INSTITUT Mines-Télécom

TELECOM SudParis

CI2 – Système de Fichiers

CSC3102 – Introduction aux systèmes d'exploitation

Elisabeth Brunet

23/09/2015

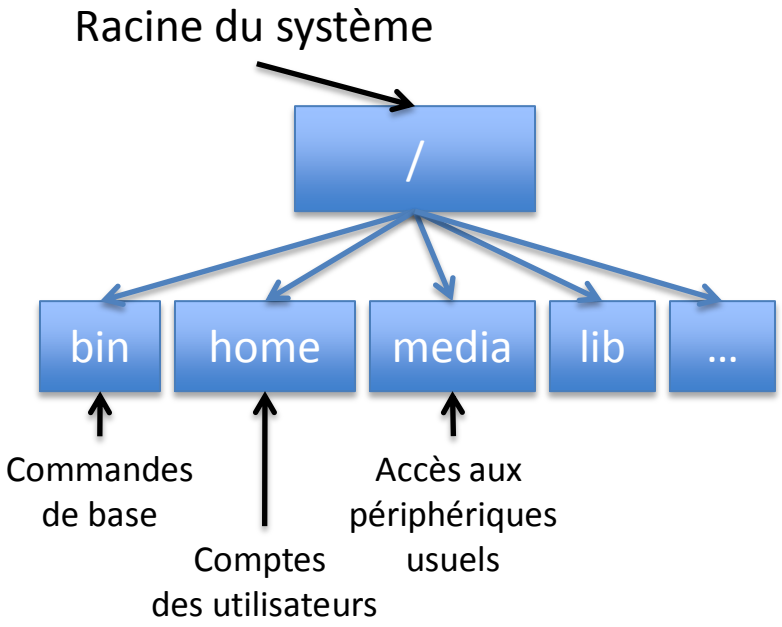
Système de Fichiers (FileSystem, FS)

- Besoin de mémoriser des informations
  - Photos, PDF, données brutes, exécutables d'application, le source de l'OS lui-même, etc.
- Organisation du stockage sur mémoire de masse
  - Localisation abstraite grâce à un chemin dans une arborescence
  - Unité de base = Fichier

- Exemples de systèmes de fichiers
  - NTFS pour Windows, Ext2-3-4 pour Linux, HFSX pour Mac-OS
  - FAT pour les clés USB, ISO pour les CD
  - A plus grande échelle : Base de données

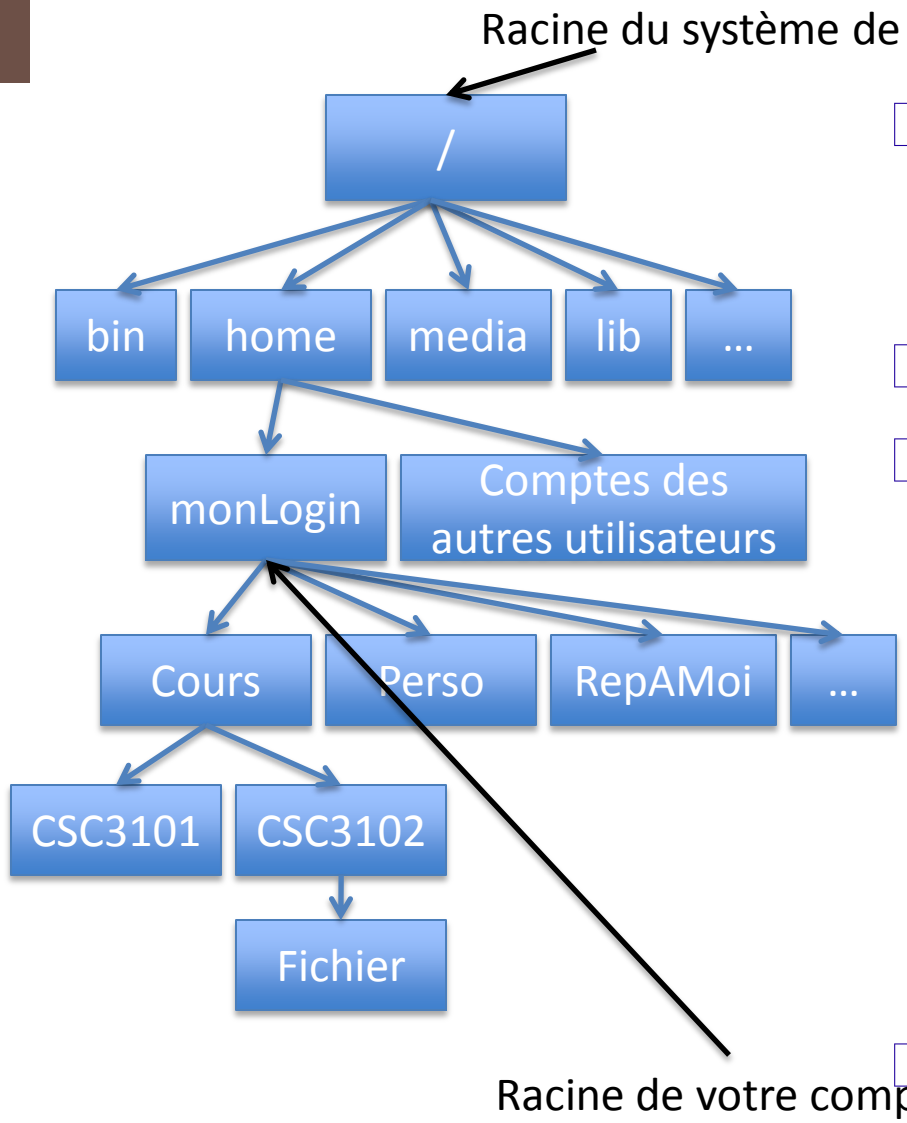
SYSTÈME DE FICHIERS EXT UTILISÉ DANS LINUX

Point de vue utilisateur : Arborescence du système de fichiers



- Point d'entrée = « / » = racine
- Toujours la même base
  - `/bin`, `/media`, `/home`, etc.
- Différents types d'entrées
  - Répertoire : Dossier
  - Fichier : Données
  - Lien : Passerelle vers une autre entrée du système de fichiers
  - Fichiers spéciaux : Accès unifié à un périphérique

Arborescence de votre compte



- `monLogin` = votre compte = votre répertoire de connexion
- En général, dans `/home`
- A TSP, dans un répertoire particulier
  - Partagé par NFS (Network File System) pour une accessibilité sur les serveurs TSP et tous les postes des salles de TP
  - `/mci/ei1417/monLogin`
- Organisation personnelle

Chemin absolu – Chemin relatif

- Positionnement dans le système de fichiers
  - Répertoire courant (ou répertoire de travail)
  - A la connexion, racine de votre compte

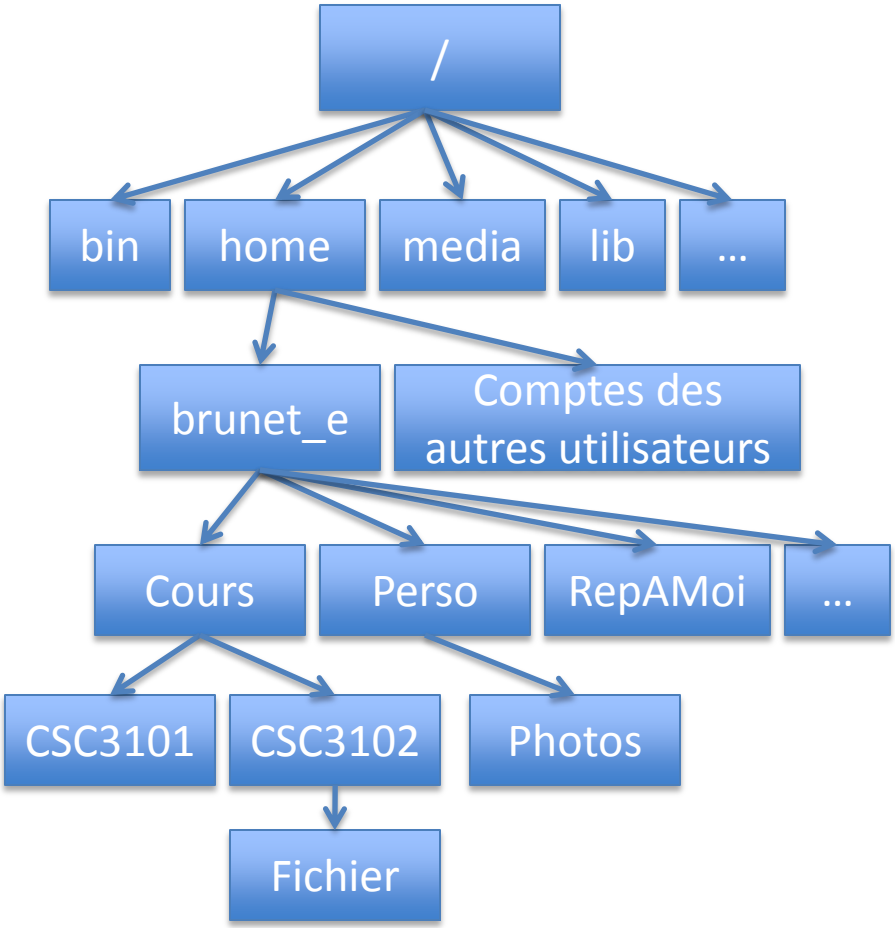
- Chemin absolu = chemin depuis la racine du système de fichiers
  - Correct quelque que soit le répertoire courant
  - Chemin raccourci vers la racine de votre compte : ~
- Chemin relatif = chemin par rapport au répertoire courant
  - Le chemin relatif « . » correspond au répertoire courant
  - Le chemin relatif « .. » correspond au répertoire parent du répertoire courant

Positionnement dans l'arborescence

- Consultation
  - `pwd` : affiche le chemin absolu du répertoire courant
  - `ls [opt][chemin]` : liste les entrées d'un répertoire
    - Sans argument : celles du répertoire courant
    - Celles du répertoire dont le chemin est donné en argument
    - Options supplémentaires à consulter dans le man

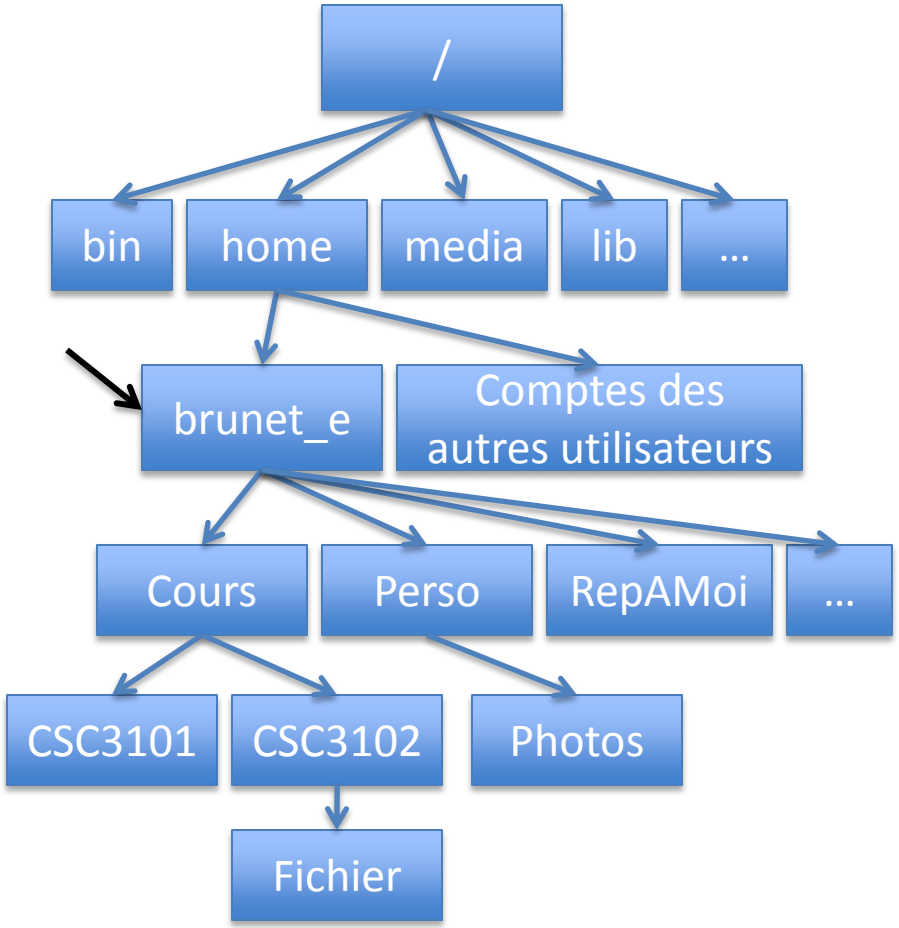
- Navigation
  - `cd [chemin]` : permet de changer de répertoire de travail
    - `cd` = *change directory*
    - Sans argument : retour au répertoire de connexion
    - Déplacement suivant le chemin (absolu ou relatif) donné en argument

Illustration en tant que brunet_e



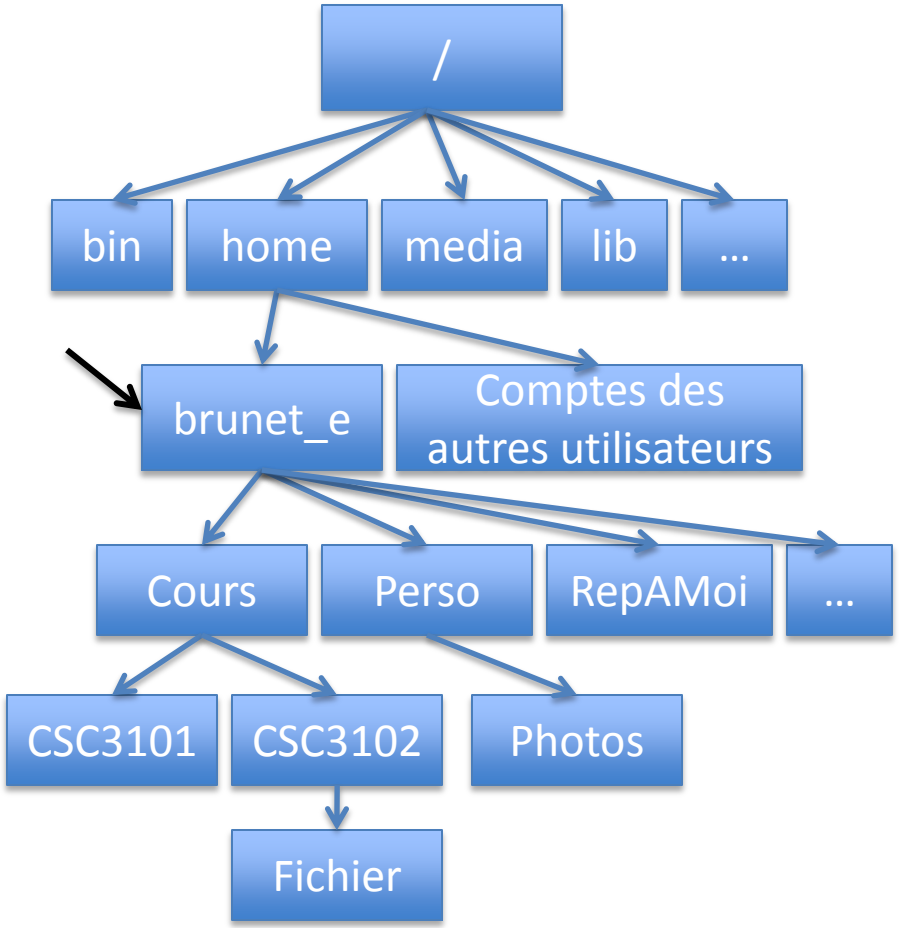
$

Illustration en tant que brunet_e



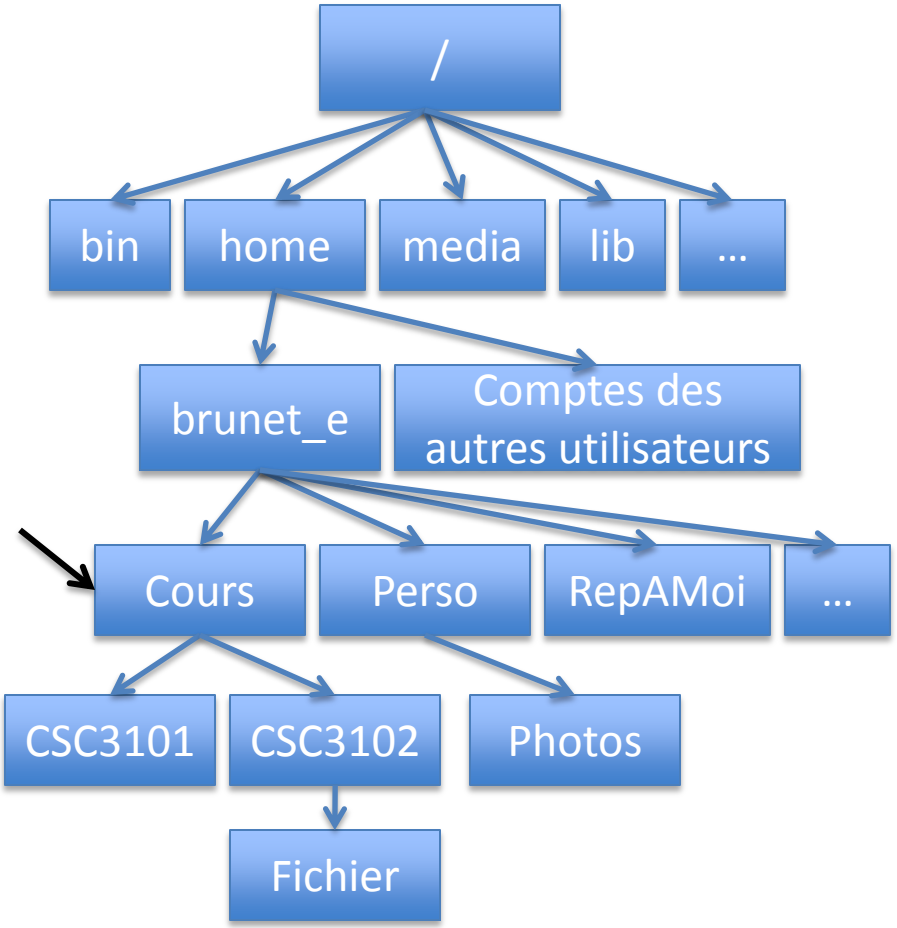

```
$ pwd
/home/brunet_e
$
```

Illustration en tant que brunet_e



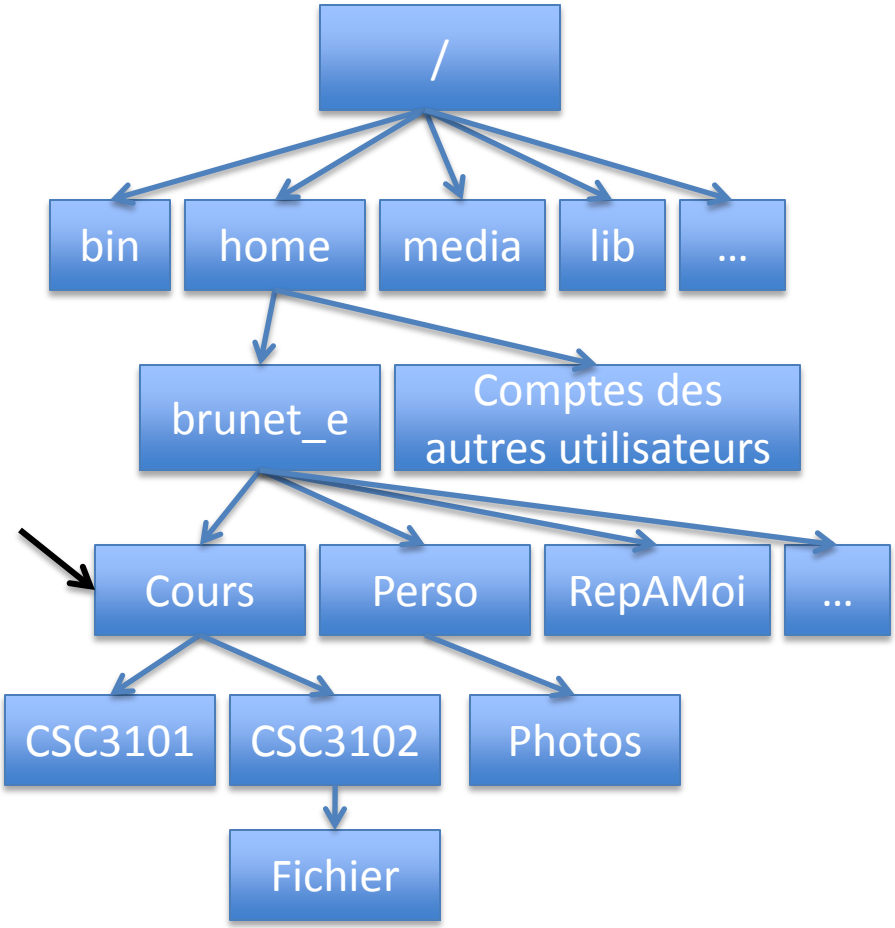

```
$ pwd
/home/brunet_e
$ ls
Cours Perso RepAMoi
$
```

Illustration en tant que brunet_e



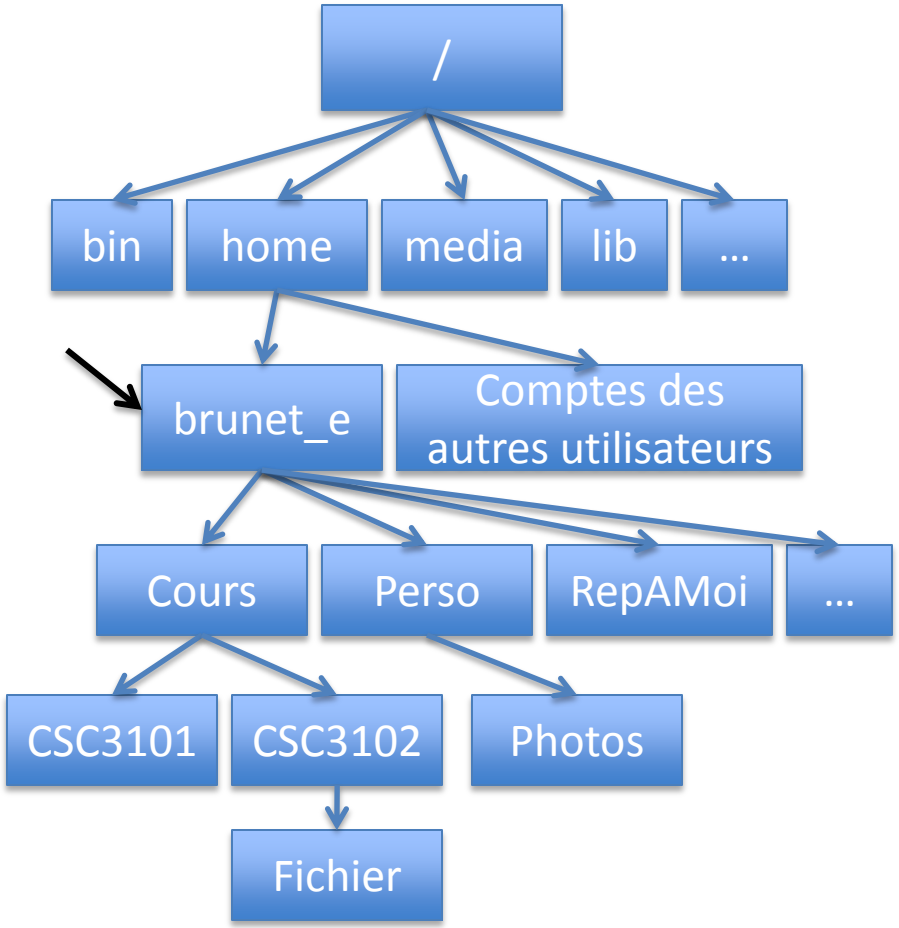

```
$ pwd
/home/brunet_e
$ ls
Cours Perso RepAMoi
$ cd Cours
$
```

Illustration en tant que brunet_e



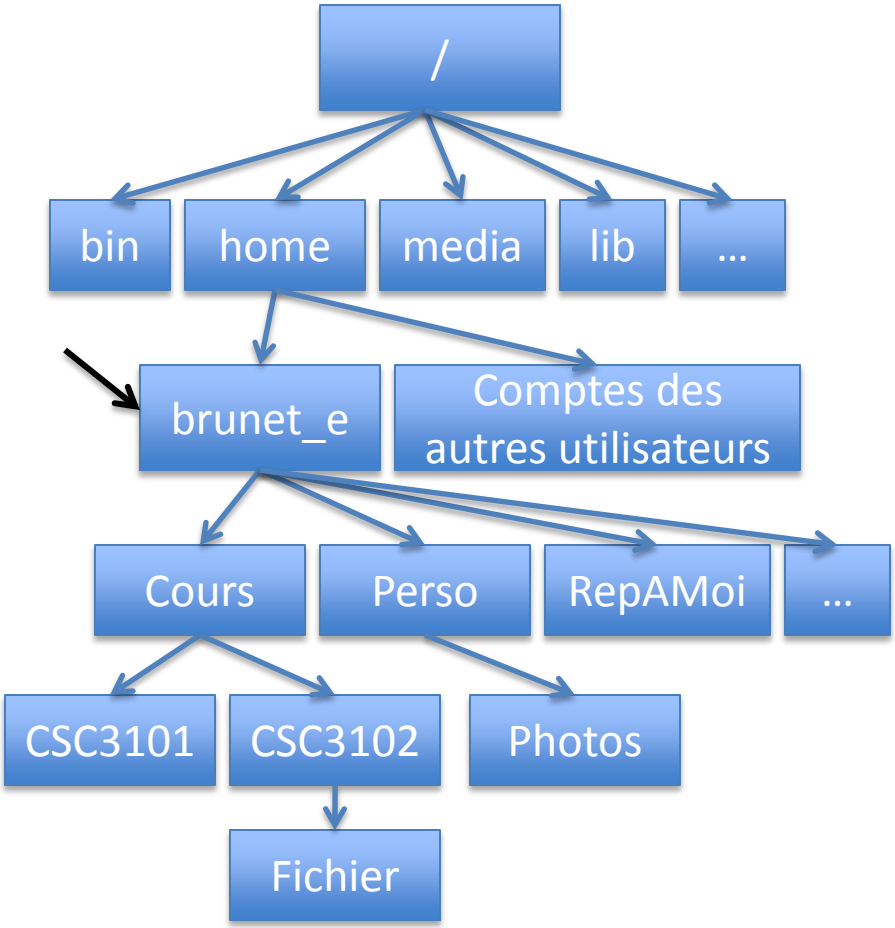

```
$ pwd
/home/brunet_e
$ ls
Cours Perso RepAMoi
$ cd Cours
$ ls
CSC3101 CSC3102
$
```

Illustration en tant que brunet_e



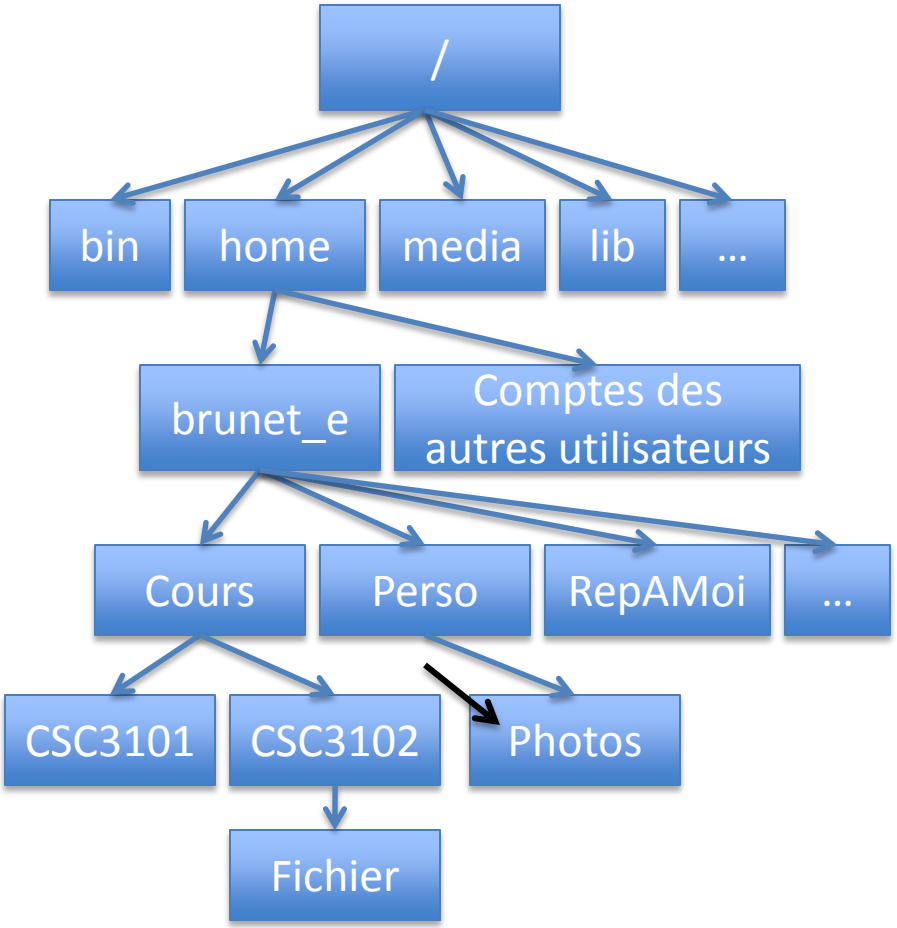

```
$ pwd
/home/brunet_e
$ ls
Cours Perso RepAMoi
$ cd Cours
$ ls
CSC3101 CSC3102
$ cd ..
$
```

Illustration en tant que brunet_e



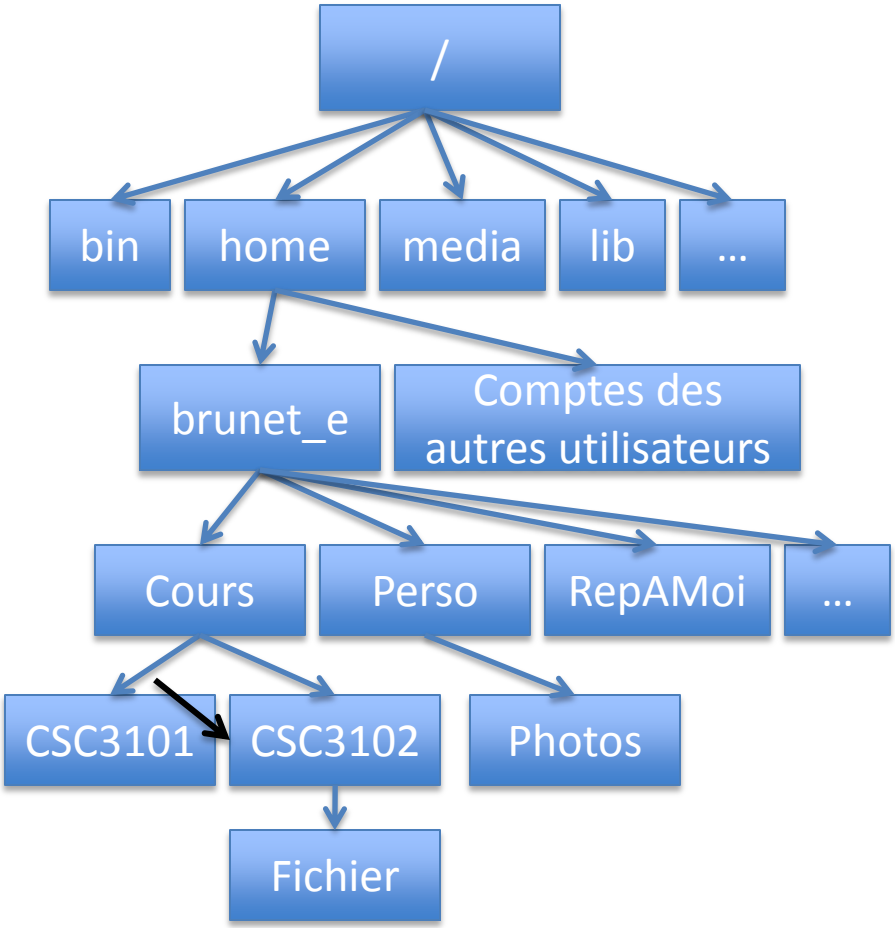

```
$ pwd
/home/brunet_e
$ ls
Cours Perso RepAMoi
$ cd Cours
$ ls
CSC3101 CSC3102
$ cd ..
$ pwd
/home/brunet_e/
$
```

Illustration en tant que brunet_e



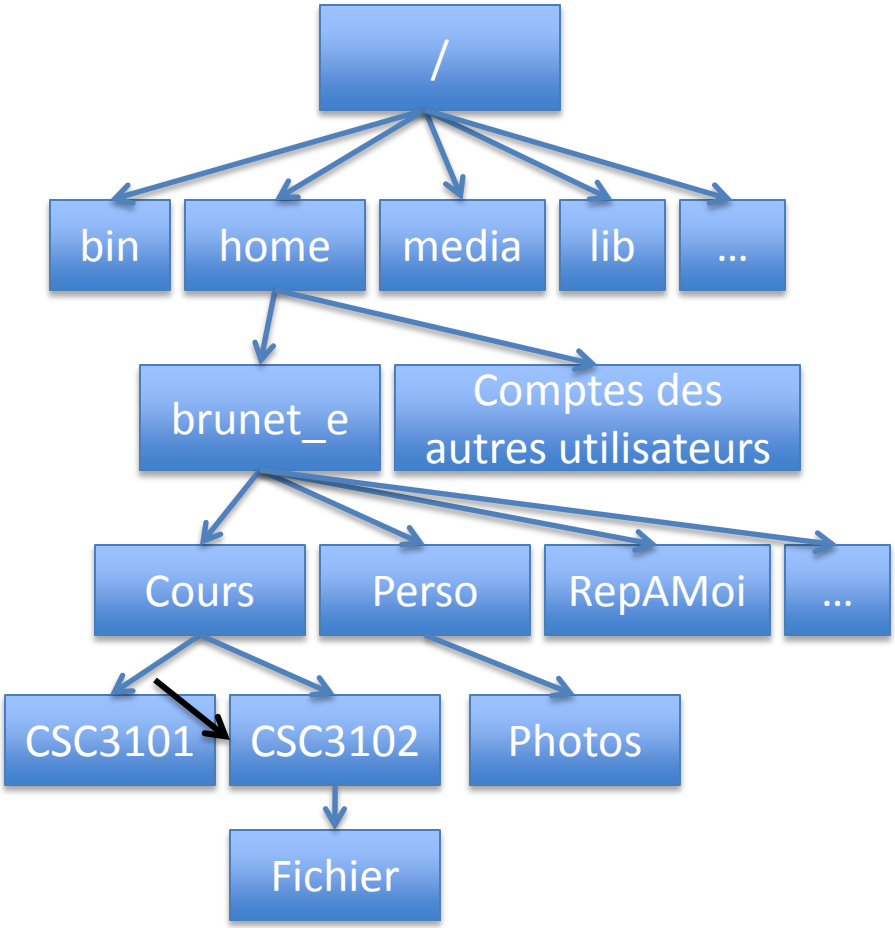

```
$ pwd
/home/brunet_e
$ ls
Cours Perso RepAMoi
$ cd Cours
$ ls
CSC3101 CSC3102
$ cd ..
$ pwd
/home/brunet_e/
$ cd /home/brunet_e/Perso/Photos
$
```

Illustration en tant que brunet_e



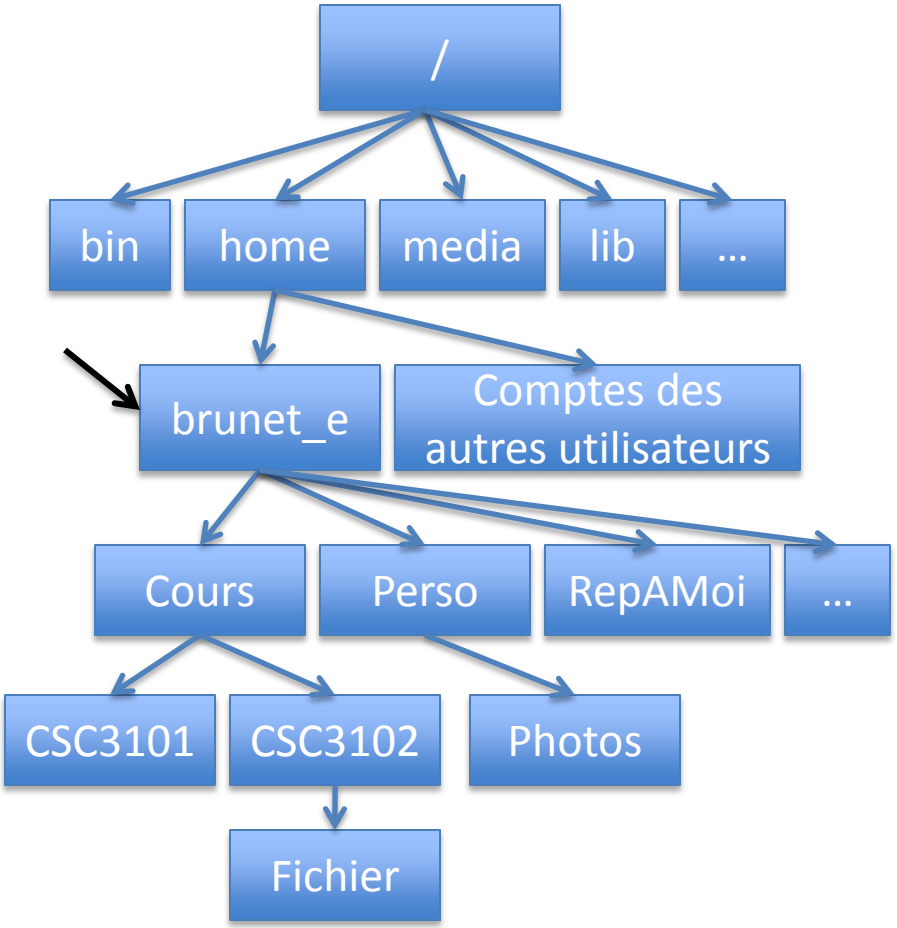

```
$ pwd
/home/brunet_e
$ ls
Cours Perso RepAMoi
$ cd Cours
$ ls
CSC3101 CSC3102
$ cd ..
$ pwd
/home/brunet_e/
$ cd /home/brunet_e/Perso/Photos
$ cd ../../Cours/CSC3102
$
```

Illustration en tant que brunet_e



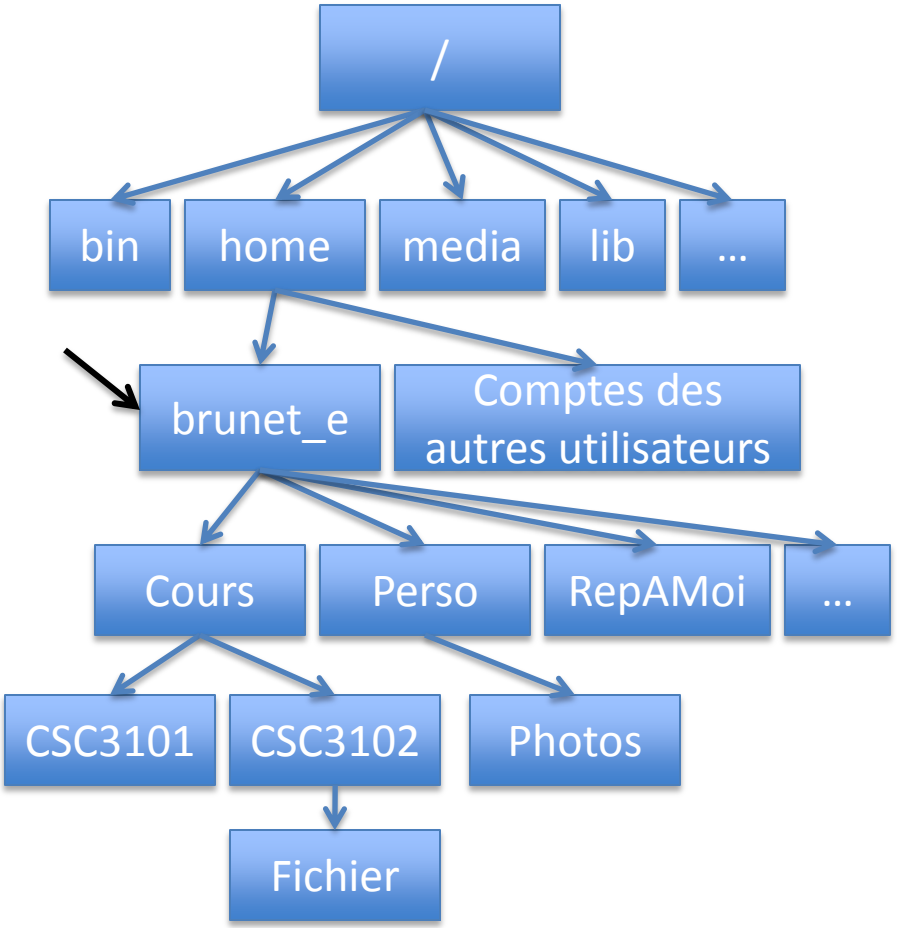

```
$ pwd
/home/brunet_e
$ ls
Cours Perso RepAMoi
$ cd Cours
$ ls
CSC3101 CSC3102
$ cd ..
$ pwd
/home/brunet_e/
$ cd /home/brunet_e/Perso/Photos
$ cd ../../Cours/CSC3102
$ ls
Fichier
$
```

Illustration en tant que brunet_e




```
$ pwd
/home/brunet_e
$ ls
Cours Perso RepAMoi
$ cd Cours
$ ls
CSC3101 CSC3102
$ cd ..
$ pwd
/home/brunet_e/
$ cd /home/brunet_e/Perso/Photos
$ cd ../../Cours/CSC3102
$ ls
Fichier
$ cd
$
```

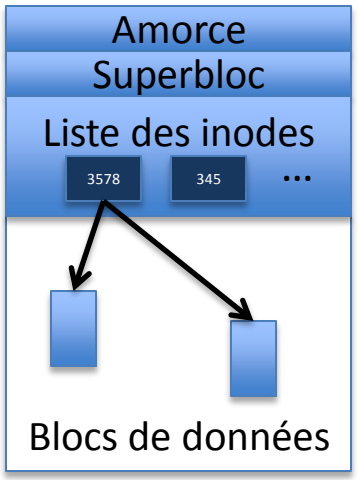
Illustration en tant que brunet_e




```
$ pwd
/home/brunet_e
$ ls
Cours Perso RepAMoi
$ cd Cours
$ ls
CSC3101 CSC3102
$ cd ..
$ pwd
/home/brunet_e/
$ cd /home/brunet_e/Perso/Photos
$ cd ../../Cours/CSC3102
$ ls
Fichier
$ cd
$ pwd
/home/brunet_e
```

Point de vue infrastructure : Partitions de l'espace mémoire

- Installation du FS sur une partition
- Structuration d'une partition en 4 zones
  - Amorce : programme chargeur (boot)
    - Utilisée uniquement par la partition stockant le code de l'OS
  - Superbloc : descripteur du FS lui-même
  - Liste d'inodes : descripteurs de chaque entrée du FS
  - Blocs de données
    - unité de stockage indivisible de taille fixe
      - Généralement 4Ko



Point de vue infrastructure : Partitions de l'espace mémoire

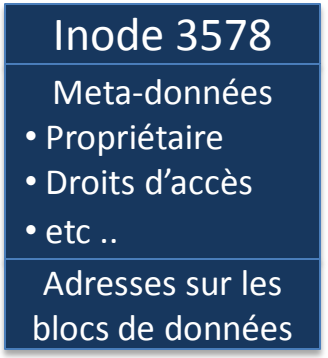
- Division de l'espace disque en plusieurs partitions
  - Initialement pour des contraintes de taille
    - À présent, une taille d'une partition peut aller jusqu'à 32To
  - Cloisonnement pour sauvegarde, réinstallation, installation double-boot, etc.
  - Souvent, `/`, `/home`, `/tmp` sont sur des partitions séparées
- 1 partition *maître bootable* liée aux autres par points de montage
- Transparent pour l'utilisateur ! (à part à l'installation de l'OS)

Concept d'inode

- Chaque entrée du FS est décrite par un inode

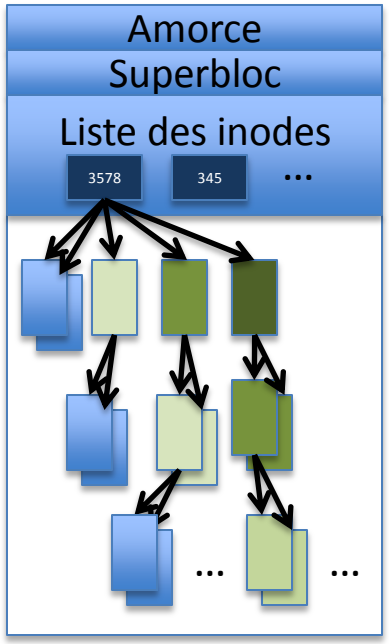
- 128 octets de méta-données
  - Identifiant unique **au sein de sa partition**
    - Consultation : `ls -i`
  - Type (répertoire, fichier…)
  - Propriétaire
  - Groupe propriétaire
  - Droits d'accès
  - Compteur de liens
  - Autres : taille, historique…

  - Adresses sur les blocs stockant les données



Fichiers ordinaires

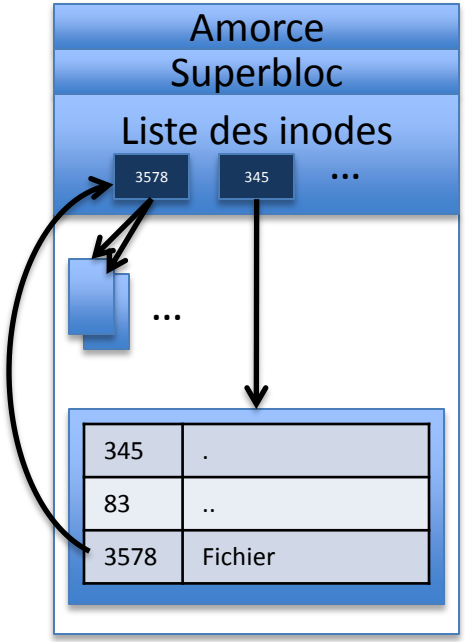
- Octets stockés les uns à la suite des autres dans des blocs de mémoire d'une partition
- Adresses des blocs utilisés pour stocker les données :
  - 10 blocs de données en direct
  - 1 bloc d'adresses à indirection simple
  - 1 bloc d'adresses à indirection double
  - 1 bloc d'adresses à indirection triple



- Allocation des blocs à la demande
- Taille de fichier potentiellement énorme!
- Accès rapide assuré aux fichiers de petite taille
- Toujours transparent du point de vue utilisateur !

Répertoires

- Regroupement d'entrées
- Met en correspondance le nom de ces entrées avec leur numéro d'inode
- Tout répertoire contient au moins des références :
  - sur lui-même « . »
  - sur son répertoire parent « .. »



Liens

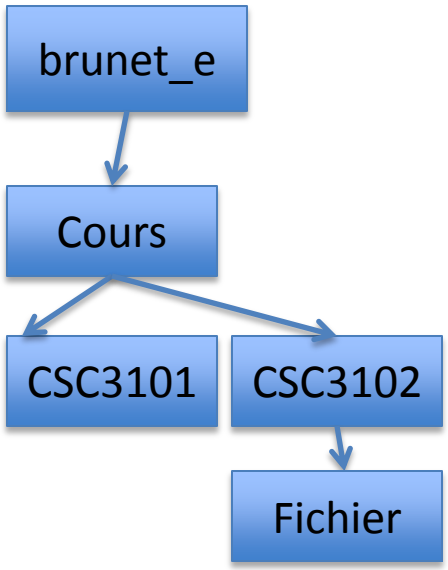
- Accès à une même entrée depuis plusieurs répertoires
- Deux sortes
  - Lien physique au sein d'une même partition
  - Lien symbolique pour aller au-delà d'une même partition

Lien physique

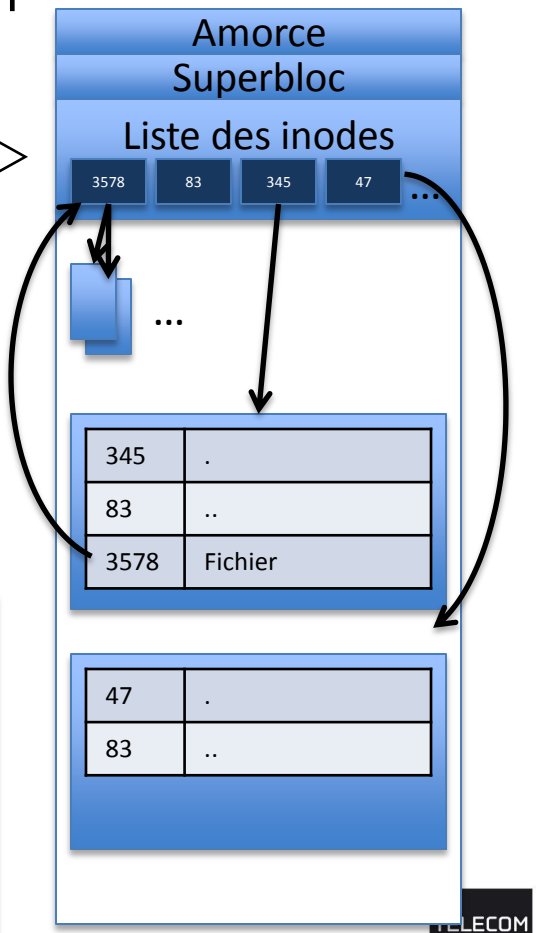
- Accès direct totalement transparent à un **même** inode
  - Cohérent uniquement au sein d'une même partition
- Uniquement sur un fichier existant
- Commande `ln <fichier cible> <lien>`

Lien physique

- Accès direct totalement transparent à un **même** inode
  - Cohérent uniquement au sein d'une même partition
- Uniquement sur un fichier !!
- Commande `ln <fichier cible> <lien>`
- Exemple :

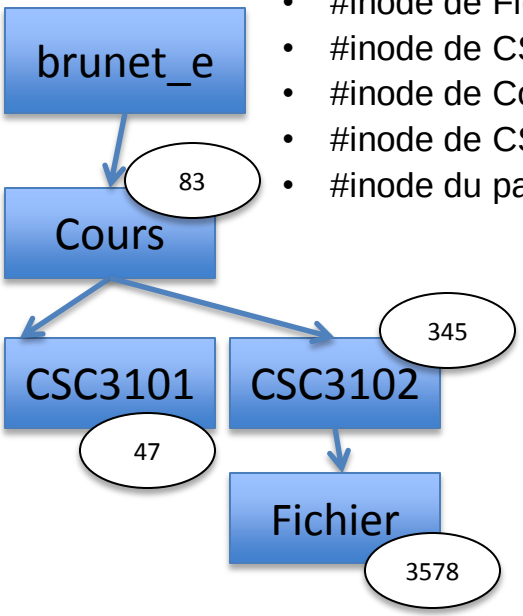



```
$
```



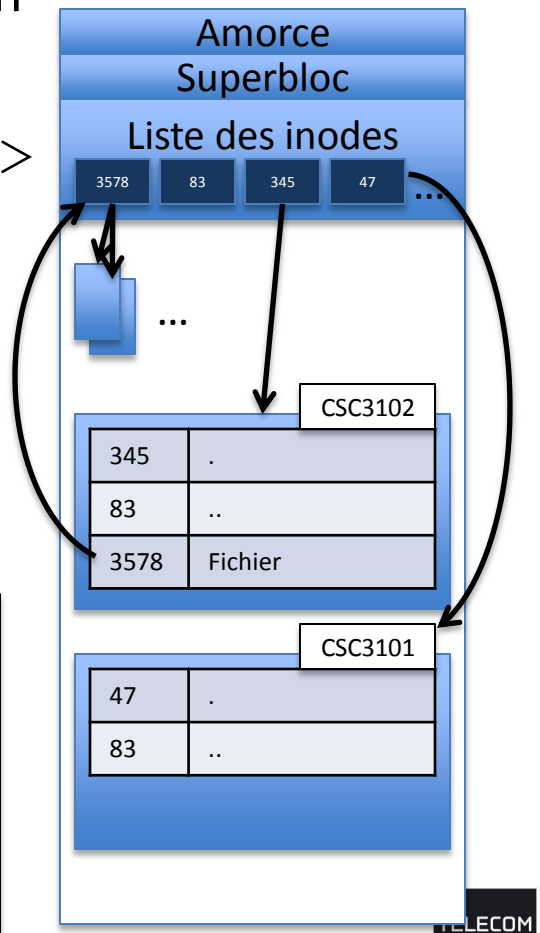
Lien physique

- Accès direct totalement transparent à un **même** inode
  - Cohérent uniquement au sein d'une même partition
- Uniquement sur un fichier !!
- Commande `ln <fichier cible> <lien>`
- Exemple :



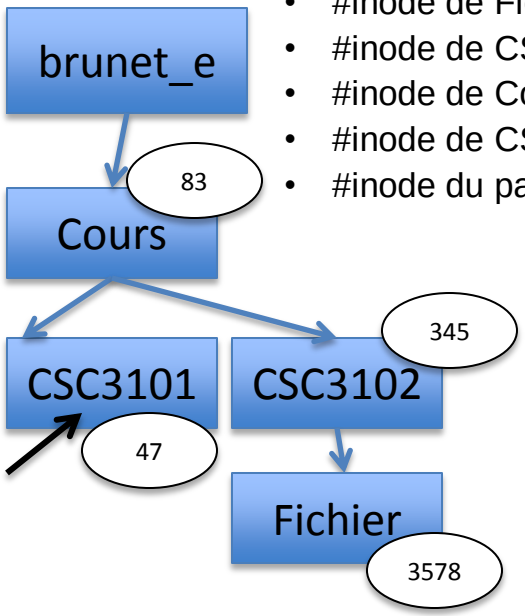
- #inode de Fichier = 3578
- #inode de CSC3102 = 345
- #inode de Cours = #inode du parent de CSC3102 = 83
- #inode de CSC3101 = 47
- #inode du parent de CSC3101 = 83, i.e. le répertoire Cours

```
$
```



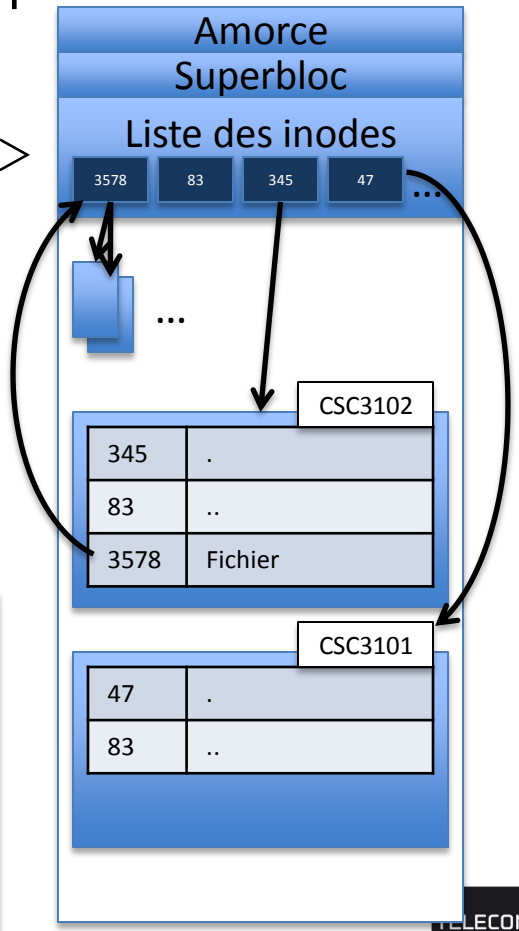
Lien physique

- Accès direct totalement transparent à un **même** inode
  - Cohérent uniquement au sein d'une même partition
- Uniquement sur un fichier !!
- Commande `ln <fichier cible> <lien>`
- Exemple :



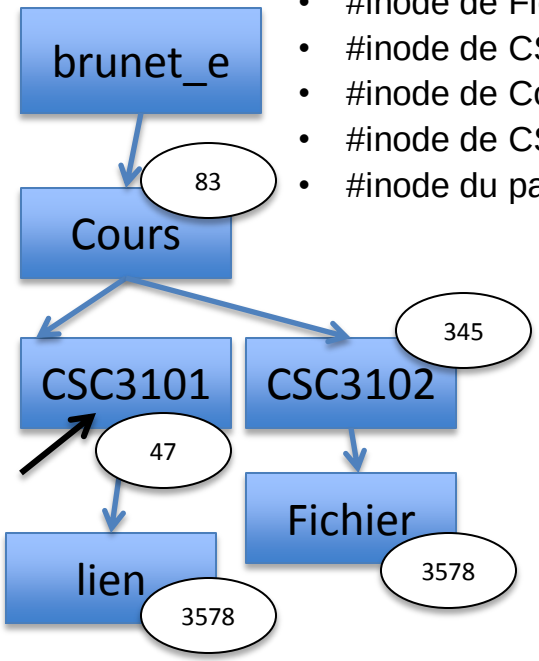
- #inode de Fichier = 3578
- #inode de CSC3102 = 345
- #inode de Cours = #inode du parent de CSC3102 = 83
- #inode de CSC3101 = 47
- #inode du parent de CSC3101 = 83, i.e. le répertoire Cours

```
$ cd ~/Cours/CSC3101
$
```



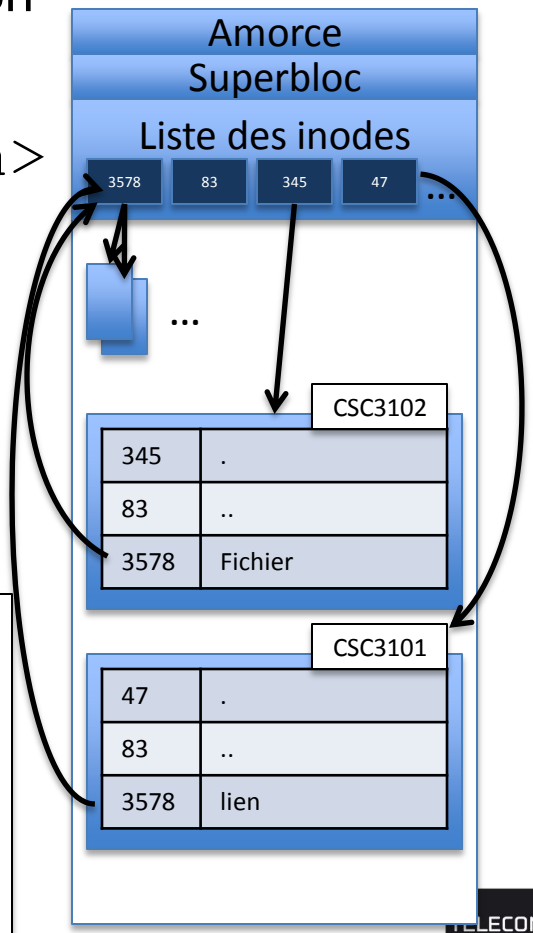
Lien physique

- Accès direct totalement transparent à un **même** inode
  - Cohérent uniquement au sein d'une même partition
- Uniquement sur un fichier !!
- Commande `ln <fichier cible> <lien>`
- Exemple :



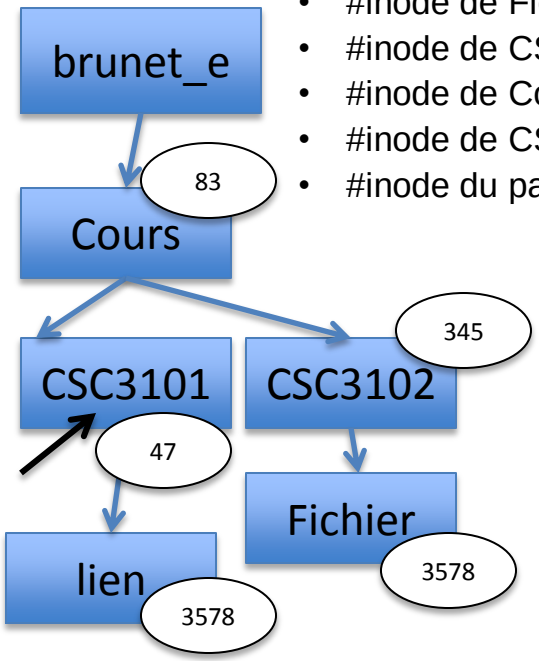
- #inode de Fichier = 3578
- #inode de CSC3102 = 345
- #inode de Cours = #inode du parent de CSC3102 = 83
- #inode de CSC3101 = 47
- #inode du parent de CSC3101 = 83, i.e. le répertoire Cours

```
$ cd ~/Cours/CSC3101
$ ln ../CSC3102/Fichier ./lien
$
```



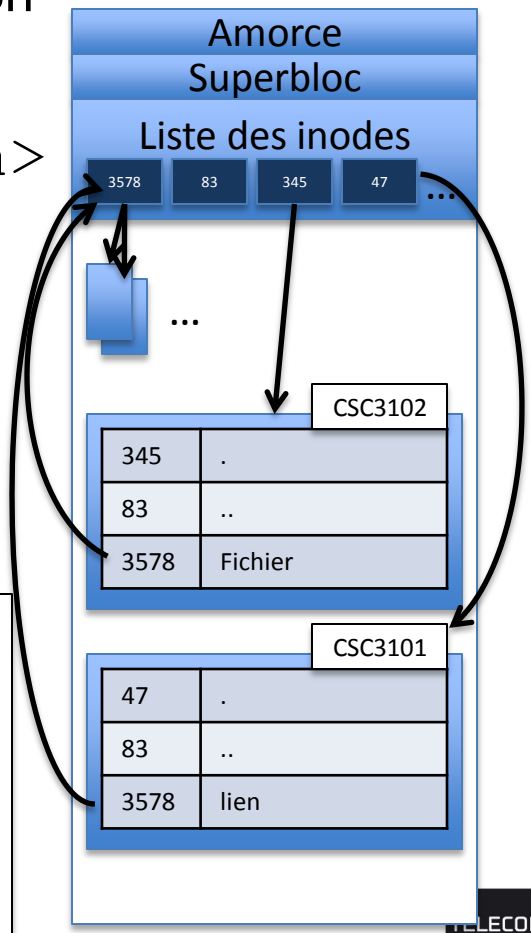
Lien physique

- Accès direct totalement transparent à un **même** inode
  - Cohérent uniquement au sein d'une même partition
- Uniquement sur un fichier !!
- Commande `ln <fichier cible> <lien>`
- Exemple :



- #inode de Fichier = 3578
- #inode de CSC3102 = 345
- #inode de Cours = #inode du parent de CSC3102 = 83
- #inode de CSC3101 = 47
- #inode du parent de CSC3101 = 83, i.e. le répertoire Cours

```
$ cd ~/Cours/CSC3101
$ ln ../CSC3102/Fichier ./lien
$ ls -i ../CSC3102/Fichier lien
3578 ../CSC3102/Fichier
3578 lien
$
```

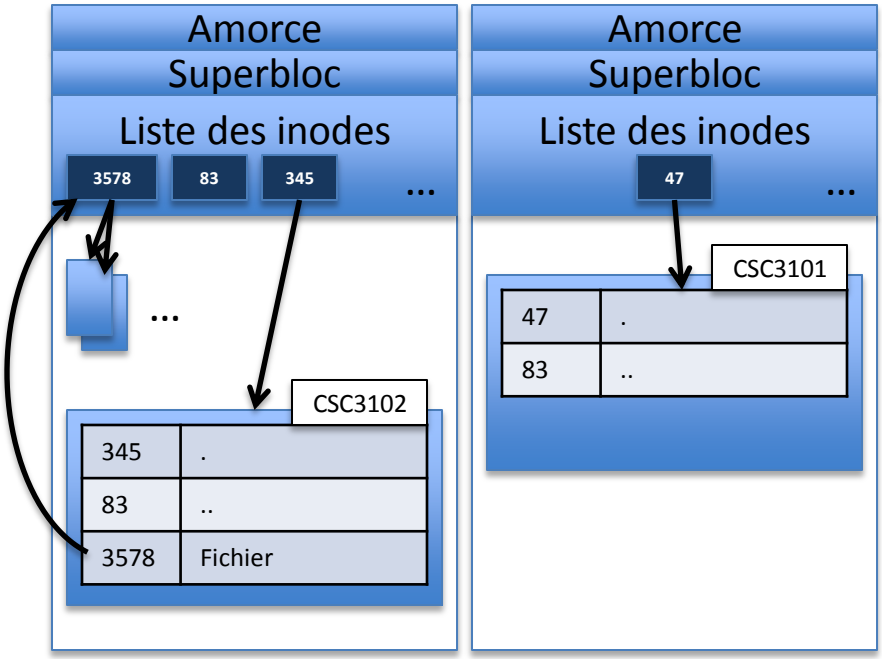


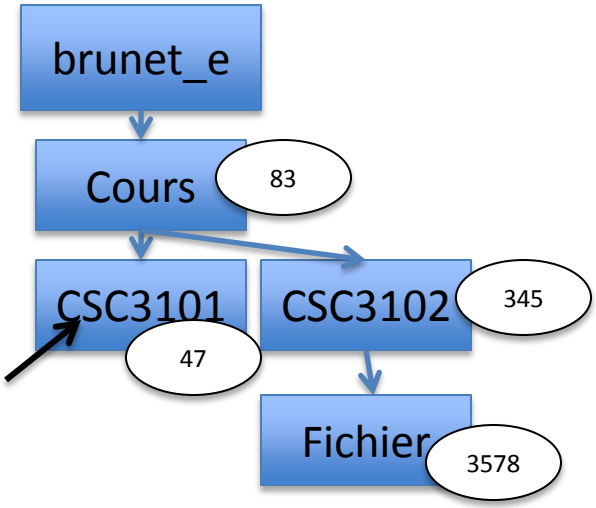
Lien symbolique

- Pour aller au-delà d'une partition
- Création d'un nouvel inode dans la partition du lien
  - Stocke le chemin de l'entrée cible
  - Droits d'accès de l'élément pointé
- Commande `ln -s <cible> <lien>`

Lien symbolique

- Pour aller au-delà d'une partition
- Création d'un nouvel inode dans la partition du lien
  - Stocke le chemin de l'entrée cible
  - Droits d'accès de l'élément pointé
- Commande `ln -s <cible> <lien>`

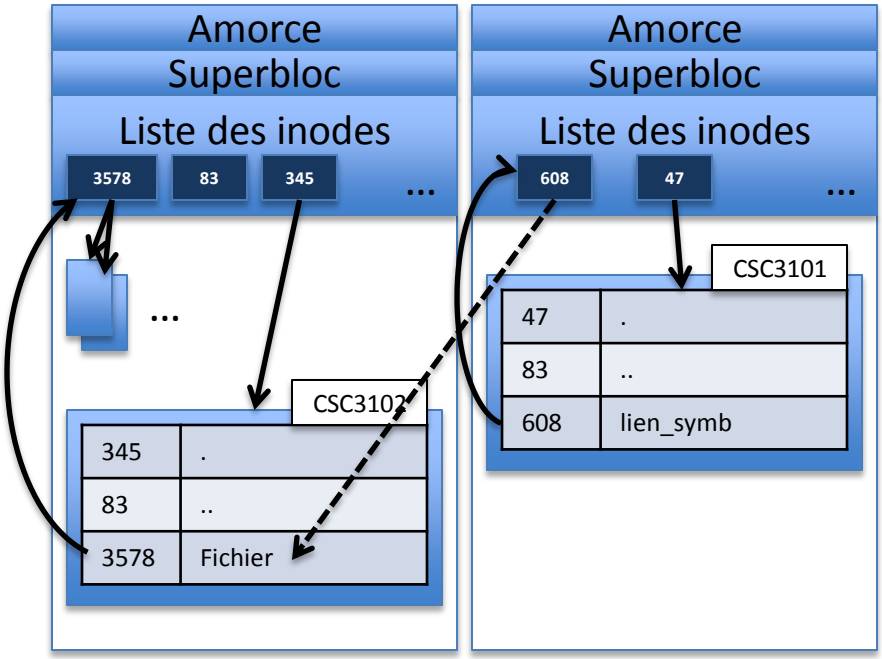


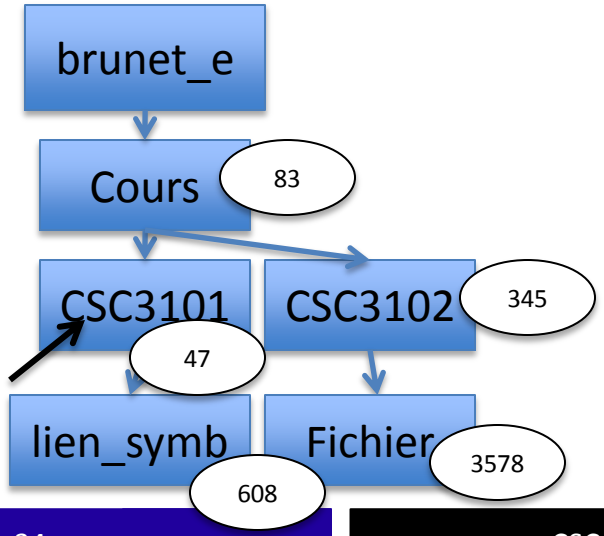



```
$ cd ~/Cours/CSC3101
$
```

Lien symbolique

- Pour aller au-delà d'une partition
- Création d'un nouvel inode dans la partition du lien
  - Stocke le chemin de l'entrée cible
  - Droits d'accès de l'élément pointé
- Commande `ln -s <cible> <lien>`

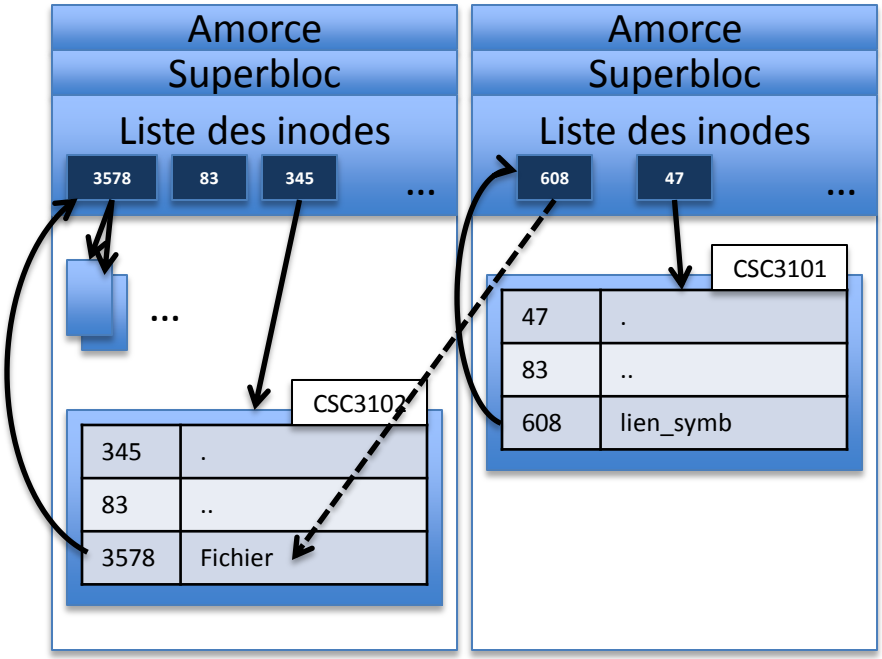


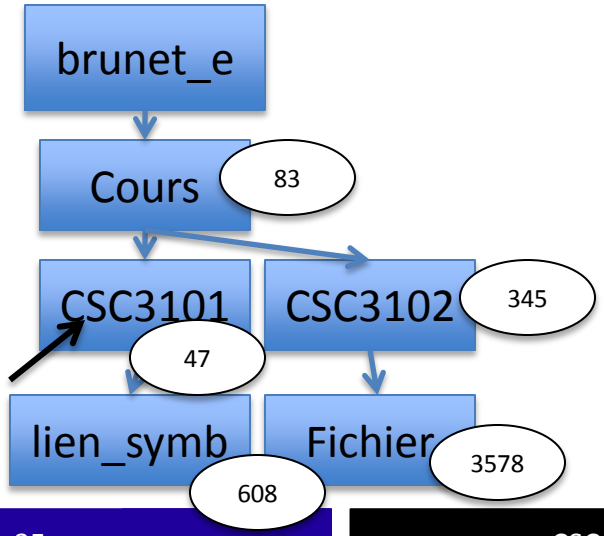



```
$ cd ~/Cours/CSC3101
$ ln -s ../CSC3102/Fichier ./lien_symb
$
```

Lien symbolique

- Pour aller au-delà d'une partition
- Création d'un nouvel inode dans la partition du lien
  - Stocke le chemin de l'entrée cible
  - Droits d'accès de l'élément pointé
- Commande `ln -s <cible> <lien>`






```
$ cd ~/Cours/CSC3101
$ ln -s ../CSC3102/Fichier ./lien_symb
$ ls -i ../CSC3102/Fichier lien_symb
3578 ../CSC3102/Fichier
608 lien_symb
$
```

Commandes utilisateur

- Commandes de base
  - Création
  - Suppression
  - Copie
  - Déplacement / Renommage
  - Consultation
  - Recherche
- Commandes utilitaires bien pratiques
  - Principales vues en CI3

Création d'une entrée dans le FS

- Fichier :
  - Au travers de logiciels
    - en particulier les éditeurs : emacs, gedit, etc.
  - `touch` : création d'un fichier vide
- Répertoire :
  - `mkdir <rep>` : *make directory*
- Lien :
  - Lien physique : `ln <fichier cible> <lien>`
  - Lien symbolique : `ln -s <entrée cible> <lien>`

Suppression

- Dans l'inode du répertoire parent, mise à 0 du n° d'inode de l'entrée supprimée
- Décrémentation du nombre de liens de l'inode
- Si plus aucun lien sur l'inode,
  - libération de l'inode et des blocs de mémoire associés

- Fichier et lien
  - `rm <entrée>` : *remove*
- Répertoire
  - `rmdir <rep>` : suppression d'un répertoire vide
  - `rm -r <rep>` : suppression récursive d'un répertoire

Copie

- Création d'un nouvel inode et duplication des blocs de mémoire associés
- `cp [-r] <src> <dest>` : *copy*
  - `src` correpond au chemin de l'entrée à copier
  - `dest`, au chemin où doit être copiée l'entrée ciblée

Copie

- Création d'un nouvel inode et duplication des blocs de mémoire associés
- `cp [-r] <src> <dest>` : *copy*
  - `src` correpond au chemin de l'entrée à copier
  - `dest`, au chemin où doit être copiée l'entrée ciblée

```
$ ls
fichier repertoire
$
```

Copie

- Création d'un nouvel inode et duplication des blocs de mémoire associés
- `cp [-r] <src> <dest>` : *copy*
  - `src` correpond au chemin de l'entrée à copier
  - `dest`, au chemin où doit être copiée l'entrée ciblée

```
$ ls
fichier repertoire
$ cp fichier fichier2
$
```

Copie

- Création d'un nouvel inode et duplication des blocs de mémoire associés
- `cp [-r] <src> <dest>` : *copy*
  - `src` correpond au chemin de l'entrée à copier
  - `dest`, au chemin où doit être copiée l'entrée ciblée

```
$ ls
fichier repertoire
$ cp fichier fichier2
$ ls
fichier fichier2 repertoire
$
```

Copie

- Création d'un nouvel inode et duplication des blocs de mémoire associés
- `cp [-r] <src> <dest>` : *copy*
  - `src` correpond au chemin de l'entrée à copier
  - `dest`, au chemin où doit être copiée l'entrée ciblée

```
$ ls
fichier repertoire
$ cp fichier fichier2
$ ls
fichier fichier2 repertoire
$ cp fichier repertoire
$
```

Copie

- Création d'un nouvel inode et duplication des blocs de mémoire associés
- `cp [-r] <src> <dest>` : *copy*
  - `src` correspond au chemin de l'entrée à copier
  - `dest`, au chemin où doit être copiée l'entrée ciblée

```
$ ls
fichier repertoire
$ cp fichier fichier2
$ ls
fichier fichier2 repertoire
$ cp fichier repertoire
$ ls
fichier fichier2 repertoire
$
```

Copie

- Création d'un nouvel inode et duplication des blocs de mémoire associés
- `cp [-r] <src> <dest>` : *copy*
  - `src` correpond au chemin de l'entrée à copier
  - `dest`, au chemin où doit être copiée l'entrée ciblée

```
$ ls
fichier repertoire
$ cp fichier fichier2
$ ls
fichier fichier2 repertoire
$ cp fichier repertoire
$ ls
fichier fichier2 repertoire
$ ls repertoire
fichier
```

Déplacement / Renommage

- Renommage
  - Modification du nom de l'entrée dans le répertoire parent
- Déplacement
  - Suppression de l'entrée du répertoire parent
  - Insertion dans le répertoire cible
- Inode et blocs de données inchangés
- `mv <src> <dest>` : *move*

Déplacement / Renommage

- Renommage
  - Modification du nom de l'entrée dans le répertoire parent
- Déplacement
  - Suppression de l'entrée du répertoire parent
  - Insertion dans le répertoire cible
- Inode et blocs de données inchangés
- `mv <src> <dest>` : *move*

```
$ ls
fichier fichier2 repertoire
$
```

Déplacement / Renommage

- Renommage
  - Modification du nom de l'entrée dans le répertoire parent
- Déplacement
  - Suppression de l'entrée du répertoire parent
  - Insertion dans le répertoire cible
- Inode et blocs de données inchangés
- `mv <src> <dest>` : *move*

```
$ ls
fichier fichier2 repertoire
$ mv fichier fichier_src
$
```

Déplacement / Renommage

- Renommage
  - Modification du nom de l'entrée dans le répertoire parent
- Déplacement
  - Suppression de l'entrée du répertoire parent
  - Insertion dans le répertoire cible
- Inode et blocs de données inchangés
- `mv <src> <dest>` : *move*

```
$ ls
fichier fichier2 repertoire
$ mv fichier fichier_src
$ ls
fichier_src fichier2 repertoire
$
```

Déplacement / Renommage

- Renommage
  - Modification du nom de l'entrée dans le répertoire parent
- Déplacement
  - Suppression de l'entrée du répertoire parent
  - Insertion dans le répertoire cible
- Inode et blocs de données inchangés
- `mv <src> <dest>` : *move*

```
$ ls
fichier fichier2 repertoire
$ mv fichier fichier_src
$ ls
fichier_src fichier2 repertoire
$ mv repertoire/fichier .
$
```

Déplacement / Renommage

- Renommage
  - Modification du nom de l'entrée dans le répertoire parent
- Déplacement
  - Suppression de l'entrée du répertoire parent
  - Insertion dans le répertoire cible
- Inode et blocs de données inchangés
- `mv <src> <dest>` : *move*

```
$ ls
fichier fichier2 repertoire
$ mv fichier fichier_src
$ ls
fichier_src fichier2 repertoire
$ mv repertoire/fichier .
$ ls
fichier fichier_src fichier2 repertoire
```

Droits d'accès

- Toute opération sur une entrée du FS est soumise à droits d'accès
  - Message d'erreur « *Permission denied* »

- 3 types d'accès
  - `r` : droit en lecture
    - Si fichier, consultation du contenu du fichier
    - Si répertoire, consultation de ses entrées
  - `w` : droit en écriture
    - Si fichier, droit de modification du contenu du fichier
    - Si répertoire, droit de création, de renommage et de suppression d'une entrée dans le répertoire
  - `x` :
    - si fichier, droit d'exécution
    - si répertoire, droit de traversée
- 3 catégories d'utilisateurs:
  - Propriétaire (`u`) ; groupe propriétaire(`g`) ; tous les autres (`o`)

- Consultation des droits d'accès d'une entrée : `ls -l [entrée]`

Droits d'accès initiaux

- Masque de droits d'accès **!retirés!** à la création de toute entrée dans le FS
  - Commande `umask` (pour user mask)
  - En standard, masque par défaut = `022`
    - [`r = 100` en binaire = 4 en octal, `w = 010 = 2`, `x = 001 = 1`]
    - Droits retirés `--- -w- -w-` → Droits appliqués `rwx r-x r-x`
    - !! Le droit d'exécution est systématiquement retirés aux fichiers
      - Avec 022, fichiers créés avec les droits `rw- r-- r--`
  - Modification du masque grâce à la commande `umask`
    - Action sans effet rétroactif sur les entrées préexistantes

Droits d'accès – modification

- Modification sur une entrée existante ciblée
- `chmod <droit> <entrée>` : *change mode*
- Droits à **!appliquer!** à l'entrée
  - Catégories : `u`, `g`, `o` ou `a`(= `all` = `ugo`)
  - Opérations : Ajout (+), retrait (–), affectation (=)

```
$
```

Droits d'accès – modification

- Modification sur une entrée existante ciblée
- `chmod <droit> <entrée>` : *change mode*
- Droits à **!appliquer!** à l'entrée
  - Catégories : `u`, `g`, `o` ou `a`(= `all` = `ugo`)
  - Opérations : Ajout (+), retrait (–), affectation (=)

```
$ ls -l fichier
-rwx r-- --- fichier
$
```

Droits d'accès – modification

- Modification sur une entrée existante ciblée
- `chmod <droit> <entrée>` : *change mode*
- Droits à **!appliquer!** à l'entrée
  - Catégories : `u`, `g`, `o` ou `a`(= `all` = `ugo`)
  - Opérations : Ajout (+), retrait (–), affectation (=)

```
$ ls -l fichier
-rwx r-- --- fichier
$ chmod u-x fichier
$ ls -l fichier
-rw- r-- --- fichier
$
```

Droits d'accès – modification

- Modification sur une entrée existante ciblée
- `chmod <droit> <entrée>` : *change mode*
- Droits à **!appliquer!** à l'entrée
  - Catégories : `u`, `g`, `o` ou `a`(= `all` = `ugo`)
  - Opérations : Ajout (+), retrait (–), affectation (=)

```
$ ls -l fichier
-rwx r-- --- fichier
$ chmod u-x fichier
$ ls -l fichier
-rw- r-- --- fichier
$ chmod 740 fichier
$ ls -l fichier
-rwx r-- --- fichier
```

Démonstration

$

Démonstration

```
$ cd ; mkdir CSC3102/TP2/Droits
$
```

Démonstration

```
$ cd ; mkdir CSC3102/TP2/Droits
$ cd CSC3102/TP2/Droits
$
```

Démonstration

```
$ cd ; mkdir CSC3102/TP2/Droits
$ cd CSC3102/TP2/Droits
$ cp /etc/passwd .
$
```

Démonstration

```
$ cd ; mkdir CSC3102/TP2/Droits
$ cd CSC3102/TP2/Droits
$ cp /etc/passwd .
$ ls -l
-rw-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$
```

Démonstration

```
$ cd ; mkdir CSC3102/TP2/Droits
$ cd CSC3102/TP2/Droits
$ cp /etc/passwd .
$ ls -l
-rw-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ chmod u-r passwd
--w-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$
```

Démonstration

```
$ cd ; mkdir CSC3102/TP2/Droits
$ cd CSC3102/TP2/Droits
$ cp /etc/passwd .
$ ls -l
-rw-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ chmod u-r passwd
--w-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ cat passwd
cat: passwd: Permission denied
$
```

Démonstration

```
$ cd ; mkdir CSC3102/TP2/Droits
$ cd CSC3102/TP2/Droits
$ cp /etc/passwd .
$ ls -l
-rw-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ chmod u-r passwd
--w-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ cat passwd
cat: passwd: Permission denied
$ mkdir Rep_test
$
```

Démonstration

```
$ cd ; mkdir CSC3102/TP2/Droits
$ cd CSC3102/TP2/Droits
$ cp /etc/passwd .
$ ls -l
-rw-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ chmod u-r passwd
--w-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ cat passwd
cat: passwd: Permission denied
$ mkdir Rep_test
$ ls -l
drwxr-xr-x 1 brunet_e brunet_e 162 juil 29. 21:12 Rep_test
$
```

Démonstration

```
$ cd ; mkdir CSC3102/TP2/Droits
$ cd CSC3102/TP2/Droits
$ cp /etc/passwd .
$ ls -l
-rw-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ chmod u-r passwd
--w-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ cat passwd
cat: passwd: Permission denied
$ mkdir Rep_test
$ ls -l
drwxr-xr-x 1 brunet_e brunet_e 162 juil 29. 21:12 Rep_test
$ cd Rep_test
$
```

Démonstration

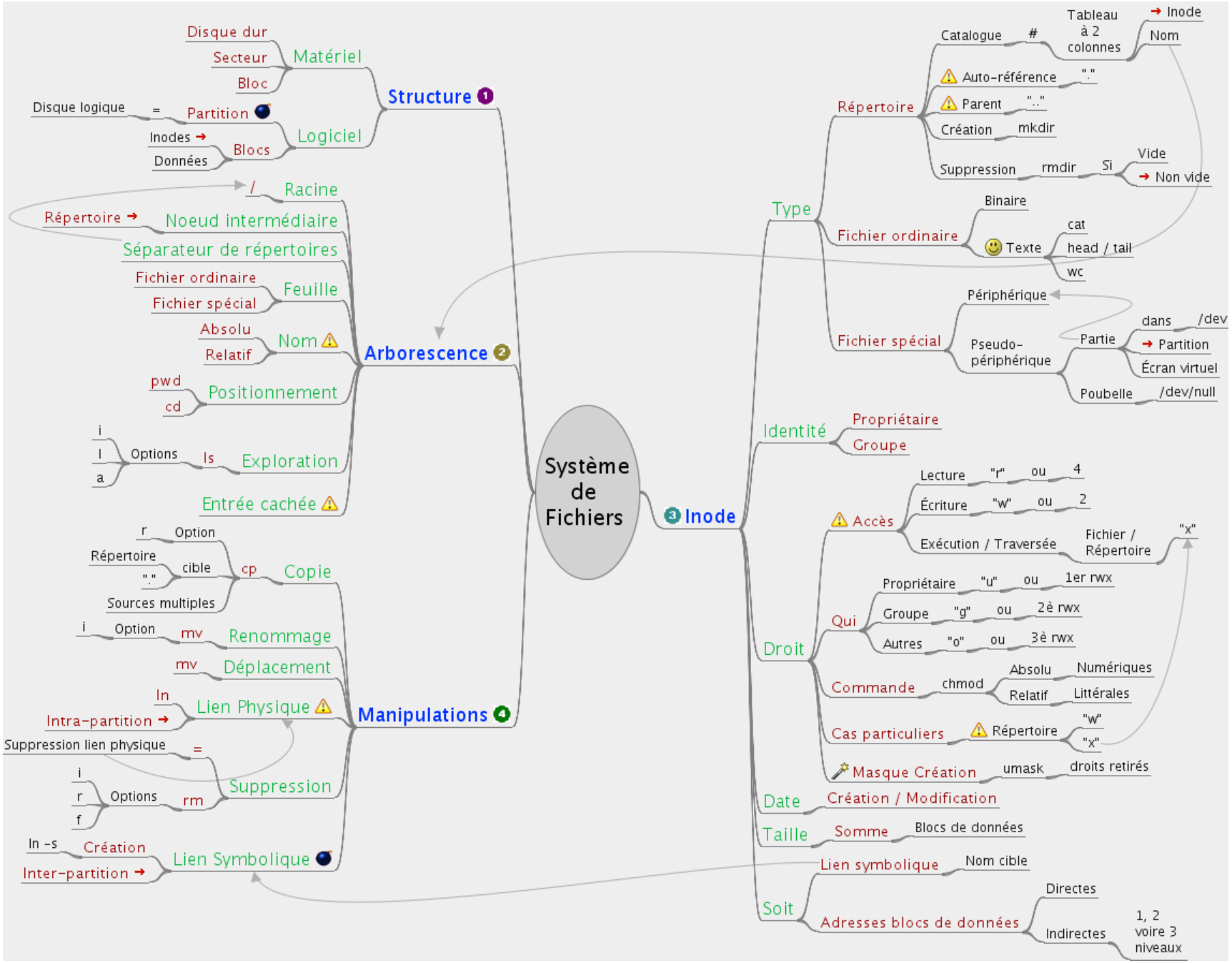
```
$ cd ; mkdir CSC3102/TP2/Droits
$ cd CSC3102/TP2/Droits
$ cp /etc/passwd .
$ ls -l
-rw-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ chmod u-r passwd
--w-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ cat passwd
cat: passwd: Permission denied
$ mkdir Rep_test
$ ls -l
drwxr-xr-x 1 brunet_e brunet_e 162 juil 29. 21:12 Rep_test
$ cd Rep_test
$ cd ..
$
```

Démonstration

```
$ cd ; mkdir CSC3102/TP2/Droits
$ cd CSC3102/TP2/Droits
$ cp /etc/passwd .
$ ls -l
-rw-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ chmod u-r passwd
--w-r--r-- 1 brunet_e brunet_e 162 juil 29. 21:00 passwd
$ cat passwd
cat: passwd: Permission denied
$ mkdir Rep_test
$ ls -l
drwxr-xr-x 1 brunet_e brunet_e 162 juil 29. 21:12 Rep_test
$ cd Rep_test
$ cd ..
$ chmod u-x Rep_test
bash: cd : Rep_test: Permission denied
$
```

Conclusion

- Concepts clés :
  - Arborescence, racine du système de fichier, répertoire de connexion, répertoire courant (ou de travail)
  - Chemin absolu, chemin relatif
  - Droits d'accès
  - Partition, inode
  - Fichier, répertoire, liens (physique et symbolique)
- Commandes clés :
  - `pwd`, `cd`, `ls`
  - `chmod`, `umask`
  - `mkdir`, `ln`, `rm`, `rmdir`, `cp`, `mv`


Système de Fichiers
Structure 1
Matériel
Disque dur
Secteur
Bloc
Logiciel
Partition
Disque logique
=
Blocs
Inodes
Données
Arborescence 2
Racine
/
Noeud intermédiaire
Répertoire
Séparateur de répertoires
Feuille
Fichier ordinaire
Fichier spécial
Nom
Absolu
Relatif
Positionnement
pwd
cd
Exploration
ls
Options
i
l
a
Entrée cachée
Manipulations 4
Copie
cp
Option
r
cible
Répertoire
"."
Sources multiples
Renommage
mv
Option
i
Déplacement
mv
Lien Physique
ln
Intra-partition
Suppression
rm
Options
i
r
f
Suppression lien physique
=
Lien Symbolique
ln -s
Création
Inter-partition
Inode 3
Type
Répertoire
Catalogue
#
Tableau à 2 colonnes
Inode
Nom
Auto-référence
"."
Parent
".."
Création
mkdir
Suppression
rmdir
Si
Vide
Non vide
Fichier ordinaire
Binaire
Texte
cat
head / tail
wc
Fichier spécial
Périphérique
Pseudo-périphérique
Partie
dans
/dev
Partition
Écran virtuel
Poubelle
/dev/null
Identité
Propriétaire
Groupe
Droit
Accès
Lecture
"r"
ou
4
Écriture
"w"
ou
2
Exécution / Traversée
Fichier / Répertoire
"x"
Qui
Propriétaire
"u"
ou
1er rwx
Groupe
"g"
ou
2è rwx
Autres
"o"
ou
3è rwx
Commande
chmod
Absolu
Numériques
Relatif
Littérales
Cas particuliers
Répertoire
"w"
"x"
Masque Création
umask
droits retirés
Date
Création / Modification
Taille
Somme
Blocs de données
Soit
Lien symbolique
Nom cible
Adresses blocs de données
Directes
Indirectes
1, 2 voire 3 niveaux

En route pour le TP !