

Chapitre X : Système d'exploitation

I - Informations générales

a) Définition

Un système d'exploitation, OS (*Operationg System*) est un **programme exécuté au démarrage** d'une machine. Il permet de gérer les fichiers, les répertoires, les processus, les périphériques en proposant des outils.

Les OS les plus répandus sont Windows, MacOS, GNU/Linux (avec différentes distributions dont Debian, Ubuntu, Fédora) pour ordinateur, et Androïd, iOS pour mobile.

Le système d'exploitation permet aussi de gérer l'authentification de chaque utilisateur ainsi que les **différents droits d'accès sur les fichiers (lecture, écriture, suppression)**. On distingue ainsi un compte administrateur et des comptes utilisateurs.

Un utilisateur interagit avec des programmes (jeu, navigateur, traitement de texte, ...). Ces derniers ont besoin d'utiliser les ressources de la machine pour effectuer leurs tâches (lire ou sauvegarder des fichiers, afficher des images à l'écran, gérer l'utilisation du clavier ou de la souris, ...). Le système d'exploitation offre alors des fonctions permettant d'interagir avec le matériel.

Dans le système d'exploitation, on retrouve :

- l'ordonnanceur qui décide quel programme s'exécute à un instant donné sur le processeur ;
- le gestionnaire de mémoire qui répartit la mémoire vive entre les différents programmes en cours d'exécution ;
- les différents systèmes de fichiers qui définissent la manière de stocker les fichiers sur les supports physiques (disque dur, clé USB, ...) ;
- la pile réseau qui implémente entre autre des protocoles tels que TCP/IP ;
- les pilotes de périphériques qui permettent de gérer les périphériques matériels (carte graphique, disques durs, ...).

Exemple : voici un programme python :

```
1 from os.path import getsize
2 from tkinter import *
3 from tkinter.filedialog import askopenfilename
4
5 Fenetre = Tk()
6 fichier = askopenfilename()
7 if fichier:
8     print("le fichier",fichier,"fait",getsize(fichier),"octets")
9 Fenetre.destroy()
```

Il utilise le module tkinter (module avant tout graphique permettant de créer une IHM en python), mais on utilise ici la fonction `askopenfilename()` qui crée une boîte de dialogue permettant de sélectionner un fichier. Une fois le fichier sélectionné, cette fonction renvoie le chemin du fichier sélectionné ; la fonction `getsize` en détermine alors sa taille en octets.

Lorsqu'on exécute ce programme, le reste du système continue de fonctionner (y compris si le fichier sélectionné est déjà ouvert). En programmant, on n'a pas à s'en soucier, c'est le système d'exploitation qui s'en occupe.

On peut enfin remarquer que le temps d'affichage de la phrase ne dépend pas de la taille du fichier. Le système d'exploitation conserve cette information relative à chaque fichier (c'est une métadonnée) - elle est d'ailleurs le plus souvent affichée dans l'explorateur de fichiers.

b) Le standard POSIX et l'interface

Même s'il existe une grande diversité de systèmes d'exploitation, il existe un ensemble de standards regroupés sous le nom de POSIX.

Windows n'est pas toujours compatible avec ces standards, certaines commandes seront ainsi différentes de celles que je vais présenter. Voilà pourquoi nous ne travaillerons pas avec le Shell windows (les lignes de commandes pouvant être différentes) mais nous devons utiliser une interface Linux.

Je ne vous demande pas d'installer Linux sur votre ordinateur (même si ce serait une éventualité), je vous propose ainsi de travailler de l'une des deux façons suivantes :

- utiliser une machine **Weblinux** de puis un navigateur.

Se rendre à l'adresse

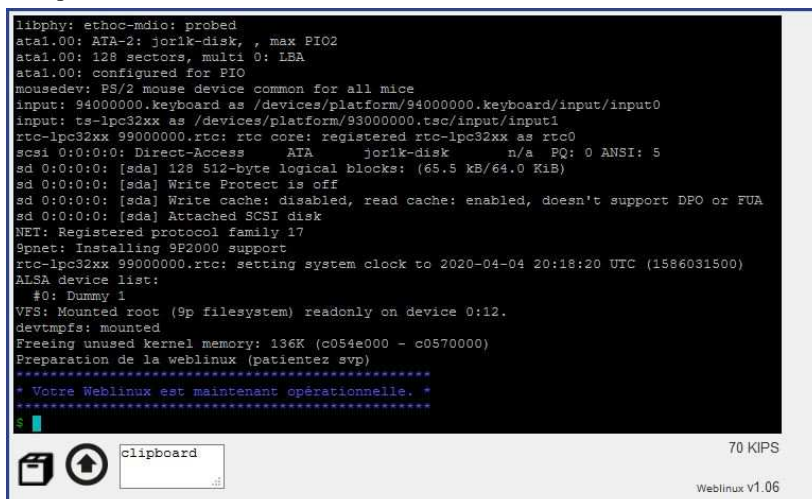
<http://weblinux.univ-reunion.fr>.

Après lancement, on se connecte automatiquement, en tant qu'utilisatrice alicia et on se trouve dans son répertoire personnel.

C'est celui dans lequel on reviendra lorsqu'on écrira `cd ~` - voir la liste des commandes plus bas.

Il correspond à `/home/alice`.

Les commandes s'écrivent toujours après le symbole `$`.



```
libphy: ethoc-mdio: probed
ata1.00: ATA-2: jorik-disk, , max PIO2
ata1.00: 128 sectors, multi 0: LBA
ata1.00: configured for PIO
mousedev: PS/2 mouse device common for all mice
input: 94000000.keyboard as /devices/platform/94000000.keyboard/input/input0
input: ts-lpc32xx as /devices/platform/93000000.tsc/input/input1
rtc-lpc32xx 99000000.rtc: rtc core: registered rtc-lpc32xx as rtc0
scsi 0:0:0:0: Direct-Access ATA jorik-disk n/a PQ: 0 ANSI: 5
sd 0:0:0:0: [sda] 128 512-byte logical blocks: (65.5 kB/64.0 KiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
sd 0:0:0:0: [sda] Attached SCSI disk
NET: Registered protocol family 17
9pnet: Installing 9P2000 support
rtc-lpc32xx 99000000.rtc: setting system clock to 2020-04-04 20:18:20 UTC (1586031500)
ALSA device list:
#0: Dummy 1
VFS: Mounted root (9p filesystem) readonly on device 0:12.
devtmpfs: mounted
Freeing unused kernel memory: 136K (c054e000 - c0570000)
Preparation de la weblinux (patiencez svp)
*****
* Votre Weblinux est maintenant opérationnelle. *
*****
$
```

- Installer la suite de programmes Cygwin (<https://www.cygwin.com> - seulement pour Windows - en téléchargeant le fichier `setup-x86_64.exe` de la partie « Installing Cygwin » puis en l'exécutant). C'est une suite de logiciels libres qui permet de simuler un système POSIX sous Windows.

Dans ce cas, le dossier `home` contient uniquement un dossier portant votre nom d'utilisateur qui sera vide. Pour travailler dans les mêmes conditions, il suffit de télécharger le fichier nommé `alice.zip` et le décompresser dans le dossier `C:\cygwin64\home`.

Lancer le programme `Cygwin64 Terminal` puis écrire `cd /home/alice` pour vous situer dans la même configuration que la Weblinux.

Travailler avec la Weblinux est pratique car aucune installation n'est à effectuer. Des dossiers et des fichiers supplémentaires sont présents par rapport au dossier présent dans `alice.zip`, mais ils ne serviront pas. Par contre, il ne sera pas toujours facile de se repérer dans les dossiers et vous devrez régulièrement utiliser les commandes `pwd` (pour savoir où vous êtes) mais surtout `ls` pour connaître le contenu du dossier.

ATTENTION : si vous fermez votre navigateur et que vous revenez ensuite, vous reprendrez la configuration initiale, il n'y aura aucun moyen d'enregistrer pour y revenir ensuite.

Avec Cygwin, vous pouvez travailler avec l'explorateur de fichiers à côté. Cela vous permet de contrôler ce que vous faites (quand vous créez ou supprimez des fichiers) ou quand vous vous déplacez dans l'architecture.

ATTENTION : ne sortez surtout pas du dossier `/home/alice` car vous pourriez faire des bêtises sur votre ordinateur...

Si vous voulez aller un peu plus loin, vous pouvez tout à fait installer une distribution Linux sous Windows, mais ce n'est pas automatique :

- Installer WSL. Les instructions se trouvent ici : <https://docs.microsoft.com/fr-fr/windows/wsl/install-win10>;
- télécharger Ubuntu 18.04 sur le Windows Store;
- puis initialiser la distribution : <https://docs.microsoft.com/fr-fr/windows/wsl/initialize-distro>.
Je vous conseille de mettre un mot de passe très simple, du style « linux », car vous risquez de ne pas vous en servir très souvent.

c) Gestion des droits

L'un des intérêts des systèmes conformes au standard POSIX est d'avoir su dès le départ gérer rigoureusement les droits sur les fichiers et répertoires, d'où une bonne solidité de ces systèmes (massivement utilisés en particulier sur les serveurs).

Un utilisateur fait partie de groupe(s), dont un par défaut. En fait, le système d'exploitation l'identifie par un numéro (UID, identifiant d'utilisateur ou user ID) ainsi que par ses groupes (par leur GID).

Vous pouvez utiliser la commande `id` (pas incontournable, pour la suite), afin de déterminer les groupes auxquels appartient l'utilisateur actuel (alice sur la Weblinux et votre nom d'utilisateur sur Cygwin).

Quand il crée un fichier ou un répertoire, ce dernier « appartient » à cet utilisateur, ainsi qu'à son groupe par défaut (des droits sont alors choisis, configurables avec la commande `umask` que nous expliquerons plus bas). Pour définir ces droits associés au fichier, on divise alors le monde en trois catégories :

- l'utilisateur propriétaire (désigné par `u`) ;
- les membres du groupe propriétaire (`g`) ;
- tous les autres utilisateurs (`o` pour *others*).

Et pour chacune de ces catégories, on attribue ou non chacun des droits suivants :

- lecture (`r` pour *read*) qui autorise donc la copie, pour un fichier ordinaire ; pour un répertoire, il permet d'obtenir la liste de ses fichiers ;
- écriture (`w` pour *write*) qui permet notamment la modification (pour un répertoire, l'ajout, la suppression, le renommage des fichiers qu'il contient) ;
- Exécution (`x` pour *execute*, qui indique pour un fichier ordinaire qu'il peut-être considéré comme une commande, pour un répertoire, cela autorise à se positionner dedans, par exemple avec `cd`).

Testons un peu pour voir

Se positionner à la racine du dossier alice (on peut vérifier si on y est en tapant `pwd` - qui affiche le répertoire courant, et si besoin taper `cd /home/alice` pour être dans le répertoire d'alice).

Saisir la commande : `ls -l`

`ls` permet de lister le contenu du répertoire. L'option `-l` permet d'avoir un affichage sous la forme d'une liste avec des informations supplémentaires par répertoire ou fichier (en ligne de commande, toutes les options sont précédées d'un tiret et peuvent être cumulées par exemple `ls -al`).

On peut aussi ajouter un argument pour n'avoir que certains fichiers ou dossiers.

La première ligne (total 12) indique le poids total du contenu : 12 ko ici.

Prenons alors deux exemples dans cette liste :

– le répertoire Documents :

```
drwxr-xr-x      1  alice    user              0  Apr      6  07:17  Documents
```

– le fichier fichier1.txt :

```
-rw-rw-r--      1  alice    user            3024  Apr      6  07:17  fichier1.txt
```

J'ai pris des lignes de Weblinux, sur Cygwin c'est très légèrement différent.

Le premier élément de cette ligne est la déclaration des droits.

- Le premier caractère `d` signifie que c'est un répertoire (*directory*), on met un tiret `-` pour un fichier.
- Les trois suivants sont les droits du propriétaire : pour le répertoire, `rw` signifie que le propriétaire peut lire, écrire et exécuter (s'y déplacer) le répertoire. Pour le fichier, alice peut lire et écrire.

Le fichier est un fichier `txt`, ce n'est pas un exécutable, donc le `x` sera absent pour toutes les personnes.

On a toujours trois caractères, idem pour les permissions suivantes, l'absence d'autorisation est indiquée par un tiret `-`.

- les trois suivants sont les droits de n'importe quel membre du groupe.

Un membre du groupe peut lire et exécuter le dossier. Il ne peut pas ajouter de fichier dedans (w est absent).

Un membre du groupe peut lire et modifier le fichier.

- Les trois derniers sont pour tout autre utilisateur non membre du groupe d'alice. Pour le répertoire, les droits des autres utilisateurs sont les mêmes que pour les membres du groupe d'alice. Par contre pour le fichier, il n'est pas possible de modifier le fichier.

Les droits peuvent être modifiés par l'administrateur du système et par le propriétaire. On utilise la fonction `chmod` (*change mode*).

La syntaxe est de la forme : `chmod modifications fichier`, où modifications est composé

- du public (une ou plusieurs lettres parmi u, g et o définis ci-dessus, voire a pour « tous », soit all), puis
- d'un opérateur (= pour attribuer des droits et seulement ceux-là, + pour ajouter des droits à ceux déjà donnés et - pour en ôter à ceux qui existent) et enfin
- du ou des droits désignés par une ou plusieurs des lettres parmi r, w et x.

Exemples :

- `chmod g-w fichier1.txt` → ôte les droits en écriture aux membres du groupe sur ce fichier ;
- `chmod a+w fichier1.txt` → ajoute les droits en écriture sur le fichier à tous les utilisateurs
- `chmod a+w *.txt` → c'est toujours l'ajout de droits en écriture, mais cette fois-ci, « *.txt » signifie que cela s'appliquera à tout fichier dont le nom se termine par .txt (donc à tous les fichiers texte).

Remarques :

- il existe des syntaxes (comme *) permettant d'accéder à plusieurs fichiers et ainsi effectuer un traitement en une ligne sur plusieurs fichiers ou dossiers, mais je ne détaillerai pas ceci ;
- sur les répertoires, on peut ajouter l'option -r (juste après chmod) pour que cela s'applique récursivement à tout ce que contient le répertoire.

Voici un lien pour ceux qui souhaiteraient aller plus loin :

<https://openclassrooms.com/fr/courses/43538-reprenez-le-controle-a-laide-de-linux/39044-les-utilisateurs-et-les-droits>

II - Commandes Linux

Liste des commandes de base sous Linux

- Commandes d'aide
 - man** : retourne le mode d'emploi de la commande s'il existe (avec toutes les options possibles).
exemples : `man ls` `man cd` (ls et cd sont des commandes, voir plus loin)
 Taper « q » pour quitter l'affichage
 - help** : Affiche l'aide de la commande
`help cd` → affiche l'aide de la commande cd
- Commandes « arborescences des dossiers et fichiers »
 - cd** : Change le répertoire de travail du shell.
exemples `cd /home` → nous renvoie dans le dossier home (père du répertoire personnel)
`cd ..` → change de répertoire de travail pour accéder au répertoire père.
`cd ~` → change le répertoire de travail pour revenir au répertoire personnel
 - cp** : Copie le(s) fichier(s)
exemples : `cp fichier.txt fichier2.txt` → crée fichier2.txt comme copie de fichier.txt
 Attention car dans ce cas, si fichier2.txt existe déjà, il sera tout d'abord détruit.
 - cp -r** Copie un répertoire et tout son contenu
exemple : `cp -r rep1 rep2` → tous les dossiers, répertoires, sous-dossiers de rep1 seront copiés dans rep2)

- ls** Liste les éléments présents dans un répertoire
exemple : `ls /home/alice/rep1` → donne la liste de tous les éléments du répertoire rep1 de l'utilisateur alice.
 si on ne précise pas de répertoire, ls donne les éléments du répertoire courant.
- ls -a** l'option -a permet d'afficher également les fichiers cachés (ceux qui commencent par un .)
- ls -l** présente les éléments sous la forme d'une liste avec les droits et permissions
 On peut combiner des options : `ls -al`
- mkdir** Crée un répertoire
exemple : `mkdir /home/alice/toto`
- mv** Déplace ou renomme un fichier
exemples : `mv test.txt fichier.txt` (renomme le fichier test.txt en fichier.txt)
`mv test.txt /home/alice/Documents` déplacera le fichier dans le répertoire Documents (message d'erreur s'il n'existe pas)
- rm** Supprime un ou plusieurs fichiers
exemples : `rm fichier1.txt fichier2.txt` (supprime ces deux fichiers)
- rm -R** Supprime un répertoire et son contenu
exemple : `rm -R /home/test/`
- rmdir** Supprimer un répertoire vide
exemples : `rmdir dossier` `rm /home/alice/dossier`
- pwd** Affiche le répertoire courant
- Commandes « gestion des droits »
 - chmod** Modifie les permissions d'accès à un fichier ou à un répertoire.
 - chown** Change le propriétaire et le groupe propriétaire d'un fichier
 - chgrp** Change le groupe propriétaire d'un fichier.
- Commandes « Fichiers »
 - cat** pour concaténer des fichiers ou afficher le contenu d'un fichier.
exemples : `cat fichier1 fichier2` (affiche la concaténation de fichier1 et fichier2)
`cat fichier1 fichier2 > fichier3` (concaténation de fichier1 et fichier2 dans fichier3)
 - less** Affiche le contenu d'un fichier
exemple : `less test.txt` (affiche le fichier test.txt, taper q pour quitter)
 - more** Affiche le contenu d'un fichier page par page
 - touch** Change la date de modification d'un fichier. Si le fichier n'existe pas, la commande crée un fichier vide.
 - echo** Affiche une ligne de texte.
 - echo >>** Envoie une ligne de texte vers une sortie (par exemple un fichier - le crée s'il n'existe pas)
exemple : `echo "Hello World" >> fichier.txt` (écrit le texte "Hello World" à la fin du fichier "fichier.txt")
 - wc** Affiche des informations sur le (ou les) fichiers
exemple : `wc fichier1.txt` donne l'index de la dernière ligne (cela commence à 0), le nombre de mots et le nombre de caractères du fichier
 - find** Trouve (s'il existe) un fichier dans le dossier courant ou ses sous-dossiers et renvoie son chemin
exemples : `find -iname truc.txt` → renvoie le chemin situant ce fichier
`find -iname *.txt` → renvoie tous les fichiers dont le nom se termine par ".txt" (au finale tous les fichiers texte)
 -name pourrait suffire, mais -iname permet de ne pas être sensible à la casse

Il existe de nombreuses autres commandes.

Vous pourrez en trouver sur ce site : <http://juliend.github.io/linux-cheatsheet/>.

III - Tester différentes commandes

Cette partie est à faire dans moodle en utilisant les commandes précédentes.

Remarque : pour écrire une nouvelle commande, le curseur doit se situer après le symbole \$.

Si la Weblinux ne comprend pas une commande et que le curseur ne revient pas après le symbole \$, utiliser Ctrl+C ou q pour quitter.

Sources :

1. Numérique et sciences Informatiques (Ellipse) : *Balabonski, Conchon, Filliâtre, Nguyen*
2. Spécialité umérique et sciences Informatiques (Ellipse) : *Bays*
3. site informatiquelycee : https://pixees.fr/informatiquelycee/n_site/nsi_prem.html
4. Fichier Ligne_de_commande_POSIX.pdf de JM Gervais.