

Linux

Presentation

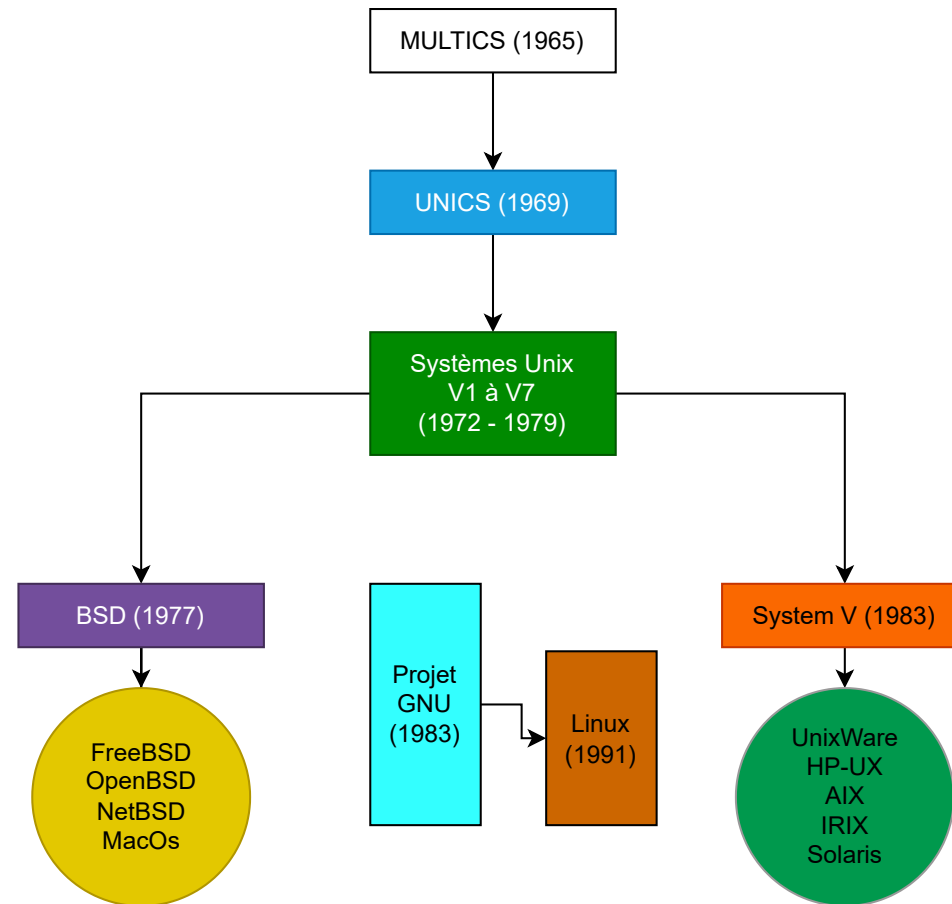
Le système **Unix** est un système d'exploitation **multi-utilisateurs**, **multi-tâches**, ce qui signifie qu'il permet à un ordinateur **mono** ou **multi-processeurs** de faire exécuter simultanément plusieurs programmes par un ou plusieurs utilisateurs.

Il possède une grande portabilité, ce qui signifie qu'il est possible de mettre en oeuvre un système Unix sur la quasi-totalité des plates-formes matérielles.

Un peu d'histoire

- C'est en 1969 que Kenneth THOMPSON, employé chez Bell, développe un nouveau prototype de système à temps partagé ; son nom est Unics et sera Unix.
- En 1980, des chercheurs de l'université de Berkeley développent leur propre UNIX (BSD).
- En 1983, la société ATT tente une commercialisation d'un UNIX système V et de concurrencer l'UNIX BSD.

- En 1984, Richard Stallman lance le projet GNU qui vise à créer son système 'Unix' complètement libre.
- En 1991, un étudiant finlandais, Linus TORVALDS, crée un noyau UNIX qui a été ajouté aux travaux du projet GNU de Stallman, a donc donné naissance à GNU/Linux.



Les caractéristiques de Linux

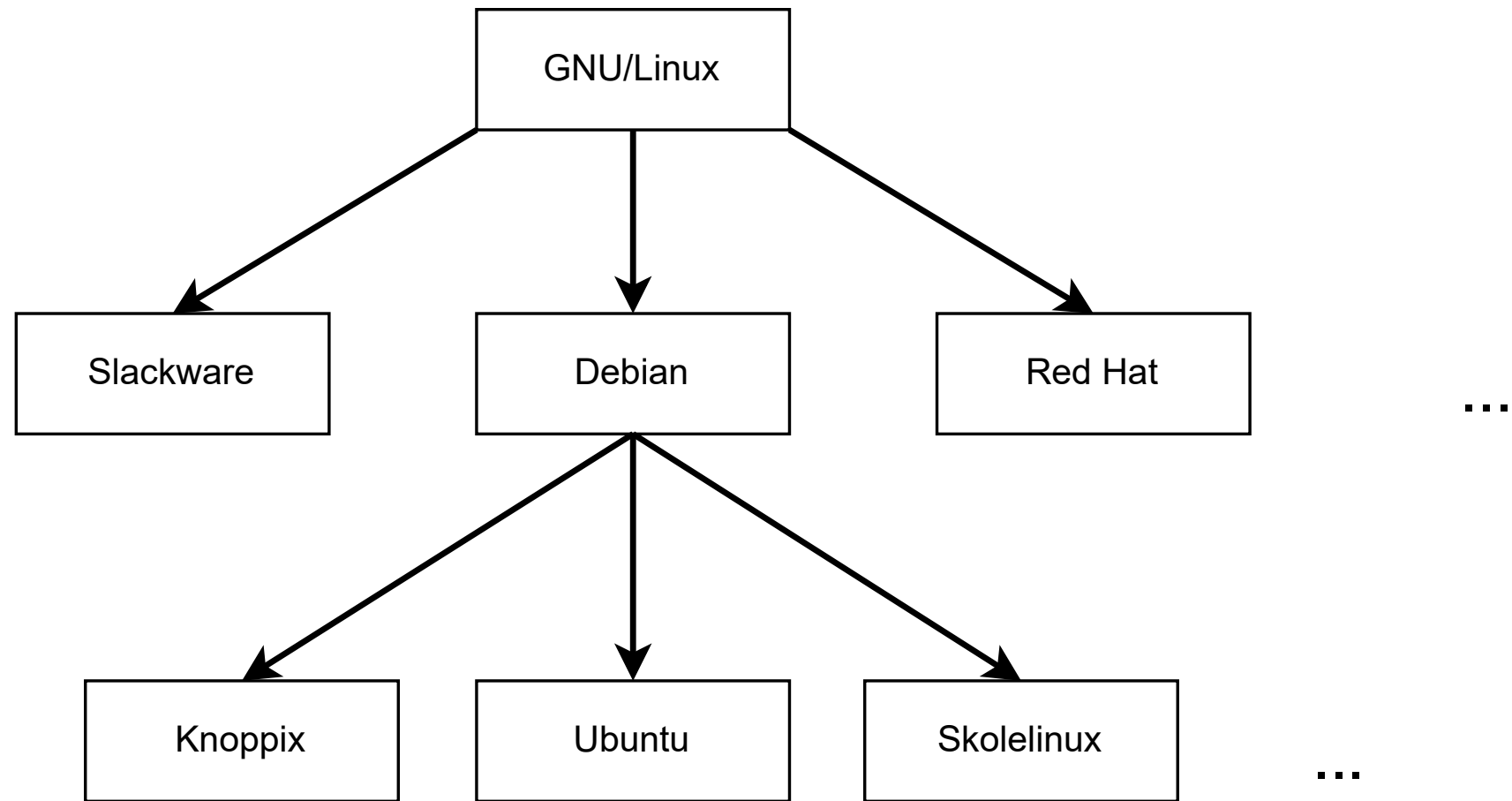
Linux ou **GNU/Linux** est une famille de systèmes d'exploitation open source de type Unix fondé sur le noyau Linux, créé en 1991 par Linus Torvalds.

Linux est :

- Gratuit
- Open Source
- Sécurisé
- Multitâches
- Multi-utilisateurs
- Stable
- Scalable

Les distributions Linux

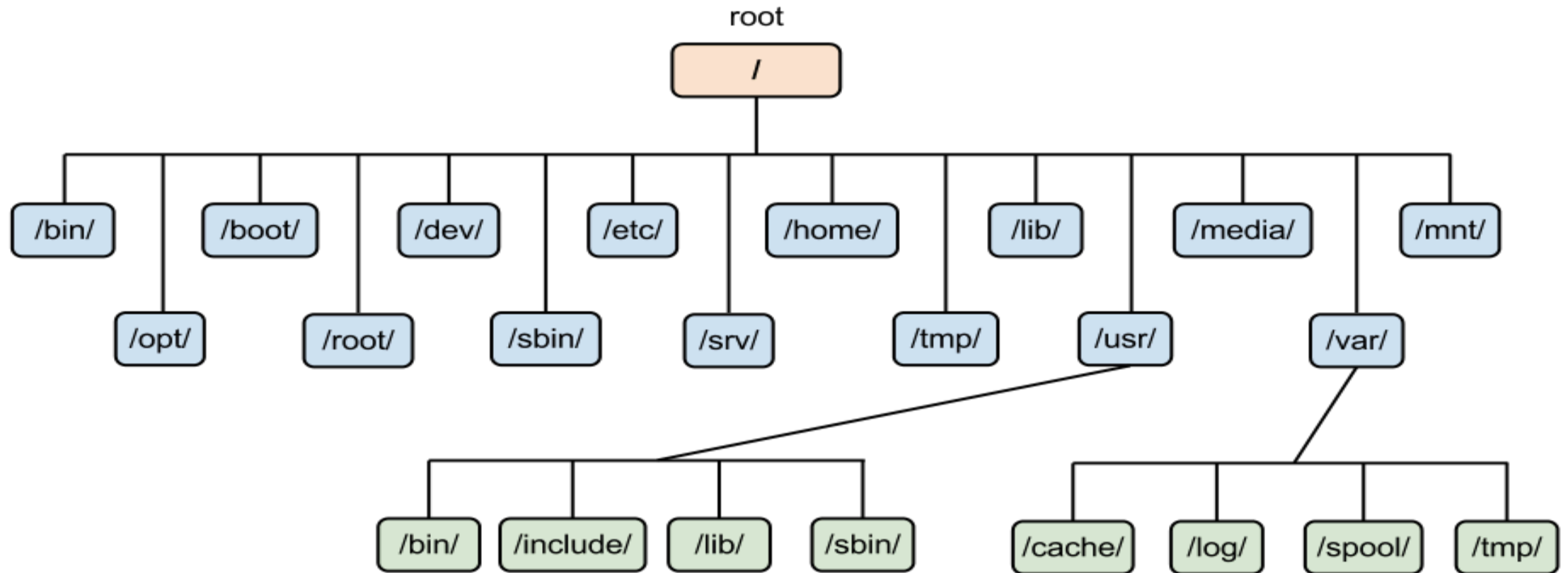
- On appelle distribution **GNU/Linux** (ou distribution Linux) une solution prête à être installée par l'utilisateur final.
- Une distribution Linux se compose d'un noyau, de packages, et d'outils pour gérer leurs dépendances.
- Les distributions sont développées pour répondre à un besoin (serveur, poste de travail ou autre).
- Debian, Red Hat et Slackware sont les 3 premières distributions Linux historiques



Installation

- Pour installer un environnement terminal sur Windows, on peut chercher des applications comme celle-ci:
[MicrosoftStore Ubuntu](#)
- Facile à installer et plus rapide à l'exécution

Le système de fichier avec Linux



Un peu plus détaillé



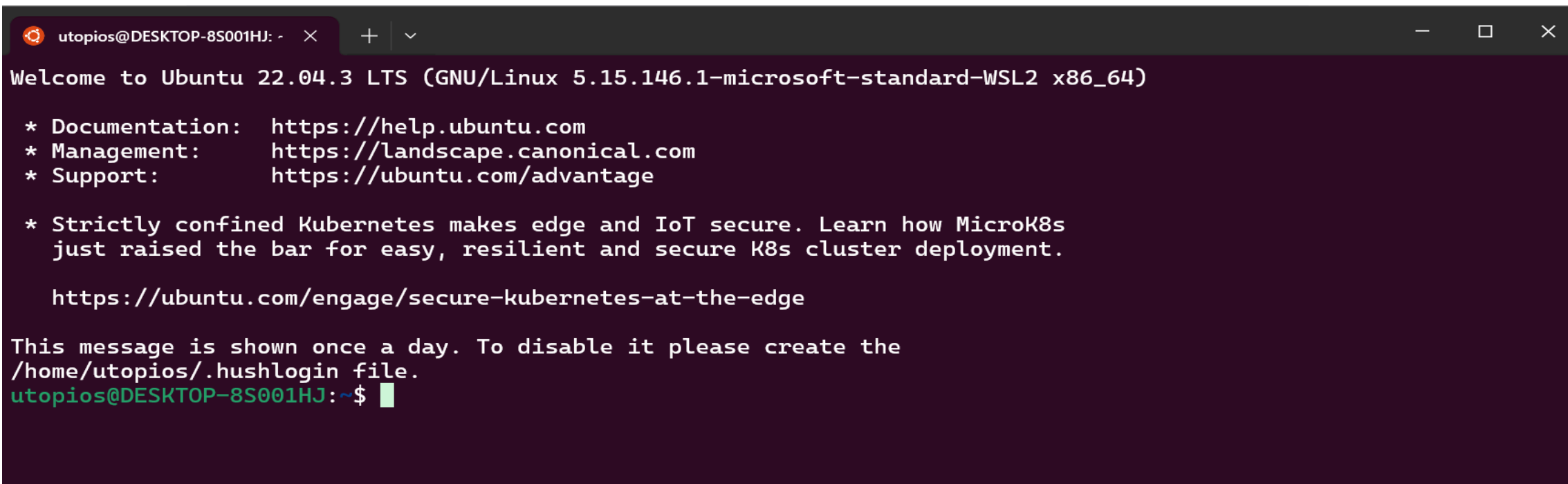
Répertoire	Signification
/bin/	Répertoire contenant les commandes systèmes générales nécessaires à l'amorçage. Ce répertoire doit être impérativement placé dans le système de fichiers racine. Tous les utilisateurs peuvent utiliser les commandes de ce répertoire.
/lib/	Répertoire contenant les bibliothèques partagées (« DLL » en anglais, pour « Dynamic Link Library ») utilisées par les commandes du système des répertoires /bin/ et /sbin/. Ce répertoire doit être impérativement placé dans le système de fichiers racine.
/etc/	Répertoire contenant tous les fichiers de configuration du système. Ce répertoire doit être impérativement placé dans le système de fichiers racine.
/tmp/	Répertoire permettant de stocker des données temporaires. En général, /tmp/ ne contient que des données très éphémères. Il est préférable d'utiliser le répertoire /var/tmp/. En effet, le répertoire /tmp/ ne dispose pas nécessairement de beaucoup de place disponible.

Répertoire	Signification
/usr/	Répertoire contenant les fichiers du système partageables en réseau et en lecture seule.
/var/	Répertoire contenant toutes les données variables du système. Ce répertoire contient les données variables qui ne pouvaient pas être placées dans le répertoire /usr/, puisque celui-ci est normalement accessible en lecture seule.
/opt/	Répertoire historique contenant les applications qui ne font pas réellement partie du système d'exploitation. En particulier, sur les anciennes distributions, le gestionnaire de bureau KDE était installé dans le sous-répertoire /opt/kde/, mais à présent il est considéré comme partie intégrante du système et est donc installé directement dans le répertoire /usr/ sur les distributions récentes.
/sys/	Répertoire contenant le pseudo système de fichiers des gestionnaires de périphériques. Ce pseudo système de fichiers contient des fichiers permettant d'obtenir des informations sur l'ensemble des objets du noyau, en particulier sur l'ensemble des périphériques de l'ordinateur.

Répertoire	Signification
/root/	Répertoire contenant le répertoire personnel de l'administrateur. Il est donc recommandé que le répertoire personnel de l'administrateur soit placé en dehors de /home/ pour éviter qu'un problème sur le système de fichiers des utilisateurs ne l'empêche de travailler.
/media/	Répertoire réservé au montage des systèmes de fichiers sur périphériques amovibles (CD-ROM, disquettes, etc.). Ce répertoire peut contenir plusieurs sous répertoires pour chaque périphérique amovible, afin de permettre d'en monter plusieurs simultanément.
/proc/	Répertoire contenant le pseudo système de fichiers du noyau. Ce pseudo système de fichiers contient des fichiers permettant d'accéder aux informations sur le matériel, la configuration du noyau et sur les processus en cours d'exécution.
/home/	Répertoire contenant les répertoires personnels des utilisateurs.

l'interpréteur de commandes - Shell

- L'interpréteur de commandes (**shell** en anglais) est un logiciel fondamental pour travailler sur un système UNIX.



```
utopios@DESKTOP-8S001HJ: ~  
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.146.1-microsoft-standard-WSL2 x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s  
  just raised the bar for easy, resilient and secure K8s cluster deployment.  
  
  https://ubuntu.com/engage/secure-kubernetes-at-the-edge  
  
This message is shown once a day. To disable it please create the  
/home/utopios/.hushlogin file.  
utopios@DESKTOP-8S001HJ:~$
```

- 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

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)

```
utopios@DESKTOP-8S001HJ:~$
```

Saisie

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

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.

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

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
```

- Syntaxe générale des commandes

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

- Raccourcis historique :
 - ↑ commande précédente
 - ↓ commande suivante

List segments

La commande ls 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 -al
```

Pour afficher les informations concernant un fichier en particulier, il faut indiquer son nom en argument :

```
$ ls -l toto.txt
```


Catenate

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

```
$ cat tata.txt
```

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

Manual

La commande man permet d'afficher le manuel de la commande en argument :

```
$ man ls
```

Exercice

Unix possède un manuel « en ligne ». La commande `man` permet d'explorer ce manuel.

- Comment est structurée la documentation de ce manuel ?
- Comment accède-t-on à la page du manuel concernant la commande `write` ?
- Commande `ls` : précisez les options que vous savez utiliser et celles que vous pourriez éventuellement utiliser

Editeurs de texte

- 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 est une variante de vi avec de nombreuses améliorations

Vi

- Signifie : *visual*
- 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
```

Commande vi

- i passage en mode insertion au caractère situé sous le curseur
- a passage en mode insertion au caractère situé après le curseur
- l 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

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.

Déplacement

- ← ou h : déplacement vers la gauche
- ↓ ou j : déplacement vers le bas
- ↑ ou k : déplacement vers le haut
- → ou l : déplacement vers la droite
- w/W : déplacement au mot suivant
- b/B : déplacement au mot précédent

- 0 : aller en début de ligne
- \$: aller en fin de ligne
- { : déplacement au paragraphe précédent
- } : déplacement au paragraphe suivant
- G : déplacement a la dernière ligne
- :<n> ou <n>G : déplacement a la ligne numéro n (ligne 6 -> :6 + entrer ou 6G)

Suppression / Remplacement

- x : supprime le caractère sous le curseur
- d : supprime la ligne
- D : supprime la fin de la ligne à partir du curseur
- r<x> : remplace le caractère sous le curseur par <x>
- cw : remplace la fin du mot à partir du curseur
- C : remplace la fin de la ligne à partir du curseur
- cc : remplace toute la ligne
- u : annule la modification précédente
- ESC : annuler la commande en cours

Copier-coller / Couper-coller

- <n>yy : copier (n = nombre de lignes)
- <n> dd : couper
- p : coller
- :j : joindre la ligne suivante à la ligne courante

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
:s/foo/bar remplace foo en bar

Fichier

- :w : sauvegrader le fichier (s'il a déjà un nom)
- :w : sauvegarder le fichier sous le nom indiqué
- :q : quitter
- :wq ou :x : sauvegarder le fichier et quitter
- :q! : quitter sans sauvegarder le 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 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

Copy

- La commande cp 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 » ?
```


Move

- 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 écraser « tata.txt » ?
```

Remove

- 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 » ?
```

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.

Strucuture 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
```

Link

- 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
```

Make Directories

- 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
```

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
```

Change directory

- La commande cd 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 designer 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, que l'on peu retrouver via la commande pwd(print working directory)

Chemin d'accès absolu

- 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 absolu est non ambigu, il désigne toujours le même fichier, indépendamment de l'endroit où l'on se trouve.

Chemin d'accès relatif

- 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 relatif, dépend du répertoire courant.

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

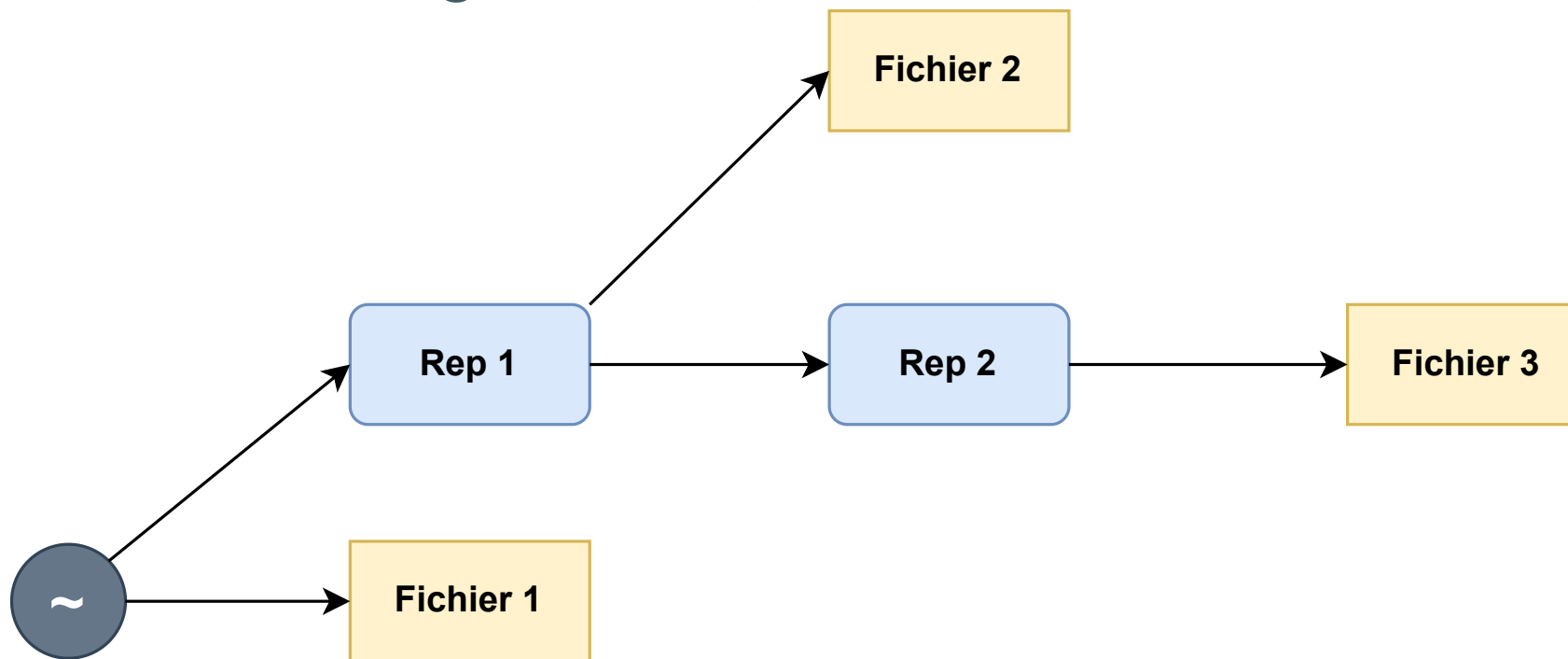
```
$ rmdir rep  
rmdir: échec desuppression de « rep »: Le dossier n'est pas vide
```

- La commande `rm`, utilisée avec l'option `-r` (recursive), permet de supprimer un ou plusieurs répertoires et tout leur contenu :

```
$ rm-r rep  
$ rm-r rep1 rep2 rep3
```

Exercice

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



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`

Métacaractères

- 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 caractères "?"

- Le métacaractère ? correspond à exactement un caractère quelconque.
- exemple :

```
$ ls test?  
test1  test2  test3  tests
```


Le caractères "[""]"

- Plus précis que ?, une expression entre crochets correspond également à exactement un caractère mais uniquement parmi ceux indiqués entre les crochets.
- Une expression entre crochets peut contenir une liste de caractères, un intervalle de caractères de même nature ou une combinaison de ces deux syntaxes .

```
$ ls test[23]
test2  test3
$ ls test[0-9]
test1  test2  test3
```

- 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.

```
$ ls test[!23]  
test1  tests
```

```
$ ls test[^23]  
test1  tests
```

Le caractères "*"

- Le métacaractère * correspond à Zéro, un ou plusieurs caractèresquelconques.
- exemple :

```
$ ls te*  
test1  test2  test3  tests  
  
$ ls *e*  
test1  test2  test3  tests  
  
$ ls *s  
tests
```

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

SUID

- 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 répertoire

SGID

- 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
```

- 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

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.

Octal Representation

0	000	- - -	No permissions
1	001	- - x	Only Execute
2	010	- w -	Only Write
3	011	- w x	Write and Execute
4	100	r - -	Only Read
5	101	r - x	Read and Execute
6	110	r w -	Read and Write
7	111	r w x	Read, Write and Execute

JULIA EVANS
@b0rk

unix permissions

4

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:

rw- rw- r-- bork staff
↑ ↑ ↑
bork (user) staff (group) ANYONE
can can can
read & write read & write read

File permissions are 12 bits

setuid setgid
↓ ↓
000 110 110 100
sticky 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 & access 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`
rw- 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.

unix! why?? it's a long story unix

- 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 110 100 100

- Par définition, le nombre binaire 110100100 se décompose en puissances de 2 ainsi :

$$110100100 = 1 \times 2^8 + 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$

- soit, en factorisant la puissance de 2 la plus petite de chaque ligne :

$$110100100 = (1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0) \times 2^6 + (1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0) \times 2^3 + (1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0) \times 2^0$$

- 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 »...

Change mode

- commande : `chmod`
- 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 argument 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.

Modification symbolique

- Il est possible de modifier les droits d'accès indépendamment grâce à l'écriture symbolique :
 - u propriétaire (user)
 - g groupe (group)
 - o les autres (others)
 - a tous (all)
- + ou – pour ajouter ou retirer un droit
- Suivi du/des droit(s) à ajouter ou retirer
- ex: `chmod u+x fichier` (ajoute l'exécution à l'utilisateur)

User file creation mask

- commande : umask
- La commande umask affiche (sans argument) ou modifie (avec un argument) le masque de création des fichiers de l'interpréteur de commandes :

- affichage

```
$ umask
```

- modification

```
$ umask 077
```

Permission	Fichier			Répertoire		
	user	group	other	user	group	other
Prédéfini	6	6	6	7	7	7
Umask	0	0	2	0	0	2
Defaut	6	4	4	7	5	5

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) :

	User			Group			Other		
	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

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

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

Les types possibles sont les suivant :

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

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

- c octet
- k kibioctet
- M medioctet
- G gibioctet

- 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

- -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 { } \ ;
```

- 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
```

- L'opérateur de négation ! (qui doit être inhibé, le point d'exclamation ! étant un caractère spécial pour l'interpreteur 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 priorite

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


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 associés à votre utilisateur

Disk Usage

- Commande : du
- 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

Disk free

- Commande : df
- 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

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 :
 - a 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.

Gzip

- La commande gzip compresse les fichiers en arguments, qui seront remplacés par des fichiers de mêmes noms auxquels sera rajoutée l'extension .gz
- La commande gunzip décompresse les fichiers en arguments
- Utiliser la commande gzip avec l'option -d(ou -decompress ou --uncompress) a le même effet

Bzip2

- 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

Xz

- 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 compression (-0 est le moins efficace, -6 est le niveau par défaut, -9 est le plus efficace)
- La commande unxz décomprime les fichiers en arguments

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 recreation 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 comprimée, sauf si elle contient un grand nombre de fichiers déjà comprimés, comme des images JPEG, des fichiers audio MP3 ou des fichiers comprimés avec les commandes qui viennent d'être étudiées.
- Une archive (comprimée au besoin) peut ainsi facilement être stockée sur un support externe (clé USB, etc.) ou transférée grâce au réseau informatique

Tape archiver

- Commande : tar
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
- 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 est capable d'exécuter uniquement une instruction à la fois, mais il peut être partagé 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

Cycle de vie

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

- 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

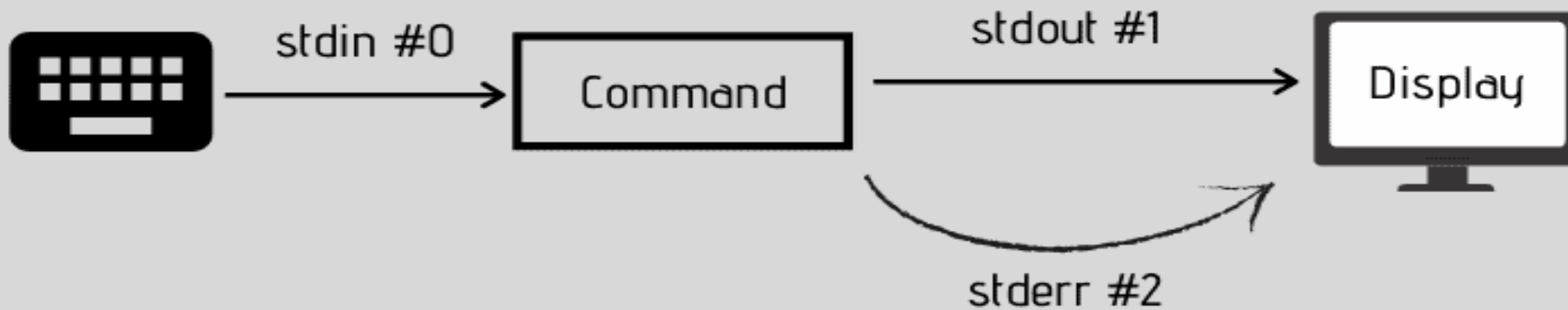
L'ordonnanceur

- 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



- 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

- 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 processus

- Commande ps
- permet de voir l'état des processus en cours d'exécution sur une machine

```
$ ps [option]
```


- Rarement utilisé sans option
 - Options:
 - -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 Kill

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

```
$kill PID  
$kill -9 PID
```

Redirection entrées / sorties

- Redirection = débrancher la connexion au terminal pour brancher l'entrée ou la sortie sur autre chose, par exemple un fichier.
- Sortie standard > Entrée standard
- Ce qui nous donne :
 - commande < fichier d'entrée -> Lit un fichier entrée
 - commande > fichier de sortie -> Créer/Ecrit un fichier en sortie
 - commande >> fichier de sortie -> Modifier un fichier en sortie

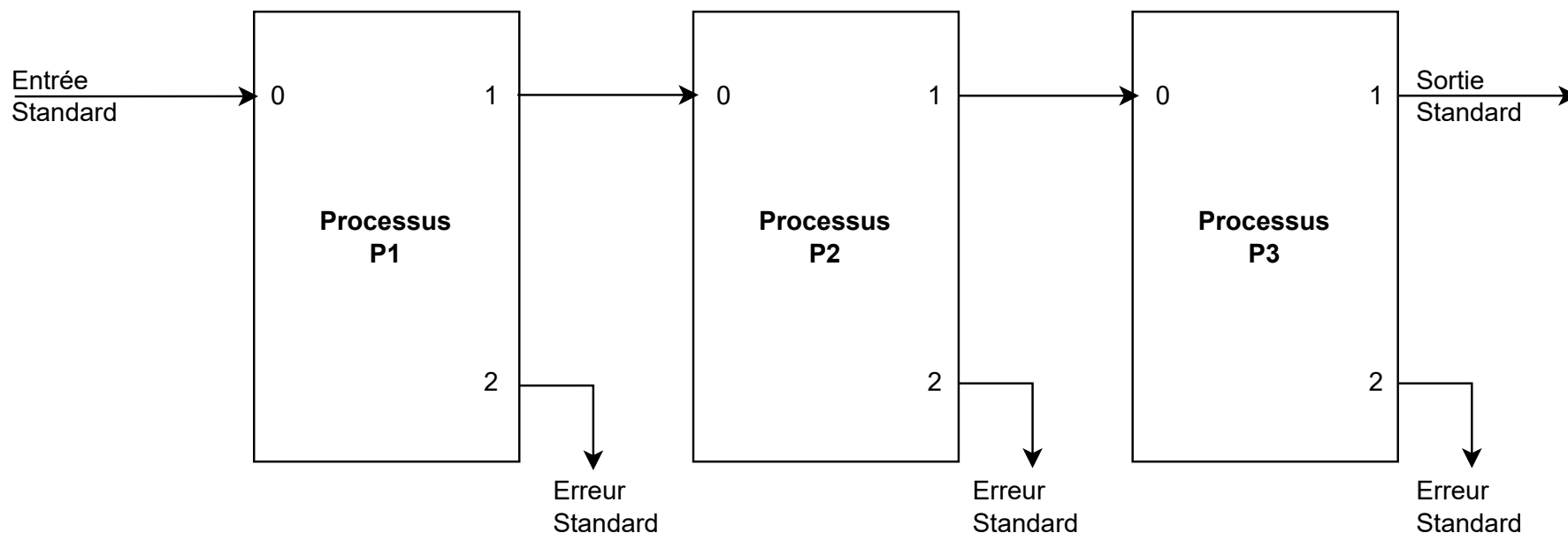
- Exemple:
 - cat < fichier (ou cat 0 < fichier) :
ici cat prend en entrée les données du fichier
 - cat > fichier (ou cat 1 > fichier)
ici cat crée/écrit dans fichier les informations qu'il retourne
 - cat 2 > fichier
ici cat crée/écrit dans fichier les erreurs qu'il retourne
 - cat < toto > titi 2 > tutu
cat lit les informations dans toto puis les écrit les informations dans titi et écrit les erreurs dans tutu

Exercice

1. Ecrire le message « bonjour tout le monde » dans un fichier appelé « test » en redirigeant la sortie de la commande echo.
2. Ecrire le message « au revoir » dans le même fichier « test » en redirigeant la sortie de la commande echo et sans écraser le contenu de « test » vérifier avec cat
3. exécuter la commande `find /etc -name hosts`, y a t-il des messages d'erreurs qui sont affichés? rediriger les messages d'erreur de la commande précédente vers le fichier « err.txt »
4. rediriger maintenant la sortie standard et la sortie d'erreur de la commande `find /etc -name hosts` vers deux fichiers différents (std.out et std.err)

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.



- 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

- 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.

Exercice

- la commande cat nous retourne
 - \$ 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 ?

Exercice

- On a un fichier Eleve.txt qui contient des informations sur les étudiants
 - 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'étudiants'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.

Gestionnaire de package

- Les paquets sous Debian ont une extension en .deb (**Debian**).
- Il y a un principe de gestion des dépendances.
- Les paquets sont regroupés dans des dépôts.
- **Un dépôt** : c'est le serveur sur lequel on va télécharger nos paquets.
- La gestion de paquet se fera grâce à « apt-get » (apt-cache) ou à « dpkg ».
- Avec dpkg (Debian package), les dépendances ne sont pas gérées.

- **apt-get update** : Cela va permettre de mettre à jour la liste des paquets présents dans le dépôt. A faire régulièrement.
- **apt-get install <package_name>** : installer un nouveau package sur votre ordinateur
- **apt-cache search <package_name>** : pour rechercher le paquet que nous voulons télécharger si nous ne connaissons pas son nom exact.
- **apt-get remove <package_name>** : Supprimer le paquet mais pas les dependances inutiles.

- **apt-get autoremove <package_name>** : Supprimer le package ainsi que les dépendances inutiles.
- **apt-get upgrade** : Cela met à jour les paquets installés et corrige aussi les failles de sécurité.

