

R1.04 – Cours 2

Gestion de fichiers, Processus

Département Informatique

IUT2, UGA

2022/2023

Plan du cours

- 1 Système de fichiers
- 2 Métacaractères (jokers, *wildcards*)
- 3 Commandes pour manipuler fichiers et répertoires
- 4 Entrées cachées
- 5 Processus
- 6 Résumé

Plan du cours

- 1 Système de fichiers
- 2 Métacaractères (jokers, *wildcards*)
- 3 Commandes pour manipuler fichiers et répertoires
- 4 Entrées cachées
- 5 Processus
- 6 Résumé

Contenu d'un système de fichiers (SF)

Dans tous les systèmes :

- **Fichiers** : suite d'octets représentant un texte (selon un certain encodage), un programme exécutable, des données binaires, ...
- **Répertoires** ou **dossiers** (*directory*) : contiennent des fichiers et/ou des répertoires

Dans les systèmes Unix et Linux :

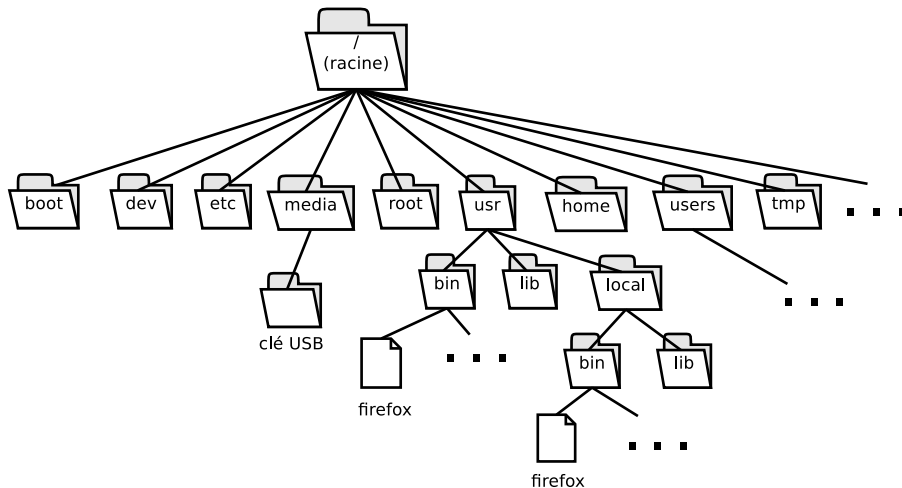
- **Liens symboliques** : vus plus tard
- Fichiers «spéciaux» représentant les périphériques du système
- Autres : tubes nommés, sockets

Organisation d'un système de fichiers

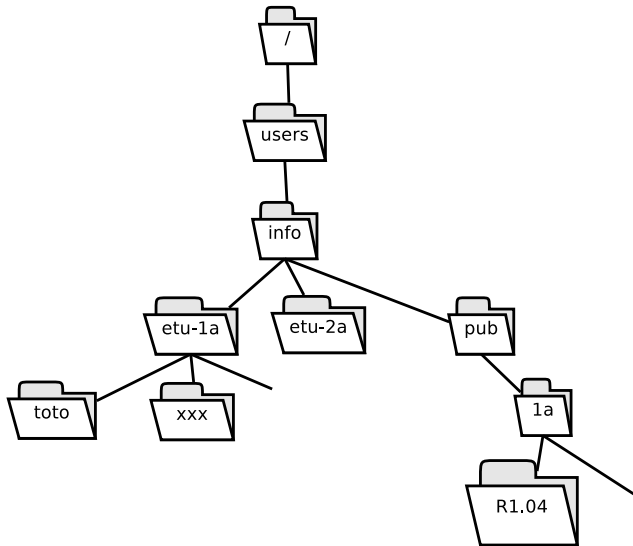
- Organisation hiérarchique → **arbre**
 - Fichiers et répertoires accessibles indépendamment des supports physiques de stockage (disques durs, serveurs de fichiers)
- Pas de «lecteurs» ni de «lecteurs réseau»
- La racine de l'arbre est un dossier appelé "/" (*slash*)

Arborescence Linux standard

Standard : FHS (*Filesystem Hierarchy Standard*)



Arborescence spécifique à l'IUT2



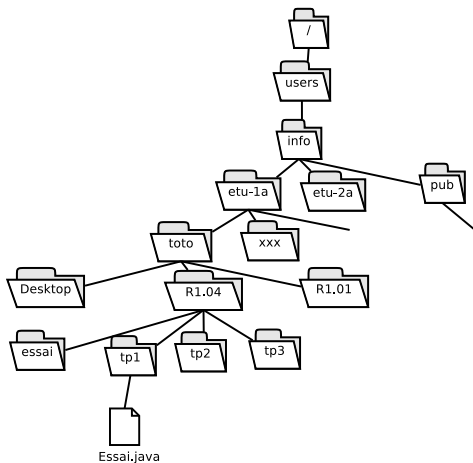
Vocabulaire, notations

- **Entrée** : tout fichier ou répertoire
- **Répertoire racine** (*root directory*)
 - c'est le répertoire qui contient tout le SF
 - notation : / (*slash*)
- **Répertoire père** d'une entrée
 - c'est le répertoire auquel cette entrée appartient
 - notation : . . (point point)
- **Répertoire personnel** d'un utilisateur (*home directory*, *homedir*)
 - c'est le répertoire qui lui est réservé pour stocker ses fichiers
 - il porte en général le nom de cet utilisateur (*toto*)
 - notation : ~ (tilde)
- **Répertoire courant** d'un *shell* (*current/working directory*)
 - c'est le répertoire de travail de ce *shell*
 - au lancement d'un *shell*, c'est initialement le répertoire personnel
 - notation : . (point)

Désigner une entrée (chemin d'accès)

- Pour identifier une entrée dans une commande du *shell*, il faut la désigner par son nom et par un chemin qui permet d'y accéder au sein du SF
- Pour construire ce chemin, on énumère la liste des répertoires qu'il faut traverser au sein du SF pour atteindre l'entrée.
- Il y a deux points de départ possibles pour ce chemin :
 - le répertoire racine (/) : on parle alors de **chemin absolu**
 - le répertoire courant (.) : on parle alors de **chemin relatif**
- Les noms des différents répertoires qui composent un chemin sont énumérés, séparés par le caractère /

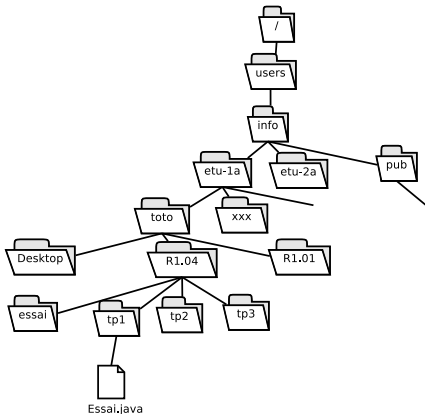
Chemin absolu



`/users/info/etu-1a/toto/R1.04/tp1/Essai.java`

`~/R1.04/tp1/Essai.java`

Chemin relatif simple

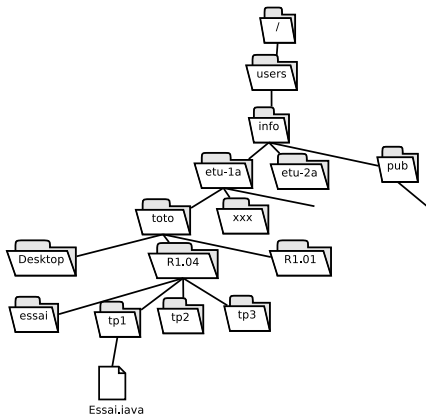


Répertoire courant : `/users/info/etu-s1/toto/`

`./R1.04/tp1/Essai.java`

`R1.04/tp1/Essai.java`

Chemin relatif avec remontée dans l'arbre



Répertoire courant : `/users/info/etu-s1/toto/R1.04/tp2/`

`../../tp1/Essai.java`

`../tp1/Essai.java`

Plan du cours

- 1 Système de fichiers
- 2 Métacaractères (jokers, *wildcards*)
- 3 Commandes pour manipuler fichiers et répertoires
- 4 Entrées cachées
- 5 Processus
- 6 Résumé

Utilité et principe

- Permettent de manipuler plusieurs entrées en une seule commande
- Principe : désigner plusieurs entrées sans les nommer explicitement
- Utilisent les caractères spéciaux * ? []
- Le caractère * permet de remplacer une suite quelconque (y compris vide) de caractères
- Le caractère ? permet de remplacer un caractère et un seul, n'importe lequel
- Entre crochets [] on peut énumérer une liste de caractères possibles

Premiers exemples

- Commandes multiples sans jokers

```
cp -r tp1 /media/toto/MaCle/  
cp -r tp2 /media/toto/MaCle/  
...
```

- Commande unique sans jokers

```
cp -r tp1 tp2 tp3 tp4-afinir tp5-afinir /media/...
```

- Commandes avec jokers

```
cp -r tp* /media/toto/MaCle/  
cp -r tp? /media/toto/MaCle/  
cp -r tp[12] /media/toto/MaCle/
```

Exemples courants

Un joker peut être utilisé au début, à la fin, au milieu d'un chemin ou tout seul

- Copie tous les fichiers dont le nom se termine par `.java`
`cp *.java /media/...`
- Copie tous les répertoires de mon homedir qui commencent par `R` (répertoires des ressources IUT2)
`cp -r R* /media/...`
- Copie tous les TP à finir
`cp -r tp*-afinir /media/...`
- Copie toutes les entrées du répertoire courant
`cp -r * /media/...`

Exemples plus complexes

- On peut combiner jokers et autres notations

```
cp -r ~/R* /media/...
```

- On peut utiliser plusieurs jokers

```
rm *~
```

```
rm */*~
```

```
rm */*/*~
```

... (on verra comment généraliser)



- On peut utiliser plusieurs jokers différents

```
ls -l R*/tp?/*.*txt
```

- Le tilde n'a pas la même signification selon l'endroit où il est utilisé

```
rm ~/*~
```

Mise en garde

- Copie tous les fichiers qui terminent par .txt
`cp *.txt essai`
- La même commande avec espace donne un résultat très différent !
`cp * .txt essai`
- Commande pour supprimer les fichiers de *backup*
`rm *~`
- La même commande avec espace
donne un **résultat catastrophique**
`rm * ~`  Ne pas taper cette commande!!! 
- Version prudente de la commande
`rm -i *~`

Mécanisme

- Le *shell* lui-même effectue la substitution expression avec jokers → liste d'entrées
- Le logiciel lancé par une commande avec jokers ne voit que la liste d'entrées
- Exemple
 - je tape la commande

```
mv *.jpeg ~/Pictures
```
 - le logiciel `mv` est lancé ainsi

```
mv toto.jpeg chaton.jpeg ~/Pictures
```
- Conséquence intéressante
 - on peut utiliser des jokers quand on lance un logiciel qu'on a écrit soi même (programme en Java qui ouvre plusieurs fichiers, script en *shell*)
 - ce logiciel n'a pas besoin de gérer les jokers lui-même

Plan du cours

- 1 Système de fichiers
- 2 Métacaractères (jokers, *wildcards*)
- 3 Commandes pour manipuler fichiers et répertoires
- 4 Entrées cachées
- 5 Processus
- 6 Résumé

Manipuler des répertoires

- `pwd` *(print working directory)*
Affiche le chemin absolu du répertoire courant
- `cd [RÉPERTOIRE]` *(change directory)*
Change le répertoire courant. Sans paramètre, c'est le répertoire personnel qui devient le répertoire courant (`cd` \equiv `cd ~`)
- `ls [RÉPERTOIRE]...` *(list)*
Affiche le contenu du répertoire. Sans paramètre, c'est le contenu du répertoire courant qui est affiché (`ls` \equiv `ls .`)
- `mkdir RÉPERTOIRE...` *(make directory)*
Crée un répertoire
- `rmdir RÉPERTOIRE...` *(remove directory)*
Supprime un répertoire s'il est vide
- `rm -r RÉPERTOIRE...` *(remove)*
Supprime un répertoire et tout ce qu'il contient
- `cp -r RÉP-SOURCE RÉP-DESTINATION` *(copy)*
Copie le répertoire source et tout ce qu'il contient vers la destination ou le nouveau nom indiqué (selon que RÉP-DESTINATION existe ou pas)
- `mv RÉP-SOURCE RÉP-DESTINATION` *(move)*
Renomme/déplace répertoire source en/vers répertoire destination (selon que RÉP-DESTINATION existe ou pas)

Visualiser ou éditer des fichiers texte

- `cat [FICHIER] ...`
Affiche le contenu du(des) fichier(s) en une fois.
- `more [FICHIER] ...`
Affiche, page par page, le contenu du(des) fichier(s).
On ne peut pas revenir en arrière.
- `less [FICHIER] ...`
Affiche, page par page, le contenu du(des) fichier(s).
On peut faire défiler dans les 2 sens avec les flèches haut et bas.
- `gedit [FICHIER] ... &`
Edite/crée le(s) fichier(s) avec l'éditeur de texte gedit

Copier des fichiers et des répertoires

- `cp [-r] ENTRÉE-SOURCE ENTRÉE-DESTINATION`

Copie un fichier ou un répertoire source unique.

La copie s'appellera ENTREE-DESTINATION

ou sera contenue dans ENTRÉE-DESTINATION

```
cp toto.txt titi.txt      # création de titi.txt (ou écrasement)
cp -r tp1 tp2            # création de tp2 s'il n'existait pas avant
cp toto.txt /tmp         # création de /tmp/toto.txt (ou écrasement)
cp toto.txt /tmp/monfichier.txt
                           # création de /tmp/monfichier.txt (ou écrasement)
```

- `cp [-r] ENTRÉE-SOURCE... REP-DESTINATION`

Copie plusieurs fichiers ou répertoires sources

vers un répertoire destination préexistant

```
cp -r * /tmp/toto        # copie de tous les fichiers et répertoires dans /tmp/toto.
                           # /tmp/toto doit exister au préalable.
```

Renommer ou déplacer des fichiers et des répertoires

- **mv ENTRÉE-SOURCE ENTRÉE-DESTINATION**

Renomme ou déplace un fichier ou un répertoire source unique.

La source sera renommée en ENTRÉE-DESTINATION

ou sera contenue dans ENTRÉE-DESTINATION

```
mv toto.txt titi.txt           # renomme toto.txt en titi.txt
mv tp1 tp2                     # renomme tp1 en tp2 s'il n'existait pas avant
mv toto.txt /tmp               # déplace toto.txt vers /tmp/
mv toto.txt /tmp/monfichier.txt # déplace et renomme toto.txt
```

- **mv ENTRÉE-SOURCE... REP-DESTINATION**

Déplace plusieurs fichiers ou répertoires sources

vers un répertoire destination préexistant

```
mv * /tmp/toto                # déplace tous les fichiers et répertoires dans /tmp/toto.
                              # /tmp/toto doit exister sinon erreur.
```


Supprimer des fichiers et des répertoires

- `rm FICHER...`
Supprime le(s) fichier(s)
- `rmdir RÉPERTOIRE...`
Supprime le(s) répertoires(s)
Ils doivent être vides.
- `rm -r ENTRÉE...`
Supprime le(s) fichier(s) et répertoire(s)
Les répertoires sont supprimés avec tout leur contenu !

Affichage et messages d'erreur

- Les commandes `pwd`, `ls`, `cat`, `more`, `less`, ...
affichent leur résultat dans le terminal
- Les commandes `mkdir`, `cd`, `cp`, `mv`, ...
n'affichent rien en cas de succès
- En cas d'échec, elles affichent un message d'erreur qu'il faut savoir analyser et interpréter
 - No such file or directory
 - Permission denied
 - ...

Plan du cours

- 1 Système de fichiers
- 2 Métacaractères (jokers, *wildcards*)
- 3 Commandes pour manipuler fichiers et répertoires
- 4 Entrées cachées**
- 5 Processus
- 6 Résumé

Définition, Utilité

- Entrée qui n'est pas montrée par défaut à l'utilisateur
- Utile pour
 - fichiers qu'on ne manipule pas souvent
 - fichiers dont la manipulation est «dangereuse»
 - fichiers/dossiers que les utilisateurs novices ne doivent pas manipuler
- Par convention les entrées cachées sont celles qui commencent par un point (.)

Exemples

- Fichiers cachés dans votre homedir

```
.bashrc  
.profile  
.bash_history  
...
```

- Dossiers cachés dans votre homedir

```
.config  
.kde  
.local  
.mozilla  
...
```

Voir les entrées cachées

- Dans le *shell*
 - option `-a` du logiciel `ls`
 - exemple : `ls -al`
- Dans les logiciels graphiques
 - voir TP

Plan du cours

- 1 Système de fichiers
- 2 Métacaractères (jokers, *wildcards*)
- 3 Commandes pour manipuler fichiers et répertoires
- 4 Entrées cachées
- 5 Processus**
- 6 Résumé

OS multi-tâches, multi-utilisateurs

- Plusieurs utilisateurs peuvent utiliser le système simultanément
- Un utilisateur peut lancer plusieurs logiciels en même temps
- L'exécution d'un logiciel donne naissance à une tâche ou **processus**
- Un processus est donc un **programme en train de s'exécuter**
- C'est une entité dont le noyau Linux contrôle l'état, de la vie à la mort
- Système à **temps partagé** : le processeur est partagé entre plusieurs processus concurrents (simultanés)
- Système **multi-tâches préemptif** : l'ordonnanceur (*scheduler*) peut interrompre de force un processus pour redonner le contrôle du processeur au noyau Linux ou à un autre processus

Gestion des processus

- Les processus sont créés par duplication (clonage) d'un processus existant
- Les processus sont donc organisés en arbre : chaque processus a un et un seul processus père (celui à partir duquel il a été cloné)
- On peut visualiser la liste des processus avec les commandes :
 - `ps`
 - `pstree`
 - `top`
 - `ksysguard`, `gnome-system-monitor`
- On peut mettre fin à un processus avec les commandes :
 - `kill` (par numéro de processus)
 - `killall` (par nom de processus)
 - `ksysguard`, `gnome-system-monitor`

Arborescence de processus

```
systemd+-agetty
|-apache2---11*[apache2]
|-cron
|-cupsd
|-dhclient
|-fail2ban-server---2*[{fail2ban-server}]
|-inetd
|-lightdm+-Xorg---{Xorg}
|   |-lightdm+-lightdm-gtk-gre---{gmain}
|-rpc.gssd
|-rpc.idmapd
|-rpc.statd
|-rpcbind
|-rsyslogd
|-rwhod---rwhod
|-sshd+-sshd---sshd
|   |-sshd---bash---pstree
|   `--sshd---sshd---sftp-server
|-x2goagent
```

Attributs d'un processus (extrait)

- PID : identifiant unique numérique
- PPID : numéro de son père
- UID : utilisateur au nom duquel le processus s'exécute
- GID : groupe auquel le processus est rattaché
- Priorité (en fait courtoisie)
 - influe sur la fréquence à laquelle l'ordonnanceur donnera le processeur au processus
 - varie de -20 (le plus prioritaire) à 19 (le moins prioritaire)
 - modifiable par les commandes `nice` et `renice`
 - 0 par défaut
 - un utilisateur de base ne peut que diminuer la priorité de ses processus
- ...

```
toto@pc-dg-xxx-xx:~$ ps -l
F S  UID  PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 R  1226 25260 25259  0  75   0 -   993 -          pts/1    00:00:00 bash
0 R  1226 25380 25260  0  75   0 -   582 -          pts/1    00:00:00 ps
```

Ressources d'un processus

- Environnement hérité du processus père (un *shell* par exemple)
 - répertoire courant
 - ligne de commande qui a lancé le processus
 - variables d'environnement
- Mémoire (RAM) séparée de celle des autres processus (mémoire virtuelle)
- Liste des fichiers ouverts, dont au moins 3 par défaut
 - entrée standard
 - sortie standard
 - erreur standard
 - on utilisera ces entrées/sorties standard plus tard...

Plan du cours

- 1 Système de fichiers
- 2 Métacaractères (jokers, *wildcards*)
- 3 Commandes pour manipuler fichiers et répertoires
- 4 Entrées cachées
- 5 Processus
- 6 **Résumé**

Résumé

- Je connais les **commandes de base** par cœur
- Je sais **retrouver les autres commandes** rapidement
- Je connais l'existence des **fichiers cachés**, mais j'évite d'y toucher à moins de bien savoir ce que je fais
- Je sais visualiser et gérer les **processus** qui tournent sur une machine