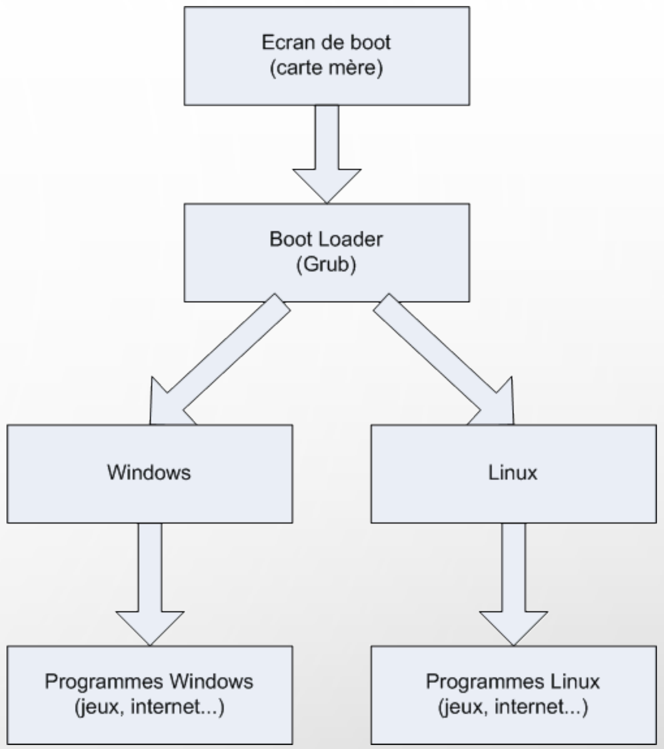
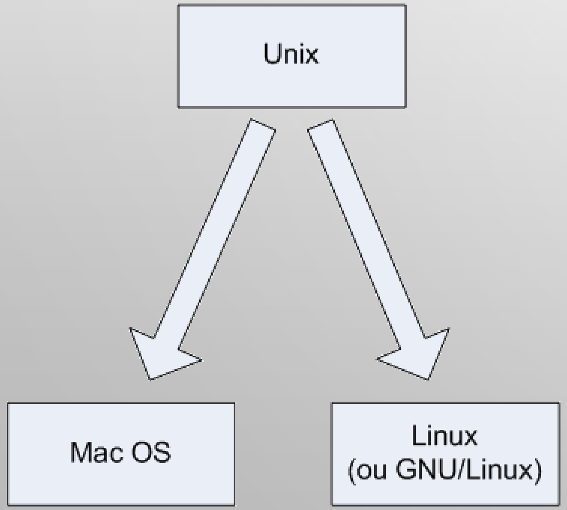
LINUX OPERATING SYSTEM

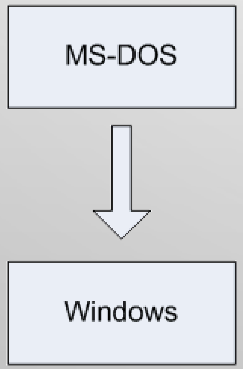
BY: MOHAMED AZIZ TOUSLI

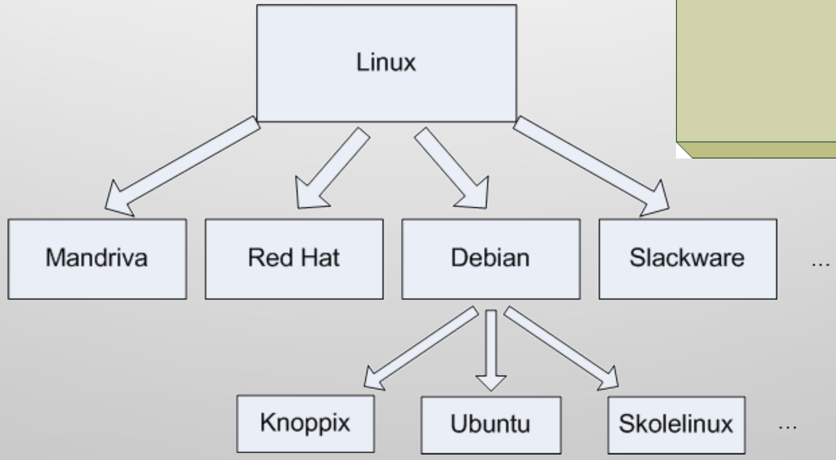
ABOUT

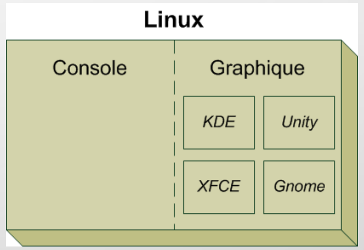
- Open source: We can have the source code, i.e. the "manufacturing recipe“
- Distribution: To simplify users' lives and allow them to make a choice (Packaging of Linux)
  - Console mode: ⇔ DOS
  - Graphical mode: ⇔ Windows
- X: Server program that manages graphical mode under Linux → Desktop management

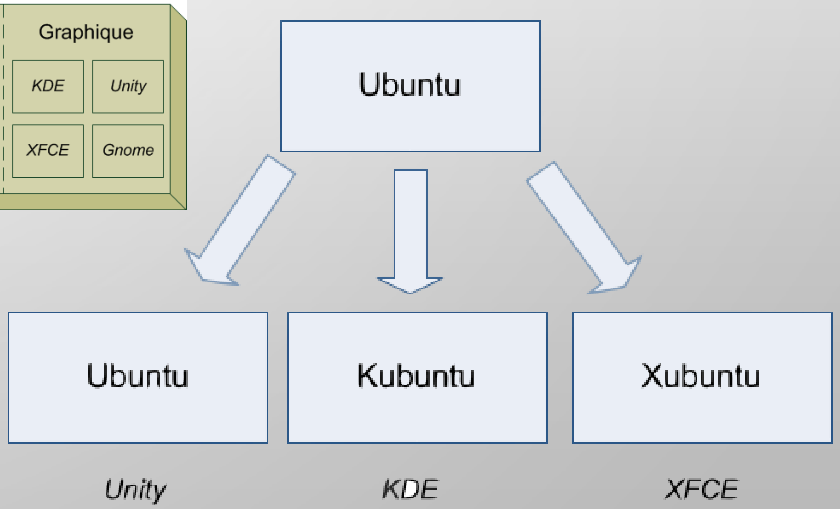




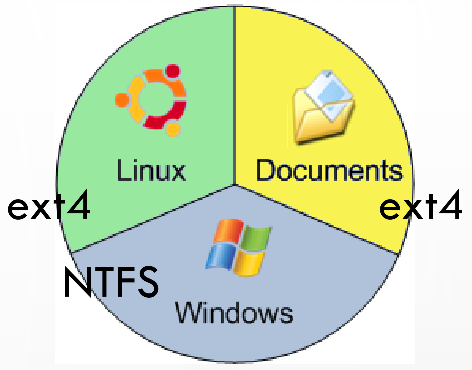








DISK PARTIONNING

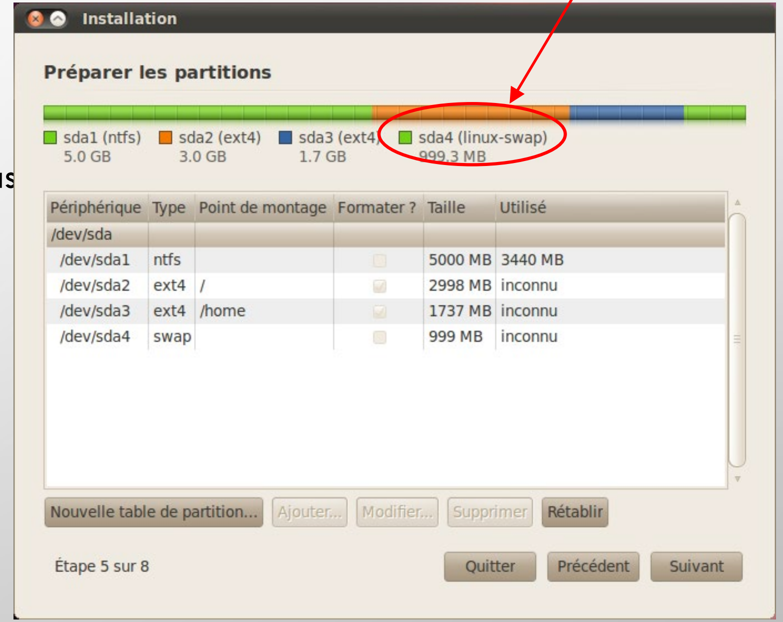


- Disk defragmentation: Collect fragmented files using defragmenter
  - Files are cut into several pieces and scattered on the disk → Files are fragmented
  - It is only for hard disks HDD (magnetic mass memory)and not SSD (flash memory)
- Disk portioning: Cut the disk into several parts
- File system: File organization on each partition
  - Window:
    - FAT 16: Capable of handling 4 GB of data
    - FAT 32: Capable of handling 2 TB of data → It fragments files much
    - NTFS: Capable of handling 16 Eo of data → It fragments files less and It recovers data when disk cras
  - Linux:
    - ext2: It fragments files less (Under Linux, there is no need to do defragmentation)
    - ext3: Journaling
    - ext4: Improves support for large hard drives and decreases file fragmentation issues
- Disk names under Linux: (*da1)
  - *: Disk is IDE ("h") or SCSI/S-ATA ("s")
  - d: This letter does not change
  - a: First disk; "b" for second disk…
  - 1: First partition in disk, "2" for second partition in disk…

→ GRUB: The program to choose the OS to start at startup

Nautilus: File explorer of Ubuntu
Dolphin: File explorer of Kubuntu

It is an extension of the RAM on your hard drive. When your RAM is full, Linux continues to work but goes through the hard drive, thanks to the partition "swap".


Installation

Préparer les partitions

🟩 sda1 (ntfs) 5.0 GB 🟧 sda2 (ext4) 3.0 GB 🟦 sda3 (ext4) 1.7 GB 🟩 sda4 (linux-swap) 999.3 MB

Périphérique	Type	Point de montage	Formater ?	Taille	Utilisé
/dev/sda					
/dev/sda1	ntfs		☐	5000 MB	3440 MB
/dev/sda2	ext4	/	✓	2998 MB	inconnu
/dev/sda3	ext4	/home	✓	1737 MB	inconnu
/dev/sda4	swap		☐	999 MB	inconnu

Nouvelle table de partition… Ajouter… Modifier… Supprimer Rétablir

Étape 5 sur 8 Quitter Précédent Suivant

- Partition types under Ubuntu:
  - Primary: Basic and classical partition (four per disc)
  - Logical: Partition that can contain many sub-partitions

LINUX SHORTCUTS

- Ctrl + Alt + Fi → Open terminal i in full screen (tty_i) ; $1 \leq i \leq 6$
  - We can open 6 terminals at one time
- Ctrl + Alt + F7 → Return to desktop mode
- Alt + ImpEc + K / Ctrl + Alt + Backspace → Restart X server
- Telnet: Protocol to communicate with a server (non secured)
- SSH: Protocol to communicate with a server (secured) → PuTTY software

TERMINAL

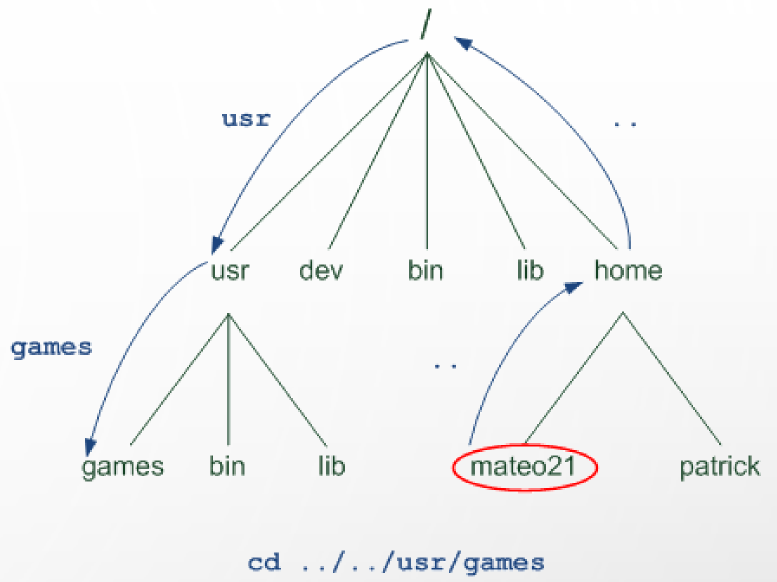
- pseudoName@desktopName: currentFolder S
  - ~ → home
  - S=$ → Normal user mode (limited rights)
  - S=# → Super user mode = root (non limited rights)
- command -p (value) --parameter(=value) //Command with parameters
- Find a command:
  - First letters of command + 2 x Tab //Autocompletion
  - history //Command to show history of used commands
  - Ctrl + R + some letters //Find used commands way before

TERMINAL SHORTCUTS

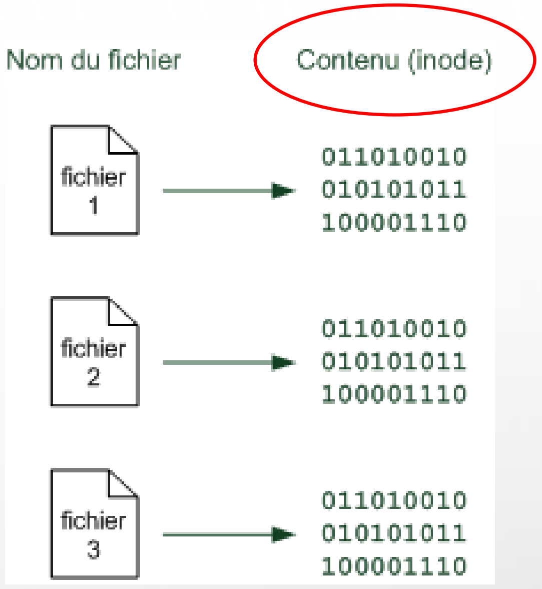
- Ctrl + L → Delete content of console (clear command)
- Ctrl + D → Close console (exit command)
- Shift + PgUp / PgDown → Show messages sent by console
- Ctrl + A / Origin → Send cursor to beginning
- Ctrl + E / End → Send cursor to end
- Ctrl + U → Delete on the left of the cursor
- Ctrl + K → Delete on the right of the cursor
- Ctrl + Y → Paste what was deleted

FILES & FOLDERS

- File types:
  - Classic files: .txt, .doc, .png, …
  - Special files: CD player…
  - → Under Linux, everything is considered as a file
- "/" → Root folder
  - On Windows: "C:\...\..."
  - On Linux: "/…/…"
- pwd //Print current directory
- which command //Give localization of command (i.e. program)
- ls //Show content of current directory (ls fileName to show fileName)
  - -a (hidden files), -F (type of element), -l (detailed list), -h (size in bytes), -t (sort by date of modification)
- cd X//Change directory
  - Relative path: X → folderName, .. (parent folder), ~ (home folder), no X (home folder)
  - Absolute path: X → /home/username
  - First letters of X + Tab //Autocompletion
- du //Disk usage → Show size of folder and subfolders
  - -h (size in bytes), -a (folders and files), -s (total size)



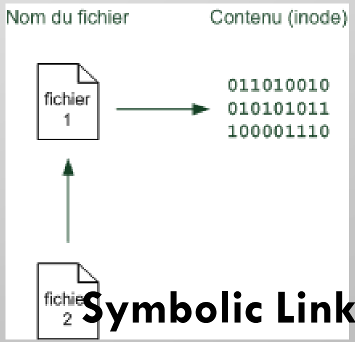
bin: Programs (executable)
dev: Files containing the devices
home: Personal files
lib: Shared libraries (.so files ⇔ .dll files)
var: "Variable" data



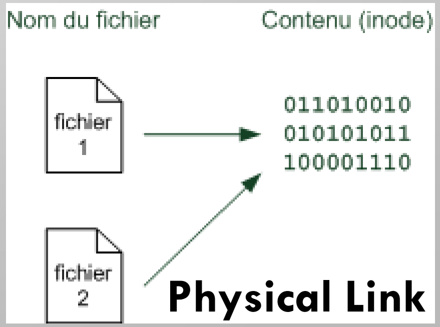
FILE MANIPULATION

- cat file //View full file content (-n to show number of lines)
- less file //View file content page per page {**Space** → Next page; **Entry** → Next line}
- head file //View the beginning of the file (-n for the first n lines)
- tail file //View the ending of the file (-n for the last n lines) (-f -s x to get the end refreshed, every x seconds)
- mkdir folder1 folder2 //Create folder (-p to create folders and sub folders)
- cp fileToCopy directory/fileCopied //Copy file (-R to copy folders)
- rm file1 file2 //Remove file (-r to remove folders) (-i to demand confirmation) (-v to tell what is happening)
- rmdir directory //Remove directory only if it is **empty**
- mv fileToMove directory //Move file to directory
- mv fileToRename fileRenamed //Rename file
- ln file1 file2 //Create **physical link** between files (ls -i to verify this)
- ln -s file1 file //Create **symbolic link** between files (Can be done on folders)

  PS: If file 1 is deleted → Link is dead (file 2 won't be deleted)

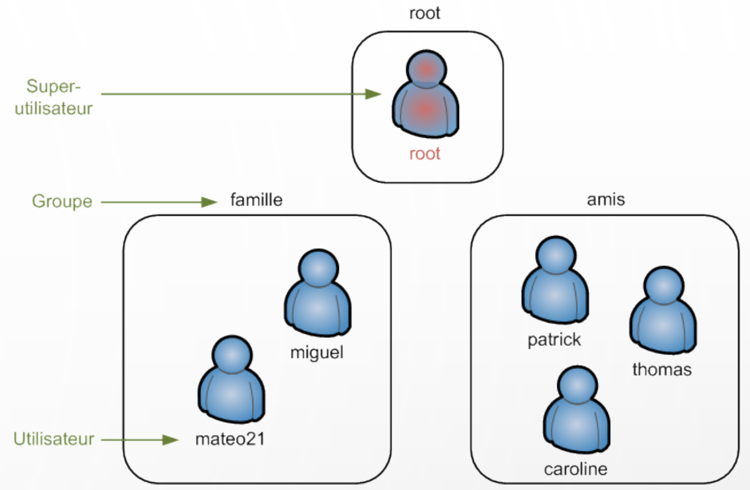


Symbolic Link



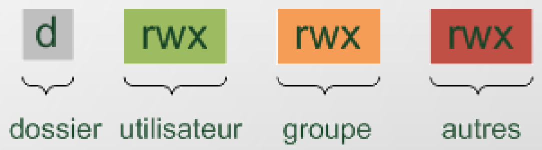
Physical Link

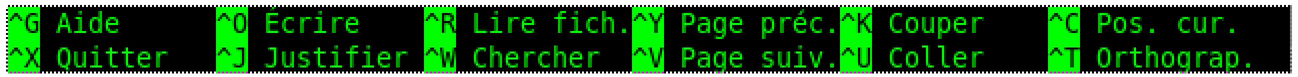
USERS & RIGHTS



- sudo command //Become a root for an instant (**Substitute User DO**)
- sudo su //Become root indefinitely (exit to exit root mode)
- adduser userName //Add new user (By default new user = new group)
- passwd userName //Set password to user
- deluser userName //Delete user (--remove-home to remove home folder)
- addgroup groupName //Add new group
- usermod userName //Change user (-g{aG} groupName{setOfGroups} to change group) (-l newName to change name)
- delgroup groupName //Delete group
- chown userName:groupName fileName //Change user and group ownership of fileName (-R for directories)
- chmod u=rwx, g+r , o=- fileName //Change access rights to fileName (user **rwx**, group **add r**, other **nothing**)
- chmod number fileName //Change access rights of fileName (number=*** in binary{for u,g,o}, eg. *777* for everything)
  - You just need to own the file in order to change the access rights (-R for directories)

- d: folder
- l: link
- r: read
- w: write
- x: execute





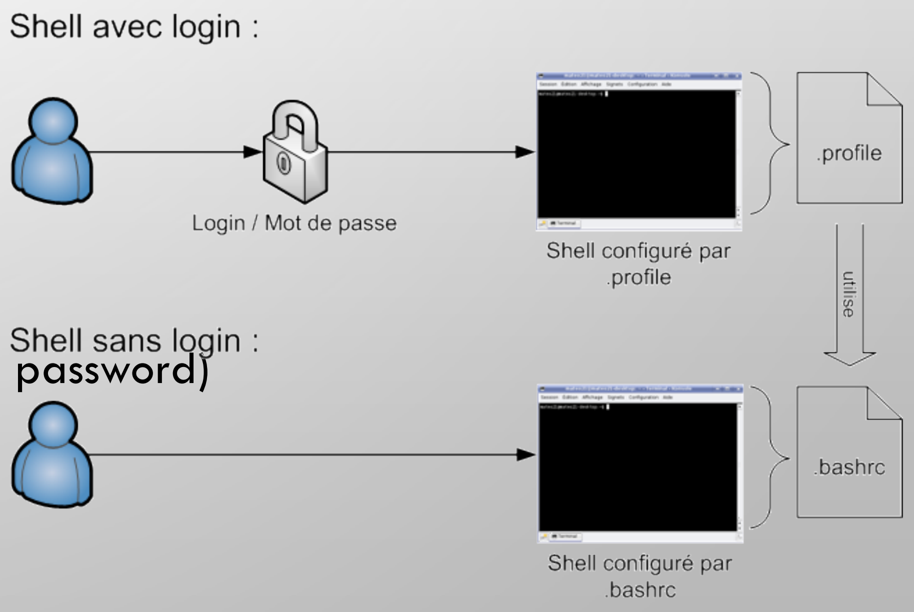
```
^G Aide       ^O Ecrire     ^R Lire fich.^Y Page préc.^K Couper     ^C Pos. cur.
^X Quitter    ^J Justifier  ^W Chercher  ^V Page suiv.^U Coller     ^T Orthograp.
```

TEXT EDITING WITH NANO

- **A text editor** allows you to write plain text files, without formatting
- **A word processor** allows you to write plain text files, with formatting
- nano fileName //Open text file [-m to use mouse] [-i to indendate automatically] [-A to return automatically]
  - If the file does not exist, it will be automatically created
  - .nanorc //File that configurate nano (global nano file → /etc/nanorc)
    - set mouse, set autoindent, set smarthome
  - .bashrc //File that set aliases (global bash file → /etc/bash.bashrc)
    - alias shortcutName='commandName'

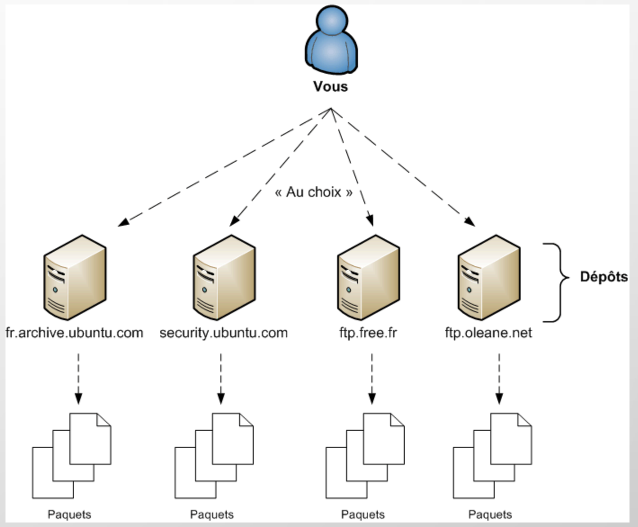
.profile → When a new console in which you log in (requires password)

.bashrc → When you open a console in which you do not log in (doesn't require password)



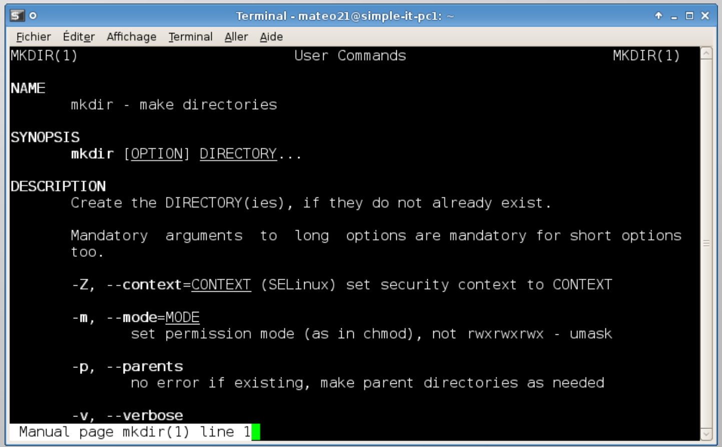
INSTALLING PROGRAMS

- Under Ubuntu, we do not have installation programs; we have what are called **packages** (.deb ⇔ .exe)
  - There is **dependency** management of the program
    - Programs/Packets are **dependent** on "bits of programs" called **libraries**/**other packets** to work
  - All .debs are gathered in the same place on the same server called **repository**
    - **Repository**: This is a server that offers all packages in one place
    - sudo nano /etc/apt/sources.list //List of repistories (repository + version of distrubtion + section)
      - **deb** → To download the compiled (binary) version of programs
      - **deb-src** → To recover the source code of the program
  - Graphical tool: System → Administration → Software Sources
- apt-get update //Update our cache if it is not already done
- apt-cache search packName //Search for the package
- apt-get install packName … //Install the package
- apt-get remove packName //Uninstall the package
- apt-get autoremove packName //Uninstall the package with its dependencies
- apt-get upgrade //Update all installed packages on your system at once (Make update before)



READ THE F****** MANUAL (RTFM)

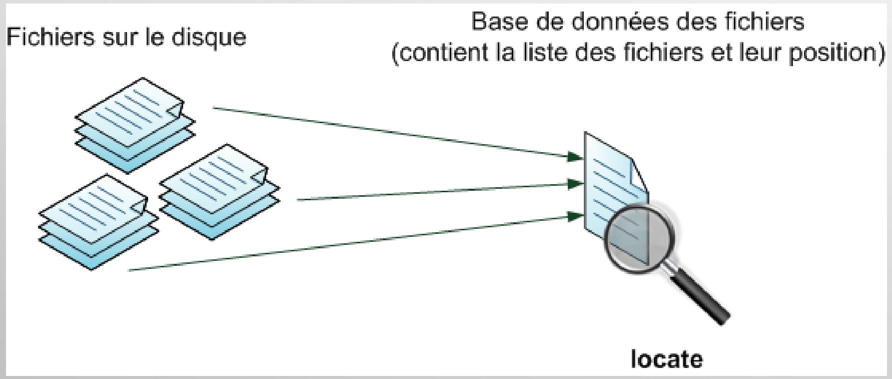
- man commandName //Check the manual of command (Click on '/' to search for a word)
- NAME:
  - **Bold** → type the word exactly as shown
  - Underline → Replace the underlined word with the appropriate value in your case
  - [-hvc] → All -h, -v and -c options are optional
  - a | b → you can write the option "a" **XOR** "b", but not both at the same time
  - Option… → The ellipses indicate that the option can be repeated as many times as you want
- apropos commandName //Search for a command
- commandName --help -h //Open help of a command
- whatis commandName //Small definition of a command

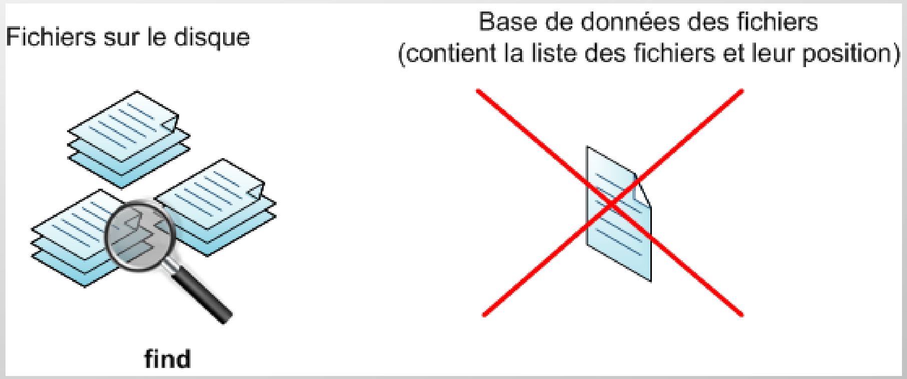

Terminal - mateo21@simple-it-pc1: ~
Fichier Éditer Affichage Terminal Aller Aide

```
MKDIR(1)                         User Commands                        MKDIR(1)

NAME
       mkdir - make directories

SYNOPSIS
       mkdir [OPTION] DIRECTORY...

DESCRIPTION
       Create the DIRECTORY(ies), if they do not already exist.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -Z, --context=CONTEXT (SELinux) set security context to CONTEXT

       -m, --mode=MODE
              set permission mode (as in chmod), not rwxrwxrwx - umask

       -p, --parents
              no error if existing, make parent directories as needed

       -v, --verbose
 Manual page mkdir(1) line 1
```

SEARCH FILES

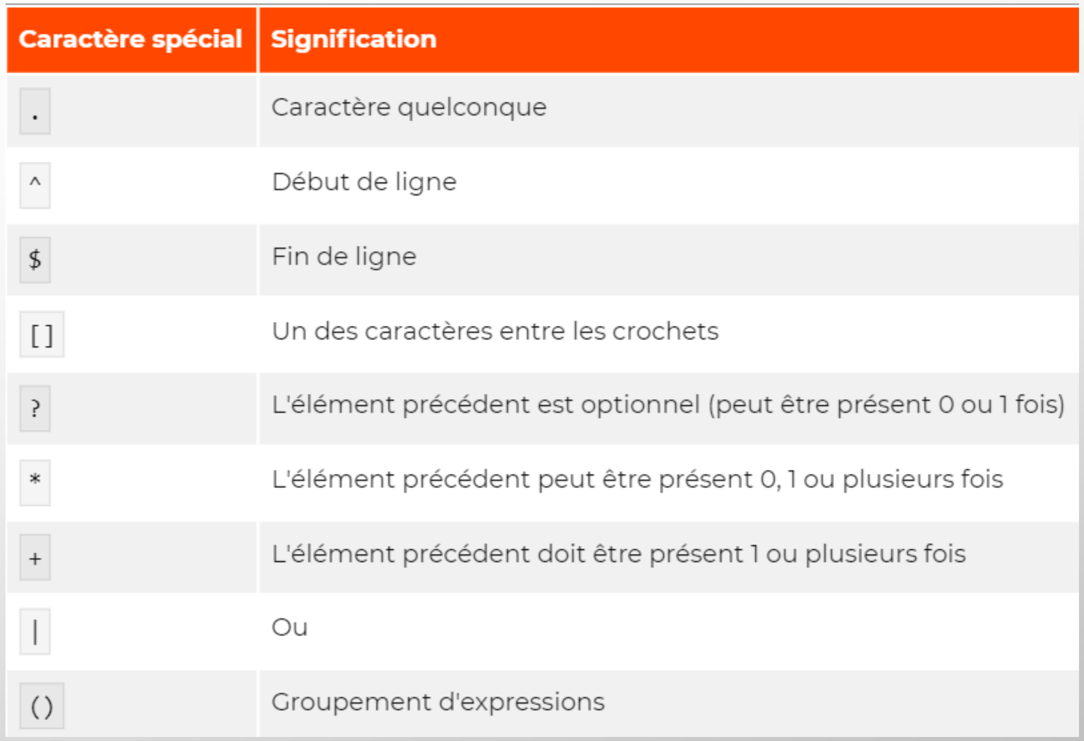
- locate fileName //Look for file in ALL file data base (slocate fileName)
  - File data base get updated every 24 hours (updatedb to update manually)
- find {directory} -name "*fileName*" {action} //Look for *fileName*
  - (-size {+/-}10KMG) (-atime {+}nDays) (-type d/f) (action=-printf " ", -delete, -exec/-ok{yes or no) command { } \;)


Fichiers sur le disque
Base de données des fichiers
(contient la liste des fichiers et leur position)
locate

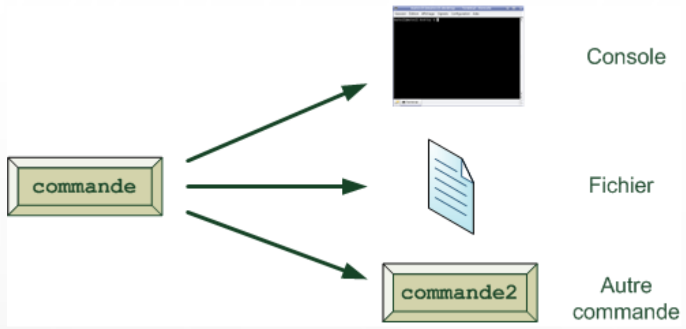

Fichiers sur le disque
Base de données des fichiers
(contient la liste des fichiers et leur position)
find

EXTRACT, SORT AND FILTER DATA

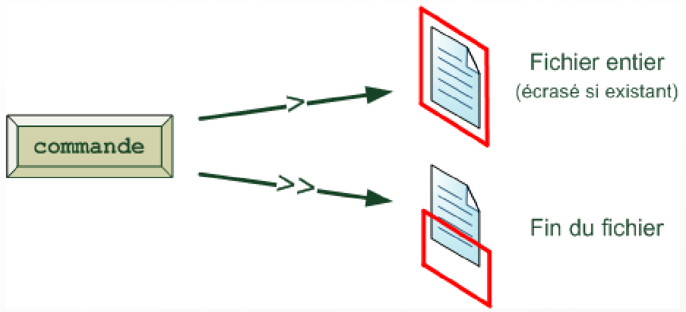
- grep “text text” fileName //search for a word in a file and display the lines in which that word was found
    - -i → Ignore (upper/lower cases)
    - -n → Get number of lines
    - -v → Ignore lines that contain ‘text’ (Reversed grep)
    - -r directoryName → Search in all files and subfolders
    - -E → Use regular expressions (Check table) (-E is optional)
- sort fileName //Sort elements alphabetically
    - -o anotherFile → Place sorted elements in another file
    - -r → Reversed sort; -R → Random sort; -n → Sort numbers
- wc fileName //Count → Return number of lines/words/bytes
    - -l → Number of lines; -w → Number of words; -c → Number of bytes;
    - -m → Number of characters
- uniq fileName anotherFile//Return unique elements (must be used after sort)
    - -c → Number of occurrences; -d → Lines with doubles
- cut -c {x}-{y} fileName //Cut characters from x to y
- cut [-d w -f x-y,z] fileName //Cut field x to y & z after delimiter w



Caractère spécial	Signification
.	Caractère quelconque
^	Début de ligne
$	Fin de ligne
[ ]	Un des caractères entre les crochets
?	L'élément précédent est optionnel (peut être présent 0 ou 1 fois)
*	L'élément précédent peut être présent 0, 1 ou plusieurs fois
+	L'élément précédent doit être présent 1 ou plusieurs fois
|	Ou
( )	Groupement d'expressions

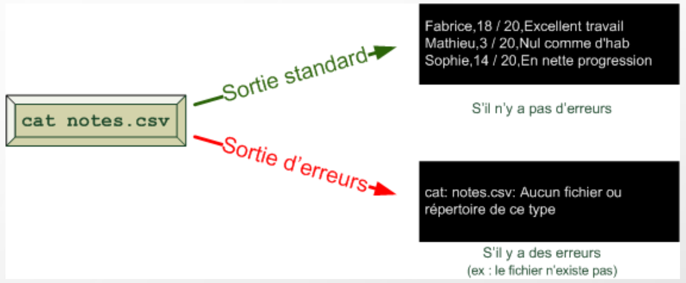


REDIRECTION FLOWS



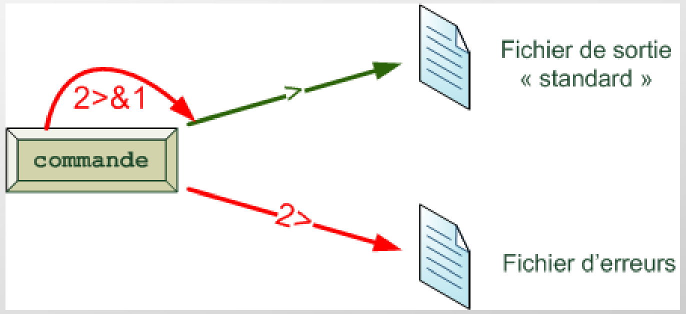
- command > anotherFile //Redirect result to another file
  - /dev/null → Good place to redirect results that we don't want to see in console
- command >> anotherFile //Redirect result at the end of another file
- command 2> errors.log //Redirect errors to errors.log
- command 2>> errors.log //Redirect errors at the end of errors.log
- command > anotherFile 2>&1 //Redirect errors to another file
- command >> anotherFile 2>&1 //Redirect errors to end of another file
- command < fileName //Read from a file
- command << stopKeyWord //Read from keyboard until stopKeyWord
- command1 | command2 //Use output of command1 with command2

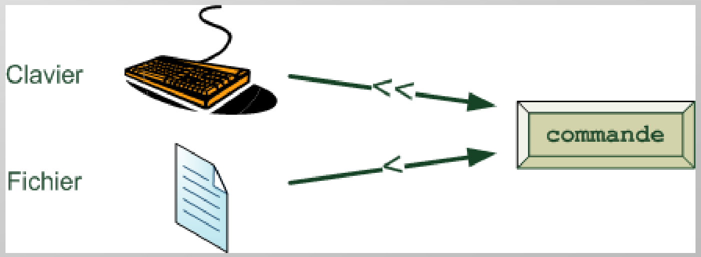
  → Chain of commands

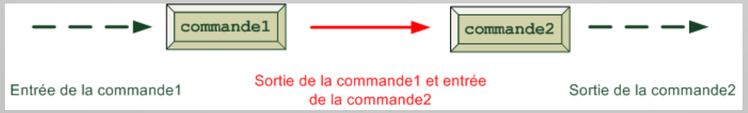


```
Fabrice,18 / 20,Excellent travail
Mathieu,3 / 20,Nul comme d'hab
Sophie,14 / 20,En nette progression
```

```
cat: notes.csv: Aucun fichier ou
répertoire de ce type
```





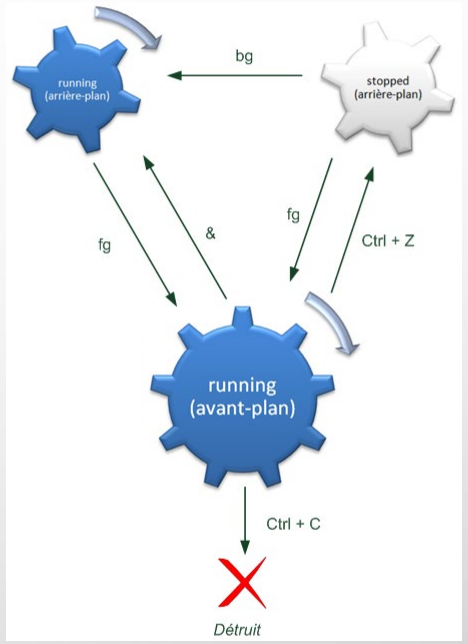


SYSTEM ACTIVITY

- Linux is **multi-task** and **multi-user** (via Internet)
- w //Command to tell who does what
  - date //Return date
  - uptime //Return operating time of computer and load of CUP usage
  - tload //Return CPU usage graphically
  - who //Return who does what
    - TTY=Name of console; FROM=IP@; LOGIN@=Login hour; IDLE=Inactivity duration; WHAT=Command executed now
- **Process** is a program that runs in memory
- ps //Command to return **static** list of processors (by same user (≠ root) and same console)
  - PID=Process ID; TIME=Execution duration; CMD=Program that generated this process; UID=User ID
  - -ef=All users & all consoles; -ejH=Tree model for processes; -u userName=Processes by userName
- top //Command to return **dynamic** list of processors (%CPU)
- Ctrl+C //Kill process in console
- kill processID… //Kill process (-9 to kill it aggressively)
- killall processName //Kill process
- halt //Shutdown computer; reboot //Reboot computer; shutdown //Related to halt and reboot

PROCESSES ON BACKGROUND

- command & //Start a process on background (exit console → exit process)
  - Return number of process in background in this console & PID of process
- nohup command //Start a process on background (exit console ↛ exit process)
- Ctrl + Z //Pause process execution
- bg //Pass the process in the background
- jobs //Return the processes that run in the background in a console
- fg %processNumber //Resume a process in the foreground
- screen //Open a program that is a terminal multiplier {–r numberOfScreen to open an already open screen}
  - Ctrl + a + (Release) + ? //Get help of screen
  - Ctrl + a + c //Open a new window
  - Ctrl + a + s //Split screen into several parts
  - Ctrl + a + Tab //Pass from a window to another
  - Ctrl + a + {0…9} //Open window with specific number
  - Ctrl + a + X/k //Close a window
  - Ctrl + a + d //Detach screen from console

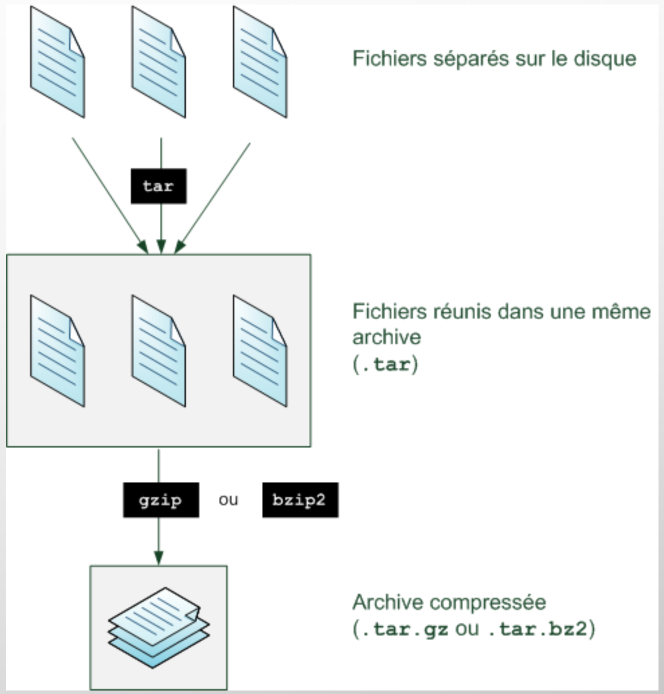


DATE

- date //Show current date {"+%Hh%Mm%Ss"} (Only the letter following the % is interpreted)
- date MMDDhhmm{YYYY} //Change date
- at hh:mm MM/DD/YY //Execute 1 command at a specific time → Write command → Ctrl + D
- at now +x minutes/hours/days/weeks/months/years //Execute 1 command at a specific time
- atq; atm jobNumber //List and delete pending jobs
- command1; command2 ⇔ command1 && command2 //Execute command1 then command 2
- sleep x //Pause system for x seconds (xm, xh, xd for minutes, hours, days)
- echo "export EDITOR=nano" >> ~/.bashrc //Change editor by default to nano
- Crontab → Modify the list of the programs to execute; Cron → Execute them
- crontab //{-l to show crontab, -e to modify crontab, -r to delete crontab}
- minute hour dayOfMonth month dayOfWeek command //Program a command in crontab
    - * → all values | x-y → x to y | x,y → x and y | */x → multiples of x

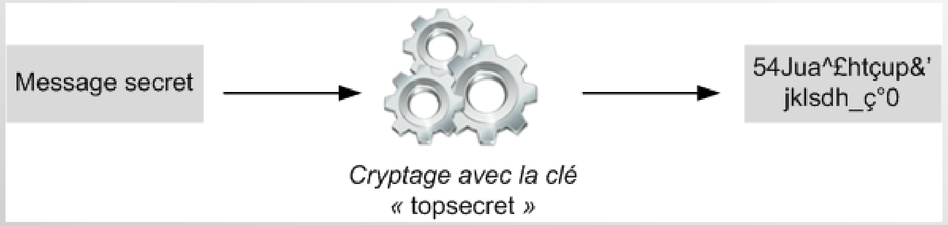
ARCHIVE & COMPRESS

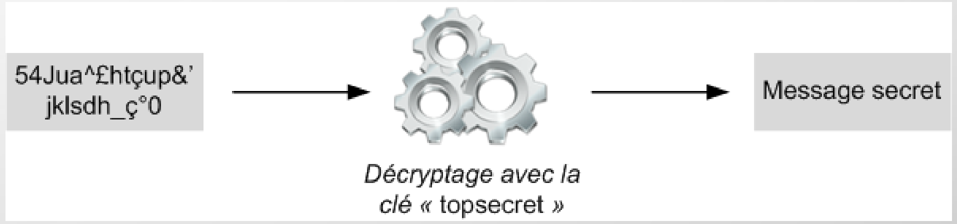
- Archive:
  - tar -cvf archiveName.tar folderName //Assemble folder in an archive to compress it
  - tar -cvf archiveName.tar fileName1 … //Assemble files in an archive (Not recommended)
  - tar -tf archiveName.tar //Show content of archive without extraction
  - tar -rvf archiveName.tar newFile //Add file to archive
  - tar -xvf archiveName.tar //Extract files from archive
- Compress:
  - gzip archiveName.tar (or fileName directly) //Compress archive into .gz format (common one)
  - gunzip arciveName.tar.gz //Uncompress .gz into archive format
  - bzip2 archiveName.tar //Compress archive into .bz2 format (slower + less size)
  - bunzip2 archiveName.tar.bz2 //Uncompress .bz2 into archive format
- tar -zcvf archiveName.tar.gz folderName //Archive and compress a folder into .gz format (Replace –z with –j to use .bz2 format)
- tar -zxvf archiveName.tar.gz //Uncompress .gz into normal folder
- tar -ztf archiveName.tar.gz //Show content of archive without extraction
  - zip and rar archive and compress at the same time (Install **unzip** and use zip –r and unzip commands; Install **unrar** and use unrar e command)


Fichiers séparés sur le disque
tar
Fichiers réunis dans une même archive (.tar)
gzip
ou
bzip2
Archive compressée (.tar.gz ou .tar.bz2)

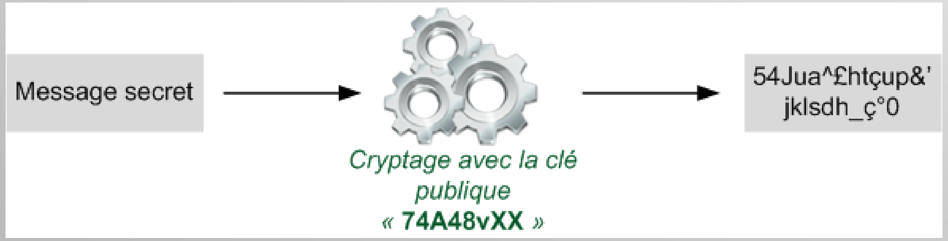
ENCRYPTION AND DECRYPTION

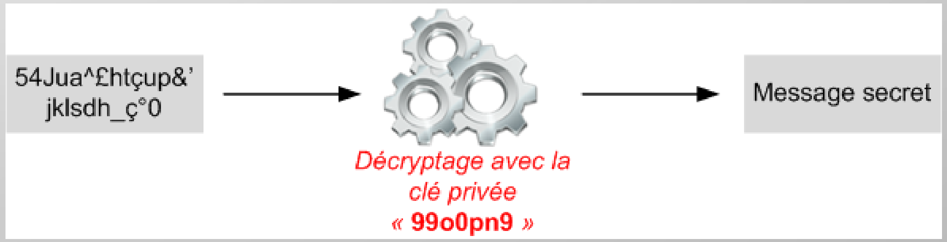
- A **server** is a computer that stays on 24/7
- A **protocol** is a language for computers to communicate with each other in a network
  - **Telnet**: Simple but dangerous vs **SSH**: Secured (Uses symmetric and asymmetric encryptions)
- **Symmetric encryption**: The person who encrypts and who decrypts must have the same key (Dangerous)

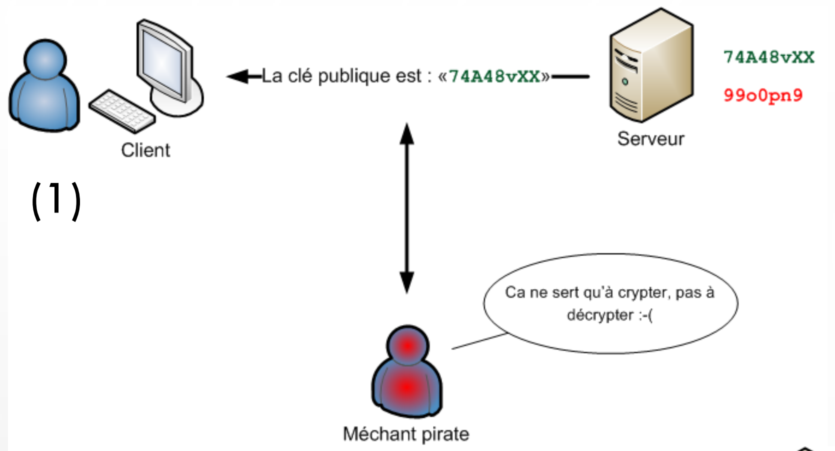


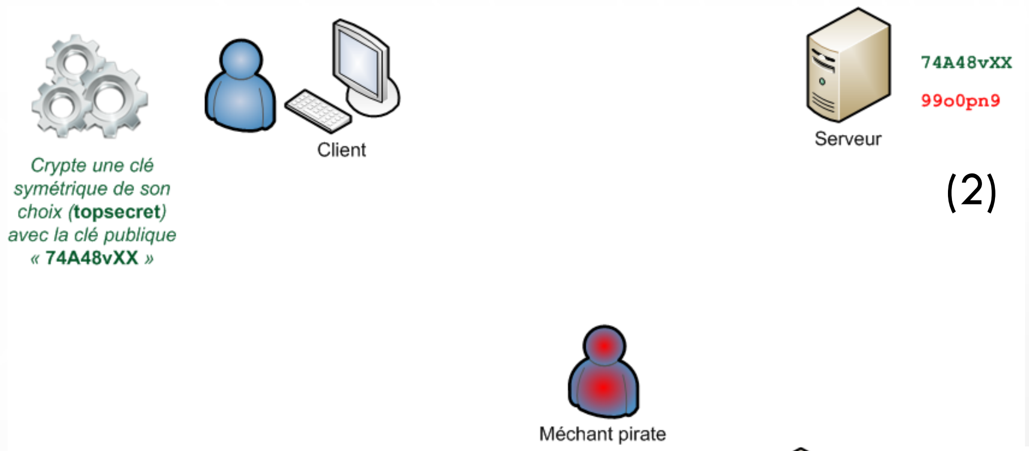

54Jua^£htçup&' jklsdh_ç°0
Message secret
Décryptage avec la clé « topsecret »

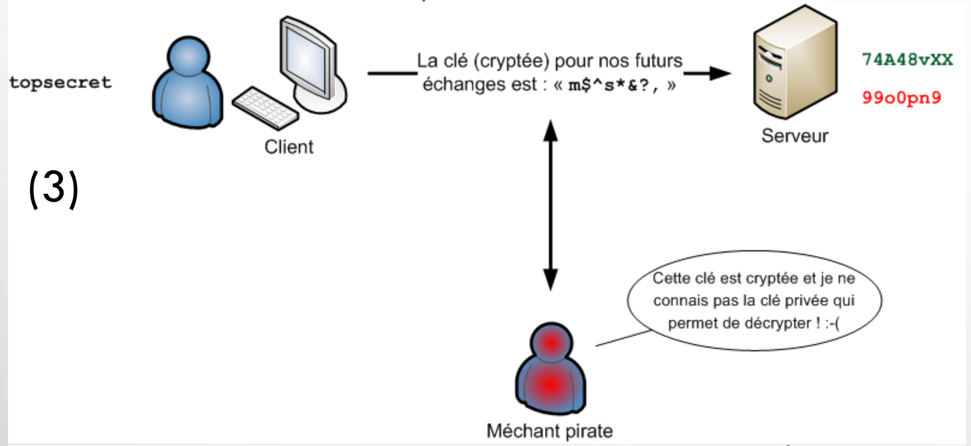
- **Asymmetric decryption**: A public key for encryption and a private key for decryption (Slower than symmetric)

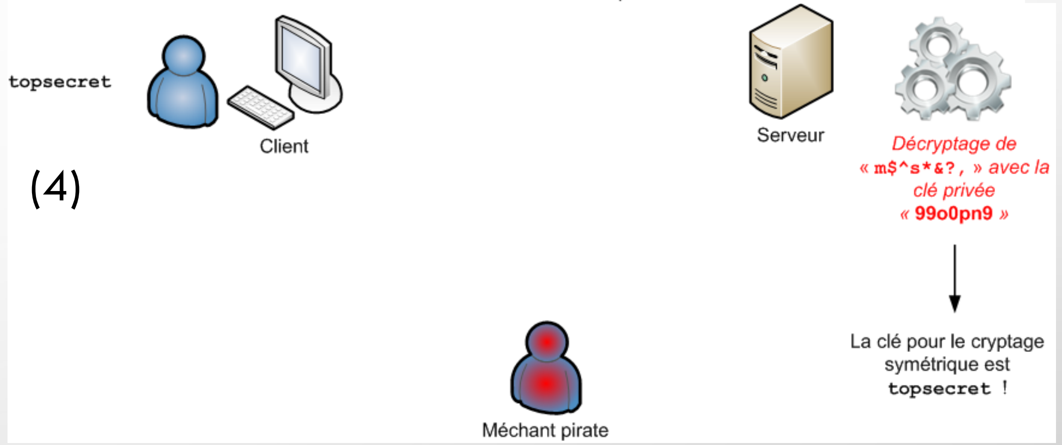

Message secret
54Jua^£htçup&' jklsdh_ç°0
Cryptage avec la clé publique « 74A48vXX »

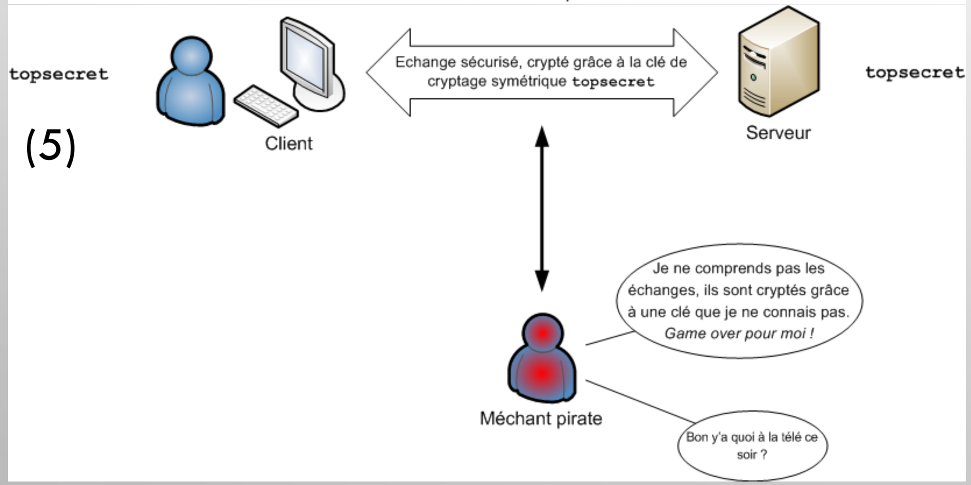

54Jua^£htçup&' jklsdh_ç°0
Message secret
Décryptage avec la clé privée « 99o0pn9 »

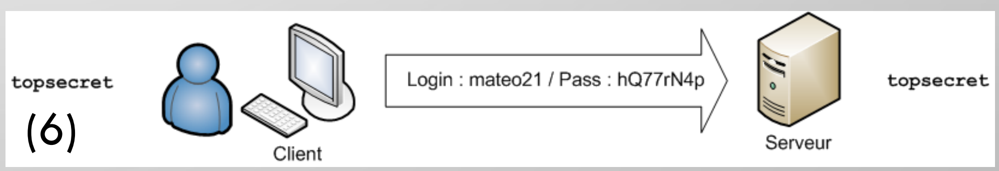

La clé publique est : «74A48vXX»
74A48vXX
99o0pn9
Client
Serveur
(1)
Ca ne sert qu'à crypter, pas à décrypter :-(
Méchant pirate


Crypte une clé symétrique de son choix (topsecret) avec la clé publique « 74A48vXX »
Client
74A48vXX
99o0pn9
Serveur
(2)
Méchant pirate


topsecret
Client
La clé (cryptée) pour nos futurs échanges est : « m$^s*&?, »
74A48vXX
99o0pn9
Serveur
(3)
Cette clé est cryptée et je ne connais pas la clé privée qui permet de décrypter ! :-(
Méchant pirate


topsecret
Client
(4)
Serveur
Décryptage de « m$^s*&?, » avec la clé privée « 99o0pn9 »
La clé pour le cryptage symétrique est topsecret !
Méchant pirate


topsecret
Client
Echange sécurisé, crypté grâce à la clé de cryptage symétrique topsecret
Serveur
topsecret
(5)
Je ne comprends pas les échanges, ils sont cryptés grâce à une clé que je ne connais pas. Game over pour moi !
Méchant pirate
Bon y'a quoi à la télé ce soir ?

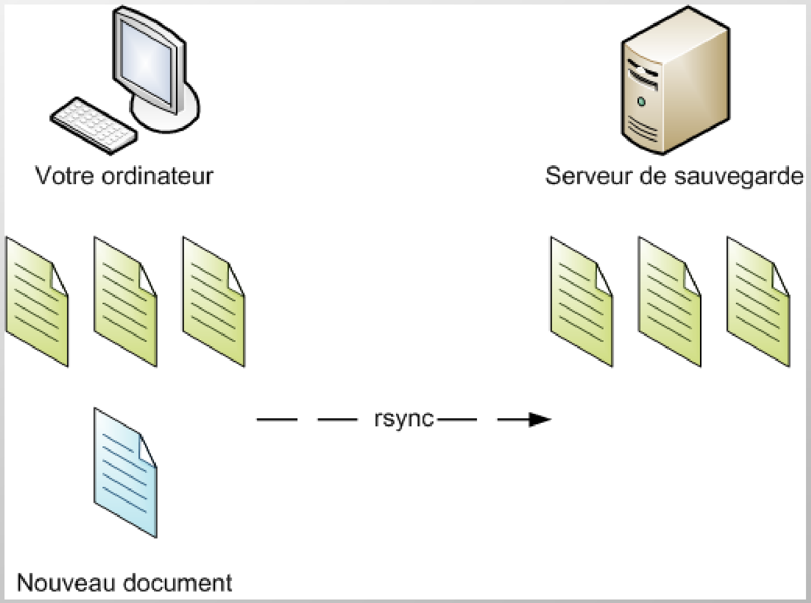

topsecret
(6)
Client
Login : mateo21 / Pass : hQ77rN4p
Serveur
topsecret

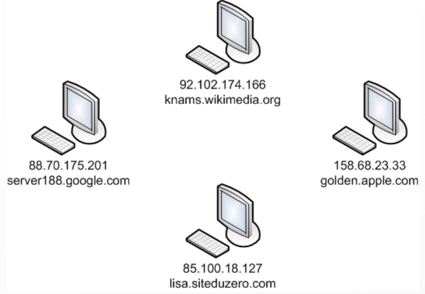
SSH

- Server:
  - sudo apt-get install openssh-server //Turn machine into server
  - sudo /etc/init.d/ssh start //Start server
  - sudo /etc/init.d/ssh stop //Stop server
- Client:
  - ssh login@ip //Connect on server via Linux {-p portNumber (if server is not on port 22)}
  - Download PuTTY //Connect on server via Windows
    - IP=Internet IP if global (whatismyip), =Local IP if local (ifconfig), =127.0.0.1=localhost if on same computer
- Types of authentification:
  - **Password** authentication
  - **Public** and **private** key authentication

FILE TRANSFER

- wget downloadLink //Download file {-c to resume a stopped download} {--background to download in background}
- scp fileToCopy login@ip/domainName:downloadLocation //Copy file from my PC to another PC {-P portNumber}
- scp login@ip:fileToCopy downloadLocation //Copy file from another PC to my PC
- rsync folderName login@ip:backupName //Synchronize files for backup = Smart scp {--delete to guarantee deletion}
- File transfer:
  - **Anonymous mode:** Download a file from a public FTP server
  - **Authenticated mode:** Download/Send a file from/to a private FTP server
- ftp ftpWebSite //Login = anonymous
- Use typical commands to move within an FTP server
- put fileName //Send a file to the server (Not possible with public servers)
- get fileName //Download a file from the server
- !commandName //Execute command within computer and not FTP server
- sftp -oPort=portNumber login@ip //Secured FTP

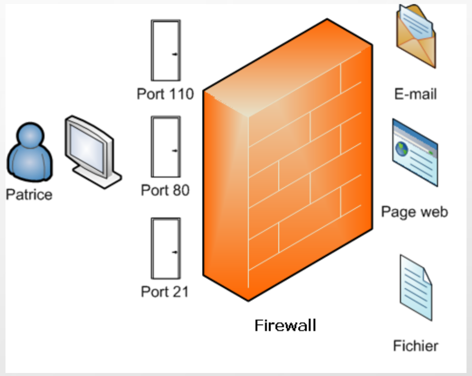




FIREWALL

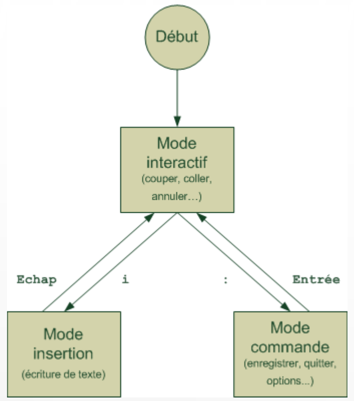
- host hostName/IPaddress //Return correspendant IPaddress/hostName (DNS servers do this conversion)
  - We can do the DNS work by editing the file /etc/hosts
- whois domainName //Return information about domainName (=hostName)
- ifconfig //Return list of networking interfaces
  - eth0 → Ethernet connection | lo → Local connection | wlan0 → Wi-Fi connection
- ifconfig interface{eth0, wlan0} state{up, down} //Activate/desactivate interfaces
- netstat //Show networking interfaces {-i for statistics, -ta for open connections, -n for portNumber instead of portName}
- iptables //Open firewall {-L for rules, -F for reinitialization, -n like netstat}
- iptables -A chain{INPUT/OUTPUT} -p protocole{tcp/udp} --dport port{ssh/22…} -j {ACCEPT/REJECT/DROP}

  → //Add/Delete rules
- iptables -A INPUT -p icmp -j ACCEPT //Authorize pings
- iptables -A INPUT -i lo -j ACCEPT //Authorize local connections
- iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT //Authorize already established connections
- iptables -P INPUT DROP //Refuse all other connections by default



COMPILE A PROGRAM FROM SOURCES

- To add a new program:
    1. Use apt-get install
    2. Download .deb package from internet
    3. Compile program from sources
- **Compilation** is a process that transforms the source code of a program into an executable that can be used
- sudo apt-get install build-essential //Download compilation tool
- ./configure //Check if all the necessary tools to compile the software you want to install are present
    - Results of errors → Install necessary packages
- make //Compile the program
- make install //Install the program after compilation
- make uninstall //Uninstall the program after installation


Début
Mode interactif (couper, coller, annuler…)
Echap
i
:
Entrée
Mode insertion (écriture de texte)
Mode commande (enregistrer, quitter, options…)

VIM TEXT EDITOR

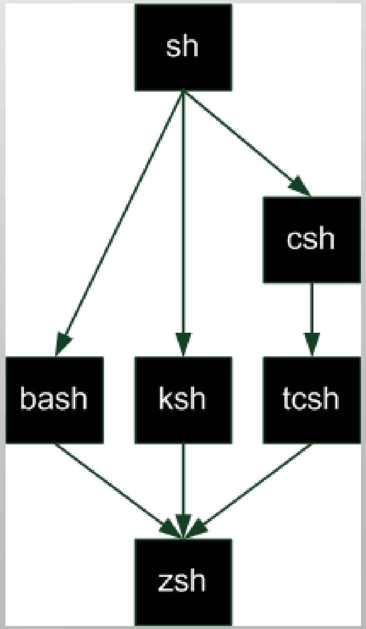
- **Shell:** Programming minilanguage built into all Linux systems that allows you to automate repetitive tasks
- vim [fileName] //Open file using vim
  - **Interactive mode:** No to writing | Yes to moving around, copying and pasting, deleting…
  - **Insertion mode:** Normal text editor mode (i and Esc buttons)
  - **Command mode:** Launch commands such as save and quit (: and Return buttons)
- **Activate options in a configuration file:** → /etc/vim
  - syntax on //Activate syntaxic coloration
  - set background=dark //Change background color
  - set number //Show line numbers
  - set showcmd //Show current command (**Useful!**)
  - set ignorecase //Ignore case when searching
  - set mouse=a //Enable mouse support → **Visual mode**

Abbr.	Action
:w	Save file
:q	Quit
:q!	Force quit
:wq	Save and quit
:s/old/new	Replace first occurence
:%s/old/new/g	Replace all occurences
:r otherFile	Merge files
:sp	Split screen H
:vsp	Split screen V
:!command	Start external command
:set option	Act. option
:set nooption	Desact. option
:set option=value	Change option
:set option?	Know state

Abbr.	Action
0/Beg	Go to beginning
$/Fin	Go to ending
W	Move word to word
5x	Cut next 5 letters
5dd	Cut next 5 lines
5dw	Cut next 5 words
d0/y0	Cut/Copy to beg.
d$/y$	Cut/Copy to end.
5yy	Copy next 5 lines
5yw	Copy next 5 words
5p	Paste 5 times
r	Replace a letter
R	Replace letters
5u	Undo 5 modifications
Ctrl+R	Undo undo
5G	Go to 5th line
/	Search forward
?	Search backward
n	Next search
N	Last search

SHELL/BASH

- Shells = Different console environments (sh, bash, ksh,…)
- **Shell** is the program that manages the command prompt → The program that waits for you to enter orders
- vim shellFile.sh; chmod +x shellFile.sh; ./shellFile.sh //Create shell file, give rights to execute it and execute it
    - PS: Shell file needs to be copied in PATH repository (Eg. /bin, /usr/bin, /usr/local/bin…) to execute it from any place
- bash –x shellFile.sh //Execute shell file in debugging mode
- #!/bin/bash //Initialize shell file to open with bash
- #This is a comment (→ Write commands in shell script now)


$$\begin{tikzcd}
 & sh \arrow[dl] \arrow[d] \arrow[r] & csh \arrow[d] \\
bash \arrow[dr] & ksh \arrow[d] & tcsh \arrow[dl] \\
 & zsh &
\end{tikzcd}$$

VARIABLES

- **Variables:** They allow us to temporarily store information in memory
- str='message' //Define a string (We must do str='message' and not str = 'message'
- echo parameter1 … "parameter 3" //Print parameters
- $str //Get value of variable
- echo '&str' → &str //**Simple quotes** → Print message at it is
- echo "&str" → message //**Double quotes** → Print variable content
- echo \`pwd\` → /home/… //**Back quotes** → Execute and print return of command
- read variableName1 … //Read content {-p 'message'} {-n to limit number of characters} {-t to limit time} {-s for password}
  - (The last variable gets the rest of the words)
- let "a=b + - * / % ** c" //Define an integer {bc command for floats}
- env //Show environment variables = global variables {export command to define them}
- ./fileName.sh param1… //$# → Number of parameters | $0 → scriptName | $1 → First parameter
- shift //Shift the parameters in the variables ($1:param1 → $1:param2)
- array=('value0' 'value1' 'value2'); ${array[i]}; array[i]=newValue //Work with arrays (* Indexing does not have to be continuous)

CONDITIONING

```
if [ ! test1 ] //We must do [ test ], and not [test]
then
        /* commands here */
elif [ test2 ] && | | [ test3]
then
      /* commands here */
else
      /* commands here */
fi
```

Strings	Integers	Files
$str1 = (==) $str2	$num1 -eq $num2	-e $f ⇔ f exist
$str1 != $str2	$num1 -ne $num2	-d $f ⇔ f directory
-z $str ⇔ str is zero	$num1 -gt $num2	-f $f ⇔ f file
-n $str ⇔ str is not zero	$num1 -ge $num2	-L $f ⇔ f shortcut
	$num1 -lt $num2	-rwx $f ⇔ access rights
	$num1 -le $num2	$f1 -nt $f2 ⇔ newer than (-ot)

```
case variable in
      "value1")
            /* commands here */
            ;;
esac
```

LOOPS

```
while [ test ]

do

        /* commands here */

done
```

```
until [ test ]

do

        /* commands here */

done
```

```
for variable in ‘value1' ‘value2’

do

        /* commands here */

done
```

- PS1: No need to define the parameter before going into the while/until loop
- PS2: We can use this command to iterate : `seq min pace max`

FUNCTIONS

- A **function** is a set of instructions that allows you to perform multiple tasks with different input parameters

```
functionName ()

{

        /* commands here */

}
```

```
function functionName

{

        /* commands here */

}
```

- functionName //Call function