

Chapitre 2: Le Shell, la gestion des fichiers

Chérifa Boucetta



Plan du cours

- Le shell Bash
- La gestion des fichiers
- Les filtres et les utilitaires

Le shell bash

- Le shell: interpréteur de commandes permet d'exécuter des instructions saisies au clavier ou au sein d'un script.
- Le shell fonctionne au sein d'un terminal.
 - interface utilisateur “de base” (interlocuteur avec le syst.)
 - interprétation ligne à ligne
 - plusieurs shell: sh, bash, ksh, ...

Syntaxe d'une commande

- La syntaxe d'une commande est généralement la suivante :

commande [options] [arguments]

- commande est le nom de la commande à exécuter
 - Les options modifient le comportement de la commande. Elles commencent habituellement par le caractère - (signe moins) suivi d'une ou plusieurs lettres.
 - Les arguments spécifient les objets (fichiers ou variables) sur lesquels la commande va s'appliquer.
- **N-B:**
 - Tous les shell font la **distinction** entre **les lettres minuscules et majuscules** pour les commandes et les noms de fichiers contrairement à MS-DOS.

Syntaxe des commandes Unix

- Attention : la console est sensible à la casse : « **d** » est différent de « **D** »

- `commande -D`

On exécute une commande avec comme option « **D** »

- `commande -d -a -e`

`commande -dae`

On exécute une commande avec plusieurs options, « **d** » « **a** » et « **e** »

La gestion des fichiers

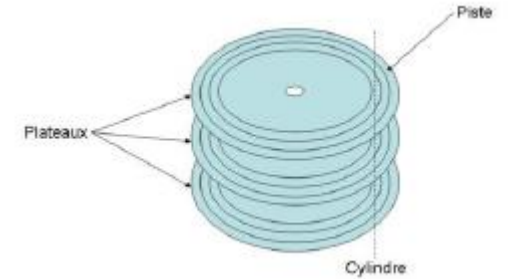
- Un fichier UNIX est une séquence d'octets organisé sous forme de blocs de taille fixe (512 octets dans les premiers systèmes et 4Koctets récemment).
- On distingue 3 types de fichiers :
 - Les fichiers ordinaires: texte, image, script, etc...
 - Les répertoires
 - Les fichiers spéciaux: servent d'interface pour les divers périphériques et se trouvent principalement dans le répertoire /dev s'ils représentent des périphériques

Opération sur les fichiers

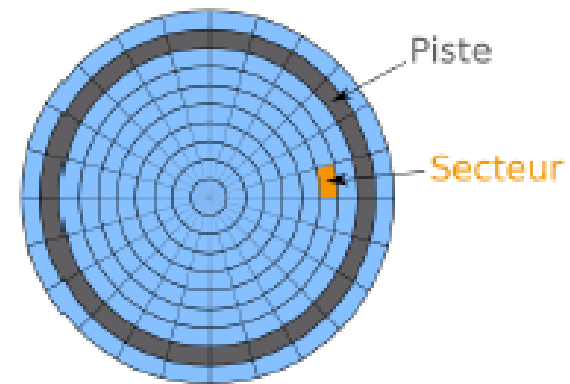
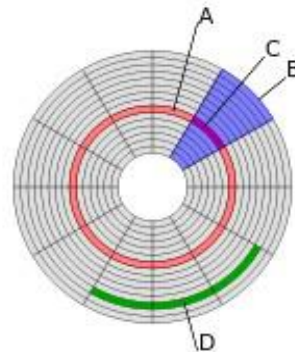
1. Création
2. Écriture : pointeur d'écriture → position d'écriture
3. Lecture : pointeur de lecture
4. Positionnement dans un fichier
5. Suppression : libération d'espace
6. Ajout d'informations
7. Renommage
8. Ouverture : le fichier devient associé à un processus qui en garde les attributs, position, etc.
9. Fermeture : ouverture et fermeture peuvent être explicites (open, close) ou implicites

Représentation sur le disque

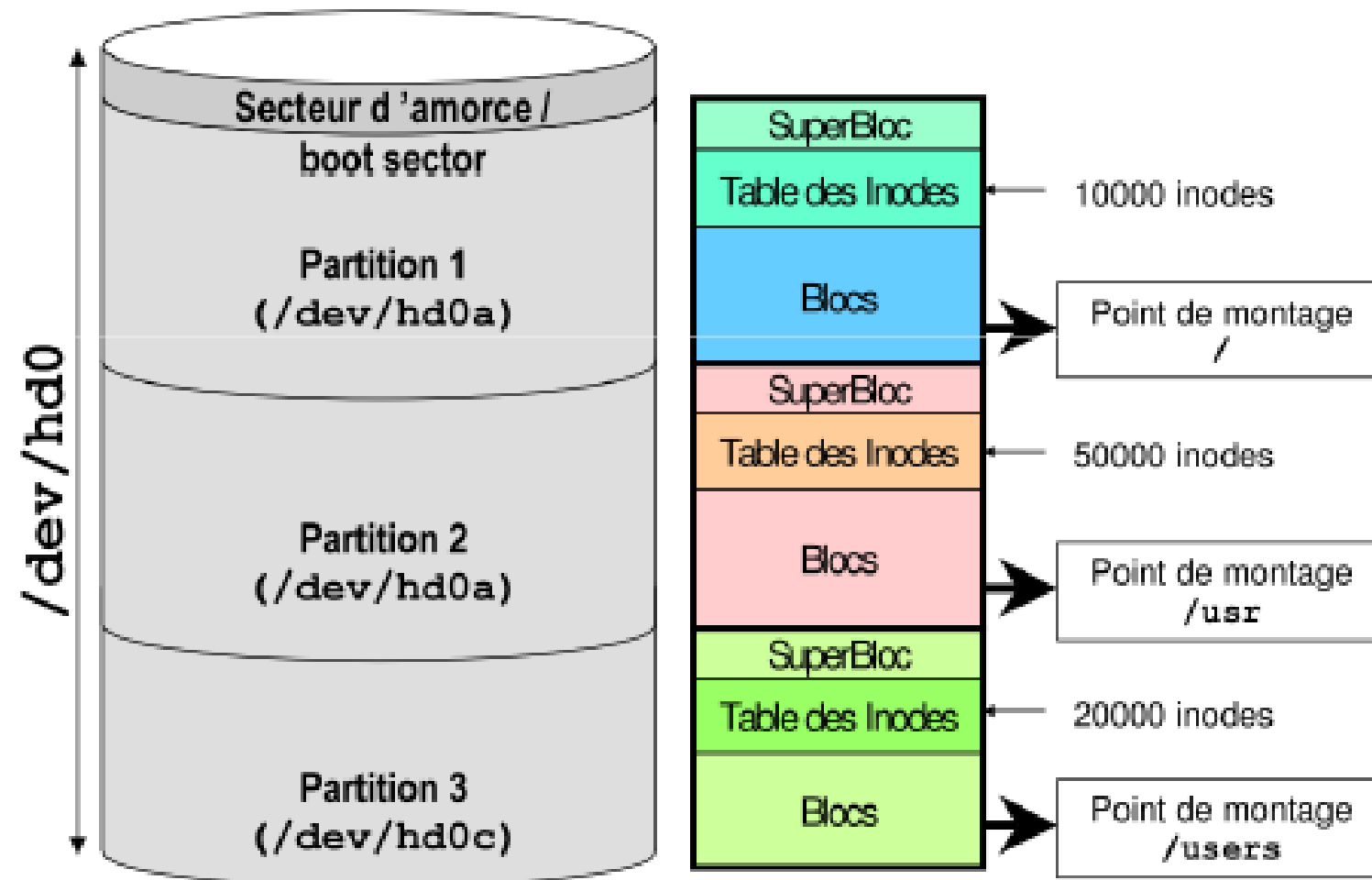
- Un disque dur est constitué de plusieurs plateaux,
- Chacun contient plusieurs pistes découpés en secteurs d'une taille donnée (C)



Structure d'un disque :
(A) piste
(B) secteur géométrique
(C) secteur de disque
(D) bloc

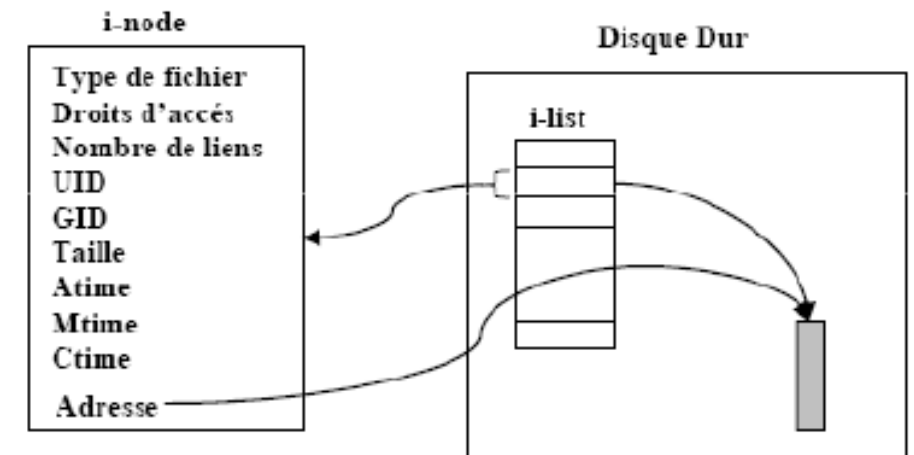


Organisation du disque



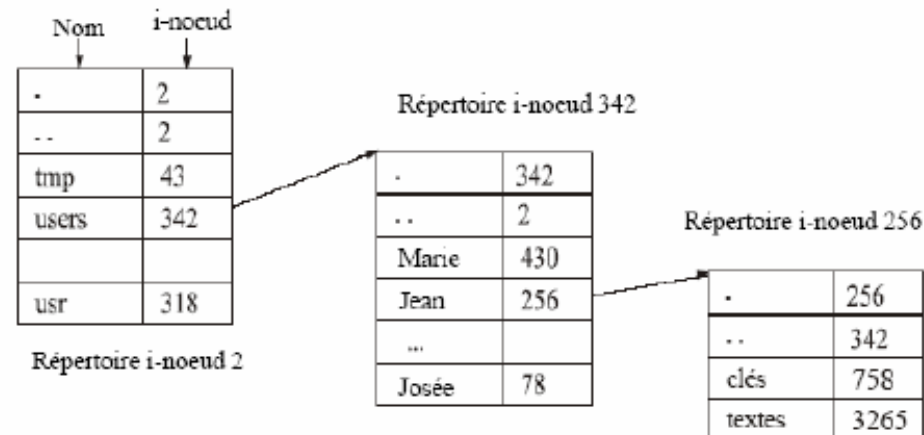
Représentation sur le disque

- La i-list est une table qui décrit l'ensemble des fichiers implantés sur un disque.
- Sa taille détermine le nombre de ses entrées. Elle est fixée à l'initialisation du disque.
 - Elle doit être proportionnelle au nombre maximum de fichiers autorisés sur ce disque.
- Chaque entrée de la i-list s'appelle un i-node structuré en 9 champs.
- La destruction d'une partie de la i-list signifie que les fichiers qui y sont décrits sont inaccessibles, mêmes s'ils sont parfaitement intègres sur le disque.



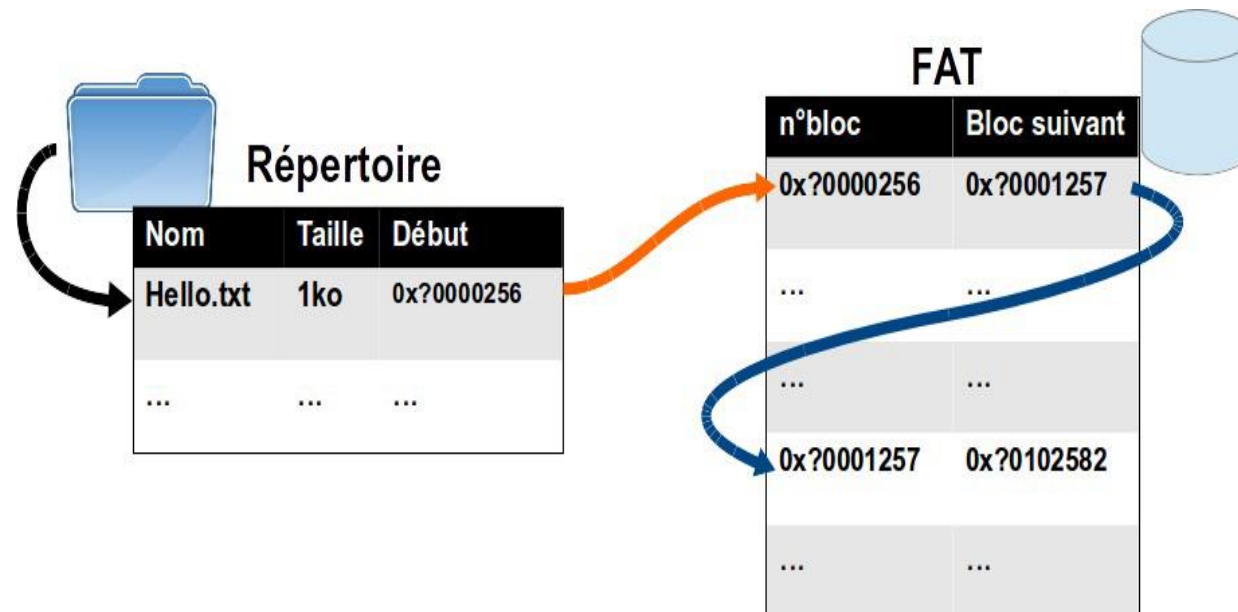
Représentation sur le disque

- Un fichier dépassant la taille d'un secteur est réparti sur plusieurs secteurs (si possible contigus),
 - La fin d'une partie est suivie par le numéro du secteur suivant.
 - Indiquer le secteur où commence un fichier suffit à lire le fichier en entier.
- Un dossier sera un pseudo-fichier contenant les associations « **nom de fichier / secteur initial** »

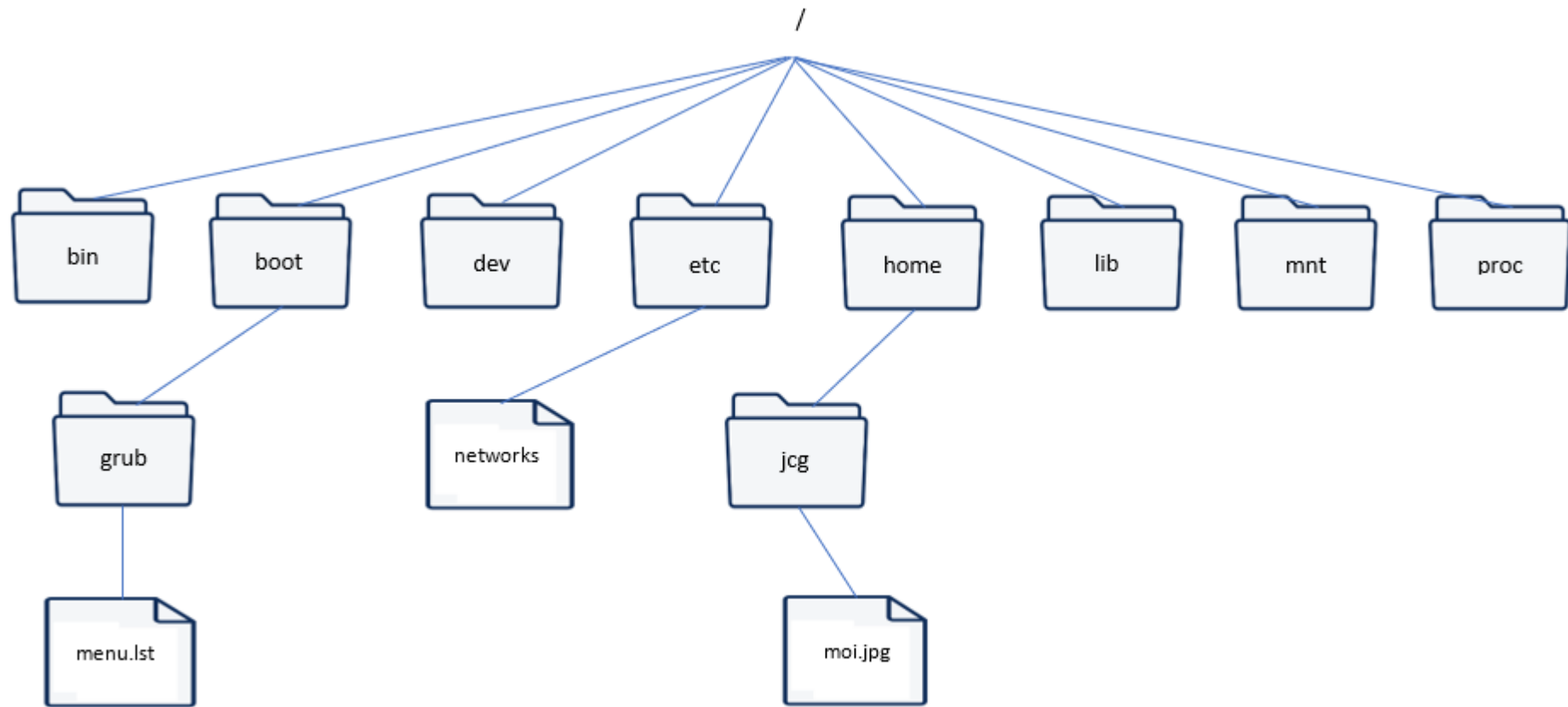


Principe de stockage

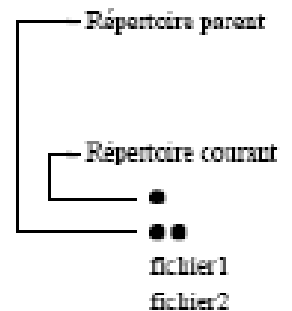
- Table d'allocation (FAT et FAT32)
 - Regroupe les index des \neq blocs des fichiers
 - Chaque rép table des noms \Rightarrow table des noms + index de début + index de début



Arborescence



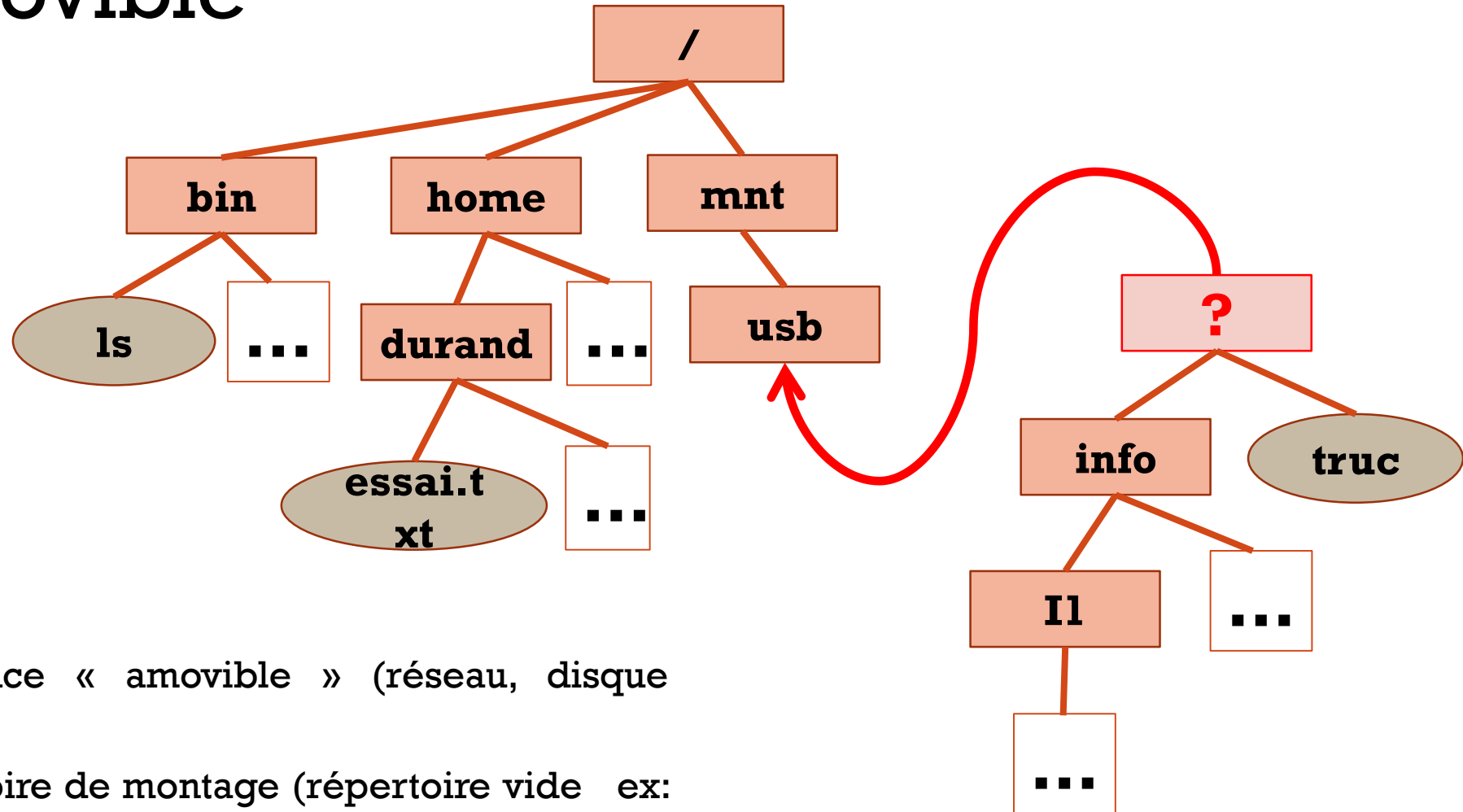
- La racine root est noté /
- '.' Correspond au répertoire courant et '..' au répertoire parent
- le répertoire de login: ~



Arborescence

- **/bin** : Contient les commandes système
- **/sbin** : Contient les commandes liées aux services
- **/boot** : Contient le noyau
- **/dev** : Contient les fichiers périphériques
- **/etc** : Contient les scripts de démarrage et les fichiers de configuration
- **/home** : Contient les répertoires des utilisateurs
- **/lib** : Contient les librairies
- **/tmp** : Espace de stockage temporaire
- **/usr** : Contient les applications installées
- **/var** : Contient les données des différents services

Support amovible



- Le point de montage:
 - Pour une arborescence « amovible » (réseau, disque amovible, clé USB,...)
 - Sous Linux : un répertoire de montage (répertoire vide ex: `/mnt/usb`)
 - La commande `mount`

Commandes de base – Se déplacer dans l'arborescence

- **Commande « cd »**

- Signification : Change directory
- But : naviguer dans les répertoires

- Exemples d'utilisation

- `cd` (sans argument) : permet de revenir au répertoire utilisateur (« **/home/toto** »). Cette commande a le même effet que `cd ~`
- `cd ..` : permet de remonter au répertoire parent
- `cd /` : permet de remonter à la racine
- `cd /usr/lib/` : on se place dans le répertoire « **/usr/lib** » (chemin absolu : commence par « / »)
- `cd Images` : on se place dans le dossier Images (chemin relatif) présent dans le répertoire courant

Commandes de base – Se déplacer dans l'arborescence

- **Commande « `pwd` »**
 - Signification : **Print Working Directory**
 - But : afficher le chemin absolu du répertoire actuel
- Exemples d'utilisation
 - `pwd` (sans argument) :

Commandes de base – Lister des fichiers

- Commande « **ls** »
 - Signification : **List**
 - But : **lister un répertoire** → afficher son contenu
- Options fréquentes :
 - `ls -l` : affichage détaillé
 - `ls -a` : affiche les fichiers et répertoire cachés
 - `ls -h` : avec l'option "**-l**", affiche les tailles de fichiers en ko/Mo/Go pour plus de lisibilité

Commandes de base – Manipuler des fichiers

- Commande « cp »
 - Signification : Copy
 - But : copier des fichiers ou répertoires ...
- Options fréquentes :
 - `cp -i src dest` : demande la permission avant d'écraser la destination
 - `cp -a src dest` : copie en conservant la date, les droits, propriétaire, groupe etc...
 - `cp -v src dest` : suivre la copie des fichiers
 - `cp -r src dest` : copie un répertoire et tout son contenu
- Exemples d'utilisation
 - `cp fichier.txt dossier` : copie « fichier.txt » dans le repertoire « dossier »
 - `cp -r dossier /chemin/conteneur` : copie le répertoire « **dossier** » dans « **/chemin/conteneur** » et créé le dossier conteneur s'il n'existe pas

Commandes de base – Manipuler des fichiers

- « **mv** » : déplacer des fichiers ou répertoires
- « **rm** » : supprimer des fichiers et dossiers (définitivement, sans passer par la corbeille !)
- « **mkdir** » : créer des dossiers

Commandes de base – Ouvrir des fichiers

- Commande « **cat** »
 - Signification : **Concatenate**
 - But : **afficher** un fichier ; **concaténer** plusieurs fichiers
- Exemples d'utilisation :
 - `cat -n fichier.txt` : affiche « fichier.txt » d'un coup en numérotant les lignes
 - `cat fichier1.txt fichier2.txt` : affiche « **fichier1.txt** » et « **fichier2.txt** » à la suite comme s'ils étaient collés (concaténation)

Manuel

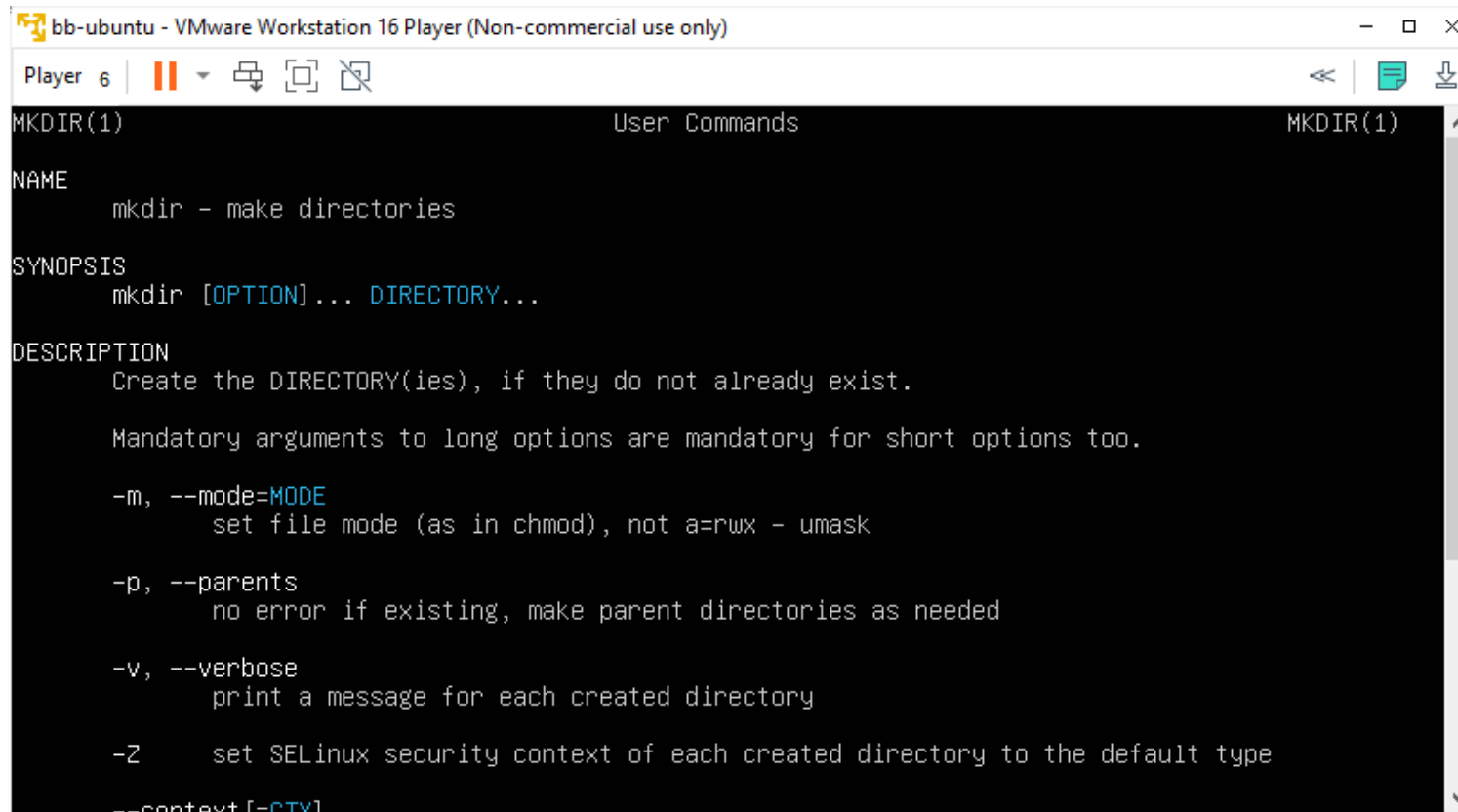
- Chaque commande possède un manuel !

```
man commande
```

- Affiche une page contenant des informations sur la commande avec une mise en page normalisée

Manuel

- Chaque commande possède un manuel !



The screenshot shows a terminal window titled "bb-ubuntu - VMware Workstation 16 Player (Non-commercial use only)". The terminal displays the manual for the 'mkdir' command. The window has a title bar with standard Linux window controls and a toolbar with icons for back, forward, and search. The terminal content is as follows:

```
MKDIR(1)                                User Commands                                MKDIR(1)
NAME
    mkdir - make directories

SYNOPSIS
    mkdir [OPTION]... DIRECTORY...

DESCRIPTION
    Create the DIRECTORY(ies), if they do not already exist.

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

    -m, --mode=MODE
        set file mode (as in chmod), not a=rwx - umask

    -p, --parents
        no error if existing, make parent directories as needed

    -v, --verbose
        print a message for each created directory

    -Z      set SELinux security context of each created directory to the default type

    --context[=CTY]
```

Exemple de commandes

- **ls** : liste les noms des fichiers et répertoires du répertoire courant
- **man** : obtenir de l'aide sur une commande à l'écran
- **cat** : affiche le contenu du (ou des fichiers) donnés en arguments
- **date** : affiche la date
- **echo** : affiche des lignes à l'écran
- **who** : affiche la liste des utilisateurs connectés au système
- **id** : affiche les informations d'identification d'un utilisateur
- **ps** : affiche la liste des processus en cours d'exécution
- **whoami** : affiche le login
- **Hostname** : affiche le nom de la machine
- **exit** : clôture de session

Caractères génériques

- Certaines commandes acceptent plusieurs noms de fichiers en arguments.
- Il existe plusieurs caractères génériques qui, incorporés dans les noms de fichiers, permettent d'avoir des notations de raccourci pour l'écriture d'une telle liste.
- Parmi ces caractères :
 - le caractère **?** qui peut remplacer n'importe quel caractère
 - le caractère ***** qui peut remplacer n'importe quelle chaîne de caractères . y compris la chaîne vide.

- Exemple

```
User1@localhost > ls
fich fich1 fich2 fich3 fich33 mbox rep1 rep2 rep3
User1@localhost > ls fich*
fich fich1 fich2 fich3 fich33
User1@localhost > ls fich??
fich33
```

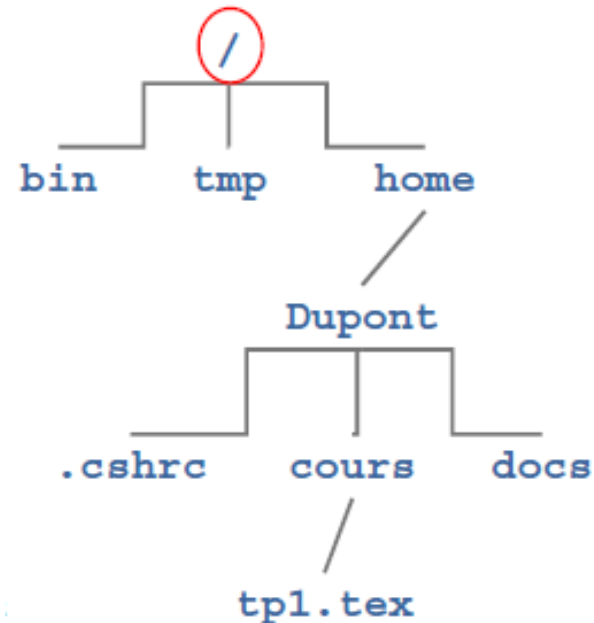
Caractères génériques

- On peut aussi utiliser des caractères bien précis ou bien une plage de caractère avec les []
 - [abc] : un caractère parmi a,bou c
 - [a-d] : un caractère dans la plage de a à d.
- **Exemple**
 - **ls -l *.txt** : affiche seulement les fichiers ayant l'extension txt
 - **rmtd[1-3].pdf** : effacer les fichiers td1.pdf, td2.pdf et td3.pdf
 - **cp/tmp/?2013.data .**
 - copier dans le répertoire courant tous les fichiers qui sont dans le répertoire /tmp dont le nom est composé d'une chaine de cinq caractères qui se termine par 2013 et qui ont l'extension data

Chemin absolu et chemin relatif

- Un **chemin absolu** ou complet :
 - démarre de la racine, donc commence par un **/**.
 - décrit tous les répertoires à traverser pour accéder à l'endroit voulu.
- Tout chemin qui ne commence pas par **/** est interprété comme **un chemin relatif** au répertoire courant.

- Exemple:
 - Chemin d'accès au fichier `tp1.tex`:
 - **`~#cat /home/Dupont/cours/tp1.tex`**
 - **`~#cat cours/tp1.tex`**



Lien physique et lien symbolique

- Les liens sont utiles pour faire apparaître un même fichier dans plusieurs répertoires ou sous noms différents.
- Liens physiques ou liens durs
 - Associent deux ou plusieurs fichiers à un même espace sur le disque, les fichiers restent indépendants. Il permet de donner plusieurs noms à un fichier.

#ln <nom_fic> <nouveau_nom_fic>

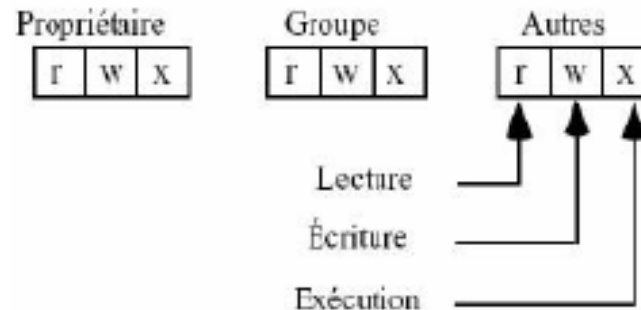
- Liens symboliques
- Le lien symbolique fait référence à un fichier dans un répertoire.

#ln -s <nom_fic> <nouveau_nom_fic>

- crée un **raccourci**
- fonctionne aussi pour les répertoires

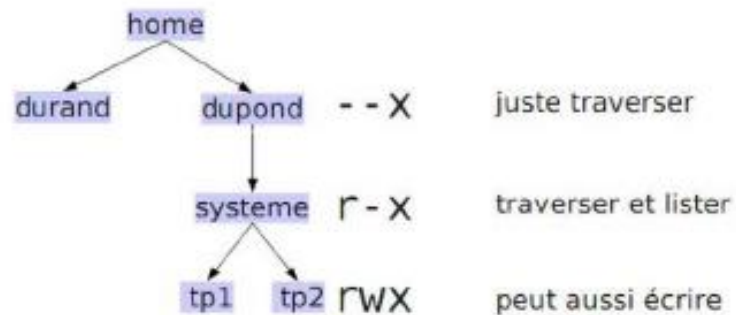
Droits d'accès

- La protection d'un élément repose sur trois droits d'accès qui contrôlent les trois opérations de lecture (droit **r** pour **read**), d'écriture (droit **w** pour **write**) et d'exécution (droit **x** pour **execute**).
- Ces trois droits (**rwX**) sont appliqués à trois catégories d'utilisateurs qui sont le propriétaire (**user : u**) de l'élément, son groupe (**group : g**) et les autres utilisateurs (**others : o**).
- La notion de groupe correspond simplement à un ensemble d'utilisateurs auxquels on peut attribuer les mêmes droits d'accès aux fichiers.



Droits d'accès

- Un groupe est un ensemble d'utilisateurs ayant les mêmes droits d'accès par rapport aux fichiers d'un utilisateur
- Par exemple : `rwX r-x r--` signifie :
 - **Lecture, écriture et exécution** autorisées pour le **propriétaire**
 - **Lecture et exécution** autorisées pour le **groupe**
 - **Lecture seule** autorisée pour les **autres**

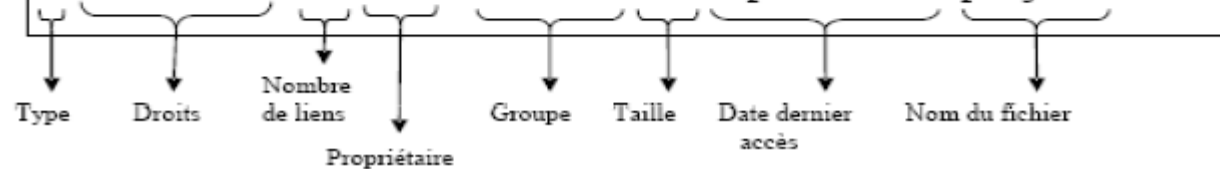


Code	répertoires	fichiers
r (lecture)	Explorer.	Voir le contenu.
w (écriture)	Ajouter ou supprimer des fichiers	Modifier le contenu.
x (exécution)	Autorise l'accès au repertoire	Exécuter.

Droits d'accès

- Pour visualiser les droits d'accès, on utilise la commande `ls -l`.
- Le 1er caractère spécifie le type du fichier. Les 9 caractères suivants identifient les droits d'accès

```
User1@localhost > ls -l
total 7
-rw- --- --x  2 user1 etudiant 49 Sep 18 12:37 file
-rw- r-- r-x  1 toto  etudiant 50 Sep 18 12:35 devoir.c
-rw- rw- r-x  1 toto  etudiant 70 Sep 18 12:30 concertir.sh
lrw- --- --x  2 toto  etudiant 82 Sep 18 12:40 fic3
drwx rwx --x  1 user2 etudiant 95 Sep 18 12:50 projet
```



Type	Droits	Nombre de liens	Propriétaire	Groupe	Taille	Date dernier accès	Nom du fichier
-rw-	--- --x	2	user1	etudiant	49	Sep 18 12:37	file
-rw-	r-- r-x	1	toto	etudiant	50	Sep 18 12:35	devoir.c
-rw-	rw- r-x	1	toto	etudiant	70	Sep 18 12:30	concertir.sh
lrw-	--- --x	2	toto	etudiant	82	Sep 18 12:40	fic3
drwx	rwx --x	1	user2	etudiant	95	Sep 18 12:50	projet

Droits d'accès

- Seul, le propriétaire d'un fichier ou le super utilisateur (root) peut modifier les droits d'accès d'un fichier.
 - il utilise la commande `chmod` avec une description octale ou symbolique.
- Pour la description symbolique `chmod` a la syntaxe suivante :

`#chmod [who]op[permission] nom_fichier`

- Où :
 - **who** est une combinaison de lettre u (user), g(group), o(others) ou a (all) pour ugo. Si aucune classe n'est spécifiée. toutes les classes sont concernées.
 - **op** est l'un des symboles (+ ajouter un droit d'accès, - supprimer un droit d'accès et = pour affecter un droit absolu : tous les autres bits sont remis à zéro).
 - **Permission** est une combinaison des lettres r, w, x.

Droits d'accès

- Pour la description octale, une lettre est équivalente à 1 et un tiret est équivalent à 0 en binaire.
- Chaque 3 bits sont codés à part. Par exemple, Le fichier /etc/passwd ayant les droits rw-r--r-- ce qui est équivalent à 110 100 100 = 644.
- **chown** : permet de changer le propriétaire d'un fichier ou d'un répertoire (root)
- **chgrp** : permet de changer le groupe d'un fichier ou d'un répertoire (root)

#chown [options] ident_utilisateur fich

#chgrp [options] ident_groupe fich

Changer les droits d'un fichier

- `chmod g-x fichier` : Supprime les droits d'exécution du fichier pour le groupe propriétaire du fichier
- `chmod go+rw fichier` : Ajoute les droits de lecture, écriture, exécution sur le fichier pour le groupe propriétaire du fichier et les autres utilisateurs
- `chmod go=r fichier` : fixe les permissions en lecture seulement pour le groupe propriétaire et les autres utilisateurs

Changer les droits d'un fichier

- `chmod u=rw,go=r fichier` : Fixe l'autorisation de lecture et d'écriture au propriétaire de « fichier » et une autorisation de lecture au groupe et aux autres.
- `chmod 644 fichier` : fait exactement la même chose que précédemment (6 = lecture + écriture ; 4=lecture en octal).
 - valeur numérique calculée sur le poids de **r**, **w** et **x** pour chaque catégorie : **r = 4**, **w = 2** et **x = 1**
- `chmod 741 fichier` → `rwX r-- --X`
- `chmod 600 fichier` → `rw- --- ---`
- `umask` : filtre sur les permissions par défaut

```
# umask 077
```

Exemple :

Par défaut, `umask = 022`

Création d'un fichier

`666 - 022 ?`

`= 644`

Plan du cours

- Le shell Bash
- La gestion des fichiers
- Les filtres et les utilitaires

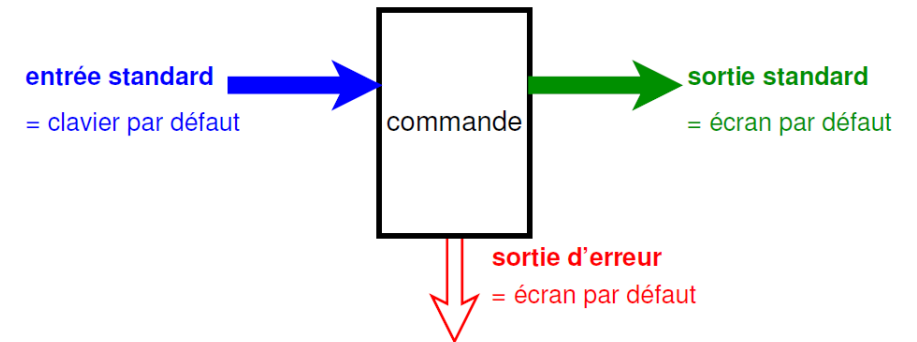
Les filtres

- Un **filtre** est un programme sachant écrire et lire des données par les canaux standards d'entrée et de sortie.
 - L'entrée peut provenir d'un fichier, de l'entrée standard, d'une redirection, etc.
 - La sortie peut être un fichier, la sortie standard, une redirection, l'entrée d'un tube, etc.
- Différents catégories de filtres:
 - Des filtres simples réalisant un traitement élémentaire: le tri
 - Des filtres simples utilisant en plus la notion d'expression régulière pour caractériser des chaînes de caractères.
 - Des filtres programmables capables d'effectuer diverses actions, comme l'éditeur de flots sed.

Flots et commandes d'entrées/sorties

- Les fichiers sont considérés comme des flots
 - Encadrés par des opérations de lectures et d'écritures
- 3 flots supplémentaires prédéfinis :
 - **Entrée standard** : clavier par défaut (descripteur n°0)
 - **Sortie standard** : écran par défaut (descripteur n°1)
 - **Sortie en erreur** : écran par défaut (descripteur n°2)

Commande UNIX \Rightarrow trois flux standard de données :

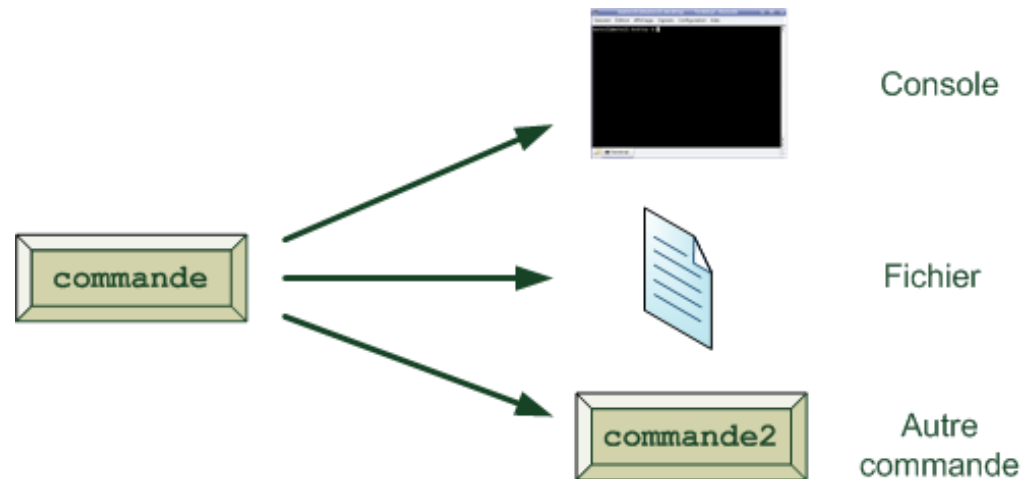


La redirection de flots

- Principe : passer les informations d'un flot vers un autre
- Opérateur « | » (**pipe**) : redirection sortie standard vers entrée standard
 - Remarque : les commandes exécutées dans un pipeline sont exécutées dans un sous shell
- Opérateur « < » : redirection fichier vers entrée standard
- Opérateur « > » : redirection sortie standard vers fichier
- >>: change la sortie standard pour l'ajouter à la fin d'un fichier existant.
- ||: exécuter la commande suivante si la première a échoué.
- && : n'exécuter la commande suivante que si la première a réussi.

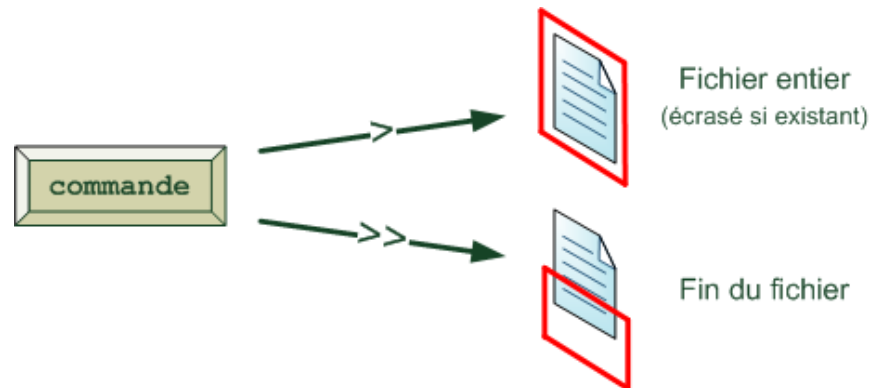
Commandes spécifiques aux entrées/sorties

- `echo texte` : affiche le texte sur la sortie standard
- `tee [nom]` : entrée standard → sortie standard ou dans "nom"
- `read [nom]` : lecture au clavier (nom peut contenir plusieurs variables séparées par des espaces)



La redirection stdout

- Très souvent, on redirige vers un fichier
- Redirection en sortie :
- `commande > fich` : le fichier `fich` est créé (effacé s'il existe déjà)
- `commande >> fich` : la sortie de commande est ajoutée à la fin de `fich`



La redirection de flots

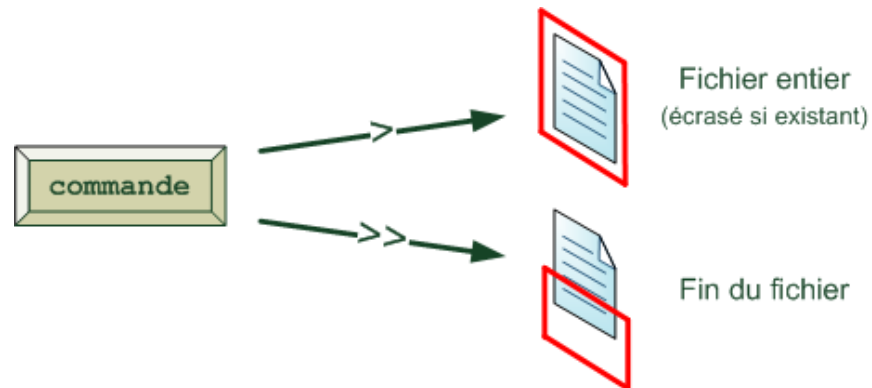
- Utilisation possible des flots prédéfinis par leurs descripteurs

```
[descripteur_flot_de_sortie]>fichier
```

- Pas d'espace avant et après le symbole « > »
- Les 2 flots de sortie (std et err) sont redirigeables
 - `cmd >fichier` : flots standard et d'erreur vers le fichier
 - `cmd 2>fichier` : uniquement du flot d'erreur vers le fichier
 - `cmd 1>fichier` : uniquement du flot standard vers le fichier
 - `cmd 2>&1` : flot d'erreur vers le flot standard (fusion des flots)

La redirection stdout

- Très souvent, on redirige vers un fichier
- Redirection en sortie :
 - `commande > fich` : le fichier `fich` est créé (effacé s'il existe déjà)
 - `commande >> fich` : la sortie de commande est ajoutée à la fin de `fich`



La redirection stdin

- **stdin(en général clavier)**
 - `<`: lire depuis un fichier
 - `<<`: lire depuis le clavier progressivement

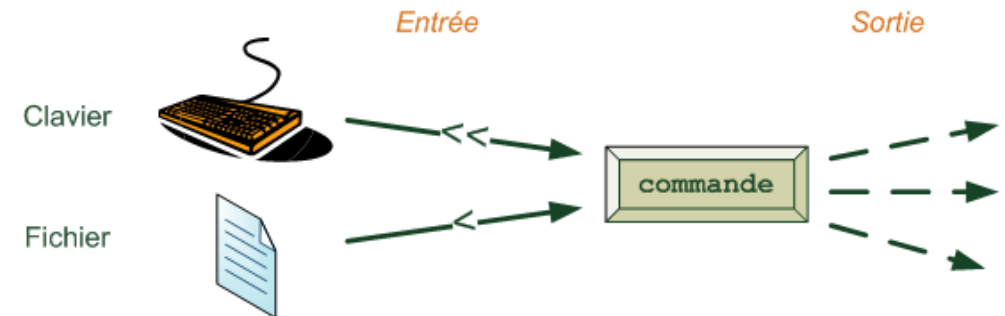
- **Exemple:**

- `wc-m<<FIN`

>Combien de caractères dans cette phrase?

>FIN

42



La redirection: Le pipe

- Le pipe (|) effectue la connexion entre la sortie d'une commande et l'entrée d'une autre.
- Exemple :

`cat notes.csv | sort`



Recherche de lignes

- Il s'agit d'extraire des lignes d'un fichier selon divers critères.
- Commande: **grep** [Options] modèle [Fichier1...]
 - Le modèle se compose de critères de recherche
- **Options:**
 - **-v** effectue la recherche inverse: toutes les lignes ne correspondant pas aux critères sont affichées.
 - **-c** ne retourne que le nombre de lignes trouvées sans les afficher.
 - **-i** ne différencie pas les majuscules et les minuscules.
 - **-n** indique le numéro de ligne pour chaque ligne trouvée.
 - **-l** dans le cas de fichiers multiples, indique dans quel fichier la ligne a été trouvée.

Recherche de lignes

- **Exemples :**

- **grep user1 /etc/passwd** → affiche la ligne de user1 dans le fichier de mots de passe
 - **grep '^#' test.py** → affiche les lignes commençant par # dans test.f90 (commentaires)
 - **grep -v '^#' test.f90** → affiche les lignes qui ne sont pas des commentaires
 - **grep -v '^ *\$' test.f90** → affiche les lignes qui ne comportent pas que des blancs
 - **ls -l | grep ^d** → affiche la liste des sous-répertoires du répertoire courant avec leurs attributs
 - **grep '[0..9]' essai.txt** → rechercher les lignes contenant un chiffre
-
- N.-B. : protéger les caractères spéciaux de l'interprétation par le shell, ici par des «'»

Les filtres simples

- Visualisation du texte:
 - **more, less**: page par page
 - **cat**: par bloc (cat -n : numéroté les lignes)
 - **tee**: Cette commande affiche l'entrée standard simultanément sur la sortie standard et sur le ou les fichiers passés en arguments

Exemple: `~# ls | tee liste_fichier`

- **wc**: (word count) permet de compter les lignes, les mots et les caractères.

wc [-l] [-c] [-w] ficl

- **-l** : compte le nombre de lignes.
- **-c** : compte le nombre d'octets.
- **-w** : compte le nombre de mots.

Les filtres simples

- **tail**: affiche la dernière partie (par défaut les 10 dernières lignes) de chacun des fichiers indiqués en paramètres.

tail [- position [lc]] [nom_fichier]

- **head**: affiche le début du fichier passé en argument.
 - Par défaut, head affiche les dix premières lignes mais on peut préciser en option, un nombre quelconque de lignes.

head -10 mon-fichier

- On obtient les 10 premières lignes à partir du début.

Les filtres simples

- **tr**: permet de substituer des caractères à d'autres

`tr [options] original destination`

- Exemple: `~# cat liste | tr "[a-z]" "[A-Z]"`
- **cut**: sélectionner des colonnes et des champs dans un fichier.

`cut -c liste [nom_fichier]`

`cut -f liste [-d délimiteur] [-s] [nom_fichier]`

- Options :
 - `-c` : extrait suivant le nombre de caractères
 - `-f` : extrait suivant le nombre de champs
 - `-dx` : Le caractère x est le séparateur de champ
 - `-s` : supprime les lignes ne présentant pas de délimiteur

Les filtres simples

- **sort** permet de trier des lignes

`sort [options] [-k pos1[,pos2]] [fic1...]`

- `-u` : supprime les lignes doublons
- `-b` : ignore les espaces et les tabulations en début de champ
- `-r` : inverse l'ordre de tri
- `-n` : trie sur des chiffres
- `-tx` : Le caractère x est considéré comme séparateur de champ
- `-f` : confond les majuscules et les minuscules
- Exemple: `~# sort -n -r -k 3 liste` → tri numérique sur le prix par produits en ordre décroissant

ecran 19 500 20

ecran 17 300 20

carte video 145 30

clavier 115 55 30

- **uniq** permet de supprimer les doublons dans des flux en entrée ou des fichiers triés

Les filtres simples

- La commande `cmp` indique si deux fichiers sont identiques.
- Syntaxe :

`cmp fichier1 fichier2`

- Si les deux sont identiques, la commande ne génère aucune sortie, s'ils sont différents la commande indique la position de la première différence (ligne et caractère), avec une sortie du genre :

`fichier1 fichier2 differ : char 34, line 2`

Les filtres simples

- Cette commande permet de retrouver dans un répertoire ou une hiérarchie de répertoires les fichiers possédant certaines caractéristiques (noms, droits, date...) ou satisfaisant une expression booléenne donnée.

find chemin expression(s) [action]

- **chemin** : endroit où chercher ; "." si on veut chercher à partir du dossier courant.

Expression (s)	Critères de recherche
-name nom	Nom du fichier
-iname nom	Nom du fichier, sans tenir compte de la casse
-user nom	Le propriétaire du fichier est nom
-type [d f l ...]	Fichier de type répertoire « d », normal « f », lien l ...
-perm nummode	Fichier ayant les permissions nummode

Les filtres simples

find chemin expression(s) [action]

Action	Description
-print	Affiche le nom complet du fichier
-exec commande {} \;	Exécute la commande spécifiée sur chaque fichier ; { } représente successivement chaque fichier trouvé \; indique que l'action -exec est finie

Fin du Chapitre 2