INSTITUT Mines-Télécom

TELECOM SudParis

Système de Fichiers

CSC3102 – Introduction aux systèmes d'exploitation

Élisabeth Brunet & Gaël Thomas

Système de Fichiers

- Besoin de mémoriser des informations
  - Photos, PDF, données brutes, exécutables d'applications, le système d'exploitation 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 types de systèmes de fichiers
  - NTFS pour Windows, ext2, ext3, ext4 pour Linux, HFSX pour Mac-OS
  - FAT pour les clés USB, ISO pour les CD
  - … et des myriades d'autres types de systèmes de fichiers

- Le système de fichiers vu par un processus
- Le système de fichiers sur disque
- Les commandes utilisateurs
- Les droits d'accès

Qu'est-ce qu'un fichier

- Un fichier est la réunion de
  - Un contenu, c'est-à-dire un ensemble ordonné d'octets
  - Un propriétaire
  - Des horloges scalaires (création, dernier accès, dernière modif)
  - Des droits d'accès (en lecture, en écriture, en exécution)

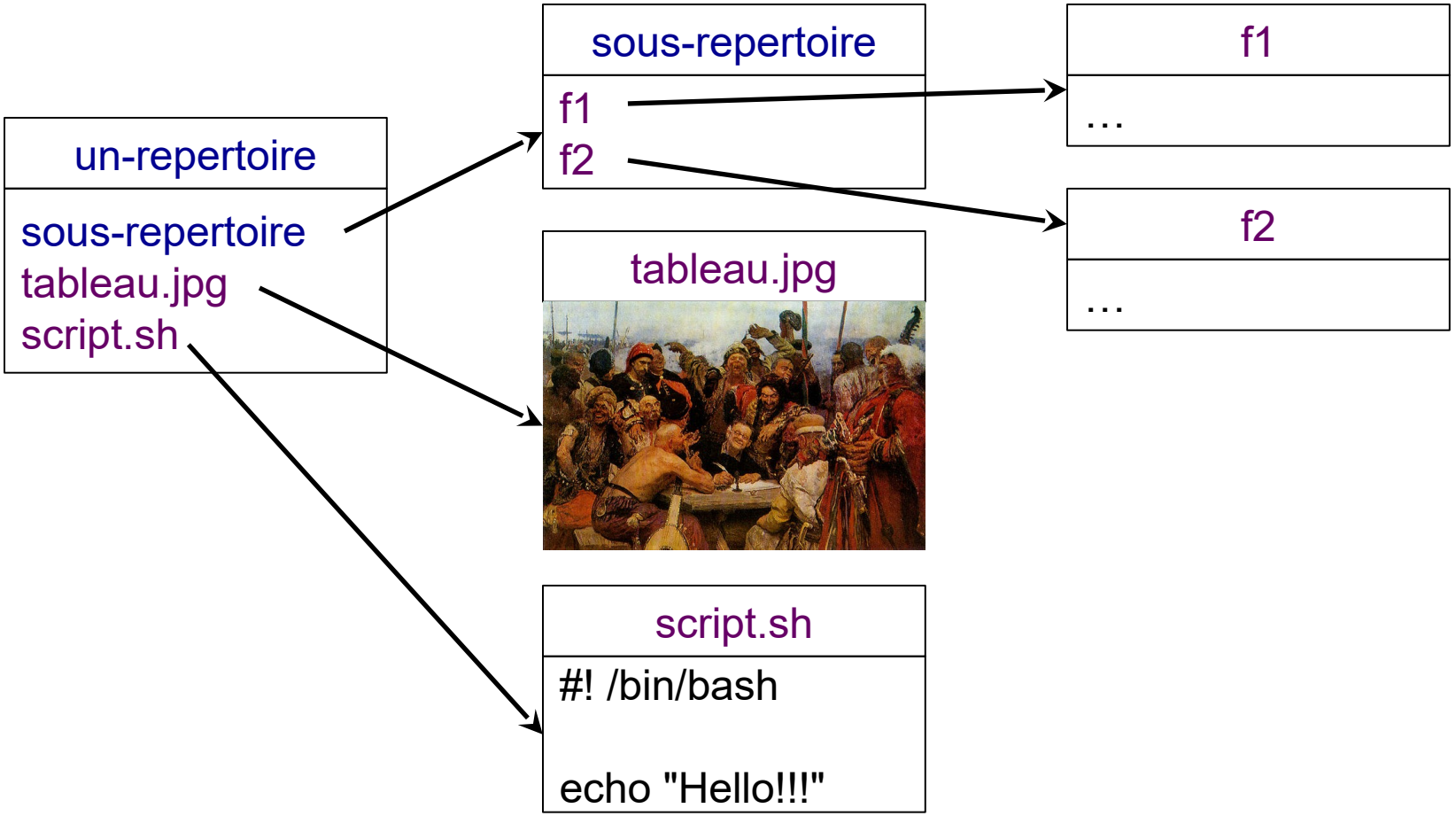
- Attention : c'est inattendu, mais un fichier est indépendant de son nom (c.-à-d., le nom ne fait pas parti du fichier et un fichier peut avoir plusieurs noms)

On stocke de nombreux fichiers

- Plusieurs centaines de milliers de fichiers dans un ordinateur
  - Plusieurs milliers gérés/utilisés directement par l'utilisateur
  - Plusieurs centaines de milliers pour le système et les applications
- Problème : comment retrouver facilement un fichier parmi des centaines de milliers ?
- Solution : en rangeant les fichiers dans des répertoires (aussi appelés dossiers)

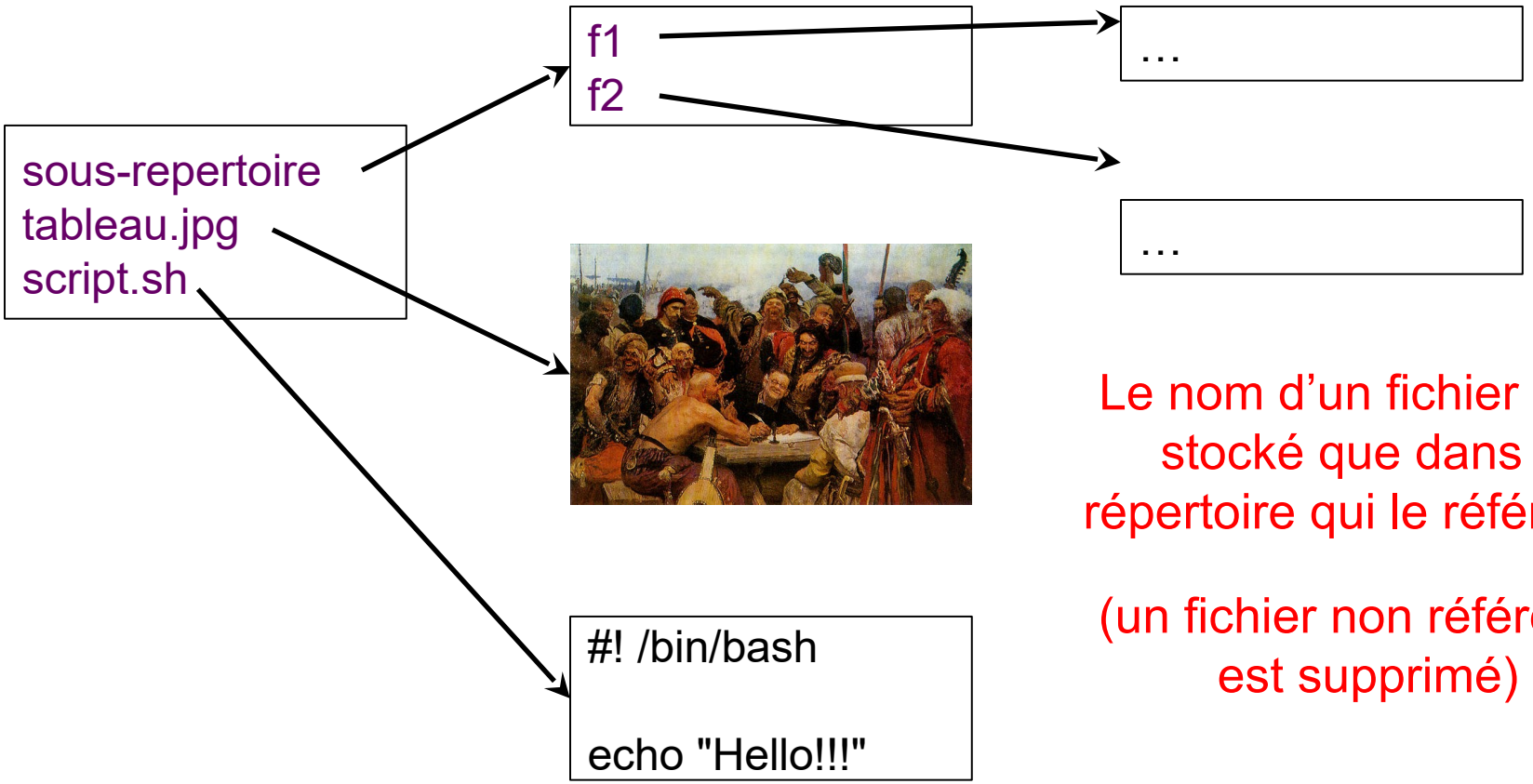
Organisation en répertoires

- Répertoire = fichier spécial qui associe des noms à des fichiers



Organisation en répertoires

- Répertoire = fichier spécial qui associe des noms à des fichiers

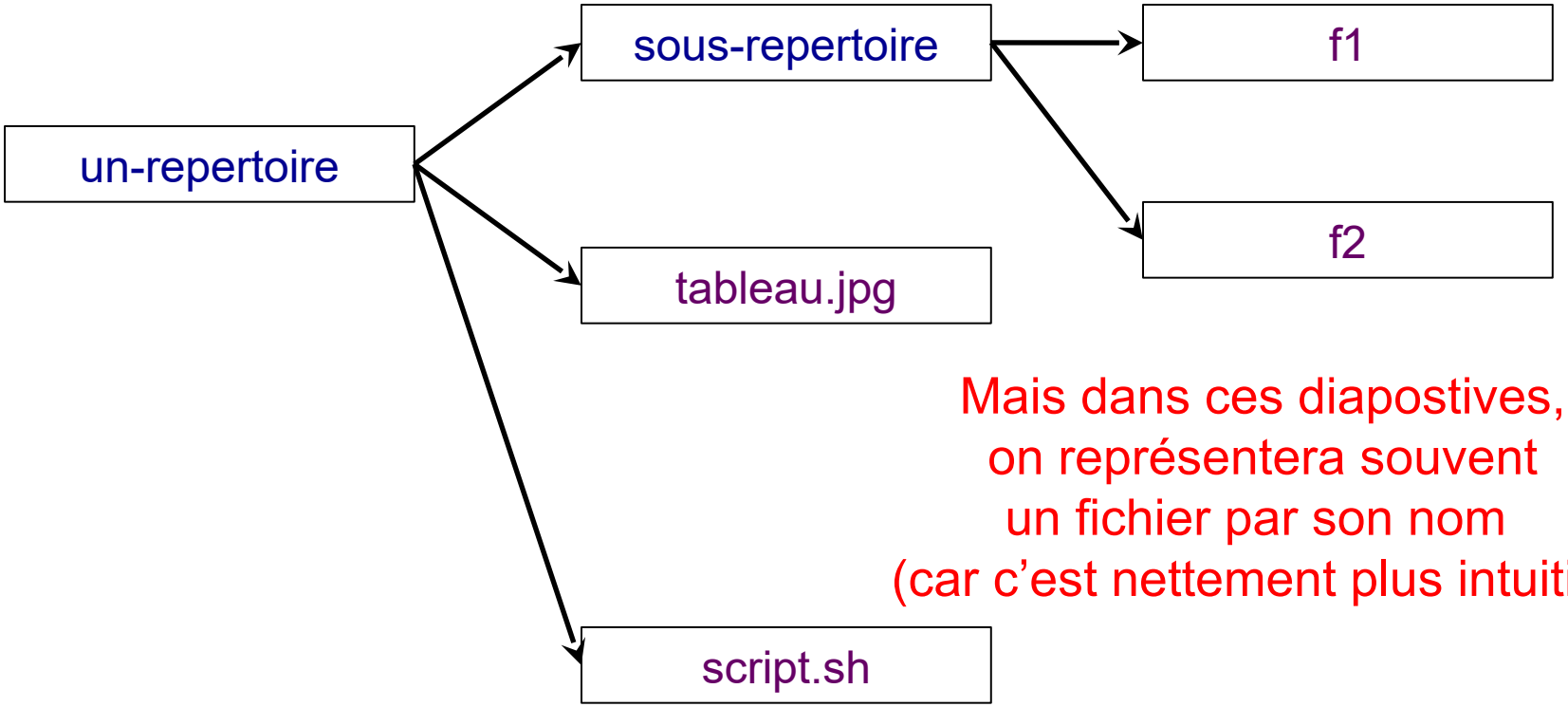


Le nom d'un fichier n'est stocké que dans le répertoire qui le référence

(un fichier non référencé est supprimé)

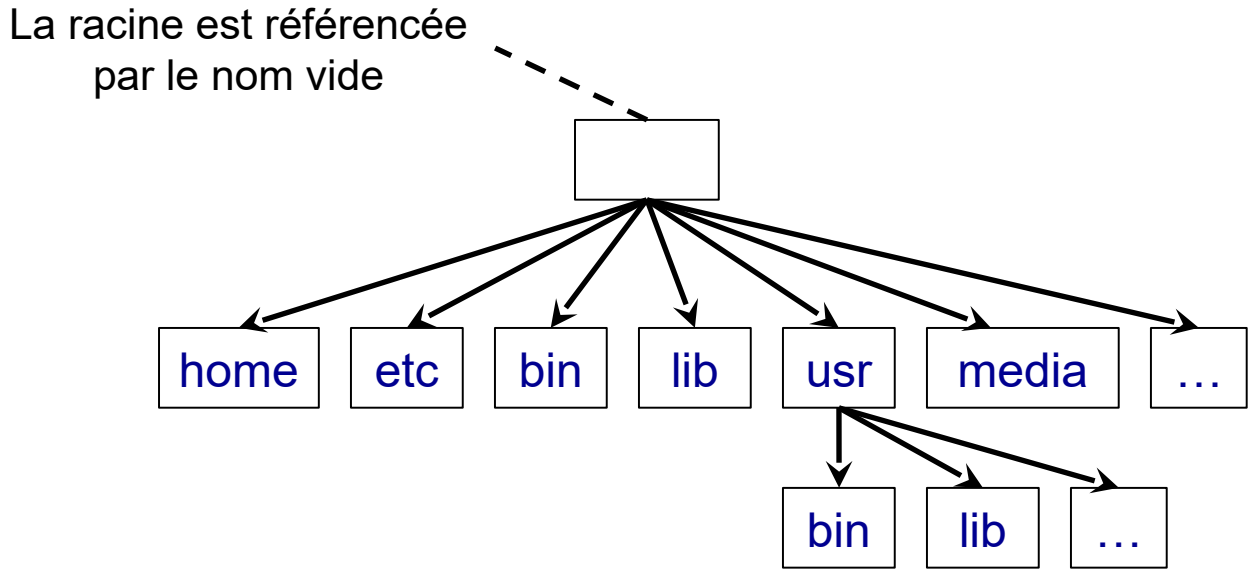
Organisation en répertoires

- Répertoire = fichier spécial qui associe des noms à des fichiers



Mais dans ces diapostives,
on représentera souvent
un fichier par son nom
(car c'est nettement plus intuitif !)

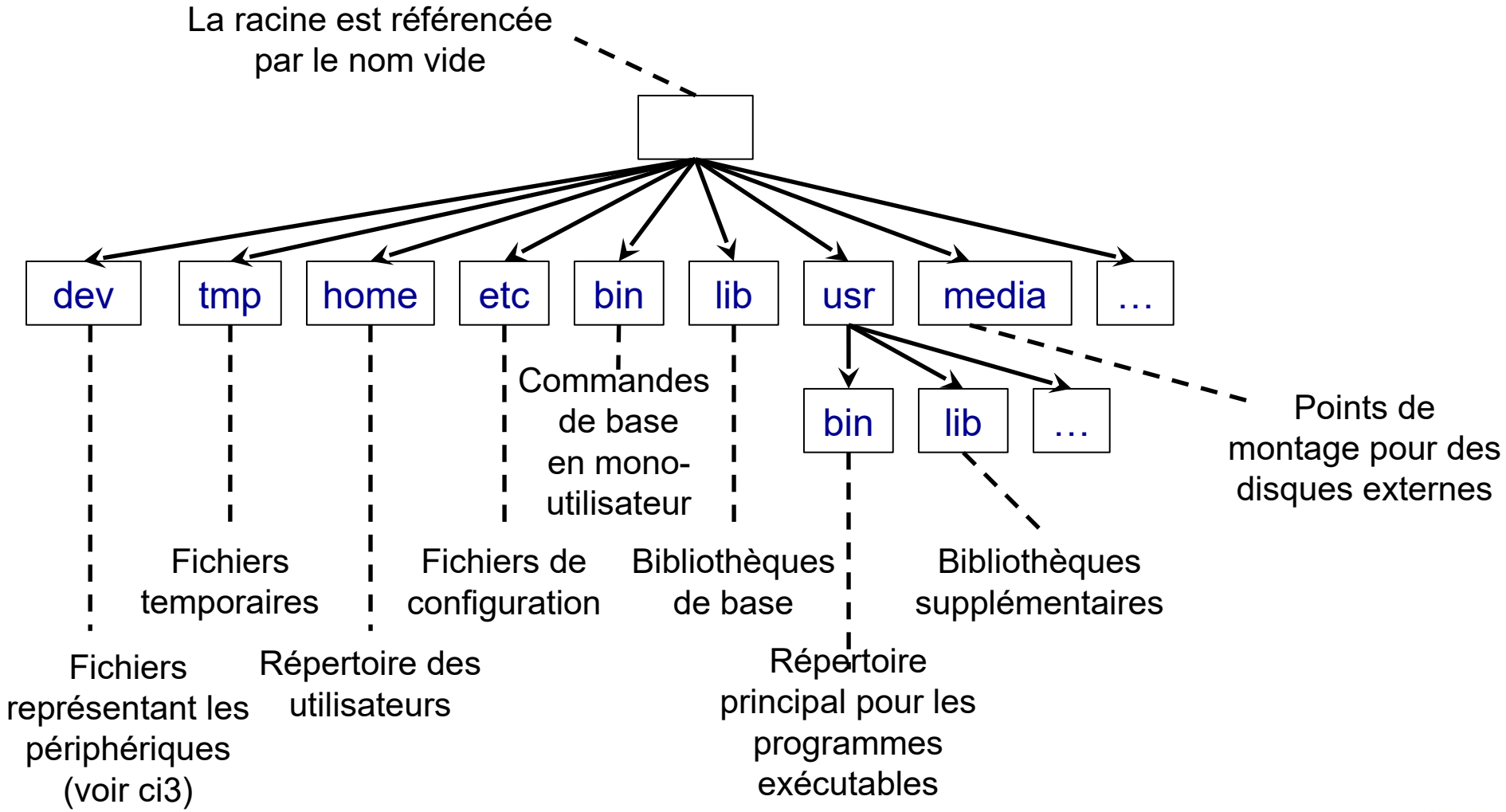
Arborescence standard des systèmes d'exploitation UNIX



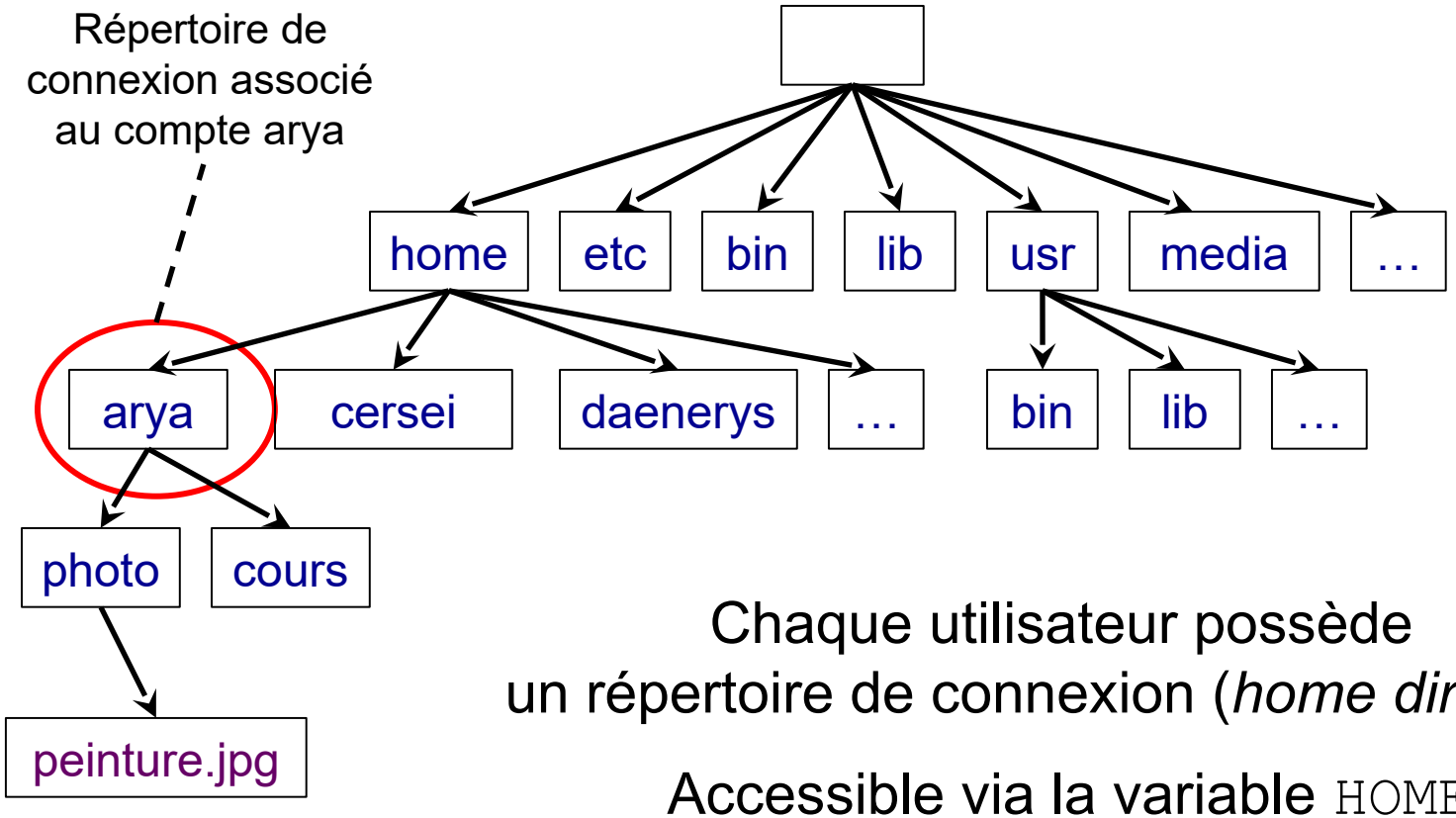
La plupart des systèmes d'exploitation Unix (GNU/Linux, BSD, MacOS…) utilisent une arborescence de base standardisée (seul Windows utilise une arborescence réellement différente)

Vous pouvez la consulter en faisant : `man hier` (pour hierarchy)

Arborescence standard des systèmes d'exploitation UNIX



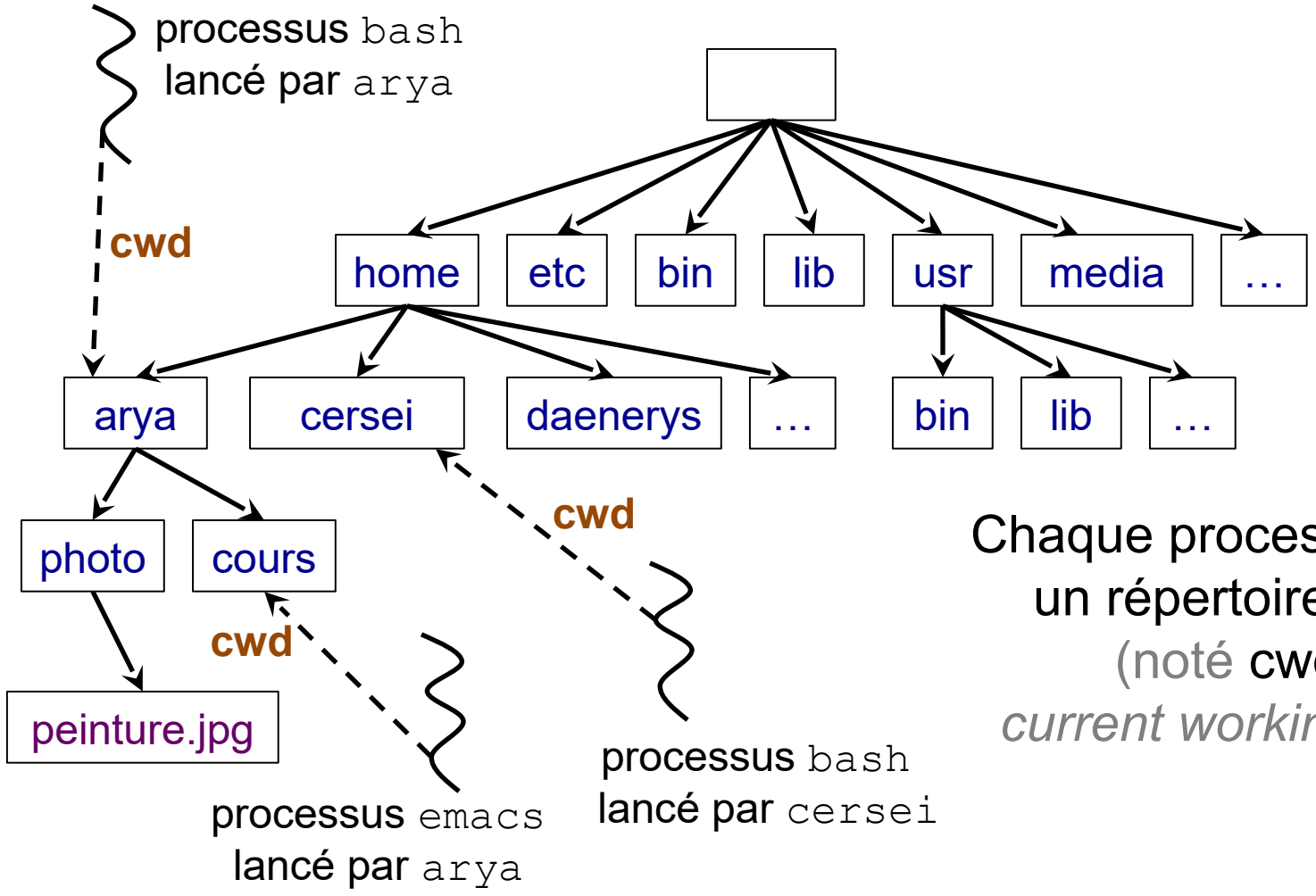
Arborescence standard des systèmes d'exploitation UNIX



Chaque utilisateur possède un répertoire de connexion (*home directory*)

Accessible via la variable `HOME`

Notion de répertoire de travail



Chaque processus possède un répertoire de travail (noté cwd pour *current working directory*)

Notion de chemin

- En `bash`, le séparateur de répertoires est le caractère `/`

- Un chemin s'écrit sous la forme `a/b/c` qui référence
  - le fichier `c`
  - se trouvant dans le répertoire `b`
  - se trouvant lui même dans le répertoire `a`

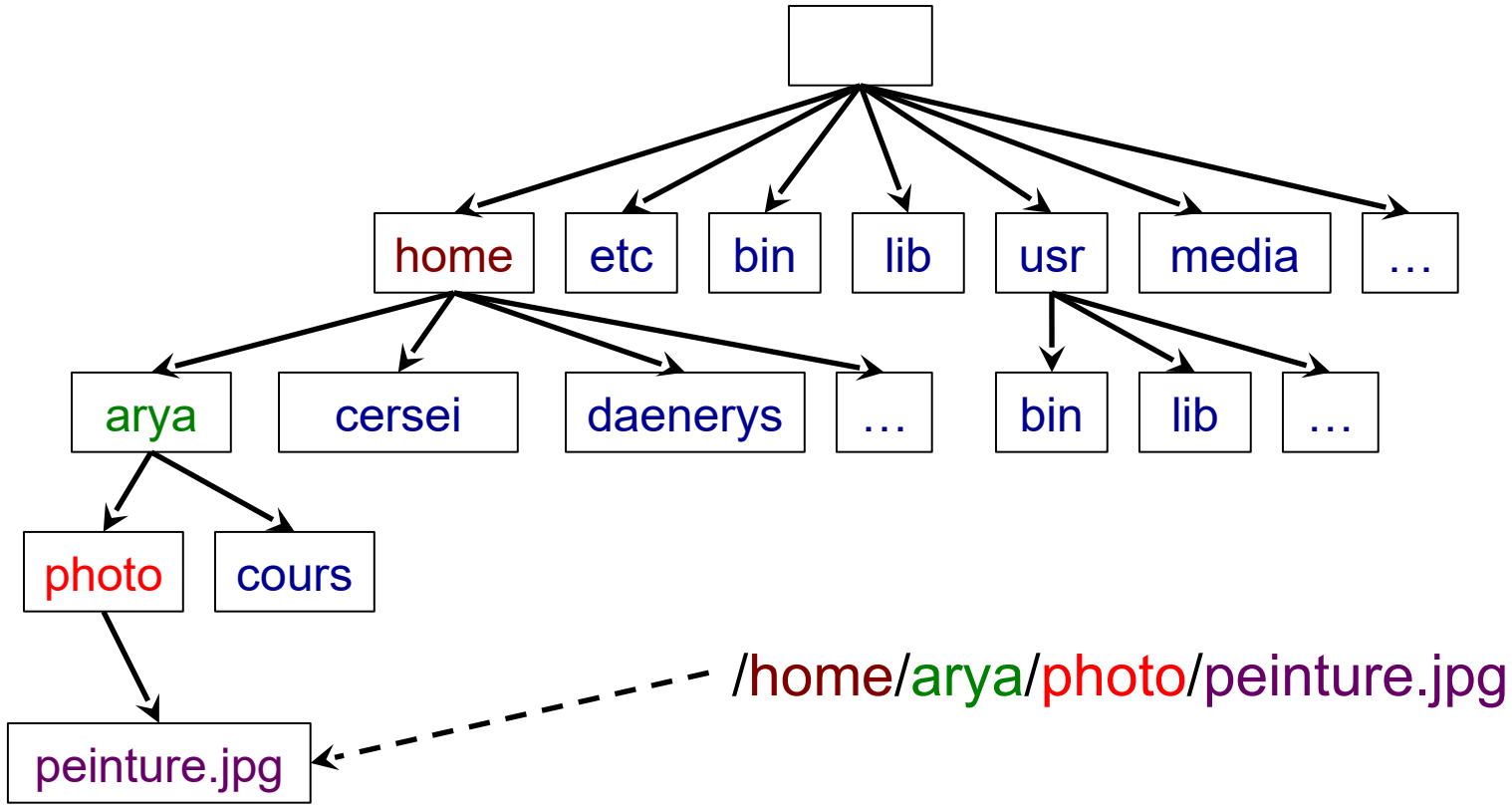
- Un chemin absolu part de la racine du système de fichiers

  Commence par le nom vide (racine), par exemple `/a/b/c`

- Un chemin relatif part du répertoire de travail du processus

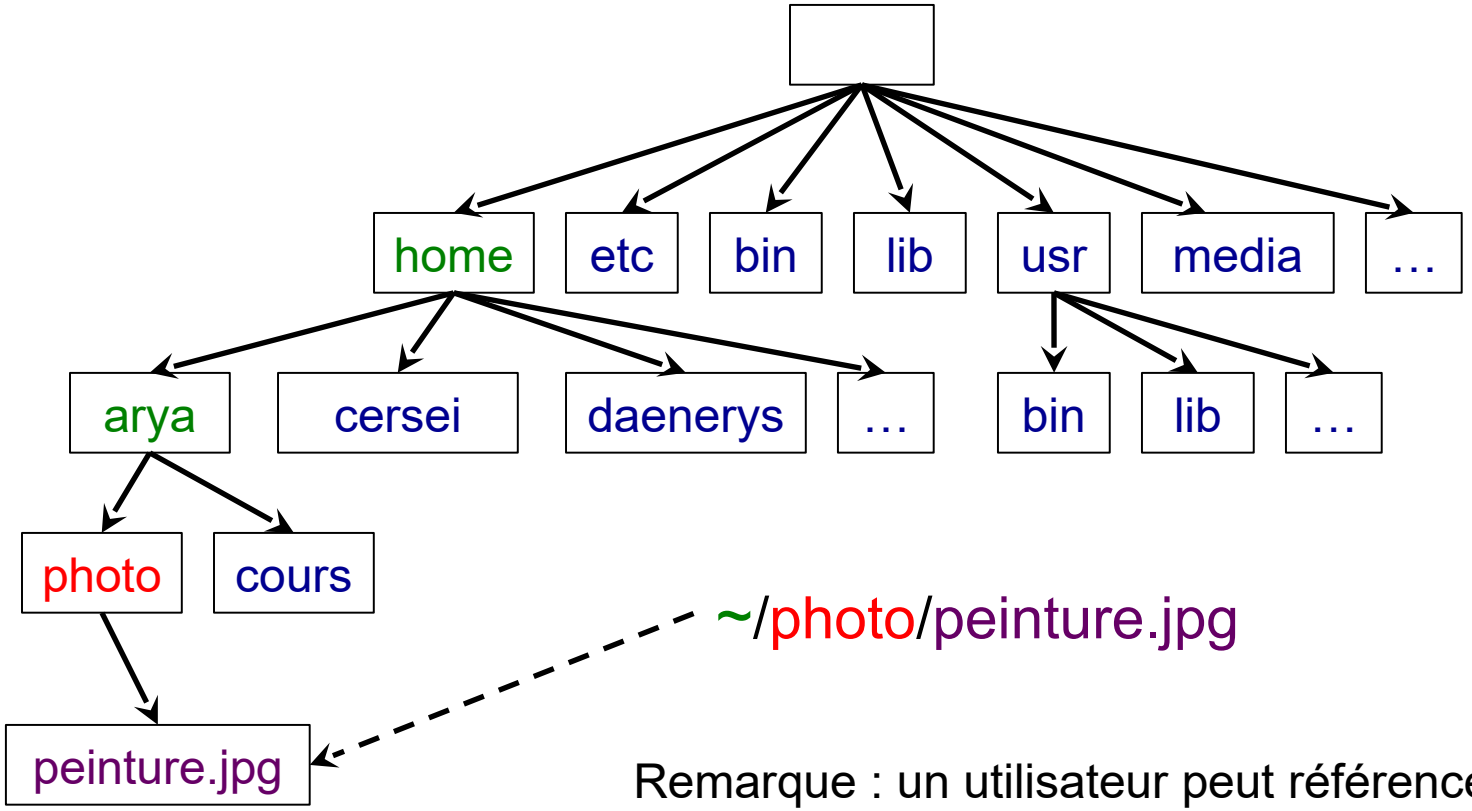
  Commence par un nom non vide, par exemple `a/b/c`

Exemple de chemin absolu (1/2)



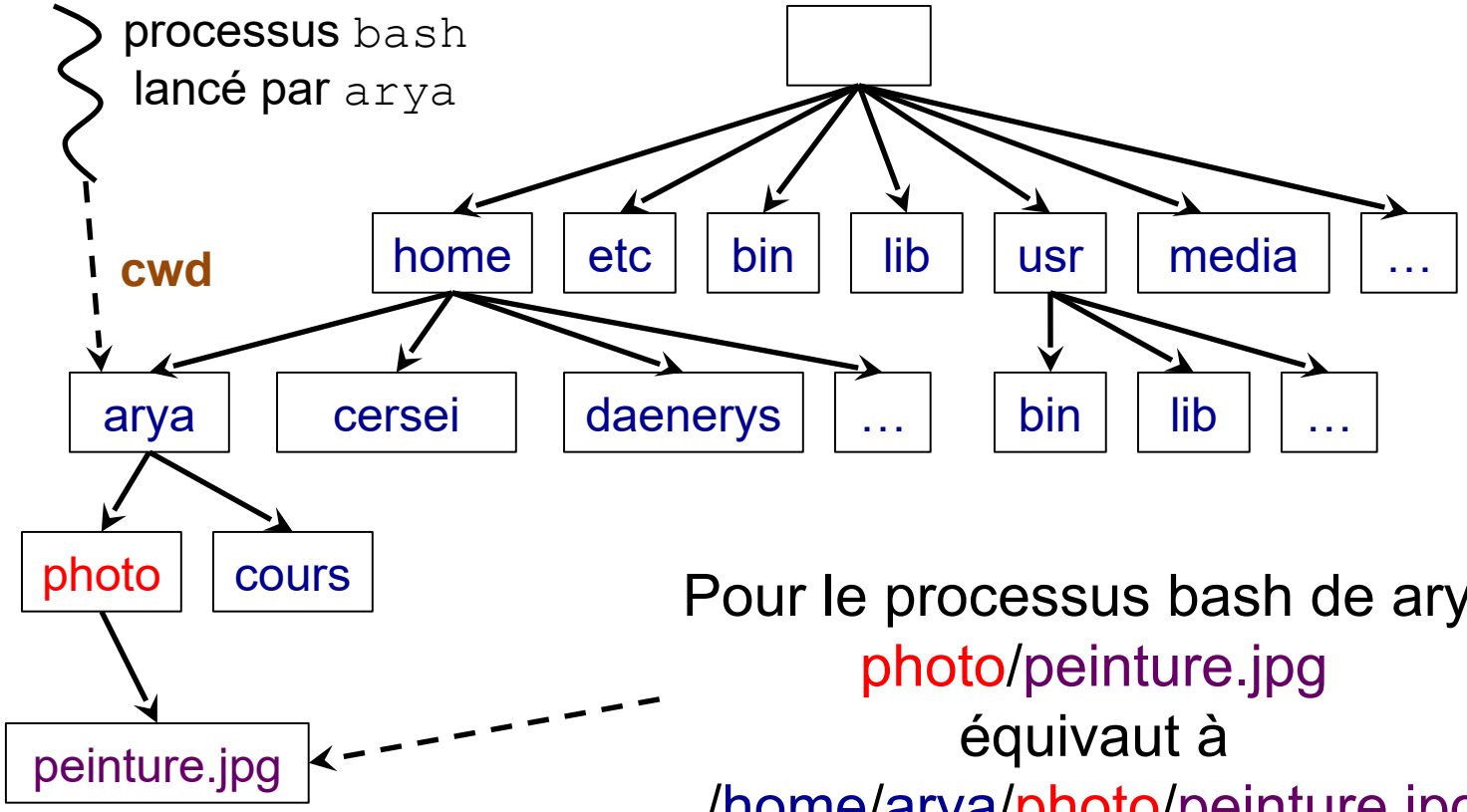
Exemple de chemin absolu (2/2)

Un utilisateur peut utiliser ~ comme raccourci pour son répertoire de connexion



Remarque : un utilisateur peut référencer le répertoire de connexion d'un autre utilisateur avec `~name` (par exemple `~arya/photo/peinture.jpg`)

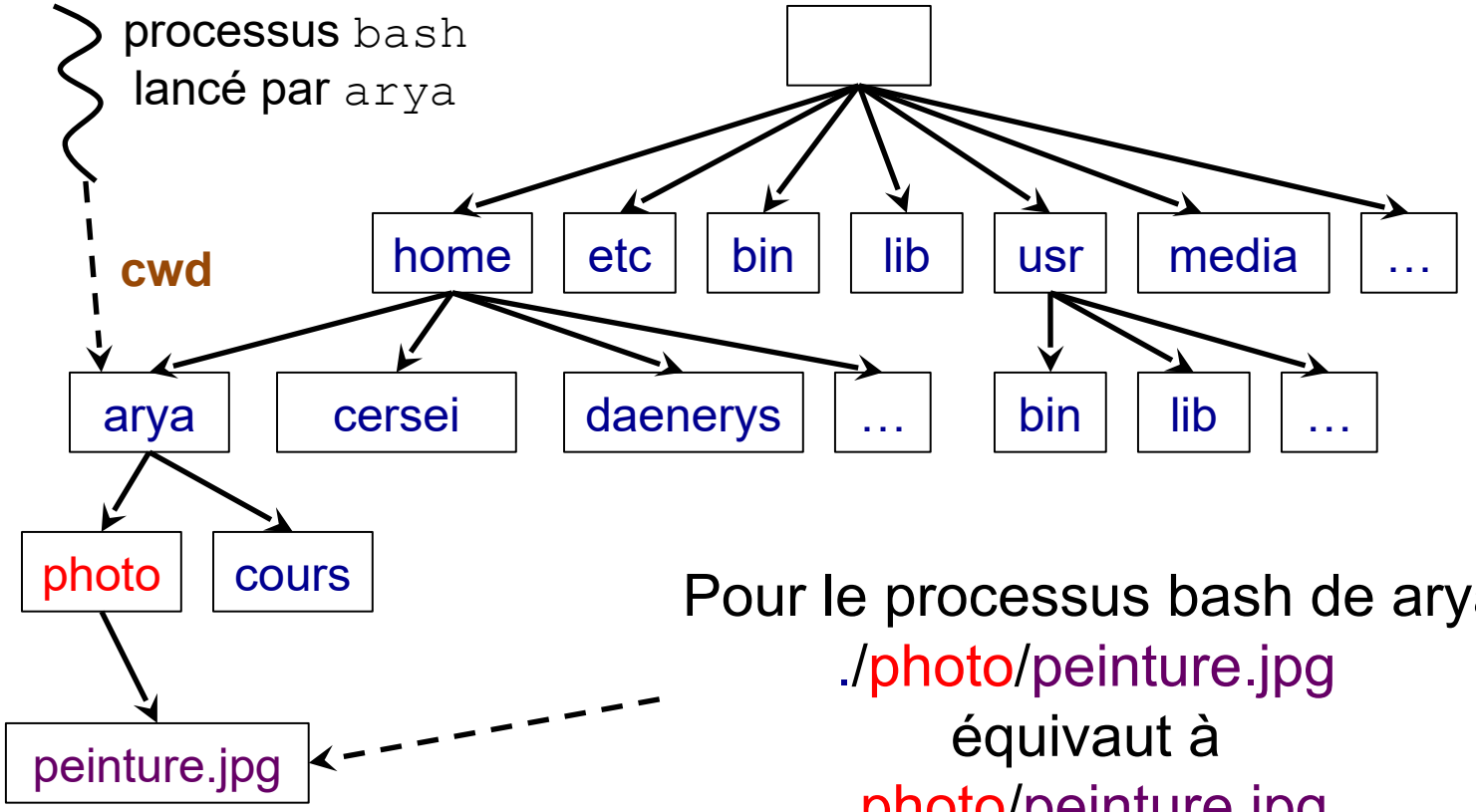
Exemple de chemin relatif (1/3)



Pour le processus bash de arya
photo/peinture.jpg
équivaut à
/home/arya/photo/peinture.jpg

Exemple de chemin relatif (2/3)

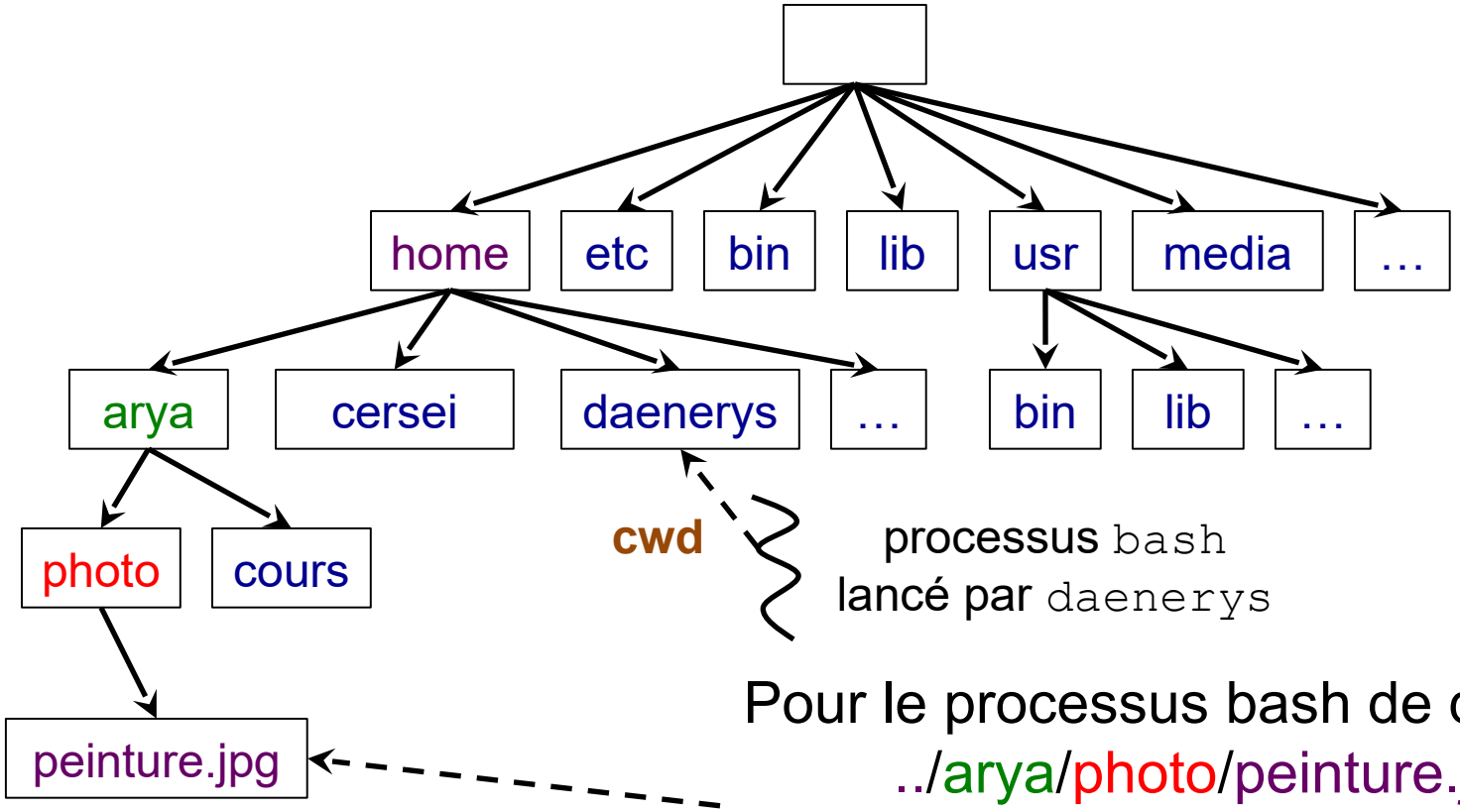
Chaque répertoire possède un fichier nommé `.` s'auto-référençant



Pour le processus bash de arya
./photo/peinture.jpg
équivaut à
photo/peinture.jpg

Exemple de chemin relatif (3/3)

Chaque répertoire possède un fichier nommé `..` référençant son parent



Pour le processus bash de daenerys
../arya/photo/peinture.jpg
équivaut à
/home/arya/photo/peinture.jpg

Remarque

- Dans `bash`, quand vous écrivez `./script.sh`, vous référencez le fichier `script.sh` du répertoire de travail du processus `bash` de votre terminal

Exemple

```
#! /bin/bash

echo "Bonjour, vous êtes dans le répertoire $PWD"
echo "Votre maison se trouve en $HOME"
echo "Et vous avez lancé le script $0"
```

/home/gael/tmp/script.sh

```
$ ./script.sh
Bonjour, vous êtes dans le répertoire /home/gael/tmp
Votre maison se trouve en /home/gael
Et vous avez lancé le script ./script.sh
$
```

Explorer l'arborescence de fichiers

- `cd chem` : *change directory*

  ⇒ change le répertoire courant vers `chem`

  Exemple : `cd ../cersei; cd /home/arya/photo`

  (sans argument, `cd` va dans votre répertoire de connexion)

- `pwd` : *print working directory*

  ⇒ affiche le répertoire de travail (⇔ `echo $PWD`)

Explorer l'arborescence de fichiers

- `ls chem` : *list*

  ⇒ liste le chemin `chem`

  - Si `chem` absent : affiche le contenu du répertoire courant
  - Si `chem` répertoire : affiche le contenu du répertoire `chem`
  - Sinon si `chem` est un fichier : affiche le nom du fichier
  - Options utiles :

    `-a` : affiche les fichiers cachés (c.-à.d., commençant par '.')

    `-l` : affichage long (propriétaire, droits d'accès, taille etc.)

    `-d` : affiche les informations sur un répertoire au lieu de son contenu (à combiner avec `-l`)

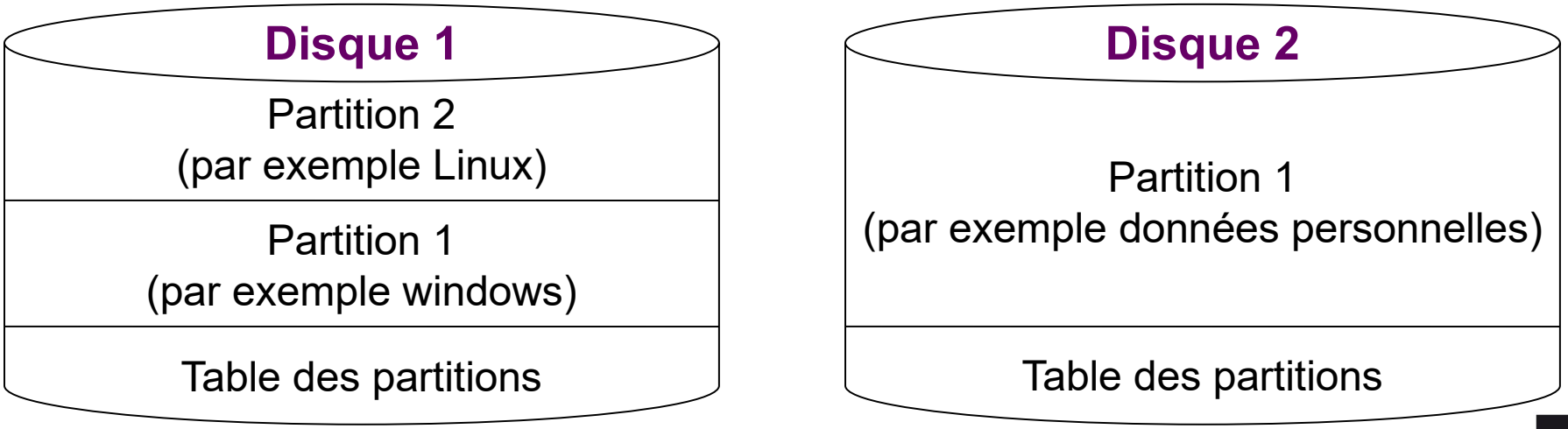
- Le système de fichiers vu par un processus
- Le système de fichiers sur disque
- Les commandes utilisateurs
- Les droits d'accès

Organisation des disques

- Une machine peut posséder plusieurs disques
- Et chaque disque peut être scindé en plusieurs partitions

  Utile pour installer plusieurs systèmes d'exploitation ou pour augmenter l'indépendance entre les données utilisateurs et le système d'exploitation

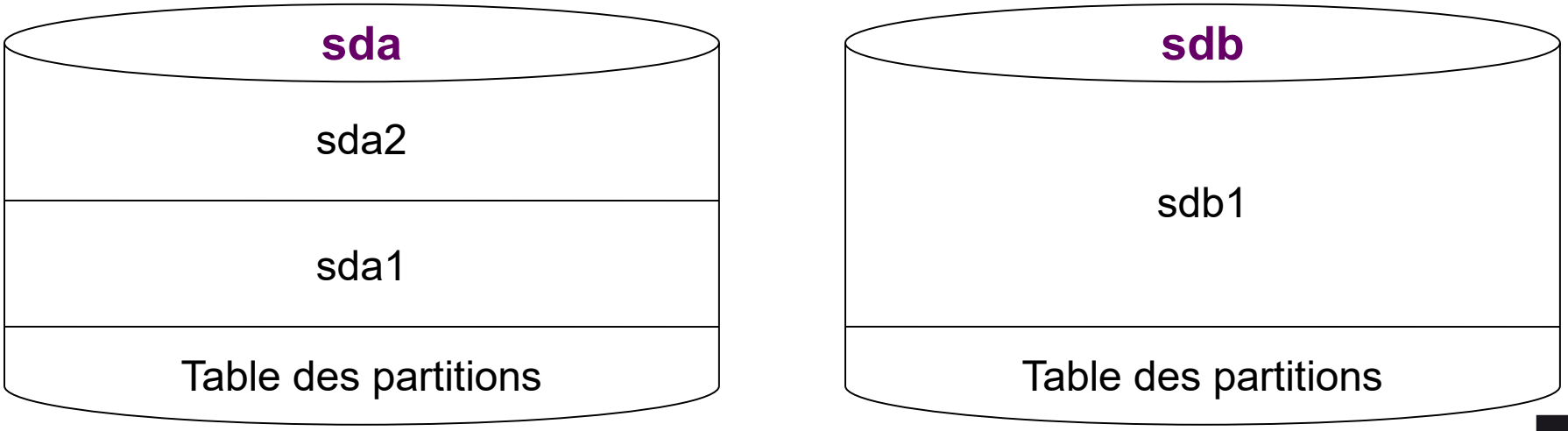
  - Chaque partition possède son système de fichiers indépendant



Les partitions dans les systèmes UNIX

- Un disque est identifié par le préfixe `sd` (*scsi drive*)
- Les disques sont numérotés `a`, `b`, `c`…
- Les partitions sont numérotées `1`, `2`, `3`…

  (vous pouvez voir les disques/partitions en faisant `ls /dev`)



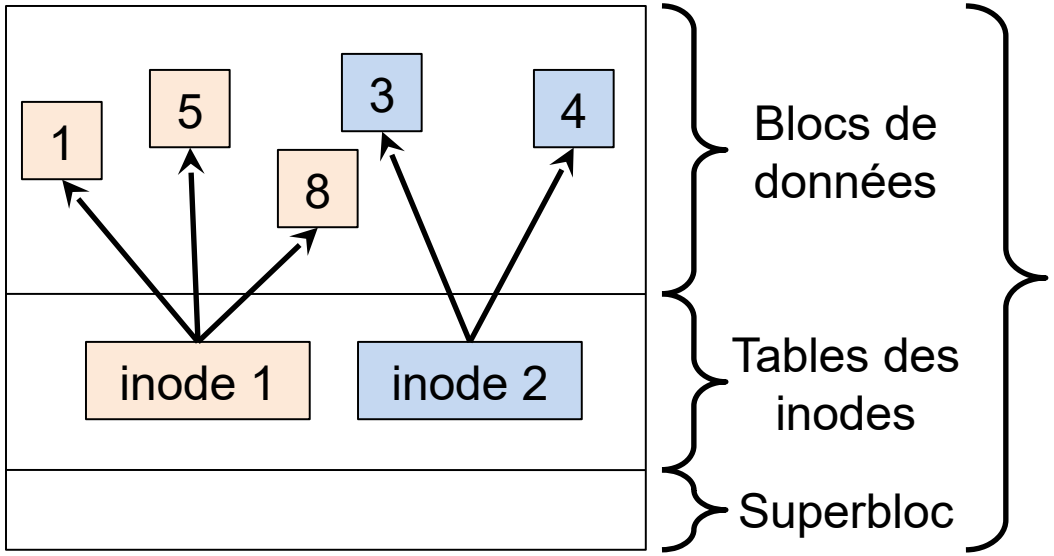
Le système de fichiers sur disque (1/2)

- 3 concepts fondamentaux
  - Le bloc : unité de transfert entre le disque et la mémoire
    (souvent 4096 octets)
  - L'inode (*index node*) : descripteur d'un fichier
    - Type de l'inode (fichier ordinaire, répertoire, autres)
    - Propriétaire, droits, dates de création/modification/accès
    - Taille
    - Liste des blocs du contenu du fichier
    - …
  - Donc, dans ce cours : fichier = inode + blocs du fichier

Le système de fichiers sur disque (2/2)

- Avec `ext`, utilisé sous GNU/Linux, trois zones principales
  - Le superbloc, au début, décrit les autres zones
  - La table des inodes contient les inodes (inode 0 = racine)
  - La zone des blocs de données contient les données des fichiers

Par exemple,
contenu de inode 1 :
4096 octets du bloc 1 puis
4096 octets du bloc 5 puis
312 octets du bloc 8



Montage d'une partition (1/2)

- Le système maintient une table des montages qui associe des chemins (points de montage) et des disques
  - / ⇒ sda1
  - /home ⇒ sdb1
  - /mnt/windows ⇒ sdb2

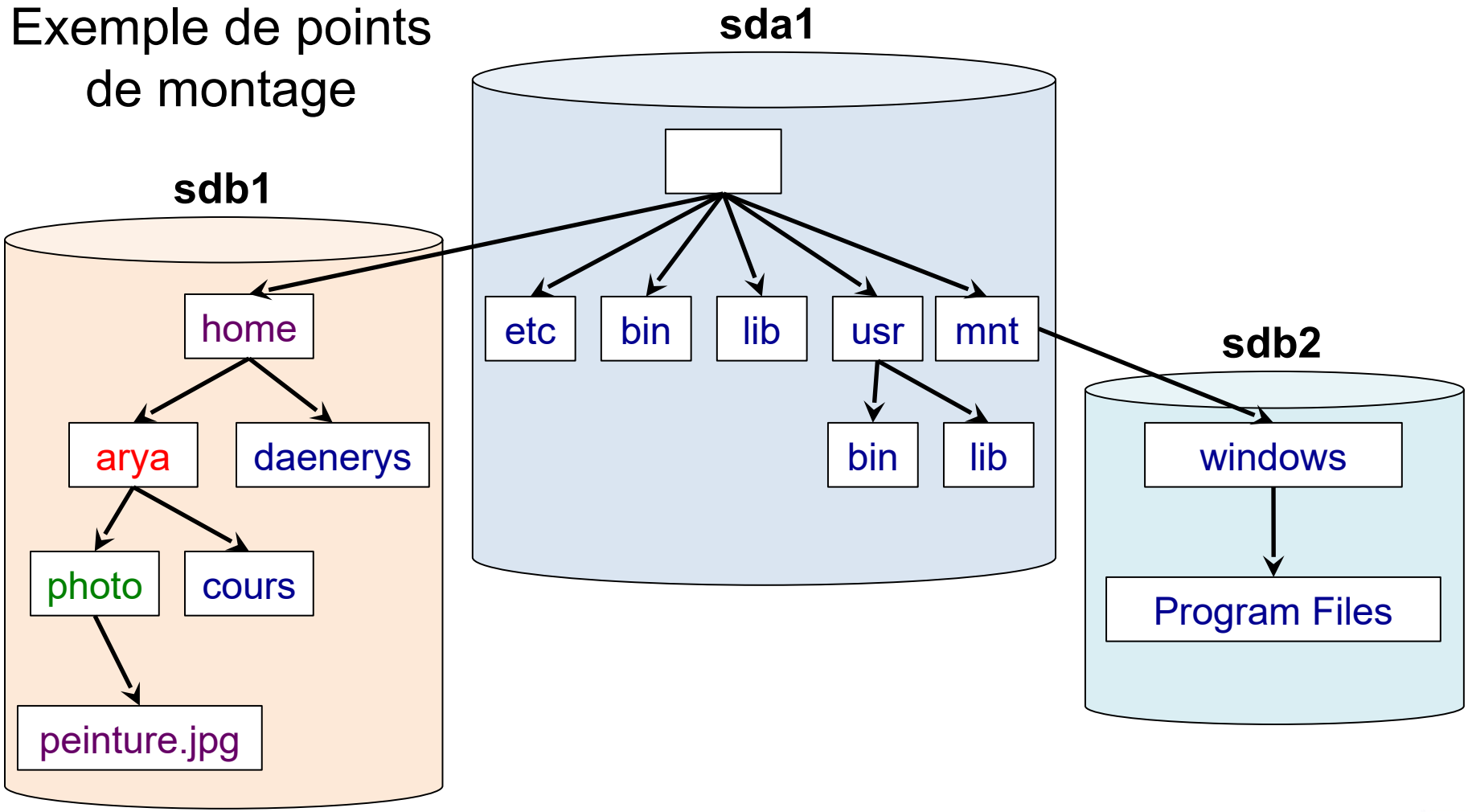
    Remarque : les partitions du disque dur peuvent se trouver sur une autre machine
    (typiquement Network File System, comme en salle TP, voir https://doc.ubuntu-fr.org/nfs)

- Lorsqu'un processus accède à un point de montage, il accède à l'inode racine du disque indiqué dans la table des montages

  Par exemple `cd /mnt/windows` accède à l'inode racine de `sdb2`

Montage d'une partition (2/2)

Exemple de points de montage



Lien direct (1/2)

- Le nom d'un inode dans un répertoire s'appelle un lien direct
  (*hard link* en anglais, aussi appelé parfois lien dur, physique ou matériel)

- On peut créer plusieurs liens directs vers le même inode

  Commande `ln chem-cible chem-lien`

  - Aucune différence entre le nom original et le nouveau nom
  - Facilite l'accès à des fichiers à partir d'emplacements connus

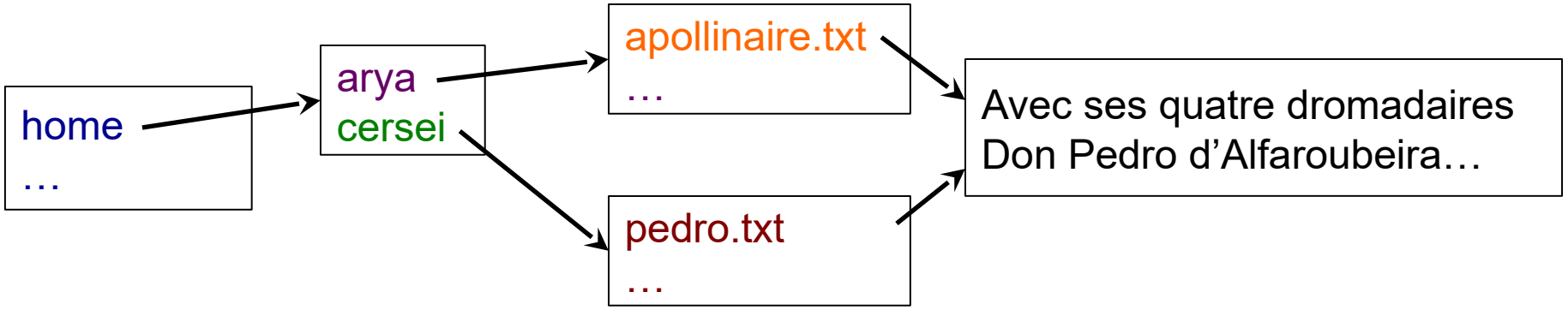
Lien direct (1/2)

- Le nom d'un inode dans un répertoire s'appelle un lien direct
  (*hard link* en anglais, aussi appelé parfois lien dur, physique ou matériel)

- On peut créer plusieurs liens directs vers le même inode

  Commande `ln chem-cible chem-lien`

  - Aucune différence entre le nom original et le nouveau nom
  - Facilite l'accès à des fichiers à partir d'emplacements connus




```
ln /home/arya/apollinaire.txt /home/cersei/pedro.txt
```

Lien direct (2/2)

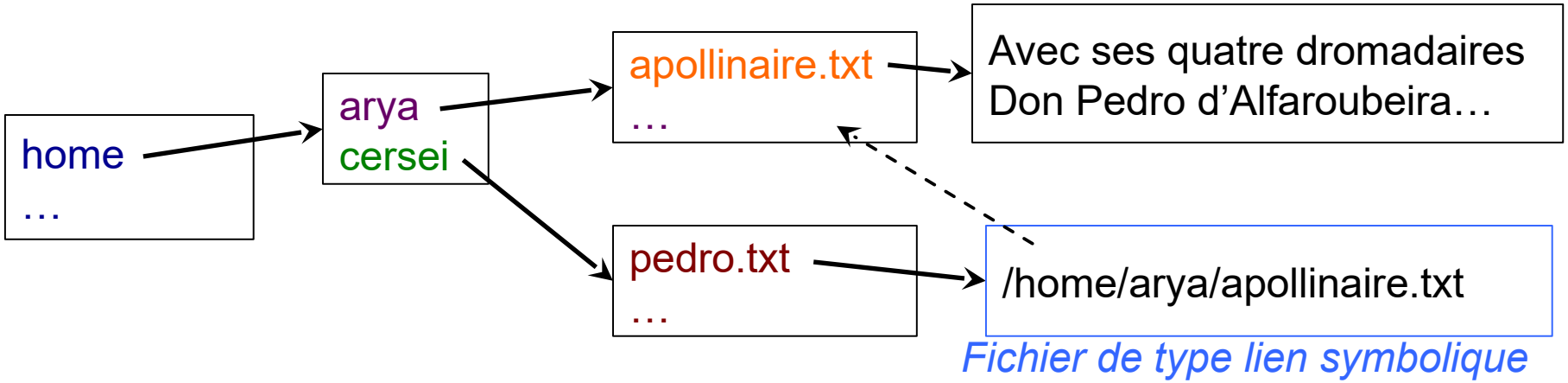
- Mais faire de multiples liens directs pour faire des raccourcis peut poser problème
  - Pour supprimer un fichier, il faut supprimer tous les liens directs vers son inode, mais les utilisateurs sont distraits et en oublient
  - Un lien direct ne peut référencer qu'un inode de la même partition (du même système de fichiers)

Notion de lien symbolique (1/2)

- Pour faire des raccourcis on utilise aussi des liens symboliques

  Comme `ln -s chem-cible chem-lien`

  - Fichier spécial (type lien) dont le contenu est un chemin cible
  - Lorsque le système doit ouvrir le fichier, il ouvre la cible à la place de l'original




```
ln -s /home/arya/apollinaire.txt /home/cersei/pedro.txt
```

Notion de lien symbolique (2/2)

- Avantages des liens symboliques
  - Dès que le fichier cible est détruit, son espace est libéré (ouvrir le lien symbolique engendre alors une erreur)
  - Un lien symbolique peut référencer un fichier quelconque, y compris appartenant à une autre partition

- Principal inconvénient des liens symboliques
  - En cas de déplacement du fichier cible, le lien symbolique peut devenir invalide

Il existe de nombreux types de fichiers

- Fichier ordinaire
- Répertoire
- Lien symbolique
- Device : un fichier qui représente un périphérique
  (disque dur, carte son, carte réseau, …)
  - Par exemple `/dev/sda1`
- Tube nommé : fichier spécial étudié en CI6
- Socket : fichier spécial proche des tubes (non étudié dans ce cours)

- Le système de fichiers vu par un processus
- Le système de fichier sur disque
- Les commandes utilisateurs
- Les droits d'accès

Commandes utilisateur

- Commandes de base sur les fichiers
  - Création
  - Suppression
  - Copie
  - Déplacement / renommage
  - Consultation
  - Recherche
- Commandes utilitaires bien pratiques
  - Principales vues en TP

Création d'un fichier

- Création d'un fichier ordinaire :
  - Au travers de logiciels
    - en particulier des éditeurs : emacs, vi, gedit, etc…
  - `touch chem` : crée fichier vide + mise à jour heures modif.
- Création d'un répertoire :
  - `mkdir rep` : *make directory*
- Création d'un lien :
  - Lien dur : `ln chem-cible chem-lien`
  - Lien symbolique : `ln -s chem-cible chem-lien`

Suppression d'un fichier (1/5)

- Supprimer un fichier (tout type, sauf répertoire)

  `rm chem` : *remove*

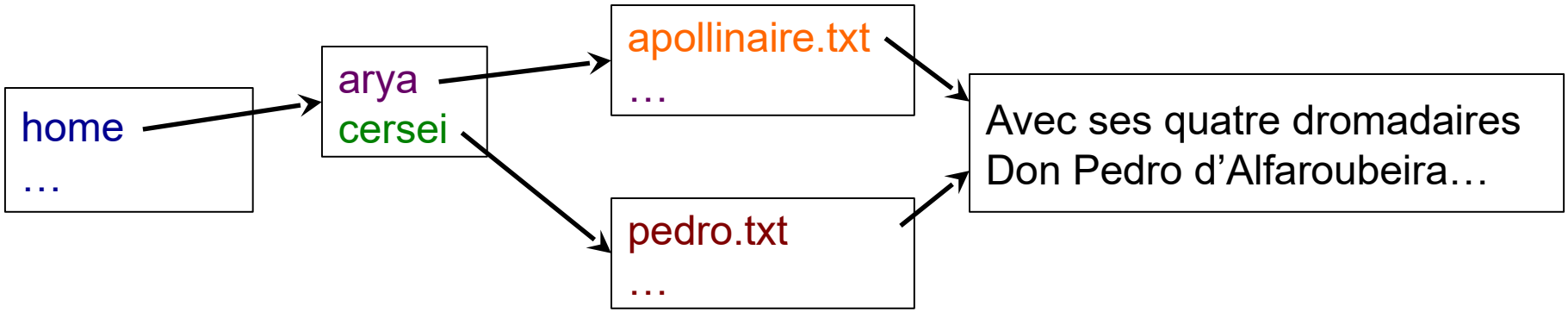
  - Suppression de l'entrée pour ce chemin dans le répertoire parent
    - Décrémentation du compteur de liens directs de l'inode
    - Libère le fichier (inode + données) si compteur tombe à zéro

Suppression d'un fichier (2/5)

- Supprimer un fichier (tout type, sauf répertoire)

  `rm chem` : *remove*

  - Suppression de l'entrée pour ce chemin dans le répertoire parent
    - Décrémentation du compteur de liens directs de l'inode
    - Libère le fichier (inode + données) si compteur tombe à zéro

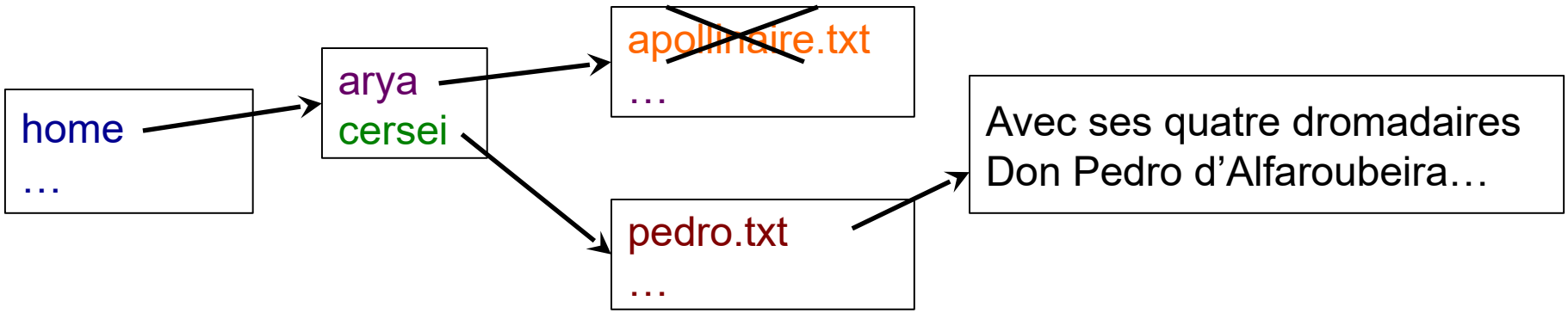


Suppression d'un fichier (3/5)

- Supprimer un fichier (tout type, sauf répertoire)

  `rm chem` : *remove*

  - Suppression de l'entrée pour ce chemin dans le répertoire parent
    - Décrémentation du compteur de liens directs de l'inode
    - Libère le fichier (inode + données) si compteur tombe à zéro



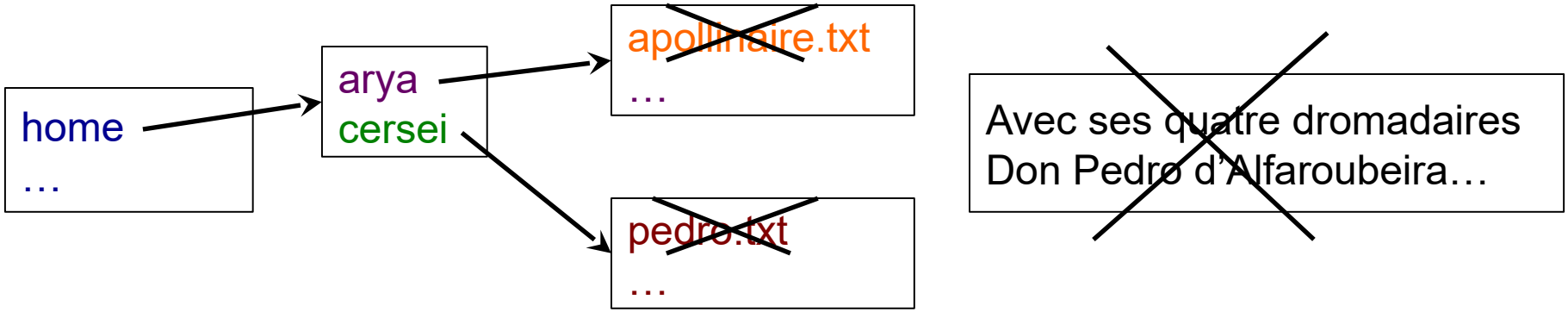
`rm /home/arya/apollinaire.txt`

Suppression d'un fichier (4/5)

- Supprimer un fichier (tout type, sauf répertoire)

  `rm chem` : *remove*

  - Suppression de l'entrée pour ce chemin dans le répertoire parent
    - Décrémentation du compteur de liens directs de l'inode
    - Libère le fichier (inode + données) si compteur tombe à zéro




```
rm /home/cersei/pedro.txt
```

Suppression d'un fichier (5/5)

- Supprimer un fichier (tout type, sauf répertoire)

  `rm chem` : *remove*

  - Suppression de l'entrée pour ce chemin dans le répertoire parent
    - Décrémentation du compteur de liens directs de l'inode
    - Libère le fichier (inode + données) si compteur tombe à zéro

- Supprimer un répertoire
  - `rmdir <rep>` : suppression d'un répertoire vide
  - `rm -r <rep>` : suppression récursive d'un répertoire et de tous les sous-fichiers (sous-répertoires inclus)

    (faites très attention avec cette commande !)
  - `rm -i <rep>` : demande confirmation avant suppression (utile !)
    - Peut être combiné avec `-r`

Copie d'un fichier (1/3)

- `cp src dest` : *copy*

  Création d'un nouvel inode et duplication des blocs de données
  - `src` correspond au chemin du fichier à copier
  - `dest`, au chemin où doit être copiée `src`

- Deux fonctionnements différents
  - Si `dest` est un répertoire, copie `src` dans le répertoire `dest` (dans ce cas, multiples copies possibles avec `cp fic1 fic2 … rep`)
  - Sinon, copie `src` sous le nom `dest`

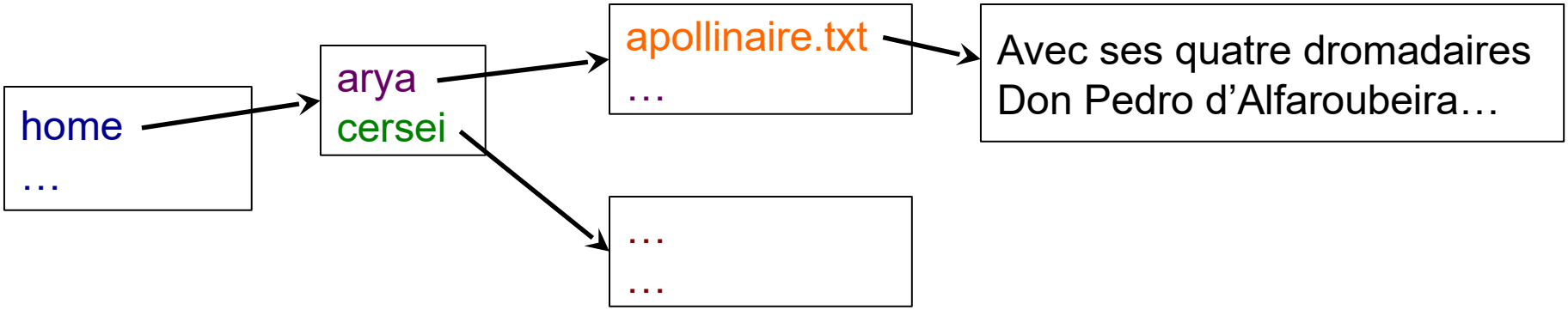
- L'option `-r` permet de copier récursivement un répertoire (sans `-r`, si `src` est un répertoire, erreur)

Copie d'un fichier (2/3)

- `cp src dest` : *copy*

  Création d'un nouvel inode et duplication des blocs de données

  - `src` correspond au chemin du fichier à copier
  - `dest`, au chemin où doit être copiée `src`

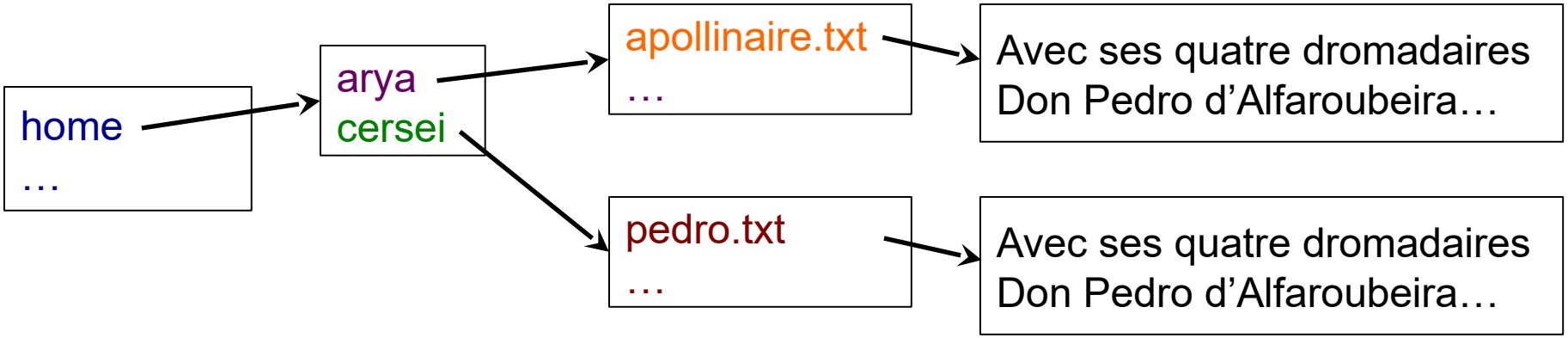


Copie d'un fichier (3/3)

- `cp src dest` : *copy*

  Création d'un nouvel inode et duplication des blocs de données

  - `src` correspond au chemin du fichier à copier
  - `dest`, au chemin où doit être copiée `src`




```
cp /home/arya/apollinaire.txt /home/cersei/pedro.txt
```

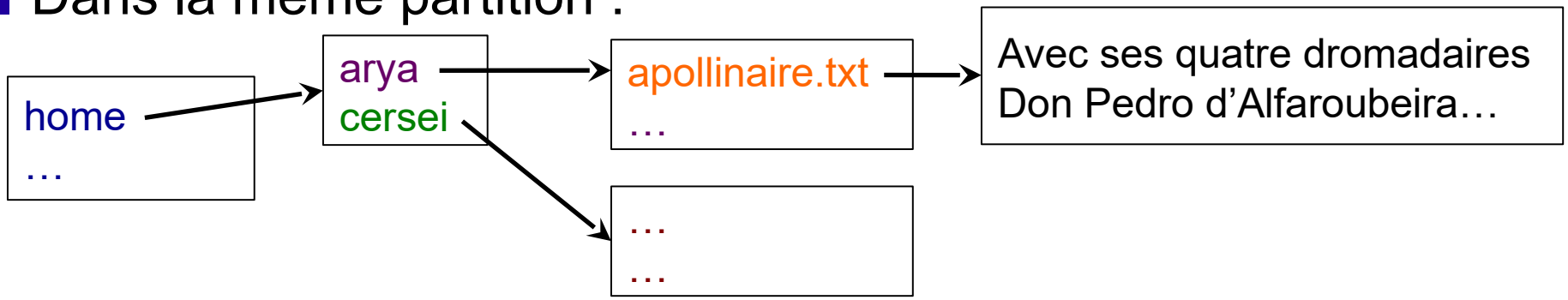
Déplacement d'un fichier (1/7)

- `mv src dest` : *move* (déplace ou renomme)
  - *`src` : fichier de type quelconque*
  - Si `dest` est un répertoire, déplace `src` dans le répertoire `dest` (dans ce cas, multiples déplacements possibles avec `mv fic1 fic2 … rep`)
  - Sinon, déplace `src` sous le nom `dest`
    - Si `dest` est dans le même répertore : renommage

- Fonctionnement :
  - Déplacement dans la même partition
    - Crée un lien direct à partir de `dest` puis supprime `src`
  - Déplacement sur une autre partition
    - Copie `src` vers `dest` puis supprime `src`

Déplacement d'un fichier (2/7)

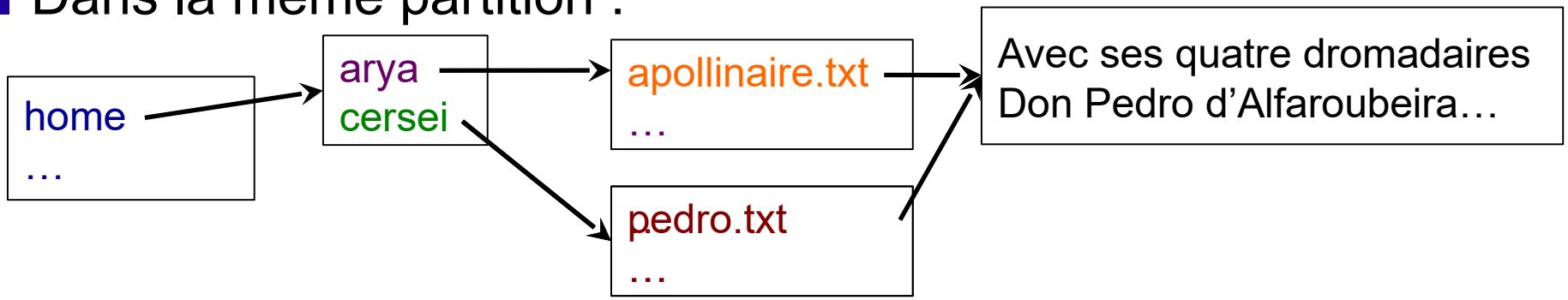
- `mv src dest` : *move* (déplace ou renomme)
  - *`src` : fichier de type quelconque*
  - Si `dest` est un répertoire, déplace `src` dans le répertoire `dest` (dans ce cas, multiples déplacements possibles avec `mv fic1 fic2 … rep`)
  - Sinon, déplace `src` sous le nom `dest`
    - Si `dest` est dans le même répertore : renommage
- Dans la même partition :



`mv /home/arya/apollinaire.txt /home/cersei/pedro.txt`

Déplacement d'un fichier (3/7)

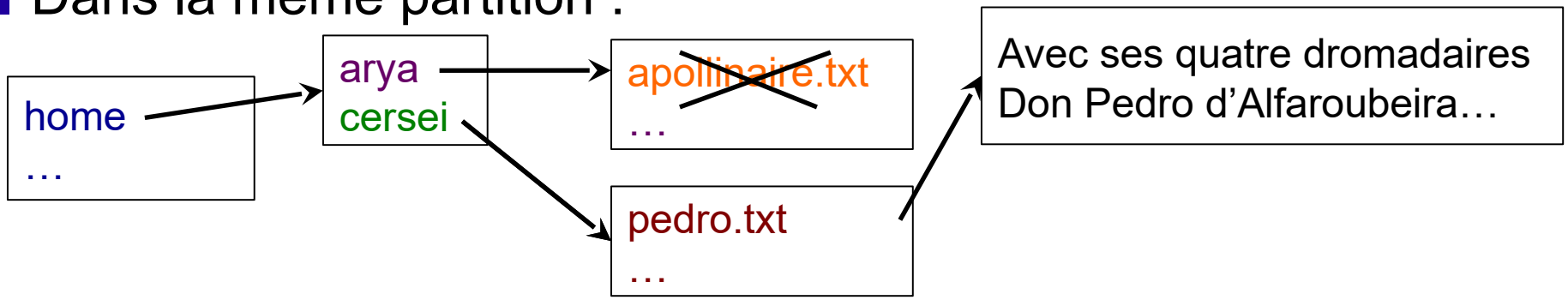
- `mv src dest` : *move* (déplace ou renomme)
  - *`src` : fichier de type quelconque*
  - Si `dest` est un répertoire, déplace `src` dans le répertoire `dest` (dans ce cas, multiples déplacements possibles avec `mv fic1 fic2 … rep`)
  - Sinon, déplace `src` sous le nom `dest`
    - Si `dest` est dans le même répertore : renommage
- Dans la même partition :



`mv /home/arya/apollinaire.txt /home/cersei/pedro.txt`

Déplacement d'un fichier (4/7)

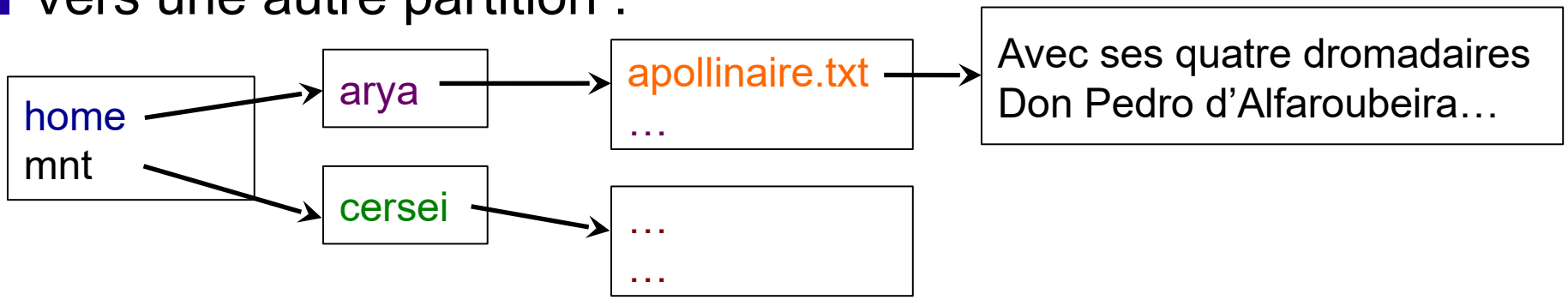
- `mv src dest` : *move* (déplace ou renomme)
  - *`src` : fichier de type quelconque*
  - Si `dest` est un répertoire, déplace `src` dans le répertoire `dest` (dans ce cas, multiples déplacements possibles avec `mv fic1 fic2 … rep`)
  - Sinon, déplace `src` sous le nom `dest`
    - Si `dest` est dans le même répertore : renommage
- Dans la même partition :




```
mv /home/arya/apollinaire.txt /home/cersei/pedro.txt
```

Déplacement d'un fichier (5/7)

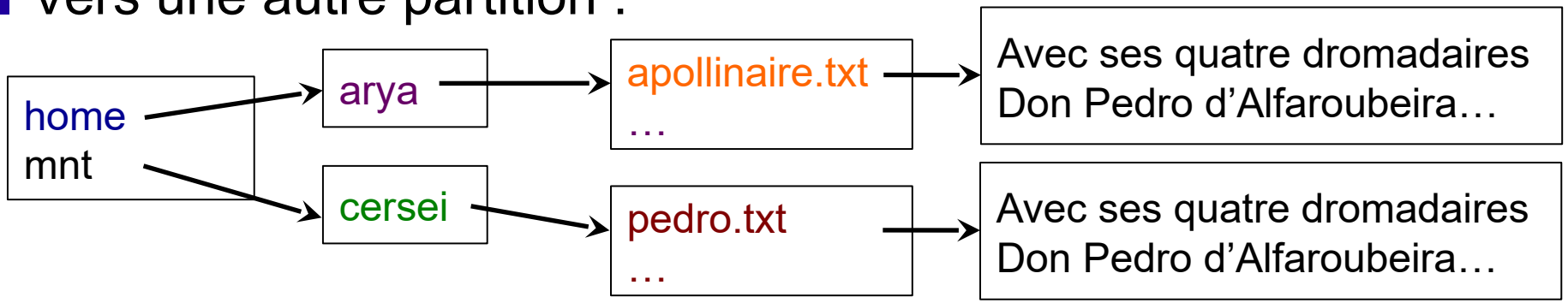
- `mv src dest` : *move* (déplace ou renomme)
  - *`src` : fichier de type quelconque*
  - Si `dest` est un répertoire, déplace `src` dans le répertoire `dest` (dans ce cas, multiples déplacements possibles avec `mv fic1 fic2 … rep`)
  - Sinon, déplace `src` sous le nom `dest`
    - Si `dest` est dans le même répertore : renommage
- Vers une autre partition :




```
mv /home/arya/apollinaire.txt /mnt/cersei/pedro.txt
```

Déplacement d'un fichier (6/7)

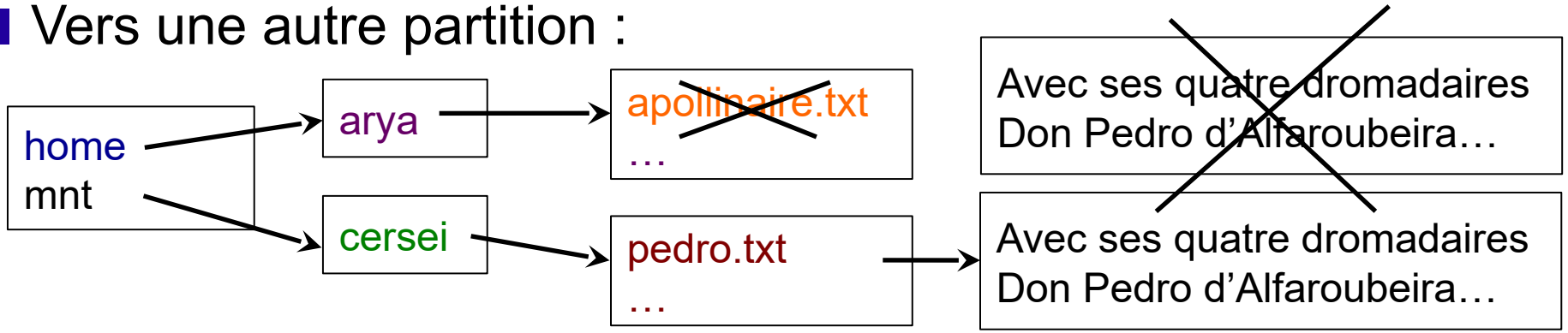
- `mv src dest` : *move* (déplace ou renomme)
  - *`src` : fichier de type quelconque*
  - Si `dest` est un répertoire, déplace `src` dans le répertoire `dest` (dans ce cas, multiples déplacements possibles avec `mv fic1 fic2 … rep`)
  - Sinon, déplace `src` sous le nom `dest`
    - Si `dest` est dans le même répertore : renommage
- Vers une autre partition :




```
mv /home/arya/apollinaire.txt /mnt/cersei/pedro.txt
```

Déplacement d'un fichier (7/7)

- `mv src dest` : *move* (déplace ou renomme)
  - *`src` : fichier de type quelconque*
  - Si `dest` est un répertoire, déplace `src` dans le répertoire `dest` (dans ce cas, multiples déplacements possibles avec `mv fic1 fic2 … rep`)
  - Sinon, déplace `src` sous le nom `dest`
    - Si `dest` est dans le même répertore : renommage
- Vers une autre partition :



`mv /home/arya/apollinaire.txt /mnt/cersei/pedro.txt`

- Le système de fichiers vu par un processus
- Le système de fichiers sur disque
- Les commandes utilisateurs
- Les droits d’accès

Droits d'accès

- Toute opération sur un fichier est soumise à droits d'accès
  - Message d'erreur « *Permission non accordée* »

- 3 types d'accès
  - `r` : droit de lecture
    - Si répertoire, consultation de ses entrées (c.-à-.d, `ls` autorisé)
    - Sinon, consultation du contenu du fichier
  - `w` : droit d'écriture
    - Si répertoire, droit de création, de renommage et de suppression d'une entrée dans le répertoire
    - Sinon, droit de modification du contenu du fichier
  - `x` :
    - Si répertoire, droit de traverser (c.-à-.d., `cd` autorisé)
    - Sinon, droit d'exécution

Droits d'accès

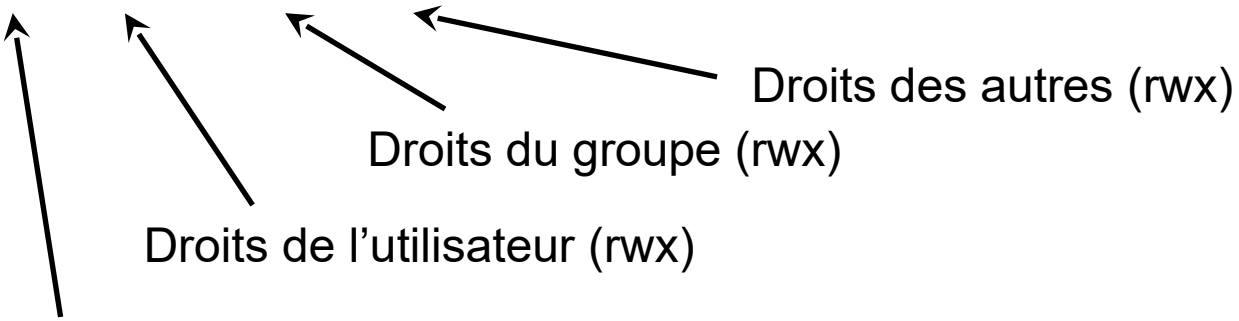
- 3 catégories d'utilisateurs :
  - Propriétaire (`u`)
  - Groupe propriétaire (`g`)
  - Tous les autres (`o`)

- Chaque catégorie possède ses types d'accès `r w x`

Droits d'accès – consultation

- `ls -ld` ⇒ donne les droits des fichiers

- Format de sortie de `ls -l`
  - `- --- --- ---`



Type du fichier :
d : répertoire
l : lien symbolique
- : fichier ordinaire

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

Droits d'accès – modification

- Modification sur un fichier existant

  `chmod droit fichier` : change mode

- Droits à **!appliquer!** au fichier
  - Catégories : `u`, `g`, `o` ou `a` (= `all` c.-à-.d., `ugo`)
  - Opérations : Ajout (+), retrait (–), affectation (=)

```
$
```

Droits d'accès – modification

- Modification sur un fichier existant

  `chmod droit fichier` : change mode

- Droits à **!appliquer!** au fichier
  - Catégories : `u`, `g`, `o` ou `a` (= `all` c.-à-.d., `ugo`)
  - Opérations : Ajout (+), retrait (–), affectation (=)

```
$ ls -ld fichier
 -rwx r-- --- fichier
$
```

Droits d'accès – modification

- Modification sur un fichier existant

  `chmod droit fichier` : change mode

- Droits à **!appliquer!** au fichier
  - Catégories : `u`, `g`, `o` ou `a` (= `all` c.-à-.d., `ugo`)
  - Opérations : Ajout (+), retrait (–), affectation (=)

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

Droits d'accès – modification

- Modification sur un fichier existant

  `chmod droit fichier` : change mode

- Droits à **!appliquer!** au fichier
  - Catégories : `u`, `g`, `o` ou `a` (= `all` c.-à-.d., `ugo`)
  - Opérations : Ajout (+), retrait (–), affectation (=)

```
$ ls -ld fichier
 -rwx r-- --- fichier
$ chmod u-x fichier
$ ls -ld fichier
-rw- r-- --- fichier
$ chmod u+x fichier
$ ls -ld fichier
-rwx r-- --- fichier
```

Démonstration

```
$ cp /etc/passwd .
$
```

Démonstration

```
$ cp /etc/passwd .
$ ls -l
total 4
-rw-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
$
```

Démonstration

```
$ cp /etc/passwd .
$ ls -l
total 4
-rw-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
$ chmod u-r passwd
$
```

Démonstration

```
$ cp /etc/passwd .
$ ls -l
total 4
-rw-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
$ chmod u-r passwd
$ cat passwd
cat: passwd: Permission non accordée
$
```

Démonstration

```
$ cp /etc/passwd .
$ ls -l
total 4
-rw-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
$ chmod u-r passwd
$ cat passwd
cat: passwd: Permission non accordée
$ mkdir rep
$
```

Démonstration

```
$ cp /etc/passwd .
$ ls -l
total 4
-rw-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
$ chmod u-r passwd
$ cat passwd
cat: passwd: Permission non accordée
$ mkdir rep
$ ls -l
total 8
--w-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
drwxr-xr-x 2 gthomas users   68 19 juil.  2016 rep
$
```

Démonstration

```
$ cp /etc/passwd .
$ ls -l
total 4
-rw-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
$ chmod u-r passwd
$ cat passwd
cat: passwd: Permission non accordée
$ mkdir rep
$ ls -l
total 8
--w-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
drwxr-xr-x 2 gthomas users   68 19 juil.  2016 rep
$ cd rep/
$
```

Démonstration

```
$ cp /etc/passwd .
$ ls -l
total 4
-rw-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
$ chmod u-r passwd
$ cat passwd
cat: passwd: Permission non accordée
$ mkdir rep
$ ls -l
total 8
--w-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
drwxr-xr-x 2 gthomas users   68 19 juil.  2016 rep
$ cd rep/
$ cd ..
$
```

Démonstration

```
$ cp /etc/passwd .
$ ls -l
total 4
-rw-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
$ chmod u-r passwd
$ cat passwd
cat: passwd: Permission non accordée
$ mkdir rep
$ ls -l
total 8
--w-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
drwxr-xr-x 2 gthomas users   68 19 juil.  2016 rep
$ cd rep/
$ cd ..
$ chmod u-x rep
$
```

Démonstration

```
$ cp /etc/passwd .
$ ls -l
total 4
-rw-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
$ chmod u-r passwd
$ cat passwd
cat: passwd: Permission non accordée
$ mkdir rep
$ ls -l
total 8
--w-r--r-- 1 gthomas users 1120 19 juil.  2016 passwd
drwxr-xr-x 2 gthomas users   68 19 juil.  2016 rep
$ cd rep/
$ cd ..
$ chmod u-x rep
$ cd rep
-bash: cd: rep: Permission non accordée
```

Droits d'accès initiaux

- Masque de droits d'accès **!retirés!** à la création de tout fichier
  - Commande `umask` (*user mask*)
  - Le masque est donné en octal (base 8) avec 3 chiffres (`u`, `g`, `o`)
  - En standard, masque par défaut = `022`
    - `r = 100` en binaire = 4 en octal, `w = 010 = 2`
    - Si droits retirés `--- -w- -w-`, alors droits appliqués `rw- r-- r--`
    - Le droit `x` est déjà retiré par défaut en général

  - Modification du masque grâce à la commande `umask`
    - Attention : `umask` sans effet rétroactif sur les fichiers préexistants
    - Attention : `umask` n'a d'effet que sur le `bash` courant

Démonstration

```
$ touch fichier_umask_defaut
$
```

Démonstration

```
$ touch fichier_umask_defaut
$ ls -lh
-rw-rw-r-- 1 amina amina 0 oct. 2 10:49 fichier_umask_defaut
$
```

Tous les fichiers sont créés avec des droits par défaut

Démonstration

```
$ touch fichier_umask_defaut
$ ls -lh
-rw-rw-r-- 1 amina amina 0 oct. 2 10:49 fichier_umask_defaut
$ mkdir repertoire_umask_defaut
$ ls -lh
-rw-rw-r-- 1 amina amina 0     oct. 2 10:49 fichier_umask_defaut
drwxrwxr-x 2 amina amina   4,0K  oct.      2 10:50 repertoire_umask_defaut
$
```

Et les répertoires aussi. Les droits des fichiers et des répertoires sont souvent différents

Démonstration

```
$ touch fichier_umask_defaut
$ ls -lh
-rw-rw-r-- 1 amina amina 0 oct. 2 10:49 fichier_umask_defaut
$ mkdir repertoire_umask_defaut
$ ls -lh
-rw-rw-r-- 1 amina amina 0     oct. 2 10:49 fichier_umask_defaut
drwxrwxr-x 2 amina amina 4,0K  oct. 2 10:50 repertoire_umask_defaut
$ umask 007
$
```

Ici, umask ne retire aucun droit au propriétaire et au groupe. Il retire tous les droits aux utilisateurs « other »

Démonstration

```
$ touch fichier_umask_defaut
$ ls -lh
-rw-rw-r-- 1 amina amina 0 oct. 2 10:49 fichier_umask_defaut
$ mkdir repertoire_umask_defaut
$ ls -lh
-rw-rw-r-- 1 amina amina 0     oct. 2 10:49 fichier_umask_defaut
drwxrwxr-x 2 amina amina   4,0K  oct.      2 10:50 repertoire_umask_defaut
$ umask 007
$ touch fichier_umask_nouveau
$ ls -lh
-rw-rw-r-- 1 amina amina 0     oct. 2 10:49 fichier_umask_defaut
-rw-rw---- 1 amina amina 0     oct. 2 10:52 fichier_umask_nouveau
drwxrwxr-x 2 amina amina 4,0K  oct. 2 10:50 repertoire_umask_defaut
$
```

A partir de là, tous les fichiers et répertoires créés n'ont plus les droits retirés par umask. Les droits des fichers existants ne changent pas

Démonstration

```
$ touch fichier_umask_defaut
$ ls -lh
-rw-rw-r-- 1 amina amina 0 oct. 2 10:49 fichier_umask_defaut
$ mkdir repertoire_umask_defaut
$ ls -lh
-rw-rw-r-- 1 amina amina 0     oct. 2 10:49 fichier_umask_defaut
drwxrwxr-x 2 amina amina   4,0K  oct.      2 10:50 repertoire_umask_defaut
$ umask 007
$ touch fichier_umask_nouveau
$ ls -lh
-rw-rw-r-- 1 amina amina 0     oct. 2 10:49 fichier_umask_defaut
-rw-rw---- 1 amina amina 0     oct. 2 10:52 fichier_umask_nouveau
drwxrwxr-x 2 amina amina 4,0K  oct. 2 10:50 repertoire_umask_defaut
$ mkdir repertoire_umask_nouveau
$ ls -lh
-rw-rw-r-- 1 amina amina 0     oct. 2 10:49 fichier_umask_defaut
-rw-rw---- 1 amina amina 0     oct. 2 10:52 fichier_umask_nouveau
drwxrwxr-x 2 amina amina 4,0K  oct. 2 10:50 repertoire_umask_defaut
drwxrwx--- 2 amina amina 4,0K  oct. 2 10:53 repertoire_umask_nouveau
```

Conclusion

- Concepts clés :
  - Arborescence, racine du système de fichier, répertoire de connexion, répertoire de travail
  - Chemin absolu, chemin relatif
  - Droits d'accès
  - Partition, inode
  - Fichier, répertoire, liens (direct et symbolique)

- Commandes clés :
  - `pwd`, `cd`, `ls`
  - `chmod`, `umask`
  - `mkdir`, `ln`, `rm`, `rmdir`, `cp`, `mv`

En route pour le TP !