

Linux

les fondamentaux

Sommaire



Chapitre 1

Introduction

Linux – C'est
quoi ?

Linux est, au sens restreint, le noyau de système d'exploitation Linux, et au sens large, tout système d'exploitation fondé sur le noyau Linux. Créé en 1991 par Linus Torvalds, c'est un logiciel libre destiné en premier lieu pour les ordinateurs personnels compatibles PC, qui avec des logiciels GNU devait constituer un système d'exploitation à part entière.

En termes de parts de marché des OS, Linux peine à s'imposer. Si Linux (en tant que noyau) a connu un certain succès sur du matériel informatique allant des téléphones portables aux superordinateurs, l'OS libre lui n'arrive pas à séduire le grand public sur le bureau, une situation qui a interpellé Linus Torvalds lui-même. En effet, il a remis en cause la fragmentation de l'écosystème comme la principale raison de cet échec.

Maintenant après plus de deux décennies d'existence, le développement de Linux est toujours maintenu activement. Le noyau jouit d'une communauté active.

Constituée d'informaticiens, de geek et de passionné. Ces individus affichent une plus forte adoption de Linux et croient toujours au potentiel que l'OS libre finisse par dominer.



▶ Chapitre 1 Introduction

Linux –
résumé

Introduction – Histoire en résumé

- Linux est un système d'exploitation libre créé en 1991 par Linus Torvalds.
- Il est composé du noyau Linux et de logiciels GNU
- Destiné à être portable sur différents type d'hardware.
- Une communauté active d'informaticiens, de geek et de passionnés.
- Son développement est toujours maintenu activement

▶ Chapitre 1

Linux – rappel

Os = Operating systems = système d'exploitation

Introduction – OS Points important

- Un système d'exploitation est un ensemble de programmes qui permet la gestion des ressources disponibles d'un ordinateur :
- Mémoire / Ram
- CPU / GPU
- Stockage
- Les accès et exploitations des périphériques
- Un environnement propice à accueillir de nouveaux programmes
- Protéger les fichiers contre tout accès non autorisé
- Collecter les informations sur les programmes utilisés ou en cours d'utilisation.

▶ Chapitre 1

Caractéristique

Introduction – Caractéristique

Fiable

Robuste

Puissant

Efficace

Utilise très peu de ressource

Peut fonctionner sur des ordinateurs bas de gamme très peu puissants.

Distribué librement avec son code source

Basé sur le principe du logiciel libre et selon le terme de la licence GPL.

Développé en général bénévolement par sa communauté

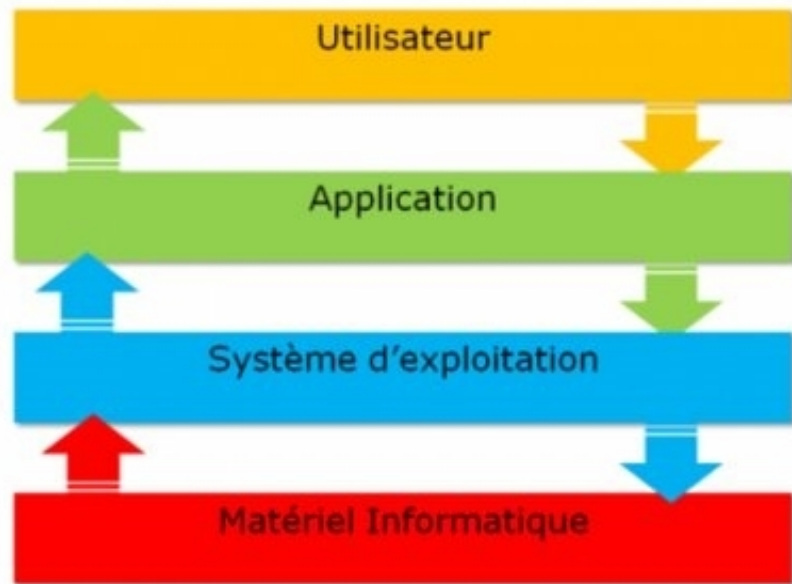


Figure 1. Fonctionnement d'un système d'exploitation



▶ Chapitre 1

Part de marché

Poste bureautique
Téléphone
Serveur

Introduction – Histoire en résumé

Poste de travail



Desktop Operating System Market Share Worldwide - March 2022

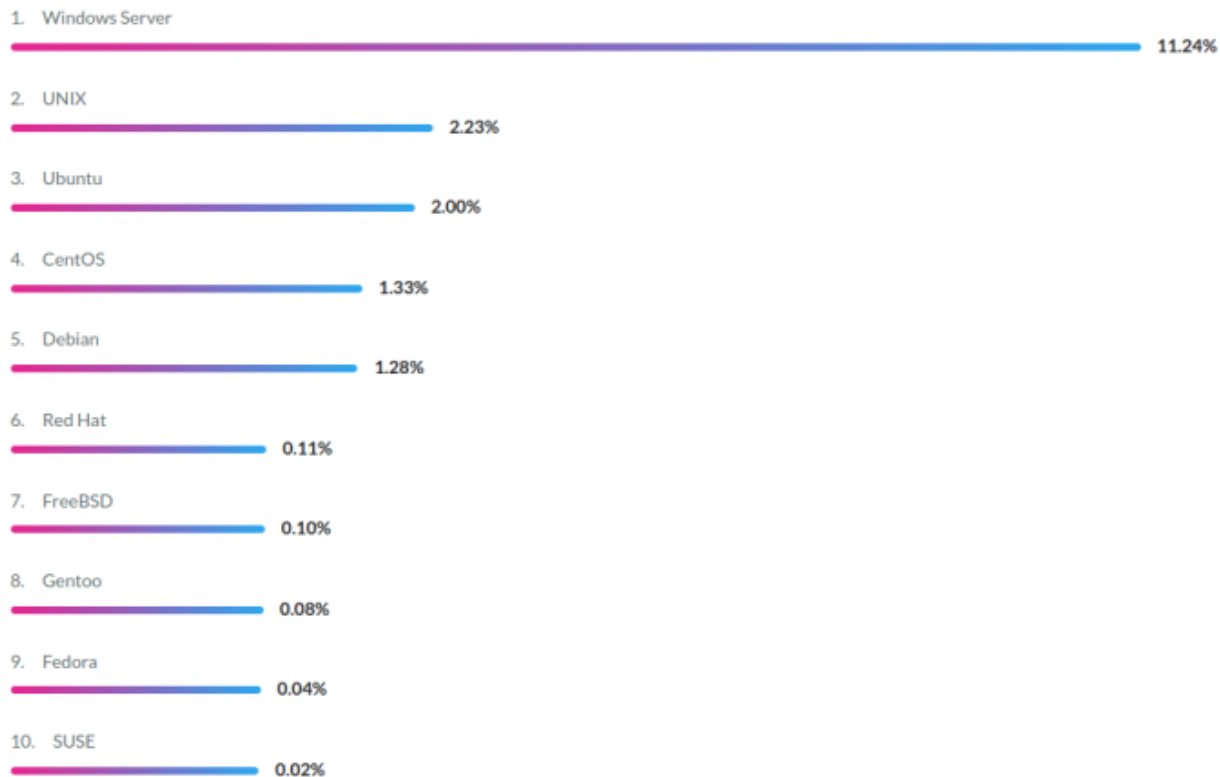
Smartphone



Mobile Operating System Market Share Worldwide - March 2022

Introduction – Histoire en resumé

Serveur





► Chapitre 1

Linux – Distributions, distrib

C'est quoi une distrib ?
Les plus connus
Leur utilité

Une distribution Linux, appelée aussi distribution GNU/Linux lorsqu'elle contient les logiciels du projet GNU, est un ensemble cohérent de logiciels, la plupart étant des logiciels libres, assemblés autour du noyau Linux, et formant un système d'exploitation pleinement opérationnel.

Le terme « distribution » est calqué sur l'anglais software distribution qui signifie « collection de logiciels » en français.

Il existe une très grande variété de distributions Linux, chacune ayant des objectifs et une philosophie particulière.

Ils partagent cependant un noyau commun et un certain nombre de commande d'Unix.

Les éléments les différenciant principalement sont :

Introduction – Distributions, distrib

La convivialité (facilité de mise en œuvre)

L'intégration (taille du parc de logiciels validés distribués)

Leur notoriété (communauté, forum etc...)

Introduction – Distributions, distrib

Leur fréquence de mise à jour

Leur gestion des paquets

Le mainteneur de la distribution (généralement une entreprise ou une communauté).

Les distributions Linux sont des ensembles cohérents de logiciels, la plupart étant des logiciels libres, assemblés autour du noyau Linux, et formant un système d'exploitation pleinement opérationnel. Il existe une très grande variété de distributions Linux, chacune ayant des objectifs et une philosophie particulière.



► Chapitre 1

Linux – Distributions, distrib

Les plus connus

Leur utilité

Introduction – Distributions, distrib

Debian

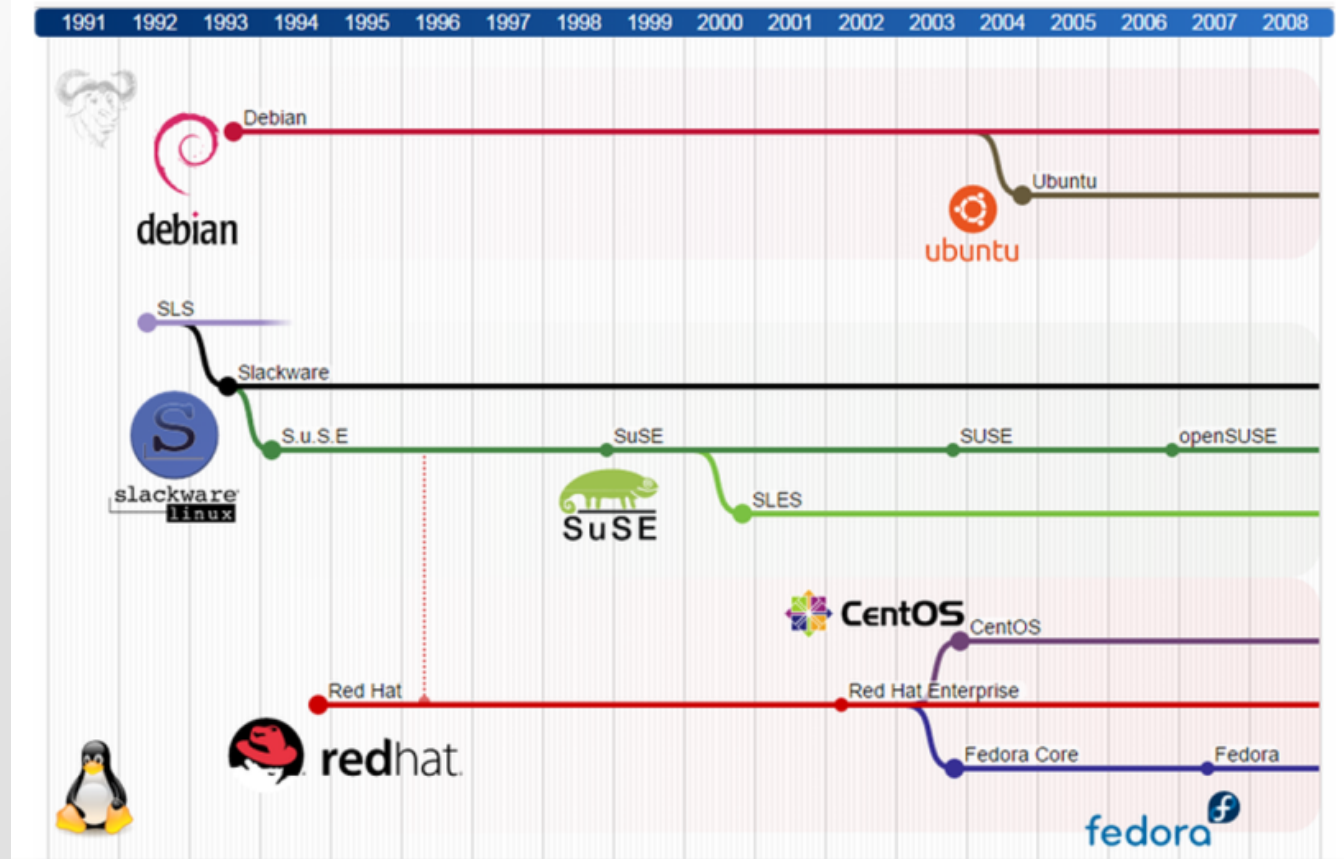
Ubuntu

Redhat

Fedora

CentOs

Suse



Introduction – Distributions, distrib (Debian)

- Une distribution non commerciale
- Développée par une organisation à but non lucratif
- Se veut universelle

Leur philosophie :

La distinction entre distributions non commerciales et commerciales est importante



Introduction – Distributions, distrib (Ubuntu)

- Fondé sur Debian
- Développée par une entreprise privé

---version--- Stable tous les 6 mois

---version--- LTS - *Long Term Support* (« Support long terme ») tous les 2 ans

Leur philosophie :
"Freedom"



Introduction – Distributions, distrib (Redhat)

- Moins de paquets, mais testé en continue, support de 10 ans pour chaque version
- Développée par une entreprise privé

Leur philosophie :

Red Hat is a meritocracy where reputation is earned by how well you help others succeed. We succeed as an open source technology company when we create more open source winners. And we succeed as individuals when we help create more winners within our ranks.





▶ Chapitre 2

concepts

Fichier, tout est
fichier

- "Everything is a file" décrit l'une des caractéristiques des systèmes Unix et de ses dérivés
 - Les documents, répertoires, disques durs, modems, clavier, imprimantes
 - Cette approche permet de diminuer le nombre d'outil dont vous pouvez avoir besoin
-
- Vous pouvez-configurer votre imprimante par exemple avec un simple éditeur de texte
-
- Vous en retrouverez 5 différents sous linux, regroupé sous 3 familles :

| Type name |
|--------------------|
| Socket |
| Symbolic link |
| Regular file |
| Block special file |
| Directory |
| Character device |
| FIFO (named pipe) |

Fichiers spéciaux

Fichier régulier

Fichier de répertoire

Socket : Fichier Socket -

Celles-ci sont associées à un numéro de port.
Les ports sont des numéros allant de 0 à 2 puissance 16-1 inclus (soit 65535). Chacun de ces ports est associé à une application (à savoir que les 1024 premiers ports sont réservés à des utilisations bien précises).

Les sockets servent à établir une transmission de flux de données (octets) entre deux machines ou applications.

| Type name |
|--------------------|
| Socket |
| Symbolic link |
| Regular file |
| Block special file |
| Directory |
| Character device |
| FIFO (named pipe) |

58.6.3 Transmission Control Protocol (TCP)

TCP provides a reliable, connection-oriented, bidirectional, byte-stream communication channel between two endpoints (i.e., applications), as shown in Figure 58-8. In order to provide these features, TCP must perform the tasks described in this section. (A detailed description of all of these features can be found in [Stevens, 1994].)

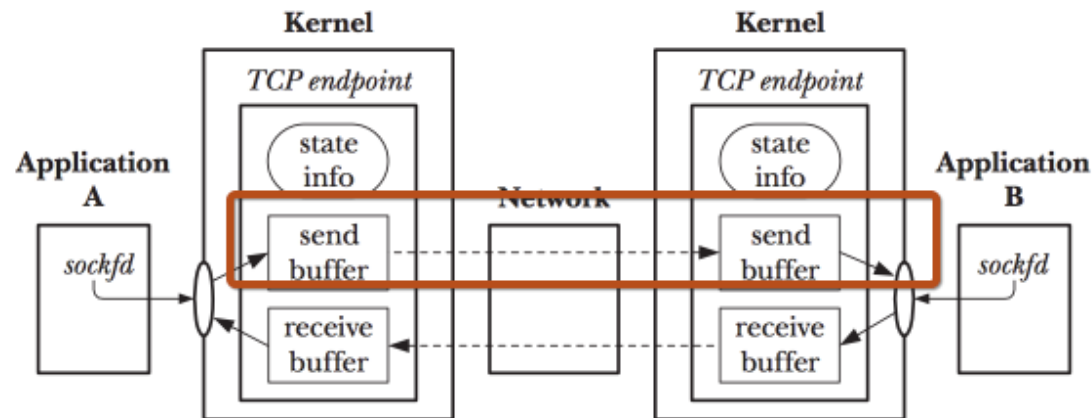


Figure 58-8: Connected TCP sockets

Socket : Fichier Symbolique -

Certains fichiers ne sont que des pointeurs vers d'autres fichiers. On les appelle des liens symboliques, "symlic" .

Un lien symbolique contient le chemin relatif ou absolu d'un fichier ou d'un répertoire qui peut être n'importe où dans l'arborescence.

Il permet de faire référence à ce fichier ou ce répertoire à un autre endroit.

Pour simplifier c'est un raccourci

| Type name |
|--------------------|
| Socket |
| Symbolic link |
| Regular file |
| Block special file |
| Directory |
| Character device |
| FIFO (named pipe) |

Socket : Fichier Régulier -

On l'appelé "régulier" principalement pour le distinguer des autres types de fichiers spéciaux.

La plupart des fichiers utilisés directement par un utilisateur humain sont des fichiers ordinaires. Par exemple, les fichiers exécutables, les fichiers texte et les fichiers image sont des fichiers réguliers.

| Type name |
|--------------------|
| Socket |
| Symbolic link |
| Regular file |
| Block special file |
| Directory |
| Character device |
| FIFO (named pipe) |

Block: fichier spécial bloc

Un fichier spécial bloc sert d'interface directe avec un périphérique.

Un périphérique de bloc est un périphérique qui effectue des I/O de données en unités de blocs.

Exemples de fichiers spéciaux de bloc :

/dev/sdxn - partitions montées

| Type name |
|--------------------|
| Socket |
| Symbolic link |
| Regular file |
| Block special file |
| Directory |
| Character device |
| FIFO (named pipe) |

Directory : fichier de répertoire

Un répertoire est un type de fichier qui contient uniquement les informations nécessaires pour l'accès aux fichiers qu'il contient.

"." symbolise le dossier courant

".." symbolise le dossier parent

| Type name |
|--------------------|
| Socket |
| Symbolic link |
| Regular file |
| Block special file |
| Directory |
| Character device |
| FIFO (named pipe) |

Character device : Fichiers de périphériques

Un fichier de caractères est un fichier matériel qui lit et écrit des données caractère par caractère .

Ces fichiers fournissent un flux série d'entrées ou de sorties et fournissent un accès direct aux périphériques matériels.

Le terminal, les ports série, etc. sont des exemples de ce type de fichier.

| Type name |
|--------------------|
| Socket |
| Symbolic link |
| Regular file |
| Block special file |
| Directory |
| Character device |
| FIFO (named pipe) |

FIFO (Pipe file) : Fichiers FIFO

Un fichier spécial FIFO envoie des données d'un processus à un autre de manière à ce que le processus récepteur lise les données selon le principe du premier entré, premier sorti (FIFO).

Un fichier spécial FIFO est également appelé un tuyau ou un FIFO.

Un fichier spécial FIFO peut également être partagé par plusieurs processus.

| Type name |
|--------------------|
| Socket |
| Symbolic link |
| Regular file |
| Block special file |
| Directory |
| Character device |
| FIFO (named pipe) |

Le premier caractère qui accompagne les droits est toujours le symbole du type de fichier

```
marchal@marchal-OMEN-Laptop-15-en1xxx:/dev$ ls -l
total 0
crw-r--r--  1 root  root    10,  235 oct.  23 01:00
dwxr-xr-x   2 root  root    800 oct.  23 01:01
crw-rw----  1 root  disk    10,  234 oct.  23 01:00
dwxr-xr-x   3 root  root     60 oct.  23 01:00
dwxr-xr-x   2 root  root   4940 oct.  23 01:02
crw--w----  1 root  tty      5,    1 oct.  23 01:01
lwxrwxrwx   1 root  root     11 oct.  23 01:00
dwxr-xr-x  18 root  root    380 oct.  23 01:00
crw-----  1 root  root    10,  124 oct.  23 01:00
crw-----  1 root  root    10,  203 oct.  23 01:00
dwxr-xr-x   8 root  root    160 oct.  23 01:00
dwxr-xr-x   2 root  root     60 oct.  23 01:00
dwxr-xr-x   3 root  root    140 oct.  23 01:00
crw-----  1 root  root   510,    0 oct.  23 01:00
crw-----  1 root  root    10,  126 oct.  23 01:00
crw-rw----  1 root  video  29,    0 oct.  23 01:00
```

| Symbole | Meaning |
|---------|----------------------|
| - | Fichier regulier |
| d | directory |
| l | link |
| c | Fichier de caractère |
| s | Socket |
| p | FIFO |
| b | block |

▶ Chapitre 2

concepts

Arborescence

Chapitre 2

Arborescence

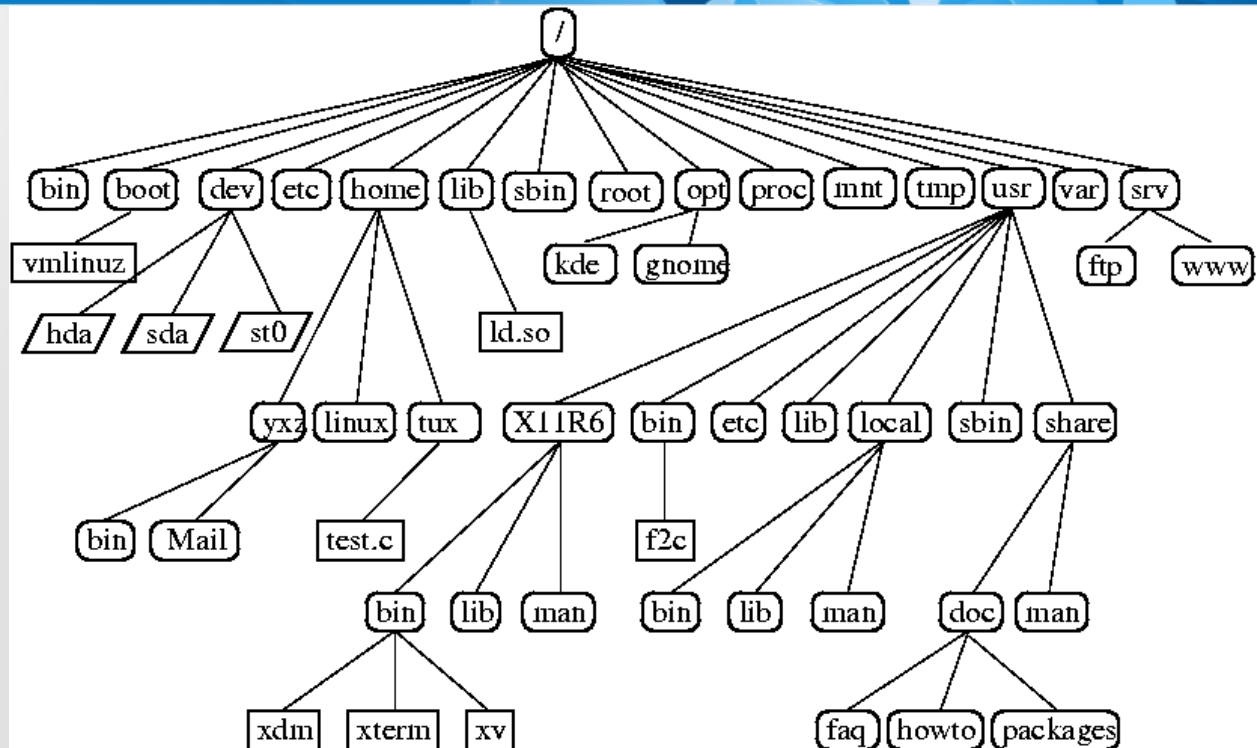
Filesystem Hierarchy Standard (« norme de la hiérarchie des systèmes de fichiers », abrégé en FHS)

L'arborescence des systèmes linux se veut en arbre ou chaque dossier peut contenir des sous-dossiers si autorisés.

La racine "/" étant le point d'origine

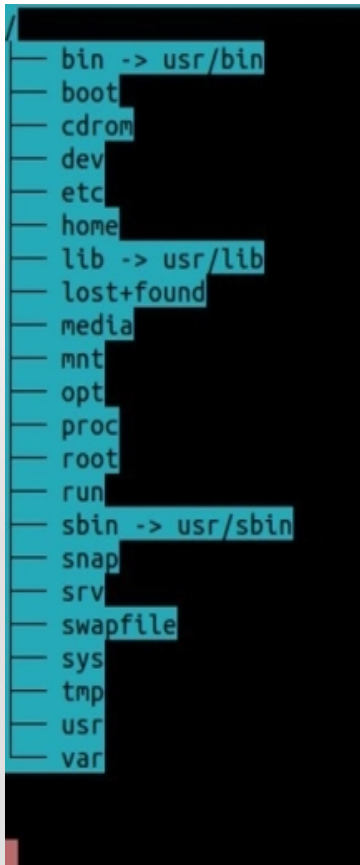
"." répertoire courant

".." répertoire parent



Chapitre 2

Arborescence



/ = la racine

/bin = Fichier **b**inaire

/dev = Driver, périphérique externe, disque externe, volume

/etc = Dossier config des programmes, services

/home = Répertoire personnel des utilisateurs

/lib = Librairie partagé par plusieurs programmes

/lost+found = Fichiers endommagé, nom du fichier perdu, un repertoire par partition

/media = Point de montage automatique

/mnt = Point de montage fait par l'utilisateur

/opt = Optional add-on software packages

/proc = Dossier des processus

/root = Répertoire de root

/run = Contient des données d'informations système décrivant le système depuis son démarrage.

/sbin = Fichier **b**inaire à destination des admins, root. Commande qui peut altérer le système

/snap = ou flatpack, dossier ou des applications conçu pour être portable sur de multiple distribution sont installé via l'équivalent apt-get

/srv = Ressources destinées à être utilisé par de multiples utilisateurs depuis des hôte différents

/swapfile = Fichier, Ram artificiel (prise du disque dur)

/sys = Systemes, dossier lié au kernel

/tmp = Dossier temporaire, supprime le contenu à chaque arrêt ou tous les 10 jours

/usr = Répertoire qui contient des données partageables et en lecture seule à destination des utilisateurs

/var = Fichiers de données variables. Il s'agit notamment des répertoires et des fichiers spool, des données d'administration et de journalisation.

/boot = Fichier important au démarrage de la machine

▶ Chapitre 2

Premier pas

Interagir avec le
système

Via le terminal / invite de commande / interpréteur qui un outil puissant qui permet a un utilisateur d'interagir avec le système d'exploitation.

Il en existe plusieurs et utilise le langage **bash** celui que nous allons voir.
Il est natif aux systèmes linux/Unix.

Via cet outil nous allons pouvoir communiquer avec la machine dans un langage qu'elle comprend.

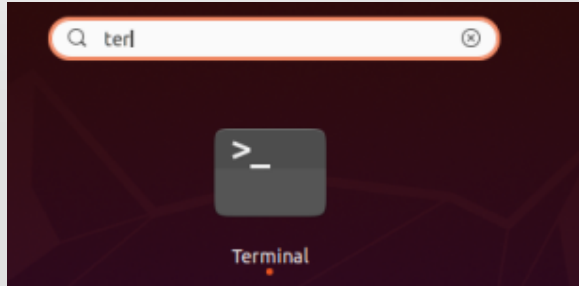
► Chapitre 2

Premier pas

Le Terminal
Invite de commande
Interpréteur

Le terminal est l'un des premiers éléments avec lequel un utilisateur va interagir sur un système sans interface graphique ou qui démarre en mode rescue / sans echec.

Sinon ctrl + alt + t est le raccourci pour lancer une invite de commande, ou depuis l'onglet "Activités" ou la touche os (en bas à gauche de votre clavier) puis rechercher "Terminal"



- Utilisateurs :

Nous avons deux types d'utilisateurs :

- root : c'est l'administrateur du système . Il possède tous les droits, peut se déplacer où il veut, modifier les fichiers qu'il souhaite... (Attention ! Quand on travaille en root, il n'y a pas de message d'avertissement avant d'exécuter une commande dangereuse).
 - user : c'est un utilisateur distinct de root. Il peut appartenir à différents groupes qui lui permettent d'avoir ou non certains droits. Le groupe « sudo » lui permet d'exécuter des commandes avec les droits administrateurs (contrairement à root, exécuter des commandes avec « sudo » nécessite le mot de passe de l'utilisateur et affiche des avertissements demandant confirmation avant l'exécution de la commande).
- u La commande « su <USER> » permet de changer d'utilisateur. Par défaut, « su - » tout seul permet passer en root.

```
jeanne@PC-M2i:~$ su
Password:
root@PC-M2i:/home/jeanne#
```

```
root@PC-M2i:~# su jeanne
jeanne@PC-M2i:/root$
```

Chapitre 2

Premiers pas

```
jeanne@PC-M2i:~$
root@PC-M2i:~#
```

Diagram illustrating the components of a terminal prompt:

- USER**: The username (jeanne or root).
- HOST**: The machine name (PC-M2i).
- répertoire courant**: The current directory (~).
- simple utilisateur**: The prompt character (\$) for a regular user.
- super utilisateur**: The prompt character (#) for the root user.

Autres exemples :

```
root@PC-M2i:/etc#
root@PC-M2i:/#
```

Ici on a l'utilisateur root sur la machine « PC-M2i » avec pour la première ligne en répertoire courant « etc » et pour la deuxième ligne, la racine du système de fichier.

ex : jeanne et root.

root a tous les droits, c'est l'administrateur.

Le répertoire courant sur les deux premiers exemples est « ~ ». Ce qui signifie qu'on se trouve dans le « home directory » de l'utilisateur (son répertoire personnel).

On remarque en dernier qu'on a soit « \$ », ou « # ». On peut différencier ainsi une commande qui sera exécuté par un simple utilisateur (\$) ou par root « # ».

Chapitre 2

Premiers pas

- Les aides intégrés aux systèmes

`man -a <commande>`: voir toutes les pages de manuels pour une commande

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ man -a man
--Man-- prochain : man(1) [ voir (entrée) | passer (Ctrl-D) | quitter (Ctrl-C) ]
--Man-- prochain : man(7) [ voir (entrée) | passer (Ctrl-D) | quitter (Ctrl-C) ]
--Man-- prochain : man(7) [ voir (entrée) | passer (Ctrl-D) | quitter (Ctrl-C) ]
```

`<commande> -h [--help (version longue)]` : affiche comment utiliser une commande

`Info -a [--all]` : n'est pas `man` mais fournit de la documentation

`help` : commande native au système indique comment utiliser une commande

`whatis` : courte indicatif

`apropos` : sujet lié

Sinon google

- Commande de base

| | |
|----------------|--|
| apropos | Search information about a command or subject. |
| cat | Show content of one or more files. |
| cd | Change into another directory. |
| exit | Leave a shell session. |
| file | Get information about the content of a file. |
| info | Read Info pages about a command. |
| logout | Leave a shell session. |
| ls | List directory content. |
| man | Read manual pages of a command. |
| passwd | Change your password. |
| pwd | Display the current working directory. |

which : affiche le répertoire du programme

clear : enleve ce qui est affiché dans le terminal

- Raccourcis clavier dans le terminal

| Key or key combination | Function |
|---|--|
| Ctrl+A | Move cursor to the beginning of the command line. |
| Ctrl+C | End a running program and return the prompt, see Chapter 4 . |
| Ctrl+D | Log out of the current shell session, equal to typing exit or logout . |
| Ctrl+E | Move cursor to the end of the command line. |
| Ctrl+H | Generate backspace character. |
| Ctrl+L | Clear this terminal. |
| Ctrl+R | Search command history, see Section 3.3.3.4 . |
| Ctrl+Z | Suspend a program, see Chapter 4 . |
| ArrowLeft and ArrowRight | Move the cursor one place to the left or right on the command line, so that you can insert characters at other places than just at the beginning and the end. |
| ArrowUp and ArrowDown | Browse history. Go to the line that you want to repeat, edit details if necessary, and press Enter to save time. |
| Shift+PageUp and Shift+PageDown | Browse terminal buffer (to see text that has "scrolled off" the screen). |
| Tab | Command or filename completion; when multiple choices are possible, the system will either signal with an audio or visual bell, or, if too many choices are possible, ask you if you want to see them all. |
| Tab Tab | Shows file or command completion possibilities. |


► Chapitre 3

Manipuler l'arborescence

Se déplacer sur le système


- Se déplacer dans l'arborescence commande `cd` :

On peut connaître le répertoire dans lequel on se situe actuellement (répertoire courant) dans le **shell** ou bien avec la commande `pwd`:



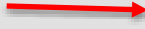
```
jeanne@PC-M2i:/usr$ pwd
/usr
```

→ Se déplacer dans un répertoire précis




```
jeanne@PC-M2i:~$ cd /usr/share
jeanne@PC-M2i:/usr/share$
```

→ Se déplacer dans le répertoire précédent




```
jeanne@PC-M2i:/usr/share$ cd ..
jeanne@PC-M2i:/usr$
```

→ Se déplacer à la racine



```
jeanne@PC-M2i:/usr$ cd /
jeanne@PC-M2i:/$
```

→ Se déplacer dans son home directory



```
jeanne@PC-M2i:/usr/share$ cd ~
jeanne@PC-M2i:~$
```

Chapitre 3

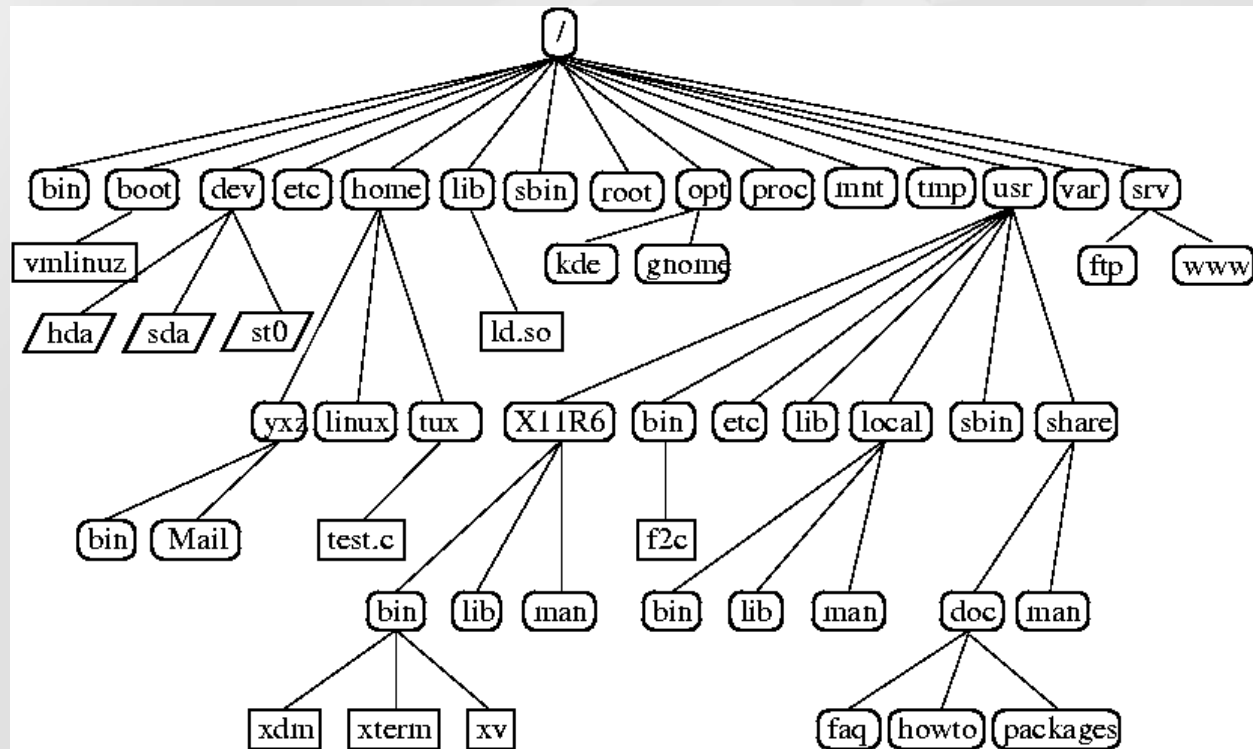
Manipuler l'arborescence

Voici une illustration d'une arborescence d'un système de fichier.

2 Notions sont importantes pour bien manipuler l'arborescence.

Chemins Relatif

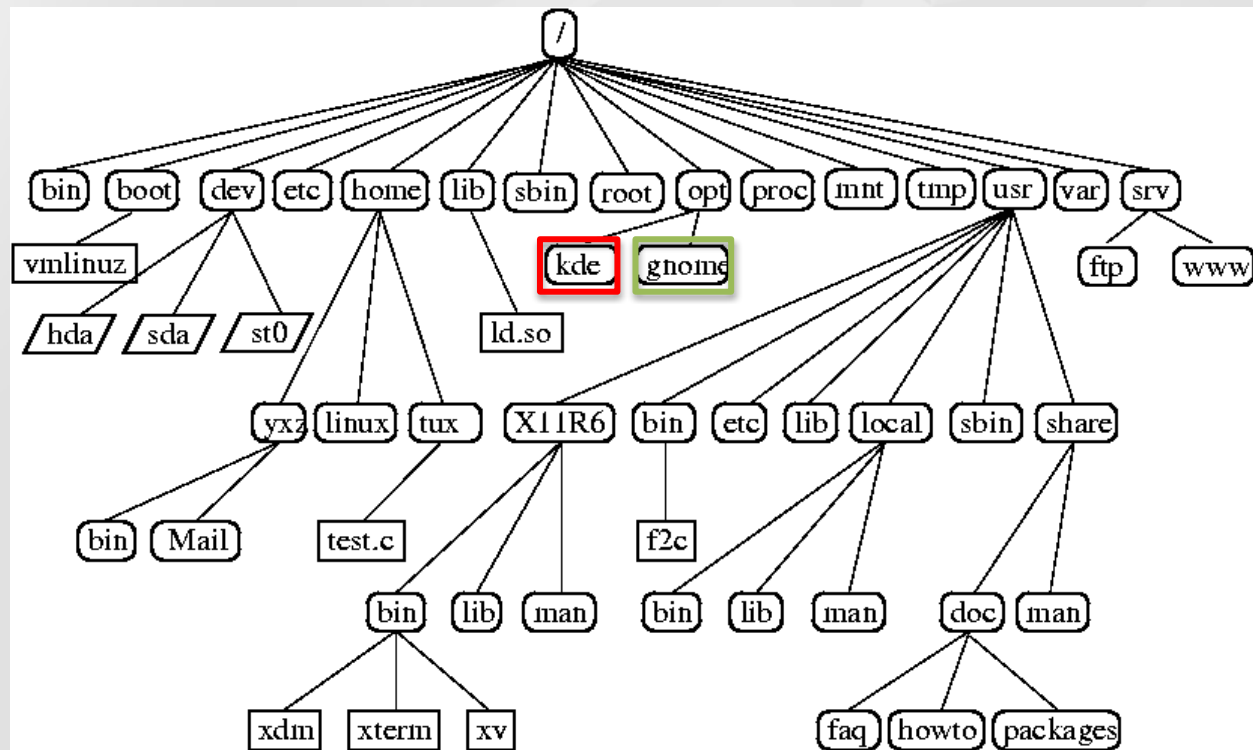
Chemin Abosolu



Chapitre 3

Manipuler l'arborescence

Pour naviguer du dossier
kde vers gnome



: Dossier courant

: Dossier cible

Pour naviguer du dossier
kde vers gnome

Relatif: `cd ../gnome`

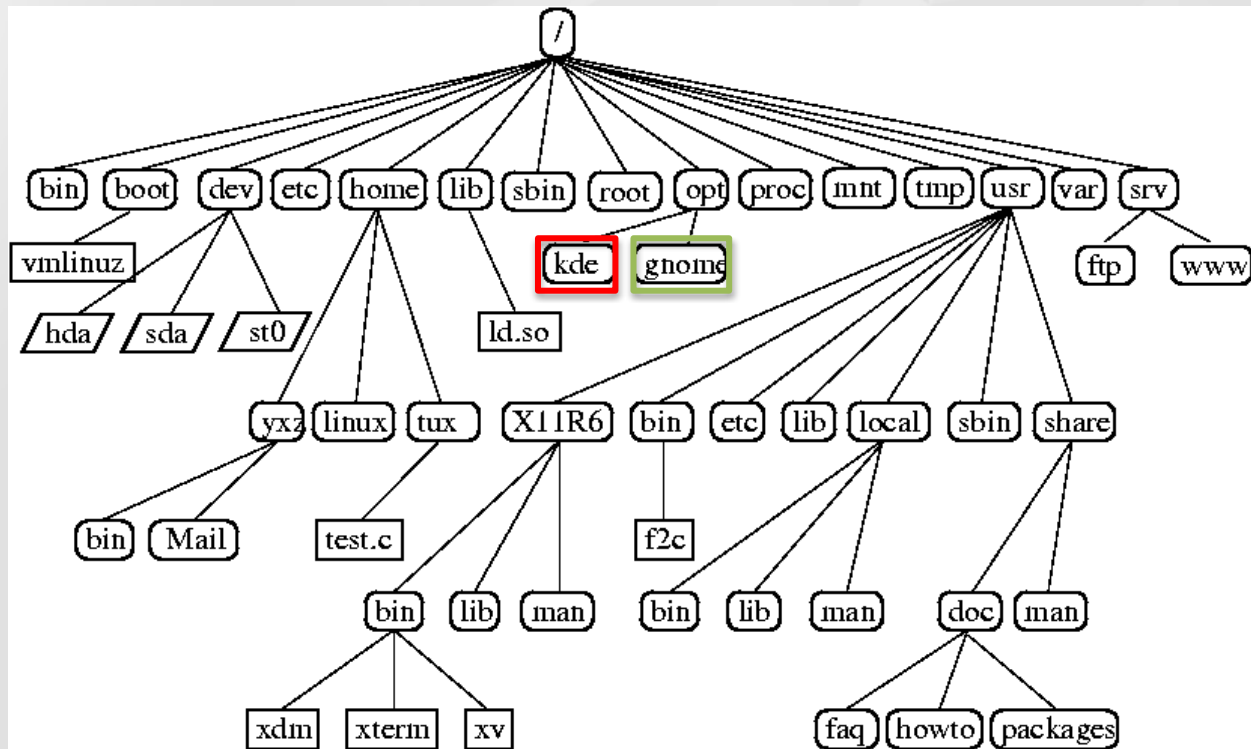
En relatif vous partez du
dossier courant

Absolu: `cd /opt/gnome`

En absolu vous partez de la
racine

 : Dossier courant

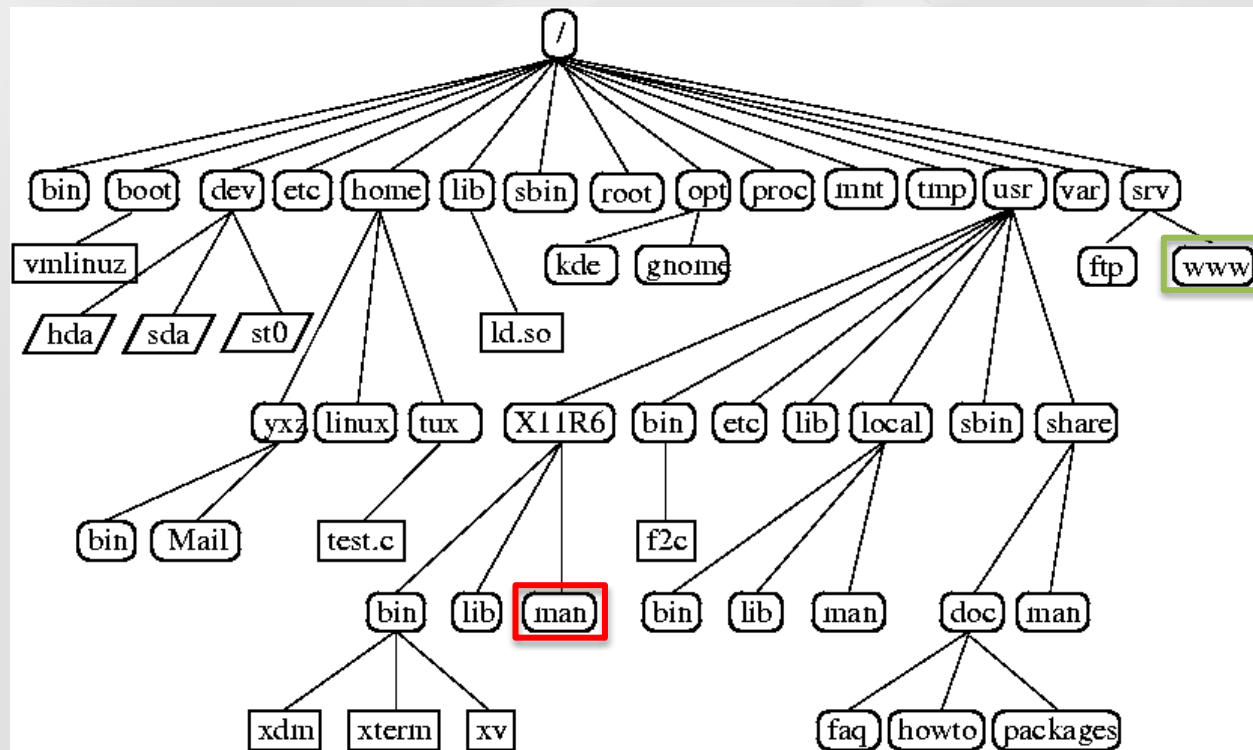
 : Dossier cible



Chapitre 3

Manipuler l'arborescence

De man vers www



 : Dossier courant

 : Dossier cible

Chapitre 3

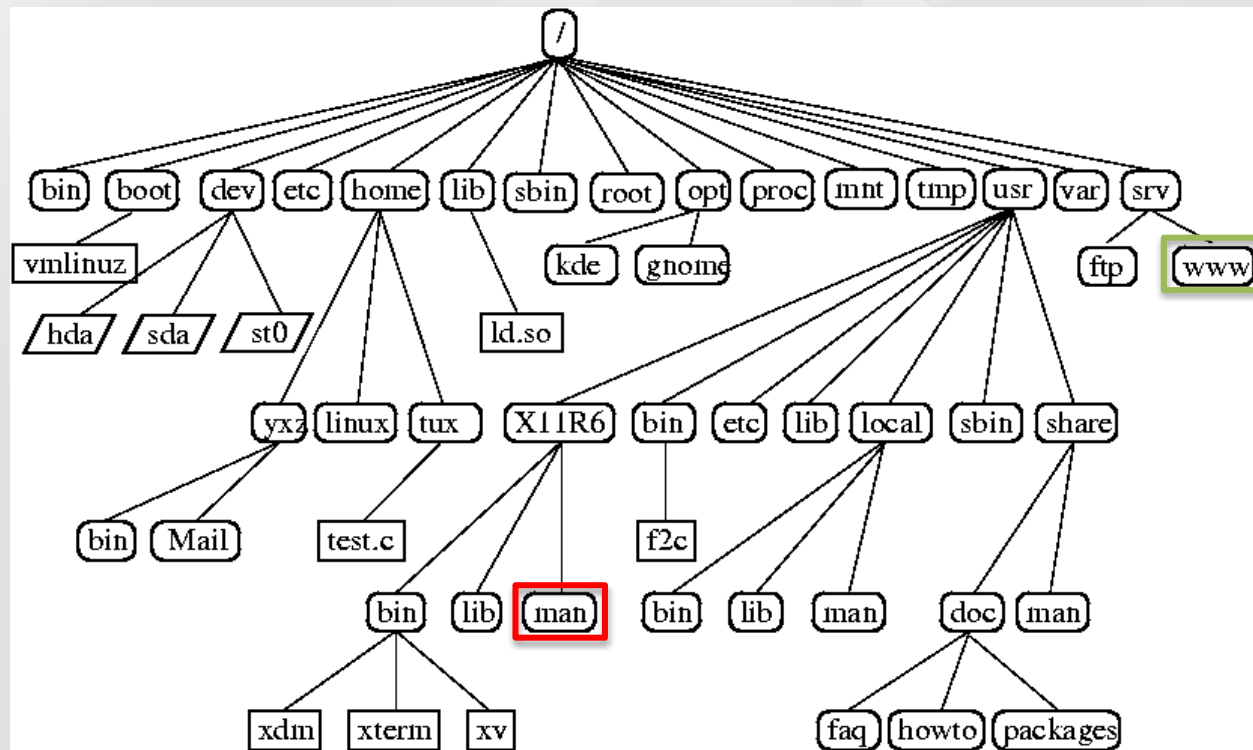
Manipuler l'arborescence

Relatif:

`../../srv/www`

Absolu:

`/srv/www`



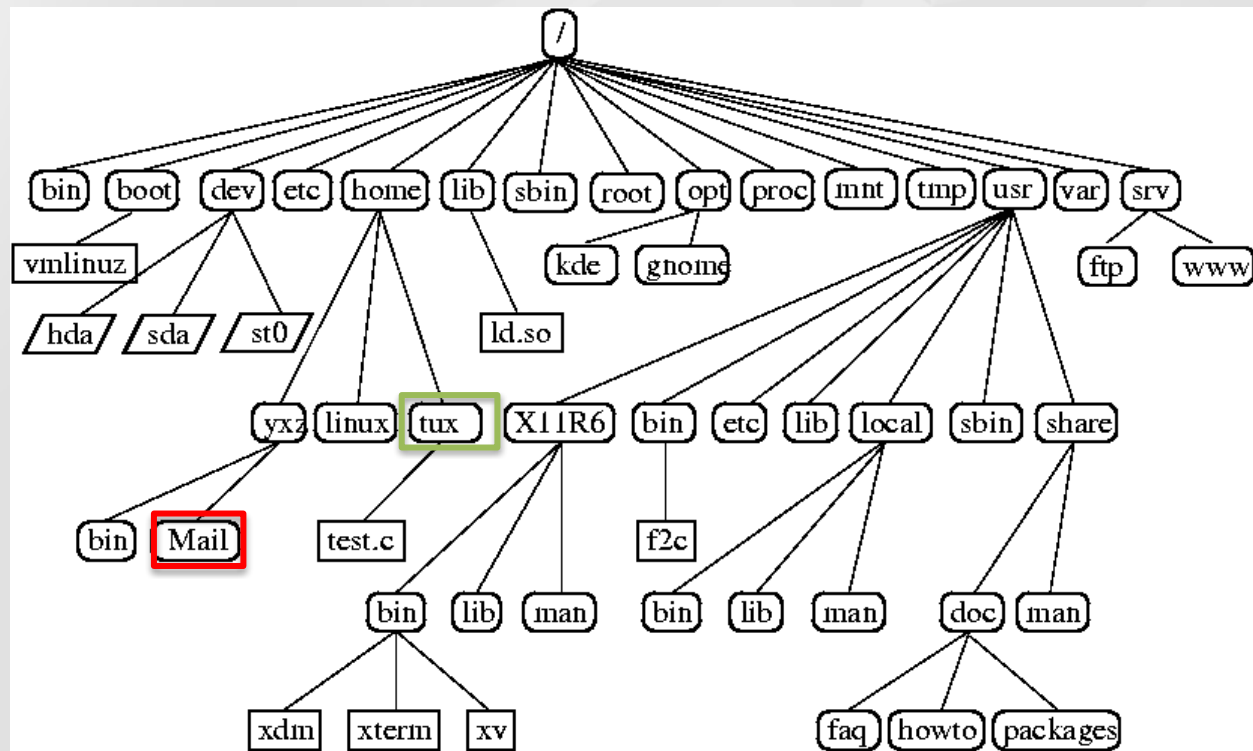
: Dossier courant

: Dossier cible

Chapitre 3

Manipuler l'arborescence

De Mail à tux



: Dossier courant

: Dossier cible

Chapitre 3

Manipuler l'arborescence

Relatif:

`cd ../../tux`

Absolu:

`cd /home/tux`

À noter que :

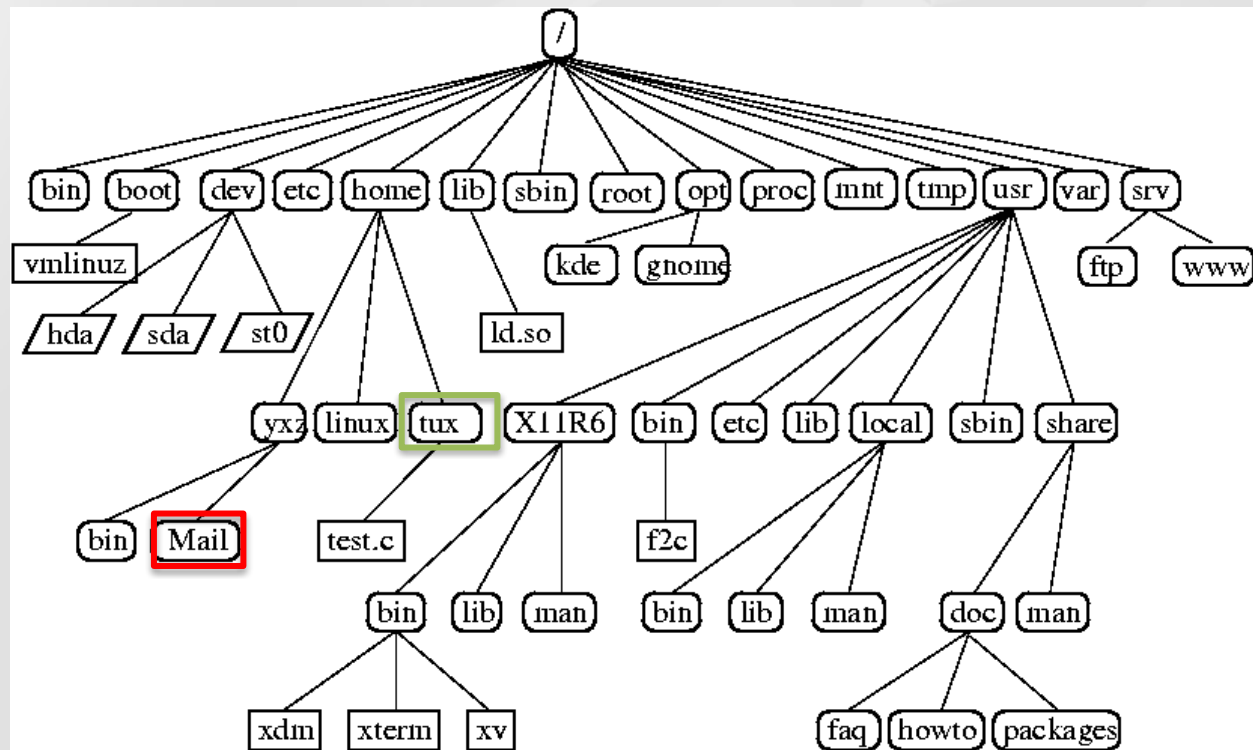
(ok) `/home/tux/`

(ok) `../../tux/`

le dernier "/" est optionnel

: Dossier courant

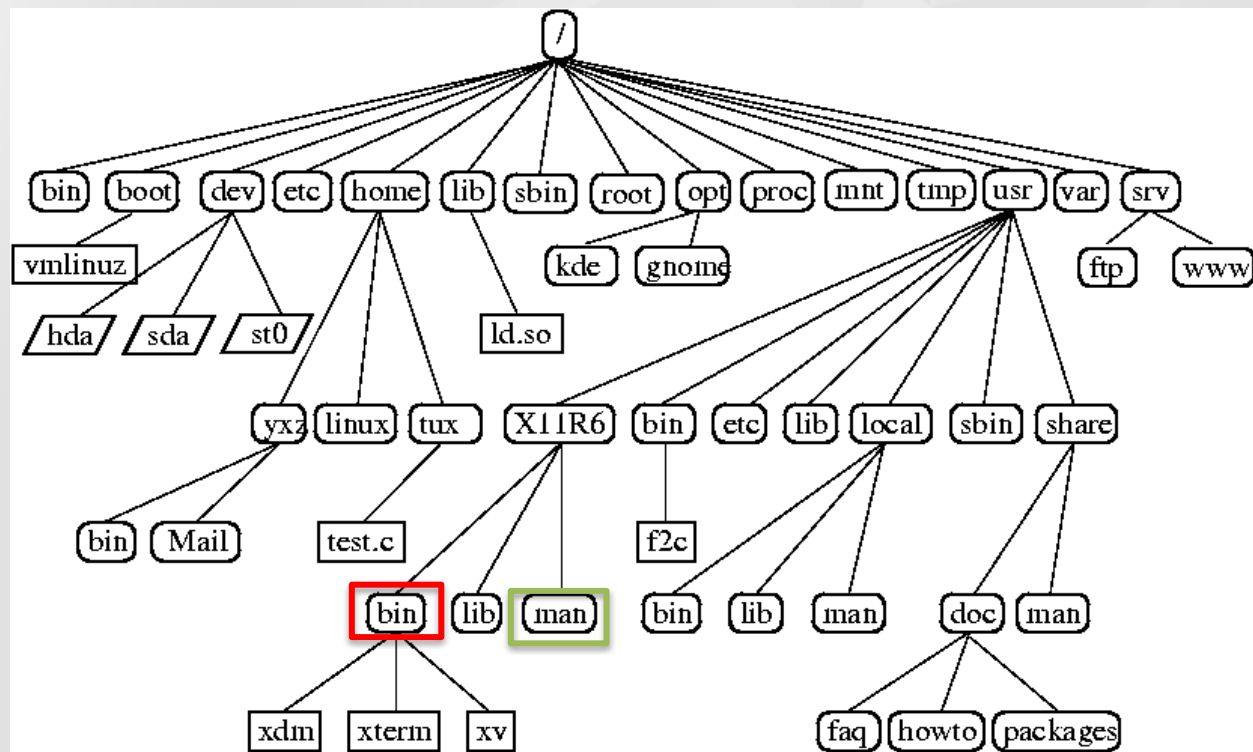
: Dossier cible



Chapitre 3

Manipuler l'arborescence

De bin à man



 : Dossier courant

 : Dossier cible

Chapitre 3

Manipuler l'arborescence

Relatif:

cd ../man

Absolu:

cd /usr/X11R6

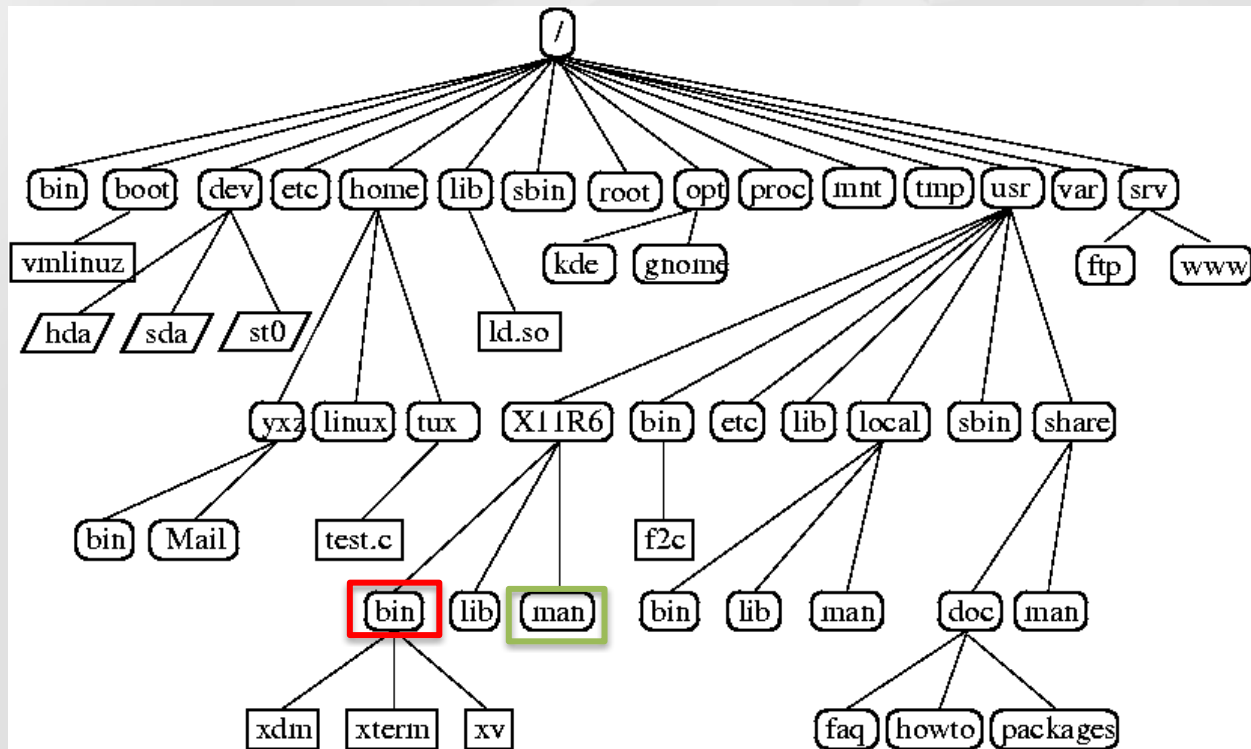
À noter que de manière générale :

Le chemin est **sensible à la casse**.

(ok) /usr/X11R6

(nok) /usr/x11R6

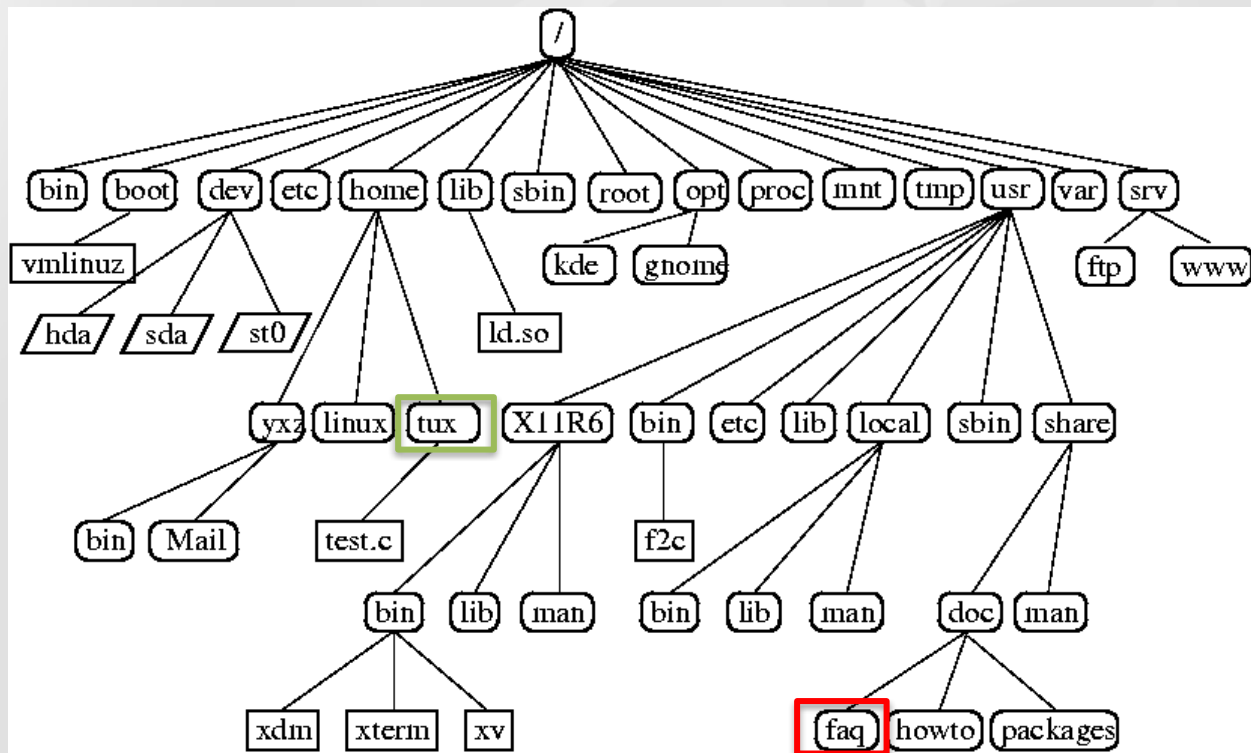
(ok) ../usr/X11R6



 : Dossier courant

 : Dossier cible

De faq à tux



☐ : Dossier courant

☐ : Dossier cible

Chapitre 3

Manipuler l'arborescence

Relatif:

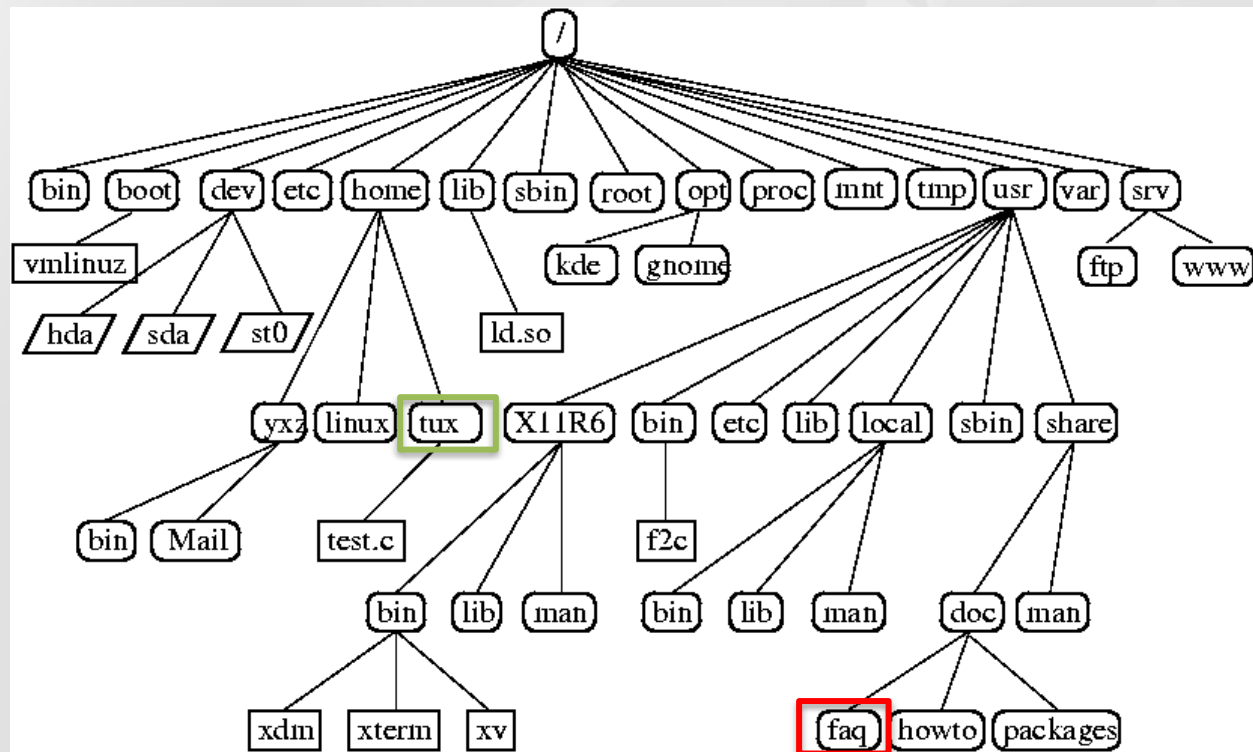
cd ~

Absolu:

cd /home/tux

A noter que :

~ symbolise le répertoire home
de l'utilisateur connecté



: Dossier courant

: Dossier cible

► Chapitre 3

Métacaractères et Variables

Métacaractères c'est quoi ?

- Les Métacaractères c'est quoi ?

Ce sont des caractères spéciaux.

Ils sont interprétés différemment par le système que par leur simple valeur textuelle, ce sont des symboles

| Symbol | Meaning |
|--------|--|
| > | Output redirection |
| >> | Output redirection (append) |
| < | Input redirection |
| * | File substitution wildcard; zero or more characters |
| ? | File substitution wildcard; one character |
| [] | File substitution wildcard; any character between brackets |
| `cmd` | Command Substitution |
| ~ | Current user directory |

Chapitre 3

Caractères spéciaux

| Symbol | Meaning |
|--------|--|
| & | Run command in the background, Background Processes |
| # | Comment |
| \$ | Expand the value of a variable |
| \ | Prevent or escape interpretation of the next character |
| << | Input redirection |

Chapitre 3

Caractères spéciaux

| Symbol | Meaning |
|-------------------------|--|
| <code>\$(cmd)</code> | Command Substitution |
| <code> </code> | The Pipe (<code> </code>) |
| <code>;</code> | Command sequence, Sequences of Commands |
| <code>[]</code> | File substitution wildcard; any character between brackets |
| <code> </code> | OR conditional execution |
| <code>&&</code> | AND conditional execution |
| <code>()</code> | Group commands, Sequences of Commands |

Chapitre 3

Caractères spéciaux

Les métacaractères du **shell** permettent :

- De construire des chaînes de caractères génériques :

* désigne une chaîne de caractères quelconque ;

? désigne un caractère quelconque ;

[...] désigne les caractères entre crochets, définis par énumération ou par un intervalle.

Exemples :

[Aa] désigne les caractères **A** ou **a** ;

[0-9a-zA-Z] désigne un caractère alphanumérique quelconque.

Remarque :

[!0-9] désigne l'ensemble des caractères sauf les chiffres.

- De modifier l'interprétation d'une commande :

; sépare deux commandes sur une même ligne ;

' délimite une chaîne de caractères contenant des espaces (à l'intérieur, tous les métacaractères perdent leur signification) ;

" délimite une chaîne de caractères contenant des espaces (à l'intérieur, tous les métacaractères perdent leur signification, à l'exception des métacaractères ` et \$) ;

` "capture" la sortie standard pour former un nouvel argument ou une nouvelle commande ;

\ annihile la signification du métacaractère qui suit ;

{ et } permettent de regrouper un ensemble de commandes et de les exécuter dans le "shell courant" ;

(et) permettent de regrouper un ensemble de commandes et de les exécuter dans un "shell fils".



► Chapitre 3

Métacaractères et Variables

Variables d'environnements

Les variables d'environnement sont un ensemble de valeurs dynamiques nommées voué à être utilisé par des programmes.

Ces variables vous permettent de personnaliser le comportement d'applications et de services spécifiques.

Chaque variable contient un nom et une valeur associée.

En général, le nom est en **MAJUSCULES** et ces valeurs sont sensibles à la casse.

Pour afficher la valeur d'une variable:
echo \$VAR

| System Variable | Meaning |
|-----------------|---|
| LANG | Used to determine the locale category for any category not specifically selected with a variable starting with LC_. |
| PATH | The search path for commands. It is a colon-separated list of directories in which the shell looks for commands. |
| PS1 | Your prompt settings. |

| System Variable | Meaning |
|-----------------|---|
| BASH_VERSION | Holds the version of this instance of bash. |
| HOSTNAME | The name of the your computer. |
| CDPATH | The search path for the cd command. |
| HISTFILE | The name of the file in which command history is saved. |

| System Variable | Meaning |
|-----------------|---|
| TMOUT | The default timeout for the read builtin command. Also in an interactive shell, the value is interpreted as the number of seconds to wait for input after issuing the command. If not input provided it will logout user. |
| TERM | Your login terminal type. |
| SHELL | Set path to login shell. |
| DISPLAY | Set X display name |
| EDITOR | Set name of default text editor. |

| System Variable | Meaning |
|-----------------|--|
| HISTFILESIZE | The maximum number of lines contained in the history file. |
| HISTSIZE | The number of commands to remember in the command history. The default value is 500. |
| HOME | The home directory of the current user. |
| IFS | The Internal Field Separator that is used for word splitting after expansion and to split lines into words with the read builtin command. The default value is <space><tab><newline>. |

- **Commande lié**

printenv : affiche les variables d'environnement

export [Arg: NOM_DE_VARIABLE=SA_VALEUR] : affecte une valeur à une variable

set : **e**quivalent à export


unset : supprime une var d'environnement

Pour que le changement soit persistant ->

~/.bashrc || (ou) ~/.profile

Rappel : ~ = current user directory

/etc/environment = pour Tous le Systeme



► Chapitre 3

Métacaractères et Variables

Différence variables locales et globales

- Locales et globales

Une variable est **une variable déclarée** dans un contexte et n'est disponible que dans ce contexte.

Les variables d'environnement ont **une portée locale**. Ce qui signifie que leur valeur est spécifique au **processus** dans lequel ou pour lequel elles ont été définies. Ainsi si vous ouvrez deux terminaux différents, c'est à dire deux processus bash différents, et que vous changez la valeur d'une variable d'environnement dans un terminal, ce changement n'affectera pas l'autre terminal ni aucun autre programme. Ce changement est local, il affecte le processus dans lequel il a été effectué, sans aucune influence sur les autres processus externes.

Pour une portée globale -> /etc/environment

https://doc.ubuntu-fr.org/variables_d_environnement

► Chapitre 4

Manipulation et contenu d'un fichier

Commande wc

wc (en référence aux termes anglais word count, « décompte des mots »)

Permet d'obtenir plusieurs informations au sujet d'un ou plusieurs fichiers :

Le nombre de lignes (plus précisément le nombre de retour à la ligne).

Le nombre de mots (comme son nom l'indique).

Le nombre d'octets.

Lorsqu'une liste de fichiers est donnée en entrée, les statistiques portent à la fois sur les fichiers individuellement et pris dans leur ensemble.

```
$ wc ideas.txt excerpt.txt
  40      149      947 ideas.txt
2294    16638    97724 excerpt.txt
2334    16787    98671 total
```

En première colonne, il s'agit du nombre de retour à la ligne

En deuxième, du nombre de mots.

En dernier du nombre d'octets.


```
wc -l <nom_du_fichier> # affiche le nombre de lignes
wc -c <nom_du_fichier> # affiche le nombre de bytes
wc -m <nom_du_fichier> # affiche le nombre de caractères
wc -L <nom_du_fichier> # indique la longueur de la plus longue ligne
wc -w <nom_du_fichier> # affiche le nombre de mots
```

wc [OPTION] ... [FILE] ...

► Chapitre 4

Manipulation et contenu d'un fichier

Commande head

La commande head

La commande **head** affiche le début d'un fichier.

Syntaxe de la commande head

```
head [-n x] fichier
```

Table 13. Options de la commande head

| Option | Observation |
|-------------|---|
| -n x | Affiche les x premières lignes du fichier |

Par défaut (sans l'option **-n**), la commande head affichera les 10 premières lignes du fichier.

Chapitre 4

head exemple

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ head arbo /etc/dhcp/dhclient.conf
==> arbo <==
/
|— bin -> usr/bin
|— boot
|— dev
|— etc
|— home
|— lib -> usr/lib
|— lost+found
|— media
|— mnt

==> /etc/dhcp/dhclient.conf <==
# Configuration file for /sbin/dhclient.
#
# This is a sample configuration file for dhclient. See dhclient.conf's
#   man page for more information about the syntax of this file
#   and a more comprehensive list of the parameters understood by
#   dhclient.
#
# Normally, if the DHCP server provides reasonable information and does
#   not leave anything out (like the domain name, for example), then
#   few changes must be made to this file, if any.
```

► Chapitre 4

Manipulation et contenu d'un fichier

Commande tail

La commande tail

La commande **tail** affiche la fin d'un fichier.

Syntaxe de la commande tail

```
tail [-f] [-n x] fichier
```

Table 14. Options de la commande tail

| Option | Observation |
|-------------|--|
| -n x | Affiche les x dernières lignes du fichier |

Par défaut, donc sans option - affiche les 10 dernières lignes

| Option | Observation |
|-----------|--|
| -f | Affiche les modifications du fichier en temps réel |

Exemple :

```
$ tail -n 3 /etc/passwd  
sshd:x:74:74:Privilege-separated sshd:/var/empty /sshd:/sbin/nologin  
tcpdump::x:72:72:::/sbin/nologin  
user1:x:500:500:grp1:/home/user1:/bin/bash
```

Avec l'option **-f**, la commande tail ne rend pas la main et s'exécute tant que l'utilisateur ne l'interrompt pas par la séquence **[CTRL] + [C]**. Cette option est très fréquemment utilisée pour suivre les fichiers journaux (les logs) en temps réel.

► Chapitre 4

Manipulation et contenu d'un fichier

Commande file

La commande file

La commande **file** affiche le type d'un fichier.

Syntaxe de la commande file

```
file fichier [fichiers]
```

Exemple :

```
$ file /etc/passwd /etc  
/etc/passwd:  ASCII text  
/etc:         directory
```

man file

► Chapitre 4

Manipulation et contenu d'un fichier

Commande strings

Chapitre 4

strings

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ file arbo
arbo: UTF-8 Unicode text
```

strings : extrait les caractères imprimables des fichiers

Ex: sur le retour de la commande tree sur un fichier texte

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cat arbo
```

```
/
├── bin -> usr/bin
├── boot
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── snap
├── srv
├── swapfile
├── sys
├── tmp
├── usr
└── var
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ strings arbo
```

```
bin -> usr/bin
boot
dev
etc
home
lib -> usr/lib
lost+found
media
mnt
opt
proc
root
run
sbin -> usr/sbin
snap
srv
swapfile
sys
tmp
usr
var
```

Chapitre 4

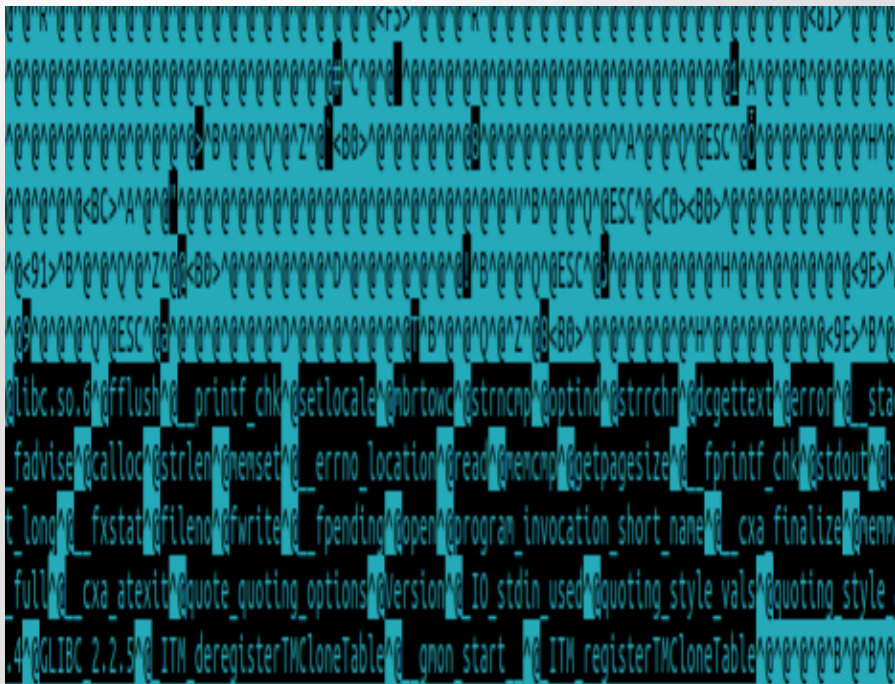
strings

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ type /bin/cat  
/bin/cat est /bin/cat
```

Très pratique pour extraire le texte d'un fichier binaire

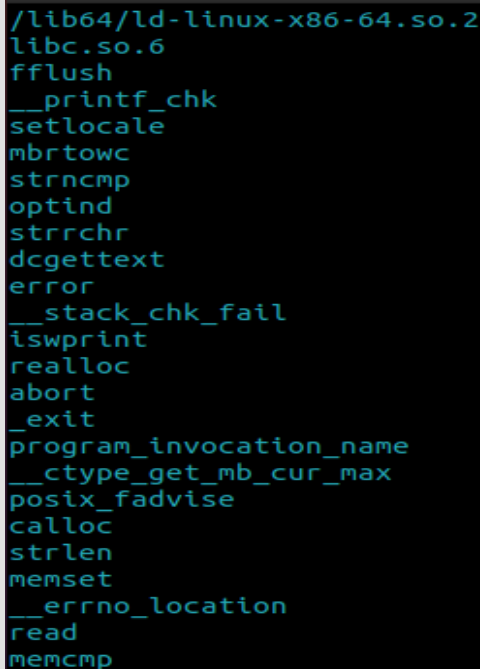
Ex2:

less /bin/cat



A screenshot of the 'less' command output for the file /bin/cat. The output is a dense, noisy pattern of characters and symbols, including letters, numbers, and special characters, which is typical of a binary file being viewed as text.

strings /bin/cat



A screenshot of the 'strings' command output for the file /bin/cat. The output is a list of strings extracted from the binary file, including system library paths like /lib64/ld-linux-x86-64.so.2 and libc.so.6, and various system functions and constants like fflush, __printf_chk, setlocale, mbrtowc, strncmp, optind, strchr, dcgettext, error, __stack_chk_fail, iswprint, realloc, abort, _exit, program_invocation_name, __ctype_get_mb_cur_max, posix_fadvise, calloc, strlen, memset, __errno_location, read, and memcmp.

► Chapitre 4

Manipulation et contenu d'un fichier

Commande od

od : affiche le contenu d'un fichier en octal

cat arbo

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cat arbo
/
├── bin -> usr/bin
├── boot
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── snap
├── srv
├── swapfile
├── sys
├── tmp
├── usr
└── var
```

od arbo

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ od arbo
00000000 005057 112342 161234 100224 112342 020200 064542 020156
00000020 037055 072440 071163 061057 067151 161012 116224 112342
00000040 161200 100224 061040 067557 005164 112342 161234 100224
00000060 112342 020200 062544 005166 112342 161234 100224 112342
00001000 020200 072145 005143 112342 161234 100224 112342 020200
00001200 067550 062555 161012 116224 112342 161200 100224 066040
00001400 061151 026440 020076 071565 027562 064554 005142 112342
00001600 161234 100224 112342 020200 067554 072163 063053 072557
00002000 062156 161012 116224 112342 161200 100224 066440 062145
00002200 060551 161012 116224 112342 161200 100224 066440 072156
00002400 161012 116224 112342 161200 100224 067440 072160 161012
00002600 116224 112342 161200 100224 070040 067562 005143 112342
00003000 161234 100224 112342 020200 067562 072157 161012 116224
00003200 112342 161200 100224 071040 067165 161012 116224 112342
00003400 161200 100224 071440 064542 020156 037055 072440 071163
00003600 071457 064542 005156 112342 161234 100224 112342 020200
00004000 067163 070141 161012 116224 112342 161200 100224 071440
00004200 073162 161012 116224 112342 161200 100224 071440 060567
00004400 063160 066151 005145 112342 161234 100224 112342 020200
00004600 074563 005163 112342 161234 100224 112342 020200 066564
00005000 005160 112342 161234 100224 112342 020200 071565 005162
00005200 112342 161224 100224 112342 020200 060566 005162 000012
0000537
```

Option

-c : printable character

-b : affiche en bytes

Pratique pour voir les caractères non affichables ou qui se ressemblent, ex : O et 0

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ echo "Hello 0 ^C" | od -cb
00000000  H  e  l  l  o      0      ^  C  \n
          110 145 154 154 157 040 060 040 040 040 136 103 012
00000016
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ man od
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ echo "Hello ^C" | od -b
00000000 110 145 154 154 157 040 040 040 040 136 103 012
00000014
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ echo "Hello ^C" | od -c
00000000  H  e  l  l  o      ^  C  \n
00000014
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ echo "Hello ^C" | od -bc
00000000 110 145 154 154 157 040 040 040 040 136 103 012
          H  e  l  l  o      ^  C  \n
00000014
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ echo "Hello ^C" | od -cb
00000000  H  e  l  l  o      ^  C  \n
          110 145 154 154 157 040 040 040 040 136 103 012
00000014
```

► Chapitre 4

Manipulation et contenu d'un fichier

Commande cmp

cmp - compare 2 fichiers octet par octet

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cp arbo warbo  
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cmp arbo warbo  
marchal@marchal-OMEN-Laptop-15-en1xxx:~$
```

Si il y a une modification de faite dans un des deux fichiers alors cmp retourne la première différence, l'octet concerné et la ligne :

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cmp arbo warbo  
arbo warbo sont différents: octet 351, ligne 23  
marchal@marchal-OMEN-Laptop-15-en1xxx:~$
```

Pratique pour vérifier que 2 fichiers sont identiques

► Chapitre 4

Manipulation et contenu d'un fichier

Commande diff

diff- compare 2 fichiers ligne par ligne

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cp arbo warbo
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cmp arbo warbo
marchal@marchal-OMEN-Laptop-15-en1xxx:~$
```

Si il y a une modification de faite dans un des deux fichiers alors diff retourneras toutes les différences :

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ diff arbo zarbo
0a1
> 1
23c24,25
<
---
> #
> 12
```

Pratique pour vérifier si il y a différence mais également son étendue, diff peut également comparer 2 dossiers.

► Chapitre 4

Manipulation et contenu d'un fichier

Commande paste

Chapitre 4

paste

paste : permet de joindre des fichiers entre eux horizontalement ou verticalement, et de changer leur "séparateur"

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ paste -s number
1      2      3      4
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ paste -s -d "|" number
1|2|3|4
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ paste number
1
2
3
4
marchal@marchal-OMEN-Laptop-15-en1xxx:~$
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ paste number
1
2
3
4
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ paste -s number
1      2      3      4
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ paste -d "|" number state capital
1|Arunachal Pradesh|Itanagar
2|Assam|Dispur
3|Andhra Pradesh|Hyderabad
4|Bihar|Patna
|Chhattisgrah|Rajpur
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ paste -d "|" -s number state capital
1|2|3|4
Arunachal Pradesh|Assam|Andhra Pradesh|Bihar|Chhattisgrah
Itanagar|Dispur|Hyderabad|Patna|Rajpur
marchal@marchal-OMEN-Laptop-15-en1xxx:~$
```

► Chapitre 4

Manipulation et contenu d'un fichier

Commande cut

cut : cette commande ne fonctionne que avec des options, permet de filtrer le texte par délimiteur, caractère ou octet

Exemple:

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cat arbo
/
├── bin -> usr/bin
├── boot
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── snap
├── srv
├── swapfile
├── sys
├── tmp
├── usr
└── var
```

-b[1-9] (bytes) : 1-3 : 1 au 3 octets

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cut -b 1-3 arbo
```

Ex: 1 à 4 octets

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cut -b 1-4 arbo
```


-c[1-9] (caractère) : 1-3 : 1 au 3 octets

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cut -b 1-3 arbo
```

Ex: 1 à 4 octets

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cut -b 1-4 arbo
```

-f[1-9] (fields) : c'est la colonne, si ne trouve pas de séparateur, affiche la ligne en entière.

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cut -f arbo
cut: valeur de champ incorrecte : «arbo»
Saisissez « cut --help » pour plus d'informations.
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cut -f 1 arbo
```

```
/
— bin -> usr/bin
— boot
— dev
— etc
— home
— lib -> usr/lib
— lost+found
— media
— mnt
— opt
— proc
— root
— run
— sbin -> usr/sbin
— snap
— srv
— swapfile
— sys
— tmp
— usr
— var
```

Après avoir ajouté une tabulation sur un des champs dans le fichier arbo

cut du field 1 sur arbo nous retourne

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cut -f 1 arbo
/
| bin -> usr/bin
| boot
| dev
| etc
|
| lib -> usr/lib
| lost+found
| media
| mnt
| opt
| proc
| root
| run
| sbin -> usr/sbin
| snap
| srv
| swapfile
| sys
| tmp
| usr
| var
```

cut du field 2 sur arbo nous retourne

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cut -f 2 arbo
/
| bin -> usr/bin
| boot
| dev
| etc
| home
| lib -> usr/lib
| lost+found
| media
| mnt
| opt
| proc
| root
| run
| sbin -> usr/sbin
| snap
| srv
| swapfile
| sys
| tmp
| usr
| var
```

-d : le délimiteur, par default c'est tabulation, utilisons plutôt l'espace sur ce fichier

cut du field 1 sur arbo avec
en délimiteur ' ' nous retourne

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cut -f 1 -d ' ' arbo
```

```
/
bin
boot
dev
etc
home
lib
lost+found
media
mnt
opt
proc
root
run
sbin
snap
srv
swapfile
sys
tmp
usr
var
```

cut du field 2 sur arbo avec
en délimiteur ' ' nous retourne

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cut -f 2 -d ' ' arbo
```

```
/
bin
boot
dev
etc
home
lib
lost+found
media
mnt
opt
proc
root
run
sbin
snap
srv
swapfile
sys
tmp
usr
var
```

► Chapitre 4

Manipulation et contenu d'un fichier

Commande awk

Awk n'est pas juste une commande, c'est un langage.

1. AWK permet de :

- (a) Analyser un fichier ligne par ligne.
- (b) diviser chaque ligne d'entrée en champs
- (c) Comparer les lignes/champs d'entrée à un modèle.
- (d) Effectué une ou plusieurs actions sur les lignes correspondantes.

2. Utile pour :

- (a) Transformer des fichiers de données
- (b) Produire des rapports

3. Constructions de programmation :

- (a) Formater les lignes de sortie
- (b) Opérations arithmétiques et sur les chaînes de caractères
- (c) Conditionnels et boucles

Syntaxe :

awk [options] 'critere _de_selection {action}' <nom_du_fichier>

```
bosko@bosko-vm:~$ cat ~/answers.txt
```

```
a,1,1  
b,3,4  
c,5,2  
d,6,1  
e,3,3  
f,3,7
```

```
bosko@bosko-vm:~$ awk -F ',' '{if($2==$3){print $1,""$2,""$3} else {print "No Duplica  
tes"}}' answers.txt
```

```
a,1,1  
No Duplicates  
No Duplicates  
No Duplicates  
e,3,3  
No Duplicates
```



► Chapitre 4

Manipulation et contenu d'un fichier

Commande sort

Chapitre 4

sort

sort : permet de trier vos résultats, par default dans l'ordre

->[a-z]puis[A-Z]puis[1-9]

-r : pour le sense inverse -u : n'affiche pas de doublons

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cut -f 2 -d ' ' arbo | sort
```

```
/
bin
boot
dev
etc
home
lib
lost+found
media
mnt
opt
proc
root
run
sbin
snap
srv
swapfile
sys
tmp
usr
var
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cut -f 2 -d ' ' arbo | sort -r
```

```
var
usr
tmp
sys
swapfile
srv
snap
sbin
run
root
proc
opt
mnt
media
lost+found
lib
home
etc
dev
boot
bin
/
```

► Chapitre 4

Manipulation et contenu d'un fichier

Commande uniq

La commande uniq permet de trouver et de supprimer les lignes doublons qui sont adjacentes les unes entres elles.

Le texte doit être trié avant avec la commande sort.

Sort contient déjà une option qui réalise la même chose.

Uniq sera utile pour ses options

-c : affiche le nombre d'occurrence

-i : ignore la casse

man uniq

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ uniq -c music
  3 I love music.
  1
  2 I love music of Kartik.
  1
  1 Thanks.
```

► Chapitre 4

Manipulation et contenu d'un fichier

Commande tr

tr est une commande qui remplace ou supprime des caractères de l'**entrée standard (stdin)** et écrit le résultat sur la **sortie standard (stdout)**.

tr permet d'effectuer différentes transformations de texte, notamment la conversion de la casse.

tr ne peut pas lire directement de fichier

tr ne fonctionne que si vous lui indiquez des options

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ tr arbo
tr: opérande manquant après «arbo»
Deux chaînes doivent être indiquées lors de la conversion.
Saisissez « tr --help » pour plus d'informations.
```

Vous devez lui envoyer directement le contenu à traiter

Via le pipe " | " ou via une redirections " < " (métacaractères)

(ok) `tr -d tmp < arbo`

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cat arbo
```

```
/
| bin -> usr/bin
| boot
| dev
| etc
| home
| lib -> usr/lib
| lost+found
| media
| mnt
| opt
| proc
| root
| run
| sbin -> usr/sbin
| snap
| srv
| swapfile
| sys
| tmp
| usr
| var
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cat arbo | tr -d tmp
```

```
/
| bin -> usr/bin
| boo
| dev
| ec
| hoe
| lib -> usr/lib
| los+found
| edia
| n
| o
| roc
| roo
| run
| sbin -> usr/sbin
| sna
| srv
| swafile
| sys
|
| usr
| var
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ head -n 3 arbo | tr [a-z] [A-Z]
```

```
/
```

```
|— BIN -> USR/BIN
```

```
|— BOOT
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ head -n 3 arbo | tr [:lower:] [:upper:]
```

```
/
```

```
|— BIN -> USR/BIN
```

```
|— BOOT
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ head -n 3 arbo | tr [:blank:] 0
```

```
0/
```

```
|—0bin0->0usr/bin
```

```
|—0boot
```

► Chapitre 5

Redirections et PIPE

Entrées / Sorties standard

standard input device

Stdin

Stdout

Stderr

standard input device

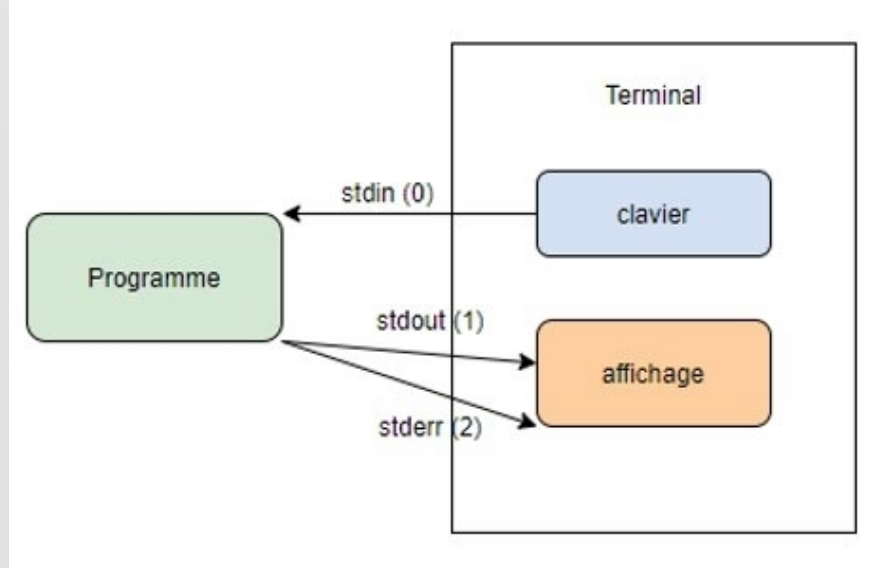
Stdin (input)(0)

Stdout (output)(1)

Stderr (error)(2)

Notions de stdin, stdout, stderr :

- **stdin** -> votre clavier
- **stdout** est la sortie standard du programme, sans erreur
- **stderr** est la sortie d'erreur du programme



► Chapitre 5

Redirections et PIPE

Redirection I/O

Input
Output

Les redirections permettent de rediriger (ce qui est censé apparaître à l'écran vers une destination) :

- `[cmd] > [fichier]` : permet de remplacer le contenu du fichier par le résultat de `stdin`.
- `[cmd] >> [fichier]` : permet d'ajouter à la fin du contenu du fichier le résultat de `stdin`.
- `[cmd] 2> [fichier]` : permet de remplacer le contenu du fichier par le résultat de `stderr`.
- `[cmd] 2>> [fichier]` : permet d'ajouter à la fin du contenu du fichier le résultat de `stderr`.

Syntaxe stdout (1)

```
[CMD] > [FICHER]
```

```
[CMD] 1> [FICHER]
```

=> STDOUT

=> Ecrase le fichier

```
[CMD] >> [FICHER]
```

```
[CMD] 1>> [FICHER]
```

=> STDOUT

=> Ecrire à la fin du fichier

Syntaxe stderr (2)

```
[CMD] &> [FICHER]
```

```
[CMD] 2> [FICHER]
```

=> STDERR

=> Ecrase le fichier

```
[CMD] &>> [FICHER]
```

```
[CMD] 2>> [FICHER]
```

=> STDERR

=> Ecrire à la fin

Chapitre 5

Redirection I/O

Syntaxe stdin (0)

```
x:~$ [cmd] < nom_du_fichier
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cat <0 arbo
bash: 0: Aucun fichier ou dossier de ce type
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cat 0< arbo
/
| bin -> usr/bin
| boot
| dev
| etc
| home
| lib -> usr/lib
| lost+found
| media
| mnt
| opt
| proc
| root
| run
| sbin -> usr/sbin
| snap
| srv
| swapfile
| sys
| tmp
| usr
| var
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cat 1< arbo
^C
```

Chapitre 5

Redirection I/O

Exemple :

```
$ echo "bonjour" > titi # redirige le message dans un fichier titi en le créant  
$ echo "belle journée ?" >> titi # ajoute le message à la fin de titi
```

```
jeanne@PC-M2i:~/documents$ cat titi  
bonjour  
belle journée ?
```

```
$ jiojio &> titi # écrase titi et écris à la place l'erreur de la commande
```

```
jeanne@PC-M2i:~/documents$ cat titi  
jiojio: command not found
```

Chapitre 5

Redirection I/O

Rediriger **stdout** ET **stderr** dans le même fichier :

```
~$ less > fichierAvecOuSansErreur 2>&1
```

```
~$ less >> fichierAvecOuSansErreur 2>&1
```

Rediriger **stdout** dans un fichier « success » **stderr** dans un fichier « error » :

```
~$ less >> fichierSansErreur 2> fichierAvecErreur
```

```
~$ less > fichierSansErreur 2> fichierAvecErreur
```

► Chapitre 5

Redirections et Pipe

"|" symbole
Le pipe

Le pipe fonctionne à peu près comme un fichier FIFO,
" | " est un **symbole**.

Il permet de lancer des commandes en parallèles et de récupérer le résultat de celui qui le précède selon **First In -> First Out (I/O)**.

La différence entre un pipe et une redirection est que, alors qu'une pipe passe **stdout** comme **stdin** à une autre commande.

Une redirection envoie **stdout**, **stderr** vers un fichier
ou
Un fichier vers **stdin**.

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cat arbo > grep bin
cat: bin: Aucun fichier ou dossier de ce type
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cat arbo | grep bin
| bin -> usr/bin
| sbin -> usr/sbin
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ cat arbo > tempArbo && grep "bin" < tempArbo
| bin -> usr/bin
| sbin -> usr/sbin
```

► Chapitre 6

Les permissions

Gestions des utilisateurs et des droits

Quelques notions :

- Utilisateur : Utilisateur connecté au système.

La liste des utilisateurs est disponible dans le fichier `/etc/passwd`

- Groupe : Groupe appartenant au système.

La liste des groupes est disponible dans le fichier `/etc/group`

Tous les fichiers à un propriétaire (p)

Tous les fichiers à un groupe propriétaire (g)

En général c'est root qui est attribué, à l'exception des répertoires personnelles des utilisateurs et des répertoires de services.

Les autres sont symbolisé par (o) = other

Rappel: root étant root, il outrepassa ces règles le concernant. Root a accès à partout

► Chapitre 6

Les permissions

quelques commandes (chmod, umask, ...).

L'accès aux dossier, fichier, binaires, devices etc...

Est autorisé ou non en fonction de 3 droits:

r : read (lecture)

w : write (écriture)

x : execute (éxecuter)

- : symbolisant l'absence de la permission

Par rapport à 3 choses

Le propriétaire, le groupe propriétaire, les autres

| | | | | | | | | |
|------------|---|--------|--------|------|-----|----|-------|----------|
| drwxr-xr-x | 1 | jeanne | jeanne | 4096 | Mar | 25 | 10:12 | dossier |
| -rw-r--r-- | 1 | jeanne | jeanne | 0 | Mar | 24 | 18:19 | fichier1 |

groupe:user

Chapitre 6

chmod

Chaque clé peut avoir 2 valeurs (c'est binaire):

r ou -

w ou -

X ou -

$$2^3 = 8$$

par défaut:

(0644) pour les fichiers réguliers

(0755) pour les dossiers

| Code binaire | valeur |
|--------------|--------|
| 000 | - - - |
| 001 | - - x |
| 010 | - w - |
| 011 | - w x |
| 100 | r - - |
| 101 | r - x |
| 110 | r w - |
| 111 | r w x |

```
drwxr-xr-x 1 jeanne jeanne 4096 Mar 25 10:12 dossier
-rw-r--r-- 1 jeanne jeanne   0 Mar 24 18:19 fichier1
```

- Commandes

Droits admins obligatoire, sudo depuis un utilisateur non root et présent dans le fichier sodoers.

Sudo n'est pas présent de base sur toutes les distribs, "su -" pour se connecter en tant que root, puis executer la commande.

- R : récursif (sur tous les sous dossiers), `man -a chmod`

```
~$ sudo chmod -R 770 test2/  
~$
```

Attention vous n'êtes pas averti en root.

▶ Chapitre 6

Les permissions

quelques commandes (chmod, umask, ...).

umask à contrario, fonctionne de façon inverse, il n'ajoute pas des droits.

Il en enlève.

Par définition sur un système GNU/Linux, les fichiers sont créés avec les droits 666 (rw-rw-rw-) qui donnent les droits de lecture et d'écriture à tout le monde. Les répertoires sont eux créés avec les droits 777 (rwxrwxrwx) qui donnent tous les droits à tout le monde.

La valeur UMASK par défaut est de 022. Ce qui veut dire qu'au final un fichier sera créé avec les droits 644 (rw- r- r-)

Un répertoire quant à lui sera créé avec les droits 755 (rwx-r-x-r-x).

Il est possible de modifier les valeurs UMASK par défaut soit de manière temporaire en exécutant simplement :

```
umask 077
```

Ou en allant modifier directement la valeur du système dans le fichier
/etc/login.defs

| umask Value Octal (xyz) | Default File Permissions | 666 - xyz | Default Directory Permissions | 777 - xyz |
|----------------------------|-----------------------------|-----------|----------------------------------|-----------|
| 000 | rw-rw-rw | 666 | rwXrwXrwX | 777 |
| 002 | rw-rw-r-- | 664 | rwXrwXr-X | 775 |
| 022 | rw-r--r-- | 644 | rwXr-Xr-X | 755 |
| 026 | rw-r----- | 640 | rwXr-X--X | 751 |
| 046 | rw--W---- | 620 | rwX-WX--X | 731 |
| 062 | rw----r-- | 604 | rwX--Xr-X | 715 |
| 066 | rw----- | 600 | rwX--X--X | 711 |
| 222 | r--r--r-- | 444 | r-Xr-Xr-X | 555 |
| 600 | ---rw-rw- | 066 | --XrwXrwX | 177 |
| 666 | ----- | 000 | --X--X--X | 111 |
| 777 | ----- | 000 | ----- | 000 |

▶ Chapitre 6

Les permissions

setuid, setgid, sticky bit

SetUID, SetGID, Sticky bit

En plus des droits normaux, ils existent sur Linux/Unix et leurs dérivés 3 droits spéciaux:

SetSUID (Set User Id) est une permission spéciale attribuée à un fichier.

Elle permet au fichier en cours d'exécution d'être exécuté avec les privilèges de son propriétaire.

Par exemple, si un fichier appartient à l'utilisateur root et que le bit setuid est activé, peu importe qui exécute le fichier, il sera toujours exécuté avec les privilèges de l'utilisateur root.

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ ls -al /usr/bin/passwd  
-rwsr-xr-x 1 root root 68208 mars 14 2022 /usr/bin/passwd
```

SetUID, Set**GID**, Sticky bit

Lorsque le bit Set Group ID est activé (2) sur un fichier

Le fichier est exécuté avec l'autorité du groupe lié.

Par exemple, si un fichier appartient au groupe des utilisateurs, peu importe qui exécute ce fichier, il sera toujours exécuté avec l'autorité du groupe de l'utilisateur.

SetUID, Set**GID**, Sticky bit

Lorsque le bit Set Group ID est activé (2) sur un dossier

Tous les dossiers enfants hériteront du groupe propriétaire

SetUID, SetGID, **Sticky bit**

Sticky bit (1) sur un dossier

Lorsque le sticky bit est activé sur un répertoire, seuls l'utilisateur root, le propriétaire du répertoire et le propriétaire d'un fichier peuvent supprimer des fichiers dans ce répertoire.

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ ls -al /tmp | grep .fon*  
drwxrwxrwt 2 root  root  4096 oct. 23 06:55 .font-unix
```


► Chapitre 7

La commande find

Critères de recherche

Chapitre 7

Critères de recherche

Effectuer une recherche avec find :

Syntaxe : `find [chemin] -name [expression]`

La commande `find` permet de rechercher des fichiers dans une hiérarchie de répertoires.

Options utiles :

`-name` : précise le nom du fichier, sensible à la casse

`-iname` : comme `-name` mais non sensible à la casse

`-type d/f/l` : retourne uniquement les fichiers de type `d`=directory, `f`=fichier ou `l`=lien symbolique

`-atime -1/+10` : recherche par date d'accès. Exemple « `-atime -1` » accès depuis moins de 1 jour ou « `-atime +10` » accès qui date de plus de 10 jours.

`-size +10M` : fichier supérieur à 10 Mo

`-perm /mode` : permissions à rechercher, par exemple « `-perm /4750` » ou `/4000` affiche fichier avec permission `setGUI`

Essayé :
`sudo find / -perm /4000`

```
jeanne@PC-M2i:~$ find . -name fichier*  
./documents/fichier1
```

► Chapitre 7

La commande find

Utilisation avancé

Exemple :

Recherche depuis la racine " / ", ne dépasse pas 2 de profondeur (dans l'arborescence).

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ find / -maxdepth 2 -name "*.conf" | head -n 3 | sort -r
find: '/root': Permission non accordée
/etc/usb_modeswitch.conf
find: /etc/ltrace.conf
'/lost+found'/etc/appstream.conf
: Permission non accordée
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ find / -maxdepth 2 -name "*.conf" | head -n 3 | sort -r
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ find /etc -maxdepth 1 -type f -name "*.conf"
/etc/appstream.conf
/etc/ltrace.conf
/etc/usb_modeswitch.conf
/etc/sysctl.conf
/etc/mttools.conf
```

```
/home/marchal/.config/pgadmin4/Default/shared_proto_db/000003.log
/home/marchal/.config/pgadmin4/Default/shared_proto_db/metadata/000003.log
/home/marchal/.pgadmin/pgadmin4.log
/home/marchal/.npm/_logs/2022-08-12T11_06_48_339Z-debug.log
marchal@marchal-OMEN-Laptop-15-en1xxx:~$ find /var /home -name *.log -type f
```

Rechercher dans /etc sous 1 de profondeur les fichiers finissant par .conf copie ensuite ce fichier en .conf.bkp

```
sudo find /etc -maxdepth 1 -name "*.conf" -exec cp {} {}.bkp \;
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:/etc$ sudo find /etc -maxdepth 1 -name "*.conf" -exec cp {} {}.bkp \;
```

```
marchal@marchal-OMEN-Laptop-15-en1xxx:/etc$ ls -al /etc | grep .bkp
```

```
-rw-r--r-- 1 root root 3028 oct. 23 11:01 adduser.conf.bkp
-rw-r--r-- 1 root root 433 oct. 23 11:01 app.conf.bkp
-rw-r--r-- 1 root root 769 oct. 23 11:01 appstream.conf.bkp
-rw-r--r-- 1 root root 26916 oct. 23 11:01 brltty.conf.bkp
-rw-r--r-- 1 root root 6821 oct. 23 11:01 ca-certificates.conf.bkp
-rw-r--r-- 1 root root 2969 oct. 23 11:01 debconf.conf.bkp
-rw-r--r-- 1 root root 604 oct. 23 11:01 deluser.conf.bkp
-rw-r--r-- 1 root root 685 oct. 23 11:01 e2scrub.conf.bkp
-rw-r--r-- 1 root root 20 oct. 23 11:01 fprintd.conf.bkp
-rw-r--r-- 1 root root 280 oct. 23 11:01 fuse.conf.bkp
-rw-r--r-- 1 root root 2584 oct. 23 11:01 gai.conf.bkp
-rw-r--r-- 1 root root 5060 oct. 23 11:01 hdparm.conf.bkp
-rw-r--r-- 1 root root 92 oct. 23 11:01 host.conf.bkp
-rw-r--r-- 1 root root 206 oct. 23 11:01 idmappd.conf.bkp
-rw-r--r-- 1 root root 110 oct. 23 11:01 kernel.img.conf.bkp
-rw-r--r-- 1 root root 1308 oct. 23 11:01 kerneloops.conf.bkp
-rw-r--r-- 1 root root 34 oct. 23 11:01 ld.so.conf.bkp
-rw-r--r-- 1 root root 27 oct. 23 11:01 libao.conf.bkp
-rw-r--r-- 1 root root 191 oct. 23 11:01 libaudit.conf.bkp
-rw-r--r-- 1 root root 371 oct. 23 11:01 libguestfs-tools.conf.bkp
-rw-r--r-- 1 root root 533 oct. 23 11:01 logrotate.conf.bkp
-rw-r--r-- 1 root root 14867 oct. 23 11:01 ltrace.conf.bkp
-rw-r--r-- 1 root root 808 oct. 23 11:01 mke2fs.conf.bkp
-rw-r--r-- 1 root root 624 oct. 23 11:01 mtools.conf.bkp
-rw-r--r-- 1 root root 542 oct. 23 11:01 nsswitch.conf.bkp
-rw-r--r-- 1 root root 552 oct. 23 11:01 pam.conf.bkp
-rw-r--r-- 1 root root 7649 oct. 23 11:01 pnm2ppa.conf.bkp
-rw-r--r-- 1 root root 350 oct. 23 11:01 popularity-contest.conf.bkp
-rw-r--r-- 1 root root 1889 oct. 23 11:01 request-key.conf.bkp
-rw-r--r-- 1 root root 717 oct. 23 11:01 resolv.conf.bkp
-rw-r--r-- 1 root root 1382 oct. 23 11:01 rsyslog.conf.bkp
-rw-r--r-- 1 root root 5211 oct. 23 11:01 rygel.conf.bkp
-rw-r--r-- 1 root root 10593 oct. 23 11:01 sensors3.conf.bkp
```

▶ Chapitre 8

Vi

Vi ou Vim (version plus récente) est un éditeur de texte, conçu pour le développement + d'une 20aine d'année.

Il est conçu pour être très ergonomique, une fois pris en main.

Vi est en général présent sur la majorité des serveurs / poste de travail linux, ce qui en fait un outil que vous devez connaître, en tous cas quelques règles.

Outils très puissants si maîtrisé

Vi à été conçu pour limiter la frappe nécessaire.

limiter les mouvements des doigts et des mains sur le clavier.

Faciliter les moyens mnémotechniques pour retenir les raccourcis, commandes

En premier lieu, vi est un éditeur de texte, c'est un programme.
Il peut se lancer soit en graphique (si version graphique installé), sinon directement avec la commande **vi**

man -a vi

Une option pratique qui fonctionne aussi avec d'autre éditeur est:
Vi +numéro_de_ligne <nom_du_fichier>

Vi +21 fichier_de_test

```
18 /snap/core/13886/bin/bash
19 /snap/core/13886/etc/apparmor.d/abstractions/bash
20 /snap/core/13886/usr/share/doc/bash
21 /snap/core/13886/usr/share/menu/bash
22 /snap/core/13741/bin/bash
23 /snap/core/13741/etc/apparmor.d/abstractions/bash
24 /snap/core/13741/usr/share/doc/bash
```


Vi à plusieurs modes, vous ne pouvez pas vi <un_fichier> et directement écrire dedans.

Il y a le mode normal accessible via **échap** pour déplacer le curseur et le mode **insertion** pour écrire là où est votre curseur

<https://linux-note.com/vim-raccourcis-clavier/>

► Chapitre 9

Sauvegarde et Restauration

Principe

Une sauvegarde d'un fichier, dossier, partitions ou d'un disque.
Permet d'assurer une redondance des données.