

IFT1015 Programmation 1

Environnement de développement traditionnel
système de fichiers, shell, Node.js

Marc Feeley, avec ajouts de Pascal Vincent et Aaron Courville

Modifications: A. Tsikhanovich

Fichiers

- Lorsqu'on veut **préserver les données** pour un usage futur, on stocke les informations dans un **fichier**
- **Fichier** : groupe de données stockées sur un **support matériel persistant** (qui préserve le contenu même lorsqu'il est éteint)
 - Exemples : disque dur, disque compact (CD, CD-RW, DVD), clé USB, carte SD, ruban magnétique

Fichiers

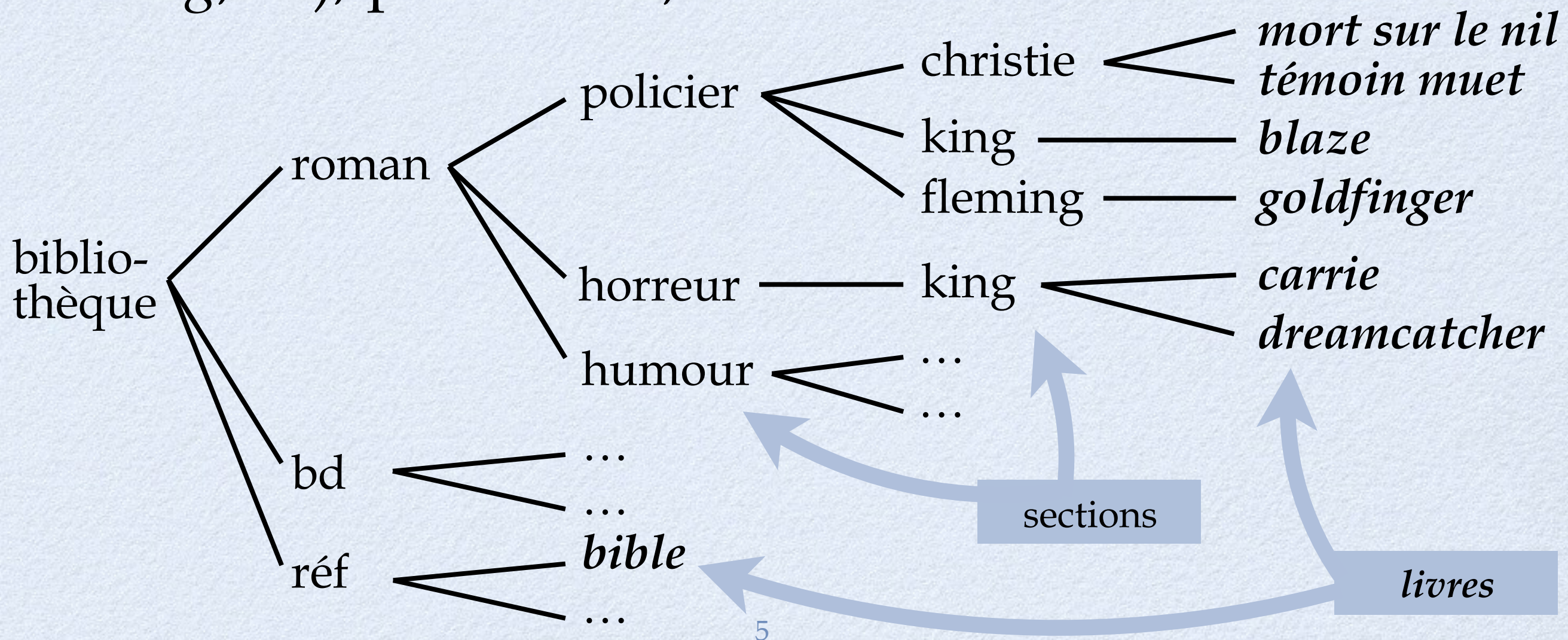
- On mesure la capacité de stockage des supports matériel en MB, GB et TB (méga/ giga/ tera octets)
 - 1 MB = 10^6 octets = 8 000 000 bits (~1/5 bible)
 - 1 GB = 10^9 octets = 8 000 000 000 bits (~200 bibles)
 - 1 TB = 10^{12} octets = 8 000 000 000 000 bits (~200 000 bibles)
- Vu la très grande capacité des supports matériel de stockage, on stocke normalement **plusieurs fichiers sur un même support**
- Il faut donc organiser les fichiers sur le support matériel... c'est le rôle du **système de fichier** ("file system") du système d'exploitation de l'ordinateur

Systeme de fichier

- Sur les systemes d'exploitation modernes, les fichiers sont organisés **hiérarchiquement** pour les regrouper logiquement
- Cette organisation facilite la **localisation** des fichiers en **regroupant les fichiers qui ont un lien commun**
- On peut faire une analogie entre un systeme de fichier qui organise des fichiers et une **bibliothèque** qui organise des livres

Systeme de fichier

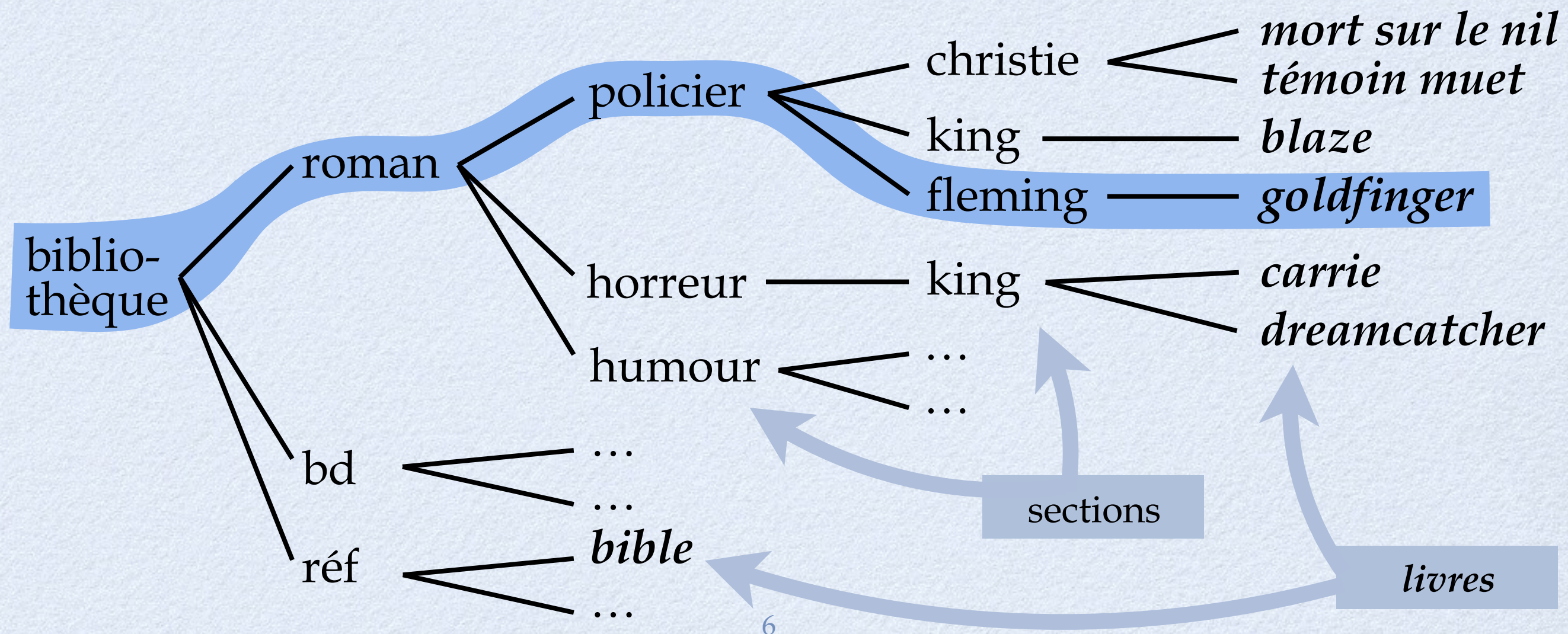
- Une bibliothèque est divisée en sections par le **genre** (*roman, bande dessinée, référence, ...*), par le **sous-genre** (*roman policier, roman humour, roman horreur, ...*), par l'**auteur** (*Agatha Christie, Ian Fleming, Stephen King, ...*), par le **titre**, etc



Systeme de fichier

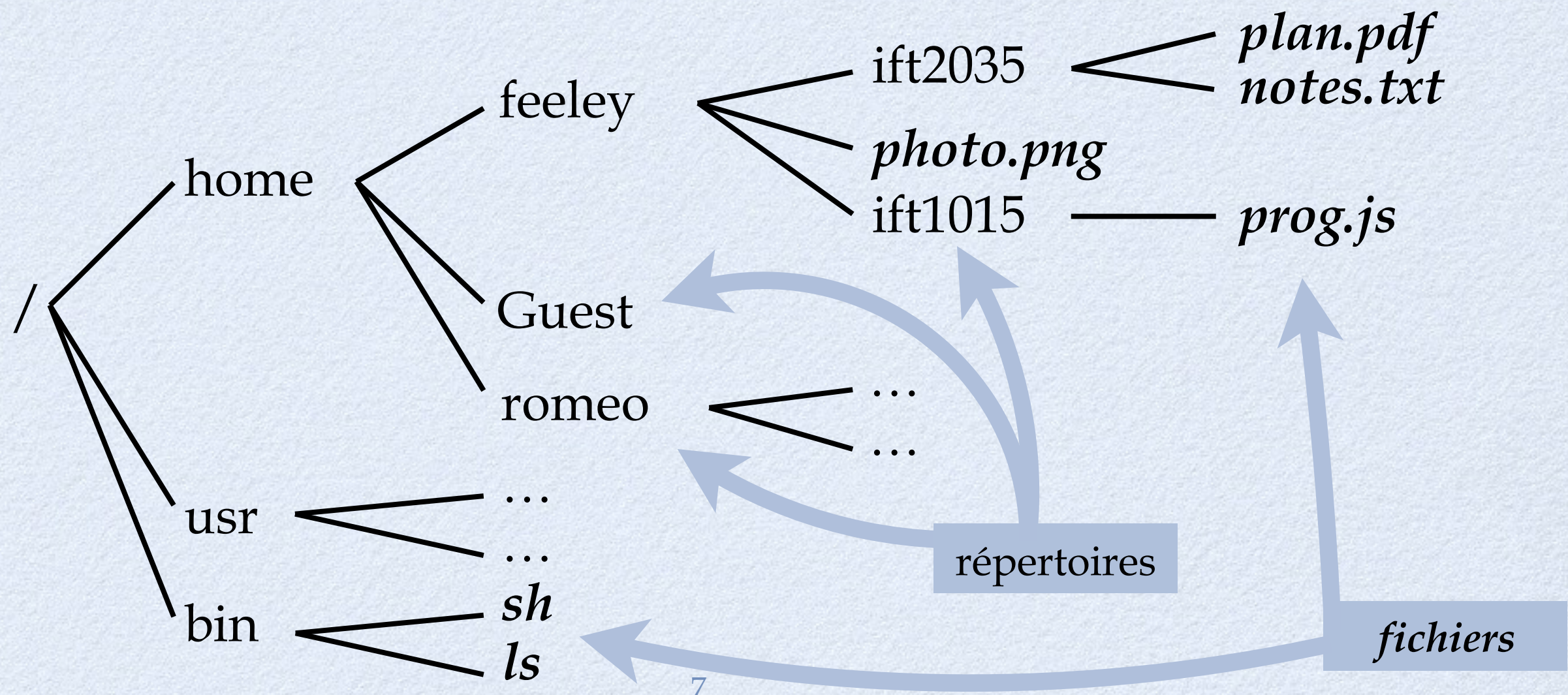
- Un livre spécifique est **identifié de façon unique** par le **chemin d'accès**, par exemple :

bibliothèque , roman , policier , fleming , goldfinger



Systeme de fichier

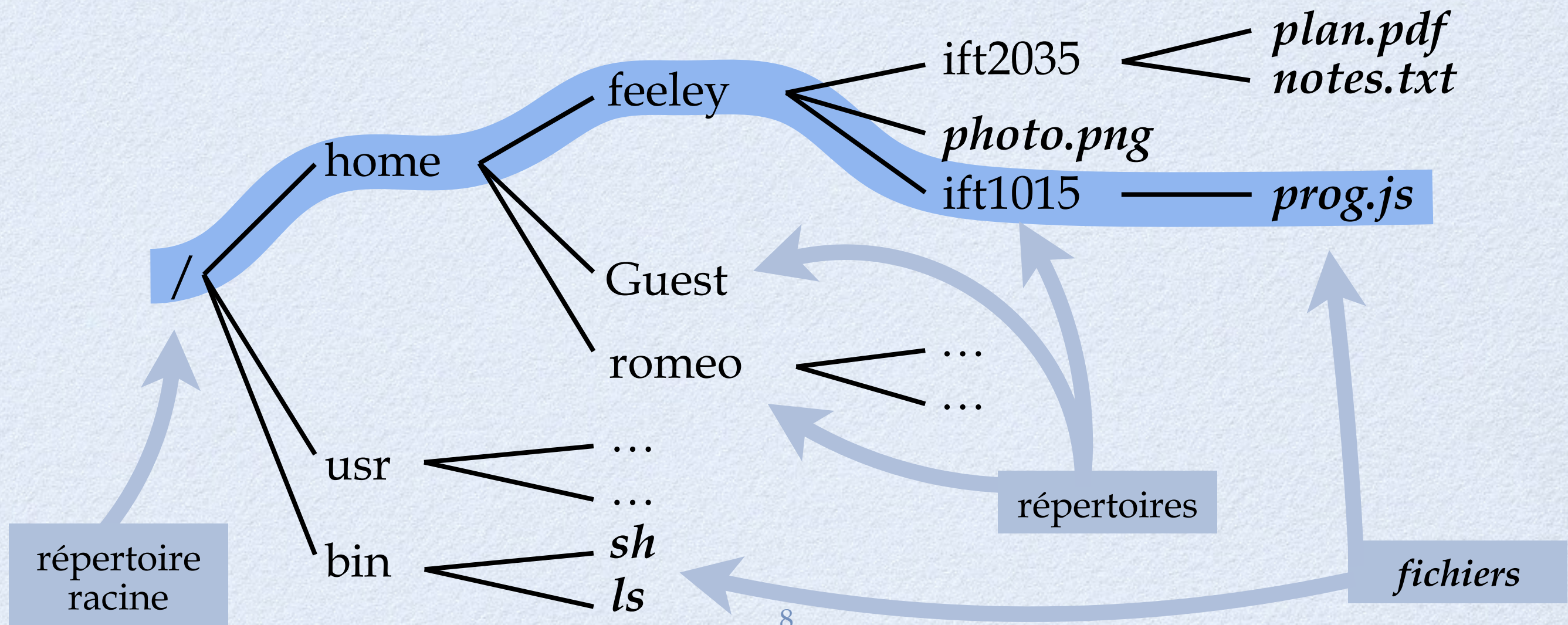
- Dans un système de fichier :
 - section = **répertoire** (“**directory**”) ou **dossier** (“**folder**”)
 - livre = **fichier** (“**file**”)
- Exemple Unix :



Systeme de fichier

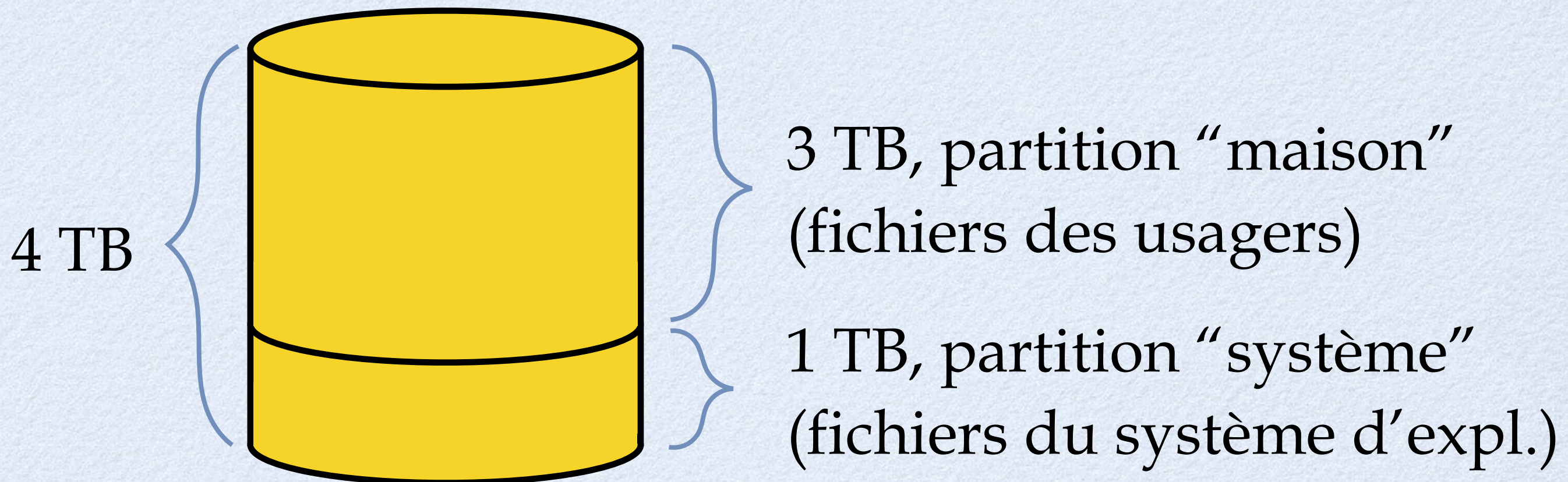
- Un fichier spécifique est **identifié** par le **chemin d'accès** ("path"), par exemple :

`/home/feeley/ift1015/prog.js` (" / " = séparateur de chemin)



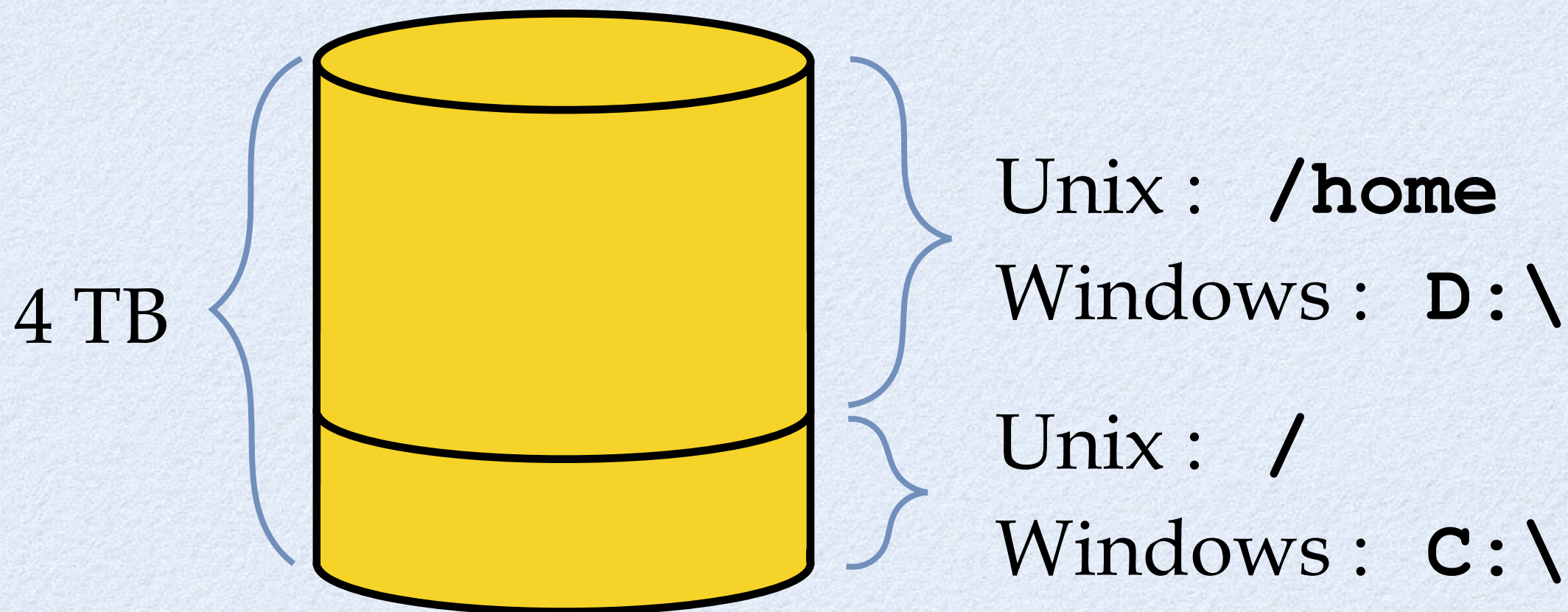
Systeme de fichier

- Pour simplifier la gestion de l'espace de stockage (sauvegarde, mise à niveau, permissions, etc) le disque dur est configuré en **partitions** chacune contenant des fichiers de même nature
- Exemple : un disque dur de 4 TB



Systeme de fichier

- Unix : les partitions se greffent à une branche du système de fichier
- Windows : les partitions ont un nom d'une lettre qui préfixe le chemin d'accès (et le séparateur de chemin d'accès est le "\\")



Systeme de fichier

- Les noms de fichiers et répertoires sont composés en général de n'importe quel caractère sauf “/”, mais il est mieux d'éviter les caractères spéciaux tel “\$”, “!”, “&”, “*” et l'espace (conflit possible avec le shell)
- Exemples :
 - Unix :
`/home/romeo/voyage-02.09.14/chat.jpg`
 - Windows :
`C:\Program Files\Internet Explorer\iexplorer.exe`

Système de fichier

- Par convention, on utilise à la fin du nom de fichier une **extension** (un point suivi de quelques lettres) pour indiquer la nature des données
- Extensions communes :

.exe	programme exécutable	.png	image
.js	code source JavaScript	.jpg	image
.java	code source Java	.gif	image
.c	code source C	.mp3	son
.txt	document textuel ASCII	.zip	fichier compressé
.doc	document Microsoft Word	.gz	fichier compressé
.ps	document Postscript		
.pdf	document Portable Document Format		

Gestionnaire ou navigateur de fichiers

- Les systèmes modernes ont des programmes graphiques pour naviguer dans le système de fichier



Windows Explorer



Mac Finder



File Manager
(Dolphin, sous KDE / linux)

- Navigateur de fichier \neq navigateur web

Shells

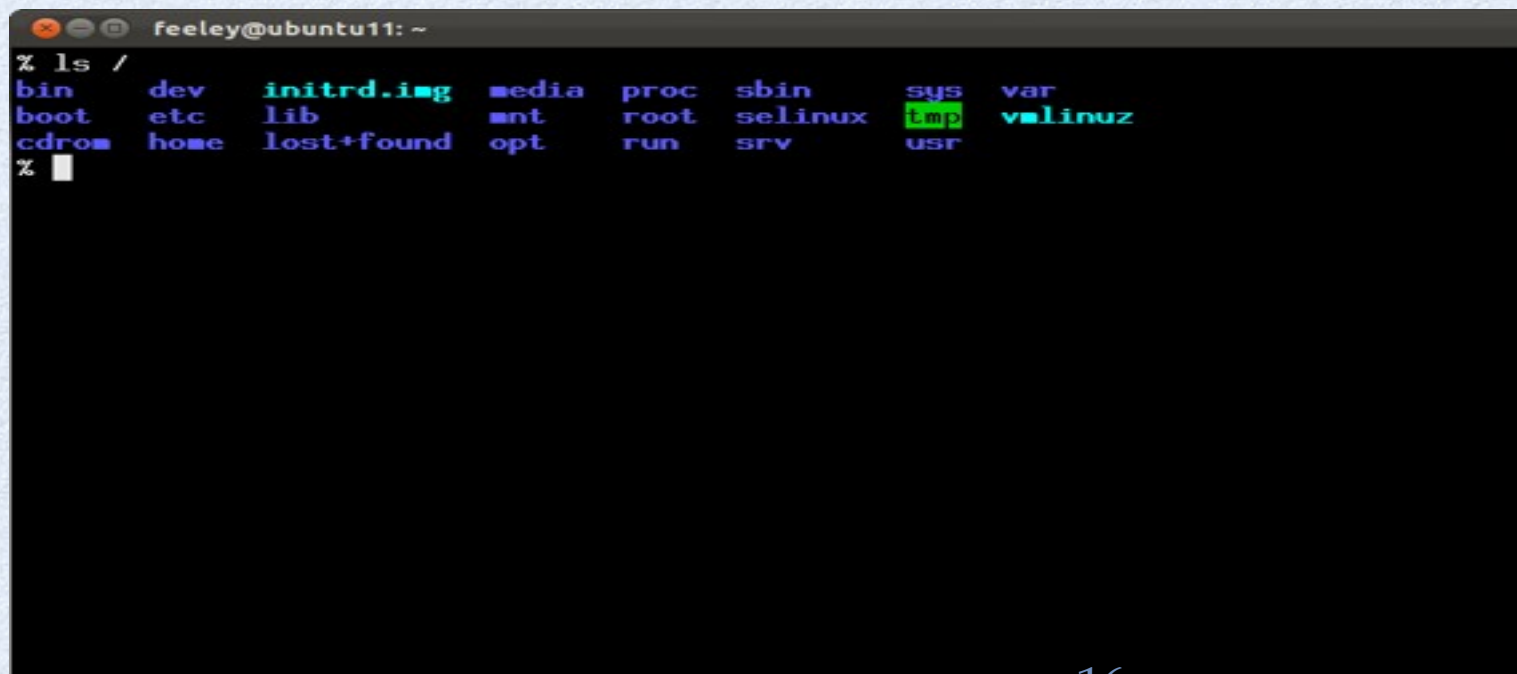
(ligne de commande)

Shell

- On se sert souvent d'un **shell** pour gérer le système de fichier (créer et éliminer des fichiers et répertoires, les lister, les déplacer, démarrer l'exécution des programmes, etc)
- Shell : **interprète interactif** permettant à l'utilisateur d'exécuter des commandes de gestion du système de fichier
- Sous Unix on a le choix de plusieurs shells : “**sh**”, “**bash**”, “**csh**”, etc (assez similaires en fonctions)
- Sous Windows on a le “**command prompt**”
- Mêmes principes mais syntaxes différentes

Shell

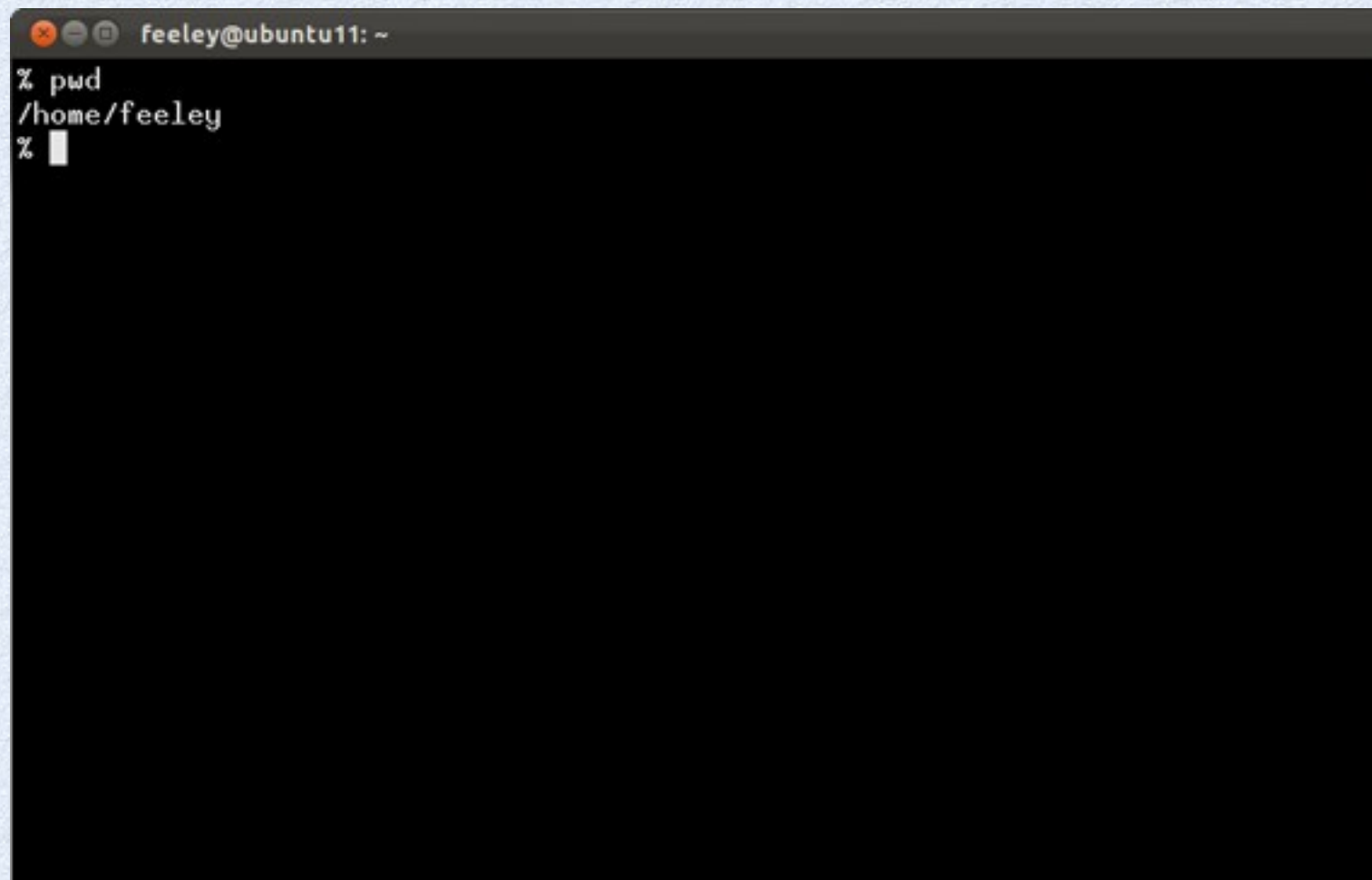
- Sous Unix on démarre un shell en ouvrant une **fenêtre de terminal** (application “Terminal”, “xterm”, “Konsole”, ...)
- Le shell affiche un **incitateur** (“prompt”) :
- Sous Windows on a le “**command prompt**”



```
feeley@ubuntu11: ~  
% ls /  
bin      dev      initrd.img  media  proc  sbin  sys  var  
boot     etc      lib         mnt    root  selinux tmp  valinux  
cdrom    home     lost+found  opt    run   srv   usr
```


Shell

- Le shell a un **répertoire de travail** (“**working directory**”) qui est positionné à un certain répertoire du système de fichier et la commande **pwd** affiche son chemin d'accès
- Au départ le répertoire de travail du shell est le **répertoire maison** (“**home directory**”) de l'utilisateur :

A terminal window titled 'feeley@ubuntu11: ~' with a dark background. The prompt is '%'. The user has entered 'pwd' and the output is '/home/feeley'. The prompt is now '% ' with a cursor.

```
% pwd
/home/feeley
% 
```


Shell

- Sans paramètre, la commande **ls** liste le contenu du répertoire de travail du shell, **dir** (**Windows**)
- Avec le paramètre **-l**, la commande **ls** donne des détails (taille des fichiers, date de dernière modification, type, permissions, ...)

```
feeley@ubuntu11: ~  
% pwd  
/home/feeley  
% ls /home/feeley  
Desktop ift1015 ift2035 photo.png  
% ls  
Desktop ift1015 ift2035 photo.png  
% ls -l  
total 2144  
drwxr-xr-x 2 feeley feeley 4096 2015-03-09 00:36 Desktop  
drwxrwxr-x 2 feeley feeley 4096 2015-03-09 00:13 ift1015  
drwxrwxr-x 2 feeley feeley 4096 2015-03-09 00:13 ift2035  
-rw-rw-r-- 1 feeley feeley 2182464 2015-03-09 00:18 photo.png  
% ls /home/feeley/ift1015  
prog.js  
% █
```


Shell

- Le répertoire de travail peut se changer avec la commande **cd** (“change directory”)
- Sans paramètre à la commande **cd**, le répertoire de travail du shell revient au répertoire maison de l'utilisateur

```
feeley@ubuntu11: ~  
% pwd  
/home/feeley  
% ls /home/feeley  
Desktop ift1015 ift2035 photo.png  
% ls  
Desktop ift1015 ift2035 photo.png  
% ls -l  
total 2144  
drwxr-xr-x 2 feeley feeley 4096 2015-03-09 00:47 Desktop  
drwxrwxr-x 2 feeley feeley 4096 2015-03-09 00:13 ift1015  
drwxrwxr-x 2 feeley feeley 4096 2015-03-09 00:13 ift2035  
-rw-rw-r-- 1 feeley feeley 2182464 2015-03-09 00:18 photo.png  
% ls /home/feeley/ift1015  
prog.js  
% cd /home/feeley/ift1015  
% pwd  
/home/feeley/ift1015  
% ls  
prog.js  
% cd  
% pwd  
/home/feeley  
% █
```


Chemins absolus et relatifs

- Lorsque le chemin d'accès débute par “/” c'est un **chemin absolu**
 - Exemple : `/home/feeley/ift1015/prog.js`
- Sinon, c'est un **chemin relatif** qui désigne un **chemin à partir du répertoire de travail**
 - Exemple : `photo.png`
 - Exemple : `ift1015/prog.js`

Répertoire parent

- Dans un chemin d'accès, on peut se servir du nom “..” (deux points) pour désigner le **répertoire parent**
- C'est utile pour naviguer le système de fichier avec la commande **cd** :
 - **cd ift1015**
 - **ls**
 - **cd ..** (revenir au répertoire précédent)

Commandes importantes, Linux

- **pwd** afficher le répertoire de travail
- **cd** *path* changer le répertoire de travail
- **ls** *path* lister le contenu du répertoire *path*
- **rm** *path* éliminer le fichier *path*
- **cp** *path1 path2* copier le fichier *path1* à *path2*
- **mv** *path1 path2* déplacer le fichier *path1* à *path2*
- **mkdir** *path* créer le répertoire vide *path*
- **rmdir** *path* éliminer le répertoire vide *path*

Editeurs de texte pour programmeur

Editeurs de texte

- Pour la création et la modification des fichiers textuels, on se sert souvent d'un **éditeur de texte** spécialisé pour l'édition de fichiers de code source.
- Il en existe un très grand nombre. Ex et suggestions:
 - Traditionnels Unix/Linux: **emacs** ou **vim** (une version d'emacs mieux intégrée à l'environnement mac, et facile à installer: **aquamacs**)
 - Plus modernes sur Linux: **gedit** ou **Kate**
 - Populaire sur Windows: **Notepad++**
 - Populaire sur Mac: **TextMate**
 - Plus modernes: **SublimeText**, **Atom**

Interpréteur Javascript node.js

- On peut installer un interpréteur Javascript permettant d'exécuter des programmes Javascript en ligne de commande (avec un shell) plutôt que dans un navigateur.
- Un des plus populaires (notamment pour écrire des serveurs web) est **Node.js** facilement installable sur votre ordinateur.
- Pour exécuter un programme dont le code source se trouverait dans un fichier `prog.js`
 - **`nodejs prog.js`**
- Remarque: dans les installations Mac et Windows la commande se nomme **node** plutôt que **nodejs**.

Scénario type

- Approche 1 pour développer un programme avec l'interprète JavaScript "**node.js**" :

- Dans l'éditeur créer **prog.js**
- Sauver **prog.js**

Sous **Linux**: **nodejs prog.js**

Sous Windows ou Mac: **node prog.js**

exécuter le programme
avec l'interprète **node.js**

note : dans certains
environnements la
commande se nomme **node**