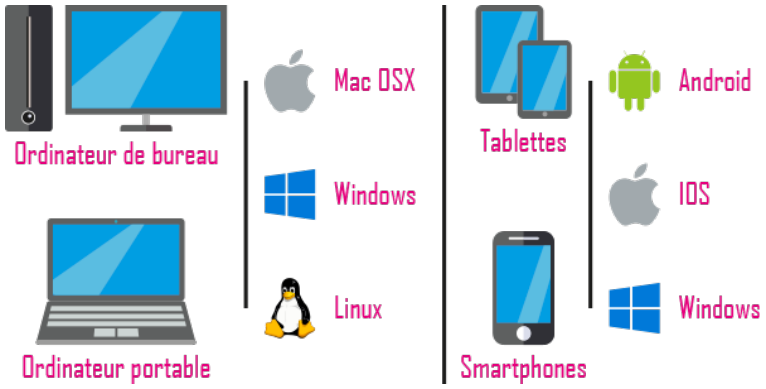
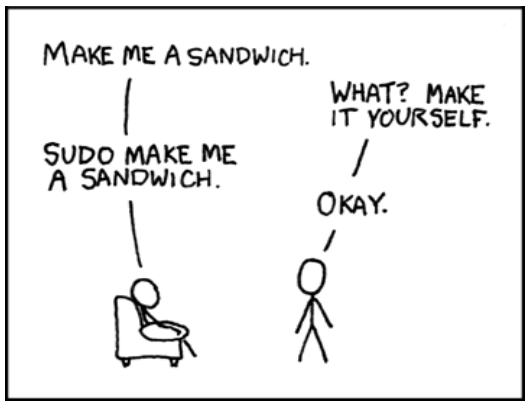


“



”

SYSTEMES D’EXPLOITATION

- *Commandes de base*
- *Droits et permissions d’accès aux fichiers*

Numérique et Sciences Informatiques
1ère

Support de cours :
Jean-Christophe BONNEFOY

Objectifs :

- Identifier les fonctions d’un système d’exploitation.
- Utiliser les commandes de base en ligne de commande.
- Gérer les droits et permissions d’accès aux fichiers.

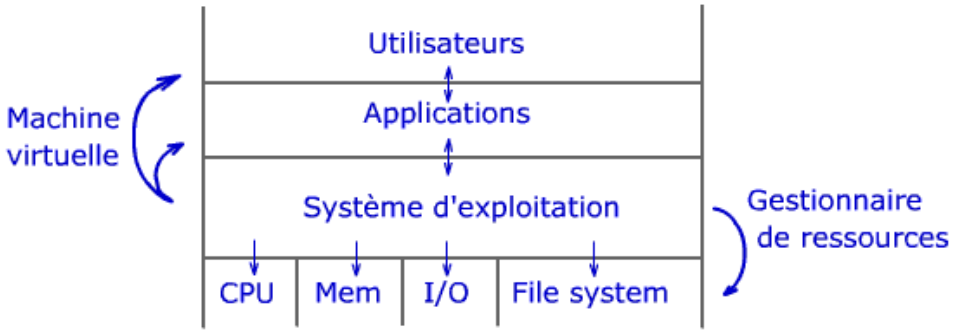
1. Qu'est-ce-qu'un système d'exploitation (SE ou OS : Operating System)

Définition : le système d'exploitation est un ensemble de programmes responsables de la liaison entre les ressources matérielles d'un ordinateur (le hardware) et les applications de l'utilisateur (le software).

> Ainsi lorsqu'un programme désire accéder à une ressource matérielle, il ne lui est pas nécessaire d'envoyer des informations spécifiques au périphérique, il lui suffit d'envoyer les informations au système d'exploitation, qui se charge de les transmettre au périphérique concerné via son pilote. En l'absence de pilotes il faudrait que chaque programme reconnaisse et prenne en compte la communication avec chaque type de périphérique !

Le système d'exploitation constitue une **machine virtuelle** qui, pour les applications, substitue des composants logiciels aux composants matériels. Il est la plateforme pour laquelle sont construites les applications.

Pour ce qui est du développement des applications : l'OS propose une interface de programmation appelée API *(Application Program Interface)*, une sorte de boîte à outils à laquelle les développeurs recourent pour construire leurs applications. Ces API procurent une vue uniforme et simplifiée des ressources de la machine. Cela permet aux applications de faire abstraction des particularités du matériel en dissimulant la diversité et la complexité du hardware.



Comme on peut le voir sur la figure ci-dessus, les 4 principaux services rendus par l'OS sont :

- La gestion du processeur (allocation du processeur entre les différents programmes)
- La gestion de la mémoire (partagée entre plusieurs applications)
- La gestion des entrées / sorties (clavier, imprimantes, …)
- La gestion des fichiers (arborescence logique)

On peut aussi en ajouter 2 tout aussi important :

- La gestion des applications (processus en cours d'exécution)
- La gestion des droits (sécurité liée à l'exécution des programmes)

Il existe divers systèmes d'exploitation qui peuvent se découper en deux familles de systèmes :

- Les systèmes propriétaires : Windows, MacOS, UNIX, …
- Les systèmes libres : Linux, Android, …

La différence essentielle est que le code d'un logiciel libre (et donc d'un système libre) est public. On peut en général le modifier ou s'en servir pour fabriquer de nouveaux produits. Les logiciels propriétaires sont en général non ouverts, il est donc plus difficile (voire illégal) de les modifier.

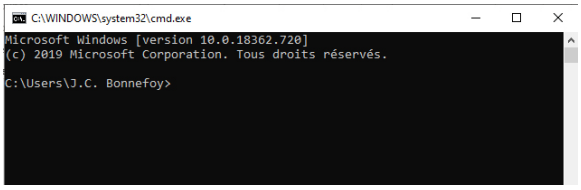
Les logiciels libres sont souvent maintenus par la communauté, mais peuvent aussi l'être par des entreprises qui les utilisent et qui ont intérêt à ce qu'ils restent efficaces et utilisés par d'autres, ce qui assure l'existence de développeurs susceptibles de participer à leur maintien. Les logiciels propriétaires quant à eux sont

essentiellement développés et mis à jour par l'entreprise qui les possède et qui peut décider d'arrêter de les maintenir. Ce sont deux modèles économiques très différents.

> Attention qui dit logiciel libre ne veut pas forcement dire logiciel gratuit (même si c'est souvent le cas), la confusion entre "libre" et "gratuit" vient de l'anglais : "free" veut à la fois dire "libre", mais aussi gratuit.

2. Interface en ligne de commande

À la "préhistoire" des systèmes d'exploitation, ces derniers étaient dépourvus d'interface graphique (système de fenêtres "pilotables" à la souris), toutes les interactions "système d'exploitation - utilisateur" se faisaient par l'intermédiaire de "lignes de commandes" (suites de caractères, souvent ésotériques, saisies par l'utilisateur). Aujourd'hui, même si les interfaces graphiques modernes permettent d'effectuer la plupart des opérations, il est important de connaitre quelques-unes de ces lignes de commandes. Sous Windows, cette interface s'appelle l'invite de commande (cmd)



Sous Linux, on appelle cette interface le Terminal

Dans la suite de ce cours, on ne s'intéressera qu'aux commandes principales (listes et options non exhaustives) de gestion des répertoires et des fichiers sur des OS de type Linux.

2.1 Commandes de gestion des répertoires

Commande Linux	Description	Exemple
pwd	Affiche votre emplacement dans le système de fichiers	`Pwd` Cela renvoie le chemin absolu du répertoire courant

cd	Change de répertoire courant	`cd ..`
		Cela permet de remonter dans le répertoire parent à partir du répertoire courant
		
		`cd /var/www/`
		Cela permet d'aller dans le répertoire /var/www

mkdir	Créé le répertoire passé en argument	`mkdir foo`
		Cela permet de créer le répertoire foo dans le répertoire courant
		
		`mkdir -p foo/bar/baz`
		Cela permet de créer l'arborescence complète foo/bar/baz

rmdir	Supprime le répertoire passé en argument (qui doit être vide)	`rmdir foo` Cela permet d'effacer le répertoire vide foo

2.2 Commandes de gestion des fichiers

Commande Linux	Description	Exemple

ls ou **dir**	Affiche le contenu d'un répertoire	`ls`
		Cela liste le contenu du répertoire (seulement les noms)
		
		`ls -l`
		Cela liste le contenu détaillé du répertoire

mv	Renomme un fichier
	Déplace un fichier
		`mv foo_bar.txt foo_baz.txt`
		Cela renomme le fichier foo_bar.txt en foo_baz.txt
		
		`mv foo/bar.txt baz/`
		Cela déplace le fichier bar.txt dans le répertoire baz

cp	Copie les fichiers	`cp foo/bar.txt baz/`
		Cela copie le fichier bar.txt dans le répertoire baz
		
		`cp -r foo/ baz/`
		Cela copie le répertoire entier foo dans le répertoire baz

rm	Supprime le(s) fichier(s)	`rm foo.txt bar.txt`
		Cela supprime les fichiers foo.txt et bar.txt
		
		`rm *.txt`
		Cela supprime tous les fichiers txt
		
		`rm -rf baz/`
		Cela supprime le répertoire baz et tout son contenu

touch	Crée un fichier vide	`touch toto.txt` Cela crée le fichier vide toto.txt

cat	Ecrire dans un fichier	`cat > toto.txt`
		`essai d'écriture`
		`sur 2 lignes`
		Cela efface le contenu et écrit les 2 lignes dans le fichier toto.txt. Pour terminer et enregistrer on tape sur *Ctrl+z*

grep	Recherche les occurrences d'un mot dans un fichier	`grep essai toto.txt`
		Cela montre les occurrences du mot essai dans le fichier toto.txt
		
		`grep -c essai toto.txt`
		Cela compte les occurrences du mot essai dans le fichier toto.txt

wc	Affiche le nombre de lignes, de mots et de caractères	`wc toto.txt` Cela donne dans l'ordre le nombre de lignes puis de mots, et de caractères dans le fichier toto.txt

2.3 Autres commandes utiles

Commande Linux	Description	Exemple
date	Affiche la date et l'heure	`date` Cela donne la date et l'heure courante
cal	Affiche le calendrier annuel	`Date 2020` Cela affiche le calendrier 2020
exit	Quitte proprement le terminal	`exit` Ferme le terminal

Pour obtenir des informations sur la commande, on utilise le suffixe **--help.** Par exemple `date --help` affichera la documentation de la commande date.

Pour connaitre toutes les options possibles d'une commande donnée, on utilise la commande **man**. Par exemple `man ls` donnera toutes les options de la commande ls

3. Gestion des utilisateurs

Les systèmes de type "UNIX" sont des systèmes multi-utilisateurs, plusieurs utilisateurs peuvent donc partager un même ordinateur, chaque utilisateur possédant un environnement de travail qui lui est propre.

Chaque fichier (et répertoire) est la propriété d'un utilisateur particulier, par défaut c'est l'utilisateur qui a créé le fichier. Les utilisateurs peuvent être réunis en groupes (un utilisateur pouvant faire partie de plusieurs groupes). Il est particulièrement intéressant de pouvoir appartenir à plusieurs groupes car cela permet de définir des ensembles d'actions qui sont autorisées pour certains et pas pour d'autres : on peut ainsi restreindre l'utilisation du lecteur DVD à certains utilisateurs et la possibilité de lire des clefs usb à d'autres, sans que ces deux groupes d'utilisateurs n'aient de liens logiques entre eux. On peut aussi créer un groupe restreint d'utilisateurs qui auraient accès à certains fichiers.

Les droits sur les fichiers sont alors définis en fonctions du type utilisateur :

- Le propriétaire, symbolisé par la lettre **u**
- Le groupe (tous les membres de ce groupe possèdent les mêmes droits), symbolisé par la lettre **g**
- Les autres (ni propriétaire, ni membre d'un groupe), symbolisé par la lettre **o**.
- Tous les types réunis (u+g+o), symbolisé par la lettre **a**.

Seul **l'administrateur**, ou **root**, appelé le **super-utilisateur** dans les systèmes Linux, **peut modifier** le propriétaire d'un fichier, en revanche chacun peut modifier le groupe propriétaire d'un de ses propres fichiers, à condition d'appartenir au nouveau groupe propriétaire.

Il y a trois types de droits (**lecture, écriture, exécution**, respectivement identifiés par les lettres **r, w et x** et les valeurs 4, 2 et 1) dont la signification est résumée dans le tableau ci-dessous :

Droits			Fichier	Répertoire
Lecture	r	4	Regarder le contenu	Lister le contenu
Ecriture	w	2	Modifier le contenu	Ajouter ou supprimer un élément
Exécution	x	1	Exécuter	Passer au travers
Aucun	-	0	Rendre inaccessible	

On modifie les droits d'un fichier (ou d'un répertoire) avec la commande **chmod** avec deux arguments : le premier correspond aux nouveaux droits et le deuxième au nom de fichier. Seul le propriétaire d'un fichier et le super-utilisateur peuvent modifier les droits d'accès à un fichier. Il faudra parfois utiliser dans le terminal en préfixe **sudo** pour être en mode super utilisateur, le mot de passe sera alors demandé !

Commande Linux	Description	Exemple
chmod	Fixe les droits d'accès	`Chmod 746 test.txt` Cela donne les droits suivants au fichier test.txt : rwx (accès total) pour le propriétaire, r (lecture seule) pour le groupe et rw (lecture écriture) pour les autres

3.1 Autorisations les plus courantes pour les fichiers

Valeur littérale	Valeur numérique	Détail
-rw-------	600	Le propriétaire peut lire et écrire. Le groupe et les autres ne peuvent rien faire avec le fichier.
-rw-r--r--	644	Le propriétaire peut lire et écrire, le groupe et les autres peuvent lire.
-rw-rw-rw-	666	Le propriétaire, le groupe et les autres peuvent lire et écrire.
-rwx------	700	Le propriétaire peut lire, écrire et exécuter. Le groupe et les autres ne peuvent rien faire avec le fichier.
-rw--x--x	711	Le propriétaire peut lire, écrire et exécuter. Le groupe et les autres peuvent exécuter.
-rwxr-xr-x	755	Le propriétaire peut lire, écrire et exécuter. Le groupe et les autres peuvent lire et exécuter.
-rwxrwxrwx	777	Le propriétaire, le groupe et autres peuvent lire, écrire et exécuter.

Remarque : le premier caractère de la valeur littéral est un *tiret* pour dire qu'il s'agit d'un fichier.

3.2 Autorisations les plus courantes pour les répertoires

Valeur littérale	Valeur numérique	Détail
drwx------	700	Seul le propriétaire peut lire et écrire dans ce répertoire.
drwxr-xr-x	755	Le propriétaire, le groupe et d'autres peuvent lire le répertoire, mais seul le propriétaire peut modifier son contenu.

Remarque : le premier caractère de la valeur littéral est un *d* pour dire qu'il s'agit d'un répertoire.

3.3 Changer le propriétaire

Si l'on est le super utilisateur, on peut changer le propriétaire d'un fichier ou d'un répertoire à l'aide de la commande **chown**.

Commande Linux	Description	Exemple

chown	Change le propriétaire d'un fichier ou d'un répertoire	`chown martin test.txt`
		Cela affecte la propriété du fichier test.txt à martin
		
		`chown -R martin foo`
		Cela affecte la propriété du répertoire foo et de tout son contenu avec l'option -R (récursif) à martin