



Institut National des Sciences Appliquées et de Technologie

Linux – LPI101

Dorsaf SEBAI



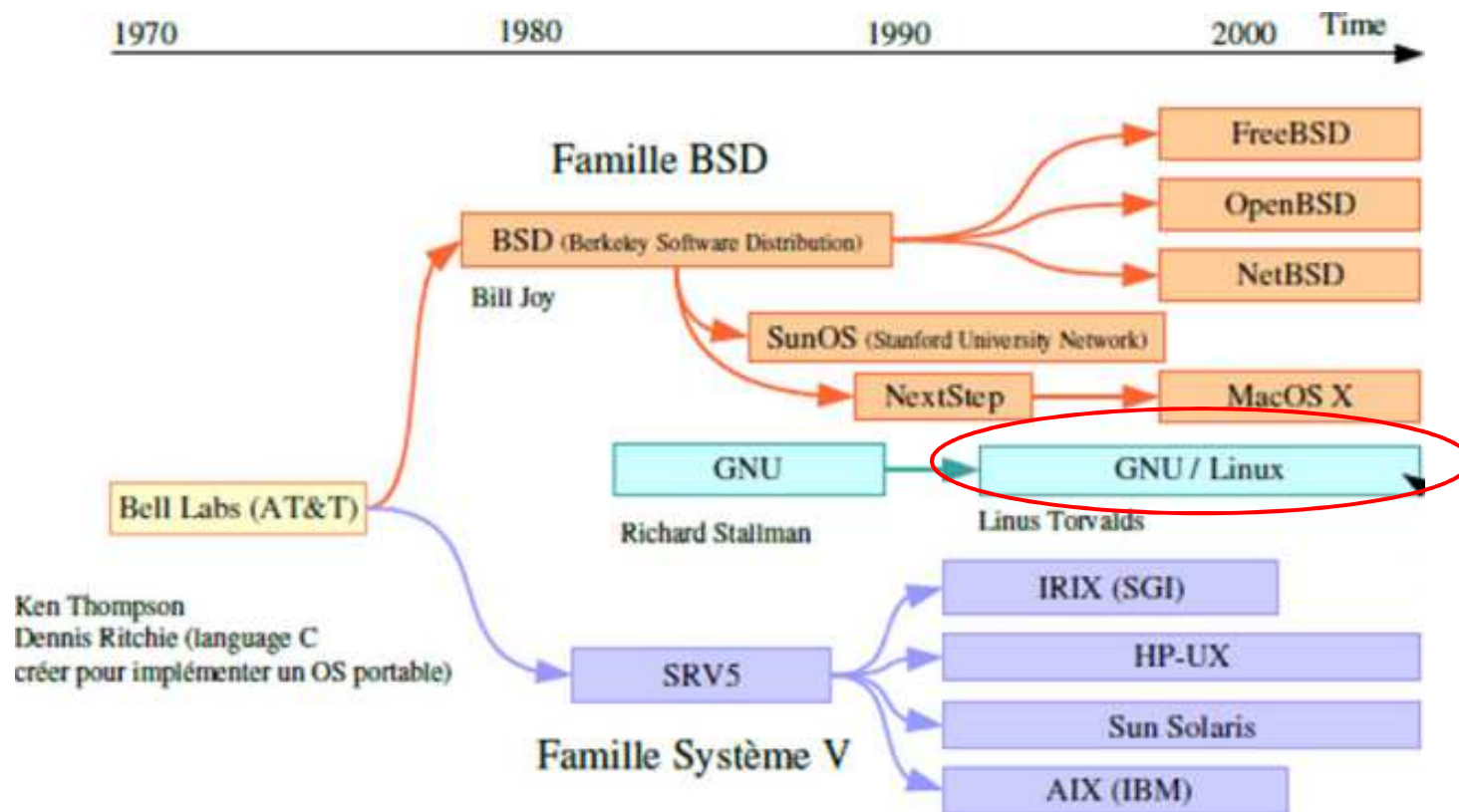
Chapitre I : Introduction à Linux

1. Historique
2. Caractéristiques
3. Système de fichiers
4. Distributions de la famille GNU/Linux

Historique (I)

- Le système d'exploitation Unix a été développé en 1969 dans les laboratoires Bell (Bell Labs) par un petit groupe dans le cadre d'un projet de recherche privé.
- Unix a été écrit en C (pas en Assembleur).
- Plusieurs clones du système original ont vu le jour :
 - ❑ BSD (Berkeley Software Distribution) en 1977 par Bell Labs (AT&T).
 - ❑ System V en 1977 par Bell Labs.
 - ❑ Coherent OS en 1983 par Mark Williams Company.
- En 1991, l'étudiant finlandais Linus Torvald a libéré le code de la première version de son OS libre : [Linux](#).

Historique (2)



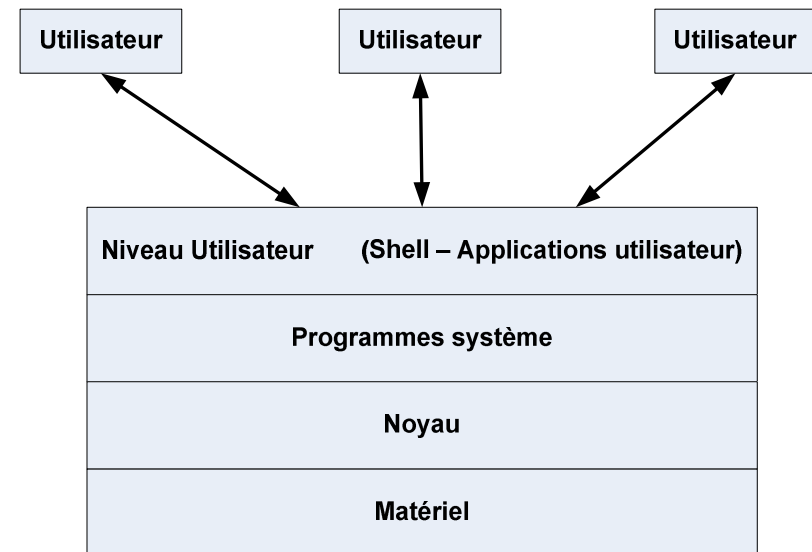
Caractéristiques (I)

- Une grande **portabilité** : Linux est écrit en majeure partie en langage C.
- Des interfaces de dialogue **shell** : environnements simples et puissants pour des traitements à un haut niveau (manipulation du contenu des fichiers, redirections, ...).
- Système **multiutilisateurs** : plusieurs personnes peuvent utiliser l'OS simultanément. Il est en mesure de partager équitablement les ressources de la machine.
- Système **multitâches** : capable de faire fonctionner plusieurs programmes simultanément sur la machine (processus, signaux, ...).

Caractéristiques (2)

Couches fonctionnelles de l'OS :

- Chaque couche est conçue de manière à être utilisée sans connaître les couches inférieures.
- L'utilisateur peut accéder aux ressources par le biais de dialogue simple offert par la couche la plus évoluée : Shell.
- Le noyau assure l'interfaçage des applications avec les ressources matérielles et prend en charge la gestion des processus et du système de fichiers.



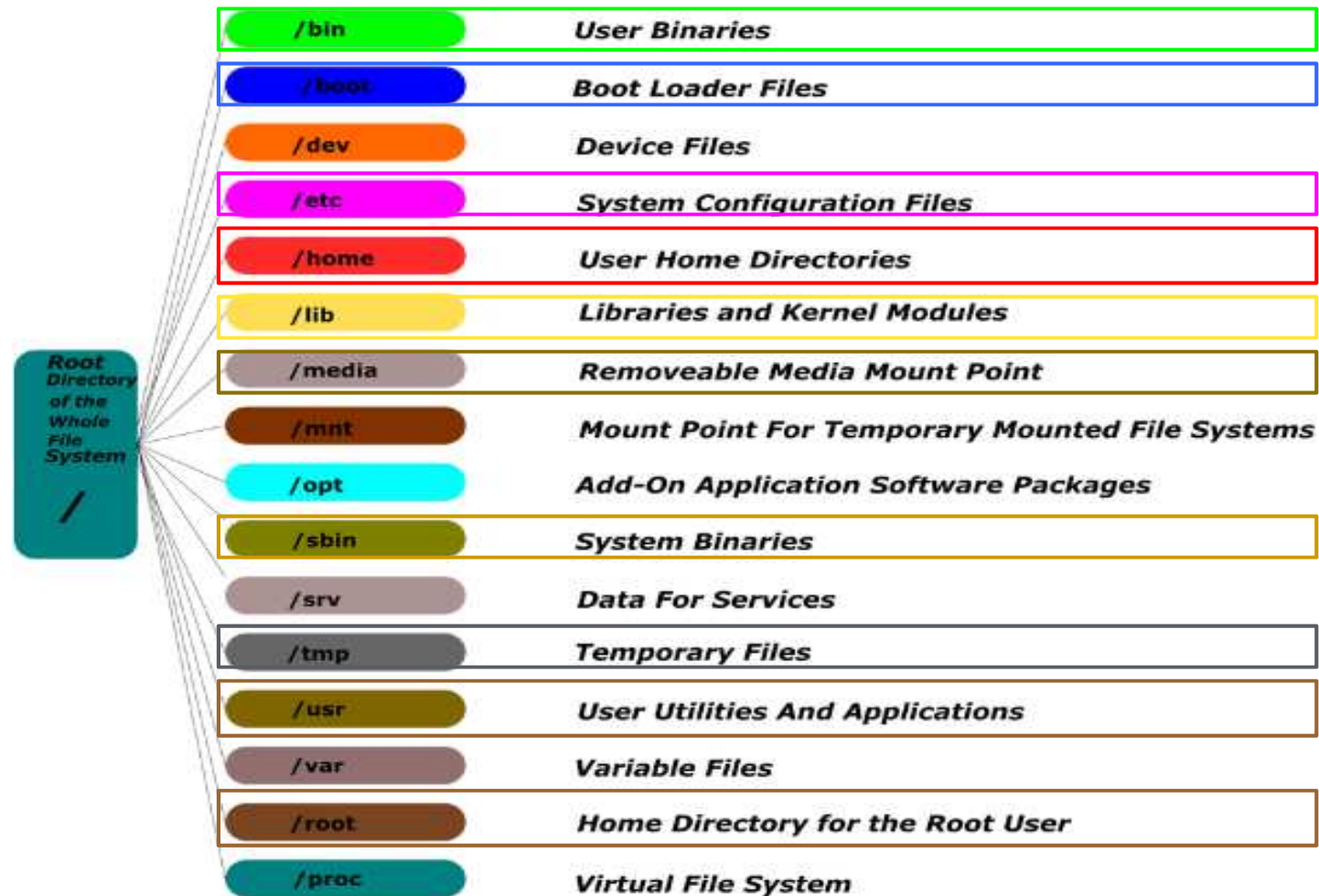
Caractéristiques (3)

- Utilisateurs :
 - ☐ Appartiennent à des groupes.
 - ☐ Ont des droits d'accès aux fichiers/répertoires.
- **Administrateur (root)** :
 - ☐ Le nom de l'utilisateur qui a tous les droits sur le système.
 - ☐ Pour se mettre en root :

```
$ sudo nano /etc/X11/xorg.conf  
Password:  
#Exécution de la commande...
```

```
$ sudo su  
Password:  
# Tapez la commande que vous désirez
```

Systemes de fichiers (I)



Système de fichiers (2)

Noms fichiers :

- ❑ Linux fait la différence entre les majuscules et les minuscules.
- ❑ En théorie, tous les caractères du clavier sont autorisés.
- ❑ En pratique, il vaut mieux ne pas utiliser certains caractères (comme * ou ?) .
- ❑ Le caractère - est déconseillé au début d'un nom de fichier car certaines commandes pourraient l'interpréter comme une option.

Système de fichiers (3)

Inodes :

- ❑ Nœud d'index ou inode est une structure de données contenant les métadonnées du fichier.
- ❑ Tout fichier possède son unique inode.
- ❑ Linux enregistre les fichiers sur la base du numéro d'inode et pas sur la base d'un nom.
- ❑ Consulter le n° d'inode d'un fichier : `ls -li nomfichier`.

Système de fichiers (4)

Inodes :

- ❑ Les métadonnées contenues dans un inode :
 - Type de fichier.
 - Droits d'accès.
 - Nombre de liens physiques.
 - UID : identifiant du propriétaire.
 - GID : identifiant du groupe primaire du propriétaire.
 - Taille du fichier.
 - atime (access time) : date du dernier accès au fichier.
 - mtime (modify time) : date de la dernière modification du contenu du fichier.
 - ctime (changing time) : date de la dernière modification des attributs du fichier.
 - Adresse du fichier.

Système de fichiers (5)

Inodes :

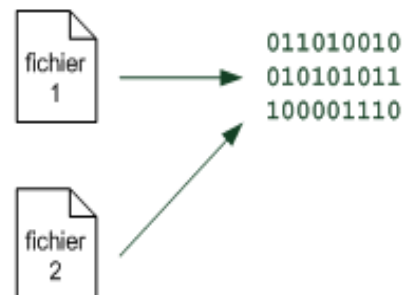
Lien physique : permet de donner plusieurs noms/chemins d'accès à un même fichier en pointant sur un **même numéro d'inode**. Un fichier peut donc avoir plusieurs noms et existera tant qu'il a au moins un nom.

```
~$ ls -li ~/fichiersource
~/Bureau/monlienphysique 5954521
~/fichiersource 5954521
```

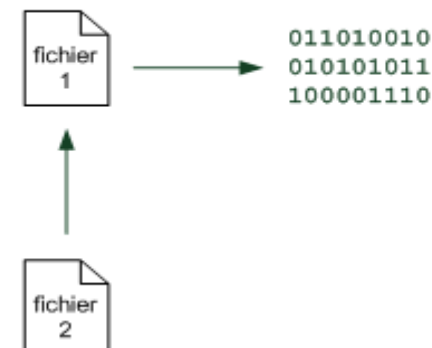
Lien symbolique : permet de donner plusieurs noms/chemins d'accès à un fichier en pointant sur un **nom de fichier**. Si le fichier pointé est supprimé, les liens symboliques pointeront vers un fichier inexistant.

```
~$ ls -li ~/fichiersource
~/Bureau/monliensymbolique 5954521
~/fichiersource 6876911
```







Nom du fichier Contenu (inode)



Nom du fichier Contenu (inode)



Distributions de la famille GNU/Linux

	Développeur	Gestionnaire de paquets logiciels
RedHat 	Société RedHat Enterprise Linux (RHEL)	RPM (.rpm) (Red Hat Package Manager)
Fedora 	Communauté Fedora Project	RPM
CentOS 	Community enterprise Operating System	RPM
Suse 	Communauté openSuse	RPM
Debian 	Communauté développeurs Debian	Deb (.deb) (Debian)
Ubuntu 	Communauté Ubuntu Foundation	Deb

Chapitre 2 : Shell

1. Présentation
2. Variables du shell
3. Lignes de commandes
4. Tubes/pipes
5. Filtres

Présentation (I)

- Le shell est un programme interpréteur qui analyse une à une les lignes de commandes entrées par l'utilisateur et assure l'exécution des tâches correspondantes.
- Malgré la diversité des interfaces graphiques, le shell reste le plus utilisé pour le dialogue système-utilisateur : permet à l'utilisateur d'accéder directement aux commandes standards Linux.

Présentation (2)

- sh (Bourne Shell)
- bash (Bourne Again Shell)
- ksh (Korn Shell)
- csh (C Shell)
- tcsh (Tenex C Shell)
- Zsh (Z Shell)

Présentation (3)

- Le positionnement du shell peut se faire :
 - ❑ par l'administrateur (fichier `/etc/passwd`) : le shell lancé au démarrage.
 - ❑ après le login par une demande utilisateur : lancer un autre shell par les commandes `sh`, `csh`, `ksh`, ...
- Liste des shells dans le système : `/etc/shells`
- Terminer une session shell : `exit`

Variables du shell (I)

- Lors de l'exécution d'une commande, le shell tient compte de certaines variables permettant de paramétrer le fonctionnement du système.
- Langue utilisée, chemins vers les fichiers exécutables, chemin vers les librairies, ...
- **Variables d'environnement** :
 - ❑ Elles constituent l'environnement d'exécution pour les programmes.
 - ❑ Elles sont connues par toutes les commandes lancées par le shell.
 - ❑ Elles sont par convention en majuscules.
 - ❑ Etendre une variable d'environnement aux shells fils si elle a été nouvellement déclarée ou modifiée.

Variables du shell (2)

- `variable=valeur` : déclarer une variable.
- `echo $variable` : afficher une variable.
- `env` : affiche les variables d'environnement et leurs valeurs.
- `set` : affiche les variables d'environnement définies par le shell avec leurs valeurs + les variables et les fonctions définies par l'utilisateur.
- `unset` : effacer une variable.
- `export` : exporter une variable d'environnement aux shells fils

```
VARIABLE=valeur  
export VARIABLE
```

```
VARIABLE=valeur  
declare -x VARIABLE
```

```
export VARIABLE=valeur
```

Variables du shell (3)

- Exemples de variables globales :
 - ❑ **PS1** : le prompt primaire (par défaut \$).
 - ❑ **PS2** : le prompt secondaire (par défaut >).
 - ❑ **HOME** : le chemin absolu du répertoire personnel (home directory).
 - ❑ **PATH** : les répertoires de recherche des exécutables.
 - ❑ **MANPATH** : les répertoires de recherche des fichiers de man.
 - ❑ **SHELL** : le shell de login déclaré dans /etc/passwd.
 - ❑ **EDITOR** : l'éditeur de texte par défaut.
 - ❑ **PWD** : le répertoire de travail de l'utilisateur.
 - ❑ **IFS** : chaîne de séparation des champs.

Lignes de commandes (I)

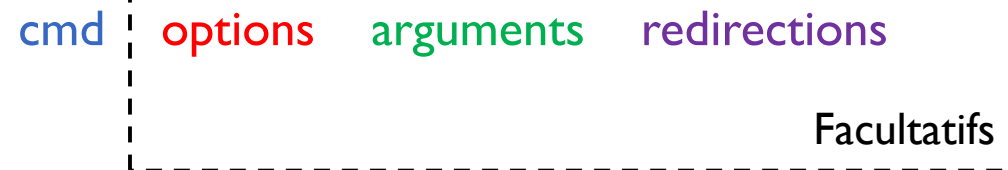
La commande est une chaîne de caractères comprenant quatre parties :



- ❑ cmd : action interne ou fichier exécutable/script shell (chemin absolu ou relatif).
- ❑ options : une ou plusieurs options (sous forme de - suivi par des lettres).
- ❑ arguments : sur quoi la commande va agir.

Lignes de commandes (2)

La commande est une chaîne de caractères comprenant quatre parties :



❑ redirections : Linux définit trois fichiers standards

- Le fichier standard d'entrée (stdin) : descripteur 0, ouvert en lecture seule (< fic).
- Le fichier standard de sortie (stdout) : descripteur 1, ouvert en écriture (> fic ou | > fic et >> fic ou | >> fic).
- Le fichier standard de sortie d'erreur (stderr) : descripteur 2, ouvert en écriture (2 > fic ou 2 >> fic).
- La commande lit à partir d'un fichier et écrit dans deux fichiers
- Les redirections créent un canal de communication entre un fichier et un processus.

Lignes de commandes (3)

Commandes usuelles

<i>man</i>	Affiche une description complète d'une commande
<i>pwd</i>	Renvoie le chemin absolu du répertoire courant (Print Working Directory)
<i>ls</i>	Liste le contenu d'un répertoire
<i>touch</i>	Crée un fichier
<i>cd</i>	Change le répertoire courant (Change Directory)
<i>cat</i>	Affiche le contenu d'un fichier
<i>echo</i>	Afficher une chaîne de caractères
<i>which</i>	Trouver l'emplacement d'une commande en effectuant sa recherche dans le contenu de la variable PATH
<i>whereis</i>	Recherche d'une commande dans les binaires et les pages de manuel en plus du contenu de la variable PATH
<i>alias</i>	Créer un alias à une commande

Lignes de commandes (4)

Commandes usuelles

<i>history</i>	Visualise les commandes récemment exécutées
<i>su</i>	Ouvrir une session en tant qu'un autre utilisateur
<i>clear</i>	Efface le contenu de l'écran
<i>rm fichier</i>	Supprime un fichier
<i>mv fichier 1 fichier 2</i>	Nomme et déplace un fichier
<i>cp fichier 1 fichier 2</i>	Copie un fichier
<i>mkdir rép</i>	Crée un répertoire
<i>rmdir rép</i>	Supprime un répertoire vide

Lignes de commandes (5)

Enchaînement de commandes

- commande **&** : permet l'exécution d'une commande en arrière plan (en parallèle).
- commande **|** ; commande **2** :
 - ❑ Enchaîner les commandes sans se soucier de la réussite ou de l'échec de l'exécution de chacune.
 - ❑ Une commande réussie : code retour/flag=0 (sinon 2).
 - ❑ Si flag=0 : Exécution de la commande suivante.
 - ❑ Si flag=2 : Déclencher un message d'erreur et exécution de la commande suivante.
 - ❑ Quand on est sûr de la bonne exécution des commandes ou quand leurs bonnes exécutions n'est pas importante.

Lignes de commandes (6)

Enchaînement de commandes

- commande 1 **&&** commande 2 :
 - ❑ Exécuter une commande que si la commande précédente a réussi.
 - ❑ Si flag=0 : Exécution de la commande suivante.
 - ❑ Si flag=2 : Mettre fin à l'exécution de la commande suivante.
 - ❑ *Exemple* : `cd /home/xxx/dossier | && touch file |`
- commande **||** commande 2 :
 - ❑ Exécuter une commande que si la commande précédente n'a pas réussi.
 - ❑ Si flag=0 : Mettre fin à l'exécution de la commande suivante.
 - ❑ Si flag=2 : Exécution de la commande suivante.
 - ❑ *Exemple* : `cd /home/xxx/dossier | || mkdir / /home/xxx/dossier |`

Lignes de commandes (7)

Regroupement de commandes

- Sert à :
 - ❑ Rediriger un ensemble de commandes vers ou depuis un même fichier.
 - ❑ Exécuter un ensemble de commandes en arrière-plan.
 - ❑ Conditionner l'exécution d'un ensemble de commandes.
- Accolades :
 - ❑ Les commandes sont exécutées par le shell courant.
 - ❑ {commande; commande; commande; ...}
- Parenthèses :
 - ❑ Les commandes sont exécutées par un shell fils.
 - ❑ (commande; commande; commande; ...)

Lignes de commandes (8)

Répétition de commandes

- La méthode la plus rapide pour utiliser l'historique des commandes : les flèches du pavé directionnel (↑ et ↓).
- Variables d'environnement :
 - ❑ **HISTFILE** : le nom du fichier contenant l'historique (par défaut ~/.bash_history).
 - ❑ **HISTSIZE** : limite des commandes historiques accessibles (<0 pas de limite).
- Il est possible d'afficher la totalité des commandes déjà lancées en tapant la commande **history**.
 - ❑ Elle retourne toutes les commandes avec leurs numéros d'entrée (première colonne) de la plus ancienne à la plus récente.
 - ❑ Vider l'historique : *history -c*

Lignes de commandes (9)

Répétition de commandes

- ❑ `!!` : commande précédente.
- ❑ `!n` : commande numéro *n*.
- ❑ `!-n` : commande numéro *n* à partir de la fin.
- ❑ `!string` : la commande la plus récente commençant par *string*.
- ❑ `!?string[?]` : la commande la plus récente contenant *string*.
- ❑ `^string / ^string2^` : la commande précédente en remplaçant *string1* par *string2*.
- ❑ `fc -l m n` : de la commande numéro *m* jusqu'à la commande numéro *n*.

Lignes de commandes (10)

Citations/quotes

- **Quote simple ' :** n'interprète pas les caractères spéciaux.

```
var='test'  
newvar='Value of var is $var'  
echo $newvar
```

Value of var is \$var

- **Quotes inversées (backquotes) ` :** utilisée pour substituer une commande par son résultat

```
DATE=`date`  
echo $DATE  
DATE=$(date)  
echo $DATE
```

Thu Feb 8 16:47:32 MST 2001
Thu Feb 8 16:47:32 MST 2001

Lignes de commandes (II)

Citations/quotes

- **Double quote "** : permet la substitution des variables précédées par \$ et des commandes entre `.

```
var="test"  
newvar="Value of var is $var"  
echo $newvar
```

Value of var is test

- **Antislash (backstroke) ** : ne permet pas l'interprétation du caractère suivant.

```
var="Le prix est 10\$"  
cp file\ 1 file\ 2
```

Tubes/pipes

- Représentés par le symbole | :

commande 1 | commande 2 | commande 3

- ☐ commande 1 : écrire vers stdout
- ☐ commande 2 : lire stdin et écrire vers stdout
- ☐ commande 3 : lire stdin

- Utilisés pour relier des commandes entre elles : la sortie standard de la commande à gauche du symbole sera utilisée comme entrée standard de la commande à droite.
- Un tube crée un canal de communication entre deux processus.

Filtres

- Un **filtre** est une commande qui lit les données sur l'entrée standard, effectue des traitements sur les lignes reçues et écrit le résultat sur la sortie standard.
- Les entrées/sorties peuvent être redirigées et enchainées avec des tubes
- *Exemples de commandes filtres :*

Commande sort	
<code>sort < /etc/passwd</code>	Tri du fichier /etc/passwd
<code>cat /etc/passwd sort</code>	Tri du fichier /etc/passwd