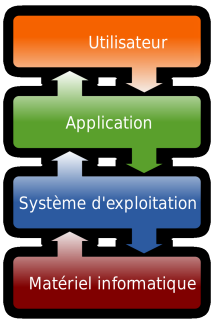
1 Fonctions d'un système d'exploitation

Pour qu'un ordinateur soit capable de faire fonctionner une application, la machine doit être en mesure d'effectuer un certain nombre d'opérations préparatoires afin d'assurer les échanges entre le processeur, la mémoire, et les ressources physiques (périphériques).
Le système d'exploitation (noté SE ou OS pour Operating System) est donc chargé d'assurer la liaison entre les ressources matérielles, l'utilisateur et les applications.



Le système d'exploitation permet ainsi d'offrir à l'utilisateur une interface homme-machine (notée «IHM») simplifiée afin de lui permettre de s'affranchir de la complexité de la machine physique.

1.1 Rôles du système d'exploitation

- **Gestion du processeur** : le système d'exploitation est chargé de gérer l'allocation du processeur entre les différents programmes.
- **Gestion de la mémoire** : le système d'exploitation est chargé de gérer l'espace mémoire alloué à chaque application et, le cas échéant, à chaque usager.
- **Gestion des entrées/sorties** : le système d'exploitation permet d'unifier et de contrôler l'accès des programmes aux ressources matérielles par l'intermédiaire des pilotes.
- **Gestion de l'exécution des applications** : le système d'exploitation est chargé de la bonne exécution des applications en leur affectant les ressources nécessaires à leur bon fonctionnement.
- **Gestion des droits** : le système d'exploitation est chargé de la sécurité liée à l'exécution des programmes en garantissant que les ressources ne sont utilisées que par les programmes et utilisateurs possédant les droits adéquats.
- **Gestion des fichiers** : le système d'exploitation gère la lecture et l'écriture dans le système de fichiers et les droits d'accès aux fichiers par les utilisateurs et les applications.

1.2 Composantes du système d'exploitation

Le système d'exploitation est composé en particulier :

- d'un **noyau** (kernel) représentant les fonctions fondamentales du système d'exploitation telles que la gestion de la mémoire, des processus, des fichiers, des entrées-sorties principales, et des fonctionnalités de communication.
- d'un **interpréteur de commande** (shell = coquille, par opposition au noyau) permettant la communication avec le système d'exploitation par l'intermédiaire d'un langage de commandes.
- d'un **système de fichiers**, permettant d'enregistrer les fichiers dans une arborescence.

1.3 Principaux systèmes d'exploitation du marché

- **Windows** : SE propriétaire préinstallé sur la très grande majorité des ordinateurs personnels. (Microsoft)
  Aujourd'hui : Windows 10.
- dérivés d'**UNIX** (très nombreux !) :
  - **macOS** : SE partiellement propriétaire des ordinateurs Apple.
  - **GNU/Linux** : SE libre s'appuyant sur le noyau Linux et les outils GNU. (début en 1991 avec Linus Torvalds)
    Il existe de nombreuses distributions (exemple : Ubuntu).
  - famille BSD (B de Berkeley).
  - **iOS** : SE propriétaire des smartphones iphone (Apple).
  - **Android** : SE de nombreux smartphones, avec un noyau Linux (Google).

Remarque : Cette liste est très loin d'être exhaustive...

On retiendra que de nombreux OS dérivent du système UNIX créé en 1969 (en langage C).

Différence système libre / système propriétaire :
Un système **libre** rend son **code source public**, avec **autorisation** de **modification** et de **redistribution**. Au contraire, un système propriétaire ne diffuse pas son code source.

Attention de ne pas confondre libre et gratuit. Ces deux termes se disent « free » en Anglais d'où une certaine confusion parfois. Un OS (ou un logiciel) peut être libre ou propriétaire et indifféremment gratuit ou commercialisé.

2 Commandes de base en ligne de commande

Même si les interfaces graphiques (IHM) permettent une interaction avec le système d'exploitation très pratique pour le grand public, il peut être intéressant et utile (pour des raisons d'efficacité ou d'actions spécifiques) d'interagir directement avec l'interpréteur de commande (shell) de l'OS.

Il existe différents langages shell, mais tous partagent des commandes basiques communes.

La première commande à connaître est : **man** *commande*.
Cela affiche les pages du **man**uel d'aide à propos de la *commande* souhaitée, avec sa syntaxe et toutes ses options (souvent nombreuses).

Le tableau suivant liste quelques commandes essentielles liées à la gestion des fichiers et dossiers (ou répertoires, *directoy* in English) :

commande	action	option / commentaire
pwd	indique le chemin absolu du répertoire courant	*present work directory*
ls	liste les fichiers et dossiers	*listing* L'option **-l** affiche plus de détails
mkdir dossier	crée un dossier (dir)	*make directory*
cd dossier	change de répertoire	*change directory* définition d'un chemin absolu ou relatif dans l'arborescence
touch fichier	crée un fichier	
cat fichier	affiche le contenu d'un fichier texte	commandes voisines : more ou less
nano fichier	édite le contenu d'un fichier texte	nano est un programme parmi d'autres permettant l'édition de fichiers (vim, emacs...)
mv source destination	déplace un fichier, ou renomme un fichier	*move*
cp fichier destination	copie un fichier	*copy*
rm fichier	supprime un fichier	*remove* Supprime aussi un répertoire vide. Pour supprimer un répertoire et son contenu, il faut ajouter l'option **-r**. L'option **-i** demande une confirmation avant suppression (prudent !)

On retiendra aussi 3 abréviations fonctionnelles des dossiers :
~ (tilde espagnol) : désigne le répertoire personnel (home) de l'utilisateur.
. (un point unique) : désigne le répertoire courant.
.. (deux points) : désigne le répertoire parent du répertoire courant (chemin relatif).

Enfin, le caractère « joker » (glob) ***** (étoile) permet de remplacer n'importe quelle chaîne de caractères dans un nom de dossier ou de fichier.

Un exemple :

```
cp ./photo*.jpg ../images/
```

Cette ligne de commande copie tous les fichiers du répertoire courant dont le nom commence par `photo` et finit par `.jpg` dans le sous-dossier `images` du répertoire parent.
Remarque : on aurait pu omettre le `./` désignant le répertoire courant (utilisé par défaut si on ne le précise pas).

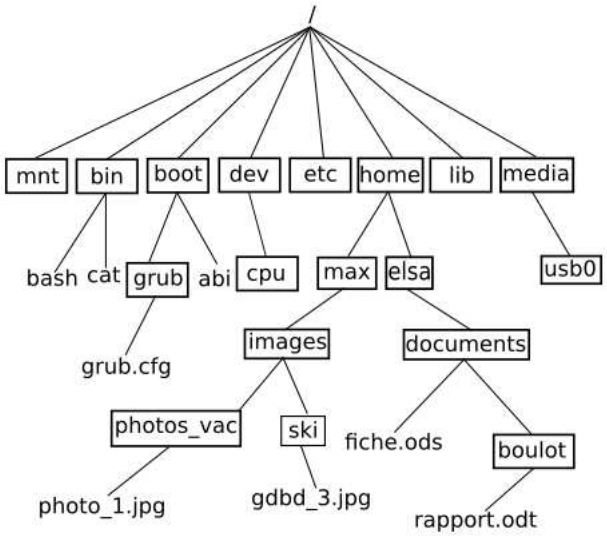
La puissance de la ligne de commande peut venir du **chaînage** possible entre les commandes avec le symbole | (pipe). Son utilisation permet d'envoyer la sortie d'une commande en entrée de la commande suivante.

Il est possible de **rediriger** la sortie d'une commande vers un fichier texte par exemple (plutôt que dans la console) avec l'utilisation d'un chevron simple > ou double >>.

3 Droits et permissions d'accès aux fichiers

Les systèmes d'exploitation permettent aujourd'hui un fonctionnement multi-utilisateurs des ressources. Le système d'exploitation prend alors en charge la gestion des droits d'accès aux ressources et en particulier aux fichiers et dossiers.

Le système de fichier d'un système GNU/Linux présente l'arborescence suivante :



Le répertoire `/` est le répertoire racine. Dans ce répertoire racine, on trouve un répertoire `home` qui contient les espaces personnels des utilisateurs (max et elsa) enregistrés sur la machine.

Chaque utilisateur possède des droits de lecture (**r**ead r), écriture (**w**rite w) ou d'exécution (e**x**ecute x) sur les fichiers de la machine. Ces droits peuvent être définis pour chaque fichier (ou dossier).

Par ailleurs, il est possible de définir des **groupes** d'utilisateurs qui partagent des droits communs. La gestion des droits peut ainsi en être faciliter : on peut attribuer un droit spécifique à tous les utilisateurs d'un groupe en une seule opération.

La commande `ls -l` liste les répertoires et fichiers en précisant les droits sous le modèle de l'exemple suivant :

```
toto@machine: ls -l
d rwx r-x r-x toto enfant dossier1
- rwx rw- r-x toto enfant fichier1
```

- Le 1er caractère indique s'il s'agit d'un répertoire (*directory* d) ou d'un fichier simple.
- Les 3 caractères suivants indiquent les droits de l'utilisateur propriétaire du fichier (droits **u**ser u). *Rappel* : r pour read, w pour write et x pour execute.
- Les 3 caractères suivants indiquent les droits des membres du groupe auquel appartient l'utilisateur (droits **g**roup g).
- Enfin, les 3 caractères suivants indiquent les droits des autres utilisateurs (droits **o**ther o).
- Apparaît ensuite le propriétaire du fichier, puis son groupe, et le nom du fichier (plus encore d'autres infos).

Il existe aussi un super-utilisateur qui a tous les droits sur tous les fichiers. Il a aussi la possibilité de changer les permissions sur tous les fichiers. En revanche, un utilisateur quelconque ne peut gérer la gestion des droits que sur ses propres fichiers.
Remarque : sur une distribution Ubuntu, chaque utilisateur peut endosser le rôle du super-utilisateur en connaissant le mot de passe (avec la "pré-commande" sudo : super user do).

Les droits peuvent être modifiés avec la commande **chmod**.

— **Méthode 1** : `chmod [ugoa] [+-] [rwx] fichier`
Exemple : `chmod a+x fichier` : ajoute le droit d'exécution à tous les utilisateurs (*all* a).

— **Méthode 2** : `chmod [ABC] fichier`
Exemple : `chmod 764 fichier` : accorde le droit de code 7 à user (rwx), le droit de code 6 à group (rw-), et le droit de code 4 à other (r- -).
Les codes fonctionnent ainsi : r=4, w=2, x=1 (codage binaire à 3 bits).

1	- - x	001
2	- w -	010
3	- w x	011
4	r - -	100
5	r - x	101
6	r w -	110
7	r w x	111