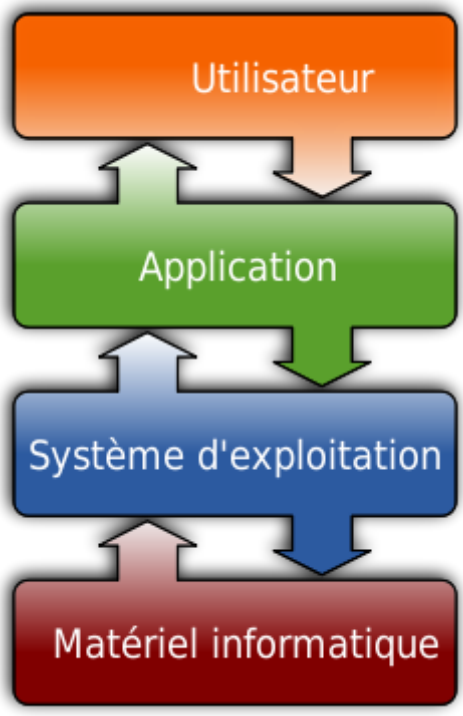
Système d'exploitation

Contenus	Capacités attendues	Commentaires
Systèmes d'exploitation	Identifier les fonctions d'un système d'exploitation. Utiliser les commandes de base en ligne de commande. Gérer les droits et permissions	Les différences entre systèmes d'exploitation libres et propriétaires sont évoquées. Les élèves utilisent un système d'exploitation libre.

Version pdf du cours

Définition d'un système d'exploitation

Un *système d'exploitation* est un logiciel, ou ensemble de programmes, qui sert d'interface entre les programmes exécutés par l'utilisateur et les ressources matérielles d'un ordinateur.



> Source : Wikipedia

Un *système d'exploitation* est à la fois :

- une *machine virtuelle* qui présente une interface simplifiée d'accès aux ressources (processeur, mémoire, périphériques
  d'entrée/sortie, réseau …) pour les autres programmes et pour l'utilisateur
- un *chef d'orchestre* et un *administrateur* :
  - c'est le premier programme exécuté au démarrage de l'ordinateur
  - il gère l'accès concurrent aux ressources par les différents programmes (ordonnancement de l'utilisation du processeur par les programmes en cours d'exécution ou processus, sécurisation de la mémoire) ou utilisateurs (droits d'accès du système de fichiers).

Attributs du système d'exploitation :

Le mode noyau

Les programmes ne peuvent pas accéder directement aux ressources matérielles (mémoire, processeur, périphériques d'entrée/sortie) sinon il y aurait des conflits : par exemple deux programmes pourraient écrire dans la même zone mémoire.

Le système d'exploitation possède un mode d'exécution privilégiée : le *mode noyau*, qui lui donne un accès unique et total aux ressources. Les autres programmes s'exécutent en *mode normal*. Pour accéder aux ressources, ils font appel au système d'exploitation à l'aide de primitives qu'on nomme *appels systèmes*.

Le contrôle de l'activité des programmes

Un programme ne peut être exécuté que si le système d'exploitation l'autorise. L'exécution d'un programme peut être interrompue par le système d'exploitation, pour libérer une ressource ou en cas d'erreur. On parle d'*interruption système*.

Un programme en cours d'exécution s'appelle un *processus* et le système d'exploitation arbitre le partage de temps de calcul du processeur par les processus concurrents. Il joue un rôle *d'ordonnanceur* pour éviter par exemple qu'un processus s'accapare le processeur.

Le contrôle du système de fichiers

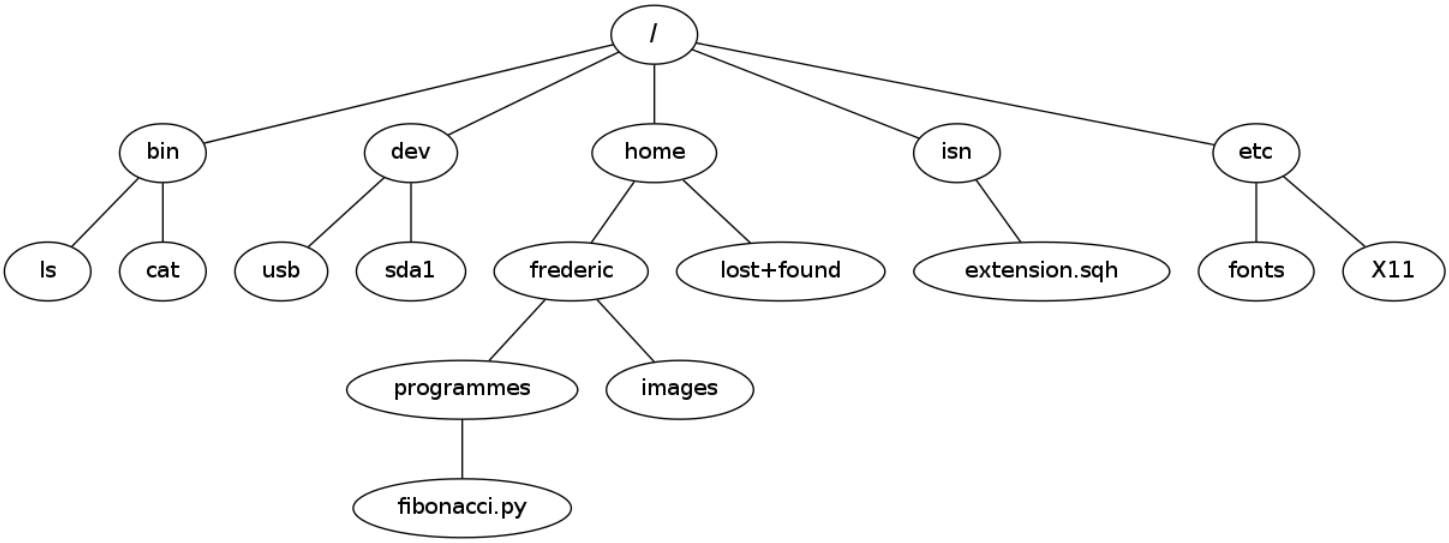
Système de fichiers

Sur les supports de mémoire persistants (disques durs, clefs USB…), les informations sont regroupées par le système d'exploitation dans des *fichiers* qui sont organisés à travers un système de fichiers.

Dans les systèmes d'exploitation qui respectent le standard POSIX, on distingue plusieurs catégories de fichiers :

- *fichiers réguliers* ou *fichiers textes* qui contiennent des suites de caractères et qui sont lisibles par des humains
- *fichiers binaires* qui sont des suites d'octets non lisibles par des humains
- *fichiers exécutables* qui sont des programmes, ils peuvent être des fichiers textes ou binaires.
- *répertoires* qui sont des listes de fichiers : ils servent de conteneur à fichiers.

Les *répertoires* pouvant contenir d'autres fichiers, un système de fichiers POSIX possède une structure hiérarchique qui est une *arborescence* avec un *répertoire racine* symbolisé par un /.



Les fichiers sont repérés par leur *chemin* :

- *chemin absolu* depuis la racine : par exemple si un fichier `exemple.txt` se trouve dans le répertoire `sandbox`, contenu dans le répertoire `maurice`, lui-même contenu dans le répertoire `home`, lui-même dans le répertoire racine, son *chemin absolu* est `/home/maurice/sandbox/exemple.txt`
- ou *chemin relatif* qui est relatif au *répertoire courant* ou *répertoire de travail* où l'utilisateur se trouve. Par exemple, si le répertoire courant est `maurice`, le *chemin relatif* du fichier précédent est `sandbox/exemple.txt`.

"Exemple"

Dans le schéma d'arborscence précédent :

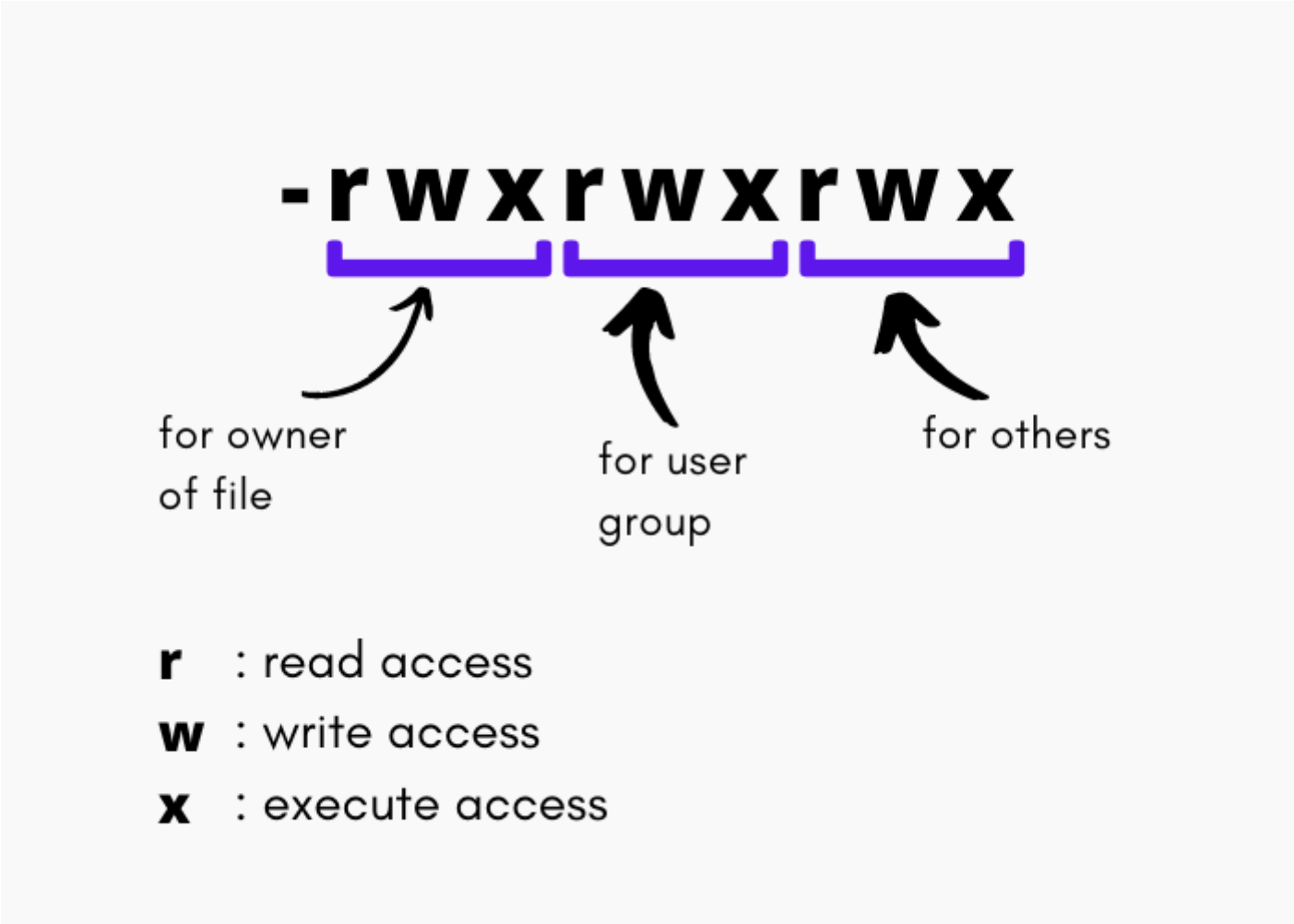
- le *chemin absolu* du fichier `fibonacci.py` est `/home/frederic/programmes/fibonacci.py`
- si on se trouve dans le répertoire `frederic` le *chemin relatif* du fichier `fibonacci.py` est `programmes/fibonacci.py`
- si on se trouve dans le répertoire `images` le *chemin relatif* du fichier `fibonacci.py` est `../programmes/fibonacci.py` . Le motif `..` désigne le répertoire parent du répertoire courant et marque une remontée d'un niveau dans l'arborescence.

Gestion des droits et permissions

Les systèmes d'exploitation modernes sont multi-utilisateurs et permettent de gérer les *droits et permissions* sur les fichiers (dont les répertoires) pour les différents *utilisateurs*.

Pour un fichier, on distingue :

- trois profils d'utilisateurs :
  - le *propriétaire* (ou *owner*) noté `u`
  - le *groupe principal* (ou *group*) noté `g`
  - un *autre utilisateur* (ou *other*) noté `o`
- trois types de permissions :
  - *lecture* (caractère *r* si attribué ou - sinon)
  - *écriture* (caractère *w* si attribué ou - sinon)
  - *exécution* (caractère *x* si attribué ou - sinon)



Source : https://s3.ap-south-1.amazonaws.com/s3.studytonight.com/tutorials/uploads/pictures/1593780021-1.png

Pour les répertoires, la signification des permissions est précisée dans ce tableau. Attention, si on a le droit d'écriture sur un répertoire on peut supprimer tous les fichiers qu'il contient même ceux dont on n'est pas propriétaire.

	Type de permission	pour un répertoire
r	lecture	lister le contenu
w	écriture	ajouter, supprimer, renommer des fichiers
x	exécution	entrer dedans

"Exemple"

```
fjunier@fjunier:~$ ls -l test.py
-rwxrw-r-- 1 fjunier fjunier 3680 juin   2  2023 test.py
```

Les permissions pour le fichier `text.py` ci-dessus sont positionnées ainsi :

Profil	Droit de lecture `r`	Droit d'écriture `w`	Droit d'exécution `x`
Propriétaire	Oui	Oui	Oui
Groupe principal du propriétaire	Oui	Oui	Non
Autres	Oui	Non	Non

Interface utilisateur d'un système d'exploitation

Le `shell`

Une interface entre l'utilisateur et le système d'exploitation s'appelle un shell ou *interpréteur de commandes*.

Le rôle d'un *shell* est de prendre une entrée de l'utilisateur, de la traduire en instructions compréhensibles par le système d'exploitation et de renvoyer la réponse du système à l'utilisateur.

Il existe deux grandes catégories de *shell* :

- les *interfaces textuelles* comme bash, le plus commun sur les systèmes de la famille UNIX.
- les *interfaces graphiques* qu'on retrouve dans les systèmes d'exploitation grand public tels que Windows

Nous allons donner des exemples de commandes textuelles en bash.

Syntaxe d'une commande `bash`

Une commande *shell* est constituée du nom de la commande suivi d'un ou plusieurs arguments. Des options précédées d'un tiret haut, peuvent modifier le comportement de la commande :

```
nom_commande -option1 -option2 ... arg1 arg2 arg3 ...
```

Ainsi, la commande `ls` permet d'afficher des informations sur répertoire ou un fichier :

- Sans argument, ni option `ls` liste le contenu du répertoire courant :

```
junier@fredportable:~/sandbox$ ls
fichier1  fichier2  fichier3  fichier4  rep1  rep2
```

- Avec l'option `-l` elle affiche des informations détaillées sur chacun des fichiers contenus dans le répertoire :

```
junier@fredportable:~/sandbox$ ls -l
total 8
-rw-rw-r-- 1 junier junier    0 août  16 21:43 fichier1
...........
drwxrwxr-x 2 junier junier 4096 août  16 21:44 rep1
drwxrwxr-x 2 junier junier 4096 août  16 21:44 rep2
```

- L'option `-a` affiche les fichiers (ou répertoires) cachés et l'option `-h` convertit les tailles de fichiers (en octets par défaut) en des multiples plus lisibles. On peut écrire `ls -l -a -h` ou regrouper les options `ls -lah` . L'ordre des options n'a pas d'importance :

```
junier@fredportable:~/sandbox$ ls -lah
total 16K
drwxrwxr-x  4 junier junier 4,0K août  16 21:49 .
drwxr-xr-x 50 junier junier 4,0K août  16 21:43 ..
-rw-rw-r--  1 junier junier    0 août  16 21:49 .cache_cache
-rw-rw-r--  1 junier junier    0 août  16 21:43 fichier1
.............
drwxrwxr-x  2 junier junier 4,0K août  16 21:44 rep1
drwxrwxr-x  2 junier junier 4,0K août  16 22:10 rep2
```

Principales commandes de l'interpréteur `bash`

Dans ce tableau `chemin` , `source` et `cible` désignent des chemins absolus ou relatifs de fichiers ou de répertoires dans le système de fichiers. Il faut bien comprendre que chaque commande est exécutée depuis un certain emplacement dans l'arborescence du système de fichiers qui est le *répertoire courant*.

Commande	Syntaxe	Fonction

Commande	Syntaxe	Fonction
`man`	`man commande`	Affiche l'entrée du manuel de documentation de `bash` pour la commande
`pwd`	`pwd`	Affiche le chemin absolu du répertoire courant
`cd`	`cd chemin`	change de répertoire courant pour celui désigne par `chemin`
`cd`	`cd ..`	change de répertoire courant pour le répertoire parent dans l'arboresence
`cd`	`cd ~`	change de répertoire courant pour le répertoire personnel de l'utilisateur `/home/user`
`ls`	`ls`	affiche les répertoires et fichiers du répertoire courant
`ls`	`ls -l`	affiche de façon détaillée les répertoires et fichiers du répertoire courant
`mv`	`mv source cible`	Déplace le fichier de chemin `source` vers le chemin `cible`, peut être utilisé pour renommer un fichier
`mkdir`	`mkdir rep`	Crée le sous-répertoire `rep` vide dans le répertoire courant
`rm`	`rm chemin`	Supprime le fichier désigné par son `chemin`
`rm`	`rm -r chemin`	Supprime le répertoire désigné par son `chemin`, et récursivement tous les fichiers et sous-répertoires qu'in contient, l'option `-r` signifie récursif

Commande	Syntaxe	Fonction
`cp`	`cp source cible`	Copie le fichier de chemin `source` vers le chemin `cible`, peut être utilisé pour renommer un fichier
`ps`	`ps -e`	Affiche tous les processus
`kill`	`kill -SIGKILL pid`	Force l'arrêt du processus identifié par son `pid`
`ping`	`ping adresse_ip_cible`	Envoie un paquet de test avec le protocole ICMP vers l'adresse ip cible
`host`	`host nom_domaine`	Résout le nom de domaine et renvoie l'adresse IP donnée par le service DNS
`traceroute`	`traceroute (nom_domaine/adresse_ip)`	Affiche les routeurs traversés pour atteindre le nom de domaine ou l'adresse IP cible

Les différents types de licences logicielles

On distingue :

- Les **licences propriétaires** :
  Un éditeur concède à titre non exclusif un droit d'usage sur un logiciel dont il conserve les droits de propriété intellectuelle. Un **CLUF** (Contrat Licence Utilisateur Final) délimite les conditions d'usage du logiciel : limitation du nombre de copies, de postes sur lequel le logiciel peut être installé, interdiction de modification
  et de redistribution . En général, le code source n'est pas accessible (on parle de logiciel fermé) ...
- Les **licences libres** ou **open-source** : elles respectent les quatre libertés fondamentales du logiciel libre :
  i. *Liberté 1* **liberté d'exécuter le programme, pour tous les usages.**
  ii. *Liberté 2* **liberté d'étudier le fonctionnement du programme, et de le modifier pour l'adapter à ses besoins**

iii. *Liberté 3* **liberté de redistribuer des copies.**

iv. *Liberté 4* **liberté de redistribuer aux autres des copies de versions modifiées**

Les libertés 2 et 4 nécessitent l'accès au code source. Il ne faut pas confondre licence libre et domaine public : mettre un logiciel sous licence libre ne signifie pas abandonner ses droits d'auteurs (droit moral inaliénable) mais donner des permissions aux usagers qui vont au-delà de ce qu'autorise le droit d'auteur par défaut.

"Exemple"

Système d'exploitation	Licence libre	Descendant de	Figure tutélaire
Noyau Linux	Oui, GNU version 2	Unix, distributions Ubuntu, Debian, Fedora ...	Linus Torvalds
Android	Oui, Apache	Unix, noyau Linux	Andy Rubin
MacOS	Partiellement, licence propriétaire et Apple Public Licence	Unix	Steve Jobs (CEO pas développeur)
Windows	Non	MS-DOS	Bill Gates

"Attention"

Ne pas confondre logiciels libres et gratuits (double sens du mot free en Anglais):

- un **freeware** désigne usuellement un logiciel distribué gratuitement, indépendamment de sa licence d'utilisation. Le code source n'est pas forcément fourni ; dans ce cas ce n'est pas un logiciel libre.
- un **shareware** désigne un logiciel distribué gratuitement et librement pendant une durée ou un nombre d'utilisations qui sont fixées par l'auteur. Au delà de ce délai il faut payer des royalties et le code source n'est pas fourni donc ce n'est pas un logiciel libre.