






	<h1>L'architecture des systèmes d'exploitation</h1>	
---	---	---

1. Introduction.

Les systèmes d'exploitation sont des logiciels destinés à faciliter l'utilisation d'un ordinateur en offrant une interface simplifiée et en faisant abstraction de la gestion des différentes ressources matérielles du dit ordinateur.

Il existe un grand nombre de systèmes d'exploitation :	
	<p>Microsoft Windows 1.0 – 3.x – 95 – 98 – Me – NT – 2000 – XP – 2003 – Vista – 2008 – 7 – 8 – 10</p>
	<p>GNU/LINUX Debian – Fedora – Gentoo – Mandriva – Red Hat – Slackware – SuSE – Ubuntu</p>
	<p>Mac OS Système 5 – 6 – 7 – 8 – 9 Mac OS X MacOS Sierra – High Sierra – Mojave – Catalina</p>
	<p>BSD (Berkeley Software Distribution) FreeBSD – NetBSD – OpenBSD – DragonFly BSD</p>
	<p>Autres Android – iOS – BeOS – Zephyr – Inferno – LynxOS – Haiku – OS/2 – QNX – Solaris – UNIX – MVS – OS/360 – OS/390 – OS/400 – Plan 9 – ReactOS – VMS – ZETA – FreeRTOS</p>

Dans ce cours nous allons uniquement nous concentrer sur les systèmes basés sur UNIX. C'est le premier système d'exploitation « moderne », développé en C par Ken Thompson et Dennis Ritchie en 1971, il est aujourd'hui la base de plusieurs systèmes comme MacOS ou BSD.

https://linuxmint.com/	https://ubuntu-fr.org/	https://www.raspberrypi.org/
		

2.Le Terminal.

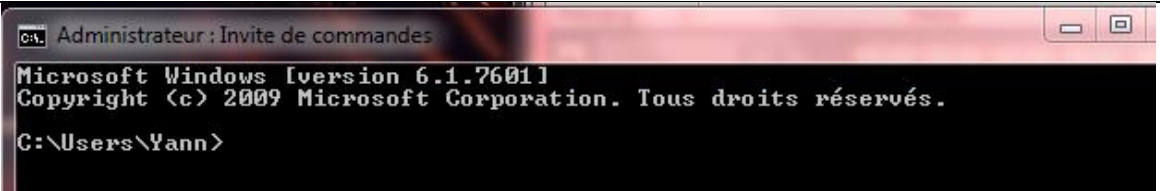
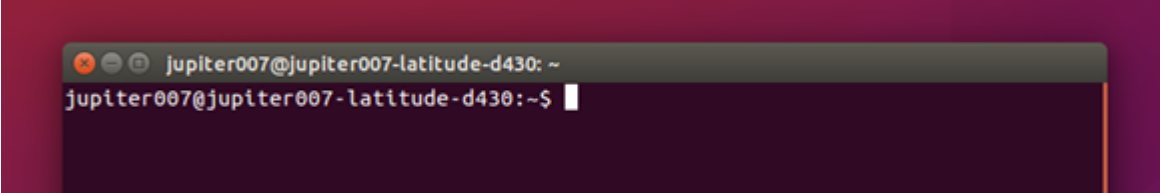
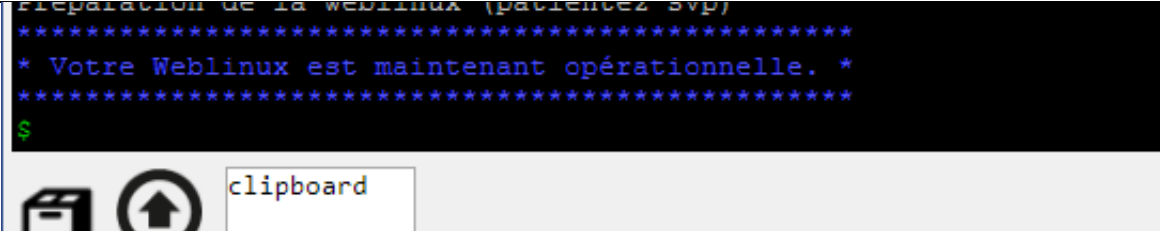
Le terminal est l'interface entre l'Homme et la machine, il offre à l'utilisateur un canal d'entrée et de sortie pour récupérer et fournir des données à un système. Il est souvent intégré à la machine, il peut être physique (écran tactile, clavier+écran, etc.) ou virtuel (console virtuelle).

Exemples de systèmes :

Systeme	Terminal
Un ordinateur	Le clavier, la souris et l'écran
Un smartphone	L'écran tactile
Site Web	Un navigateur Web (FireFox/Chrome)
Une maison	La porte d'entrée

Certains systèmes d'exploitation permettent l'utilisation de la console, terminal virtuel comme le montre les 3 exemples ci-dessous. Ainsi plutôt que d'utiliser l'interface graphique avec des icônes, on communique avec l'ordinateur en mode texte.

Exemples de consoles :

Windows	
Linux	
WebLinux	

2-1. Les commandes UNIX.

Les commandes Unix sont un mot ou une phrase qui indique une suite d'ordres d'exécution à l'ordinateur, elles sont composées d'un nom, peuvent prendre une ou plusieurs options ainsi que des paramètres. On les trouve donc sous cette forme :

```
{nom} {options} {paramètres}'
```

Prenons un exemple : Je souhaite afficher les informations relatives à mon fichier 'Cours.txt' :

- Je vais donc utiliser la commande 'ls' (abréviation de list en anglais). On l'utilise sous la forme :

```
ls {options} {paramètres}
```

```
$ ls
Books          Devoirs      Music        Sequence2    fichier1.txt
Compagnon     Documents    Pictures     Sequence3    fichier2.txt
Cours.txt     Movies       Sequence1   Sequence4    imdb
```

Néanmoins cette commande liste les fichiers de mon répertoire courant sans informations complémentaires, il faut donc que je lui donne une option pour remédier à ce problème.

- Je vais lui donner l'**option** '-l', à noter que **toutes les options de toutes les commandes commencent toujours par un '-'** (tirait du 6). Ma commande s'approche de ce que je souhaite faire :

```
ls -l {paramètres}
```

```
$ ls -l
total 12
drwxr-xr-x  1 alice  user  0 Dec  1 12:19 Books
drwxr-xr-x  1 alice  user  0 Dec  1 12:19 Compagnon
-rw-r--r--  1 alice  user  0 Dec  1 12:36 Cours.txt
drwxr-xr-x  1 alice  user  0 Dec  1 12:19 Devoirs
drwxr-xr-x  1 alice  user  0 Dec  1 12:19 Documents
```

- Comme je ne souhaite voir les informations que d'un seul fichier, je vais donner en **paramètre** le nom du fichier. Ainsi ma commande prendra cette forme :

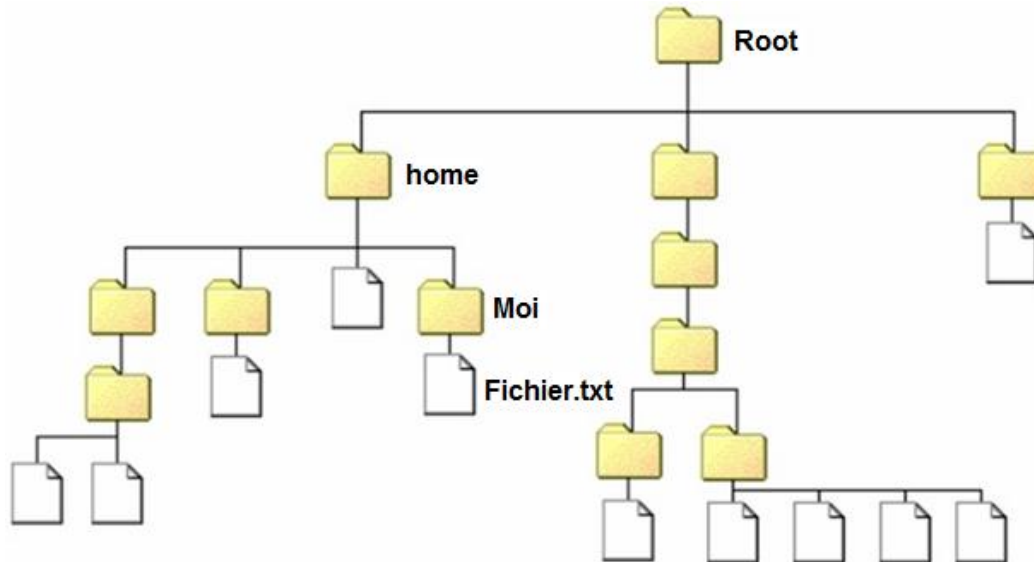
```
ls -l Cours.txt
```

```
$ ls -l Cours.txt
-rw-r--r--  1 alice  user  0 Dec  1 12:36 Cours.txt
$
```

3. Le système de gestion de fichier (SGF).

Dans un système d'exploitation, les données sont rangées dans des **fichiers**, eux-mêmes organisés hiérarchiquement dans des **répertoires**. Cette hiérarchie est un **arbre** au sens informatique dont les nœuds sont les répertoires et les feuilles les fichiers.

Exemple :



Un répertoire ne peut être contenu que dans un unique répertoire appelé **père**, et le répertoire se trouvant tout en haut de cette hiérarchie est la **racine** (root). Par convention, le père du répertoire racine est lui-même.

Exemple : Le répertoire « **home** » est le père du répertoire « **moi** »

3-1. Le chemin.

Sur un système UNIX, les répertoires et fichiers peuvent être désignés par un chemin soit depuis la racine (chemin **absolu**), soit depuis le répertoire courant (chemin **relatif**). Les chemins sont notés par les noms des répertoires traversés séparés par le caractère **oblique /** (slash).

Exemple de chemin **absolu** :

- Si un fichier « Fichier.txt » se trouve dans le répertoire utilisateur « Moi », son chemin **absolu** sera « **/home/Moi/Fichier.txt** ».

Comme vous pouvez le voir, le caractère oblique / désigne aussi la racine, ainsi tout chemin **absolu** commencera par ce caractère.

Exemple de chemin **relatif** :

- A partir du répertoire « home », le chemin **relatif** sera « **./Moi/Fichier.txt** » (ou « **Moi/Fichier.txt** »).

Le caractère point « . » désigne le répertoire courant, ici home. Il est possible de remonter l'arborescence en utilisant deux points de suite « .. » pour indiquer le répertoire père.

```

$ pwd
/home
$ ls -la
total 4
drwxr-xr-x  1 root  root   162 Nov 24 10:41 .
drwxrwxrwx  1 root  root    0 Nov 24 10:41 ..
drwxr-xr-x  1 root  root   86 Nov 24 10:56 Moi
  
```

3-2. Les droits.

Sur Unix, tout fichier et tout dossier disposent d'un propriétaire et de droits. Les utilisateurs sont réunis en 3 groupes : **Propriétaire (u)**, **groupe propriétaire (g)** et **autre utilisateur (o)**. Ces groupes permettent de donner des droits différents pour chaque utilisateur. Les différents droits pouvant être octroyés sont : **la lecture (r, 4)**, **l'écriture (w, 2)** et **l'exécution (x, 1)**.

Les différents droits		
	Fichier	Répertoire
Lecture	Voir le contenu	Lister le contenu
Écriture	Modifier le contenu	Ajouter ou supprimer un élément
Exécution	Exécuter	Passer au travers

- (Fichier)	read	write	execute	read	Not write	execute	Not read	Not write	Execute
	r	W	X	r	-	X	-	-	X
d (Répertoire)	4	2	1	4	0	1	0	0	1
	Propriétaire (u) User			groupe propriétaire (g) Group			autre utilisateur (o) Other		

La modification des droits d'un fichier ou d'un répertoire se fait de deux façons : **Absolute** ou **relative**.

Exemple de changement **absolu** :

Commande « **chmod 751 fichier.txt** »

On remarque le nombre 751, chaque chiffre correspond aux droits appliqués à un groupe. Les groupes sont toujours dans le même ordre : **Propriétaire (u)**, **groupe propriétaire (g)** et **autre utilisateur (o)**. Les valeurs représentent les droits : 7 pour tous les droits, 5 pour la lecture et l'exécution, et 1 pour l'exécution. (On remarque 7=4+2+1, 5=4+1, etc...)

```
$ ls
Fichier.txt
$ ls -l
total 1
-rwxr-x--x  1 root  root  6 Nov 24 11:25 Fichier.txt
```

Exemple de changement **relatif** :

Commande « **chmod u-r,g+w,o=x fichier.txt** »

On peut décomposer la modification des droits de cette commande en 3 parties :

- u-r : Qui au groupe **u(Propriétaire)** soustrait (-) le droit de lecture (**r**) sans modifier les autres droits du groupe.
- g+w : Qui au groupe **g(Groupe propriétaire)** octroie (+) le droit d'écriture (**w**) sans modifier les autres droits du groupe
- o=x : Qui au groupe **o(Autre)** définit (=) le droit d'exécution (**x**). On remarque que cette modification est bien **absolue**, c'est donc une autre façon de faire.

```
$ chmod u-r,g+w,o=x Fichier.txt
$ ls -l
total 1
--wxrwx--x  1 root  root  6 Nov 24 11:25 Fichier.txt
```

	Not read	write	execute	read	write	execute	Not read	Not write	Execute
- (Fichier)	r	W	X	r	W	X	-	-	X
d (Répertoire)	0	2	1	4	2	1	0	0	1
	Propriétaire (u) Owner			groupe propriétaire (g) Group			autre utilisateur (o) Other		

Exercice : A partir du tableau ci-dessous, donnez les commandes linux associées afin de modifier les droits du fichier.txt

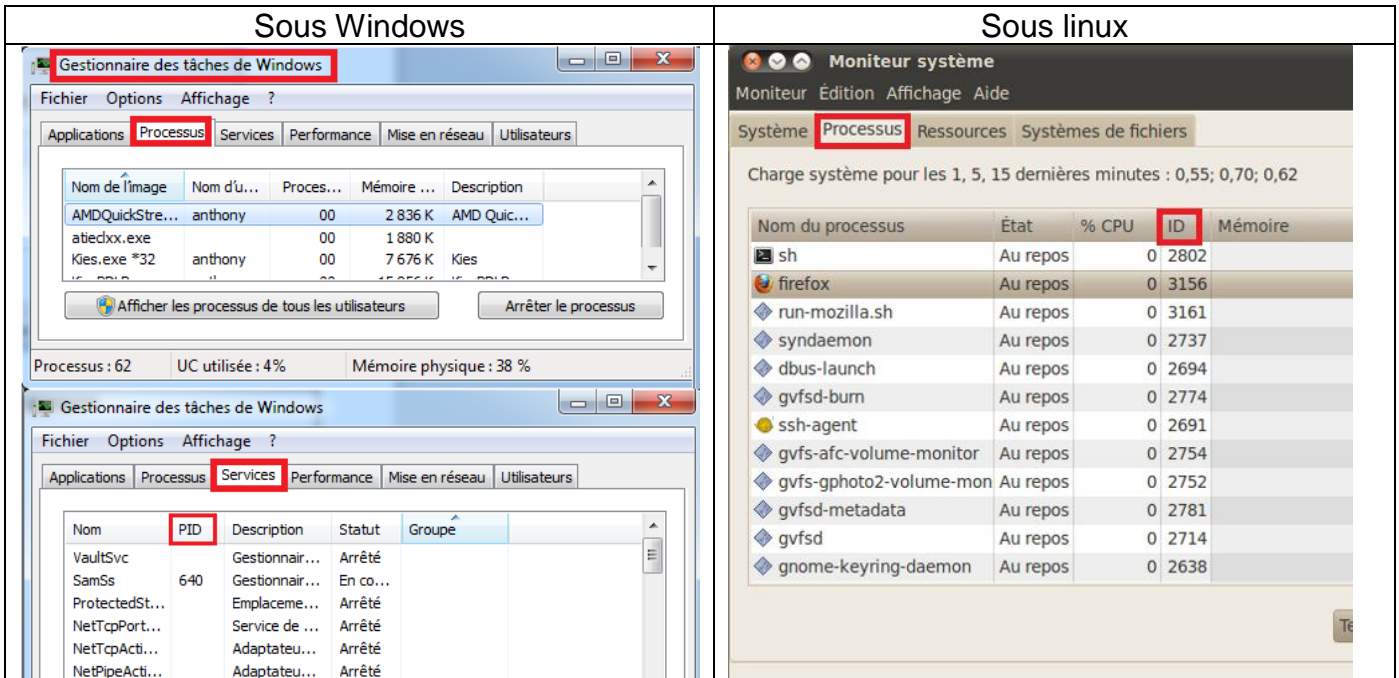
	read	write	execute	read	write	Not execute	Not read	Not write	Not Execute
- (Fichier)	r	W	X	r	-	X	-	-	-
d (Répertoire)	4	2	1	4	0	1	0	0	0
	Propriétaire (u) Owner			groupe propriétaire (g) Group			autre utilisateur (o) Other		

chmod

4. Les processus.

Un processus est la représentation d'un programme en cours d'exécution, il peut néanmoins y avoir plusieurs processus pour un unique programme, comme lorsqu'on lance simultanément deux interpréteurs python.

Exemple de processus :



Chaque processus est identifié par un entier unique appelé **PID** (Process IDentifier).

Lorsqu'un processus s'exécute, il peut être en **mode noyau** ou en **mode utilisateur**. Un processus passe en **mode noyau** dès qu'il manipule des données importantes pour le bon fonctionnement de l'ordinateur, il ne peut alors plus être interrompu afin de garantir l'intégrité des données manipulées.

Exemple : Un charpentier répare un trou dans la coque d'un navire, interrompre son travail ne serait pas une bonne idée.

Tous les processus créés le sont de la même façon :

- Un processus **père** est cloné à sa propre demande.
- Le clone se voit assigner un nouveau **PID**.
- Tout processus se voit aussi assigner un **PPID** (Parent Process IDentifier), c'est-à-dire le **PID** de son processus **père**, cela permet de garder un lien entre les 2 processus.

Exemple : Si l'un de vos parents vous demande de lui apporter quelque chose, il est important pour vous de vous rappeler lequel de vos parents vous a fait cette demande.

Par exemple : Lorsque l'on effectue la commande 'ps', le processus de la console (bash) se copie, puis on assigne à cette copie un **PID**, on lui donne le nom de 'ps'. Son **père** est le processus 'bash', ainsi son **PPID** est le **PID** de 'bash' :

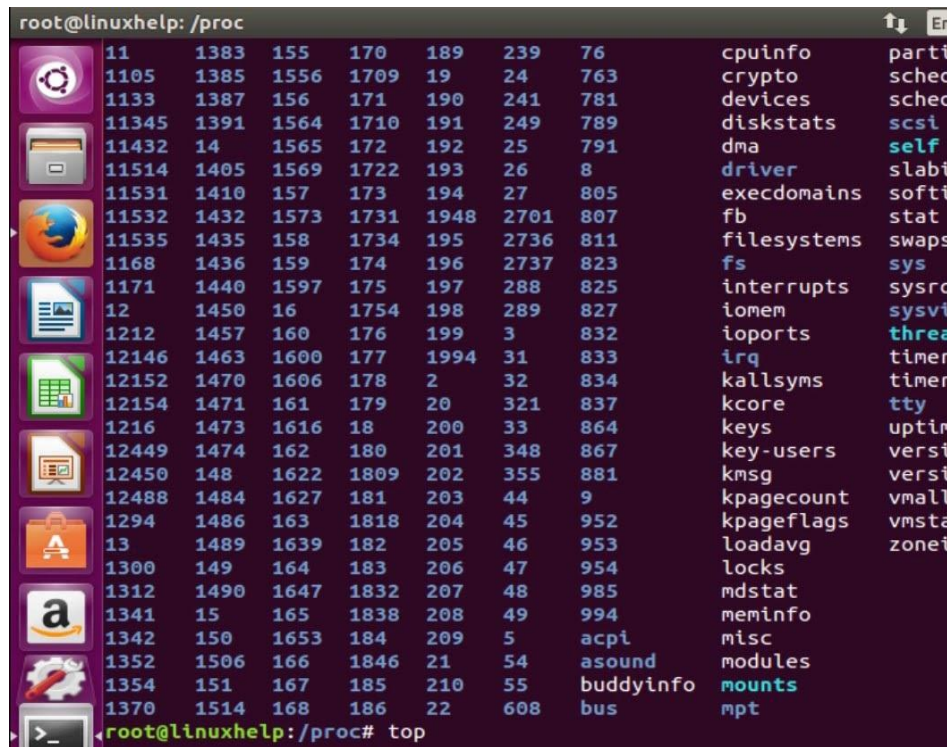
```

$ ps -l
F S  UID  PID  PPID  C  PRI  NI ADDR  SZ  WCHAN  TTY  TIME CMD
4 S  1000  59   1   0  80  0  -   339  -  ttyS0  15049-12:58:15 bash
0 R  1000  67   59  0  80  0  -   179  -  ttyS0  15049-12:58:15 ps
    
```

Lorsqu'un processus a fini sa tâche, il devient un processus dit **zombie**, il perd tous ses droits d'accès au processeur mais ses données sont conservées jusqu'à ce que son **père** constate sa fin pour le supprimer et potentiellement récupérer une donnée renvoyée.

Si un processus **père** se termine, ses fils seront alors dits **orphelins**, leur nouveau **père** sera alors le processus de **PID 1**, le processus **init**.

Sous les environnements Linux, le **SGF** (système de gestion de fichiers) contient le répertoire **/proc** qui reflète une partie des informations des processus du système. On y trouve un répertoire par processus dont le nom est le **PID** du dit processus.



```

root@linuxhelp: /proc
11      1383  155  170  189  239  76      cpuinfo  parti
1105    1385  1556 1709  19   24   763    crypto  sched
1133    1387  156   171  190  241  781    devices sched
11345   1391  1564 1710  191  249  789    diskstats scsi
11432   14    1565 172   192  25   791    dma      self
11514   1405  1569 1722  193  26   8      driver   slabi
11531   1410  157   173  194  27   805    execdomains softi
11532   1432  1573 1731  1948 2701 807    fb       stat
11535   1435  158   1734 195  2736 811    filesystems swaps
1168    1436  159   174  196  2737 823    fs       sys
1171    1440  1597 175   197  288  825    interrupts sysrq
12      1450  16    1754 198  289  827    iomem    sysvi
1212   1457  160   176  199  3    832    ioports  threa
12146   1463  1600 177   1994 31   833    irq      timer
12152   1470  1606 178   2    32   834    kallsyms timer
12154   1471  161   179  20   321  837    kcore    tty
1216    1473  1616 18    200  33   864    keys     uptim
12449   1474  162   180  201  348  867    key-users versi
12450   148  1622 1809  202  355  881    kmsg     versi
12488   1484  1627 181   203  44   9      kpagecount vmall
1294    1486  163   1818 204  45   952    kpageflags vmta
13      1489  1639 182   205  46   953    loadavg  zonei
1300    149  164   183  206  47   954    locks
1312   1490  1647 1832  207  48   985    mdstat
1341    15   165   1838 208  49   994    meminfo
1342    150  1653 184   209  5    acpi    misc
1352   1506  166   1846 21   54   asound  modules
1354    151  167   185  210  55   buddyinfo mounts
1370    1514  168   186  22   608    bus     mpt

```

root@linuxhelp: /proc# top

Annexe : Lexique

SE/OS	Système d'exploitation / Operating system
Terminal	Point d'entrée et de sortie d'un système
Console	Terminal virtuel de l'ordinateur
Bash	Bourne-Again shell, interpréteur de script sous Linux
SGF	Système de gestion de fichiers
root	Répertoire racine dans un SGF
/	Désigne le répertoire racine et permet la séparation de nom de répertoire et/ou fichier dans un chemin
~	Désigne le répertoire utilisateur
u,g,o	Les groupes utilisateurs Propriétaire, Groupe propriétaire et Autre
r,w,x	Les droits de Lecture, Écriture et Exécution
PID	Process IDentifier
PPID	Parent Process IDentifier
Zombie	Un processus qui a fini sa tâche mais qui n'est pas encore supprimé
Orphelin	Un processus dont le père n'est plus, son nouveau père devient alors init http://manpagesfr.free.fr/man/man8/init.8.html
Init	Le processus au PID 1
man	Commande permettant d'accéder aux pages de manuel installées sur le système Ex : man ls
ls	List : commande qui liste le contenu d'un répertoire
pwd	Print Working Directory
chmod	Change mode : commande de modification des droits d'accès
ps	Affiche les processus en cours d'exécution
cd	Change directory (commande pour changer de répertoire courant)