

Formation Linux et ligne de commandes

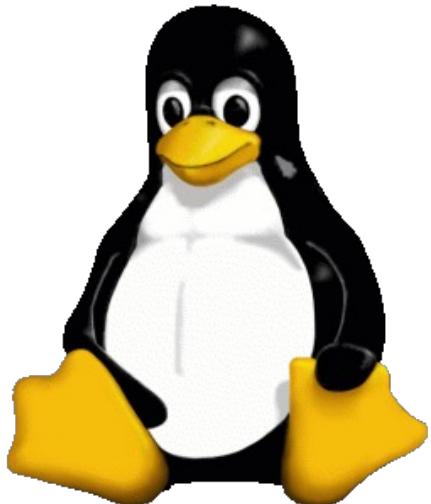
Ihab ABADI / UTOPIOS

Sommaire

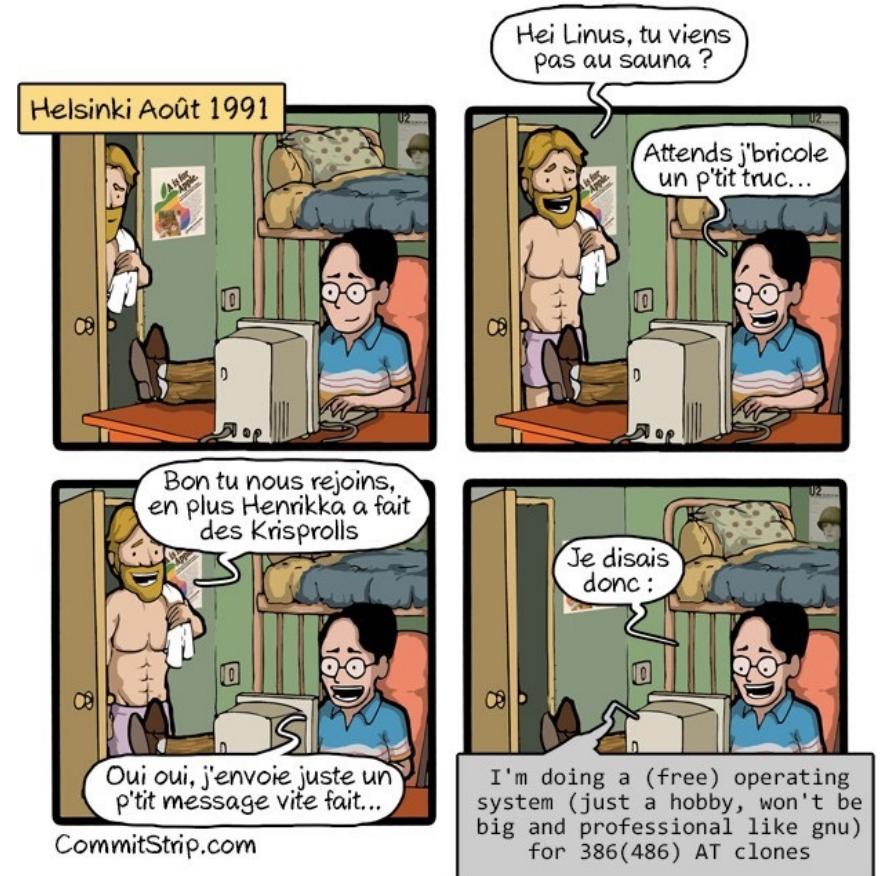
- Linux.
 - Principe et définition.
 - Rappel Distributions.
- Présentation des shells.
- Premières commandes.
- Vi et Vim.
- Commandes système de fichiers.
 - Gestion des fichiers
 - Organisation des fichiers
 - Droits d'accès.
 - Les noms des fichiers
 - Gestion de l'espace de stockage, compression et archivage
- Gestion des processus.
- Redirections et tubes
- Filtres

Linux – Principe

- Crée par Linus Torvalds, qui l'a annoncé publiquement le 25 août 1991.
- A commencé comme un *hobby*...



Ihab ABADI - UTOPIOS



Linux – Qu'est ce que c'est ?

- Linux est uniquement un noyau (*kernel*).
- Pour construire un système complet et utilisable, il faut ajouter :
 - Un chargeur de démarrage.
 - Un programme init.
 - Des bibliothèques.
 - Un environnement système.
 - Un environnement utilisateur.
 - Un environnement graphique.
- Etant donné que beaucoup de ces composants sont fournis par le projet GNU, on désigne souvent les systèmes les utilisant sous le nom GNU/Linux.
- Une *distribution* est un système complet noyau et composants.

Linux – Quelques distributions

- Il existe de très nombreuses distributions Linux.
- Les plus connues sont:
 - Arch Linux
 - CentOS
 - Debian
 - Fedora
 - Gentoo
 - Linux Mint
 - Mageia
 - openSUSE
 - Red Hat Enterprise Linux
 - Slackware
 - S U S E Linux Enterprise
 - Ubuntu

Linux – Installation d'un distribution

- Installation d'une distribution ubuntu, ou autre, sur machine virtuelle:
- En utilisant virtualBox.
- En utilisant Hyper.
- En utilisant wsl 2 Windows.

L'interpréteur de commandes (*shell*)

- L'interpréteur de commandes (***shell*** en anglais) est un logiciel fondamental pour travailler sur un système UNIX.
- En mode interactif, l'interpréteur de commandes fonctionne ainsi :
 - Il affiche une chaîne de caractères appelée invite, qui indique que l'interpréteur de commandes est prêt à accepter une nouvelle commande
 - Il permet à l'utilisateur de saisir au clavier une ligne de commande, jusqu'à ce que celui-ci appuie sur la touche Enter
 - Il analyse cette ligne et effectue certains traitements
 - Il exécute la commande
- En mode non interactif, l'interpréteur de commandes lit un fichier (qu'on appelle un ***script***) et exécute son contenu

L'interpréteur de commandes (*shell*)- Prompt

- L'*invite* (*prompt* en anglais) est une chaîne de caractères affichée par l'interpréteur de commandes lorsqu'il est prêt à accepter une nouvelle commande.
- Le contenu de l'invite est configurable et il est fréquent d'y faire figurer des informations telles que :
 - l'identifiant de l'utilisateur
 - le nom de la machine
 - le chemin d'accès du répertoire courant
- Dans les interpréteurs de commandes les plus anciens, l'invite se résumait à un unique caractère (qu'on retrouve encore à la fin de certaines invites aujourd'hui),
- qui pouvait être :
 - \$ pour les interpréteurs de commandes de la famille du Bourne shell
 - % pour les interpréteurs de commandes de la famille du C shell
 - # dans le cas où l'utilisateur est le super-utilisateur (root)

L'interpréteur de commandes (*shell*)- Saisie

- Lors de la saisie de ligne de commande, il est possible d'effectuer certains traitements :
 - ← → aller à gauche, à droite
 - ^A aller en début de ligne
 - ^E aller en fin de ligne
 - ^D supprimer le caractère sous le curseur
 - ^K supprimer la fin de la ligne

L'interpréteur de commandes (*shell*) – Caractères spéciaux

- Lors de l'analyse de la ligne saisie par l'utilisateur, l'interpréteur de commandes recherche des caractères spéciaux dans le but d'effectuer certains traitements.
- Ces caractères spéciaux sont :
 - l'espace
 - Les glyphes non alphanumériques du clavier
- Les caractères spéciaux ne peuvent donc être utilisés tels quels, par exemple dans un nom de fichier. Il faut auparavant les inhiber

L'interpréteur de commandes (*shell*) – Caractères spéciaux

- Il existe trois façons d'inhiber les caractères spéciaux :
 - faire précéder chacun d'eux d'une barre oblique inverse \
 - echo gl*u\\$b
 - gl*u\$b
 - délimiter l'expression contenant les caractères spéciaux par des apostrophes '
 - echo 'gl*u\$b'
 - gl*u\$b
 - délimiter l'expression contenant les caractères spéciaux par des guillemets "(fonctionne avec tous les caractères spéciaux sauf le dollar \$)
 - echo "gl*u#b"
 - gl*u#b

L'interpréteur de commandes (*shell*) – Commandes internes et externes

- Quasiment toutes les commandes sont des fichiers exécutables qui sont stockés quelque part dans l'arborescence du système de fichiers. Ce sont des **commandes externes**.
- Par opposition, il existe quelques **commandes internes**, qui sont directement traitées par l'interpréteur de commandes.
- Que se passe-t-il lorsqu'on exécute une commande ?
 - Si celle-ci est une commande interne, l'interpréteur de commandes reconnaît son nom et il la traite directement
 - sinon, il s'agit d'une commande externe :
 - si le nom de cette commande contient une barre oblique /, l'interpréteur de commandes essaie d'exécuter le fichier correspondant, considéré comme un chemin d'accès
 - sinon (c'est le cas général), l'interpréteur de commandes doit déterminer où se trouve le fichier exécutable de la commande dans l'arborescence du système de fichiers et, pour cela, il utilise la variable d'environnement PATH

L'interpréteur de commandes (*shell*) – Variable environnement Path

- Une **variable d'environnement** est un paramètre de configuration de l'interpréteur de commandes, dont la valeur est fixée par l'administrateur du système ou par l'utilisateur.
- On peut obtenir la valeur d'une variable en faisant précéder son nom par le caractère \$. Ainsi, on peut afficher le contenu d'une variable en utilisant la commande echo.
- La variable d'environnement PATH contient une liste de chemins d'accès absolus de répertoires séparés par des deux-points.
- Lorsqu'on exécute une commande externe dont le nom ne contient pas de barre oblique /, l'interpréteur de commandes va parcourir les répertoires de cette liste à la recherche de la commande et va exécuter la première qu'il trouve (l'ordre des répertoires a donc son importance) ou afficher un message d'erreur.

L'interpréteur de commandes (*shell*)

- Il existe six interpréteurs de commandes sous UNIX (classés ici dans l'ordre chronologique) :
- sh Bourne shell.
- Csh C shell.
- Tcsh TENEX C shell
- Bash Bourne-again shell
- Zsh Z shell

L'interpréteur de commandes (*shell*) – Premières commandes

- l'invite (*prompt*) : identifiant@ordinateur:~\$
- Dans ce qui suit, sauf cas particulier, afin de simplifier la forme des exemples, l'invite sera abrégée en : \$
- Taper la commande date : \$ date
- Taper la commande cal: \$ cal

L'interpréteur de commandes (*shell*) – Premières commandes

- Édition de la ligne de commande :
 - ← → aller à gauche, à droite
 - ^A aller en début de ligne
 - ^E aller en fin de ligne
 - ^D supprimer le caractère sous le curseur
 - ^K supprimer la fin de la ligne
- Historique:
 - ↑ commande précédente
 - ↓ commande suivante

L'interpréteur de commandes (*shell*) – Premières commandes - Syntaxe

- Syntaxe générale des commandes

```
$ commande [options] [arguments]
```

L'interpréteur de commandes (*shell*) – Premières commandes- ls

- La commande ls(*list segments*) permet d'afficher la liste de ses fichiers : **\$ ls**
- L'option -l(*long*) génère un affichage long : **\$ ls -l**
- L'option -a(*a ll*) affiche tous les fichiers : **\$ ls -a**
- Il est possible de combiner les options –a et –l **\$ ls -a -l** | **\$ ls -al**
- Pour afficher les informations concernant un fichier en particulier, il faut indiquer son nom en argument : **\$ ls -l toto.txt**

L'interpréteur de commandes (*shell*) – Premières commandes- cat

- La commande **cat**(*catenate*) permet d'afficher le contenu du ou des fichiers texte passés en arguments

```
$ cat tata.txt
```

L'interpréteur de commandes (*shell*) – Premières commandes- more

- La commande more permet d'afficher le contenu du ou des fichiers texte passés en arguments en contrôlant leur défilement :

```
$ more tata.txt
```

- Pour contrôler le défilement :
 - Enter descendre d'une ligne
 - b remonter d'une page
 - q quitter

L'interpréteur de commandes (*shell*) – Premières commandes- man

- La commande `man`(*manUal*) permet d'afficher le manuel de la commande en argument :

```
$ man ls
```

Commandes

EXERCICE

- Unix possède un manuel « en ligne ». La commande man permet d'explorer ce manuel.
 - 1) Quelle documentation contient ce manuel ? Comment est-il structuré ?
 - 2) La commande write porte le même nom que l'appel système write. Comment accède-t-on à la page du manuel concernant l'appel système write ?
 - 3) Commande ls : précisez les options que vous savez utiliser et celles que vous pourriez éventuellement utiliser

Vi et vim

- Quelques éditeurs de texte:
 - Emacs (*Editingmacros*) est le premier logiciel à avoir été distribué par le projet GNU
 - nano est un éditeur de texte très (trop) simple, inspiré de l'éditeur de texte Pico (d'où son nom)
 - vi(*visual*) est le plus ancien éditeur de texte plein écran sous UNIX (1976), d'approche difficile pour les débutants mais d'une puissance extraordinaire
 - Vim (*Vi IMproved*) est une variante de viavec de nombreuses améliorations

Vi

- Signifie : *visual*.
- Se prononce lettre par lettre à l'américaine : «vie ail».
- Le plus ancien éditeur de texte plein écran sous UNIX.
- Conçu par Bill Joy en 1976.
- D'approche difficile pour les débutants mais d'une puissance extraordinaire
- Lancement de vi

```
$ vi toto.txt
```

Vi- Passage en mode insertion, en mode commande

- i passage en mode insertion au caractère situé sous le curseur
- a passage en mode insertion au caractère situé après le curseur
- I passage en mode insertion en début de ligne
- A passage en mode insertion en fin de ligne
- o passage en mode insertion au-dessous de la ligne actuelle
- O passage en mode insertion au-dessus de la ligne actuelle
- ESC passage en mode commande

Vi

EXERCICE

- Saisir trois lignes de texte (du vrai texte, avec des mots, des espaces et de la ponctuation).
- Tester sur ce texte les commandes qui vont être abordées

Vi- Déplacement

- h gauche
- ↓ j bas
- ↑ k haut
- → l droite
- 0 aller en début de ligne
- \$ aller en fin de ligne
- w b W Bse déplacer de mot en mot
- { } se déplacer de paragraphe en paragraphe
- G aller sur la dernière ligne
- :n nG aller sur la ligne numéro n

Vi- Suppression, remplacement

- x supprimer le caractère sous le curseur
- d supprimer la fin du mot à partir du curseur
- W supprimer la fin de la ligne à partir du curseur Dsupprimer toute la ligne
- dd r<x> remplacer le caractère sous le curseur par <x>
- cw remplacer la fin du mot à partir du curseur
- C remplacer la fin de la ligne à partir du curseur
- cc remplacer toute la ligne
- U annuler la modification précédente
- Esc annuler la commande en cours

Vi- Copier-coller, couper-coller

- [n]yy copier
- [n]dd couper
- p coller
- J joindre la ligne suivante à la ligne courante

Vi- Recherche, remplacement

- / recherche (vers le bas)
- ? recherche (vers le haut)
- n occurrence suivante dans le même sens
- N occurrence suivante en sens inverse
- :s recherche et remplacement

Vi- Fichier

- :w sauvegarder le fichier (s'il a déjà un nom)
- :w <fichier> sauvegarder le fichier sous le n o m indiqué
- :q quitter
- :wq :x ZZ sauvegarder le fichier et quitter
- :q! quitter sans sauvegarder le fichier

Gestion de fichier- Le complètement

- Le complètement (*completion*) permet de ne saisir que le début des noms de fichiers et de laisser l'interpréteur de commandes compléter la suite.
- Il suffit de taper le début du nom de fichier et d'appuyer sur la touche tabulation
- ---→ :
 - s'il n'existe qu'un seul fichier dont le nom commence par ce qu'on a tapé, l'interpréteur de commandes complète le nom du fichier, le fait suivre d'une espace (au cas où l'on souhaite taper ensuite autre chose) et positionne le curseur à la suite
 - s'il existe plusieurs fichiers dont le nom commence par ce qu'on a tapé, l'interpréteur de commandes va compléter ce qu'il peut, positionner le curseur juste après, sans espace, puis laisser le soin à l'utilisateur de poursuivre manuellement

Gestion de fichier – La commande CP

- La commande cp(*copy*) permet de créer une copie d'un fichier

```
$ cp toto.txt tata.txt
```

- La commande cp n'affiche rien si elle se déroule correctement. C'est tout à fait normal et c'est le cas de beaucoup d'autres commandes, qui sont destinées à réaliser une fonction précise, pas à afficher quelque chose.
- L'option *-i* demande confirmation en cas d'écrasement :

```
$ cp -i toto.txt tata.txt  
cp: voulez-vous écraser « tata.txt » ?
```

- Si le fichier d'origine n'existe pas, un message d'erreur est affiché :

```
$ cp tutu.txt tata.txt  
cp: impossible d'évaluer « tutu.txt »: Aucun fichier ou dossier de ce type
```

Gestion de fichier – La commande MV

- La commande mv(*move*) permet de renommer un fichier :

```
$ mv toto.txt tata.txt
```

- L'option –i demande confirmation en cas d'écrasement :

```
$ mv-i toto.txt tata.txt  
mv: voulez-vous écrasert« tata.txt » ?
```

Gestion de fichier – La commande RM

- La commande rm(*remove*) permet de supprimer un ou plusieurs fichiers :

```
$rm toto.txt
```

- L'option –i demande confirmation avant la suppression de chaque fichier :

```
$ rm-i toto1.txt  
rm : supprimer fichier « toto1.txt » ? y
```

Commandes

EXERCICE

- 1) Quelle différence y a-t-il entre les commandes mv toto titi et cp toto titi ?
- 2) Copier les fichiers titi1 et toto4 dans le répertoire /tmp en une seule commande.

Gestion de fichier- Structure des fichiers

- Un fichier est composé de trois éléments :
 - un ou plusieurs noms
 - un i-nœud (*inode*)
 - son contenu proprement dit
- On peut faire afficher le numéro d'i-nœud associé à un fichier au moyen de l'option –i de la commande ls

```
$ ls -i  
68782057 toto.txt
```

```
$ ls -il  
Total 4  
68782057-rw-r--r-- 1 g u 334 juil. 30 17:02 toto.txt
```

Gestion de fichier – La commande ln

- La commande `ln(link)` permet de créer un lien supplémentaire sur un fichier :

```
$ ln toto.txt tata.txt
```

- L'option `-s` de la commande `ln` permet de créer un lien symbolique

```
$ ln -s toto.txt tata.txt
```

Organisation de fichiers – La commande mkdir

- La commande `mkdir`(*make directories*) permet de créer un ou plusieurs répertoires : `$ mkdir rep`
- Affichage du contenu du répertoire : `$ ll rep`
- Affichage des informations du répertoire lui-même
`$ ll -d rep`

Organisation de fichiers – La commande cp et les répertoires

- Copie du fichier toto.txt dans le répertoire rep

```
$ cp toto.txt rep
```

- Copie du fichier toto.txt dans le répertoire rep sous le nom tata.txt

```
$ cp toto.txt rep/tata.txt
```

- Copie d'un ensemble de fichiers dans le répertoire rep

```
$ cp fichier1 fichier2 fichier3 rep
```

Organisation de fichiers – La commande cd

- La commande `cd(change directOry)` permet de se déplacer dans un répertoire :

```
$ cd rep
```

- Le répertoire rep est maintenant le *répertoire courant (working directory)*, c'est-à-dire celui dans lequel on se trouve et où l'on peut désigner les fichiers sans les préfixer.
- La commande cd modifie l'invite de l'interpréteur de commandes.
- La partie de l'invite se situant entre :et \$ contient le chemin d'accès absolu du répertoire courant.
- La commande `pwd(print working directory)` affiche le chemin d'accès absolu du répertoire courant

Organisation de fichiers – Chemin d'accès absolu et chemin d'accès relatif

- Un chemin d'accès absolu est un chemin d'accès qui part du répertoire racine du système de fichiers et parcourt celui-ci pour aboutir au fichier à désigner.
- Un chemin d'accès absolu commence donc par /.
- Par exemple, /home/toto/toto est un chemin d'accès absolu.
- Un chemin d'accès relatif est un chemin d'accès qui ne part pas du répertoire racine du système de fichiers. Un chemin d'accès relatif ne commence donc pas par /.
- Par exemple, rep/toto est un chemin d'accès relatif.
- Un chemin d'accès absolu est non ambigu, il désigne toujours le même fichier, indépendamment de l'endroit où l'on se trouve. Un chemin d'accès relatif, en revanche, dépend du répertoire courant.

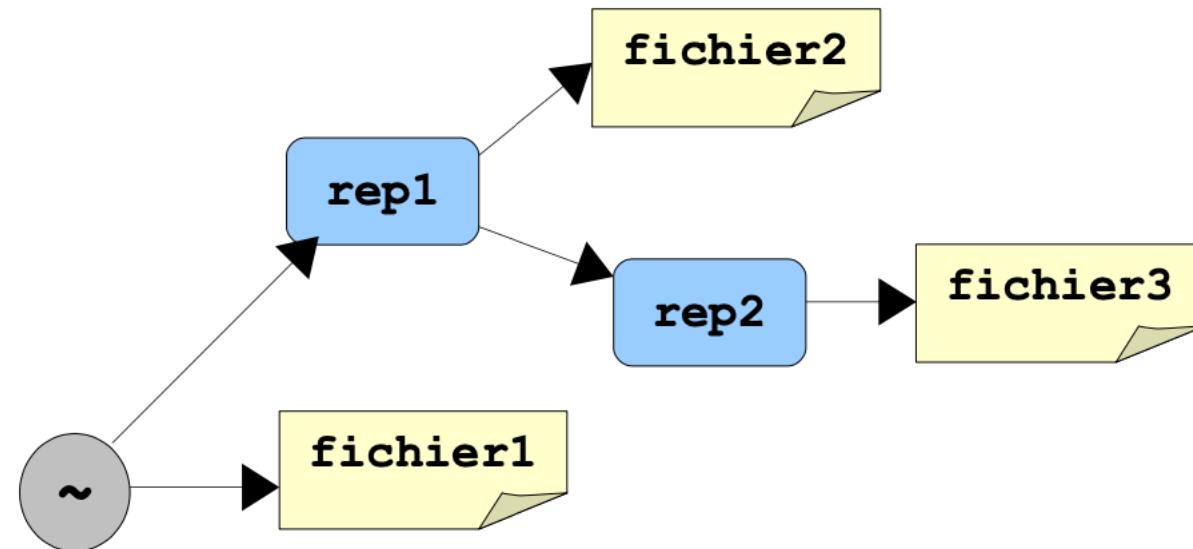
Organisation de fichiers – Suppression de répertoires

- La commande rmdir(remove directories) permet de supprimer un ou plusieurs répertoires vides :
 \$ rmdir rep
 \$ rmdir rep1 rep2 rep3
- Si le répertoire à supprimer n'est pas vide, on obtient un message d'erreur :
 \$ rmdir rep
 rmdir: échec des suppression de « rep »: Le dossier n'est pas vide
- La commande rm, utilisée avec l'option -r, permet de supprimer un ou plusieurs répertoires et tout leur contenu :
 \$ rm -r rep
 \$ rm -r rep1 rep2 rep3

Commandes

EXERCICE

- Sans bouger du répertoire racine (celui qui est à la base de l'arborescence ; il s'agit ici de ~), créez l'arborescence suivante :



Commandes

EXERCICE

- On désire aller dans le répertoire `/usr/local/games/mariokart`, et le répertoire courant est `/usr/local`. Quelle(s) commande(s) peut-on taper ?
- **A** : `cd /games/mariokart`
- **B** : `cd games/mariokart`
- **C** : `cd local/mariokart`
- **D** : `cd /usr/local/games/mariokart`
- **E** : `cd /usr/local/../../local/games/mariokart`
- **F** : `cd ../games/mariokart`

Les métacaractères de génération de noms de fichiers

- Les métacaractères de génération de noms de fichiers permettent d'exprimer un ensemble de noms de fichiers ayant des parties communes de la manière la plus compacte possible.
- Les métacaractères sont gérés par l'interpréteur de commandes, après validation et avant exécution.
- Le métacaractère ? correspond à exactement un caractère quelconque.
- Plus précis que ?, une expression entre crochets correspond également à exactement un caractère mais uniquement parmi ceux indiqués entre les crochets.

Les métacaractères de génération de noms de fichiers

- Une expression entre crochets peut contenir une liste de caractères, un intervalle de caractères de même nature ou une combinaison de ces deux syntaxes .
- Une expression entre crochets peut commencer par ! ou ^, ce qui inverse sa signification dans ce cas se sont les caractères ne faisant pas partie du reste de l'expression qui sont pris en compte.
- Le métacaractère *correspond à Zéro, un ou plusieurs caractères quelconques.

Commandes

EXERCICE

- Lister tous les fichiers :
 - se terminant par '5',
 - commençant par 'annee4',
 - commençant par 'annee4' et de 7 lettres maximum,
 - commençant par 'annee' avec aucun chiffre numérique,
 - contenant la chaîne 'ana',
 - commençant par 'a' ou 'A'

Droits d'accès

- Le droit d'accès *setUid* (aussi appelé *sUid* ou *SUID*), affecté à un fichier ordinaire exécutable, attribue à quiconque exécute ce fichier l'identité de son propriétaire, dans le contexte de son exécution :
`ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 027832 Jun 10 2022 /usr/bin/passwd`
- Le droit d'accès *setUid* est sans effet sur les répertoires.
- Le droit d'accès *setgid* (aussi appelé *sgid* ou *SGID*), affecté à un fichier ordinaire exécutable, attribue à quiconque exécute ce fichier le groupe de son propriétaire, dans le contexte de son exécution.
`ls -l /usr/bin/wall
-r-xr-sr-x. 1 root tty 15344 Jun 10 2014 /usr/bin/wall`

Droits d'accès

- Le droit d'accès *setgid*, affecté à un répertoire, fait en sorte que le groupe de ce répertoire soit hérité par les fichiers et répertoires qui sont créés à l'intérieur

```
$ id  
uid=5133(utilisateur) gid=1664(groupe) groups=1664(groupe),3351(travail) $ mkdir rep  
$ ls -ld rep  
drwxr-xr-x. 2 utilisateur groupe 6 Oct 4 10:57 rep  
$ chgrp travail rep  
$ ls -ld rep  
drwxr-xr-x. 2 utilisateur travail 6 Oct 4 10:57 rep  
$ touch rep/toto  
$ ls -l rep/toto  
-rw-r--r--. 1 utilisateur groupe 0 Oct 4 10:58 rep/toto  
$ rmrep/toto  
$ chmod2755 rep  
$ ls -ld rep  
drwxr-sr-x. 2 utilisateur travail 6 Oct 4 10:59 rep  
$ touch rep/tata  
$ ls -l rep/tata  
-rw-r--r--. 1 utilisateur travail 0 Oct 4 10:59 rep/tata
```

Droits d'accès - *Sticky bit*

- Le *sticky bit*, affecté à un répertoire accessible en écriture par tous les utilisateurs (comme /tmp et /var/tmp), limite la suppression et le renommage des fichiers et répertoires qu'il contient à leur seul propriétaire.
- Historiquement, le *sticky bit*, affecté à un fichier ordinaire exécutable au bon vouloir de l'administrateur système, avait pour effet de conserver son code dans la Zone d'échange (*swap*) après son exécution pour en permettre un relancement plus rapide. Depuis que les systèmes d'exploitation conservent en mémoire cache les données utilisées récemment, ceci n'a plus lieu d'être et le *sticky bit* n'a plus aucun effet sur les fichiers ordinaires.

Droits d'accès - Représentation numérique des droits d'accès

- Comment représenter des droits d'accès sous forme numérique plutôt que sous la forme d'une chaîne de caractères telle que rw-r--r-- ?
- La représentation des droits d'accès sous la forme d'une chaîne de caractères a deux caractéristiques remarquables :
 - chaque droit d'accès a une position bien précise;
 - chaque droit d'accès n'a que deux états possibles, accordé ou refusé.
- On peut donc envisager de représenter des droits d'accès sous la forme d'un nombre binaire comprenant neuf chiffres, un par droit d'accès, chaque chiffre correspondant au droit situé à la même position dans la représentation des droits d'accès sous la forme d'une chaîne de caractères et, pour chaque chiffre, 1 représentant l'accord d'un droit et 0 son refus.
- Ainsi, rw-r--r--peut être représenté par 110 10 0 10 0

Droits d'accès - Représentation numérique des droits d'accès

- Par définition, le nombre binaire 110100100 se décompose en puissances de 2 ainsi :
- $110\ 10\ 0\ 10\ 0 = 1 \times 2^8 + 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
 - soit, en factorisant la puissance de 2 la plus petite de chaque ligne :
 $110\ 10\ 0\ 10\ 0 = (1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0) \times 2^6 + (1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0) \times 2^3 + (1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0) \times 2^0$
- ce qui correspond au nombre 644 en base 8.
- Attention à ne pas dire « six cent quarante-quatre » car il ne s'agit pas de 644 en base 10. En l'occurrence, on ne peut pas vraiment dire mieux que « six quatre quatre »...

Droits d'accès - Droits d'accès en binaire et en octal

	r	w	x	binaire	octal
---	0	0	0	0 0 0	0
--X	0	0	1	0 0 1	1
-W-	0	1	0	0 1 0	2
-WX	0	1	1	0 1 1	3
r--	1	0	0	1 0 0	4
r-X	1	0	1	1 0 1	5
rW-	1	1	0	1 1 0	6
rWX	1	1	1	1 1 1	7

JULIA Evans
@bork

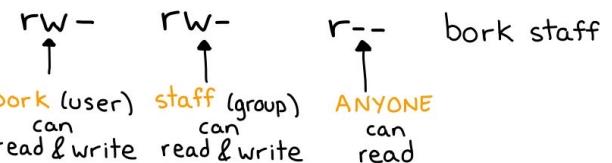
unix permissions

drawings.jvns.ca

There are 3 things you can do to a file

↓ read ↓ write ↓ execute

ls -l file.txt shows you permissions
Here's how to interpret the output:



File permissions are 12 bits

setuid setgid
 ↓ ↓
 000 110 110 100
 sticky rwx rwx rwx rwx
 user group all

For files:

- r = can read
- w = can write
- x = can execute

For directories it's approximately:

- r = can list files
- w = can create files
- x = can cd into & modify files

110 in binary is 6
So rW- r-- r--
 = 110 100 100
 = 6 4 4

chmod 644 file.txt means change the permissions to:

rW- r-- r--

Simple!

setuid affects executables
 \$ls -l /bin/ping
 rws r-x r-x root root
 this means ping always runs as root

setgid does 3 different unrelated things for executables, directories, and regular files



Droits d'accès - chmod

- Signifie :change mode.
- La commande chmod permet de modifier les droits d'accès de fichiers ou de répertoires.
- Son premier argument indique quels droits d'accès modifier dans les fichiers dont les chemins d'accès figurent dans les arguments suivants.
- Les modifications à apporter aux droits d'accès peuvent être indiquées
 - de manière symbolique :
 \$ chmod o-r toto
 - de manière numérique :
 \$ chmod 644 toto
- L'option –R permet d'agir récursivement.

Droits d'accès - La commande umask

- Signifie :*User file creation mask*.
La commande umask affiche (sans argument) ou modifie (avec un argument) le masque de création des fichiers de l'interpréteur de commandes :
 - affichage :
 \$ umask 022
 - modification :
 \$ umask 077
- Au-delà des interpréteurs de commandes, chaque processus possède son propre masque de création des fichiers — hérité de son père — et qu'il peut modifier.

Droits d'accès

EXERCICE

- Dans votre répertoire courant, vous créez un répertoire courant `essai_droit`. Par défaut, ce répertoire est à 755 (rwxr-xr-x). Quelles sont les commandes (en notation symbolique et en base 8) pour lui donner les droits suivants (on suppose qu'après chaque commande on remet le répertoire à 755) :

	<i>Propriétaire</i>			<i>Groupe</i>			<i>Les autres</i>		
	Lecture	Ecriture	Accès	Lecture	Ecriture	Accès	Lecture	Ecriture	Accès
Commande 1	oui	oui	oui	oui	non	oui	non	non	oui
Commande 2	oui	non	oui	non	oui	non	non	non	oui
Commande 3	non	oui	non	non	non	oui	oui	non	non
Commande 4	non	non	oui	oui	non	oui	non	non	non

Gestion de l'espace de stockage, compression et archivage - La commande find

- La commande find parcourt récursivement le système de fichiers à la recherche de fichiers satisfaisant aux critères indiqués.
- La syntaxe générale de la commande find est la suivante :
 \$ find [options] [répertoires] [expressions]
- La commande find s'utilise
 - avec des options
 - avec des répertoires dans lesquels effectuer les recherches (si aucun n'est indiqué, le répertoire courant sera utilisé)
 - avec des expressions pouvant être :
 - Des options affectant la recherche
 - des critères de recherche
 - des actions à effectuer
- Quelques options
 - maxdepth
 - mindepth
 - xdev
- Le critère –name permet de faire une recherche basée sur le nom (complet ou partiel) du fichier

Gestion de l'espace de stockage, compression et archivage - La commande find

- Le critère **-type** permet de restreindre la recherche à un type de fichiers particulier :

```
$ find . -type l
```

Les types possibles sont les suivants :

- b fichiers spéciaux de type bloc
- c fichiers spéciaux de type caractère d répertoires
- f fichiers ordinaires
- l liens symboliques p tuyaux nommés
- s sockets

- **-uid** restreint la recherche aux fichiers appartenant à l'UID (numérique) indiqué
- **-user** restreint la recherche aux fichiers appartenant à l'utilisateur indiqué (spécifié par son identifiant ou son UID)

Gestion de l'espace de stockage, compression et archivage - La commande find

- Le critère `-size` permet de restreindre la recherche aux fichiers de plus ou moins d'une certaine taille :

```
$ find . -size +1G
```

- Les unités les plus utiles sont les suivantes :
 - c octet
 - k kibioctet
 - M mébioctet
 - G gibioctet

Gestion de l'espace de stockage, compression et archivage - La commande find

- Les critères temporels sont les suivants :
 - -amin dernier accès en minutes
 - -atime dernier accès en jours
 - -cmin dernier changement des métadonnées en minutes
 - -ctime dernier changement des métadonnées en jours
 - -mmin dernière modification en minutes
 - -mtime dernière modification en jours

Gestion de l'espace de stockage, compression et archivage - La commande find

- delete supprime les fichiers correspondant aux autres critères :
\$ find . -name toto -delete
- -exec exécute, pour chaque fichier correspondant aux autres critères, la commande indiquée jusqu'au point-virgule (qui doit être inhibé pour éviter que l'interpréteur de commandes ne le considère), en remplaçant l'expression {} par le nom du fichier :
- \$ find . - name toto - exec rm {} \;

Gestion de l'espace de stockage, compression et archivage - La commande find

- Lorsque la commande find est utilisée avec plusieurs critères, ceux-ci sont combinés avec un « et » implicite :

```
$ find . -name\*.pdf -type l
```

- L'opérateur -a(and) combine deux critères avec un « et » explicite :

```
$ find . -name\*.pdf -a -type l
```

- L'opérateur -o(Or) combine deux critères avec un « ou » explicite:

```
$ find . -name\*u.iso -o -size +500M
```

Gestion de l'espace de stockage, compression et archivage - La commande find

- L'opérateur de négation ! (qui doit être inhibé, le point d'exclamation ! étant un caractère spécial pour l'interpréteur de commandes) inverse la signification d'un critère :

```
$ find . \! -type f
```

- Des parenthèses (qui doivent être inhibées pour la même raison) permettent de grouper plusieurs critères et d'indiquer le niveau de priorité

```
$ find . \( -name \*.jpg -o -name \*.png \) -a -size +1M
```

Find

EXERCICE

- 1) Chercher tous les fichiers dont le nom est 'passwd'.
- 2) Chercher tous les fichiers dont la date de la dernière modification remonte à plus de 10 minutes.
- 3) Trouver tous les fichiers du groupe 'root'.
- 4) Chercher tous les fichiers dont la taille est supérieure à 20Mo.
- 5) Chercher tous les répertoires se trouvant sous /etc.
- 6) Chercher tous les fichiers de l'utilisateur 'Rimbault'

Gestion de l'espace de stockage, compression et archivage - La commande du

- Signifie : *disk Usage*.
- La commande du effectue une descente en profondeur d'abord des répertoires en arguments (ou, sans argument, du répertoire courant) et affiche le volume occupé par chaque répertoire (fichiers et sous-répertoires compris).
- *-h(human readable)* affiche les volumes de données avec l'unité la plus adaptée
- *s(summarize)* affiche uniquement le volume total, sans afficher celui occupé par chaque sous-répertoire

Gestion de l'espace de stockage, compression et archivage - La commande df

- Signifie : *disk free*.
- La commande df affiche le taux d'occupation de chaque système de fichiers monté (sans argument) ou des systèmes de fichiers contenant les fichiers en arguments
- -h (*human readable*) affiche les volumes de données avec l'unité la plus adaptée
- -i (*inodes*) affiche les taux d'occupation des i-nœuds au lieu des blocs de données

Gestion de l'espace de stockage, compression et archivage - La compression

- La *compression* consiste à transformer, au moyen d'un algorithme, une suite de données (par exemple le contenu d'un fichier) en une autre suite de données plus courte mais contenant les mêmes informations. La *décompression* est l'opération inverse.
- Il existe deux types de compressions :
 - la compression dite *sans perte* lorsque la décompression restitue exactement les données d'origine ;
 - la compression dite *avec pertes* lorsque la décompression aboutit à des données légèrement différentes des données d'origine.
- La compression avec pertes ne peut s'appliquer qu'à certains types de données bien particuliers (image, son...).
- Nous n'utiliserons pour notre part que de la compression sans perte afin de pouvoir restaurer les données d'origine sans aucune altération

Gestion de l'espace de stockage, compression et archivage - La compression

- La commande gzip compresse les fichiers en arguments
- La commande gunzip décomprime les fichiers en arguments
- Utiliser la commande gzip avec l'option -d(ou –decompress ou --uncompress) a le même effet
- La commande bzip2 comprime les fichiers en arguments, qui seront remplacés par des fichiers de mêmes noms auxquels sera rajoutée l'extension .bz2
 - Les options -1 à -9 contrôlent le niveau de compression (-1 est le moins efficace, -9 est le plus efficace et est le niveau par défaut)
- La commande bunzip2 décomprime les fichiers en arguments
- La commande xz comprime les fichiers en arguments, qui seront remplacés par des fichiers de mêmes noms auxquels sera rajoutée l'extension .xz
 - Les options -0 à -9 contrôlent le niveau de com pression (-0 est le m oins efficace, -6 est le niveau par défaut, -9 est le plus efficace)
- La commande unxz décomprime les fichiers en arguments

Gestion de l'espace de stockage, compression et archivage - L'archivage

- *L'archivage* consiste à rassembler une arborescence de répertoires et de fichiers dans un unique fichier, qu'on appelle une *archive*, et qui contient toutes les informations nécessaires à la recréation de l'arborescence d'origine.
- Lorsqu'un ensemble de fichiers n'a plus d'utilité immédiate, on peut générer une archive permettant de recréer ces fichiers le jour venu. Afin d'occuper le moins de volume de stockage possible, cette archive est souvent compressée, sauf si elle contient un grand nombre de fichiers déjà compressés, comme des images JPEG, des fichiers audio MP3 ou des fichiers compressés avec les commandes qui viennent d'être étudiées.
- Une archive (compressée au besoin) peut ainsi facilement être stockée sur un support externe (clé USB, bande magnétique, etc.) ou transférée grâce au réseau informatique

Gestion de l'espace de stockage, compression et archivage - L'archivage- tar

- Signifie :*tape archiver*.
La commande tar permet de gérer l'archivage d'un ensemble de fichiers
- La commande tar s'utilise obligatoirement avec au moins une option. Dans ces conditions, l'utilisation du tiret pour introduire les options n'est pas nécessaire.
- La première option, obligatoire, indique l'opération à effectuer
 - c créer une archive
 - x extraire une archive
 - t lister le contenu d'une archive
- Options de compression :
 - j compression ou décompression avec bzip2
 - J compression ou décompression avec xz
 - z compression ou décompression avec gzip
- Autres options :
 - f fichier (ordinaire ou spécial) à utiliser
 - v affichage verbeux

Gestion de l'espace de stockage, compression et archivage - L'archivage

- Nous pouvons également utiliser les formats:
 - Zip avec les commandes zip et unzip
 - Rar avec les commandes rar et unrar

Gestion de l'espace de stockage, compression et archivage - L'archivage

- Nous pouvons également utiliser les formats:
 - Zip avec les commandes zip et unzip
 - Rar avec les commandes rar et unrar

Processus

- Un processus est un programme en cours d'exécution
- Un processeur ne peut exécuter qu'une seule instruction à la fois. BUT: Partager un (ou plusieurs) processeur entre différents processus.
- Il existe deux types de processus
 - Système
 - Processus qui sont lancés au démarrage du système,
 - Ces processus ne sont sous le contrôle d'aucun terminal et ont comme propriétaire l'administrateur du système
 - Utilisateurs
 - Ils correspondent à chaque exécution d'un programme par l'utilisateur, le premier d'entre eux étant l'interpréteur de commandes à la connexion.
 - Ces processus appartiennent à l'utilisateur et sont généralement attachés à un terminal

Processus – cycle de vie

- Un processus peut être en :
 - Exécution.
 - Attente.
 - Prêt.

Processus

- Les caractéristiques statiques d'un processus sont:
 - Un numéro unique: PID (Process IDentifier),
 - Un propriétaire déterminant les droits d'accès du processus aux ressources : ouverture de fichiers...
 - Un terminal d'attache pour les entrées/sorties
- Les caractéristiques dynamiques:
 - Priorité, environnement d'exécution
 - Ressources consommées

L'ordonnanceur

- L'ordonnanceur (scheduler) est le module du SE qui s'occupe de sélectionner le processus suivant à exécuter parmi ceux qui sont prêts.

Canaux de communication

- A chaque création de processus, celui ci se voit affecté trois canaux de communication :
- Entrée standard
- Sortie standard
- Sortie d'erreurs standard

Canal de communication	Fichier	Numéro logique
Entrée standard	stdin	0
Sortie standard	stdout	1
Sortie d'erreurs standard	stderr	2

Canaux de communication - Fichiers

- **Le fichier stdin :**
 - fichier à partir duquel le process va lire les données nécessaires en entrée.
 - Ouvert avec le numéro logique 0(file descriptor C)
 - Par défaut associé au clavier
- **Le fichier stdout :**
 - fichier dans lequel le process va écrire les messages qu'il produit en sortie, dans le cas d'une exécution normale.
 - ouvert avec le numéro logique 1 (file descriptor C)
 - par défaut associé à l'écran
- **Le fichier stderr :**
 - fichier dans lequel le process va écrire les messages d'erreur.
 - ouvert avec le numéro logique 2 (file descriptor C)
 - par défaut associé à l'écran

Code retour

- L' exécution de toute commande UNIX se termine par l'envoi d'un code retour au processus père. Celui-ci indique le bon déroulement ou non de la commande.
- un code retour égal à:
 - 0 : indique un bon déroulement
 - 1 : indique une erreur de syntaxe
 - 2 : une erreur d'emploi de la commande.
- La variable \$? du shell permet d'afficher le code retour de la dernière commande exécuté

Les signaux

- Un processus (ou le système) peut envoyer un signal à un autre processus (commande ou appel kill). Le processus destinataire réagit instantanément (il interrompt l'exécution de son programme, traite le signal, et éventuellement reprend son exécution)
- Le système communique avec les processus à l'aide de signaux. Un signal ne transporte pas d'information autre que son numéro. Il existe quelques dizaines de signaux distincts, définis par le système (kill -l pour obtenir la liste)
- Par exemple:
- SIGKILL Termine le processus autoritairement
- SIGSTOP Met le processus en attente (sommeil)
- SIGCONT Reprend l'exécution d'un processus endormi

Les signaux

- Exemples de génération de signaux :
- Lorsqu'un fils se termine, un signal SIGCHLD est envoyé à son père. Cependant, le père ne sait pas lequel de ses fils s'est terminé.
- Lorsqu'un processus écrit à une adresse mémoire
- invalide, un signal SIGSEGV est généré
- div. par 0 envoie le signal SIGFPE (Floating Point Exception)
- Certaines touches entraînent l'envoi d'un signal :
 - Ctrl+C Envoie SIGINT
 - Ctrl+\ Envoie SIGQUIT

Exécution séquentielle

- Le mode d'exécution par défaut
- Exécution synchrone
- Pour lancer l'exécution séquentielle de plusieurs commandes sur la même ligne de commande, il suffit de les séparer par le caractère ;

Exécution En arrière plan

- permet de rendre immédiatement le contrôle à l'utilisateur
- On utilise le caractère & pour lancer une commande en arrière-plan (cmd &)
- Si le terminal est fermé, la commande en arrière plan est interrompue automatiquement
- Solution: lancer la commande sous le contrôle de la commande nohup (nohup nom_commande &)

Commande PS

- Commande ps
- permet de voir l'état des processus en cours d'exécution sur une machine
- Syntaxe:
- ps [options]
- Rarement utilisé sans option

Option:

-e : affichage de tous les processus

-f : affichage détaillé

-x : permet de visualiser tout les processus actifs de l'utilisateur courant

-ax : permet de visualiser tous les processus de la machine de tous les utilisateurs

-aux : permet de visualiser affiche les utilisateurs associés à chaque processus

-u nom utilisateur : affiche chaque processus associés à utilisateur

Commande TP

- Permet d'afficher des informations en continu sur l'activité du système
- Permet surtout de suivre les ressources que les processus utilisent (quantité de RAM, pourcentage de CPU, la durée de ce processus depuis son démarrage)

Commande kill

Commande kill

- pour arrêter un processus, on doit aussi tuer un processus
- On doit connaître son PID (commande ps)
- Syntaxe:
 - kill -9 PID
 - kill PID

Redirections entrées sorties

- Redirection = débrancher la connexion au terminal pour brancher l'E/S sur autre chose un fichier

cat < fichier (ou cat 0< fichier)

cat > fichier (ou cat 1> fichier)

cat 2> fichier : redirige les erreurs vers fichier

Possibilité de redirections

cat < toto > titi 2> tutu

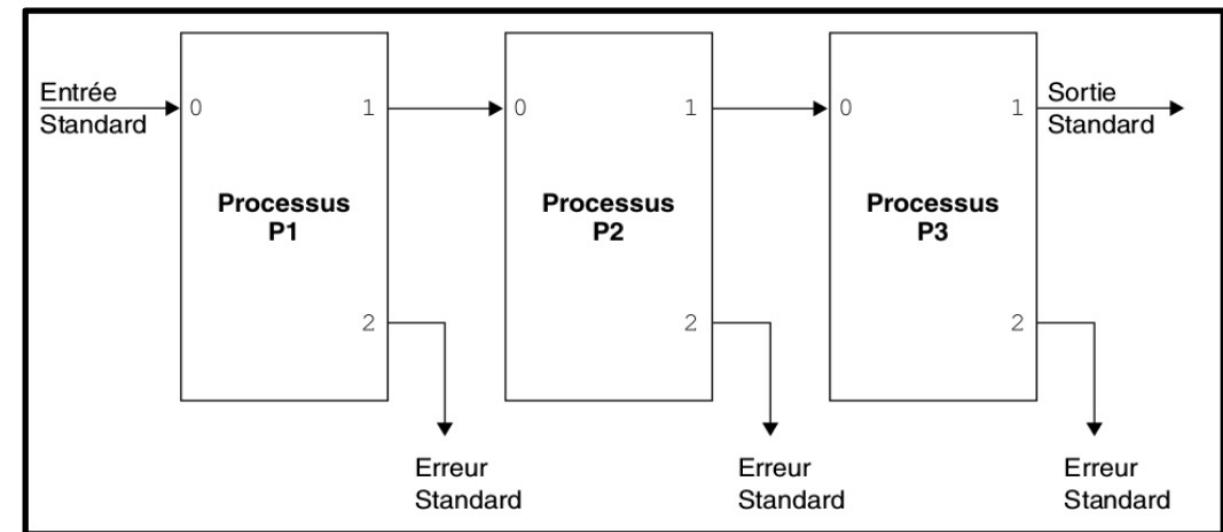
Redirections

EXERCICE

- Ecrire une ligne de commande équivalente à :
- ls -l /usr/bin > tmp less tmp rm tmp

Tubes (pipe)

- Un tube (pipe en anglais) est un flot de données qui permet de relier la sortie standard d'une commande à l'entrée standard d'une autre commande sans passer par un fichier temporaire.



Tubes (pipe)

- Dans une ligne de commandes, le tube est formalisé par la barre verticale |, que l'on place entre deux commandes :

P1 | P2 | P3

Les filtres

- Un filtre est une commande qui lit les données sur l'entrée standard, les traite et les écrit sur la sortie standard.
- Le concept de tube, avec sa simplicité, devient un outil très puissant dans Linux qui propose un choix très vaste de filtres

Les filtres

- Les filtres
 - grep : recherche les occurrences d'une chaîne de caractères.
 - egrep : une extension de grep
 - wc : compte le nombre de caractères (ou octets), mots et lignes.
 - less : affiche son entrée standard page par page.
 - dd : filtre de conversion.
 - sed : éditeur de flot : il applique des commandes de l'éditeur ed sur l'entrée standard et envoie le résultat sur la sortie standard.
 - awk : petit langage de manipulation de texte.
 - sort : filtre de tri.

- Supposons :
- \$ cat devinette.txt devinette numero 4 : pince mi et pince moi sont dans un bateau. pince mi tombe à l'eau. qui est ce qui reste ? Qu'affichent les commandes suivantes (A : 0; B : 1; C : 2; D : 3; E : 4; F : 5) :
 - 1) cat devinette.txt | grep ce | wc -l ?
 - 2) cat devinette.txt | grep 4 | wc -l ?

Filtres

EXERCICE

- On suppose qu'un fichier liste.txt contient des informations sur les étudiants (10 au moins).

Chaque ligne représente un étudiant, et contient les informations suivantes : nom, âge et filière. Les champs seront séparés par un « ; ».

Exemple : la ligne Dumont;23;L3 correspond à l'étudiant Dumont, âgé de 23 ans et appartenant à la filière L3.

- 1) Renvoyer toutes les lignes du fichier liste.txt qui correspondent à l'étudiant s'appelant 'Sami'.
- 2) Renvoyer toutes les lignes correspondant à des étudiants de la filière L3.
- 3) Renvoyer toutes les lignes des étudiants âgés de 22 ans.
- 4) Renvoyer les lignes des étudiants n'appartenant pas à la filière L3.
- 5) Renvoyer toutes les lignes contenant la chaîne 'mi' sans tenir compte de la casse.
- 6) Afficher le nom et l'âge de chaque étudiant, puis le nom et la filière.
- 7) Afficher les trois premiers caractères de chaque ligne.