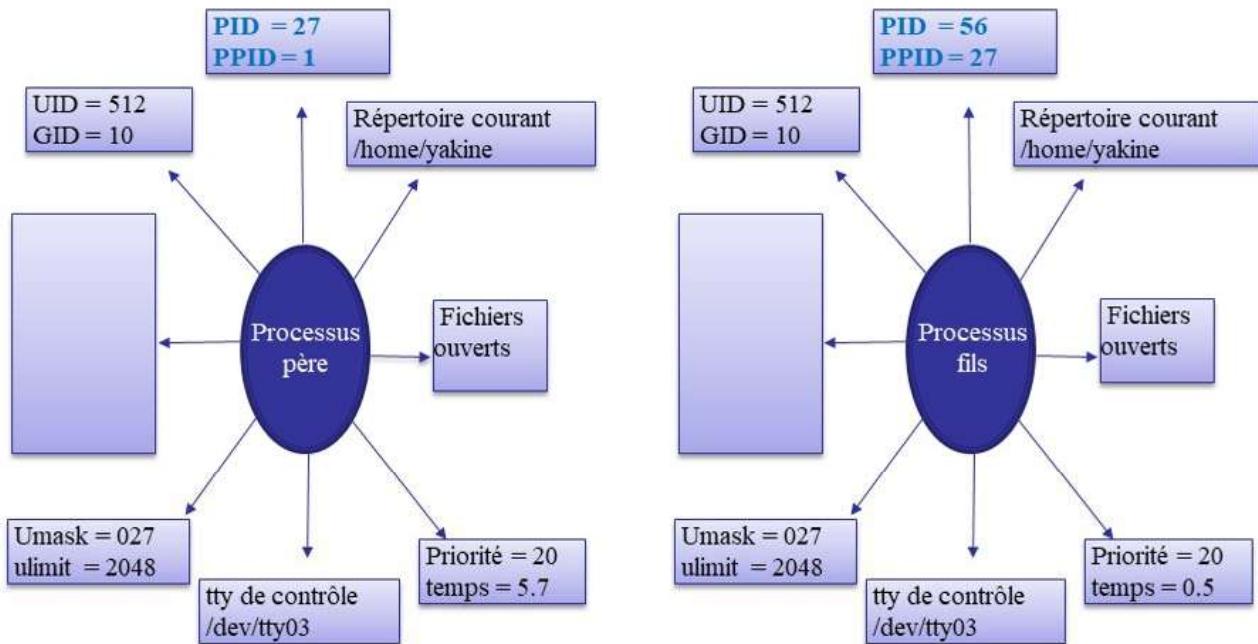


Création d'un processus

- Exécution d'un appel système de création de processus par un processus en cours d'exécution
 - ◆ L'appel système fork crée un clone du processus appelant (père)
 - ◆ Les deux processus père et fils ont la même image mémoire et les mêmes fichiers ouverts



Processus père/fils



Liste de tous les processus courants ps

- La commande `ps` permet d'afficher une liste des processus courants.
- Sans option, elle n'affiche que ceux qui vous appartiennent et qui ont été lancés dans le terminal dans lequel vous exécutez cette commande.
- Il existe de nombreuses options que vous pouvez voir dans man `ps`.

```
yakine@yakine-VirtualBox:~$ ps -u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
yakine    2553  0.0  0.8  6580  3240 pts/0      Ss  02:19   0:00 bash
yakine    2590  0.1  0.8  6580  3244 pts/0      S   03:44   0:00 bash
yakine    2611  0.0  0.2  4592  1080 pts/0     R+  03:48   0:00 ps -u
```

Liste de tous les processus courants ps

■ Les options

- ◆ L'option "-x" permet de visualiser tous les processus actifs de l'utilisateur courant
- ◆ L'option "-ax" permet de visualiser tous les processus de la machine de tous les utilisateurs
- ◆ L'option "-aux" permet de visualiser affiche les utilisateurs associés à chaque processus
- ◆ L'option "-u nom_utilisateur" affiche chaque processus associés à utilisateur
- ◆ L'option "-p PID" affiche les informations sur un processus
- ◆ L'option "-t" affiche tout les processus liées a un terminal

■ Les champs

PID est l'identificateur unique du processus attribué de façon unique par le noyau.

TTY indique le numéro de terminal auquel il est rattaché (le "?" indique que la commande n'est pas associée à un terminal)

TIME affiche le temps processeur utilisé par ce processus

CMD affiche le nom de la commande associé à ce PID

USER indique le nom de l'utilisateur du processus

START indique l'heure à laquelle le processus a été démarré

%CPU affiche l'utilisation du processeur en pourcentage

%MEM affiche l'utilisation de la mémoire vive en pourcentage

RSS donne l'utilisation de la mémoire physique utilisée en kilobytes par le processus (hors swap)

VSZ donne l'utilisation des bibliothèques partagées et la mémoire utilisée pour son fonctionnement

STAT affiche l'état actuel du processus, R comme Run ...

les états de processus

■ Un processus peut prendre plusieurs états, qui correspondent chacun à une lettre.

- ◆ La lettre **R** (run) indique que le processus est en cours d'exécution, et qu'il mobilise le processeur.
- ◆ La lettre **S** (short sleep) indique que le processus est en veille, mais prêt à fonctionner.
- ◆ La lettre **T** indique que le processus est suspendu, il ne mobilise pas le processeur.
- ◆ La lettre **I** indique que le processus est suspendu depuis plus de vingt secondes, il est en attente.
- ◆ La lettre **Z** (zombie) indique que le processus s'est terminé mais dont les ressources n'ont pas encore été libérées. Il est de la responsabilité du processus père de libérer les ressources occupées par ses fils zombies.

Avant plan et tâche de fond

- Avant plan : on tape la commande ou le nom du programme, on n'a pas la main sur le terminal, on attend que la commande finisse.
- Tâche de fond ou en arrière plan : on lance la commande ou le programme puis on a la main sur le terminal de suite.
 - ◆ Comment ? Avec le signe & à la fin de la commande
- Typiquement, les serveurs tournent en tâche de fond.

Passage avant - arrière plan

- Suspendre le processus en avant plan **Ctrl-z**
- **bg** pour redémarrer le processus en arrière plan
- **fg** pour passer le processus de l'arrière plan à l'avant plan
- **jobs** connaître les processus en arrière plan

Les signaux

- Un signal est un événement logiciel envoyé à un processus
- Un signal peut être envoyé par le noyau, l'utilisateur ou un autre processus.
- `kill -l` liste tous les signaux, désignés par un nom auquel correspond un numéro

Les signaux

- Chaque processus peut recevoir des signaux
- Chaque signal a une signification particulière
- Pour envoyer un signal on utilise la commande `kill`
- Exemple `kill -9 2345`

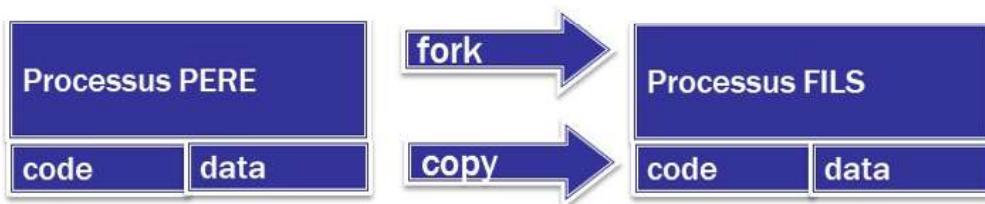
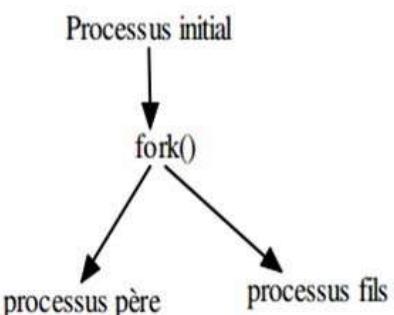
Nom du signal	Numéro	Description
SIGINT	2	Touche Ctrl-C, termine le processus
SIGKILL	9	arrêter tout processus car il ne peut être géré différemment que le comportement par défaut. L'arrêt du programme est brutal.
SIGTERM	15	Arrête le processus, mais permet d'effectuer des opérations avant l'arrêt.
SIGCHLD	17	Informe le père de la mort de son fils

Appels système pour la gestion des processus

F.YAKINE

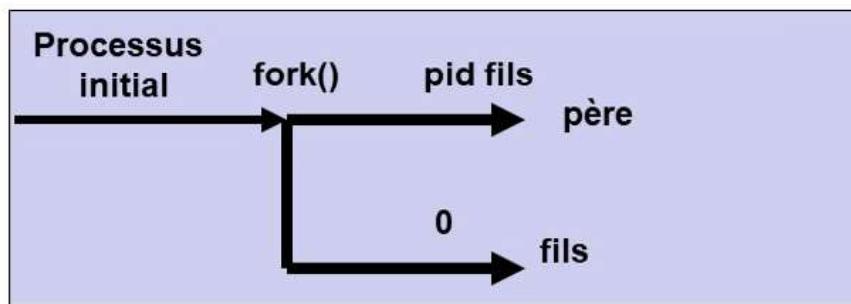
LES processus Unix/LINUX

- Création de processus
- C'est par **Clonage** du processus créateur (PERE)
- Le fils démarre à la fin du **fork()**



LES processus Unix/LINUX

- À l'issue d'un fork() les deux processus s'exécutent **simultanément**.
- La fonction fork retourne :
 - ✓ le numéro du **pid** (process id) du fils dans le père
 - ✓ 0 dans le fils



LES processus Unix/LINUX

- Le recouvrement du code d'un processus
- Il est possible de créer un processus par le remplacement d'un processus par un nouveau code provoqué par la fonction par **exec ()**

- Terminaison normale d'un processus
- **exit()** : Quand le processus fils se termine son père récupère sa valeur de sortie