UAA8 - Operating System (OS)

Le gestionnaire de fichier sous linux

1 Bibliographie

- ⇨ Aide-mémoire commandes linux dans le drive
- ⇨ Voir le cours d'open class room « Reprenez le contrôle à l'aide de Linux »
- ⇨ Possibilité d'émuler en ligne un terminal linux (donc de tester les commandes linux dans un terminal sur internet) en rajoutant l'extension **xlinux** au navigateur firefox :

  https://addons.mozilla.org/fr/firefox/addon/xlinux-console-terminal

  Ou encore sur ce site :

  https://bellard.org/jslinux/vm.html?url=alpine-x86.cfg&mem=192

- ⇨ Possibilité d'installer par Windows Store **Debian WSL** (Windows Subsystem for Linux) qui est un terminal linux basé sur une distribution Debian :

  https://wiki.debian.org/InstallingDebianOn/Microsoft/Windows/SubsystemForLinux/
  https://people.montefiore.uliege.be/nvecoven/ci/files/tuto_bash/tuto_bash.html/
  https://geekmag.fr/blog/2020/02/23/windows-10-installer-wsl-2-et-deployer-debian-ubuntu-sans-microsoft-store/

  L'accès aux dossiers Debian se fait en tapant dans la barre d'adresse de l'explorateur windows : **\\wsl$\Debian**

- ⇨ Lire les articles suivants sur les bonnes pratiques de ton navigateur **firefox**

  https://lyc-84-bollene.gitlab.io/chambon/6-navigateur/4-navigateur
  https://www.laquadrature.net/2021/05/28/les-cookies-qui-sont-ils-que-veulent-ils

  - ✓ Configure la vie privée et la sécurité :
    - Protection renforcée contre le pistage => stricte
    - Envoyer aux sites web un signal « Ne pas me pister » => toujours
    - Ne jamais conserver l'historique de navigation
    - Bloquer les pop-up et Prévenir lorsque les sites essaient d'installer des modules complémentaires
  - ✓ Installe un bloqueur de pub **uBloc Origin** pour firefox
  - ✓ Utilise par défaut les moteurs de recherche **startpage** ou **DuckDuckGo**

2 Introduction aux OS

2.1 Qu'est-ce qu'un programme ?

Les programmes demandent à l'ordinateur d'effectuer des actions.
Votre ordinateur est rempli de programmes en tous genres :

- la calculatrice est un programme ;
- votre traitement de texte est un programme ;
- votre logiciel de « chat » est un programme ;
- les jeux vidéo sont des programmes.

En bref, les programmes sont partout et permettent de faire a priori tout et n'importe quoi sur un ordinateur

2.2 Logiciels libres et logiciels propriétaires

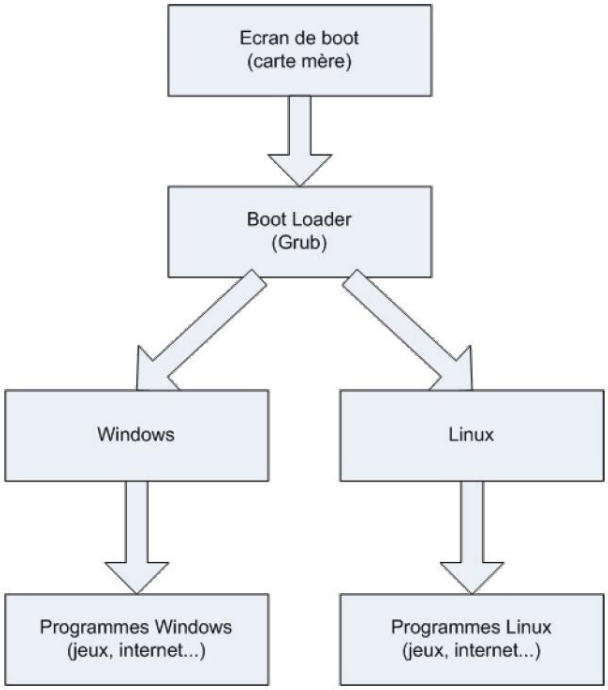
Il suffit d'avoir le binaire pour utiliser le programme ; on n'a pas besoin des sources du programme. Mais il n'existe pas de moyen de remonter aux sources complètes du programme à partir du seul binaire. Quand on achète un logiciel (Microsoft Office par exemple) ou un système d'exploitation (Windows par exemple), on a un CD qui contient le binaire, mais pas les sources. Il est donc impossible de savoir comment le programme est conçu. Par conséquent, on ne peut pas modifier le programme ; on peut seulement l'utiliser et éventuellement le copier à l'identique.
Les **logiciels propriétaires** sont les logiciels dont une licence, souvent payante, ne donne qu'un droit limité d'utilisation. On n'a la plupart du temps accès qu'à leur code binaires.
Certains logiciels propriétaires sont gratuits, on les appelle alors des **logiciels libres** ou **freewares**. Les logiciels libres sont les logiciels que l'on peut librement utiliser, échanger, étudier et redistribuer. Cela implique que l'on ait accès à leur code source (d'où le terme équivalent **Open Source**).

Vers 1983, la Free Software Foundation écrit le GNU Manifesto, dans lequel sont décrites les quatre libertés fondamentales que doit respecter un logiciel pour être qualifié de logiciel libre:

- la **liberté d'exécution** : tout le monde a le droit de lancer le programme, quel qu'en soit le but ;
- la **liberté de modification** : tout le monde a le droit d'étudier le programme et de le modifier, ce qui implique un accès au code source ;
- la **liberté de redistribution** : tout le monde a le droit de rediffuser le programme, gratuitement ou non ;
- la **liberté d'amélioration** : tout le monde a le droit de redistribuer une version modifiée du programme.

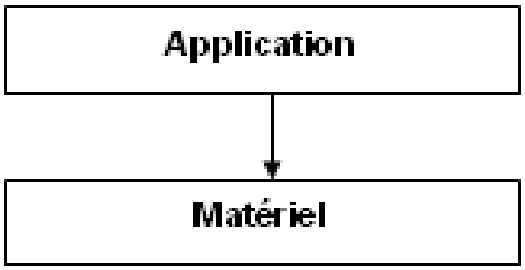
2.3 Qu'est-ce qu'un OS ?

Lors du démarrage de l'ordinateur, le programme de boot est exécuté avant le lancement Debian ou Windows. Ce programme dépend du matériel dont est constitué l'ordinateur.
Ce n'est seulement qu'une fois que Debian ou Windows est chargé que l'on peut enfin utiliser nos programmes : jeux, internet, logiciels de dessin, mail, musique …
En effet, l'ordinateur a besoin d'une sorte de « super logiciel » qui soit le chef d'orchestre. C'est lui qui doit gérer la mémoire de l'ordinateur, la répartir entre tous les programmes. Il fait **le lien entre le matériel (carte graphique, mémoire, imprimante), les logiciels, et l'utilisateur**.
Ce « super logiciel » s'appelle le **système d'exploitation** ou **Operating System (OS)**.
Windows et Linux sont donc des systèmes d'exploitation.
Si Plusieurs OS sont installés sur le PC, le **Boot Loader ou GRUB** s'intercale lors du démarrage de l'ordinateur et permet de choisir l'OS :



2.4 Fonctionnement général d'un OS

Sans système d'exploitation, chaque application (logiciel) "communique" directement avec le matériel :

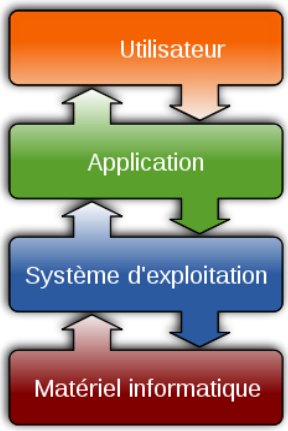


Mais pour un même type de matériel, il existe différents modèles.

Prenons l'exemple d'une imprimante. Une application devrait être en mesure de "communiquer" avec n'importe quelle imprimante. Il n'est pas possible pour les développeurs de prévoir toutes les imprimantes et encore moins celles à venir ! Une solution consisterait à acquérir le matériel (l'imprimante) qui est prévue par l'application et cela pour chaque application et chaque matériel ce qui deviendrait vite compliqué et onéreux.

Le système d'exploitation **s'interpose entre l'application et le matériel afin de faire le lien entre les deux** :

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 permet ainsi de "**dissocier**" les **programmes** et le **matériel**, afin notamment de **simplifier la gestion des ressources** et offrir à l'utilisateur une interface homme-machine (IHM) simplifiée.

2.5 Les différents OS

Les systèmes d'exploitation les plus couramment installés sur les ordinateurs actuels sont :

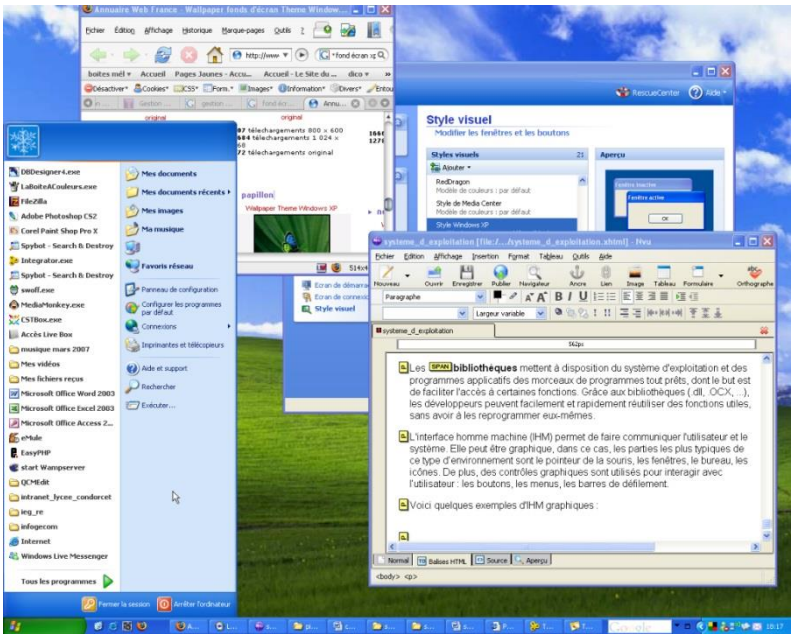
- ✓ **MS-DOS** (Microsoft Disk Operating System) : système en voie de disparition, exclusivement mono tâche, défini par un langage de commande. Il constituait la base des systèmes "Windows" de Microsoft jusqu'à Windows 3.1 inclus (c'est-à-dire que dans ces systèmes, les manipulations d'objets graphiques étaient en fait traduits en commandes MS-DOS) ;

  Taille des données manipulées : 16 bits

Interface MS-DOS

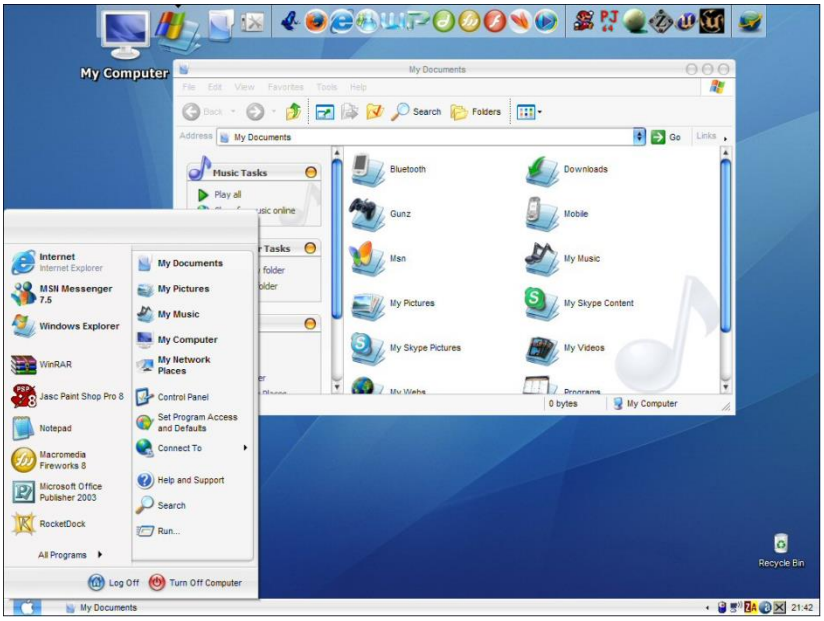
- ✓ **Windows 95, 98, XP, NT, 2000, 7, 10** : systèmes d'exploitation multitâches de Microsoft ayant pris la place de MS-DOS (la version NT est plus particulièrement destinée à la gestion des ordinateurs en réseaux)

  Taille des données manipulées par W95 -> 2000 : 32 bits
  Taille des données manipulées par WXP -> W10 : 32/64 bits



Windows XP

- ✓ la série des **MacOS** (Mac Intosh Operating System) équipe les MacIntosh d'Apple : ces systèmes ont introduit les premiers, dès 1984, les outils d'interface graphiques (menus, fenêtres...)
  Les systèmes actuels de Mac sont en fait des variantes du système **Linux**.



Mac

- ✓ **Linux** : version pour PC d'un célèbre système d'exploitation nommé **Unix**, multitâche et multi utilisateur, destiné initialement aux gros ordinateurs scientifiques, appelés aussi "stations de travail".
  Unix a été créé dans les années 60, est **codé en langage C** (crée à cette occasion). Ces deux "logiciels" : C et UNIX constituent le langage le plus important de l'histoire informatique (la majorité des langages actuels dérivent du C et la majorité des machines professionnelles tournent sur un dérivé d'UNIX).
- ✓ UNIX et ses dérivés sont présents partout : tous les smartphones fonctionnent sur un de ses dérivés (Linux pour **android**, bsd pour **iOS**) ainsi que les **mac**.
- ✓ Linux est constitué d'un langage de commande (appelé **Shell**) et sa particularité est d'avoir été écrit par des programmeurs bénévoles, qui le diffusent de manière **libre** (le code source est disponible) et gratuite.
  Il est associé à des environnements graphiques comme "Gnome" ou "KDE".
  On appelle "**distribution Linux**" l'ensemble constitué par une version de Linux, certains environnements graphiques et certains autres programmes nécessaires à son installation sur un PC.
  Exemples : Fedora, Mandriva, Debian et Ubuntu
  Debian est la seule distribution qui soit gérée par des développeurs indépendants plutôt que par une entreprise.

2.6 Accéder à une machine

On doit distinguer plusieurs manières d'accéder à une machine et de l'employer :

- Locale ou distante
- Graphique ou ligne de commande

Accès local

C'est la situation courante : vous êtes devant un ordinateur qui est raccordé à un moniteur et dispose de périphériques (clavier, souris) ; après l'avoir démarré vous avez, sur l'écran, son interface.

Accès distant

La machine distante (**serveur**) est raccordée à un réseau dont votre machine (**client**) fait partie.
Vous êtes devant le client et accédez au serveur par le **réseau**.
On emploie de nombreux protocoles pour y parvenir, parmi lesquels SSH, RDP et VNC.
À noter, il est possible d'utiliser un serveur dans le navigateur Chrome et d'y accéder par ce même navigateur sur le client.

Interface graphique

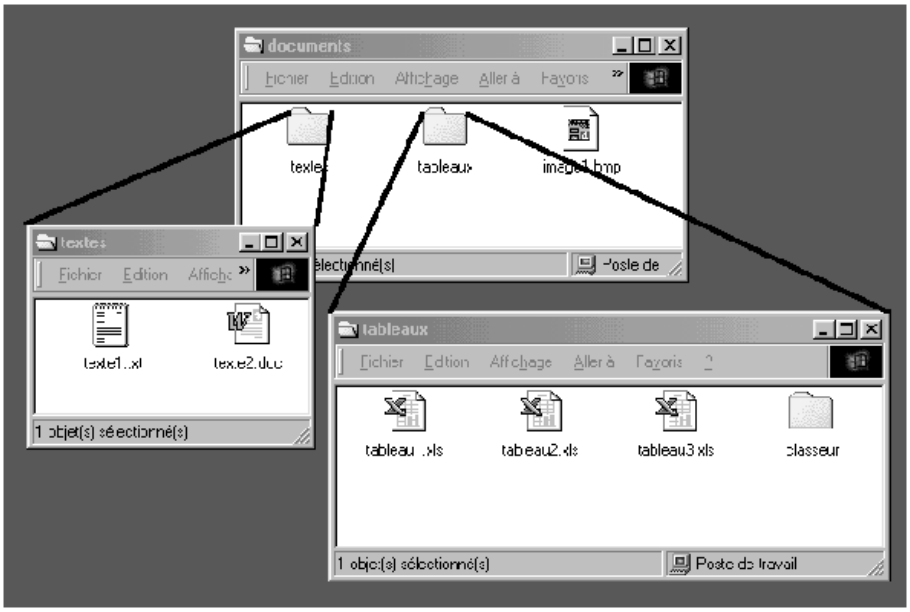
L'interface graphique notée **GUI (Graphical User Interface)** est celle à laquelle vous êtes habitué : fenêtre, menus, clic clic clic…

Interface en ligne de commande

L'interface en **ligne de commande** notée CLI (Command Line Interface) parfois appelée **shell, terminal** ou tty est l'interface courante d'un serveur distant : on tape des commandes, l'ordinateur évalue la réponse, l'affiche et on recommence.
Contrairement à ce qu'on pourrait croire, la plus puissante et pratique des manières d'utiliser un système UNIX est l'interface en **ligne de commande**.

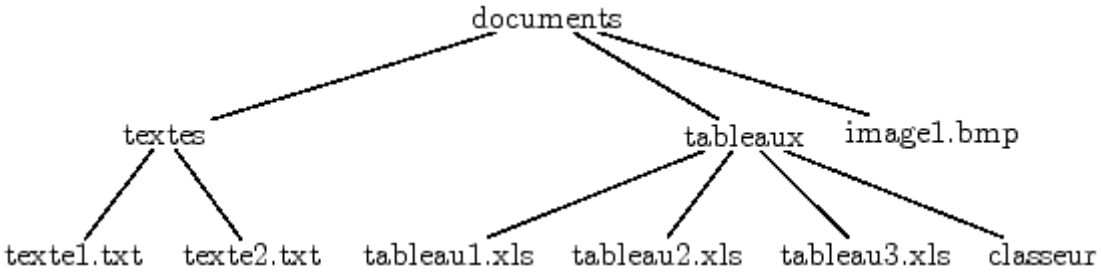
2.7 Le classement des répertoires (ou dossiers) et des fichiers

Les données stockées par les ordinateurs peuvent provenir de sources très variées : textes, formules mathématiques, images, etc., chacune correspondant à un mode de codage spécifique. Il ne saurait pourtant être question de stocker toutes ces données "en vrac" dans la (les) mémoire(s) des ordinateurs.
De même que pour classer rationnellement des documents papiers, on les range dans des pochettes et des classeurs, le système d'exploitation gère la mémoire disponible à l'aide de **fichiers** et de **répertoires** (on parle aussi de **dossiers**).
La figure suivante est une copie d'écran d'un environnement Windows où sont présents de tels composants (les traits symbolisant les liens entre un dossier et la fenêtre montrant son contenu ont été ajoutés).



- ✓ Un **fichier** sert à **stocker des données de même nature** (par exemple : caractères provenant d'un texte ou fichier son contenant la version numérisée d'une chanson). C'est une unité logique : un fichier ne correspond pas à un espace mémoire réservé une fois pour toute, il n'a pas de taille fixe prédéfinie et les données qu'il contient peuvent éventuellement ne pas être côte à côte en mémoire. Mais, pour l'utilisateur, la façon dont le système d'exploitation gère les fichiers est invisible (les informaticiens disent "transparente"). Tout est fait pour que l'utilisateur ait l'impression que les fichiers qu'il visualise se présentent comme des suites de données cohérentes.
  Dans les systèmes Windows, les fichiers reçoivent un nom qui se termine toujours par une extension de 3 lettres précédée d'un point. Ces 3 lettres permettent de repérer avec quel logiciel le fichier a été rempli : elles indiquent donc indirectement le mode de codage des données stockées dans ce fichier. Ainsi, un fichier ".txt" contient du texte, donc est codé par une succession de bits correspondant à des codes ASCII, un fichier ".exe" est un programme exécutable, codé sous formes d'instructions élémentaires, un fichier ".bmp" code une image bitmap ...
- ✓ Les **dossiers (ou répertoires)** sont plutôt à considérer comme des boîtes ou des classeurs : **ils ne contiennent pas directement des données, mais servent d'unités de rangement, pour recevoir soit des fichiers, soit d'autres dossiers** (ils peuvent aussi rester vides).
- ✓ Les fichiers et les dossiers sont structurés dans la mémoire de l'ordinateur de façon **arborescente**.

  Ainsi, par exemple, l'ensemble de fichiers et de dossiers qui apparaissent dans l'environnement Windows de la figure précédente correspond à l'organisation arborescente :



Attention : c'est un arbre avec le tronc ou la racine en haut et les feuilles en bas ....
Dans un tel arbre, les fichiers ne peuvent figurer qu'au niveau des feuilles (puisqu'eux-mêmes ne peuvent pas contenir d'autre fichier ou dossier). Les dossiers, eux, constituent les nœuds intermédiaires et n'apparaissent au niveau des feuilles que quand ils sont vides.

C'est le système d'exploitation qui gère toute cette organisation : il permet par exemple d'ajouter, de déplacer, de supprimer, de recopier... tout dossier ou fichier.

2.8 Exercice

Parmi les logiciels suivants :

- Lesquels sont des OS ? lesquels font partie intégrante d'un OS ? (entoure -les)
- lesquels sont des logiciels applicatifs indépendants de l'OS ?

Google chrome	Android	Windows 8
Linux	Internet Explorer	Microsoft word
Google drive	Debian	Firefox Mozilla
What's app	Discord	Windows explorer
Twitter	Task manager	Facebook
Start menu	Recycle bin	MSDos (invite de commande Microsoft)

Control Panel (gestion des utilisateurs, gestion des périphériques, installation de programmes)

3 Découverte de Linux

Consignes

L'objectif de ces TP est de te faire découvrir l'environnement Linux.
A la fin de chaque séance, tu dois absolument envoyer ton travail (même s'il n'est pas fini) sur le drive.
Les '**xx**' correspondent à tes **initiales** (Nom Prénom)
Les commandes les plus importantes sont décrites dans le mémo disponible dans le drive.
Tu ne dois pas retenir les commandes par cœur mais tu dois être capable de les retrouver grâce à l'aide dans le terminal, grâce à l'historique de commandes et grâce au mémo. (voir dans les tps)

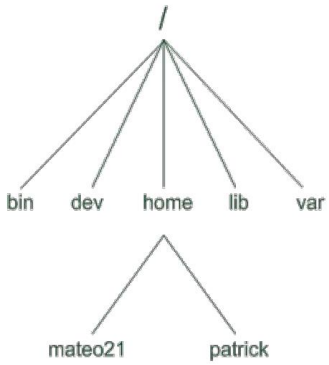
Système de fichiers Linux versus Windows

Sous Windows :

- Le système de fichiers est basé sur des lecteurs auxquels on accède via une lettre suivi de: (Exemple: c:)
- La racine est le 'backslash' \ (Exemple.: c:\ )
- L'arborescence est développée à partir de cette racine (Exemple.: c:\windows\system32)

Sous Linux

- La notion de lecteur n'existe pas
- Tout part du 'slash' **/** , c'est la **racine**
- Si plusieurs disques sont présents, ils sont intégrés au système de fichiers
- En ligne de commande, l'accès aux lecteurs externes se fait en montant une partition (Exemple: mount /dev/cdrom /temp pour accéder au lecteur cd-rom)



Les commandes shell

Un **shell Unix** est un **interpréteur de commandes** destiné aux systèmes d'exploitation Unix qui permet **d'accéder aux fonctionnalités internes du système d'exploitation**. Il se présente sous la forme d'une interface en ligne de commande accessible depuis la console ou un **terminal**. L'utilisateur lance des commandes sous forme d'une entrée texte exécutée ensuite par le shell. Dans les différents systèmes d'exploitation Microsoft Windows, le programme analogue est command.com, ou cmd.exe.
Le shell de linux utilise un interpréteur en ligne de commande appelé **bash**.
La syntaxe d'une commande bash se présente ainsi :

```
commande arguments
```

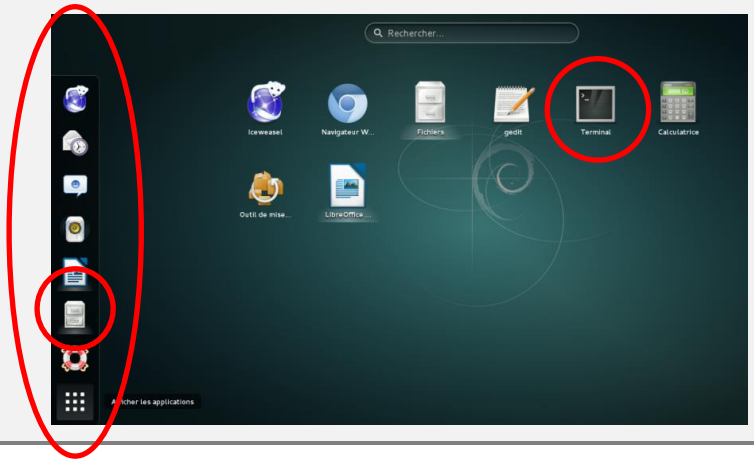
ou très souvent :

Commande -paramètres dossiers/fichiers

Le terminal

- ✓ Démarre le pc sous Debian, logue-toi sur le compte de ta classe.
- ✓ Ouvre un terminal (interpréteur de commande shell) ainsi que l'explorateur de fichier afin de vérifier ton travail tout le long des tp:

Activités -> Afficher les applications -> Terminal



- ✓ Change ton mot de passe si tu es connecté avec ton compte utilisateur et si tu le souhaites (note-le dans ton journal de classe ou dans une farde !!!)

passwd t5infoN

(N correspond à ta classe : i, j, k)

4 TP1 : Manipulation dossiers et fichiers

4.1 Test1 : Créer le dossier uaa8_linux pas à pas

✓ Par l'explorateur de fichier, va dans **public_html** puis ouvre un terminal

Souris -> clock droit -> ouvrir un terminal

✓ Vérifie dans quel dossier tu es (**pwd : Print Workind Directory**):

```
pwd
```

Résultat

```
/home/t5infoN/public_html
```

✓ Crée le dossier **uaa8_linux** (**mkdir : MaKe DIRectory**)

```
mkdir uaa8_linux
```

✓ Liste les éléments contenus dans le dossier et vérifie que tu as bien créé le dossier uaa11_linux (**ls : LiSte**)

```
ls
```

Résultat

```
uaa8_linux
```

✓ Entre dans le dossier tp1 (**cd : Change Directory**)

```
cd uaa8_linux
```

✓ Vérifie que tu es bien dans ton compte

```
pwd
```

Résultat

```
/home/t5infoN/public_html / uaa8_linux
```

✓ Vérifie qu'il n'y a aucun dossier ni fichier :

```
ls
```

Résultat

```
.
```

- ✓ Reviens dans le dossier parent (c'est-à-dire dans tp1 ; le « .. » correspond au **dossier parent**):

```
cd ..
```

- ✓ De la même façon, crée sous public_html les dossier suivants :

```
UAA7_BD_Sqlite, UAA11_Prog_Python, UAA12_Web
```

Historique de commandes

- ✓ Crée un fichier vide xx_tp1_readme.txt, vérifie toujours qu'il a bien été créé (ls), et vérifie qu'il est bien vide (« **more** » permet d'afficher le contenu d'an fichier dans le terminal)

```
touch xx_tp1_readme.txt
ls
more xx_tp1_readme.txt
```

- ✓ Affiche à l'écran l'historique des commandes tapées dans ce terminal

```
history
```

Résultat

```
Liste de toutes les commandes tapées ….
```

- ✓ Sauvegarde l'historique des commandes dans ton fichier **xx_tp1_readme.txt** (l'historique des commandes n'est plus affiché à l'écran mais est redirigé vers xx_tp1_readme.txt grâce au « > », voir le paragraphe plus loin sur les entrées /sorties)

```
history > xx_tp1_readme.txt
```

Attention « > » écrase le contenu de ton fichier

Vérifie que l'historique de commande est bien sauvegardé dans ton fichier xx_tp1_readme.txt.

Remarques

- ✓ **ls puis pwd** correspondent au double click souris dans l'explorateur de fichier.
- ✓ « **.** » correspond au **dossier courant** (où tu te trouves)
- ✓ « **..** » correspond au **dossier parent** (ce n'est pas la peine de préciser lequel puisqu'il y n'y en a toujours qu'un seul dans une structure en arborescence)
- ✓ « **/** » correspond au **dossier racine**

  « **/** » sert aussi de séparateur de dossier.

Exemple : /home/t5infoN/public_html / uaa8_linux

- ✓ la **flèche vers le haut** te permet de rappeler les anciennes commandes tapées.
- ✓ **Touche TAB**

  Quand tu le début d'une commande, tu peux faire afficher la fin automatiquement en appuyant une fois sur la touche tabulation.
  Si rien ne s'affiche, c'est qu'il y a plusieurs possibilités ou aucune ; dans ce cas, appuie deux fois sur tabulation pour avoir la liste des possibilités.
  De la même manière, quand tu connais le début d'une commande mais pas la fin, tu peux taper le début puis appuyer deux fois sur la touche Tabulation, il te donnera la liste des commandes existantes

Exemples :

- **« cd +TAB »** te permet de trouver les dossiers disponibles dans le dossier courant.
- **« cd xx+TAB »** te permet de trouver les éléments disponibles commençant par '1' (en l'occurrence, 'tp1 ')
  Combiné avec « **/** », il permet de se déplacer rapidement dans l'arborescence (à partir du dossier courant)

  cd /uaa11_prog/tp1

- ✓ **« commande --help »** te donne une aide sur l'utilisation de la commande tapée et des arguments disponibles.

  Exemple : find --help

- ✓ **Si l'aide n'est pas suffisante, « man commande » donne accès aux pages du manuel**
- ✓ **« clear »** permet d'effacer l'écran (du terminal)

4.2 Test2: Créer des dossiers et fichiers

- ✓ En t'aidant des commandes découvertes lors du test1, crée les dossiers et fichiers suivants :
  - Dans uaa8_linux, crée un dossier **tp1**, puis un sous-dossier **test2** (sous tp1),
  - dans test2, crée les dossiers d1, d2, d3,
  - dans d1, crée d4
  - dans d1, crée le fichier xx_f1.txt, dans d2, xx_f2.txt, dans d3, xx_f3 .txt, dans d4, f4.txt
  - dans test2, crée xx_f5.txt
- ✓ Affiche l'arborescence (commande tree) que tu viens de créer et vérifie qu'elle est correcte :

```
tree
```

Résultat

```
├── d1
│   ├── d4
│   │   └── d4.txt
│   └── xx_f1.txt
├── d2
│   └── xx_f2.txt
├── d3
│   └── xx_f3.txt
└── xx_f5.txt

4 directories, 5 files
```

- Crée un fichier **xx_test2_readme.txt** avec ton historique de commande
- Sauvegarde aussi ton arborescence **xx_test2_readme.txt**

Attention à ne pas écraser le contenu de ton fichier mais à rajouter ton arborescence à la fin de ton fichier. Pour cela, utilise « **>>** » au lieu de « **>** »

```
tree >> xx_tes2_readme.txt
```

4.3 Test3: Copier, déplacer des dossiers et fichiers

Crée un sous-dossier **test3** sous tp1 dans lequel tu travailleras. Par l'explorateur de fichiers, copies-y le contenu de test2.

Chemins relatifs et absolus

- ✓ Un **chemin relatif** est un chemin qui dépend du dossier dans lequel tu te trouves. C'est celui que tu viens d'utiliser.

  Par exemple, va dans d4 et affiche le contenu de xx_f2.txt dans d2. Pour cela, tu dois préciser les noms des dossiers que tu dois parcourir, séparés par des « / ».

- ⇨ Pense à utiliser la **Touche TAB** (voir page précédente) pour compléter automatiquement les noms de fichiers et dossiers et ainsi éviter des erreurs.

```
more  .. / .. /d2 / xx_f2.txt
```

- ✓ Contrairement aux chemins relatifs, les **chemins absolus** fonctionnent quel que soit le dossier dans lequel on se trouve. Un chemin absolu est facile à reconnaître : il commence toujours par la racine (/). Tu dois ensuite faire la liste des dossiers dans lesquels tu te trouves.

  Par exemple, pour afficher le contenu de xx_f2.txt d'où qu'on soit dans l'arborescence :

```
more  /home/t5infoN/public_html/uaa8_linux/test2/d2/xx_f2.txt
```

Copier, déplacer, renommer, effacer

- ✓ « **mv** » (mv MoVe).permet de **déplacer** un dossier, un fichier.

  La même commande permet de **renommer** un dossier, un fichier.

```
mv  départ  arrivée
```

- Dans test3, renomme d1 en d1_new (d1_new n'existant pas, d1 est directement renommé)

```
mv  d1  d1_new
```

- Déplace d1_new dans d2 (d2 existant, d1_new y est déplacé)

```
mv  d1_new  d2
```

- Déplace f3.txt dans d4

⇨ Attention, tu ne peux appeler que les fichiers et dossiers visibles à partir du dossier courant (listés par ls). S'ils sont ailleurs dans l'arborescence, tu dois préciser le chemin relatif pour y accéder. Pour cela, pense à utiliser la **Touche TAB** (voir page précédente) pour compléter automatiquement les noms de fichiers et dossiers et ainsi éviter des erreurs.

```
mv  d3/f3.txt  d1/d4
```

✓ **« cp »** (CoPy) permet de copier un fichier, un dossier.

Pour copier un dossier, rajoute l'argument **–R** (**Récursif**, la commande entre dans les dossiers pour tous les copier)

```
cp  fichier_départ  fichier_arrivée
cp – R dossier_départ dossier_arrivée
```

- Copie f1.txt dans d4

```
cd d1
cp f1.txt d4
```

- Copie d1_new d3 (en étant sous test3)

```
cp –R d2/d1_new d3
```

- **« rm »** permet d'effacer un fichier
  Pour effacer un dossier, rajoute l'argument **–R** (tous les dossiers et fichiers à l'intérieur du dossier sont aussi effacés)
  Attention à ne pas vous tromper, on ne vous demander aucune confirmation !

```
rm  fichier
rm –R dossier
```

Exercice

A l'aide de ces 3 commandes :

- copie test2 dans test2_copie, déplace-toi dans ce dernier
- déplace f1.txt dans d2
- efface f2.txt
- efface d3
- renomme d4 en d5
- renomme f5.txt en f6.txt
- sauvegarde ton historique de commande et ton arborescence dans **xx_test3_readme.txt**

4.4 Test4:Automatisation des commandes : création d'un bash

Le Bash (acronyme de **Bourne-Again shell**) est un interpréteur en ligne de commande de type script ; c'est-à-dire que ces commandes peuvent être lancées à partir d'un fichier script lui-même exécuté dans le terminal afin d'automatiser des tâches.

- ✓ Dans tp1, crée un dossier tes4 puis un fichier **test4.sh**
- ✓ Rajoute-lui les droits en exécution (les explications sur l'utilisation de l'instruction chmod seront données au tp2)

```
chmod +x test4.sh
```

- ✓ Edite-le et indique sur la 1ère ligne que le script doit être exécuté en bash

```
#!/bin/bash
```

- ✓ Récupère les commandes correspondant aux test2 dans xx_tp1_readme.txt puis ordonne-les afin que test4.sh refasse entièrement le test2 et recrée ainsi l'ensemble des dossiers et fichiers.
- ✓ **Attention, ne le teste pas n'importe où** !!!

  Vérifie bien que tu travailles dans ce nouveau dossier test4.
  Trouve l'argument pour effacer l'historique de commande.

```
man history
history  --help
```

- ✓ Lance ton script pour le tester

```
./test4.sh
```

- ✓ Vérifie que tous les dossiers et fichiers ont été créés et que tu as généré un nouveau fichier xx_tp1_readme.txt contenant les commandes exécutées par ton script et l'arborescence.

5 TP2 Organisation des dossiers Linux, les utilisateurs et les droits

5.1 Organisation des dossiers Linux

- ✓ Dans uaa8_linux, crée dossier **tp2** et dedans, un fichier **xx_tp2_readme.txt**
- ✓ Remonte jusqu'à la racine et liste les dossiers

```
cd /
ls
```

Résultat

bin	**dev**	**home**	**media**	**opt**	**root**	**sbin**	**sys**	**var**
boot	**etc**	**lib**	**mnt**	**proc**	**run**	**srv**	**usr**	

Ce sont les mêmes dossiers dans tous les systèmes UNIX (un serveur de la nasa, votre iPhone, cet ordinateur, un ordinateur de 1976, un réfrigérateur connecté, un ChromeCast etc.).

- **/root**: dossier de l'administrateur de l'ordinateur. C'est l'utilisateur **root (racine)** ou **super-utilisateurs**.
- **/home**: contient les dossiers des utilisateurs. Chaque compte crée sur la machine se voit attribuer un dossier dans /home.
  - ✓ Liste les dossiers utilisateurs et sauve-les dans le fichier xx_tp2_readme.txt
- /**boot** : contient les fichiers nécessaires au **démarrage** de la machine.
  - ✓ Recherche le grub dans boot et sauvegarde ton résultat de recherche dans xx_tp2_readme.txt.

    La recherche de fichier dans une arborescence de dossiers se fait avec la commande « **find** » ; n'hésite pas à consulter l'aide pour plus de détails.

    Attention à bien préciser le chemin d'accès à ton fichier readme après les « >> ».

```
find boot -name grub.cfg
```

- /**bin** (pour **BINaries**, **programmes compilés**) : contient les **commandes de base**.
  - ✓ Cherche l'emplacement des commandes ls, cp, cd grâce à la commande « **which** » et sauvegarde ton résultat dans xx_tp2_readme.txt.

```
which ls
```

- **/lib** (pour **LIBraries) :** contient les **bibliothèques partagées** essentielles. Il s'agit d'un ensemble de programmes, de fichiers qui peuvent être utilisés par plusieurs programmes. Plutôt que de les intégrer dans chaque programme qui s'en sert, on les range dans une librairie. Sous windows les librairies sont généralement des fichiers *.dll
- /**sbin** (pour **S**ystem **BIN**aries) : contient les programmes binaires permettant de lancer le système une fois le noyau lancé.

  Exemple : commande « shutdown » permet d'arrêter l'ordinateur
- **/usr :** contient les logiciels installés avec le système.
- /**dev** : pour **devices** (appareil). Contient les fichiers représentant les éléments matériels et les **périphériques** de la machine.

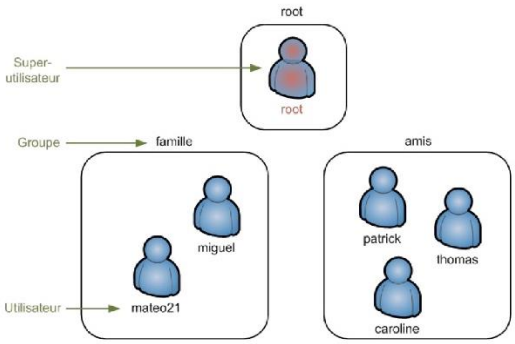
  Dans les systèmes UNIX chaque composant matériel est représenté par un fichier dans /dev. Par exemple, les disques de stockage sont dans /dev/sda, /dev/sdb etc. (**S**torage **D**evice). Si un disque a plusieurs partitions, elles sont numérotées : /dev/sda1 etc. C'est un fonctionnement très différent de ce qu'on rencontre sous windows où fichiers, matériels et services sont séparés.

  Pour simplifier : UNIX voit tous les éléments matériels comme des fichiers.
- **/etc** : contient les fichiers de **configurations** des éléments principaux et des **périphériques**.

  Exemple : le fichier /etc/passwd, définit les mots de passe des utilisateurs
- **/media** et /**mnt :** sont les répertoires utilisés pour monter temporairement un système de fichiers (**clé USB, DVD** …).
- /**tmp** : les fichiers temporaires.
- /**var** : un dossier pour les fichiers susceptibles de changer régulièrement, par exemple.
  Exemple : /var/log : les journaux enregistrant le déroulement des programmes

5.2 Les utilisateurs et les droits

On peut créer autant **d'utilisateurs** que l'on veut, eux-mêmes répartis dans des **groupes**.



✓ Va sous ton compte utilisateur, liste les dossiers

```
ls -l
```

Résultat pour public_html

```
drwxr-x---    …    t5infoi/j/L    www-data    ….    public_html
```

Propriétaires du dossier ou fichier

- La 3ème colonne correspond au **nom de l'utilisateur** propriétaire du dossier ou fichier. Ici, l'utilisateur t5infoi/j/L est propriétaire du dossier public_html.
- La 4ème colonne correspond au **nom du groupe** propriétaire du dossier ou fichier. Ici www-data est le nom de l'utilisateur du serveur apache.

  Si on ne précise rien, le nom de groupe sera celui de l'utilisateur

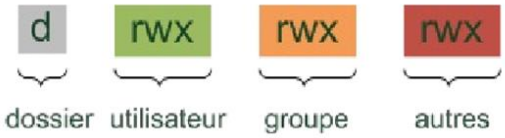
Droits sur le dossier ou le fichier

- « d » signifie qu'il s'agit d'un dossier, un fichier sera précisé par un « - »
- **rwx** donne les droits en lecture (**Read**), écriture ou modification (**Write**), exécution (**eXecute**) pour respectivement l'utilisateur, le groupe et les autres utilisateurs.

  Un « **-** » ne donne pas le droit, selon sa position, en lecture, écriture ou exécution.

  Un « x » pour un dossier signifie que l'on peut l'ouvrir.
- rwx est codé sur 3 bits : r, w, x correspondent à un bit à 1 (le droit existe), « - » à un bit à 0 (il n'y a aucun droit).
- Les droits sont découpés en fonction des utilisateurs et r, w, x sont répétés trois fois en fonction des utilisateurs :

- le premier triplet rwx indique les droits que possède le **propriétaire** du dossier sur ce dernier ;
- le second triplet rwx indique les droits que possèdent les autres membres du **groupe** sur ce dossier ;
- le dernier triplet rwx indique les droits que possèdent tous les **autres utilisateurs** de la machine sur le dossier.



- On peut ne donner aucun droit sur un dossier (--- ou 000) jusqu'à donner tous les droit (rwx ou 111). Les droits sont donc codés avec une base octale de 0 à 7.
- On peut aussi ne donner aucun droit sur un dossier ni à l'utilisateur, ni au groupe, ni aux autres utilisateurs. (--- --- --- ou 000 000 000) jusqu'à donner tous les droit à tous (rwx rwx rwx ou 111 111 1111), soit en octal de 0 à 777.

- Ici, t5infoi/j/L a droit de lecture, écriture, exécution (rwx = 7), www-data n'a pas le droit d'écriture (r-x = 5) et les autres utilisateurs n'ont pas accès au dossier (--- = 0).

  Les droits de ce dossier ont donc été définis avec la commande « **chmod** » suivante :

```
chmod 750 t5infoi/j/L
```

Manipulation du chmod

✓ Vérifie les droits de tp2

```
ls -l
```

Résultat

```
drwxr-xr-x … t5infoi/j/L www-data …. public_html
```

✓ Tu dois d'abord calculer quelle commande permet d'obtenir ces droits (drwxr-xr-x correspond à 755)

✓ Si tu veux être le seul à accéder à ce dossier, tu ne modifieras pas les droits pour t5infoi/j/L (rwx = 7), mais www-data et les autres utilisateurs n'auront aucun droit (--- = 0)

```
chmod 700 t5infoi/j/L
```

Résultat

```
drw------    …    t5infoi/j/L    www-data    ….    public_html
```

- ✓ Si au contraire tu souhaites que tout le monde puisse lire et ouvrir ce dossier (r-x = 5), soit tu appliques le code correspondant (755), soit tu rajoute à tous les utilisateurs les droits de lecture et d'exécution :

Solution 1

```
chmod  755  t5infoi/j/L
```

Solution 2

```
chmod  +r t5infoi/j/L
chmod  +w t5infoi/j/L
```

Résultat

```
Drwr-wr-w    …    t5infoi/j/L    www-data    ….    public_html
```

Exercice

- ✓ Va à la racine \ .

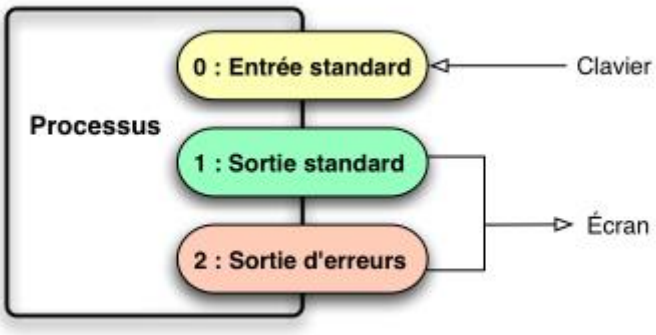
  Vérifie que tous les dossiers de la racine appartiennent à root.

  Vérifie que root a les droits rwx sur tous les dossiers et fichiers alors que tu n'as pas les droits en écritures r-x sur la majorité d'entre eux.

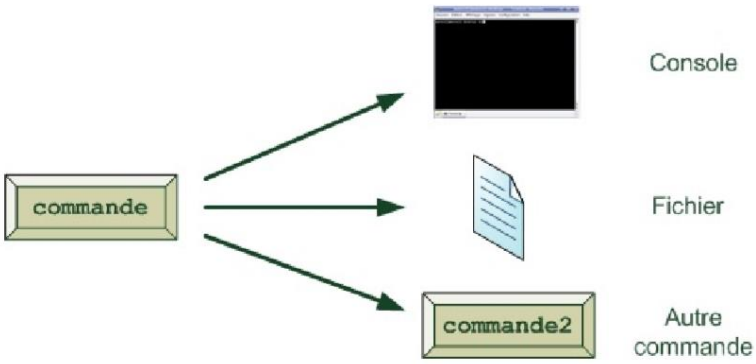
  Essaie de lire le contenu d'un fichier appartenant à root puis de le modifier. Est-ce cohérent ?

- ✓ Va dans tp1 ; pour www-data et les autres utilisateurs :
  - Change les droits de d1 en lecture seule
  - Interdis l'ouverture de d2
  - Change les droits de f5.txt en ---
- ✓ Log-toi au compte eleves (« **su eleve** », le password est « eleve ») dans une 2ème fenêtre
  - Lis f5.txt
  - Crée xx_f6.txt
  - Accède à d2
  - Accède à d1, puis d3
  - Dans d3, lis le contenu de xx_f3.txt et essaie de le renommer
- ✓ Liste les dossiers et fichiers que tu viens de modifier, ainsi que leurs droits d'accès dans **xx_tp2_readme.txt**
- ✓ Sauvegarde ton historique de commande dans **xx_tp2_readme.txt**

6 TP3 Entrées et sorties standard



Sous unix chaque processus dispose de trois "descripteurs de flux" : l'entrée standard, la sortie standard et la sortie d'erreurs.
L'entrée standard est en général le **clavier**, les **sorties standard** et **d'erreurs** sont en général **l'écran** mais pas forcément ! On peut rediriger les uns et les autres vers d'autres processus.
En combinant (parfois très astucieusement) ces entrées et sorties on conçoit des programmes complexes à partir de briques très simples.



Exemple

- ✓ Tu as déjà redirigé les sorties des commandes « history » dans les tests précédents :

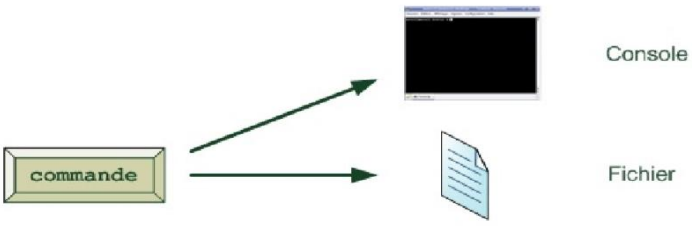
```
history > xx_tp1_readme.txt
more "hello" >> f5.txt
```

- ✓ Affiche la liste des fichiers de /usr/bin
  Comme il y en a beaucoup, affiche-les page par page grâce à la commande « **less** ».
  Less affiche le contenu d'un fichier, mais à la différence de more, less ne quitte pas la pagination.

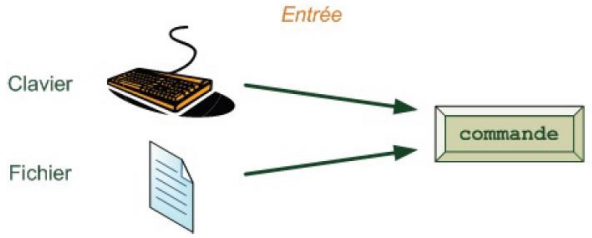
```
ls – l /usr/bin | less
```

Synthèse des redirections

- ✓ **commande > fichier** : redirige la sortie standard dans le fichier
  commande >> fichier : la même chose mais ajoute à la fin du fichier



- ✓ **commande < fichier** : redirige le fichier vers l'entrée standard



commande1 | commande2 **(« | » pipe)**: redirige la sortie standard de commande1 vers l'entrée standard de commande2, tout ce qui sort de command2 est envoyé à commande2. Autrement dit, **(« | » permet d'enchaîner plusieurs commandes**

« Chaîner des commandes » ? Cela signifie connecter la sortie d'une commande à l'entrée d'une autre commande (comme le montre la figure suivante).

Test1 : délimiteur cut

- ✓ Dans **uaa8_linux**, crée un dossier **tp3** dans lequel tu travailleras.
- ✓ Dans calc crée le fichier des notes du dernier contrôle :

Fabrice	18 / 20	Excellent travail
Mathieu	3 / 20	Nul comme d'hab
Sophie	14 / 20	En nette progression
Mélanie	9 / 20	Allez presque la moyenne !
Corentin	11 / 20	Pas mal mais peut mieux faire
Albert	20 / 20	Toujours parfait
Benoît	5 / 20	En grave chute

- ✓ Sauvegarde-le avec le format **CSV** (Comma Separated Values) : **xx_notes.csv**
  Ce sont des fichiers texte dont les valeurs sont séparées par des virgules pour faciliter l'échange et le traitement de données.

Ces virgules servent à séparer les colonnes, à savoir dans l'ordre : dans l'ordre : le prénom ; la note ; un commentaire.

- ✓ Affiche le contenu du fichier dans le terminal
- ✓ Affiche les notes et commentaires de Corentin
  Pour cela, utiliser la commande « **grep** » qui permet de chercher une occurrence dans un fichier

```
grep Corentin xx_notes.csv
```

Résultat

```
Corentin 11 / 20      Pas mal mais peut mieux faire
```

- ✓ Affiche à l'écran la liste des noms avec la commande « **cut** » et les paramètres :
  - -d : pour indiquer le délimiteur dans le fichier
  - -f : pour sélectionner le (ou les) numéro(s) du champ à couper

```
cut -d , -f 1 xx_notes.csv
```

Résultat

```
Fabrice
Mathieu
Sophie
Mélanie
Corentin
Albert
Benoît
```

- ✓ Affiche à l'écran le contenu de xx_notes.csv trié par ordre alphabétique.

```
sort xx_notes.csv
```

Résultat

```
Albert    20 / 20    Toujours parfait
Benoît    5 / 20     En grave chute
Corentin  11 / 20    Pas mal mais peut mieux faire
Fabrice   18 / 20    Excellent travail
Mathieu   3 / 20     Nul comme d'hab
Mélanie   9 / 20     Allez presque la moyenne !
Sophie    14 / 20    En nette progression
```

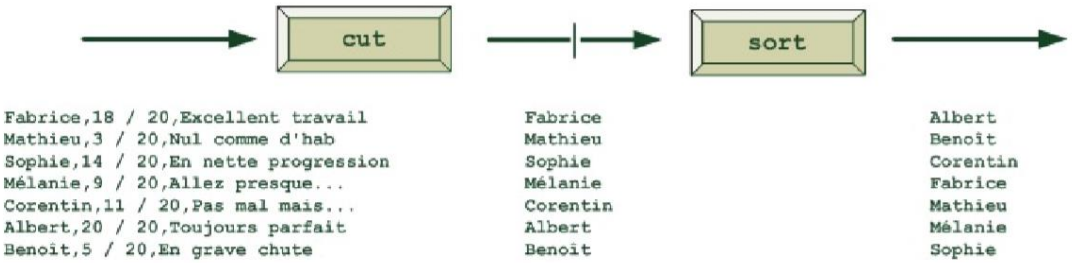
- ✓ Affiche la liste des noms et les notes, sans les commentaires.
- ✓ Crée un fichier **xx_notes2.csv** contenant la liste des noms et les notes

- ✓ Crée un fichier **xx_notes3.csv** contenant la liste des noms et les notes, suivie de la liste des noms et les appréciations.
- ✓ Affiche la liste des noms triés par ordre alphabétique puis sauvegarde-là dans un fichier **xx_notes4.csv**

```
cut -d , -f 1 xx_notes.csv | sort
```

Résultat

La pipe | effectue la connexion entre la **sortie de cut** (des noms dans le désordre) et **l'entrée de sort**:



Test2 : affiche les fichiers de ton compte par ordre décroissant, page par page

- ✓ Pour cela, utilise la commande « **du** », et regarde dans l'aide l'option **-nr** de la commande sort.
- ✓ Sauvegarde ton historique de commande dans **xx_tp3_readme.txt**