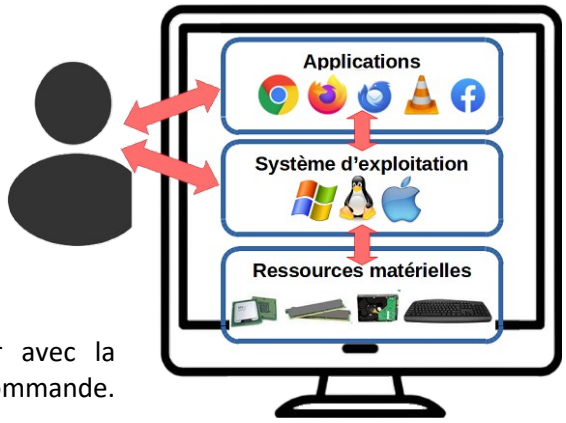
Architectures Matérielles et Systèmes d'Exploitation

Système d'exploitation

Fonctions du système d'exploitation

Le système d'exploitation (*Operating System* ou OS en anglais) est le logiciel principal d'un ordinateur. Sans système d'exploitation, un ordinateur est inutilisable. Il joue plusieurs rôles :



- **Exécution des applications** : il permet d'exécuter les programmes et gère leur utilisation du processeur.
- **Gestion des ressources partagées** : il alloue la mémoire vive (RAM) aux programmes qui s'exécutent, organise sous forme d'arborescence les fichiers stockés sur le disque dur, fait l'interface avec les périphériques (clavier, souris, écran, carte réseau, etc.)
- **Interface utilisateur** : il permet à l'utilisateur d'interagir avec la machine à travers à une interface graphique ou en ligne de commande.

Les systèmes d'exploitation les plus connus sont Microsoft Windows, Apple macOS, Google ChromeOS et les distributions Linux (Ubuntu, Debian, Zorin, etc.) sur ordinateurs ; Android et iOS sur smartphones.

Libre vs propriétaire

Linux est un système **libre** et **gratuit**[1] , par opposition à Windows et MacOS qui sont **propriétaires** et **payants**. Il est aujourd'hui le système d'exploitation utilisé sur tous les serveurs des géants du Web et les supercalculateurs.

Un logiciel libre (ou Open Source) est un logiciel qui permet l'utilisation, l'étude, la duplication et la modification de son code source.

Un logiciel libre est la plupart du temps **gratuit**, mais pas toujours[2], par exemple Red Hat commercialise une distribution payante de Linux avec notice et contrat d'assistance. Et il existe de nombreux logiciels gratuits qui ne sont pas libres (Adobe Acrobat Reader, Zoom, Dropbox, etc.)

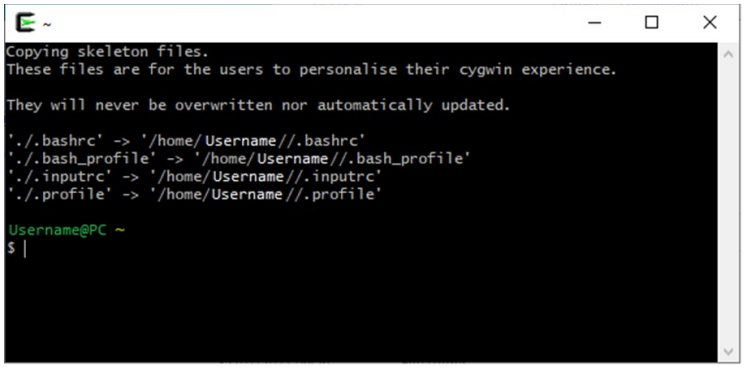
Un **système d'exploitation libre n'est pas moins sécurisé** qu'un système d'exploitation propriétaire.

Interpréteur de commande ou shell

Cette activité est réalisée en utilisant Cygwin[3], un émulateur de terminal Linux sous Windows.

1. Ouvrez l'émulateur Cygwin.

   Un terminal s'ouvre avec une invite de commande symbolisée par « $ ».



L'**interpréteur de commandes,** aussi appelé **invite de commande,** ou encore ***shell,*** permet d'interagir avec le système d'exploitation. Il est disponible dans tous les systèmes d'exploitation de type Linux. L'utilisateur lance des commandes sous forme d'une entrée texte pour être ensuite exécutées par le *shell*.

1 Python, Android, Mozilla Firefox et Thunderbird, LibreOffice.org, VLC sont d'autres exemples de logiciels libres et gratuits.
2 La confusion entre libre et gratuit vient de l'anglais qui utilise *free* pour les deux.
3 https://www.cygwin.com/

Un interpréteur de commande en mode texte peut sembler obsolète sur les ordinateurs personnels qui disposent tous d'un système d'exploitation avec une interface graphique, mais ce n'est pas le cas sur les tous serveurs. De plus, il offre certains bénéfices: il est universel à toutes les distributions et facilite les échanges entre utilisateurs ; enfin il permet d'automatiser des tâches répétitives.

2. Exécutez la commande « `echo $SHELL` » pour afficher la variable `$SHELL`. Quel est le shell utilisé ici ?

bash

À l'origine, le *shell* par défaut était `sh` ou `bsh` (pour *Bourne shell* [4]) qui donna naissance à de nombreuses variantes, dont `bash` (pour *Bourne-Again SHell*) qui est aujourd'hui le *shell* le plus répandu.

3. Observez l'invite de commande. Par défaut, **l'invite des distributions Linux montre le nom d'utilisateur et le nom de l'ordinateur séparés par @** (attention à ne pas confondre avec une adresse email), suivi du répertoire courant (ici « ~ » pour désigner le répertoire personnel de l'utilisateur).

   Déterminez votre nom d'utilisateur et d'ordinateur dans cette session.

Le même que celui de connexion au réseau de l'école.

Affichez la variable d'environnement `$USER` pour vérifier le nom d'utilisateur que vous avez trouvé

Explorer l'arborescence des fichiers : `pwd`, `ls`, `cd`

Les fichiers, répertoires[5] et sous-répertoires informatiques sont organisés dans une structure appelée une arborescence de fichiers qui part d'un répertoire appelé racine. Plusieurs commandes permettent de se déplacer dans l'arborescence et de visualiser les fichiers.

La commande **`pwd`** (pour « ***p**rint **w**orking **d**irectory* ») donne l'adresse du **répertoire courant** ou répertoire dans lequel on se trouve actuellement.

4. Exécutez la commande « `pwd` » et donnez l'adresse du répertoire courant.

/cygdrive/c/Users/Nom_utilisateur/Documents/Cygwin/home/

C'est votre r**épertoire personnel** associé à votre nom d'utilisateur par défaut.

La commande **`ls`** (pour ***list***) sert à **afficher le contenu d'un répertoire** (fichiers et répertoires).

5. Exécutez la commande « `ls` » sans paramètre pour lister ce que contient le répertoire courant. Que contient-il ?

books images python

6. Exécutez la commande « `ls images` » pour voir le contenu du répertoire `images`. Combien de fichiers y-a-t-il ?

15 fichiers jpg

On a souvent besoin d'ajouter des options à la commande `ls` pour obtenir plus d'information.

7. Exécutez la commande « `ls -a` » (`-a` pour ***all***) permet de lister également les fichiers et répertoires « cachés ». Quels sont les fichiers et répertoires cachés ?

. .. .bash_profile .bashrc .inputrc .profile

8. Observez que tous les fichiers cachés commencent par le même caractère. Quel est ce caractère ?

Un point « . »

Notez la présence de deux répertoires particuliers que l'on retrouve partout:

« **`.`** » désigne **le répertoire courant**, « **`..`** » représente l**e répertoire parent** situé au niveau juste au-dessus du répertoire courant dans l'arborescence.

4 Programmé par Stephen Bourne, d'où son nom.

5 On utilise indifféremment les termes « répertoire » et « dossier ».

9. Exécutez la commande « `ls -l` » (`-l` pour ***l**ong*) pour avoir une description plus complète (permissions, propriétaire, taille, date de modification, nom) des fichiers et répertoires.

Le premier caractère de chaque ligne indique le type de fichier s'il s'agit :
- « - » indique un fichier ordinaire;
- « d » indique un répertoire (`d` pour ***d**irectory)* ;
- « `l` » indique un raccourci (`l` pour ***l**ink)*.

On peut combiner les options `ls -a -l` en `ls -al`.

10. Exécutez la commande « `ls -al` » et entourez les 7 répertoires dans la liste ci-dessous.

```
drwxr-xr-x+ 1 Nom Nom 0    Jan 31 09:54  .
drwxr-xr-x+ 1 Nom Nom 0    Jan 31 09:54  ..
-rwxr-xr-x  1 Nom Nom 1494 Jan 17 19:51  .bash_profile
-rwxr-xr-x  1 Nom Nom 5645 Jan 17 19:51  .bashrc
-rwxr-xr-x  1 Nom Nom 1919 Jan 17 19:51  .inputrc
-rwxr-xr-x  1 Nom Nom 1236 Jan 17 19:51  .profile
drwxr-xr-x+ 1 Nom Nom 1236 Jan 17 19:51  books
drwxr-xr-x+ 1 Nom Nom 1236 Jan 17 19:51  images
-rwxr-xr-x  1 Nom Nom 1236 Jan 17 19:51  notes.txt
drwxr-xr-x+ 1 Nom Nom 1236 Jan 17 19:51  python
```

La plupart des commandes ont une option `--help` qui décrit leur utilisation. Par exemple pour `ls`, on peut utiliser `ls --help` .

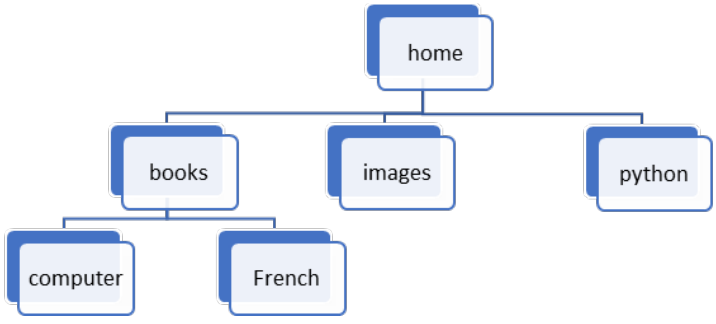
Aussi la commande `man` est très utile, en passant un nom de commande en argument (appuyer sur `q` pour quitter la commande `man`), par exemple on écrit `man ls`.

11. Testez ces deux commandes pour trouver quelle option permet de lister, triés par taille, les fichiers et répertoires cachés avec leur description ?

ls –a –l -S
ou ls –alS

La commande **`cd`** (pour « ***c**hange **d**irectory* ») permet de se **déplacer dans l'arborescence de répertoires** :
- `cd rep` : permet de se déplacer dans le répertoire `rep`.
- `cd ..` : permet de remonter d'un niveau dans le répertoire parent.
- `cd` ou `cd ~` : permettent de retourner dans son répertoire personnel, depuis n'importe quel répertoire.

12. Explorez l'arborescence complète des sous-répertoires et complétez le schéma ci-dessous.



13. Retournez dans votre répertoire personnel avec la commande `cd ~` ou `cd` .

Pour identifier un fichier ou un répertoire depuis le répertoire courant (.) on utilise un **chemin relatif**. Ce chemin n'est pas unique, **il dépend du répertoire** dans lequel on se trouve.

Par exemple, le chemin relatif de French depuis home est : `books/French` ou alors `./books/French`.

14. Allez directement dans French depuis home avec la commande « `cd books/French` ».

15. Écrivez une seule commande qui permet de se rendre directement depuis le répertoire `French` dans `computer`….

```
cd ../computer
```

16. … puis de se rendre depuis `computer` dans `python`.

```
cd ../../python
```

17. … puis de `python` dans `French`.

```
cd ../books/French
```

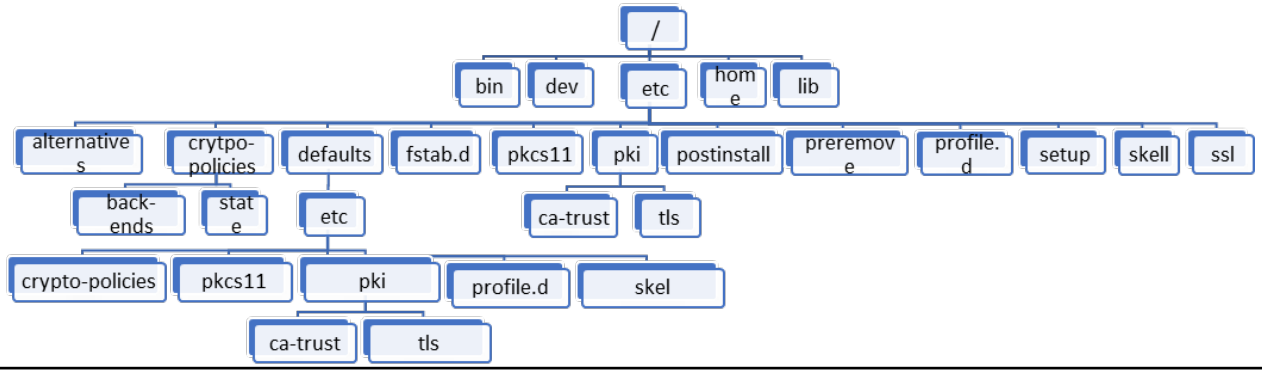
Un répertoire particulier se trouve tout au sommet de l'arborescence : le répertoire **`racine`** ou **`root ( / )`**.

- **`cd /`** : permet de se rendre directement dans le répertoire racine de l'arborescence de fichiers.

La racine contient plusieurs répertoires que l'on retrouve dans tous les systèmes Linux :

Répertoire	Signification (fr.)	Signification (ang.)	Contenu
/			Racine du système, hiérarchie primaire
/bin	utilitaires binaires	**bin**ary utilities	Commandes disponibles pour tous les utilisateurs (ex: cd, cat, ls…)
/dev	périphérique	**dev**ice	Fichiers spéciaux des périphériques
/etc	config. en mode texte	**e**diting **t**ext **c**onfig	Fichiers de configuration en mode texte de programmes et services
/home	maison	**home** directory	Répertoires personnels des utilisateurs
/lib	bibliothèques	**lib**rairies	Bibliothèques partagées essentielles et modules du noyau
/proc	processus	**proc**esses	Répertoire pour les processus système
/tmp	temporaire	**tmp**orary	Fichiers temporaires des applications
/usr	ressources système Unix	**U**nix **s**ystem **r**esources	Applications usuelles des utilisateurs et leurs fichiers
/var	variable	**var**iable	Données variables et diverses

Le schéma ci-dessous représente une partie de l'arborescence depuis la racine.



Pour identifier un fichier ou un répertoire depuis la racine (/ ou root), on utilise son **chemin absolu**. Ce chemin est unique, **il ne dépend pas du répertoire** dans lequel on se trouve.

18. Écrivez une seule commande qui permet de se rendre directement depuis n'importe quel répertoire dans pkcs11 en utilisant son chemin absolu.

```
cd /etc/defaults/etc/pkcs11
```

19. Retournez dans le répertoire `racine` puis dans `bin`. Ce répertoire contient un grand nombre de fichiers (les principales commandes du noyau Linux) et la commande `ls` est difficilement exploitable.

Pour filtrer plusieurs noms de fichiers ou répertoires, il faut utiliser des motifs :

- « * » remplace n'importe quelle suite de caractère, éventuellement vide : `ls p*` permet de lister tous les fichiers dont le nom commence par p.
- « `[0-9]` », « `[a-zA-Z]` », etc. permettent d'accepter tous les caractères d'un intervalle.

20. Écrivez une commande pour lister tous les fichiers dans le répertoire `bin`, 1/ dont le nom commence par la lettre b, 2/ dont le nom commence par p et finit par l, 3/ dont le nom contient au moins un chiffre.

```
1/ ls b*     2/ ls p*l     3/ ls [0-9]
```

Visualiser et chercher dans un fichier : `cat` et `grep`

21. Retournez dans le répertoire `books/French` et affichez le contenu du fichier `pg14287.txt` avec la commande « `cat pg14287.txt` ».
22. Recherchez maintenant le mot Verne dans le fichier pg14287.txt avec « `grep 'Verne' pg14287.txt` ». Combien de lignes contiennent Verne ?

4

La commande **`cat`** (pour « ***cat****enate* » ou concaténer) servait à l'origine à joindre deux fichiers ensembles mais est souvent utilisé pour **afficher le contenu** d'un seul fichier dans le terminal.

La commande `grep` sert à rechercher une chaîne de caractères dans un fichier.

Créer, copier, supprimer un fichier : `touch`, `mv` et `rm.`

23. Retournez dans votre répertoire personnel et créez deux nouveaux fichiers avec les commandes :
    « `touch fich1.txt` » puis « `echo 'ceci est un test' > fich2.txt` »
24. Affichez le contenu de ces deux fichiers à l'écran. Que contiennent-ils ?

fich1.txt est vide, fiche2..txt contient « ceci est un test »

25. Créez un nouveau répertoire avec la commande « `mkdir monrep` » et exécutez la commande « `ls -l` » pour vérifier que ce répertoire a été créé.
26. Déplacez le fichier `fich1.txt` dans le répertoire `monrep` avec la commande « `mv fich1.txt monrep` » et exécutez la commande « `ls -l` » pour vérifier que `fich1.txt` n'est plus dans `monrep`.
27. Copiez le fichier `fich2.txt` dans `monrep` avec la commande « `cp fich2.txt monrep` » et exécutez la commande « `ls -l` » pour vérifier que `fich2.txt` est toujours dans `monrep`.
28. Exécutez la commande « `ls monrep` ». Que contient le répertoire `monrep` ?

fich1.txt fich2.txt

29. Supprimez le fichier `fich2.txt` avec « `rm fich2.txt` » et vérifiez qu'il a bien disparu du répertoire courant.

Les commandes **`cp`**, **`mv`** et **`rm`** permettent de **copier**, **déplacer** ou **renommer** et **supprimer** un **fichier** :

- `cp fich1 fich2` : sert à **copier** le fichier `fich1` dans `fich2` (`cp` pour ***`copy`***).
- `mv fich1 rep` : sert à **déplacer** le fichier `fich1` dans un répertoire `rep` (`mv` pour ***`move`***).
- `mv fich1 fich2` : sert à **renommer** le fichier `fich1` en `fich2`.
- `rm fich` : sert à **supprimer** un fichier `fich` (`rm` pour ***`remove`***).

A noter : on peut utiliser les motifs de noms de fichier, par exemple « `rm *` » supprime tous les fichiers.

Créer et supprimer un répertoire : `mkdir` et `rmdir`

30. Supprimez le répertoire `monrep` avec la commande « `rmdir monrep` ». Quel message obtient-on ?

Un message d'erreur
```
rmdir: failed to remove 'monrep': Directory not empty
```

31. Proposez une suite d'instruction pour supprimer le répertoire monrep.

```
cd monrep
rm *
cd ..
rmdir monrep
```

Les commandes **`mkdir`** et **`rmdir`** permettent de **créer** et **supprimer** un **répertoire** :

- `mkdir rep` : sert à **créer** un répertoire `rep` (`mkdir` pour ***`make directory`***) .
- `rmdir rep` : sert à **supprimer** un répertoire `rep` (`rmdir` pour ***`remove directory`***). `rmdir` (sans option) **ne permet pas de supprimer un répertoire qui n'est pas vide**.

Gérer les droits : `id` et `chmod`

Pour définir des droits associés à un fichier ou à un répertoire[6], les systèmes d'exploitation de la famille Linux divisent les utilisateurs en trois catégories :

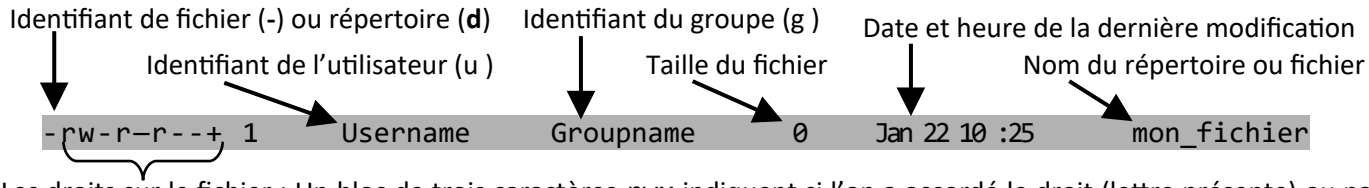
- L'utilisateur à qui appartient le fichier, désigné par « **u** » pour ***u**ser*.
- Le groupe d'utilisateurs d'un fichier, désigné par « **g** » pour ***g**roup*.
- Tous les autres utilisateurs, désignés par « **o** » pour ***o**thers*.

Tout utilisateur fait partie d'un ou plusieurs groupes. En fait, le système d'exploitation l'identifie par un identifiant (UID, ou User ID) ainsi que ses groupes (par leur GID pour Group ID).

32. Exécutez commande « `id` », afin de déterminez vos identifiants d'utilisateur et de groupe (UID et GID)

uid : u0_a117
guid : u0_a117

33. Retournez dans votre répertoire personnel et créez un fichier appelé `mon_fichier` puis exécutez la commande « `ls -l` » pour afficher ce que contient le répertoire courant. On obtient les informations suivantes :



Les droits sur le fichier : Un bloc de trois caractères `rwx` indiquent si l'on a accordé le droit (lettre présente) ou pas (`-`) pour chaque catégorie d'utilisateurs : l'utilisateur (u), puis son groupe (g), et les autres (o).

- Lecture (**r** pour *read*) qui autorise la lecture pour un fichier ordinaire ou la liste des fichiers d'un répertoire.
- Écriture (**w** pour *write*) qui permet la modification d'un fichier ou des fichiers contenus dans un répertoire.
- Exécution (**x** pour *execute*), qui indique si un programme peut être exécuté.

34. Détaillez les droits du fichier `mon_fichier`.

Lecture écriture pour l'utilisateur. Aucun droit pour le groupe et les autres utilisateurs

Le propriétaire d'un fichier peut en modifier les droits avec `chmod`[7].

35. Testez les commandes suivantes et écrivez le résultat obtenu (avec `ls -l`) puis déduisez ce qu'elles font :

- « `chmod a+x mon_fichier` »

`rwx–x--x` donne les droits en écriture et en exécution à tous les utilisateurs

- « `chmod ug=rwx,o=r mon_fichier` »

`rwxrwxr--` donne tous les droits à l'utilisateur et au groupe, mais seulement le droit en lecture aux autres .

- « chmod g-wx mon_fichier »

`rwxr–r--` enlève les droits en écriture et en exécution au groupe.

La commande **chmod** (pour ***ch**ange **mod**e*) permet de **modifier les droits associés à un fichier** ou à un **répertoire**.
• `chmod modifs fich` : sert à **modifier les droits** sur le fichier `fich` selon `modifs` composé de :
- une ou plusieurs lettres parmi « `u` », « `g` », « `o` » ou « `a` » (a pour ***a**ll*) : les utilisateurs concernés,
- « + » ou « - » : attribue des droits supplémentaires (+) ou supprime des droits (-),
- une ou plusieurs lettres parmi « `r` », « `w` » et « `x` » : les droits en lecture (r), écriture (w) et exécution (x).

36. Donnez la commande qui permet d'ôter les droits en écriture au groupe et aux autres utilisateurs de mon_fichier et vérifiez le résultat après exécution.

`chmod go-r mon_fichier`

6 Cygwin utilise le système de gestion de fichiers de Windows ce qui fonctionne avec un disque formatés en NTFS, mais pas FAT32 (souvent le cas des clés USB).
7 Ainsi que le « super-utilisateur » ou administrateur système, appelé root, qui a les pleins pouvoirs sur la machine.

Il existe **une autre manière de décrire** les droits à attribuer à un fichier en octal (base 8) en additionnant les droits associés respectivement à u, g et o en additionnant les valeurs :

- 4 pour r,
- 2 pour w,
- 1 pour x.

Par exemple, rwx vaut 7 (4 + 2 + 1), rx vaut 5 (4 + 1). On peut choisir directement des droits avec cette notation, mais pas en ajouter ou en ôter.

37. Indiquez quels droits sont attribués par « `chmod 754 mon_fichier` » puis vérifiez le résultat.

rwxr-xr--

38. Quelle commande permet d'attribuer les droits `rw-rw-r--` à `mon_fichier` ?

chmod 664 mon_fichier

Canaux d'entrée et sorties standards, redirections : >, >> et |

On a vu à la question 23 comment écrire un texte dans un fichier avec la commande « `'echo ceci est un test> fich2.txt'` ». De la même façon on peut envoyer la réponse à une commande dans un fichier plutôt que l'afficher à l'écran.

Toutes les commandes exécutées dans le shell supposent la présence d'entrées sorties appelées canaux :

- entrée standard ou canal 0, *stdin* (par défaut, la saisie au clavier)
- sortie standard, ou canal 1, *stdout* et
- sortie d'erreur, canal 2, *stderr*

Par défaut, le canal d'entrée standard est la saisie au clavier et les deux canaux de sortie se font par affichage dans le shell mais on peut aussi choisir de rediriger ces canaux dans des fichiers avec « `>` » et « `>>` ».

39. Exécutez la commande « `ls -al > sortie.txt` » pour créer un fichier `sortie.txt` avec la sortie standard (*stdout*) de la commande `ls`. Affichez le contenu du fichier pour vérifier le résultat.

Si le fichier `sortie.txt` existe, il est écrasé. Pour ne pas l'écraser, on utilise « `ls -al >> sortie.txt` ».

De la même façon, on peut associer la sortie d'erreur (*stderr*) dans un fichier.

40. Testez la commande « `ls z* > sortie.txt 2> erreur.txt` » et affichez le contenu de `sortie.txt` et de `erreur.txt`. Que contiennent-ils ?

sortie.txt est vide, et erreur.txt contient « ls : cannot access 'z*' : No such file or directory »

41. Écrivez une commande qui écrit la sortie de la commande `id` dans un fichier `monid.txt`.

id > monid.txt

L'outil « pipe », noté `|` , permet de rediriger la sortie d'une première commande vers l'entrée d'une seconde commande. En écrivant `cmd1 |cmd2` , la sortie de `cmd1` est redirigée en entrée de `cmd2`.

Par exemple en utilisant la commande « `wc -w text` » qui compte le nombre de mots dans `text`, on peut écrire « `ls | wc -w` » pour compter le nombre de fichiers dans un répertoire : la sortie de `ls` est une liste de fichiers qui est envoyée en entrée de `wc -w` pour être comptée.

42. En prenant pour exemple la commande « `cut -c2-8 text` » qui affiche les caractères 2 à 8 de `text`, écrire une commande qui écrit l'identifiant de utilisateur et l'écrit dans un fichier `monuid.txt` (exemple `10117`).

```
id | cut -c5-10 > monuid.txt
```

Variables et scripts

On peut utiliser des variables en leur affectant une valeur avec le signe « = » suivi de la valeur (sans espace). Ensuite, on précède le nom de la variable d'un $ pour l'utiliser. Le shell utilise aussi des variables, dites d'environnement. On a déjà utilisé à la question 2 la valeur de la variable SHELL avec « `echo $SHELL` ».

43. Créez une variable ex et affectez lui une valeur de votre choix. Puis affichez cette valeur de ex ?

```
ex=5
echo $ex
```

La puissance des commandes Linux est exploitable dans des scripts. Pour exécuter un fichier script.sh, il doit contenir une première ligne indiquant le shell à utiliser, par exemple «`#!/bin/bash`, et avoir des droits d'exécution.

44. Saisissez les commandes suivantes et déterminez ce que fait ce script.

```
echo –e '#!/bin/bash\necho Hello World!' > hi.sh
chmod +x hi.sh
./hi.sh
```

Les scripts Linux sont puissants, ils permettent d'écrire de vrais programmes avec des conditions, des boucles, etc.

Résumé des commandes

Commande	**Description**
cmd **>** *fich*	Écrit la sortie de commande *cmd* dans un fichier *fich*
cmd1 **|** *cmd2*	Renvoie la sortie de la commande *cmd1* sur l'entrée de *cmd2*
cat *fich*	Affiche le contenu du fichier *fich*
cd *rep*	Change de répertoire courant pour aller dans *rep*
cd ou **cd ~**	Change de répertoire courant pour aller dans le répertoire utilisateur
cd ..	Change de répertoire courant pour aller dans le répertoire parent
chmod *modifs fich*	Modifie les droits (rwx) des utilisateurs (u, g , o ou a) pour le fichier *fich*
cp *fich1 fich2*	Copie le fichier *fich1* dans *fich2*
echo *blabla* **>** *fich*	Crée un fichier *fich* contenant le texte *blabla*
grep *texte fich*	Recherche la chaîne de caractères *texte* dans *fich*
id	Affiche l'identifiant d'utilisateur (uid) et de groupes (gid) de l'utilisateur
ls	Affiche le contenu du répertoire courant (fichiers et répertoires)
ls *rep*	Affiche le contenu du répertoire *rep*
ls –a	Affiche le contenu du répertoire, y compris les fichiers et répertoires cachés
ls -l	Affiche le contenu du répertoire avec une description complète
man *cmd*	Affiche la documentation sur la commande *cmd*
mkdir *rep*	Crée un répertoire *rep* dans le répertoire courant
mv *fich rep*	Déplace le fichier *fich* dans le répertoire *rep*
mv *fich1 fich2*	Renomme le fichier *fich1* en *fich2*
pwd	Afficher l'emplacement du répertoire courant où l'on se situe actuellement
rm *fich*	Supprime le fichier *fich*
rmdir *rep*	Supprime le répertoire *rep* (s'il n'est pas vide)
touch *fich*	Crée un fichier *fich*