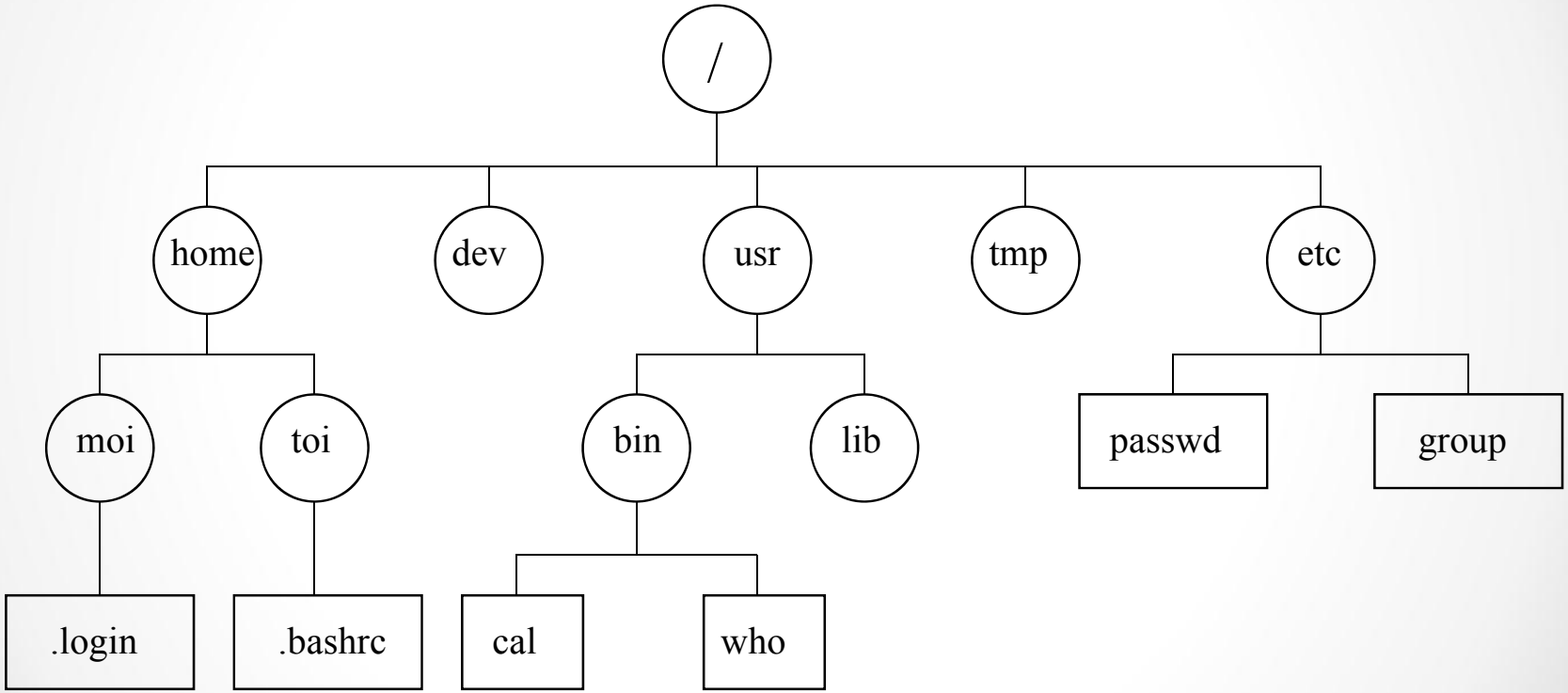
Exemple de système de fichiers sous Unix



Sous Linux, toute l'arborescence part de « / », appelé « root » ou racine. Dans ce répertoire racine se trouvent tous les dossiers élémentaires

Quelques répertoires

/bin	les binaires (exécutables) vitaux pour l'utilisateur
/dev	tous les périphériques
/home	les répertoires personnels des utilisateurs
/mnt	les dossiers pour les points de montage temporaires
/proc	accès direct aux paramètres du noyau
/root	répertoire personnel du super utilisateur (root)
/var	les fichiers qui changent fréquemment, tels que : les logs, les mails, les queues d'impression

Les chemins

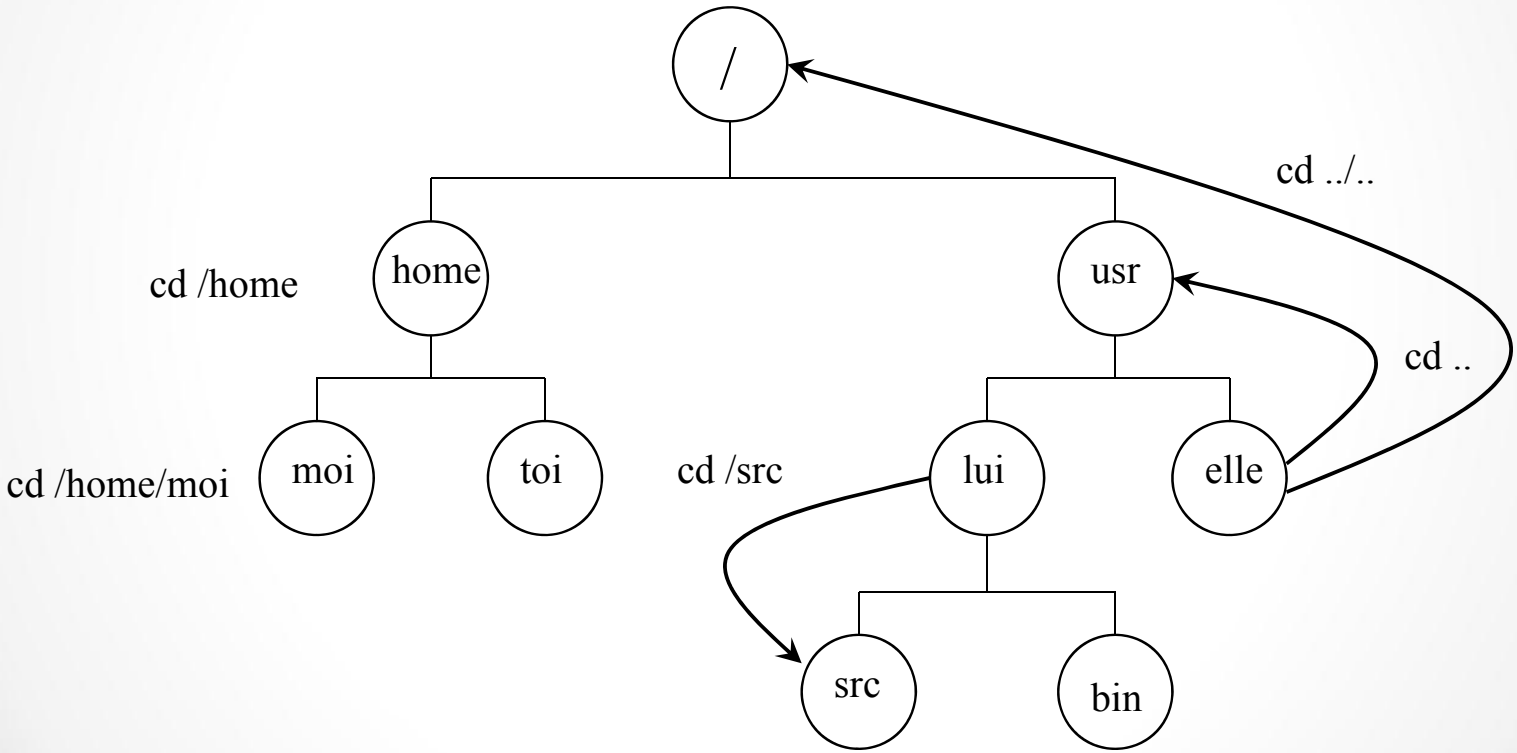
- Le chemin d'**accès absolu** décrit l'itinéraire à emprunter depuis la racine de l'arborescence jusqu'au fichier.
- Le chemin d'**accès relatif** décrit l'itinéraire à emprunter depuis le répertoire de travail courant. Si le fichier à atteindre est situé au dessous du répertoire de travail, l'utilisateur doit mentionner le chemin qu'il reste à parcourir.
- Le symbole « **.** » désigne le répertoire courant et le symbole « **..** » sont père
- Attention : le symbole « . » devant un nom de fichier signifie qu'il est caché.

Exemple de chemins

- Par exemple **/usr/bin/cal** est le chemin absolu du fichier ***cal***.
- Si le répertoire de travail est le répertoire **/home/moi** :
  - **.login** est équivalent au chemin absolu **/home/moi/.login**
  - **./.login** est équivalent au chemin absolu **/home/moi/.login**
  - **../toi/.cshrc** est équivalent au chemin absolu **/home/toi/.cshrc**

NB : Le symbole «~» désigne le répertoire de connexion de l'utilisateur. Par exemple, si l'utilisateur est **moi**, alors **~/.login** correspond au chemin relatif **/home/moi/.login**

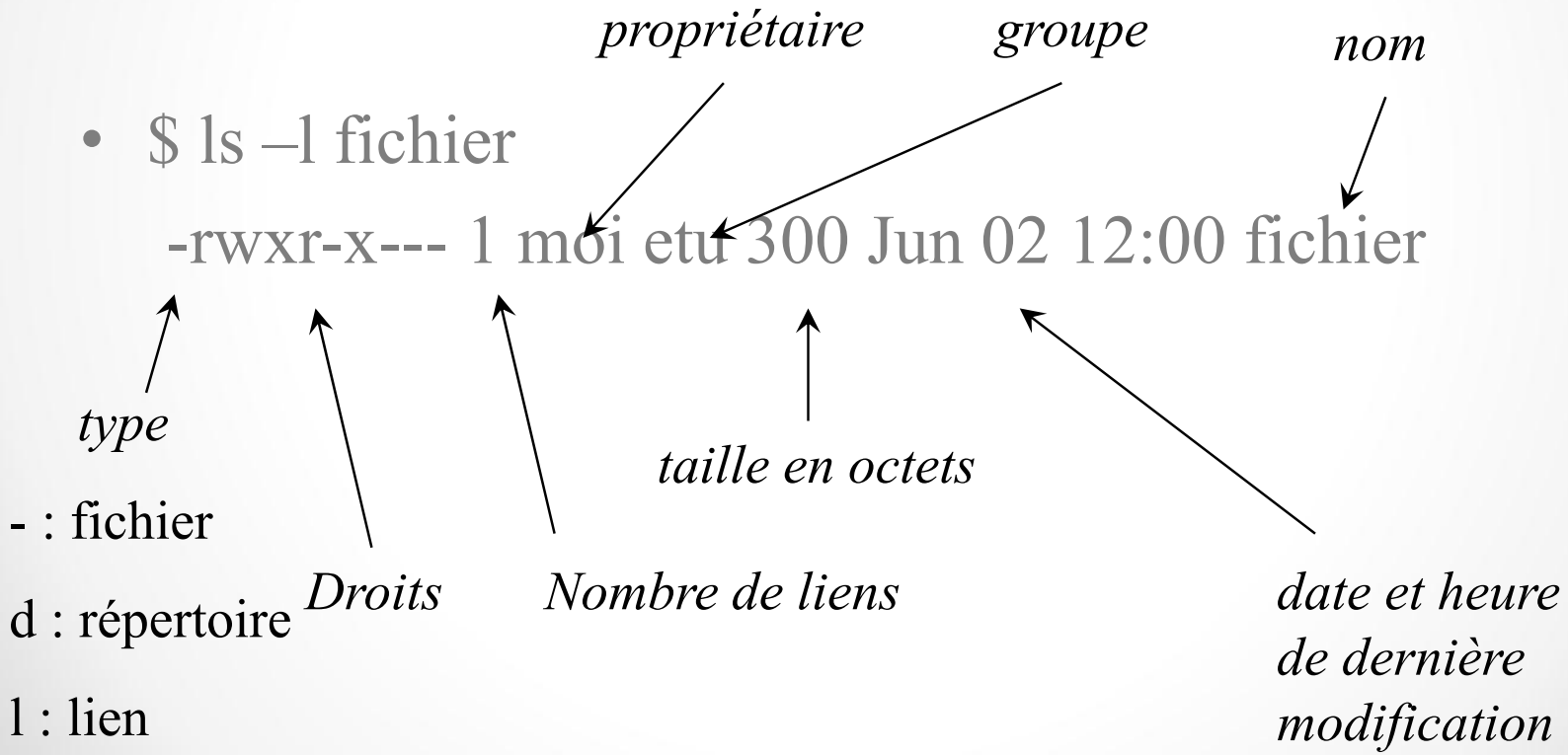
La commande cd



Les attributs des fichiers

- $ ls –l fichier

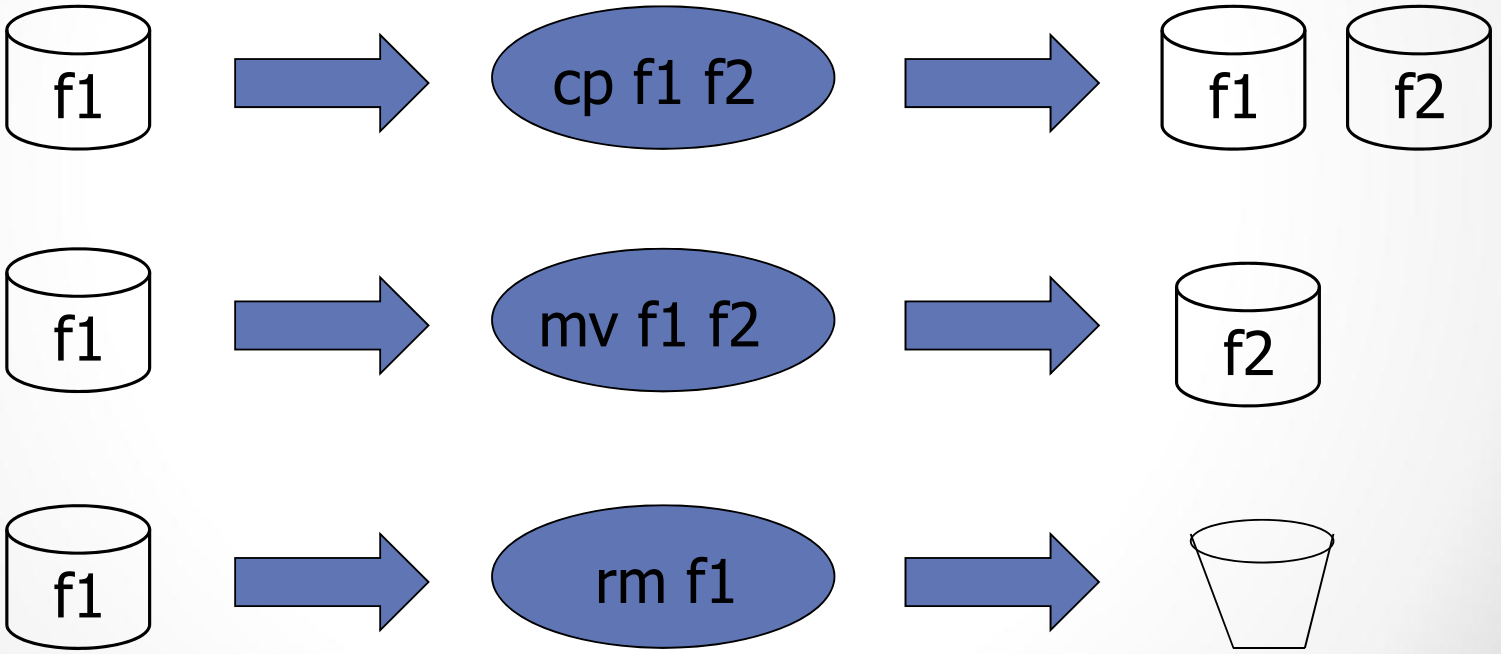
  -rwxr-x--- 1 moi etu 300 Jun 02 12:00 fichier



Quelques commandes pour les fichiers

- **cp** : copie d'un fichier.
- **rm** : détruit un fichier.
- **mv** : change le nom d'un fichier, déplace un fichier.
- **cat, more** : affiche le contenu d'un fichier .
- **file** : affiche le type du fichier.
- **cmp, comm, diff** : compare des fichiers.

Copier, détruire, renommer un fichier



Quelques commandes pour les répertoires

- **pwd** : affiche le répertoire courant.
- **mkdir** : crée un répertoire.
- **rmdir** : supprime un répertoire.
- **du** : affiche la taille d'une arborescence.
- **find** : recherche des fichiers dans une arborescence.

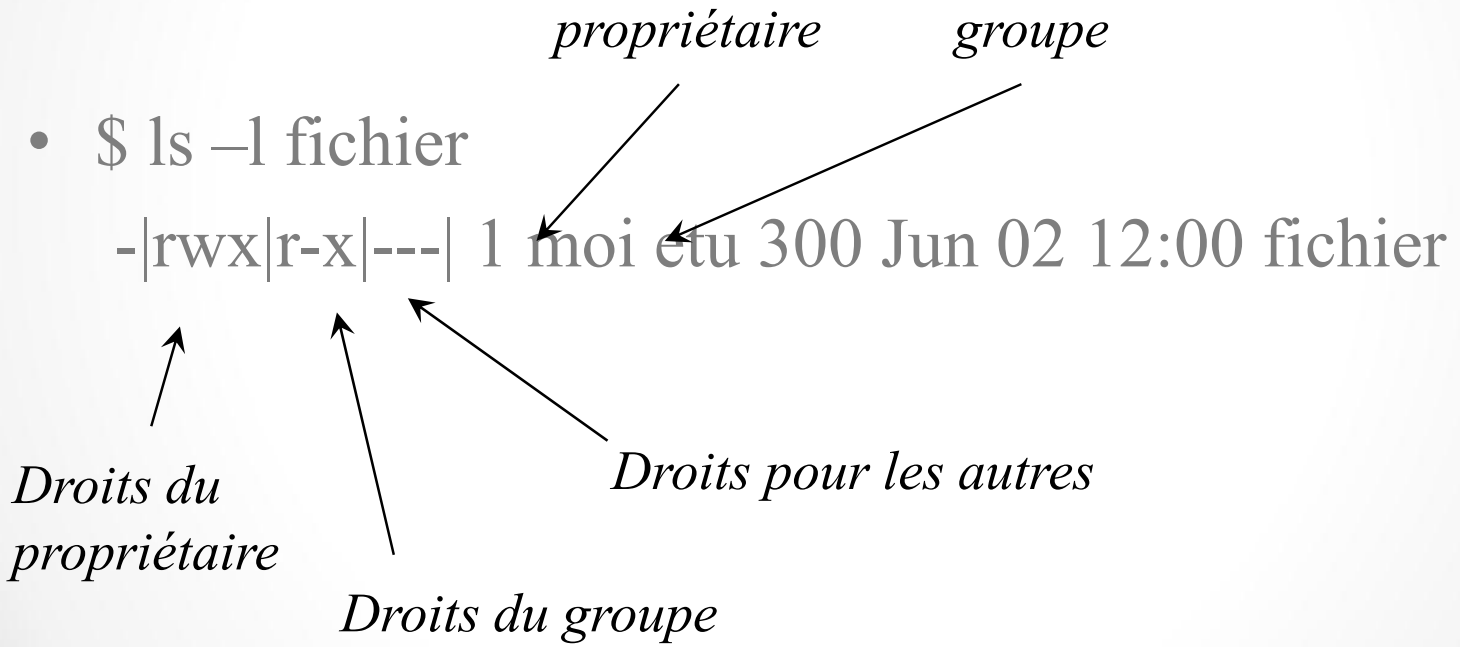
Utilisateurs et groupes, protections d'accès

- **id** : Affiche l'identité de l'utilisateur.
- **chmod** : change les droits d'un fichier.
- **chrgp** : change le groupe d'un fichier.
- **su** : change l'identité d'un utilisateur.
- **newgrp** : change le groupe de l'utilisateur
- **chown** : change le propriétaire d 'un fichier ou d 'un répertoire

Connaître les droits

- $ ls –l fichier

  -|rwx|r-x|---| 1 moi etu 300 Jun 02 12:00 fichier



Droits sur les fichiers et répertoires

- **Notation symbolique**
  - Qui ?
    - U = user (propriétaire)
    - G = group (le groupe)
    - O = others (les autres)
  - Permission
    - R = read (consulter le contenu)
    - W= write (modifier et détruire)
    - X = execute (éxecuter un script ou un programme)

Droits sur les fichiers et répertoires

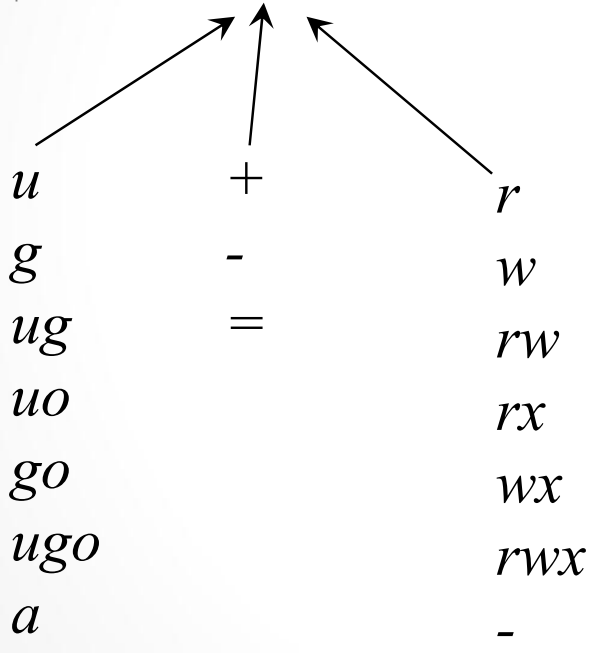
- **Notation octale**

  - u, g et o sont remplacés par des valeurs octales selon la règle suivante :
    - 0 = aucun droit
    - 1 = exécution (x)
    - 2 = écriture (w)
    - 4 = lecture (r)

  - exemple : $ chmod 400 fichier

Droits sur les fichiers et répertoires

$ chmod u-w fichier



$ chmod 400 fichier

Droit de lecture pour le propriétaire

Droits sur les fichiers et répertoires : exemples

- **$ chmod 751 exemple.c**

  -rwxr-x--x 1 retraint users 1829 Mar 18 2002 exemple.c

  rwx = 4+2+1 = 7 (utilisateur)
  r-x = 4+1 = 5 (groupe)
  --x = 1 (autres)

$ chmod 700 ~

Permet de transformer son répertoire de connexion en répertoire privé

Liens physiques et symboliques

- La commande « ln » permet de créer un lien pointant vers un fichier (à la manière des raccourcis de Windows).
- Le lien apparaît lorsque l ’on passe la commande « ls -l ».
- Sans l ’option « -s », « ln » crée un lien physique (hard link) au lieu d ’un lien symbolique.
- Les liens symboliques permettent de ne pas dupliquer l ’espace disque utilisé par les fichiers.
- Pour supprimer un lien, on utilise la commande « rm ». C ’est bien le lien et non le fichier pointé qui est détruit.
    - $ ln -s **fichier_ou_répertoire** *lien*

Liens physiques et symboliques : exemple

- $ ln -s fich fich2

  $ ls - l

-rw - r- -r- - 1 retraint users 1829 Mar 5 14:25 fich

lrwxrwxrwx 1 retraint users 4 Mar 9 14:03 fich2 ->fich

  La dernière ligne indique le lien symbolique entre les fichiers fich et fich2

- $ rm fich
- cat fich2

  Cat : fich2 : No such file or directory

Les redirections des entrées sorties

Il existe des méthodes pour rediriger les entrées et

les sorties des commandes

- Les redirections standard
  - le signe > redirige la sortie vers le fichier dont le nom suit (et le remplace)
  - le signe >> redirige la sortie vers le fichier dont le nom suit (ajout à la fin du fichier)
  - < redirige l'entrée vers le fichier dont le non suit
  - << permet de donner les arguments à une commande jusqu'au marqueur désigné

Les redirections des entrées sorties

- Rediriger l'entrée standard (le clavier)
  $ commande [options] [arguments] < fichier
  Les données qui alimentent la commande viennent du fichier

- Rediriger la sortie standard (l'écran)
  $ commande [options] [arguments] > fichier
  les résultats produits par la commande sont écrits dans le fichier

Les redirections : exemples

```
$ cat fich1 >fich2
$ cat <<FIN >sortie
> Bonjour,
> Comment ça va ?
> Au revoir.
> FIN
$ cat sortie
Bonjour,
Comment ça va ?
Au revoir.
```

Les redirections des erreurs

- Les commande envoient les lignes de texte qui ont valeur de résultat sur la sortie standard et les messages qui signalent une erreur de syntaxe ou d'exécution sur la sortie des standard des erreurs.

- Pour rediriger la sortie standard des erreurs produites par une commande, il faut mentionner dans la ligne de commande 2>fichier_de_redirection

  $ ls –l 2>/dev/null

- Pour ignorer les messages d'erreur qui parasite la lisibilité des résultats on les redirige vers le périphérique /dev/null. Elle sont perdues

Les tubes

Le tube, exprimé par le symbole |, est un mécanisme de communication qui permet d'échanger des données entre commandes.

commande1 | commande2 | commande3

$ ls –l | more : affiche le résultat de la commande ls, page par page

$ ls –l | lp : imprime le résultat de la commande ls

Les alias

L'utilisation des alias est un moyen pratique de définir des synonymes de commandes avec des options intégrées.

Créer un alias :	$ alias dir='ls –lba'
Utiliser un alias :	$ dir
Afficher la liste des alias :	$ alias
Détruire un alias :	$ unalias

Visualisation du contenu des fichiers

- La commande « cat » permet de visualiser le contenu des fichiers.
  - $ cat fich1 fich2
- « pg », « more » et « less » permet de visualiser un fichier page à page et permettent de se déplacer dans le fichier.
  - exemples des commandes dans « less » :
    - <espace> : page suivante.
    - /chaîne : cherche la chaîne de caractères spécifiée.
    - q : pour quitter le fichier.

Visualisation du contenu des fichiers

- « head » permet de visualiser un nombre de lignes du début d ’un fichier.
- « tail » traite la fin du fichier.
- « tail -f » permet de demander l ’affichage d ’un fichier au fur et à mesure que de nouvelles lignes sont créées.
- « wc » permet de compter les caractères, mots, lignes d ’un fichiers (« -c » : caractères, « -w » : mots, « -l » : lignes). Sans option, la commande « wc » donne les 3 informations.
- « file » informe sur le contenu du ou des fichiers.
- « nl » numérote les lignes d’un fichier

Visualisation du contenu des fichiers : exemples

- *$ head –3 fich*

ligne 1
ligne 2
ligne 3
donne les 3 premières lignes du fichier.

- *$ tail –2 fich*

ligne 14
ligne 15
donne les 2 dernières lignes du fichier.

Visualisation du contenu des fichiers : exemples

- *$ tail -2 fich | head –1* # donne l'avant dernière ligne.

ligne 14

- *$wc fich*

350 126 30 fich

- *$ file exemple.c*

exemple.c: commands text

- $ ls | wc –l *# compte le nombre de fichiers du répertoire.*

5

- *$ tail +3 fich # affiche de la troisième ligne à la fin du fichier.*

Commandes avancées sur les fichiers

- Extraction d'éléments dans un fichier
  - « cut » extrait des colonnes d'un fichier

    $ cut -f*liste* [-ddélimiteur] [-s] [fichier]

    $ cut -c*liste* [fichier]

    * L'unité associée à l'option –f est choisie quand les lignes sont composées de zones de longueur variable, séparées les unes des autres par un tabulateur

    * L'unité associée à l'option –c est utilisée si les lignes du fichier sont composées de zones de longueur fixe

Commandes avancées sur les fichiers : exemples

$ cat ville

10000,Troyes,Aube

51000,Reims,Marne

$ cut -f1,3 -d ’,’ ville

10000,Aube

51000,Marne

$ cut -c1-5 ville

10000

51000

Liens physiques et symboliques

- La commande « ln » permet de créer un lien pointant vers un fichier (à la manière des raccourcis de Windows).
- Le lien apparaît lorsque l ’on passe la commande « ls -l ».
- Sans l ’option « -s », « ln » crée un lien physique (hard link) au lieu d ’un lien symbolique.
- Les liens symboliques permettent de ne pas dupliquer l ’espace disque utilisé par les fichiers.
- Pour supprimer un lien, on utilise la commande « rm ». C ’est bien le lien et non le fichier pointé qui est détruit.
    - $ ln -s **fichier_ou_répertoire** *lien*

Liens physiques et symboliques : exemple

- $ ln -s fich fich2

  $ ls - l

-rw - r- -r- - 1 retraint users 1829 Mar 5 14:25 fich

lrwxrwxrwx 1 retraint users 4 Mar 9 14:03 fich2 ->fich

  La dernière ligne indique le lien symbolique entre les fichiers fich et fich2

- $ rm fich
- cat fich2

  Cat : fich2 : No such file or directory

Visualisation du contenu des fichiers : exemples

- *$ head –3 fich*

ligne 1
ligne 2
ligne 3
donne les 3 premières lignes du fichier.

- *$ tail –2 fich*

ligne 14
ligne 15
donne les 2 dernières lignes du fichier.

Visualisation du contenu des fichiers : exemples

- *$ tail -2 fich | head –1* # donne l'avant dernière ligne.

ligne 14

- *$wc fich*

350 126 30 fich

- *$ file exemple.c*

exemple.c: commands text

- $ ls | wc –l *# compte le nombre de fichiers du répertoire.*

5

- *$ tail +3 fich # affiche de la troisième ligne à la fin du fichier.*

Commandes avancées sur les fichiers

- Extraction d'éléments dans un fichier
  - « cut » extrait des colonnes d'un fichier

    $ cut -f*liste* [-ddélimiteur] [-s] [fichier]

    $ cut -c*liste* [fichier]

    * L'unité associée à l'option –f est choisie quand les lignes sont composées de zones de longueur variable, séparées les unes des autres par un tabulateur

    * L'unité associée à l'option –c est utilisée si les lignes du fichier sont composées de zones de longueur fixe

Commandes avancées sur les fichiers : exemples

$ cat ville

10000,Troyes,Aube

51000,Reims,Marne

$ cut -f1,3 -d ’,’ ville

10000,Aube

51000,Marne

$ cut -c1-5 ville

10000

51000

Commandes avancées sur les fichiers

- Trier des fichiers
  - « sort » permet de trier un fichier.
  - Par défaut, le tri est alphabétique dans le respect de l'ordre ascii. Les numériques apparaissent avant les majuscules, qui apparaissent avant les minuscules.

- Découper un fichier
  - « split » découpe un fichier en segments de n lignes (1000 par défauts) en créant des fichiers fic_outaa, fic_outab ..
  - split [-n] [fic_in] [fic_out]

Commandes avancées sur les fichiers : exemples

- *$ ls –l | sort –n +6 – 8*

drwxrwxr-x 2 retraint users 16384 Jan 4 10:00 bin
drwxrwxr-x 29 retraint users 8192 Jan 4 10:02 khoros
drwxr-xr-x 2 retraint users 8192 Mar 9 14:35 rep1

tri selon les colonnes 6 à 8, mode numérique (jour,heure)

- *$ split –10 exemple.c fich*

$ ls -l

-rw-r--r-- 1 retraint users 232 Mar 9 16:25 fichaa
-rw-r--r-- 1 retraint users 373 Mar 9 16:25 fichab
-rw-r--r-- 1 retraint users 305 Mar 9 16:25 fichac

Commandes avancées sur les fichiers

- Comparaison de fichiers ou répertoires
  - « diff » compare ligne par ligne le contenu des fichiers textes et écrit les différences sur la sortie standard.

- Repérage des mots clés
  - $ grep [options] ”motif ” {fichiers}
  - « grep » écrit sur la sortie standard, les lignes des fichiers contenant une chaîne de caractères satisfaisant à l’expression « motif ».

Commandes avancées sur les fichiers

- Recherche des fichiers
  - « find » permet de rechercher des fichiers en fonction des critères indiqués.
  - précéder de la condition « \! » , on recherche les fichiers ne satisfaisant pas la condition donnée.
  - exemples :
    - $ find /
    - $ find / -name ‘fic*’
    - $ find / \! -name ‘fic*’
    - $ find / -mtime 2 -name ‘ *.bak ’ -exec rm {} \;

Connaissance de l'espace disque disponible/utilisé

- Taille des fichiers
  - « du » affiche l'espace disque disponible utilisé pour toute l 'arborescence courante ou pour celle spécifiée.
  - du [options] {répertoires(s)}

- Espace disponible
  - « df » affiche l 'espace disque libre et utilisé par tous les utilisateurs sur les « file systems » montés qu'ils soient locaux ou distants (disques réseau).
  - df [options] {file_system ou répertoire}

Les processus

- On appelle processus une instance d ’un programme en cours d ’exécution. Un processus est unique et est identifié par un numéro « PID », que le noyau du système attribue au lancement.
- La mémoire utilisée par un processus lui est propre et n ’est pas visible par les autres processus. La mémoire est représentée de manière logique.
- Les variables d ’environnement qu’un processus utilise lui sont propres (l’identité de l’utilisateur, le répertoire courant, …)

Commandes de gestion des processus

- Liste de processus
  - « ps » affiche les informations concernant les processus, notamment le « PID », son état, le temps « CPU » utilisé, la commande exécutée.
    - Ps [options] [PID]
- Tuer un processus
  - il peut arriver qu'un processus soit trop long ou ne réponde pas. « kill » permet d 'intervenir sur ce processus.
  - malgré son nom, la commande « kill » ne tue précisément pas le processus, mais elle lui envoie un signal.

Commandes de gestion des processus : exemples

$ ps

PID TTY TIME CMD

18101 ttyp9 0:00.01 sleep 4000

$ kill –9 18101

le processus “ *sleep 4000* ” est tué. (-9 ~ -s KILL)

$ sleep 200

[1] 12623

$ kill –s STOP 12623

le processus n’est pas tué, ni terminé, il est tout simplement stoppé.

Les jobs : manipulation

Un job est un travail crée et contrôlé par un Shell. Il peut être composé d'un ou plusieurs processus, par exemple reliés par un tube.

- Lancement en arrière plan
  - « cmd » : en avant plan (foreground)
  - « cmd & » : en arrière plan (background), en tâche de fond
- Liste des jobs
  - « jobs » liste les jobs du shell courant (suspendus ou s'exécutant en arrière plan) avec leurs « jobID »

Les jobs : manipulation

- « CTRL-C » arrête le job courant en avant plan.
- « kill %jobID » tue le job de numéro « jobID ».
- « CTRL-Z » suspend un job lancer en avant plan. Pour continuer le job, il suffit de lui envoyer le signal « CONT »
- On peut faire passer un job en arrière plan ou en avant plan :
  - La commande « bg » le fait passer en arrière plan
  - La commande « fg jojID » fait passer le job « jobId » au premier plan.

Les jobs : exemples

$ (ls –l >fic.tmp ; sleep 60) &
[1] 22325
$ ps

PID	TTY	TIME	CMD
7840	ttyp5	0:07.73	ls -l
22325	ttyp5	0:00.00	-csh (csh)

$ ps

PID	TTY	TIME	CMD
21903	ttyp5	0:00.01	sleep 60
22325	ttyp5	0:00.00	-csh (csh)

Les jobs : exemples

$ sleep 200 &
[1] 20547
$ jobs
[1] + Running sleep 200
$ kill -s STOP %1
[1] + Stopped (signal) sleep 200 le job 1 est stoppé.
$ bg %1 relance le job en arrière plan
[1] sleep 200 &
$ *fg %1* lance le job au premier plan
sleep 200