

# Les systèmes Linux et le Shell

---

# Objectifs



Historique de Linux



Architecture de l'OS



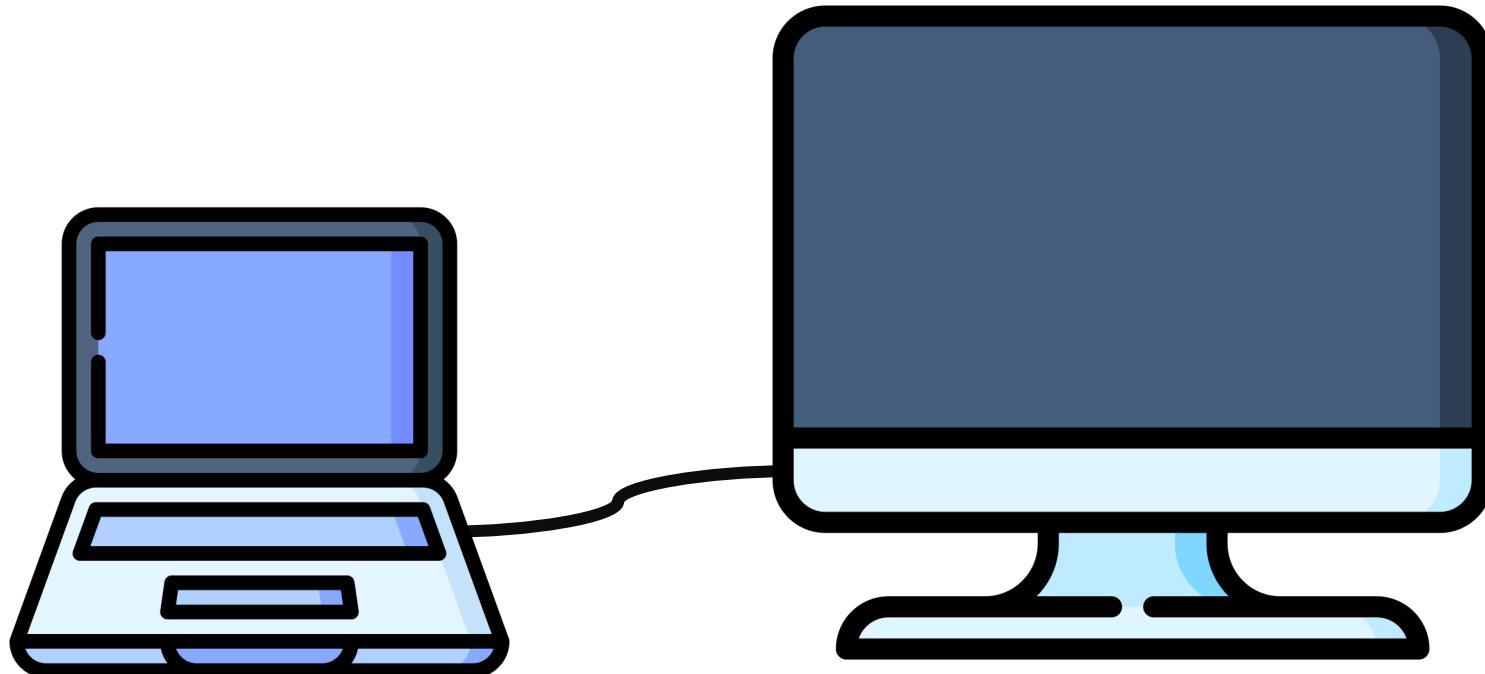
Fonctionnement d'un Shell

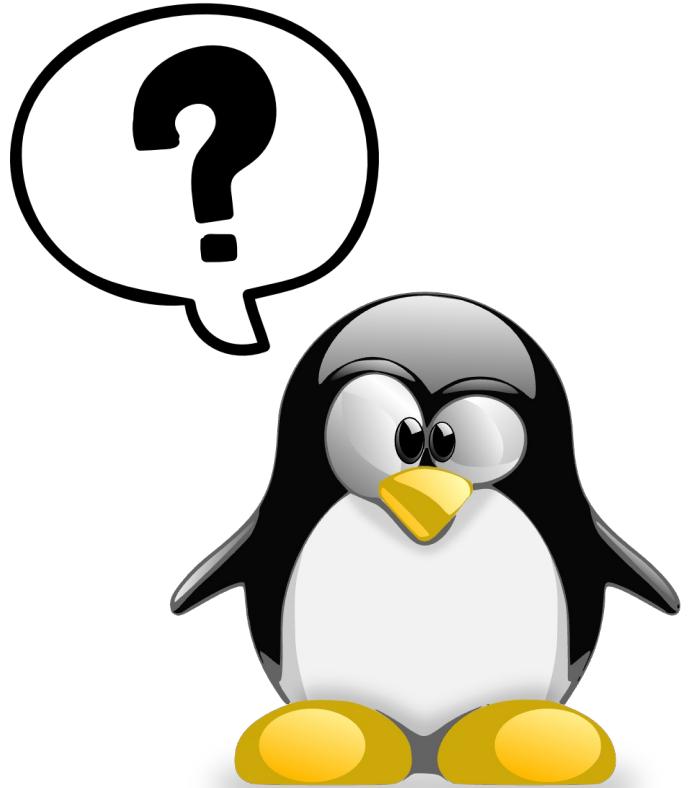


Commandes Shell principales

# Recommandation

---





# Qu'est ce que Linux ?

---

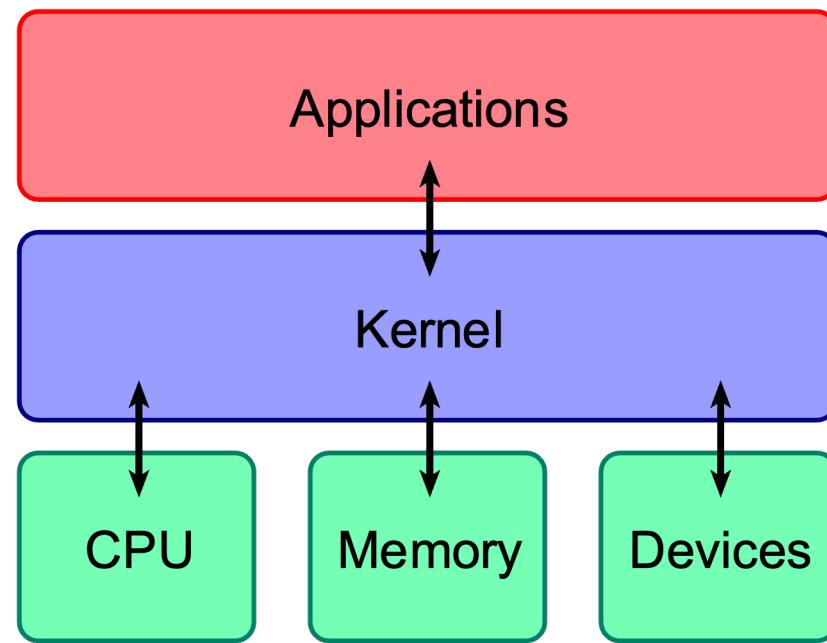


# Qu'est ce que Linux

---

Au sens strict : le Kernel Linux

Au sens large : OS basé sur un noyau Linux



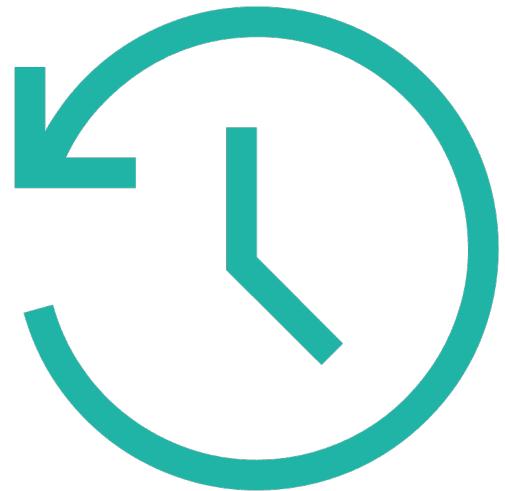
# Qu'est ce que Linux

---

Linux c'est ...

- OS de type “UNIX”
- Open Source
- Grosse implication de la communauté
- Prépondérance de la ligne de commande





# Histoire de Linux

---

# L'avant Linux : Unix

---

- UNIX, OS créé dans les labos Bell de AT&T dans les 70's
- Objectifs :
  - Développement logiciel modulaire et minimaliste
  - Avoir des programmes qui travaillent ensemble, et avec des flux de texte



- Initialement développé en assembleur par

- Ken Thompson

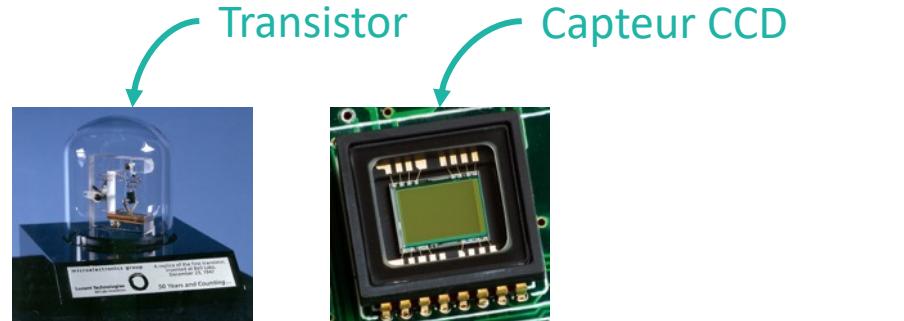
- Dennis Ritchie

} c } Créateurs du langage B

# Histoire d'Unix

---

- 1969 : création d'UNIX dans les labos Bell de AT&T



- 1970's: Bell obligé d'accorder des licences à qui demande

← Naissance de BSD, Xenix, SunOS

- 1983 : **Richard Stallman** annonce le projet **GNU**



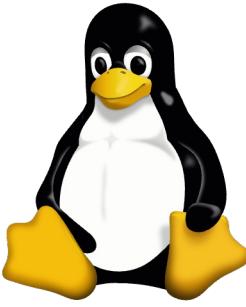
← C'est un Unix like  
**GNU is Not Unix**

- 1984 : AT&T se sépare de Bell qui peut vendre Unix (licence propriétaire)

← Naissance de Aix, Solaris

- 1985 : **Stallman** crée la FSF et publie **GNU GPL** (General Public License)

# Linux !

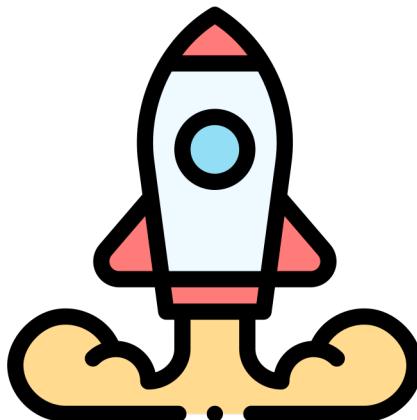


- 1991 : Linus Torvalds, frustré par MINIX, crée Linux
- 1991 : Les développeurs interfacent les composants GNU
- 1993 : Sortie de Debian et Slackware
- 2005 : Linus Torvalds crée GIT

Plus vieilles distro



# 1ère version publique de Linux



Newsgroup ! →

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)

Newsgroups: comp.os.minix

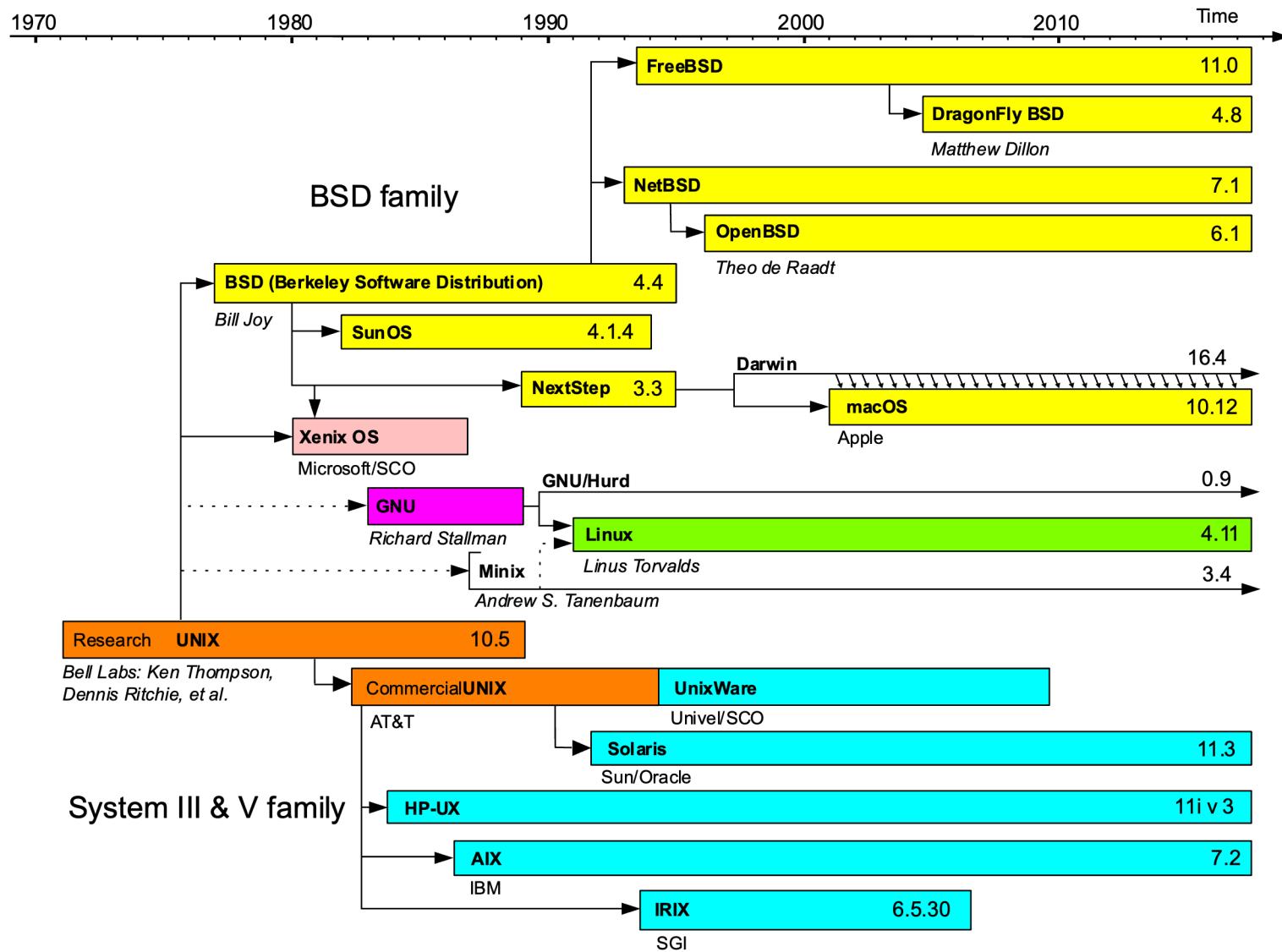
Subject: Free minix-like kernel sources for 386-AT

Date: 5 Oct 91 05:41:06 GMT

Do you pine for the nice days of minix-1.1, when men were men and wrote their own device drivers? Are you without a nice project and just dying to cut your teeth on a OS you can try to modify for your needs? Are you finding it frustrating when everything works on minix? No more all-nighters to get a nifty program working? Then this post might be just for you :-)

As I mentioned a month(?) ago, I'm working on a free version of a minix-lookalike for AT-386 computers. It has finally reached the stage where it's even usable (though may not be depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02 (+1 (very small) patch already), but I've successfully run bash/gcc.gnu-make/gnu-sed/compress etc under it.

Sources for this pet project of mine can be found at nic.funet.fi ← FTP !  
(128.214.6.100) in the directory /pub/OS/Linux. ...



# Généalogie Unix

Src: <https://en.wikipedia.org/wiki/Linux>

# Pourquoi s'intéresser à Linux ?

# Parts de marché

---

- Entre 1% et 3% d'après Net Applications, StatCounter, W3Counter (en 2018)
- Environ 8.5% d'après Developpez.com (en 2018)
- Environ 70% et 96% sur les serveurs d'après W3Techs et W3Cook (en 2015)
- 100% du top 500 des supercalculateurs !

# Architecture compatibles

---

DEC Alpha	IBM	PowerPC
ARM	Imagination META	RISC-V
Atmel	Intel Itanium	C
ETRAX CRIS	x86	rH
Texas Instruments TMS320 (DSP)	M32R	osys
68k	Microblaze	s+core
Fujitsu FR-V	MIPS architecture	Tilera
Qualcomm Hexagon	MN103 from Panasonic Corporation	Xtensa
PA-RISC family	OpenRISC	UniCore32
Renesas H8	Power ISA	...

# Plateformes compatibles variées

---



PC de bureau



Serveurs / Mainframe



Virtualisation



Systèmes embarqués

Jeux-vidéo (un peu)



Clusters / Supercalculateurs



Devices mobiles



# Faibles besoins matériels

---

## Debian 10

700 MB iso for install

256 MB RAM

10 GB HDD

1GHz Pentium 4



## Puppy Linux 8

250 MB iso for install

128 MB RAM

1 GB HDD

233Mz Pentium III



## OpenWrt 18

8 MB flash memory

64 MB RAM

No HDD required

SoC or processor

No GUI



## Windows 10 (32bits)

4 GB iso for install

1024 MB RAM

16 GB HDD

1GHz processor



# Publication sous GNU GPL



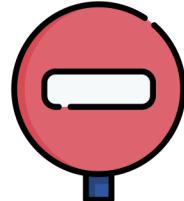
GPL = General Public Licence



Droit de modification / redistribution

programme qui utilise du GPL devient GPL

On peut vendre 1 produit/service



Pas le droit de modifier les termes du GPL

Pas le droit de mettre un copyright



Implication de la communauté

Sécurisation de plateforme

On peut tout modifier !

# Et plein d'autres avantages

---



Gratuité



Support du matériel âgé/exotique



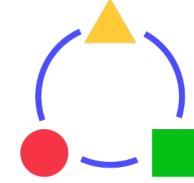
Privacy Policy



Sécurité (et pas par obscurité)



Fiabilité

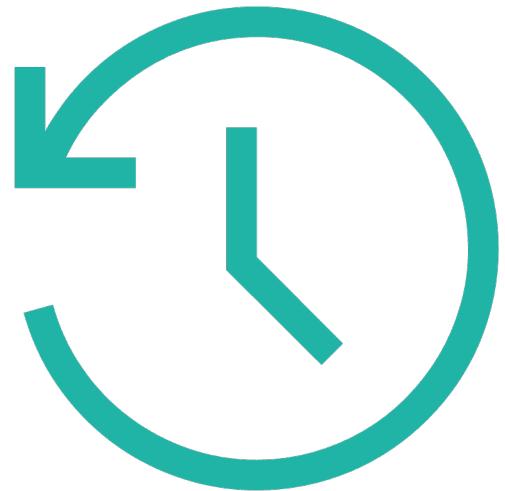


Des distros pour tous les goûts

# Des inconvénients aussi

---

- Manque de drivers pour certains hardware
- Manque de jeux (pas si anecdotique que ça)
- Manque de logiciels pro
- (Politique du changement)

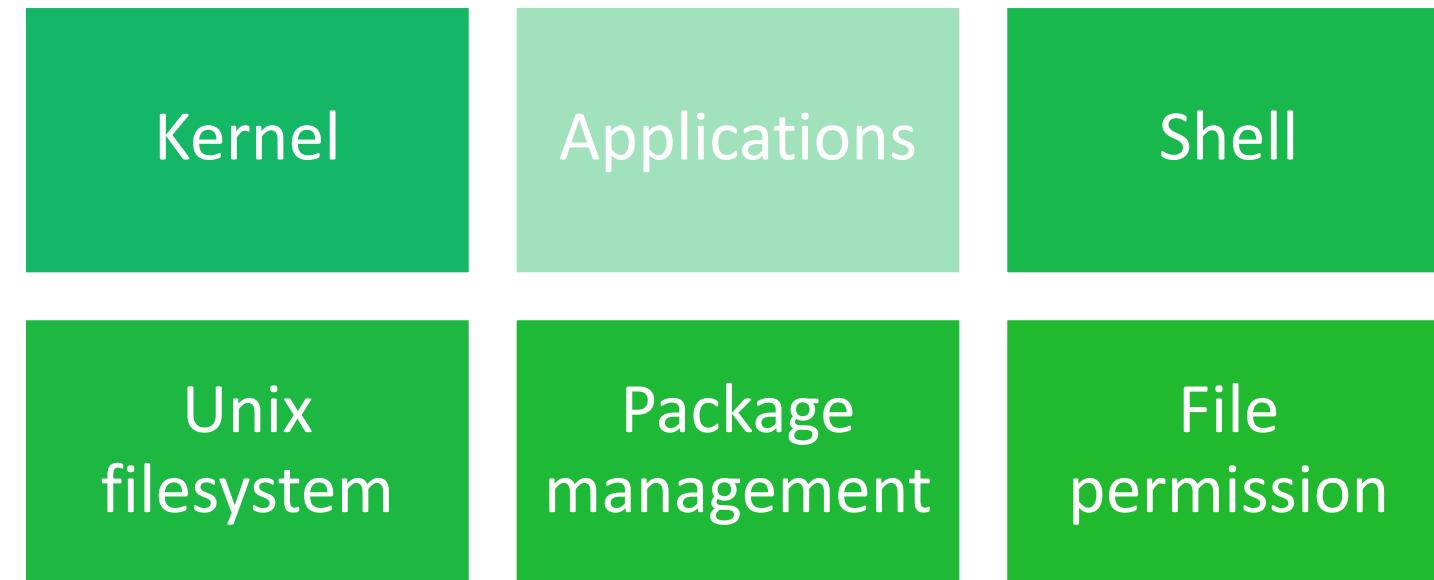


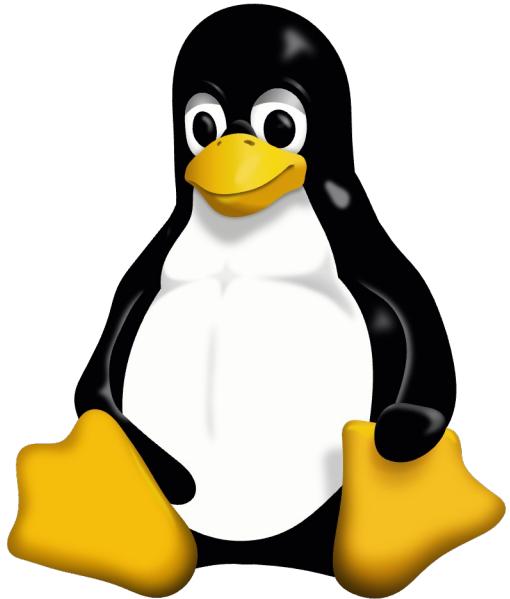
# Linux : ce qu'on va aborder

---

# Planning des fondamentaux

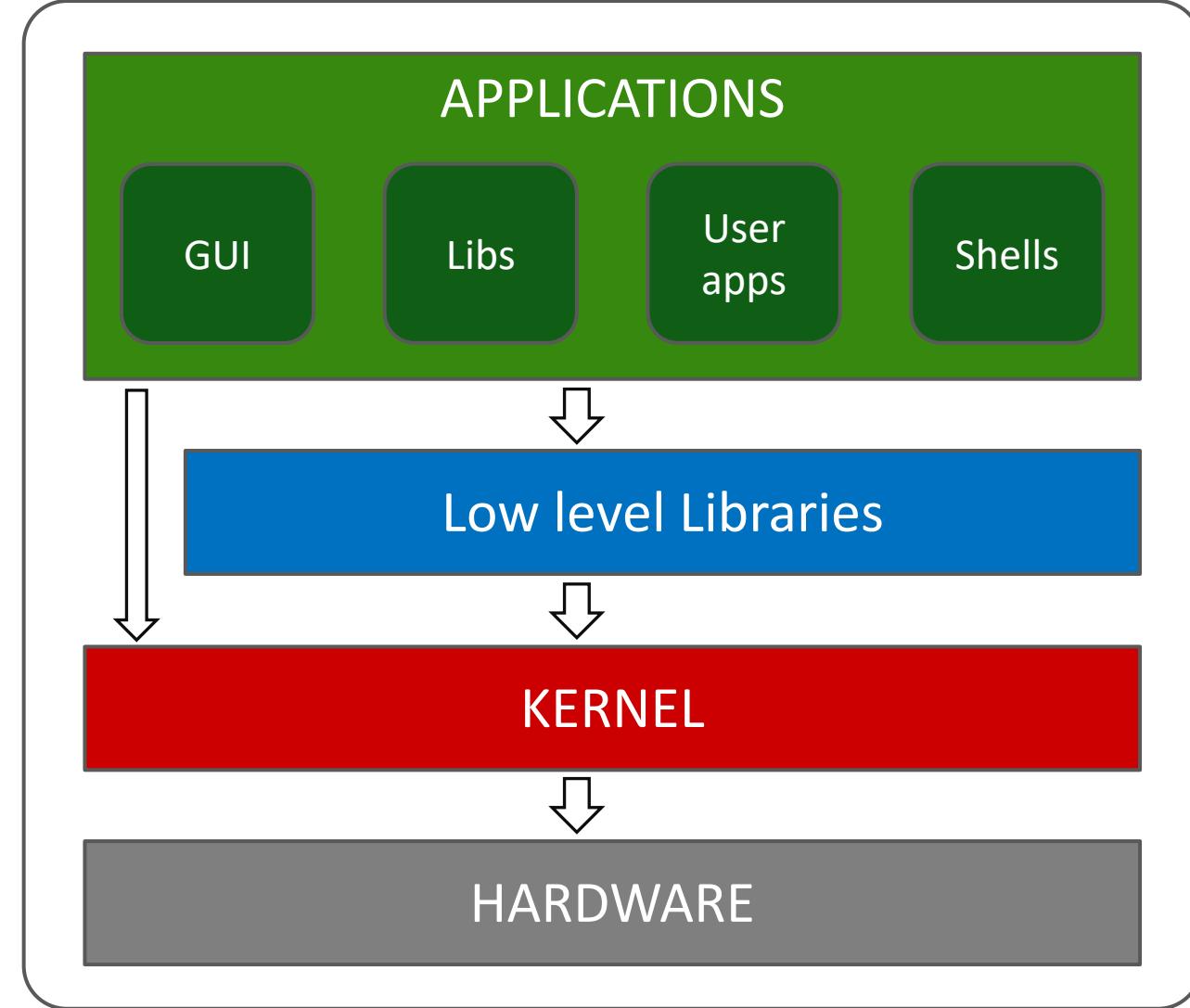
---





# Le Kernel Linux

---



# Architecture d'un OS

---

# Le Kernel Linux

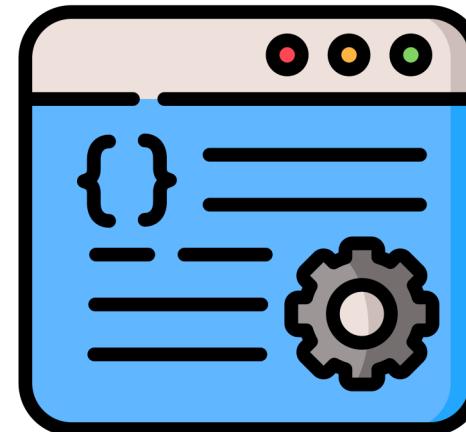
---



En 2020,  
commit n° 1.000.000 !

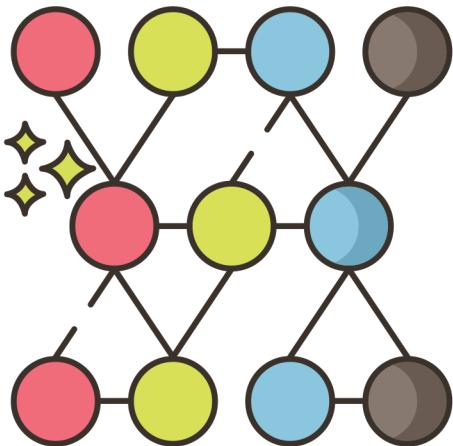
- Plus de 27 millions de lignes de codes
- Environ 5000 contributeurs & 75000 commits par an
- Structure monolithique mais modules qu'on peut load/unload

Ex : drivers



# Le Kernel Linux

---

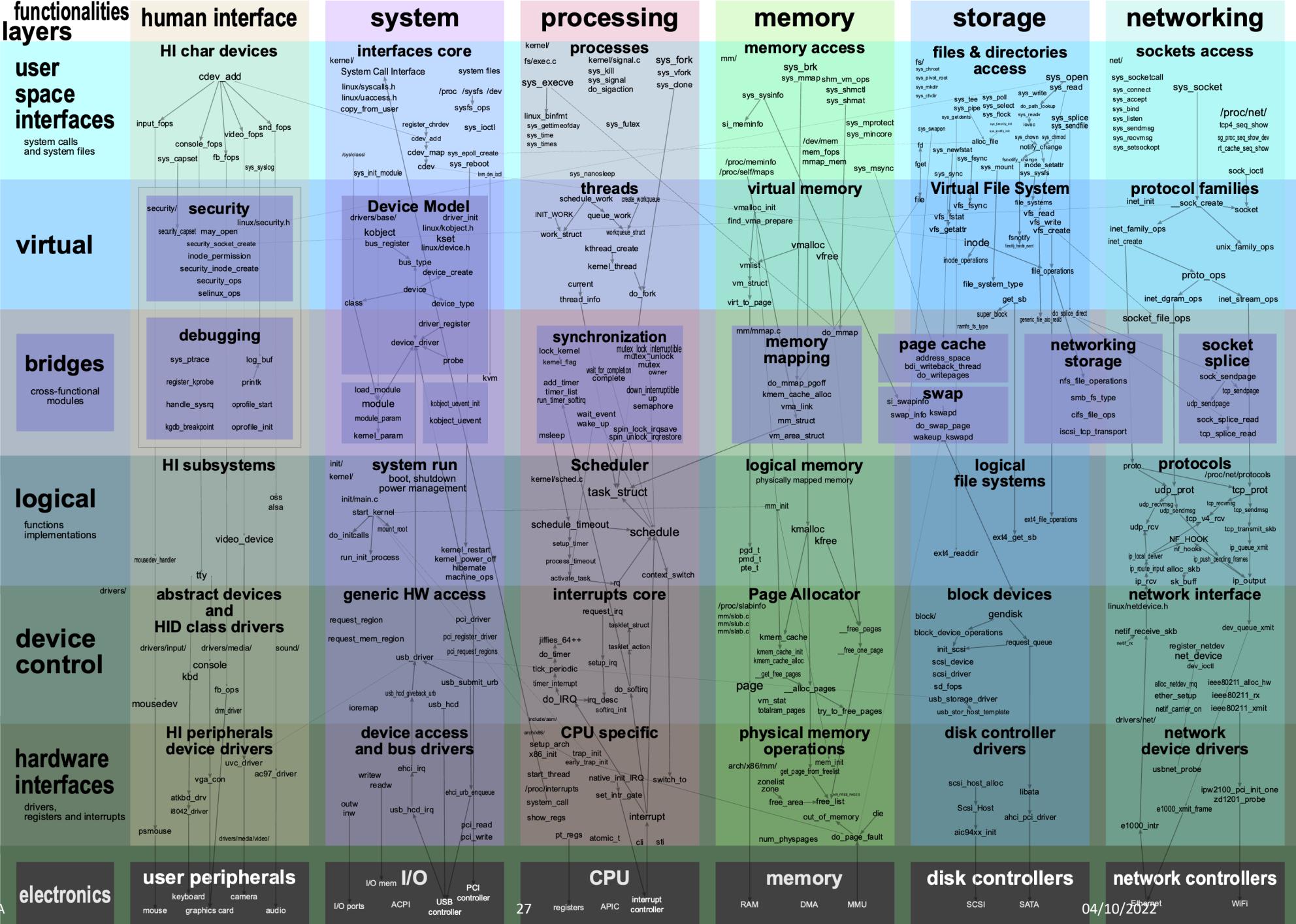


Gère

- Accès au matériel
- La mémoire
- Ordonnancement
- Le stockage
- Le réseau
- La sécu
- L'affichage
- ...



# Linux kernel map





# Les distributions

# Qu'est ce qu'une distribution ?

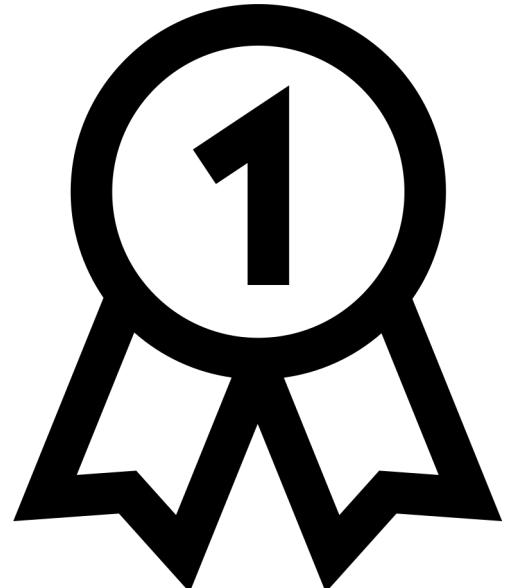
---



Kernel + Applications

# Distribution la plus populaire ?

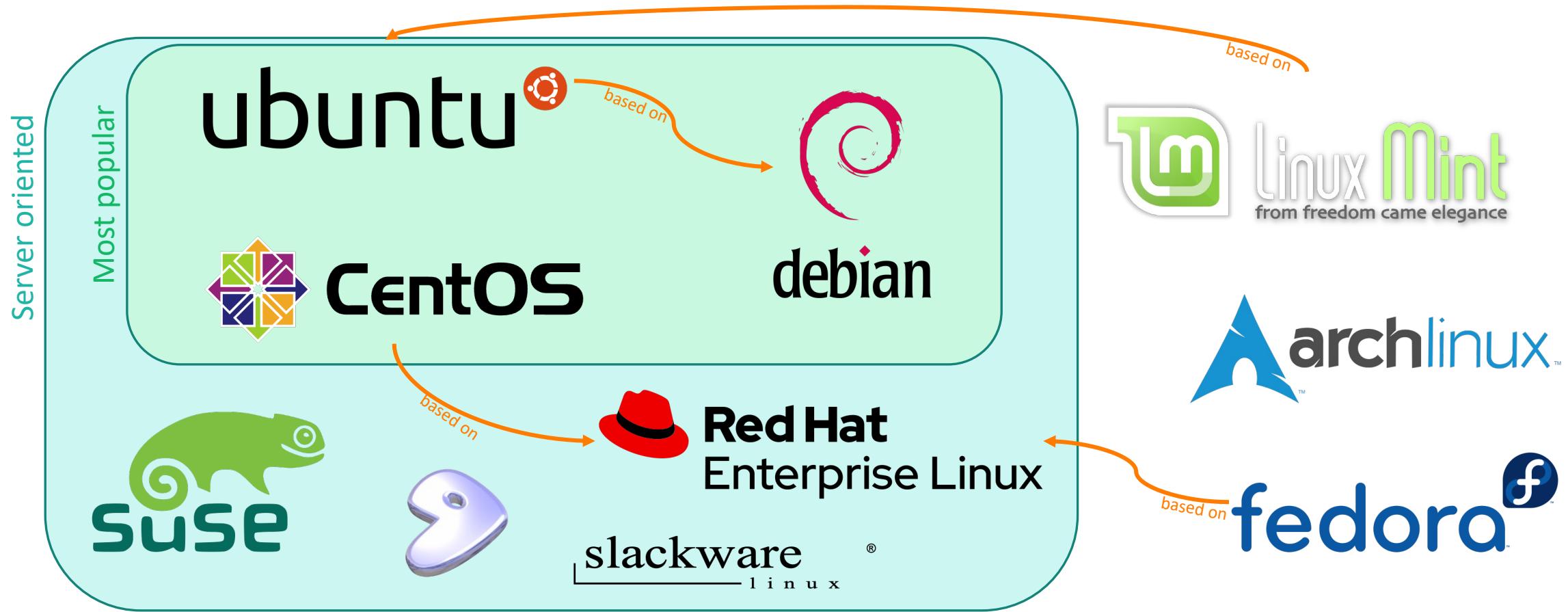
---



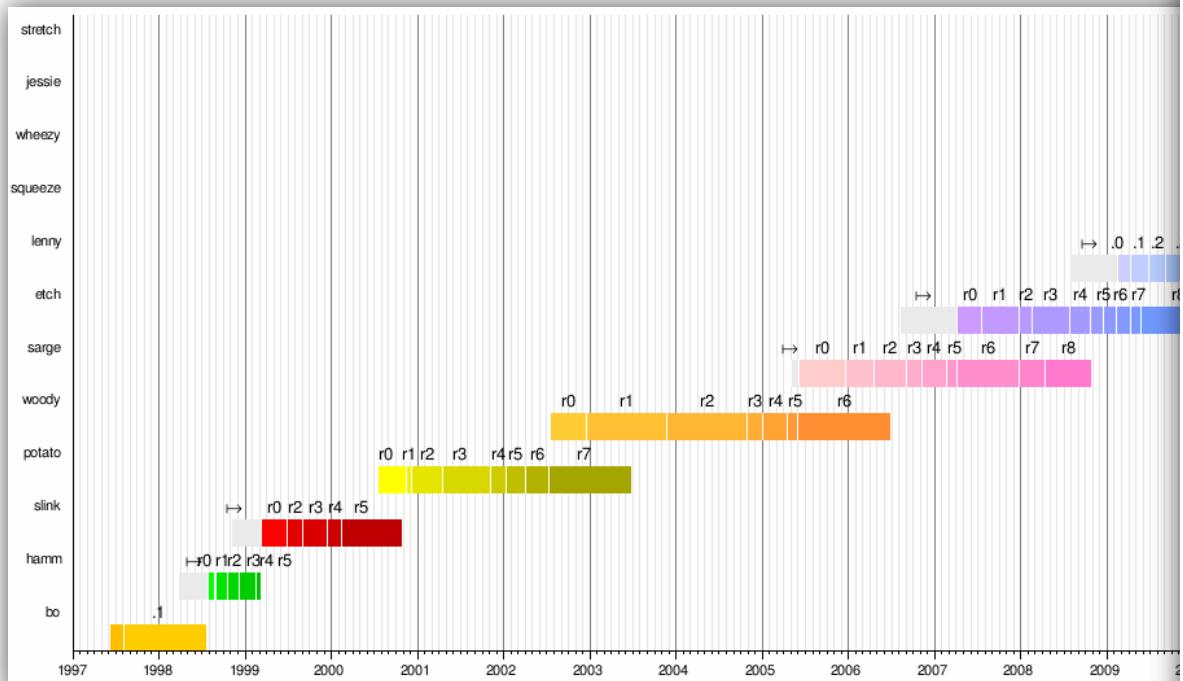
**android**

2.5 milliards de devices android actifs en 2021

# Distributions populaires

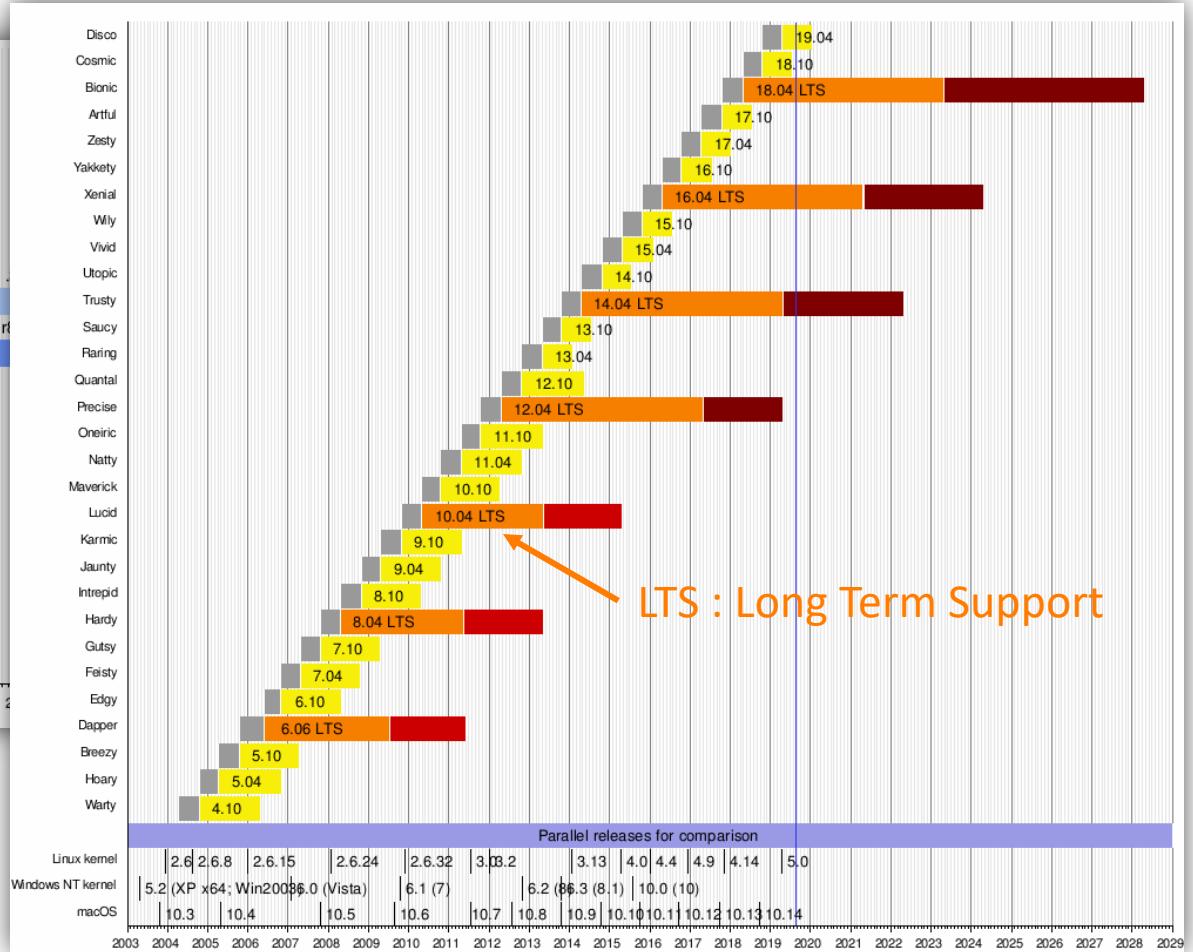


# Maintenance: importance de la distro



Debian

Ubuntu





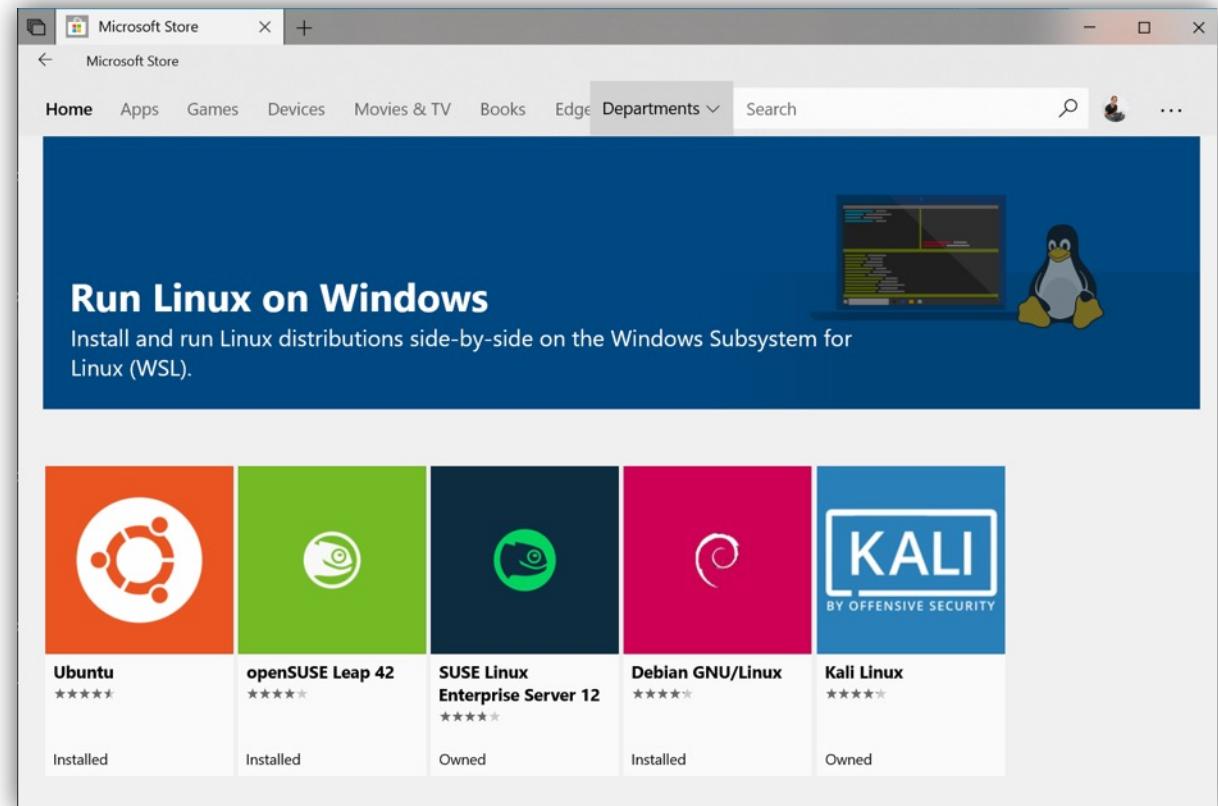
# WSL

---

# WSL

---

## Windows Subsystem for Linux



# WSL : installation

---

**Comment installer WSL ?**

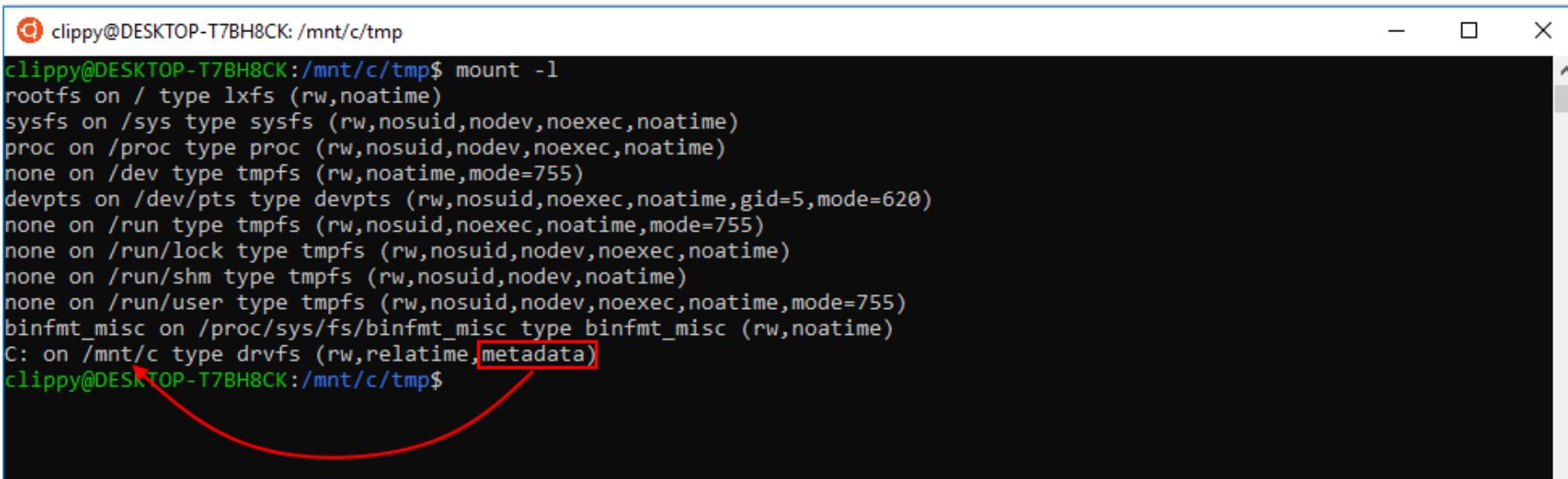
Depuis un terminal/powerShell lancé en mode admin

```
wsl --install -d Ubuntu
```

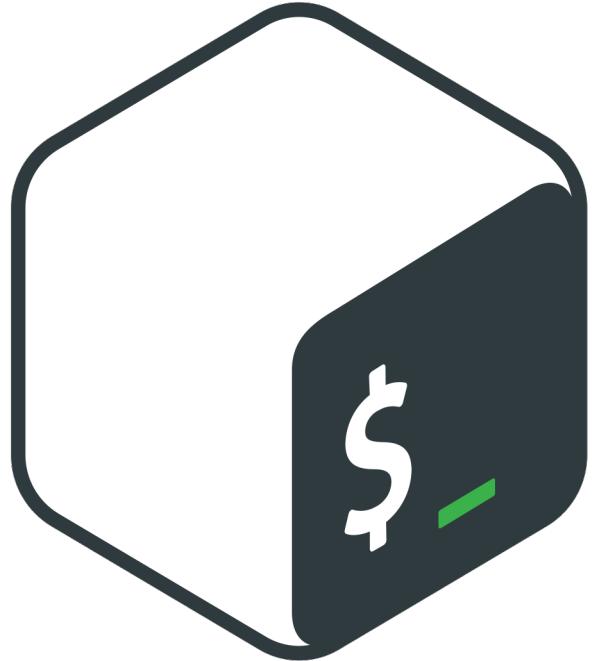
# WSL

---

- Partage automatique des lecteurs Windows
- On peut les afficher avec la commande « mount »



```
clippy@DESKTOP-T7BH8CK:/mnt/c/tmp$ mount -l
rootfs on / type lxfss (rw,noatime)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,noatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,noatime)
none on /dev type tmpfs (rw,noatime,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,noatime,gid=5,mode=620)
none on /run type tmpfs (rw,nosuid,noexec,noatime,mode=755)
none on /run/lock type tmpfs (rw,nosuid,nodev,noexec,noatime)
none on /run/shm type tmpfs (rw,nosuid,nodev,noatime)
none on /run/user type tmpfs (rw,nosuid,nodev,noexec,noatime,mode=755)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,noatime)
C: on /mnt/c type drvfs (rw,relatime,metadata)
clippy@DESKTOP-T7BH8CK:/mnt/c/tmp$
```



# Le shell, les bases

---

# Le shell

---

Terminal

+

Shell

=

Interaction avec l'OS

```
[root@localhost ~]# ping -q fa.wikipedia.org
PING text.pmpa.wikimedia.org (208.80.152.2) 56(84) bytes of data.
^C
--- text.pmpa.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 540.528/540.528/540.528/0.000 ms
[root@localhost ~]# pwd
/root
[root@localhost ~]# cd ./var
[root@localhost var]# ls -la
total 72
drwxr-xr-x. 18 root root 4096 Jul 30 22:43 .
drwxr-xr-x. 23 root root 4096 Sep 14 20:42 ..
drwxr-xr-x. 2 root root 4096 May 14 00:15 account
drwxr-xr-x. 11 root root 4096 Jul 31 22:26 cache
drwxr-xr-x. 3 root root 4096 May 18 16:03 db
drwxr-xr-x. 3 root root 4096 May 18 16:03 empty
drwxr-xr-x. 2 root root 4096 May 18 16:03 games
drwxrwx--T. 2 root gdm 4096 Jun 2 18:39 gdm
drwxr-xr-x. 38 root root 4096 May 18 16:03 lib
drwxr-xr-x. 2 root root 4096 May 18 16:03 local
lrwxrwxrwx. 1 root root 11 May 14 00:12 lock -> ../run/lock
drwxr-xr-x. 14 root root 4096 Sep 14 20:42 log
lrwxrwxrwx. 1 root root 10 Jul 30 22:43 mail -> spool/mail
drwxr-xr-x. 2 root root 4096 May 18 16:03 nis
drwxr-xr-x. 2 root root 4096 May 18 16:03 opt
drwxr-xr-x. 2 root root 4096 May 18 16:03 preserve
drwxr-xr-x. 2 root root 4096 Jul 1 22:11 report
lrwxrwxrwx. 1 root root 6 May 14 00:12 run -> ../run
drwxr-xr-x. 14 root root 4096 May 18 16:03 spool
drwxrwxrwt. 4 root root 4096 Sep 12 23:50 tmp
drwxr-xr-x. 2 root root 4096 May 18 16:03 yp
[root@localhost var]# yum search wiki
Loaded plugins: langpacks, presto, refresh-packagekit, remove-with-leaves
rpmpfusion-free-updates | 2.7 kB  00:00
rpmpfusion-free-updates/primary_db | 206 kB  00:04
rpmpfusion-nonfree-updates | 2.7 kB  00:00
updates/metalink | 5.9 kB  00:00
updates | 4.7 kB  00:00
updates/primary_db          73% [=====] 62 kB/s | 2.6 MB  00:15 ETA
```

# Le shell

---

- Shell = Command Line Interpreter
  - Exécute
    - soit « builtin »
    - soit « binaires » présents dans le \$PATH
  - Peut être lancé
    - de manière interactive dans un terminal
    - De manière non-interactive par le système
- Liste de dossiers où se trouvent les binaires sur le système

# Le shell

---

- On utilise le + souvent Bash
- Evolution de « Bourne Shell » créé par Stephen Bourne



# Le terminal : comprendre l'affichage

The diagram illustrates a terminal session with the following components:

- prompt**: The terminal prompt, indicated by a green arrow pointing to the first character of the command line.
- Dossier de travail actuel**: The current working directory, indicated by a green arrow pointing to the path prefix in the command line.
- Résultat d'exécution**: The output of the command, indicated by a green arrow pointing to the text displayed below the command line.
- commande**: The command being entered, indicated by a green box around the text `/etc`.
- argument**: The argument passed to the command, indicated by a green box around the text `echo hello`.
- Erreur d'exec de la cmd précédente**: An error message indicating a command not found, indicated by a green arrow pointing to the text `zsh: command not found: qsdfll`.
- Appui sur 'tab' qui autocomplete**: A note explaining the use of the tab key for autocompletion, indicated by a green arrow pointing to the partially completed path `/etc cd /etc/ssl/`.

```
/etc echo hello
hello
→ /etc qsdfll
zsh: command not found: qsdfll
→ /etc cd /etc/ssl/ ← Appui sur 'tab' qui autocomplete
ssh/ ssl/
```

# Le terminal : comment interagir avec

---

- « gauche | droite » pour se déplacer dans la saisie de la commande
- « haut | bas » pour naviguer l'historique des commandes
- CTRL+A : se déplacer en début la saisie de la commande
- CTRL+E : se déplacer en début la saisie de la commande
- CTRL+C : annule la saisie de la commande
- CTRL+D : quitte le shell

# Le shell : philosophie du scripting

---

- Il y a beaucoup de petites commandes unitaires
  - On peut enchaîner les traitements dans un script
  - On peut lier les commandes entre elles
-  Vitesse de traitement moindre que du code natif
-  Beaucoup de commandes paramétrables => on code rapidement

# Le shell : exécution d'une cmd

---

## Interprétation

Commande à exécuter ?

Arguments ? (grace à IFS)

Alias ?



## Exécution

Builtin ?

Binaire dans le \$PATH ?

```
~$ echo Hello
Hello
~$ echo Hello      World
Hello World
~$ echo "Hello      World"
Hello      World
```

# Le shell : code de retour

---

Commandes génèrent un code de retour pour savoir si le traitement est ok.

« \$? » permet de vérifier si une commande s'est bien exécutée.

\$? == 0 alors OK

```
~$ cat file
Hello world
~$ echo $?
0
```

\$? != 0 alors NOK

```
~$ cat filemissing
cat: filemissing: No such file or directory
~$ echo $?
1
```

```
~$ fakecommand
-bash: unrealcommand: command not found
~$ echo $?
127
```

# Le shell : les commandes pour survivre

```
$ ls
```

Lister les fichiers

```
$ echo string
```

Afficher un message

```
$ cat file
```

Afficher un fichier

```
$ cd dir
```

Aller dans un dossier

```
$ cd ..
```

Aller au dossier parent

Attention, modification des fichiers/dossiers

```
$ rm file
```

Supprimer un fichier

```
$ rmdir folder
```

Supprimer un dossier

```
$ mkdir folder
```

Créer un dossier

```
$ mv source target
```

Renommer

```
$ cp source target
```

Copier

```
$ touch file
```

Change la date d'accès  
d'un fichier (ou le crée)

# La plainte

---

On ne sait pas toujours comment va se comporter une commande

Mac OS 12

```
~$ echo "toto\n"  
toto  
  
~$ which echo  
echo: shell built-in command
```

Ubuntu 22.04

```
~$ echo "toto\n"  
toto\n  
~$ which echo  
/usr/bin/echo
```



Ca dépend de différents facteurs (souvent liés à la distro)

- Built-in VS binaire
- Shell utilisé
- Unix VS Linux

# Exo shell

---



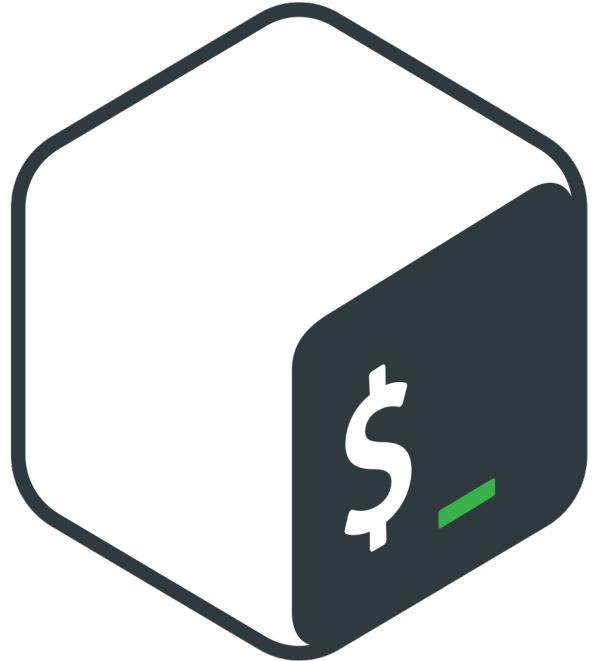
1. Allez dans le dossier "/etc"
2. Listez le contenu du répertoire
3. Affichez le contenu du fichier "shells" (fichier qui contient tous les shells disponibles sur votre machine)

# Exo shell spécial WSL

---



1. Allez dans le dossier "/mnt/c/Users/USERNAME/Desktop"
2. Créez un dossier "test-linux" »
3. Créez le fichier vide "readme.txt"
4. Depuis Windows, éditez le contenu du "readme.txt" »
5. Affichez le fichier modifié depuis le terminal



# Le shell : la suite

# Le shell : les variables

---

- Création d'un var très simple.
  - Nommage avec majuscule, minuscule, underscore
  - Variable systèmes toujours en majuscules
- 
- **\$VAR** pour accéder au contenu de la variable **VAR**
  - Remplacement d'une variable par son contenu = « parameter expansion »

```
$ my_var=hello  
$ echo $my_var  
hello
```

# Le shell : les variables

---

Il en existe plusieurs par défaut, les plus utiles dans l'environnement

```
$ env
```

Affiche les variables d'environnement

```
~$ env
SHELL=/bin/bash
WSL_DISTRO_NAME=Ubuntu
NAME=pc
PWD=/home/verdie_b
[...]
```

```
$ set
```

Affiche les variables locales du shell

C'est ici qu'on va trouver les variables  
qu'on créé depuis le terminal

# Le shell : « quoting »

---

**Quoting** : placer des protections pour que le Shell n'interprète pas certains caractères

Extrait du man Bash :

- Escape Character: How to remove the special meaning from a single character.
- Single Quotes: How to inhibit all interpretation of a sequence of characters.
- Double Quotes: How to suppress most of the interpretation of a sequence of characters.

# Le shell : démo « quoting »

---

```
~$ echo $SHELL  
/bin/bash  
~$ echo "$SHELL"  
/bin/bash  
~$ echo '$SHELL'  
$SHELL  
~$ echo \$SHELL  
$SHELL
```

\$ my\_var=hello world



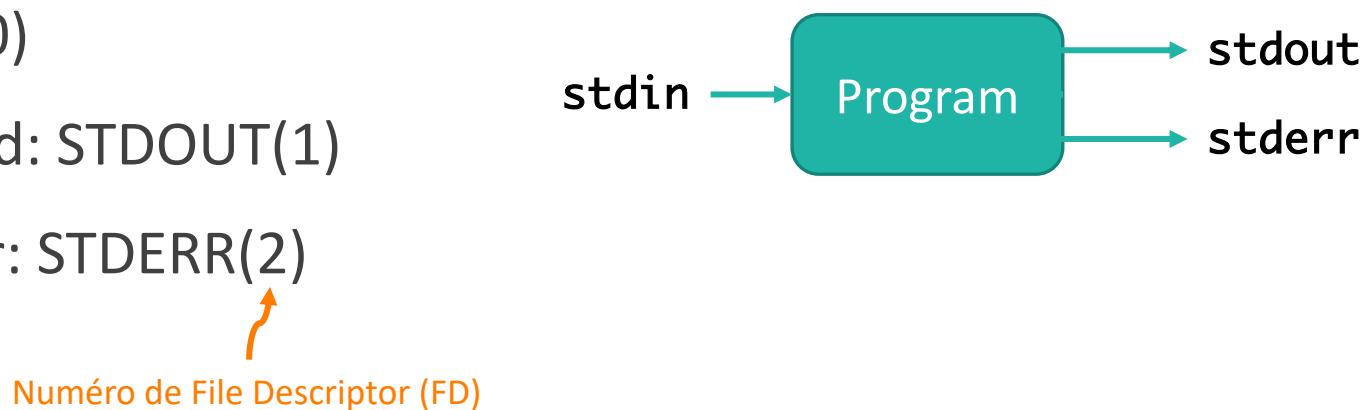
```
~$ my_var=hello world  
World: command not found
```

# Le shell : les flux de données standard

---

Chaque processus a 3 flux:

- un flux d'entrée: STDIN(0)
- un flux de sortie standard: STDOUT(1)
- un flux de sortie d'erreur: STDERR(2)



# Le shell : les flux de données standard

---

Flux STDOUT

```
~$ echo hello  
hello
```

Message sur STDOUT

Flux STDERR

```
~$ cat notfoundfile  
cat: notfoundfile : No such file or directory
```

Message sur STDERR

Flux STDIN

```
~$ cat
```

Lecture depuis STDIN

# Le shell : les redirections

---

- Les chevrons permettent rediriger vers/depuis des fichiers
- Chevron simple pour une redirection simple

Redirection simple de STDOUT

```
~$ echo foobar > f1
~$ cat f1
foobar
~$ echo toto > f1
~$ cat f1
toto
```

Création d'un fichier vide

```
~$ > myfile
```

Redirection simple de STDIN

```
~$ cat f1
toto
~$ cat < f1
toto
```

# Le shell : les redirections

---

Chevrons doubles pour une concaténation

Pour concaténer STDOUT

```
~$ echo -n foo > f1
~$ echo bar >> f1
~$ cat f1
foobar
```

# Le shell : les redirections

---

Pour rediriger STDERR, on spécifie explicitement le numéro de FD (STDERR=2)

```
~$ rm lmgdfkj > out.txt  
rm: cannot remove lmgdfkj ': No such file or directory  
~$ rm lmgdfkj 2> err.txt ← STDERR redirigé  
~$ ← Rien sur STDOUT  
~$ rm lmgdfkj 1> out.txt ← STDOUT redirigé mais c'est un msg STDERR  
rm: cannot remove lmgdfkj ': No such file or directory
```

Pour faire disparaître du texte avec /dev/null (c'est un pseudo device)

```
~$ echo foobar > /dev/null
```

# Exo shell

---



1. Affichez le contenu de la variable '\$SHELL' dans le terminal
2. Allez dans le dossier '/tmp'
3. Créez un dossier 'simpleTests' dans le dossier '/tmp' et allez dedans
4. Utilisez une redirection écrire le contenu de '\$SHELL' dans le fichier 'myCurrentShell' (tout en restant dans le dossier 'simpleTests')
5. Créez un dossier 'emptyFolder' dans le dossier 'simpleTests'
6. Tapez la commande `$ rm emptyFolder mycurrentShell`
7. Comme rm ne peux pas supprimer un dossier, trouvez la bonne commande.



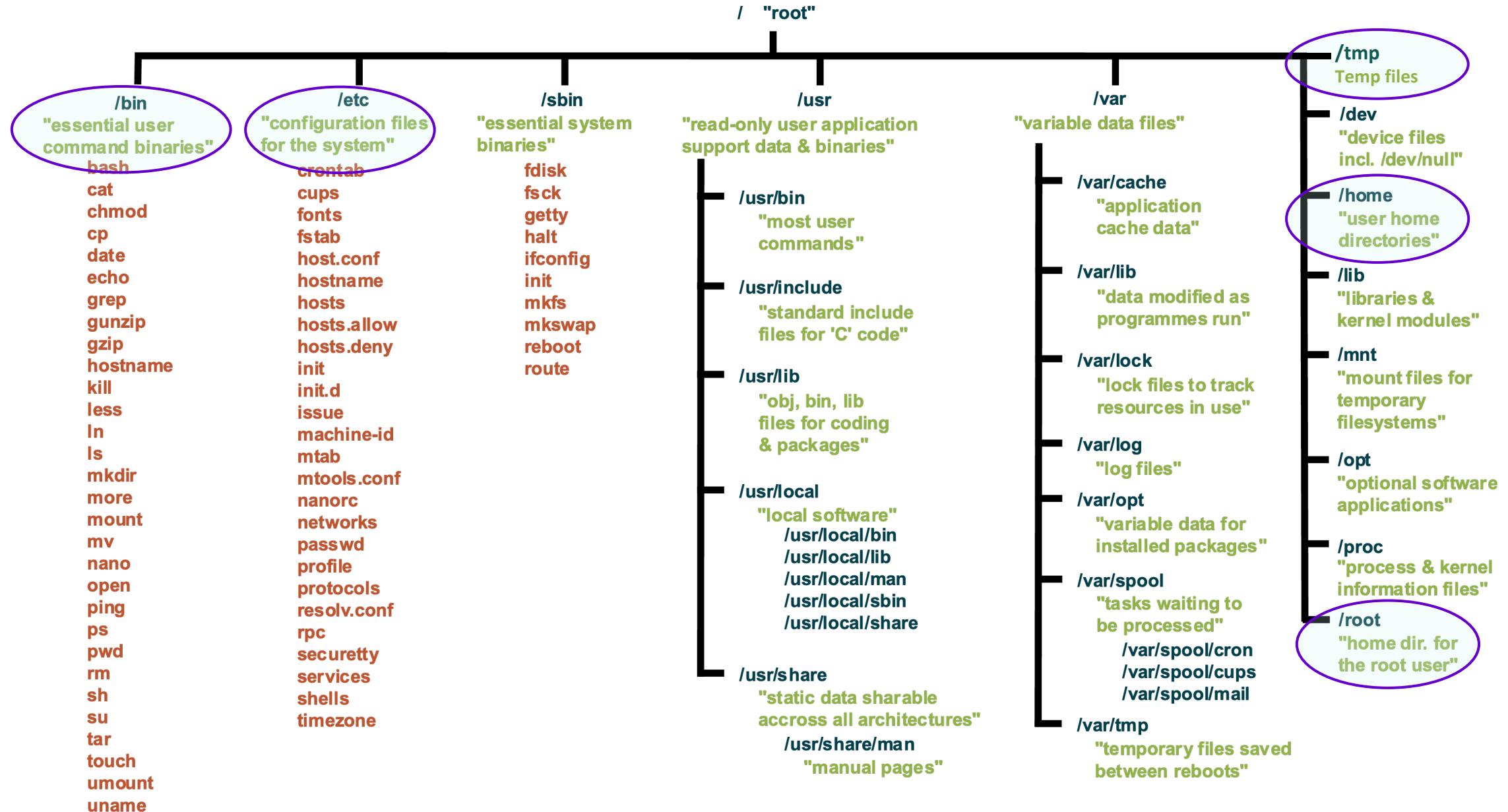
# Filesystem Hierarchy Standard

---

# Filesystem Hierarchy Standard

---

- Fichiers sont des composants centraux de l'OS dans Unix
- FHS (**Filesystem Hierarchy Standard**) est le standard Unix et Linux
- Tout est fichier
  - Les fichiers
  - Les partages réseaux
  - Les périphériques
  - Les processus
  - ...
- / est à la racine de tout



# Linux Processes

---

PID	USER	%CPU	%MEM
1	root	0.0	0.0
2	root	0.0	0.0
3	root	0.0	0.0

# Linux processes

---

- Tout programme qui s'execute a un **Process Identifier (PID)**
- Notion centrale dans Linux
- PID 0 = le scheduler, fait partie du Kernel
- PID 1 = process « init », premier à se lancer

# Linux processes

---

- Les infos sur les processus sont dans /proc
- proc est un pseudofilesystem ← Importance du système de fichier dans Linux

```
$ ls /proc  
1 10185 9276 9277 bus cgroups cmdline cpuinfo filesystems interrupts loadavg  
meminfo mounts net self stat sys tty uptime version version_signature
```

```
$ ls /proc/1  
attr auxv cgroup cmdline comm cwd environ exe fd gid_map limits maps  
mountinfo mounts mountstats net ns com_adj oom_score_adj root schedstat  
setgroups smaps stat statm status task uid_map
```

Ligne de commande

Lien vers le binaire

Lien vers les fichiers ouverts par le processus

# Linux processes : ps

---

ps [options]

\$ ps Affiche les process du user et rattachés à un tty

```
$ ps
  PID TTY      TIME CMD
 9277 tty1    00:00:00 bash
10190 tty1    00:00:00 ps
```

\$ ps aux Affiche tous les process avec plus d'infos

```
$ ps aux
USER        PID %CPU %MEM      VSZ      RSS TTY      STAT START      TIME COMMAND
root          1  0.0  0.0    8892     164 ?      Ssl  Aug30      0:00 /init
root        9276  0.0  0.0    8896      92 tty1      Ss  Aug31      0:00 /init
kuro        9277  0.0  0.0   15036    1524 tty1      S  Aug31      0:00 -bash
kuro       10191  0.0  0.0   16640    1776 tty1      R  11:42      0:00 ps aux
```

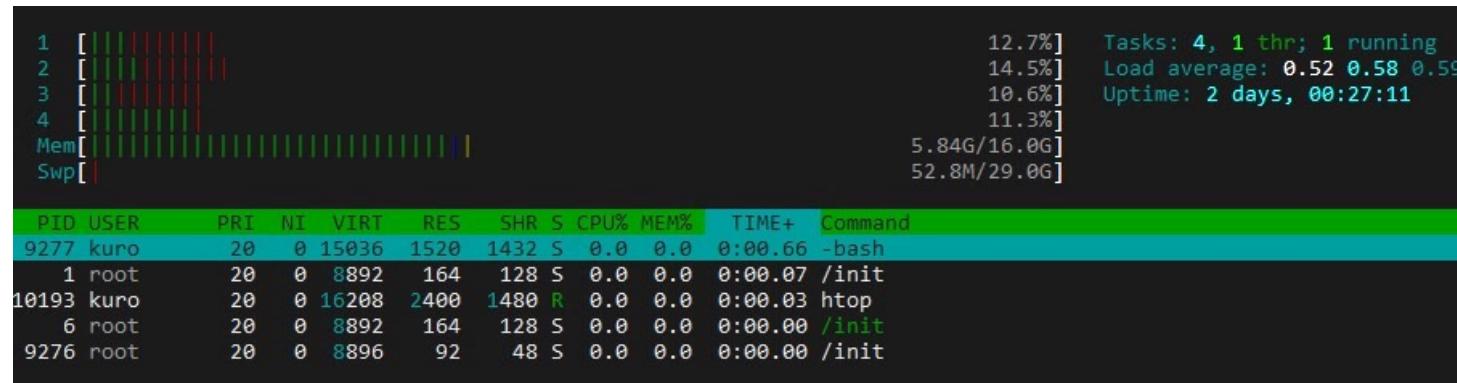
# Linux processes : (h)top

---

```
top - 11:53:24 up 2 days, 26 min, 0 users, load average: 0.52, 0.58, 0.59
Tasks: 4 total, 1 running, 3 sleeping, 0 stopped, 0 zombie
%Cpu(s): 9.9 us, 10.6 sy, 0.0 ni, 79.1 id, 0.0 wa, 0.4 hi, 0.0 si, 0.0 st
MiB Mem : 16332.8 total, 10165.1 free, 5943.7 used, 224.0 buff/cache
MiB Swap: 29673.9 total, 29621.1 free, 52.8 used. 10258.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	8892	164	128	S	0.0	0.0	0:00.07	init
9276	root	20	0	8896	92	48	S	0.0	0.0	0:00.00	init
9277	kuro	20	0	15036	1516	1436	S	0.0	0.0	0:00.66	bash
10192	kuro	20	0	16964	1992	1396	R	0.0	0.0	0:00.09	top

top



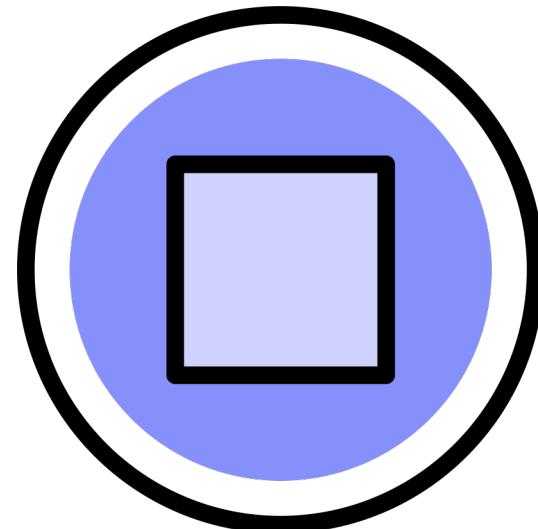
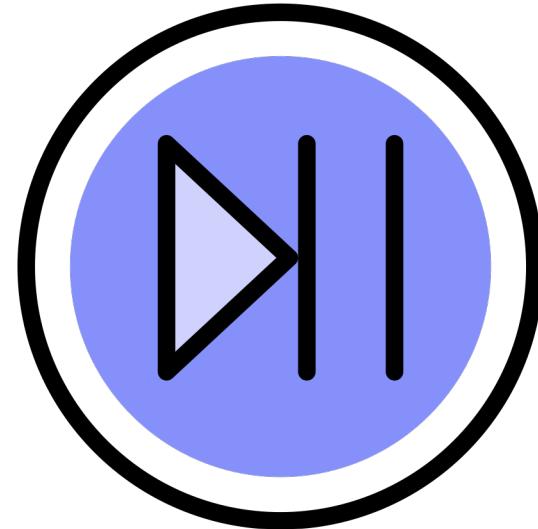
htop

## Raccourcis

- P : sort by CPU%
- M : sort by MEM%
- T : sort by TIME%

# Shell : contrôle de l'exécution

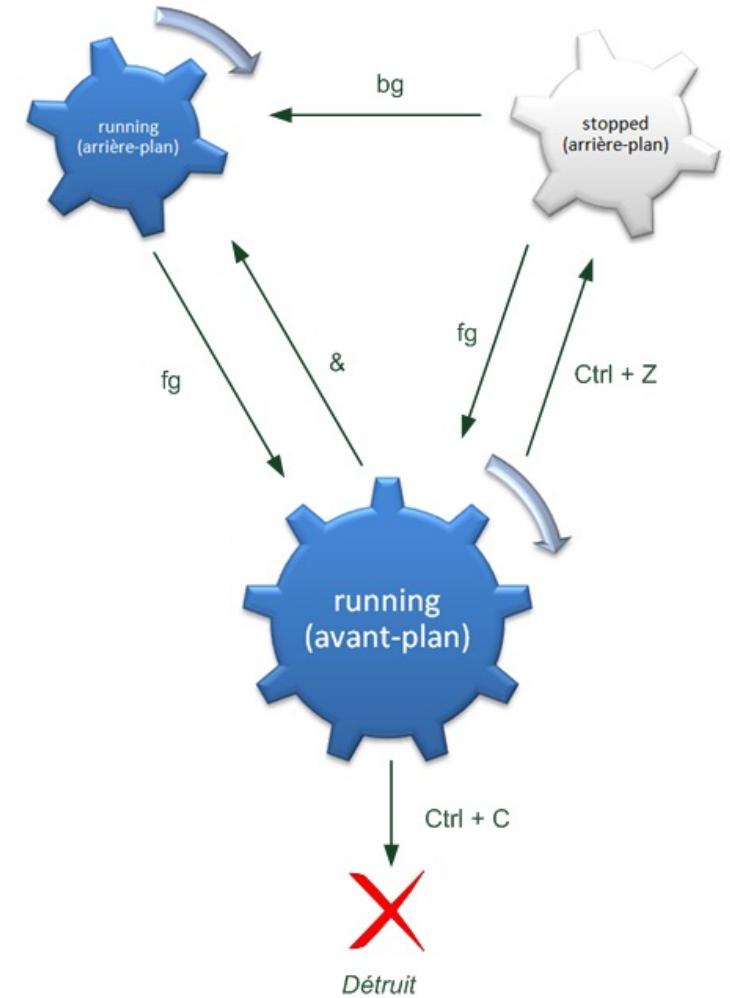
---



# Shell : contrôle de l'exécution

---

- Une commande shell tourne en fg par défaut
- CTRL+Z => tache se stoppe, il faut une action pour reprendre
- CTRL+C => envoi d'un signal d'interruption à la tache



# Shell : contrôle de l'exécution

---

```
$ sleep 10
```

On attend 10 secs  
=> pas d'autres interactions possible avec le shell

```
$ sleep 10 &  
[1] 10406  
$ jobs  
[1]+ Running sleep 10
```

Le process tourne en bg  
On a son PID  
On peut le voir dans les jobs

# Shell : contrôle de l'exécution

---

```
kill [-signal_number] pid ...
```

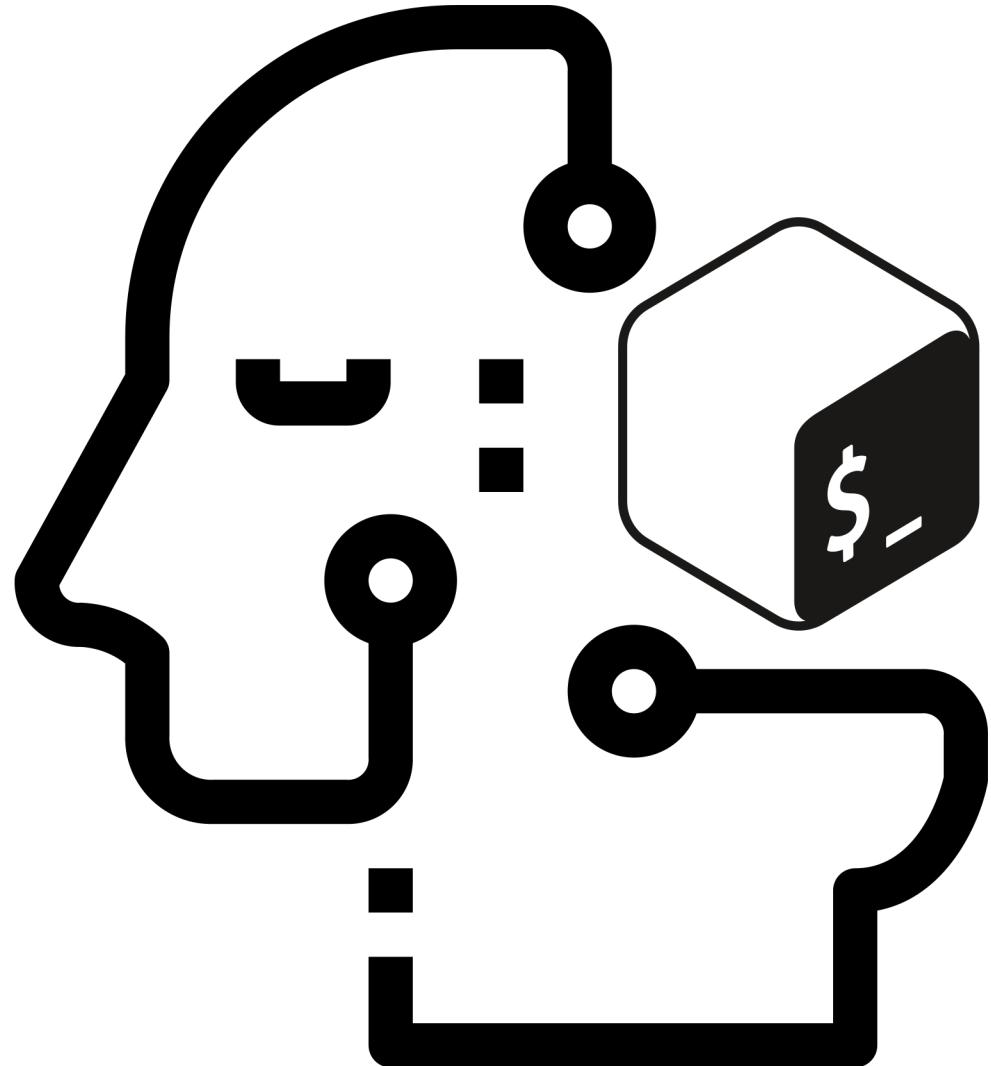
- Envoi des signaux à un process
- Il existe plusieurs signaux
  - SIGINT : ce qu'envoie le terminal quand on appuie sur CTRL+C
  - SIGTERM : demande de « termination » [DEFAULT]
  - SIGKILL : le programme se fait tuer par le système

# Shell : contrôle de l'exécution

---



1. Lancez un sleep
  2. Faites un CTRL+C => arrêt de l'exécution
- 
1. Lancez un sleep en background
  2. Récupérez l'id de process avec 'ps' et faites un 'kill' dessus



# Shell : les commandes de bases

---

# Shell : man

---

- « man » affiche les manuels
- Raccourcis :
  - « / » pour rechercher dans le man
  - « q » pour quitter

[man page](#)



# Shell : man

---

```
MAN(1)                                Manual pager utils                               MAN(1)

NAME
    man - an interface to the system reference manuals

SYNOPSIS
    man [man options] [[section] page ...] ...
    man -k [apropos options] regexp ...
    man -K [man options] [section] term ...
    man -f [whatis options] page ...
    man -l [man options] file ...
    man -wl-W [man options] page ...

DESCRIPTION
    man is the system's manual pager.  Each page argument given to man is normally the name of a program, utility
    or function.  The manual page associated with each of these arguments is then found and displayed.  A section,
    [...]
```

Souvent, aide simpliste depuis le binaire lui même

```
verdie_b@imac:~$ touch --help
Usage: touch [OPTION]... FILE...
[...]
```

```
verdie_b@imac:~$ man -h
Usage: man [OPTION...] [SECTION] PAGE...
[...]
```

# Shell : echo

Commande toute bête pour afficher, mais avec des subtilités

echo string

\$ echo -n

Pas de retour à ligne

```
verdie_b@imac:~$ echo toto  
toto  
verdie_b@imac:~$ echo -n toto  
totoverdie_b@imac:~$
```

\$ echo -e

Activation l'interprétation des  
char spéciaux avec backslash

```
verdie_b@imac:~$ echo "toto\n"  
toto\n  
verdie_b@imac:~$ echo -e "toto\n"  
toto
```

```
verdie_b@imac:~$
```

```
verdie_b@imac:~$ echo -e toto\n  
toton
```

?

# Shell : cat

---

Affiche le contenu d'un ou plusieurs fichiers  
(ou STDIN)

`cat [options] file ...`

`$ cat -n myfile`

Numérote les lignes

`$ cat -e myfile`

Affiche les « non-printing char » et un \$ en fin de ligne

# Shell : ls

- LS = LiSting des fichiers

ls [options] [file ...]

\$ ls Listing simple

```
Default
verdie_b@benoits-imac-5k:~$ ls
.bashrc#          Movies/
.gitconfig#        Music/
.gitignore_global# Pictures/
97VX972N5E.cer   Postman/
Adlm/             Projects/
Applications/    Public/
Applications (Parallels)/ Sites/
CityTesting/      StudioProjects/
Creative Cloud Files/ Web-visual.config
Desktop/          Web.config
Documents/         databases/
Downloads/        iCloud Drive (Archive)/
Extensions/       lfs/
```

\$ ls -a Liste aussi les fichiers cachés

```
Default
verdie_b@benoits-imac-5k:~$ ls -a
.bashrc#          .loadui/
.gitconfig#        .local/
.gitignore_global# .mono/
./                .mysql_history
../               .nbprofiler/
.AndroidStudio2.1/.netrc
.CF89AA64         .node-gyp/
.CFUserTextEncoding .node_repl_history
.DS_Store         .npm/
.DataGrip10/      .npm-packages/
.HIKCharacterizationTool6/.npmrc
.IdentityService/ .nuget/
.ServiceHub/      .omnisharp/
```

# Shell : ls

```
Default                               ⌂⌘5
verdie_b@benoits-imac-5k:~$ ls -l
total 120
-rw-r--r--  1 verdie_b  staff   1.4K Jan 14  2019 #.bashrc#
-rw-r--r--  1 verdie_b  staff   416B Feb 21  2014 #.gitconfig#
-rw-r--r--  1 verdie_b  staff   4.4K Mar  1  2016 #.gitignore_global#
-rw-r--r--  1 verdie_b  staff   1.4K May 24  2017 97VX972N5E.cer
drwxrwxrwx  4 verdie_b  staff  128B Aug  7  2017 Adlm/
drwx-----  5 verdie_b  staff  160B Jan 24  2019 Applications/
drwx-----  2 verdie_b  staff   64B Aug 18  2016 Applications (Parallels)/
drwxr-xr-x  6 verdie_b  staff  192B Feb  5  2019 CityTesting/
drwxrwxr-x@ 3 verdie_b  staff   96B Jun 10  09:28 Creative Cloud Files/
drwxr-xr-x@ 27 verdie_b staff  864B Aug 29 16:19 Desktop/
```

\$ ls -l

Liste avec les attributs

\$ ls dir

Listing du contenu du rép

```
Default                               ⌂⌘4
verdie_b@benoits-imac-5k:~$ ls Movies/
$RECYCLE.BIN/      desktop.ini
4K Video Downloader/
```

\$ ls -d dir

Listing des attributs du répertoire

```
Default                               ⌂⌘4
verdie_b@benoits-imac-5k:~$ ls -ld Movies
drwxr-xr-x+ 7 verdie_b  staff   224B Aug 28  2018 Movies/
```

# Shell : manip de fichiers/dossiers

---

- `touch` = créer des fichiers vides
  - `mkdir` = **MaKe DIRectory** : créer un répertoire vide
- 
- `rm` = **ReMove** : supprimer des fichiers et/ou dossiers
  - `rmdir` = **ReMove DIRectory** : supprimer un dossier vide
- 
- `mv` = **MoVe** : déplacer des fichiers et/ou dossiers
  - `cp` = **CoPy** : copie des fichiers ou dossiers
- 
- `cd` = **Change Directory** : pour changer de dossier courant
  - `pwd` = **Print Working Directory** : affiche le dossier actuel

# Shell : cd

---

\$ cd mydir

Change directory to mydir

\$ cd ..

Aller au dossier parent

\$ cd .

Aller au dossier actuel => neutre

\$ cd  
\$ cd ~

Aller vers la \$HOME du user

\$ cd -

Aller vers le précédent dossier

```
verdie_b@imac:~$ pwd  
/home/verdie_b  
verdie_b@imac:~$  
verdie_b@imac:~$ cd /tmp  
verdie_b@imac:/tmp$  
verdie_b@imac:~$ cd -  
/home/verdie_b  
verdie_b@imac:~$
```

# Shell : mkdir

---

```
mkdir [OPTION] directory ...
```

```
$ mkdir folder
```

Crée le dossier « folder »

```
$ mkdir -p folder/sub/subsub
```

Crée le dossier et ses sous dossiers  
si nécessaire

# Shell : rm

---

`rm [options] file ...`

`$ rm file1 file2`

Supprime les fichiers

`$ rm -rf folder`

Supprimer le dossier et tout son contenu

`$ rm -i file1 file2`

Suppression avec confirmation pour chaque élément

# Shell : mv

---

Déplacer des fichiers

`mv [options] source target`

```
$ touch file.txt  
$ mv file.txt readme.txt
```

**Cas standard simple**

Renomme « file.txt » en « readme.txt »

```
$ touch f1 f2  
$ mv f1 f2
```

Renomme f1 en f2, écrasant f2 au passage ☹

```
$ touch f1 f2  
$ mv -i f1 f2
```

Question interactive si écrasage

# Shell : mv

---

Fonctionne aussi avec les dossiers

```
$ mkdir d1  
$ mv d1 d2
```

Renomme le dossier d1 en d2

```
$ mkdir d1 d2  
$ mv d1 d2
```



Déplace le dossier d1 dans d2

# Shell : cp

---

- cp = CoPy
- Copie des fichiers ou des dossiers

```
cp [options] source target
cp [options] source ... target_dir
```

```
$ cp f1 f2
```

Copie f1 vers f2. Ecrase f2 s'il existe

```
$ cp f1 f2 mydir
```

Copie f1 et f2 vers mydir

```
$ cp -R d1 d2
```

Copie récursivement d1 vers d2.  
Crée d2 s'il n'existe pas, sinon copie d1 dans d2

```
$ cp -a d1 d2
```

Copie « archive » : récursivement en préservant tous les attributs

# Shell : Quizz fichiers/dossiers



```
$ mkdir d1  
$ mkdir d2/sub
```

cannot create dir, d2 no such file or directory

```
$ mkdir d1  
$ mkdir d1 d2
```

d1 already exists

```
$ touch f1 f2  
$ mv -i f1 f2
```

overwrite f2 ?

```
$ rm -rf /
```

Essaye de supprimer tout le système 😊

```
$ touch "hello world"  
$ FILE_TO_MOVE="hello world"  
$ mv $FILE_TO_MOVE /tmp
```



"hello : no such file" et "world : no such file" (bash)  
Ok sous Zsh



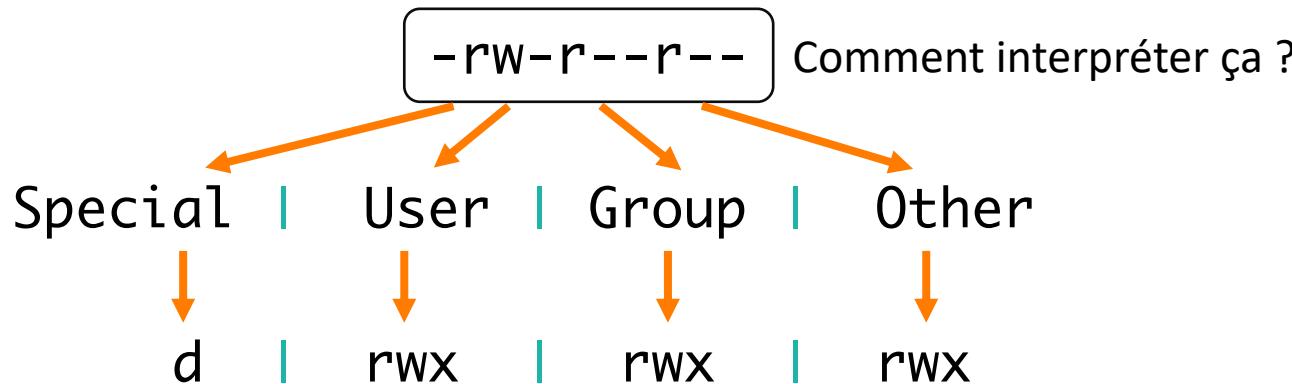
# Linux file permission

---

# Les permissions des fichiers

Droits présents dans le listing

```
→ removeTesting ls -l
total 8
-rw-r--r-- 1 benoit.verdier wheel 7 Jan 7 15:27 myfile
drwxr-xr-x 2 benoit.verdier wheel 64 Jan 7 14:59 test
```



R	W	X
Read	Write	Execute
4	2	1

# Les permissions des fichiers

---

	R	W	X
Signification	Read	Write	Execute
Valeur	4	2	1
Signification	Lecture du contenu du fichier	Modification du contenu du fichier	Droit d'exécution

# Shell : chmod (en absolu)

```
chmod [options] mode[,mode] file1 [file2 ...]
```

- chmod : **C**Change file **M**ODEs = change les autorisations d'accès aux fichiers/dossiers
- Les droits sont donnés par 3 : User Group Other
- chmod -R : pour changer en récursif

Mode « absolute » = on redonne tous les droits

```
$ chmod 644 myfile
```

User	Group	Other
6 = 4 + 2	4	4
RW-	R--	R--

R	W	X
Read	Write	Execute
4	2	1

# Shell : chmod pour un dossier

	R	W	X
Signification	Read	Write	Execute
Valeur	4	2	1
Signification	Listing du contenu	Modification des fichiers dans le dossier (add, delete, rename )	Dossier traversable et contenu accessible

Ce qu'on peut faire ensuite sur les fichiers des dossiers dépend de leurs droits respectifs



```
$ chmod 500 folder
```

On peut juste lister le contenu du dossier. Impossible de créer un fichier/dossier

```
$ chmod 300 folder
```

Boîte noire : on peut modifier les fichiers mais sans pouvoir lister le contenu du dossier

# Shell : chmod (en relatif)

```
chmod [references][operator][modes] file1 ...
```

Mode « symbolic » = on ajuste de manière relative les droits

Ex : ajouter les droits en lecture au groupe

```
$ chmod g+r f1
```

Reference	Class	Description
u	user	file owner
g	group	members of the file's group
o	others	users who are neither the file's owner nor members of the file's group
a	all	all three of the above, same as 'ugo'

Operator	Description
+	adds the specified modes to the specified classes
-	removes the specified modes from the specified classes
=	the modes specified are to be made the exact modes for the specified classes

```
$ chmod u+r f1
```

```
$ chmod go-x f1
```

# Shell : Quizz chmod



```
$ chmod 750 script.sh
```

User : RWX  
Group : R - X  
Other : ---

```
$ mkdir mydir  
$ echo salut > mydir/readme  
$ chmod -R 600 mydir
```

600 en récursif, y compris sur le dossier  
=> On ne peut plus rentrer dans son propre dossier

```
$ chmod 644 file1
```

User : RW-  
Group : R --  
Other : R --

```
$ chmod -R a+r mydir
```

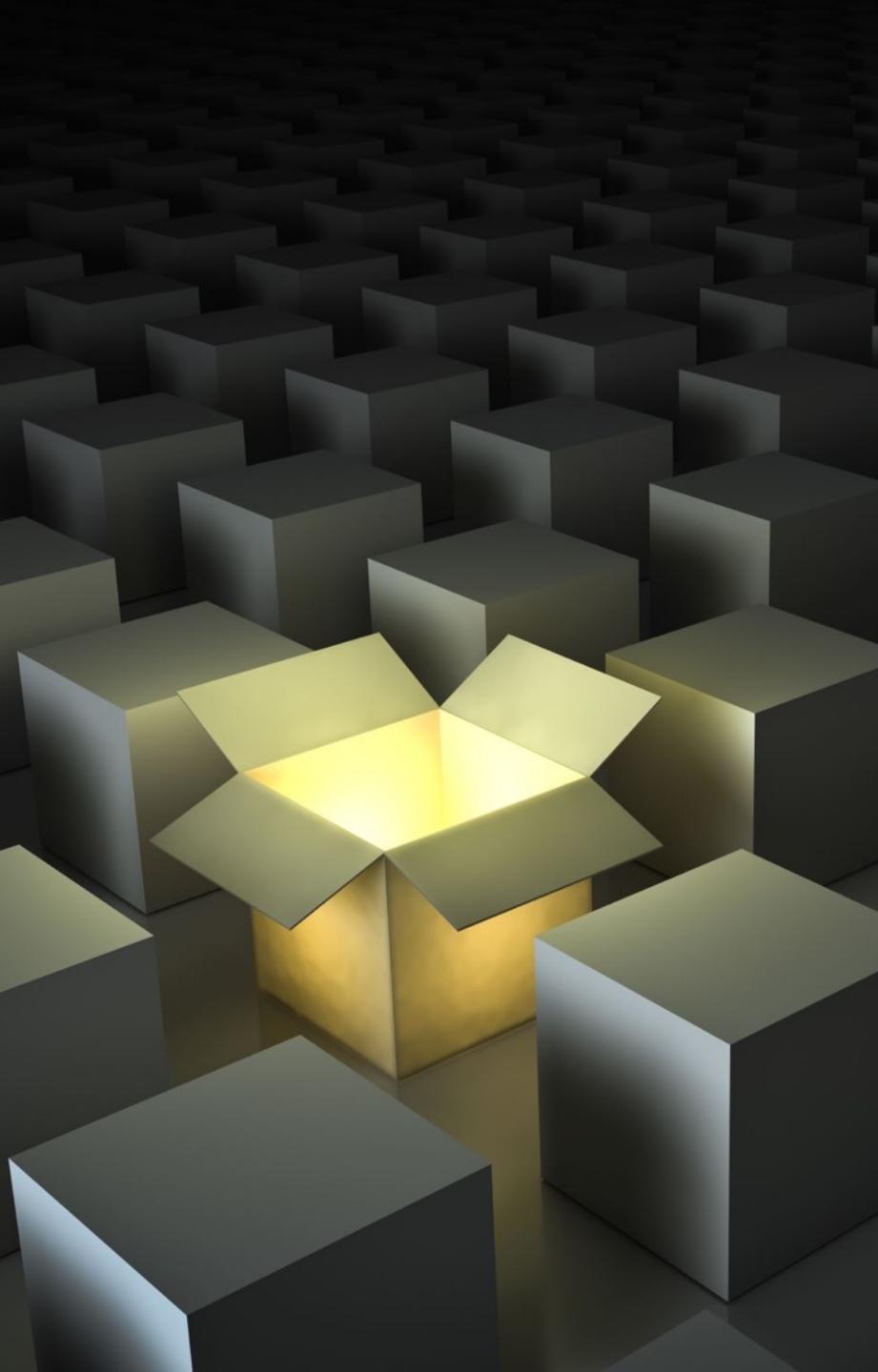
Tout le monde : ajout du *read* en récursif

# Exo shell

---



1. Créez un dossier « projects » et allez dedans
2. Avec la commande 'echo' et une redirection, créez un fichier « readme » avec du contenu
3. Modifiez les droits du fichier « readme » pour le passer en lecture seule pour vous et rien pour les autres
4. Essayez d'afficher (doit fonctionner) et supprimer (ne doit pas fonctionner) le fichier readme
5. Supprimez tous les droits sur votre fichier readme. Il ne doit plus être affichable ni modifiable (comme vous êtes le proprio vous pouvez les remettre ensuite)



---

# Package management

# Package management

---



- « **dpkg** » = **debian package** : outil pour installation d'un package
- « **apt** » = **advanced package tool**: gestionnaire de paquets (collection d'outils en fait)
- apt utilise une source de liste de paquets (`/etc/apt/sources.list`)
- apt gère les dépendances

# Apt : commandes principales

---

apt update	: met à jour la liste de paquets
apt upgrade	: met à jour le système
apt dist-upgrade	: met à jour le système en prenant en compte les dépendances
apt search pkg	: cherche un paquet
apt show pkg	: affiche des infos sur un paquet
apt remove pkg	: supprimer un package
apt install pkg	: installe un paquet et ses dépendances

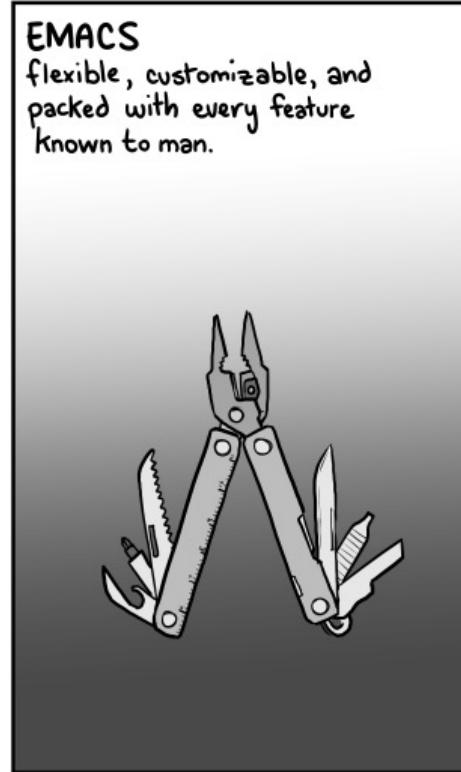


# Les éditeurs textes

---

# Les éditeurs textes (en mode console)

---



# Les éditeurs textes : emacs



- Il existe depuis 1976
- Variante GNU Emacs la + populaire
- Coloration syntaxique, mode graphique, customisation poussée, extensions

Shortcut	Command
Ctrl+X, Ctrl+S	Save
Ctrl+X, Ctrl+C	Quit
Ctrl+K	Cut line after prompt
Ctrl+Y	Paste
Ctrl+S	Incremental search
Ctrl+/ ou Ctrl+_	Undo

Shortcut	Command
Ctrl+A	Go to line start
Ctrl+E	Go to line end

# Editeur texte avec GUI

---

- Linux en mode graphique
  - Windows
  - Mac
- } Il y a des éditeurs de code user friendly !

⚠ Placez les fichiers à un endroit accessible

⚠ Attention aux types de retours à la ligne

# Exo

---



1. Installez un éditeur texte (s'ils ne sont pas déjà tous présents)
2. Testez la création et sauvegarde d'un fichier

```
$ sudo apt update  
$ sudo apt install emacs-nox  
$ cd  
$ emacs toto.txt
```

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

# Shell scripting

---

# Shell scripting

---

```
$ echo '#!/bin/bash' > script.sh
$ echo 'echo hello' >> script.sh
$ chmod 700 script.sh
$ ./script.sh
hello
```

Il faut les droits d'exec

- « #! » est appelé « shebang » (**S**Harp **B**ANG ou **h**a**S**H **B**ANG)
- Shebang permet aux systèmes Unix de savoir quel interpréteur exécuter

# Shell scripting

---



Faites un (mauvais )script qui affiche « hello world » et débugger le.

Pensez à ajouter les droits d'execution.

On peut afficher ce que fait le shell avec « set -x » => pratique pour le débug

Fichier script.sh

```
#!/bin/bash
set -x

MSG=hello world
echo $MSG
```

```
$ ./myscript.sh
+ MSG=hello
+ world
./myscript.sh: line 3: world: command not found
+ echo
```



# Unix Pipeline

---

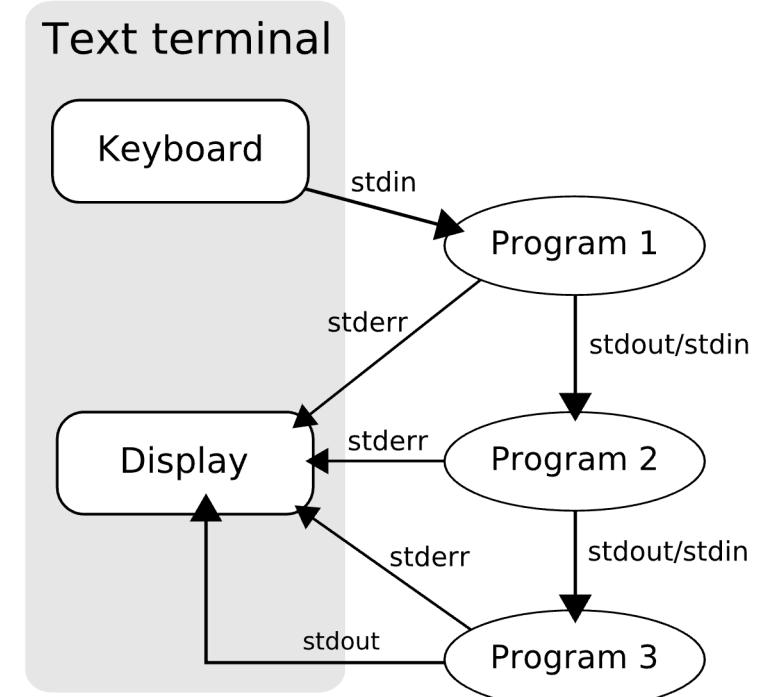
# Unix pipeline

- Mécanisme de communication inter-processus
- Les processus sont connectés par leurs flux standards (stdin,stdout,stderr)

process1 | process2 | process3

STDOUT du process n-1 devient STDIN du process n

STDERR est affiché normalement



# Unix pipeline : un exemple

---

```
$ ls /dev > output  
$ wc -l output  
210 output
```



```
$ ls /dev | wc -l  
210
```

Preuve que 'wc' accepte stdin

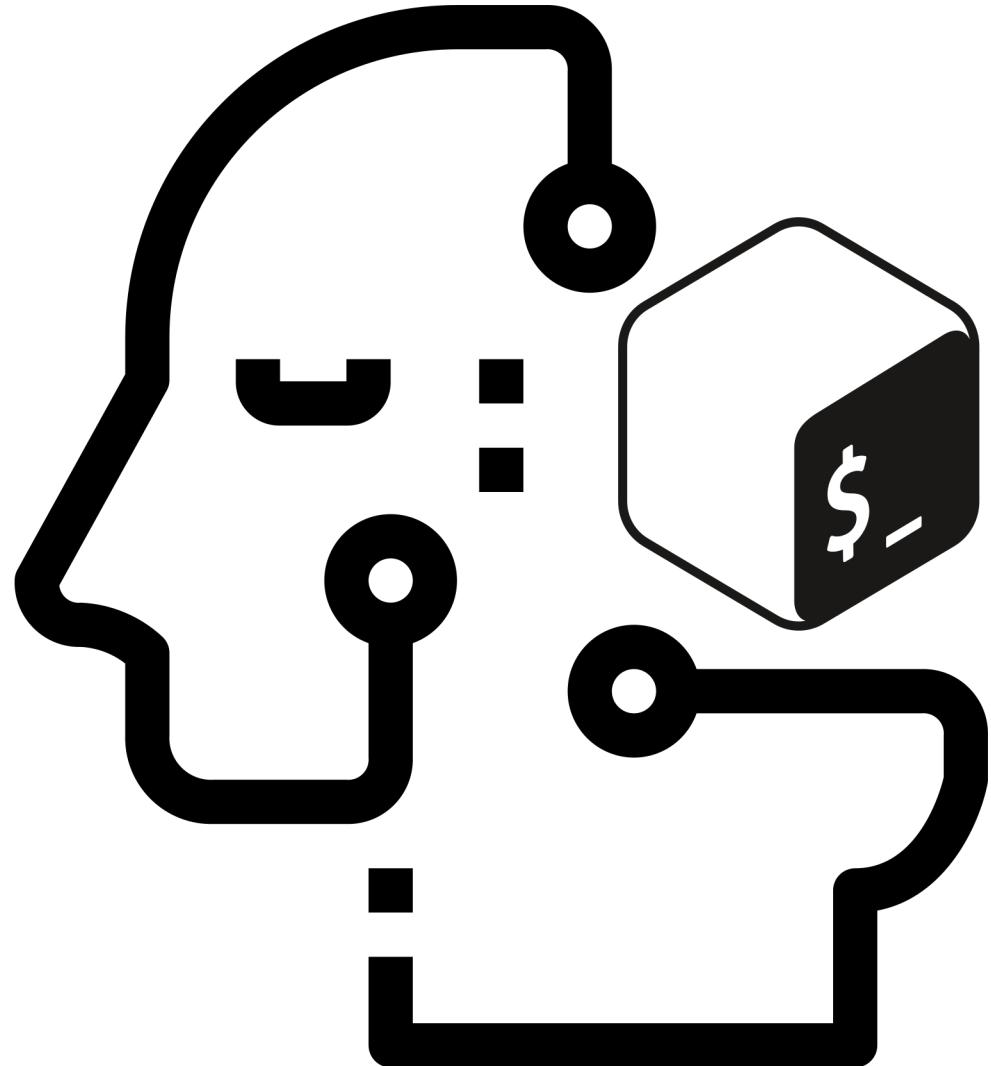
```
$ ls /dev > output  
$ wc -l < output  
210
```

Attention à ne pas faire des pipe pour rien

```
$ cat /tmp/myfile | wc -l
```



```
$ wc -l /tmp/myfile
```



Shell : des  
commandes  
en plus

---

# Shell : wc

---

- wc = word count : compte octet/mot/retour à la ligne
- Utilisable sur STDIN et sur un fichier

wc [options] [file]...

```
$ echo -e "salut\ncomment ca va ?" > msg
$ wc msg
    2 5 22 file
```

# Shell : wc

---

```
$ wc -l
```

Nombre de LineFeed

```
$ echo -n "salut" | wc -l  
0
```

Compte les sauts à la lignes  
et non les lignes



```
$ wc -c
```

Nombre de Bytes

```
$ echo -n "toto" | wc -c  
4
```

Problème en UTF8, les char peuvent être multibytes

```
$ echo -n "fée" | wc -c  
4
```

```
$ echo -n "fée" | wc -m  
3
```

```
$ wc -m
```

Nombre de caractères

# Shell : head et tail

---

- Affiche les premières/dernières lignes d'un fichier
- Utilisable sur STDIN

head [-n count] file

tail [-n count] file

```
$ ls /etc/ | head -n 1  
afpovertcp.cfg  
$ ls /etc/ | tail -n 2  
zshrc  
zshrc_Apple_Terminal
```

# Shell : cut

---

- cut: extrait des portions de chaque ligne d'un texte
- Texte passé en argument ou en STDIN

```
cut OPTION... [FILE]...
```

```
cut -d CHAR -f LIST
```

« cut » suivant un délimiteur  
Garde les « fields » sélectionnés

```
$ echo hello my little friend | cut -d ' ' -f 1  
hello  
$ echo hello my little friend | cut -d ' ' -f 1,3  
hello little
```

Garde des char à une pos donnée

```
cut -c LIST
```

```
$ ls /usr/ | cut -c 1  
b  
g  
i  
l  
l  
s  
s  
s
```

# Shell : cut

---

- Fonctionne aussi sur un fichier, traitement ligne par ligne

Fichier names.csv

```
John,Smith,34,London  
Arthur,Evans,21,Newport  
George,Jones,32,Truro
```

Extraction du prénom

```
$ cut -d ',' -f 1 names.csv  
John Arthur George
```

# Shell : tr

---

- Remplace une lettre par une autre (**translate**)
- Efface des lettres (**delete**)
- Elimine les répétitions de lettres (**squeeze**)

```
tr string1 string2  
tr -d string1  
tr -s string1
```

```
tr string1 string2
```

Translate

```
$ echo vive bash | tr aei 431  
v1v3 b4sh
```

Substitution lettre par lettre

```
$ echo vive bash | tr aei -  
v-v- b-sh
```

Substitution par 1 seule lettre

```
tr -d string1
```

Delete

```
$ echo file6543 | tr -d "123456"  
file
```

```
tr -s string1
```

Squeeze

```
$ echo "hello world" | tr -s " "  
hello world
```

# Shell : su

---

su = Substitute User : se faire passer pour un autre user

su [user]

- On passe par un nouveau shell
- Root est le « user » par défaut
- Demande le password de l'utilisateur ciblé

```
verdie_b@imac:~$ echo $USER  
verdie_b  
verdie_b@imac:~$ su  
password: ← Saisie du password root  
root@imac:~#  
root@imac:~# echo $USER  
root  
root@imac:~# exit  
verdie_b@imac:~$  
verdie_b@imac:~$ echo $USER  
verdie_b
```

# Shell : sudo

---

- sudo = **S**ubstitute **U**ser **D**O
- On exécute une commande en tant qu'un autre user
- Il faut être dans la liste de sudoers (/etc/sudoers)
- Root est le « user » par défaut
- Demande le password de l'utilisateur qui lance sudo

```
sudo [-u user] <command>
```

```
verdie_b@imac:~$ sudo cat /etc/passwd  
[sudo] password for verdie_b:
```

```
$ sudo su
```

?

Pour devenir root sans avoir besoin  
de connaître le password root

# Shell : changer son shell

---

**chsh : change shell**

```
$ chsh -s pathtoshell
```

Change the shell a new shell

Les choix sont stockés dans le fichier /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time
Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
kuro:x:1000:1000,,,,:/home/kuro:/bin/bash
messagebus:x:104:110::/nonexistent:/usr/sbin/nologin
Debian-exim:x:105:111::/var/spool/exim4:/usr/sbin/nologin
```

# Shell : ln

---

- **ln = LiNk** : créé des liens entre des fichiers/dossiers (Pour les curieux, inode conservé)

```
$ ln -s
```

Créé un lien symbolique (pour les curieux, nouvel inode)

```
/tmp/test$ touch f1
/tmp/test$ ln -s f1 f2
/tmp/test$ ls -al
drwxr-xr-x  4 verdie_b  wheel  128B Sep  3 13:43 .
drwxrwxrwt 26 root     wheel  832B Sep  3 13:43 ..
-rw-r--r--  1 verdie_b  wheel    6B Sep  3 13:43 f1
lrwxr-xr-x  1 verdie_b  wheel    2B Sep  3 13:43 f2@ -> f1
```

# Exo shell

---



Nous allons utiliser le fichier /etc/passwd

Faites un script avec les fonctionnalités suivantes

- Affichage du nombre de comptes présents sur la machine (wc)
- Affichage du nom de chaque compte ainsi que le shell qui lui est rattaché (cut)

# Shell : et quoi d'autre ?

---

Des commandes que vous connaissez et que  
vous voulez partager ?



# Grep

---

# Grep

---

```
grep [-abcdDEFGHhIiJllmnOopqRSsUVvwxZ] [-A num] [-B num] [-C[num]] [-e pattern] [-f file] [--binary-files=value]
[--color[=when]] [--colour[=when]] [--context[=num]] [--label] [--line-buffered] [--null] [pattern] [file ...]
```

- Permet d'afficher les lignes correspondantes à un pattern dans un flux
- Fonctionne depuis STDIN ou dans des fichier

Cas standard super simple ☺

Ex : n'afficher que les lignes contenant « bash »

```
$ grep bash /etc/passwd
_mbsetupuser:*:248:248:Setup User:/var/setup:/bin/bash
```

# Grep

---

- Fonctionne en multi fichier.
- Ici nombre d'occurrence de « bash » dans les fichier du dossier /etc

```
$ grep bash /etc/* 2> /dev/null | wc -l  
18
```

**grep -i** Ignore la casse

```
$ grep -i BaSh /etc/* 2> /dev/null | wc -l  
21
```

**grep -R** Mode récursif

```
$ grep conf /var/log/* --directories=skip | wc -l  
270  
$ grep -R conf /var/log | wc -l  
320
```

# Grep

---

`grep -c` Pour compter le nombre d'occurrences

```
$ grep -c bash /etc/passwd  
1
```

`grep -v` Pour inVerser le filtrage

```
$ wc -l /etc/passwd  
108 /etc/passwd  
  
$ grep -cv bash /etc/passwd  
107
```

# Grep

---

grep -e Pour utiliser les expressions régulières !



```
$ grep -e '/bin/[a-z]*sh' /etc/passwd
root:*:0:0:System Administrator:/var/root:/bin/sh
_mbsetupuser:*:248:248:Setup User:/var/setup:/bin/bash
```

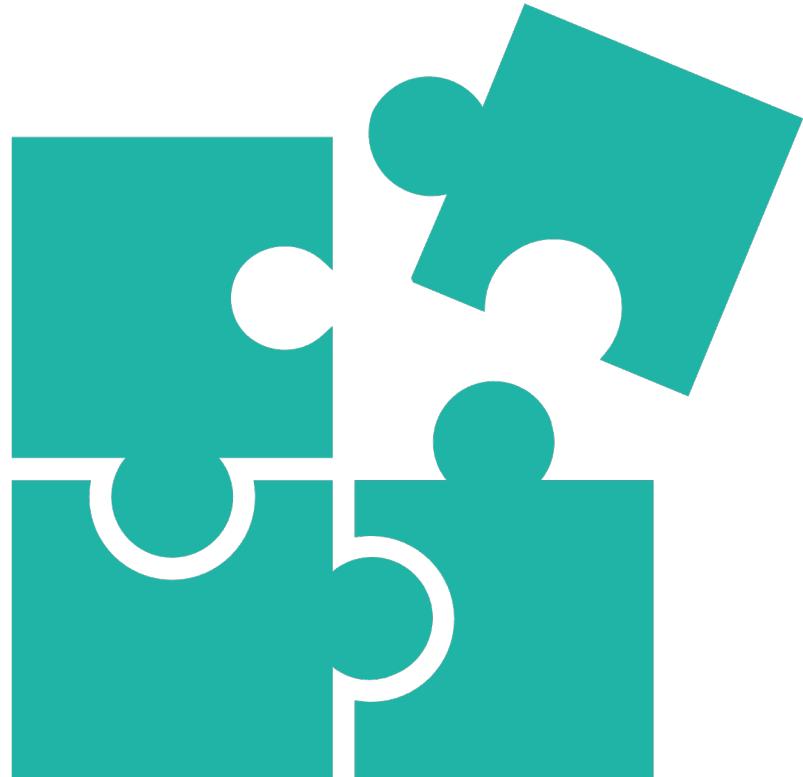
# Exo shell

---



Faites un script avec le comportement suivant :

- Affichage du nombre de fichiers présents dans « /usr/bin »
- Affichage du nombre de fichiers dans « /usr/bin » qui contiennent la string « mk »



# Bash expansion

---

# Bash expansion

---

- Cette étape se déroule lors du parsing de la commande
- Expansion par ordre de traitement
  - Brace Expansion
  - Tilde Expansion
  - Parameter Expansion
  - Command Substitution
  - Arithmetic Expansion
  - Word Splitting
  - Filename Expansion

# Brace expansion

---

- Permet de générer des chaînes de caractères
- Pas besoin que ce soit des dossiers ou fichiers qui existent

```
$ echo bon{jou,soi}r  
bonjour bonsoir  
$ mkdir v{1,2,3}
```

```
$ echo src/{projet-{fun,test},demo}/{src,build}  
src/projet-fun/src src/projet-fun/build src/projet-test/src src/projet-test/build src/demo/src src/demo/build
```

# Command substitution

---

- Remplace la commande par le résultat de son exécution

`$(command)`

``command``

```
$ pwd  
/Users/verdie_b  
$ ls -ld $(pwd)  
drwxr-xr-x+ 146 verdie_b  staff   4.6K Sep  2 13:45 /Users/verdie_b/
```

```
$ which echo  
/bin/echo  
$ ls -l $(which echo)  
-rwxr-xr-x  1 root  wheel    18K May  4  09:05 /bin/echo*
```

# Arithmetic Expansion

---

- Pour faire des calculs

`$(expression)`

```
$ echo $((1+5))  
6
```

```
$ V1=5; V2=6  
$ echo $((V1+V2))  
11
```

# Filename expansion

---

- Permet de « match » des noms de fichiers qui existent
- Différent des RegEx (que Bash ne supporte pas)

Question Mark (?)

Représente 1 caractère

```
$ ls /bin/??  
/bin/cp /bin/dd /bin/df /bin/ed /bin/ln /bin/ls /bin/mv /bin/ps /bin/rm /bin/sh
```

Asterisk (\*)

Représente entre 0 et  $\infty$  caractères

```
$ ls -d /*bin  
/bin/ /sbin/
```

```
$ ls /var/log/*.log | wc -l  
11
```



# Bash Conditional Expressions

---

# Bash : conditional expressions

---

`[[ expression ]]`

- Permet de tester des conditions
  - Existence ou propriétés d'un fichier
  - Existence d'une variable
  - Comparaison maths entre 2
  - Comparaison entre 2 strings
- Résultat : true ou false

`[[ 5 -eq 6 ]]`

# Bash : conditional expressions

---

```
[[ 5 -eq 6 ]]
```

Egalité mathématique ?

```
[[ "Bonjour" == "bonjour" ]]
```

String égales ?

```
[[ -d /tmp ]]
```

Dossier existe ?

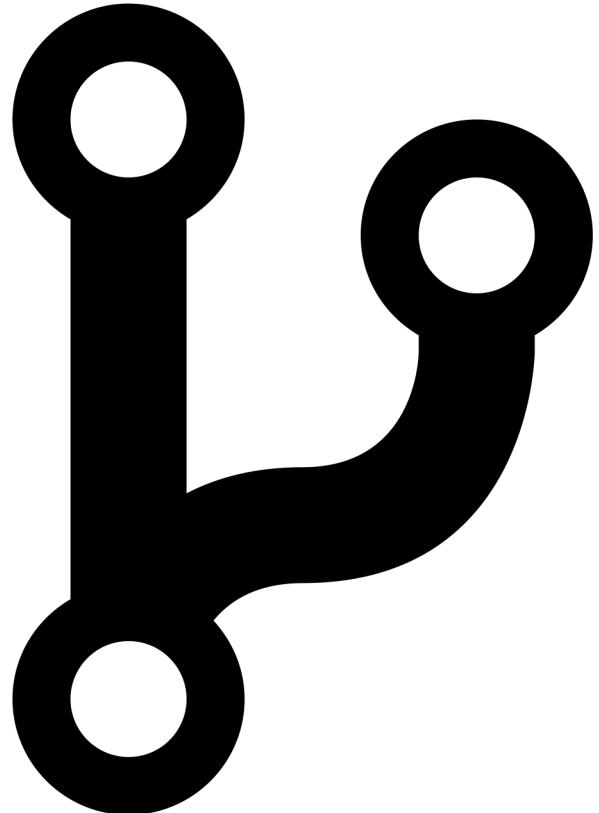
```
[[ -e /dev/random ]]
```

Fichier existe ?

```
$ [[ 5 -eq 6 ]]  
$ echo $?  
1
```

```
$ [[ 5 -eq 5 ]]  
$ echo $?  
0
```

← Ce n'est jamais utilisé comme ça 😊  
Avec « if then else » ca ira mieux !



# Bash : conditional constructs

---

# Conditional constructs : if then else

---

- Utilise le code de retour des binaires
- Code = 0 : OK      => true = 0
- Code ≠ 0 : NOK      => false = 1

```
if test-commands; then
    consequent-command
[else
    alternate-command]
fi
```

```
if [[ $USER == "verdie_b" ]]; then
    echo "benoit est connecté"
else
    echo "quelqu'un est connecté"
fi
```

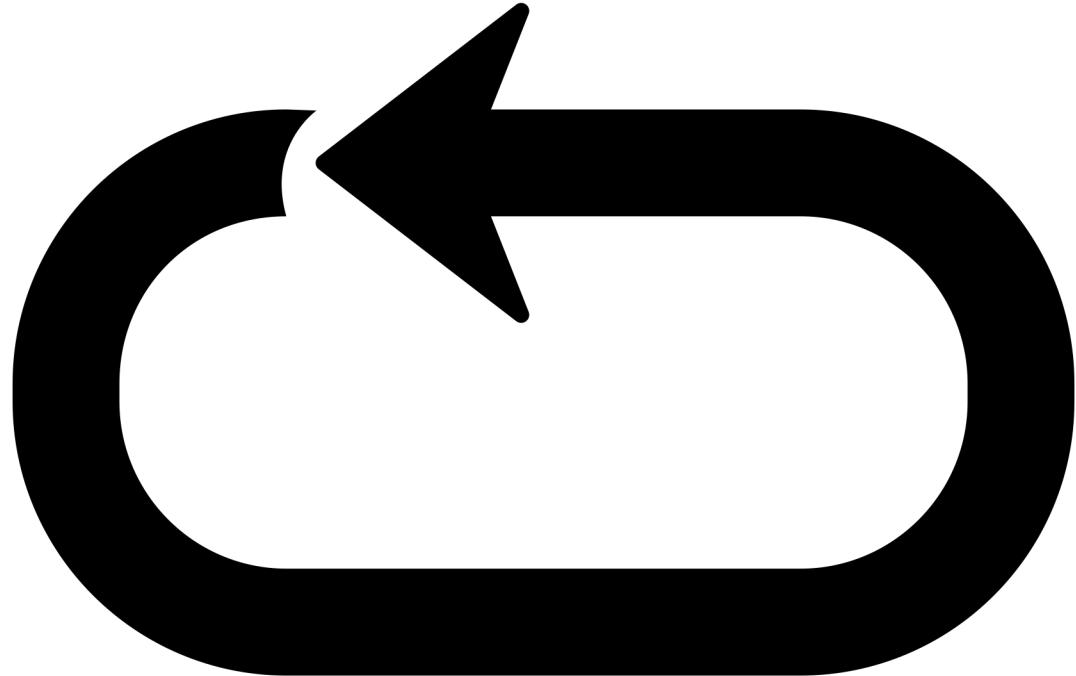
```
echo "hello"
if [[ $? -eq 0 ]]; then echo "cmd ok"; fi
```

# Conditional constructs : case

---

```
case word in
    [ [C] pattern [I pattern]...) command-list ;;
esac
```

```
echo -n "Enter the name of an animal: "
read ANIMAL
echo -n "The $ANIMAL has "
case $ANIMAL in
    horse | dog | cat) echo -n "four";;
    man | kangaroo ) echo -n "two";;
    *) echo -n "an unknown number of";;
esac
echo " legs."
```



# Bash : looping constructs

---

# Looping constructs : while

- Simple boucle

```
NB=0
while [[ $NB -lt 10 ]]; do
    echo $NB
    NB=$((NB + 1))
done
```

?

```
while test-commands; do
    consequent-commands;
done
```

## Boucle plus complexe avec read

Lecture de STDIN en continu

```
while read -r INPUT; do
    echo "read : $INPUT"
done
```

Variable créées à la volée

Traitement élément par élément

```
ls /bin/c* | while read -r FILE; do
    echo "$FILE"
done
```

# Looping constructs : while

---

Lecture d'un fichier ligne par ligne

```
INPUT="/path/to/txt/file"
while IFS= read -r LINE; do
    echo "$LINE"
done < "$INPUT"
```



# Looping constructs : for

On boucle parmi une liste

Code de retour de la dernière commande exécutée

```
for name in [words ...]; do  
    commands  
done
```

Un peu à la mano ☺

```
for NB in 0 1 2 3; do  
    echo $NB  
done
```



Binaire externe ☺

```
for NB in $(seq 0 9); do  
    echo $NB  
done
```



Shell expansion ☺

```
for NB in {0..9}; do  
    echo $NB  
done
```

Binaire externe ☺

```
for LOG in $(ls /var/log/*.log); do  
    echo "$LOG"  
done
```



```
for LOG in /var/log/*.log; do  
    echo "$LOG"  
done
```

# Exo shell

---



- Faites un script qui itère avec la boucle « for » sur la liste de Prénom « Fred Sophie Frank Céline » (il ne faut pas de noms avec espace sinon ca ne fonctionne pas)
- Afficher chaque prénom avec le nombre de caractères qui le compose.

# Exo shell

---



- Nous allons utiliser le fichier /etc/passwd.
- Faites un script qui affiche le nombre de comptes présents sur la machine.

=> commande « wc »

Ensuite affichez le nom de tous les comptes ainsi que leur shell.

=> lecture du fichier passwd ligne par ligne avec « while» et ensuite on découpe avec « cut »

Pour chaque utilisateur dont le shell est bash, affichez la mention « powered by bash ».

=> commande « if » pour vérifier la string

# RegEx

---

# Regex

---

- Regular Expressions décrivent des motifs textes
- Regex sont très utilisées dans Linux
  - grep
  - sed
  - awk
  - Perl, PHP, Python, dotnet, ...

On va utiliser « egrep -o » pour tester facilement

=> affiche le texte qui *match* les *patterns*

```
$ echo abc123 | egrep -o 'bc'  
bc
```

Metacaractère	Description
^	Position de début de la string.
.	n'importe quel caractère (sauf retour à la ligne)
[ ]	Bracket expression : Match un des caractères contenus entre les crochets. [abc] match "a", "b", or "c". [a-z] match entre a et z
[^ ]	Match ce qui n'est pas entre les crochets
\$	Position de la fin de la string
( )	Défini une sous expression réutilisable + tard On peut utiliser le pipe pour des alternatives : (chien chat)
\n	Référence la n-ème sous expression
*	Répétition de l'élément précédent 0 fois ou plus [0-9]* match n'importe quel nombre (mais aussi le vide) (ab)* match ab, abab, ababab (mais aussi le vide)
+	Répétition de l'élément précédent 1 fois ou plus
?	Répétition de l'élément précédent 0 ou 1 fois
{n}	Répétition de l'élément précédent n fois
{m,n}	Répétition de l'élément précédent entre m et n fois

# Regex Courantes

---

# Regex : classes de caractères

---

Classe	Signification
[:alpha:]	n'importe quelle lettre
[:digit:]	n'importe quel chiffre
[:xdigit:]	caractères hexadécimaux
[:alnum:]	n'importe quelle lettre ou chiffre
[:space:]	n'importe quel espace blanc
[:punct:]	n'importe quel signe de ponctuation
[:lower:]	n'importe quelle lettre en minuscule
[:upper:]	n'importe quelle lettre capitale
[:blank:]	espace ou tabulation
[:graph:]	caractères affichables et imprimables
[:cntrl:]	caractères d'échappement
[:print:]	caractères imprimables exceptés ceux de contrôle

# Regex : exemples

---

Chaque lettre match

```
$ echo abc123 | egrep -o '[a-z]'  
a  
b  
c
```

Match 1 lettre suivie d'un chiffre

```
$ echo abc123 | egrep -o '[:alpha:][:digit:]'  
c1
```

?

Match le premier mot

```
$ echo "Lorem ipsum dolor sit amet" | egrep -o '^[a-zA-Z]+'  
Lorem
```

?

Match un nombre de 5 chiffres

```
$ MSG="Envoi amour au 81212"  
$ echo "$MSG" | egrep -o '[0-9]{5}'  
$ if [[ $? -eq 0 ]]; then echo "SSPPAMMM"; fi  
SSPPAMMM
```

?

# Regex

---

- Attention, regex sont gloutonnes (« greedy ») => match le + long motif

```
$ echo "mais quelle belle maison, vous avez de la chance" | egrep -o 'mai.*on'  
mais quelle belle maison
```



```
<([a-z]+)>[[:alpha:]]* </\1>
```

?

```
$ echo "<from>Jani</from>" | egrep -o '<([a-z]+)>[[:alpha:]]* </\1>'  
<from>Jani</from>
```

# Regex

Super outils en ligne

<https://regex101.com/>

The screenshot shows the regex101.com interface. In the 'REGULAR EXPRESSION' section, the pattern is `/[0-9]+ (chat|chien){1}s? /gm`. The 'TEST STRING' is "j'ai toujours vécu avec 3 chats, 1 chien et 1 poisson". The 'EXPLANATION' panel details the regex components: `[0-9]+` matches one or more digits, `(chat|chien){1}` matches either 'chat' or 'chien' once, and `s?` is a greedy quantifier for zero or one space character. The 'MATCH INFORMATION' panel shows two matches: 'Match 1' with a full match from index 24 to 31 ('3 chats') and Group 1 from index 26 to 30 ('chat').

REGULAR EXPRESSION

2 matches, 50 steps (~1ms)

`/ [0-9]+ (chat|chien){1}s? /gm`

TEST STRING

SWITCH TO UNIT TESTS ▾

j'ai toujours vécu avec 3 chats, 1 chien et 1 poisson

EXPLANATION

▼ / [0-9]+ (chat|chien){1}s? / gm

▼ Match a single character present in the list below

[0-9]+

+ Quantifier — Matches between **one** and **unlimited** times, as many times as possible, giving back as needed (greedy)

0-9 a single character in the range between 0 (**index 48**) and 9 (**index 57**) (case sensitive)

matches the character literally (case sensitive)

▼ 1st Capturing Group (chat|chien){1} ▾

MATCH INFORMATION

Match 1

Full match 24-31 3 chats

Group 1. 26-30 chat

# Regex : Exo

---

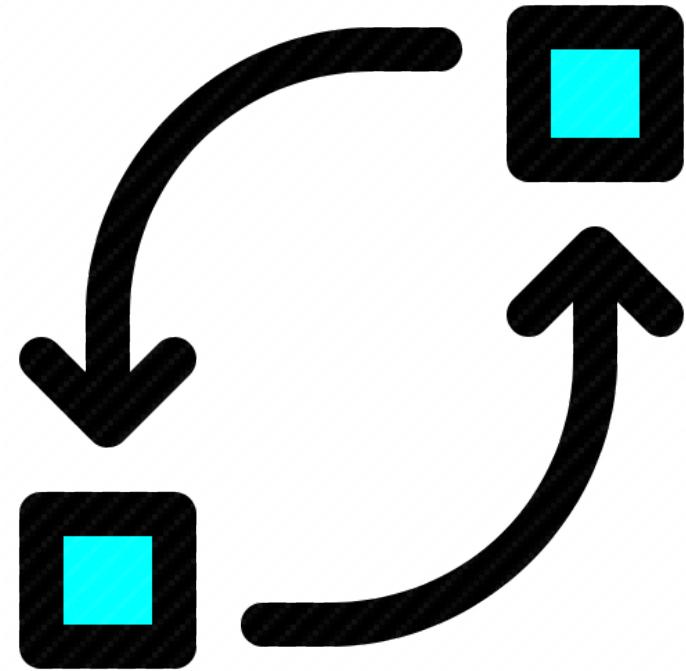
- Ecrire la regex qui match un numéro de téléphone
- Ecrire la regex qui match les noms de films d'action  
(Terminator, Predator, Cybernator, Desintegrator, ...)
- Ecrire une regex qui match qu'un password ne contient pas autre chose que des chiffres et des lettres, et fait entre 6 et 15 caractères de long.
- Ecrire une regex qui match une mac adress

```
[[:digit:]]{10}
```

```
^[[:alpha:]]+ator$
```

```
^[[:alnum:]]{6,15}$
```

```
([0-9a-f]{2}:){5}[0-9a-f]{2}
```



Sed

---

# Sed

---

```
sed [-Ealn] [-e command] [-f command_file] [-i extension] [file ...]
```

- Sed est éditeur de flux
- Il travail soit sur un fichier, soit sur STDIN
- On peut substituer, supprimer, insérer
- Programme très riche => man + tuto en ligne

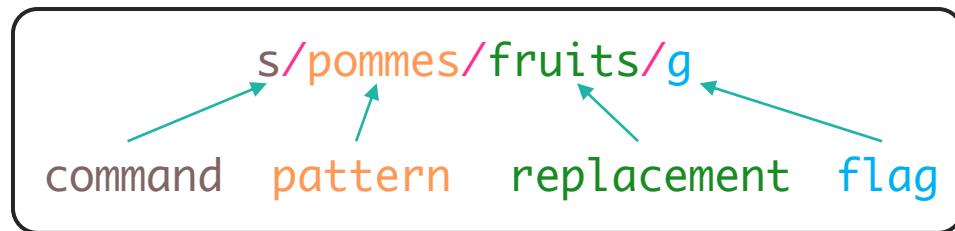


# Sed

---

## Cas standard

substitution d'un texte par un autre



On utilisera le texte suivant en exemple

```
$ echo -e "J'aime les pommes, mais surtout les pommes rouges. Et les poires bien fermes.\nEt les tomates, et pas que les rouges, contrairement aux pommes." > txt
```

```
$ sed 's/pommes/fruits/g' txt
```

```
J'aime les fruits, mais surtout les fruits rouges. Et les poires bien fermes.  
Et les tomates, et pas que les rouges, contrairement aux fruits.
```

# Sed : exemples

---

Fichier txt

```
J'aime les pommes, mais surtout les pommes rouges. Et les poires  
bien fermes.
```

```
Et les tomates, et pas que les rouges, contrairement aux pommes.
```

```
$ sed 's/po.*s/fruits/g' txt
```

```
J'aime les fruits.  
Et les tomates, et fruits.
```

```
$ sed 's/po[a-z]*s/fruits/g' txt
```

```
J'aime les fruits, mais surtout les fruits rouges. Et les fruits bien fermes.  
Et les tomates, et pas que les rouges, contrairement aux fruits.
```

# Sed : exemples

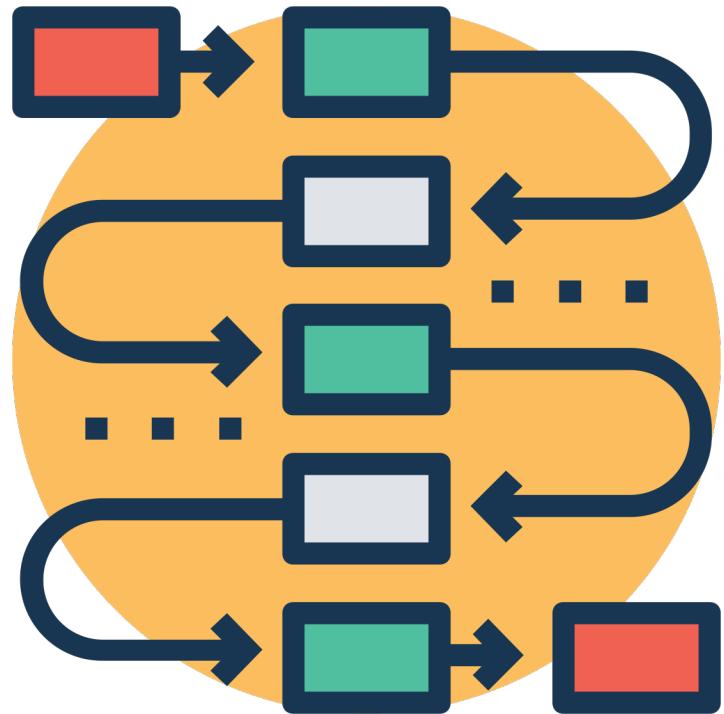
---

- Possibilité d'utiliser le mode « edit in place »
  - Modifie le fichier source
  - Crée automatiquement un backup avec extension fournie en param

```
$ sed -e 's/p[a-z]*s/fruits/g' -i .bak txt
$ ls txt*
txt      txt.bak

$ cat txt
J'aime les fruits, mais surtout les fruits rouges. Et les fruits bien fermes.
Et les tomates, et fruits que les rouges, contrairement aux fruits.
$ cat txt.bak

J'aime les pommes, mais surtout les pommes rouges. Et les poires bien fermes.
Et les tomates, et pas que les rouges, contrairement aux pommes.
```



# Awk

---

# Awk

---

```
awk [ -F fs ] [ -v var=value ] [ 'prog' | -f progfile ] [ file ... ]
```

- Awk est un langage de script pour traiter des données
- Il est très riche (on peut même coder dans un fichier séparé)
- Il lit tout seul le fichier et permet de faire du scripting



Example simpliste de découpage du fichier passwd avec le char : et affichage personnalisé

```
$ awk 'BEGIN{FS=":"} {print $1 " uses shell " $7}' /etc/passwd
root uses shell /bin/sh
daemon uses shell /usr/bin/false
_uucp uses shell /usr/sbin/uucico
_taskgated uses shell /usr/bin/false
[...]
```



*That's all Folks!*