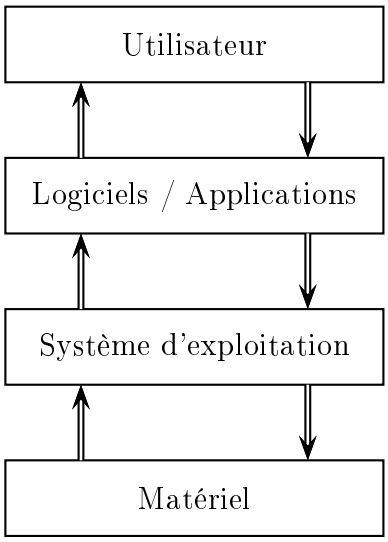
Chapitre 1

Systèmes d'exploitation

1.1 Systèmes d'exploitation

En informatique, un **système d'exploitation** (en anglais OS pour Operating System) est un ensemble de programmes qui dirige l'utilisation des ressources d'un ordinateur par des logiciels applicatifs. Il reçoit des demandes d'utilisation des ressources de l'ordinateur – stockage des mémoires (accès à la mémoire vive, aux disques durs) ; calcul du processeur central ; communication vers des périphériques ; via le réseau – de la part des logiciels applicatifs.
Il s'agit du deuxième programme exécuté après le programme d'amorçage (en anglais bootloader) lors de la mise en marche de l'ordinateur. Il offre une suite de services généraux facilitant la création de logiciels applicatifs et sert d'intermédiaire entre ces logiciels et le matériel informatique, évitant les interférences entre ceux-ci.



Il existe sur le marché des dizaines de systèmes d'exploitation différents, très souvent livrés avec l'appareil informatique. C'est le cas de Windows, Mac OS, GNU/Linux, (pour lequel il existe de nombreuses distributions), Android ou iOS. Les fonctionnalités offertes diffèrent d'un système à l'autre et sont typiquement en rapport avec l'exécution des programmes, l'utilisation de la mémoire centrale ou des périphériques, la manipulation des systèmes de fichiers, la communication, ou la détection et la gestion d'erreurs.

Actuellement, les deux familles de systèmes d'exploitation les plus populaires sont UNIX (dont Mac OS, GNU/Linux, iOS et Android) et Windows. Ce dernier détient un quasi-monopole sur les ordinateurs personnels avec près de 90 % de part de marché mais GNU/Linux et Android sont massivement utilisés pour les serveurs Web et les objets connectés du quotidien (télévisions, enceintes, voitures, imprimantes, etc). Il existe enfin des systèmes d'exploitation spécialement conçus pour certaines tâches comme les OS « temps réel » qui sont notamment utilisés dans l'industrie (pilotage de robots, pilotes automatiques, etc) où le temps de réaction de l'ordinateur doit être le plus court possible.

On peut enfin distinguer deux types de systèmes d'exploitation :

Les propriétaires : comme Windows ou Mac OS dont les codes sources ne sont pas accessibles et qui ne sont pas partageables et modifiables ;

Les libres ou open source : comme Linux avec les célèbres distributions Debian et Ubuntu ou Android dont les codes sources sont accessibles et qui sont partageables et modifiables ; ce qui engendre des myriades de variantes de ces systèmes d'exploitation.

Les systèmes d'exploitation libre peuvent être aussi bien développés et maintenus par des particuliers ou des communautés (comme pour Debian) que par des entreprises (comme pour Ubuntu et Android).

1.2 Un peu d'histoire

Les premiers ordinateurs n'avaient pas à proprement parler de système d'exploitation, et cette situation dura de 1949 jusqu'en 1956. À l'époque, les langages utilisés étaient très proches de la machine physique et chaque utilisateur avait la machine pour lui seul pendant une tranche de temps déterminée. Le pionnier fut CTSS (Compatible Time-Sharing System) qui en 1961 pouvait servir 4 utilisateurs connectés à un ordinateur via des terminaux.

En 1963-64, le MIT (Massachusetts Institute of Technology) entreprit un projet de grande ampleur : le projet MAC (Man And Computer) qui devait donner naissance à Multics (Multiplexed Information and Computing Service). La réalisation de ce projet se montra plus difficile que prévu. Le système commença à fonctionner en 1969 mais ses performances étaient très loin des objectifs fixés. Multics a utilisé et perfectionné tous les concepts scientifiques de base des sytèmes d'exploitation. Il a mis en évidence l'importance de l'organisation structurée et du partage de l'information, via le **système de gestion de fichiers** (SGF). Le modèle établi par Multics est toujours à la base des systèmes actuels.

Des ingénieurs ayant travaillé sur ce projet, Ken Thompson et Dennis Ritchie, décidèrent de réaliser un système minimal. Ils commencèrent, en 1969, le développement d'un système conversationnel mono-utilisateur qu'ils baptisèrent Unics. Devenu multi-utilisateur, le système prit le nom d'Unix. La première utilisation commerciale eut lieu en 1972. Vendu assez cher aux entreprises, il fut diffusé quasi gratuitement aux universités, créant une communauté très active qui contribua à l'évolution du système.

En 1991, Linus Torvalds entreprit le développement d'un système d'exploitation qui deviendra le noyau Linux. Il choisit rapidement de publier ce noyau sous licence GNU GPL.

La première version des systèmes d'exploitation Apple est née en 1984 : System 1. Contrairement à ses prédécesseurs, Mac OS fait partie de la famille des systèmes d'exploitation Unix. La première version du système est Mac OS X Server 1.0, commercialisée en 1999, suivie par une version orientée pour le grand public en mars 2001.

En 1981, Microsoft développe initialement le système d'exploitation DOS pour IBM. MS-DOS est longtemps resté la base des systèmes d'exploitation grand public de Microsoft. Avec Windows 95, l'utilisateur accède directement à une interface graphique et MS-DOS devient invisible à l'utilisateur. Ce système d'exploitation n'est aujourd'hui plus maintenu.

1.3 Fonctionnalités et constitution

1.3.1 Fonctionnalités

Le principal rôle du système d'exploitation est d'articuler les différentes architectures informatiques et d'organiser l'utilisation des ressources de façon rationnelle. Les services proposés et la manière de s'en servir diffèrent d'un système d'exploitation à l'autre mais on retrouve toujours les fonctionnalités suivantes :

Utilisation des périphériques : chaque périphérique a ses propres instructions, avec lesquelles il peut être manipulé ; le système d'exploitation en tient compte. Il permet au programmeur de manipuler le périphérique par de simples demandes de lecture ou d'écriture, lui évitant la perte de temps de traduire les opérations en instructions propres au périphérique.

Accès aux fichiers : en plus des instructions propres à chaque périphérique (ROM, RAM, disque dur, lecteur de CD- ROM ou DVD-ROM, SSD, clé USB), le système d'exploitation tient compte du format propre de chaque support servant au stockage des données. Il offre également des mécanismes de protection permettant de contrôler quel utilisateur peut manipuler quel fichier.

Accès aux ressources : une des fonctions du système d'exploitation est de protéger les ressources contre l'utilisation par des personnes non autorisées et de résoudre les conflits lorsque deux utilisateurs demandent simultanément la même ressource.

Détection et récupération en cas d'erreur : lorsqu'une erreur survient, qu'elle soit matérielle ou logicielle, le système d'exploitation traite l'erreur en adoucissant son impact sur le système informatique. Il peut tenter de réitérer l'opération, arrêter l'exécution du programme fautif, ou signaler le problème à l'utilisateur.

Contrôle : un système d'exploitation peut tenir des statistiques d'utilisation des ressources, surveiller la performance, et les temps de réponse.

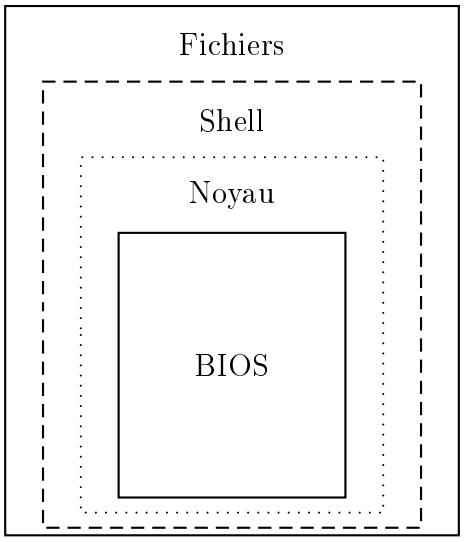
1.3.2 Constitution

Le système d'exploitation se compose de trois éléments fondamentaux :

Le noyau (kernel) : représente les fonctions fondamentales du système informatique (gestion des périphériques, de la mémoire, etc).

Un interpréteur de commande (shell) : (la « coquille » en opposition au noyau) permet d'assurer la communication avec l'OS par le biais d'un langage de commande pour permettre à l'utilisateur d'utiliser les ressources matérielles du PC sans connaître leurs caractéristiques.

Un système de fichiers : permettant d'enregistrer les données dans une arborescence.



Le BIOS (Basic Input Ouput System) ne fait pas parti du système d'exploitation. Il est utilisé par ce dernier lors du démarrage de l'ordinateur afin d'accéder au matériel et opérations de base. Il peut aussi servir à changer le système d'exploitation de l'ordinateur.

1.4 Fichiers et processus

L'utilisateur de base d'un ordinateur est généralement conscient de l'existence de deux types d'entités sur son équipement : des processus et des fichiers.

1.4.1 Processus

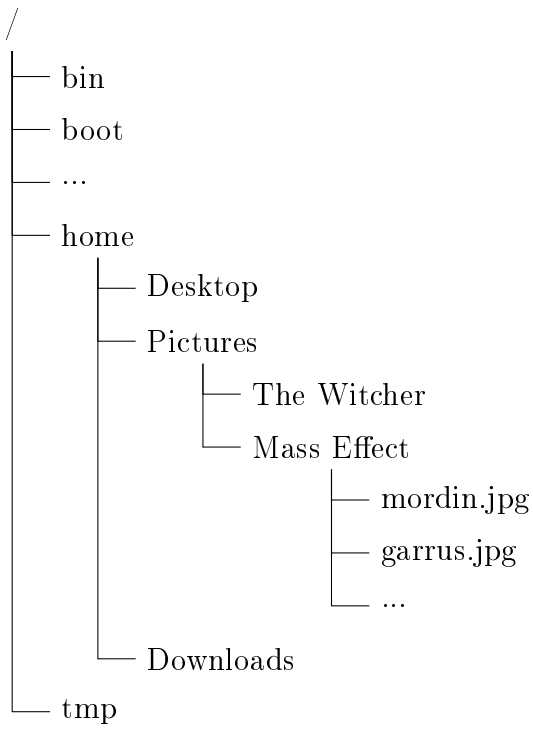
Un **processus** est l'objet dynamique associé à un programme : le programme est une suite d'instructions à exécuter associées à des données, le processus est une instance en mémoire de ce programme en train de s'exécuter. L'ensemble de la mémoire associée à un processus est appelé son contexte.

Les systèmes de type Unix sont multi-tâches : on a l'impression que plusieurs processus peuvent s'exécuter en même temps. Un processeur ne peut pourtant exécuter qu'un seul processus à la fois, qu'on appelle processus actif. L'ordonnanceur (programme du système d'exploitation) change régulièrement de processus actif en procédant à un changement de contexte. Ainsi le nouveau processus actif ne se rend pas compte qu'il a été interrompu et continue là où il en était.

1.4.2 Fichiers

Pour l'utilisateur, un fichier est un concept logique. C'est un objet qui renferme des informations et sur lequel on peut effectuer diverses opérations telles que lire, écrire, modifier, supprimer.
Dans la réalité, un fichier est un ensemble de zones d'enregistrement, pas toujours contiguës, situées sur un support d'archivage. Lire un tel fichier entraîne un grand nombre d'opérations complexes : rechercher les adresses des données sur le support d'archivage, lire les différentes zones, gérer le dialogue d'entrée-sortie avec le disque via le bus de connexion, etc. Le système d'exploitation masque tous ces traitements et permet à l'utilisateur d'en faire abstraction.

Dans un système de type Unix, les données sont rangées dans des fichiers, eux-mêmes organisés de façon hiérarchique, grâce à des répertoires.
Cette hiérarchie peut être représentée par un arbre au sens informatique du terme, dont les nœuds internes seraient les répertoires et les feuilles les fichiers réguliers (et les répertoires vides). Le vocabulaire utilisé est le vocabulaire classique quand on étudie les arbres : le répertoire qui se trouve en haut de cette hiérarchie s'appelle le répertoire racine et l'unique répertoire contenant un autre répertoire s'appelle répertoire parent désigné par « .. ». Par convention, le répertoire parent de la racine est la racine en elle-même. Le caractère « / » désigne la racine du SGF puis il permet de séparer les noms de deux répertoires lorsqu'on décrit un chemin dans l'arborescence du SGF. Le caractère « ~ » désigne le répertoire où se trouve l'utilisateur au moment où il se connecte à son compte.



Quelques répertoires classiques :

bin : contient les commandes de base du système ;

dev : contient les fichiers représentant les dispositifs matériel (devices) du système ;

etc : contient les fichiers de configuration du système ;

home : répertoire d'accueil des utilisateurs ;

lib : contient les librairies.

Il existe deux façons pour désigner un fichier :

Chemin absolu : on décrit sans ambiguïté possible comment arriver jusqu'au fichier depuis la racine.

Chemin relatif : on décrit comment arriver jusqu'au fichier en partant du répertoire courant.

Exemples : Dans les exemples suivants, le « / » permet de séparer le nom des répertoires ou des fichiers et le « \ » permet d'indiquer un espace dans le nom du répertoire.. Depuis un nœud de l'arborescence, on peut accéder qu'aux nœuds ou feuilles parents et enfants.

— `cd home/Pictures/Mass\ Effect/garrus.jpg` permet d'accéder au fichier garrus.jpg depuis la racine.

— `cd ../Mass\ Effect/garrus.jpg` permet d'accéder au fichier garrus.jpg depuis le répertoire « The Witcher ». On remonte d'abord dans l'arborescence avec le `..` avant de redescendre avec le `Mass\ Effect/garrus.jpg`.

1.4.3 Commandes du terminal sous Linux

Lorsqu'on accède au terminal (ou shell), sous Linux, on peut se déplacer dans l'arborescence des répertoires et des fichiers, les modifier, etc, grâce aux commandes suivantes :

pwd	afficher le répertoire courant [Print Working Directory].
cd *chemin*	changer de répertoire courant.
ls	lister le contenu du répertoire courant.
ls -l	lister le contenu du répertoire courant avec des informations complémentaires, notamment les droits.
mkdir *dossier*	créer un répertoire *dossier* dans le répertoire courant.
cp *source dest*	copier une *source* vers une *destination*.
mv *source dest*	déplacer une *source* vers une *destination*.
rm *fichier*	supprimer un *fichier*.
rmdir *repertoire*	supprimer un *répertoire* vide.
rm -r *repertoire*	supprimer un *répertoire* et son contenu récursivement.
chmod *mode fichier*	modifier les droits d'accès (*mode*) à un *fichier*.
cat *fichier*	afficher le contenu du *fichier*.
echo $*variable*	afficher le contenu de la *variable*.
read *variable*	créer une variable *variable* et y enregistrer un contenu.

Remarque : ces commandes possèdent de nombreuses options accessibles dans les documentations. Par exemple, dans le `rm -r`, le `-r` représente l'option de suppression récursive du contenu du répertoire.

Exemples :

- `read test` crée une variable nommée `test` et permet d'y enregistrer le contenu : « ceci est un test ».
- `echo $test` affiche le contenu de la variable test, i.e. : ceci est un test.
- `echo $test > test.txt` crée un fichier `test.txt` et y enregistre le contenu de la variable `test`.
- `echo "un autre test" > test.txt` écrase le contenu du fichier `test.txt` et y enregistre le nouveau contenu (sans utiliser de variable).
- `echo "encore un test" >> test.txt` ajoute le contenu au fichier `test.txt` sans en supprimer l'ancien.
- `cat test.txt` affiche le contenu du fichier :
  un autre test
  encore un test
- `rm test.txt` supprime le fichier.

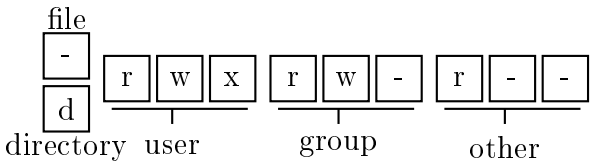
Le langage du terminal se nomme Bash et comme les autres langages informatiques, il est possible de l'utiliser pour coder les structures élémentaires que sont les tests et les boucles. Il est aussi possible d'enregistrer des programmes en Bash dans des fichiers au format .sh puis de les exécuter ensuite.

1.4.4 Droits

Sous Unix, chaque fichier (et répertoire) est la propriété d'un utilisateur particulier ; par défaut l'utilisateur qui a crée le fichier. Les utilisateurs sont réunis en groupes. Les droits sur les fichiers sont alors définis en fonction de la "position" du demandeur par rapport au propriétaire du fichier : propriétaire, faisant partie du groupe propriétaire, autre utilisateur.

Il y a trois types de droits dont le signification est résumée dans le tableau ci-dessous :

			fichier régulier	répertoire
lecture	r	4	regarder le contenu	lister le contenu
écriture	w	2	modifier le contenu	ajouter ou supprimer un élément
exécution	x	1	exécuter	passer à travers



Suivant les situations, on peut utiliser les triplets rwx ou la valeur numérique associée.

Exemples :

- `chmod g+x fichier1.py` ajoute le droit d'exécution aux utilisateurs du groupe.
- `chmod 764 fichier2.py` donne les droits 764 = rwx rw r peu importe les droits précédents.

Attention ! Il faut être très prudent avec les droits utilisateurs, on peut vite arriver à des failles de sécurité ou rendre son ordinateur inutilisable.

1.5 Exercices

Un simulateur de terminal Linux est disponible ici. Il servira à effectuer les exercices suivants.

Exercice 1.1. [Découverte du Bash]

1. Une fois le simulateur lancé, lister les dossiers et fichiers présents du répertoire courant. Dans quel répertoire êtes-vous ?
2. Créer les répertoires `Test1` et `Test2`.
3. Supprimer le répertoire `Test2`.
4. Entrer dans le répertoire `Test1` et créer deux nouveaux répertoires `Test2` et `Test3` à l'intérieur. Sortir de `Test1`.
5. Entrer dans le répertoire `Test2` à partir d'une répertoire courant en une seule commande.
6. À partir du répertoire `Test2`, aller dans le `Test3` à l'aide d'un chemin relatif.
7. Créer la variable `var` contenant un mot de votre choix. En afficher le contenu.
8. Enregistrer son contenu dans un fichier `fichier1.txt`.
9. Écraser le contenu de ce fichier en le remplaçant par un autre texte de votre choix. Vérifier l'opération.
10. Sans en écraser le contenu, y ajouter un autre texte. Vérifier l'opération.
11. Supprimer le fichier. Vérifier.
12. Créer un fichier vide `fichier2.txt`.
13. Revenir dans le répertoire parent puis supprimer les répertoires crées et leurs contenus en une seule commande. Vérifier.

Exercice 1.2. [Le droit c'est moi !]

1. Quels sont les droits accordés au fichier `fichier` avec les commandes suivantes ?

   (a) `chmod 777 fichier` (c) `chmod 507 fichier` (e) `chmod 321 fichier`
   (b) `chmod 144 fichier` (d) `chmod 666 fichier` (f) `chmod 740 fichier`

2. Quelles commandes faut-il utiliser si l'on souhaite accorder les droits suivants au triplet utilisateur, groupe, autres ?

   (a) `rwx --x --x` (b) `rw- r-- -w-` (c) `--x --- ---`

Exercice 1.3. [« I am the Law ! »]

1. Ouvrir un terminal.
2. Se rendre dans le dossier Documents et y créer un nouveau dossier Test.
3. Créer un fichier `text.txt` contenant un mot de votre choix. Vérifier.
4. Changer ses droits pour --x --- ---. Que font ces droits ? Vérifier.
5. Changer à nouveau ses droits pour r-x r-- r--. Que font-ils ? Vérifier.
6. Changer encore une fois ses droits pour rwx r-- r--. Que font-ils ? Vérifier.
7. Supprimer le fichier.
8. Sortir du dossier et le supprimer.