Université Abdelhamid Mehri – Constantine 2

Systèmes d'Exploitation 1

– TP –

Chapitre II : Système de fichiers

Staff pédagogique			
Nom	**Grade**	**Faculté/Institut**	**Adresse e-mail**
Mohamed SANDELI	**MCB**	**Nouvelles Technologies**	**mohamed.sandeli@univ-constantine2.dz**

Etudiants concernés			
Faculté/Institut	**Département**	**Année**	**Spécialité**
Nouvelles Technologies	MI	Licence 2	Tronc commun

Objectifs du TP 2

A l'issue de ce TP, vous serez en mesure de :

- Vous déplacer dans l'arborescence des systèmes de fichiers Linux, de manipuler des fichiers et des répertoires.
- Vous archiver et compresser les fichiers et les répertoires de linux.

1. Introduction

Un système de fichiers est une façon d'organiser et de stocker une arborescence sur un support (disque, disquette, cd ...). Chaque OS propriétaire a développé sa propre organisation.(voir Figure 2.1)

« Ext, Ext2, Ext3, Ext4, JFS, XFS, btrfs et swap »

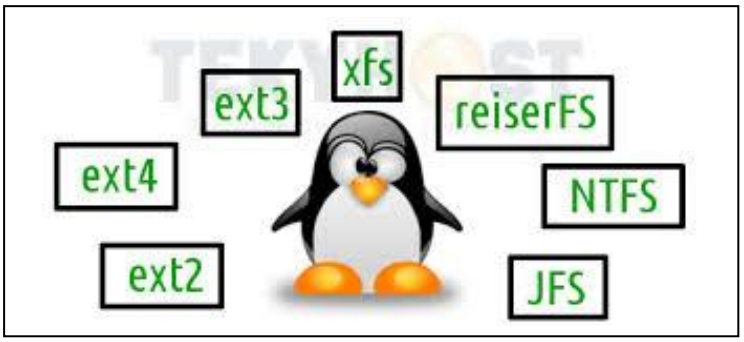


Figure 2.1 : Systeme de fichiers

2. Types de Fichiers

Dans LINUX, tous les objets de l'arborescence sont considérés comme étant un fichier.

Il existe différents types de fichiers:

- Les fichiers **(-)**
- Les répertoires **(d)**
- Les périphériques (Major device Number, Minor device number)
  - Les périphériques bloc **(b).** Les périphériques blocs (comme les disques durs, les lecteurs de CDROM) gère une certaine quantité d'octet à la fois, par groupe de bloc.
  - Les périphériques caractère **(c)** . Les périphériques caractères (comme la souris, le clavier, le port série, le port parallèle) gère un caractère transmis à la fois.
- Les canaux de communication FIFO des pipes **(p)**
- Les sockets **(s)**
- Les liens **(l)**

La commande **ls -l** : affiche par défaut une liste triée par ordre alphabétique. Les commandes **dir** et **vdir** sont des variantes de ls (voir Figure 2.2).

Options de la commande ls :

- -r : tri inversé
- -t : tri sur la date de modification (du plus récent au plus ancien)
- -u : tri sur la date d'accès avec l'option -lt
- -c : tri sur la date de changement de statuts des fichiers avec
- l'option -lt
- -l : présentation tabulaire détaillée
- -1 : présentation en liste continue

- -h : avec l'option -l affiche les valeurs de poids de fichier avec un multiplicateur (K, M, G, …)
- -i : affiche le numéro d'inode

On trouvera encore plus d'options et détails dans la page man: **man ls**.

```
guru99@VirtualBox:~$ ls
Desktop    Downloads         Music     Public     Videos
Documents  examples.desktop  Pictures  Templates
guru99@VirtualBox:~$
```

Figure 2.2 : éxecution de la commande ls

3. Organisation physique des fichiers

Un système de fichiers physique peut être séparé en deux parties distinctes(voir Figure 2.3)

- La table des inodes
- Les block de données.

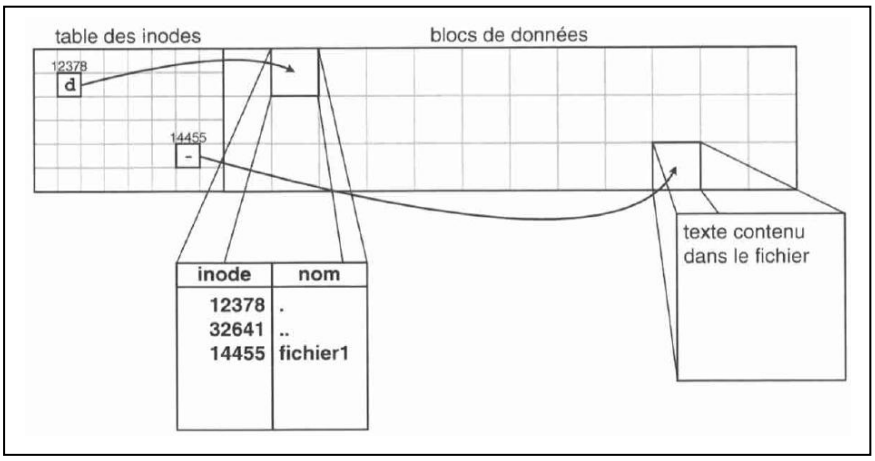


Figure 2.3 : Système de fichier physique

- **Inode (Index node):**

  Chaque fichier est représenté par une structure appelée inode. Ces inode sont regroupés dans une table et sont identifiables par un nu-méro..

  L'inode contient la totalité des information sur le fichier, sauf son nom.

  - Type de fichier : `-` (ordinaire), `d` (répertoire), `l` (lien symbolique), `b` (bloc), `c` (caractère), `p` (tube), `s` (socket).
  - Mode ou droits d'accès au fichier.
  - Nombre de liens physiques ("hard links") : correspond au nombre de références, c'est-à-dire au nombre de noms pour ce même fichier.
  - UID ou identifiant de l'utilisateur propriétaire.
  - GID ou identifiant du groupe propriétaire.
  - Taille du fichier en nombre d'octets.
  - Dates de la dernière consultation, modification du contenu et modification de l'inode du fichier.
  - Adresses pointant vers les blocs de données qui constituent ce fichier.

- **Blocs de données:**

Les données réelles des fichiers sont stockées dans les blocs de données(voir Figure 2.4).

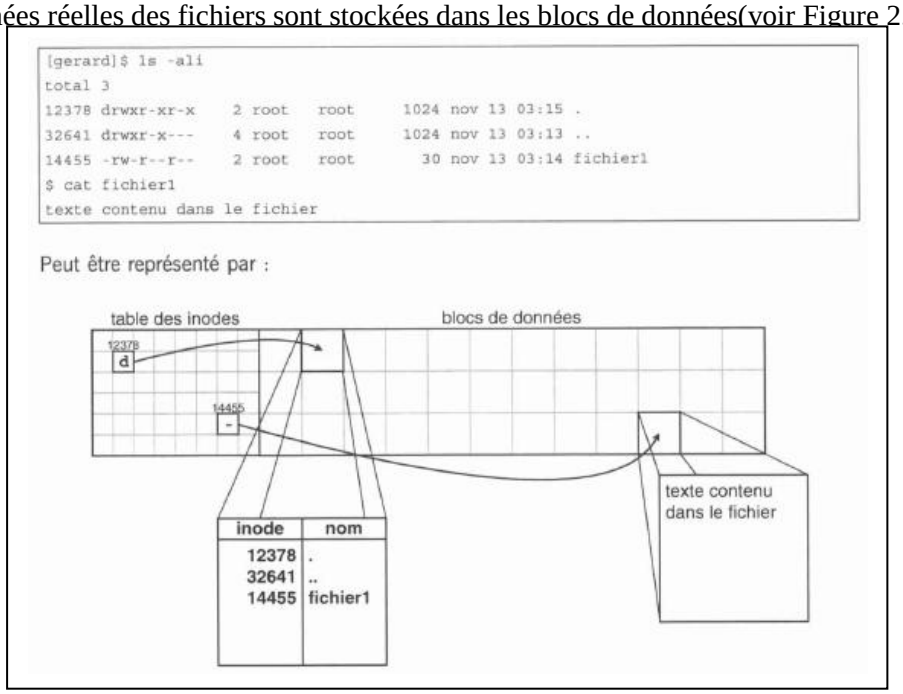


Figure 2.4 :Exemple -A- d'un Système de fichier physique

le nom d'un fichier étant stocké dans les données du répertoire et non dans l'inode de celui-ci, il est facile de donner plusieurs noms (références ou liens durs) à ce meme fichier(voir Figure 2.5).

ln [-s] <nom du fichier> <nom supplémentaire>

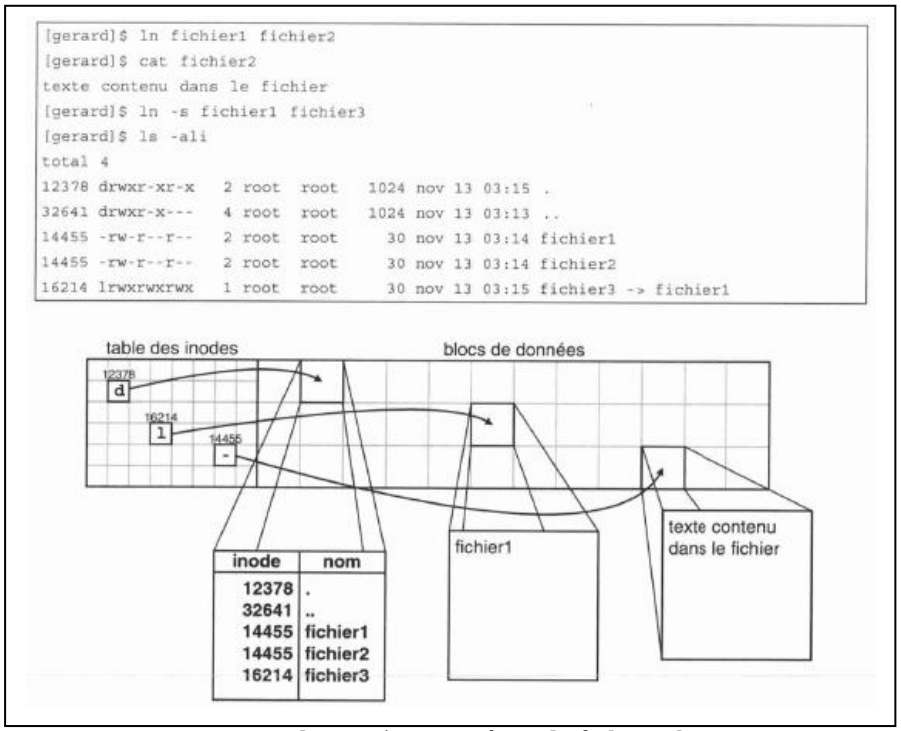


Figure 2.5 :Exemple -B- d'un Système de fichier physique

4. Organisation du Système de Fichiers

Dans les systèmes Linux, **tout est fichier**, donc, contrairement aux systèmes Windows un fichier Linux peut être :

- Un fichier
- un périphérique
- une partition
- un programme en cours d'exécution
- un répertoire …
- Pour structurer ces fichier, Linux ne dispose pas de d'unités **C:** , **D:** , **E:** …(voir Figure 2.6).

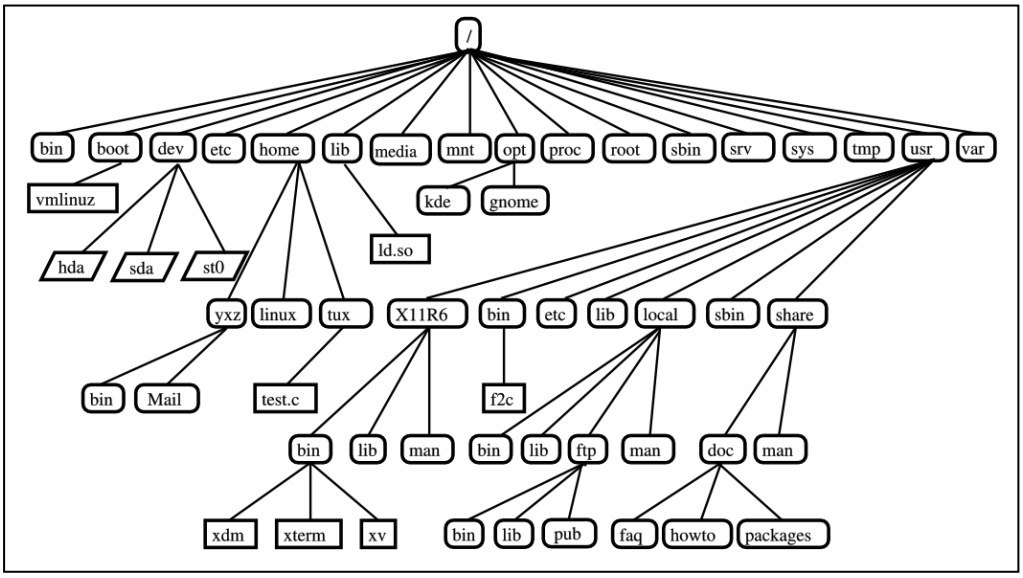


Figure 2.6 : Système de fichier

Parmi ces répertoires, plusieurs sont importants, on peut en citer :

- /bin/ : Contient toutes les commandes de base nécessaires au démarrage et à l'utilisation d'un système minimaliste (par exemple : cat, ls, cp, sh).
- /sbin : Contient les commandes systèmes réservées aux administrateurs.
- /boot : Contient les fichiers nécessaires au démarrage du système d'exploitation.
- /dev : Contient des fichiers correspondants à un périphériques (disques , disquettes …).
- /etc : Contient la plupart des fichiers de configuration du système.
- /home/ : Utilisé pour stocker les répertoires utilisateurs (exemple : /home/user1).
- /opt : Utilisé comme emplacement d'installation d'un logiciel utilisé.
- /tmp/ : Utilisé pour stocker les fichiers ***temp****oraires* tout comme */var/tmp* et */run/tmp e*t généralement vidé à chaque démarrage.

5. Déplacement dans le Système de Fichiers

Chemin absolu : liste des noms de répertoires, jusqu'au fichier concerné, depuis la racine, séparés par des /

Exemples : **/etc/passwd**

/home/ahmed/bureau

Un chemin absolu commence toujours par /

Chemin relatif (au répertoire courant) : On utilise **..** pour représenter le répertoire père (et . représente le répertoire courant).

Exemples : on est dans **/home/ahmed/bureau**

- le fichier **fic1** dans **TP** : **TP/fic1** (équivalent à **./TP/fic1**)

- le fichier **fic2** de **ahmed**: **../ahmed/fic2**

Un chemin relatif ne commence jamais par /

Répertoire personnel :

- Chaque utilisateur (sauf **root**) dispose d'un répertoire personnel à son nom situé dans **/home**. Par exemple, le répertoire personnel de l'utilisateur ahmed est **/home/ahmed**.
- Le chemin absolu du répertoire personnel peut s'écrire de manière abrégée avec le caractère ~ (*tilde*). Par exemple, le chemin **~/music/**pour l'utilisateur ahmed correspond au chemin **absolu /home/ahmed/music/**.

Commandes à connaître

- **pwd** (*print working directory*) affiche le chemin du répertoire courant.
- **ls** (*list*) affiche le contenu du répertoire courant.
- **cd** (*change directory*) permet de se déplacer dans le système de fichiers en changeant de répertoire courant.
    - **cd monrep** :fait du répertoire monrep le répertoire courant.
    - **cd ..** : permet de remonter d'un niveau dans l'arborescence.
    - **cd /** : permet de revenir à la racine de l'arborescence.
    - **cd** : permet de revenir à la racine du répertoire personnel.

6. Montage d'un système de fichiers

Pour être utilisée (en lecture et écriture), une unité de stockage (partition de disque dur, clé USB, cartes, CD-ROM, unités distantes, etc …) doit être accessible au système d'exploitation, c'est-à-dire avoir un chemin d'accès dans l'arborescence du système.

L'action qui consiste à rendre une unité de stockage accessible s'appelle le montage. Elle est réalisée par la commande "**mount**". Le montage utilise un répertoire déjà existant et y crée un point de montage.

L'opération inverse, le démontage, supprime le point de montage, ce qui rend inaccessible l'unité de stockage / partition et rend de nouveau accessible le contenu du répertoire que le montage avait masqué. Cette opération est effectuée par la commande "**umount**".

Les étapes :

- **sudo fdisk –l** (pour trouver le périphérique USB)
- **sudo mkdir /media/usb-key** (création du dossier «usb-key»)
- **sudo mount /dev/sdg1 /media/usb-key** (effectuer le montage)
- **ls –lah /media/usb-key** (lister le contenu du lecteur)
- **sudo umount /media/usb-key** (quand vous avez fini d'utiliser)

7. Opérations sur les fichiers

- **mkdir** (*make directory*) crée un nouveau répertoire.

  mkdir monrep (crée le répertoire monrep dans le répertoire courant)

- **touch** : crée un nouveau fichier (vide) ou met à jour la date de modification d'un fichier existant.

  touch fic1.txt (crée un fichier vide fic1.txt dans le répertoire courant)

- **cp** (*copy*) : copie des fichiers ou des répertoires.

  cp fic1.txt monrep/ (copie le fichier fic1.txt dans le répertoire monrep)

  cp fic1.txt fic2.txt (duplique le fichier fic1.txt sous le nom fic2.txt)

- **mv** (*move*) : déplace ou renomme des fichiers ou des répertoires.

  mv fic1.txt monrep/ (déplace le fichier fic1.txt dans le répertoire monrep)

  mv fic1.txt fic2.txt (renomme le fichier fic1.txt en fic2.txt)

- **rm** (*remove*) : supprime des fichiers.
- **rm -r** : supprime des répertoires.

  rm fic1.txt (supprime le fichier fic1.txt)

  rm -r monrep (supprime le répertoire monrep ainsi que tout son contenu)

- **echo** : affiche un texte.

  echo Bonjour (affiche le texte "Bonjour")

- **cat** : permet (entre autres) d'afficher le contenu d'un fichier.

  cat monfic (affiche le contenu du fichier monfic)

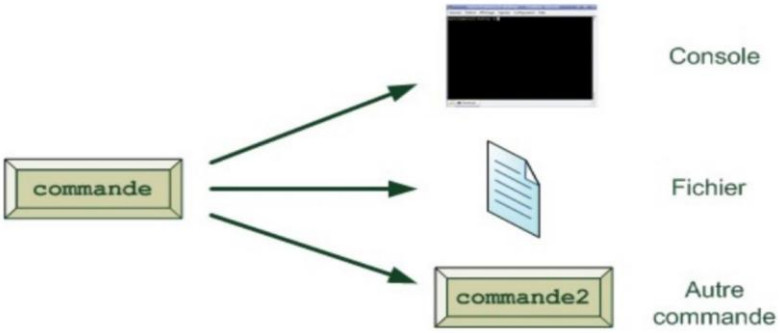
- **more** : affiche le contenu d'un fichier écran par écran avec la possibilité de défilement.
- **less** : même comportement que more mais avec la possibilité de défilement en arrière.
- **tac** : c'est l'inverse de cat affiche un fichier en commençant par la fin
- **od** : (octal dump) affiche le contenu d'un fichier en octal ou sous d'autres format suivant les options (-x en hexadécimal).
- **Strings** : affiche les chaines de caractères lisibles contenues dans les fichiers binaires.
- **head** : par défaut affiche les 10 premiers lignes.
- **tail** : par défaut affiche les 10 derniers lignes.
- **file** : donne le type de fichier.
- **Sort:** trier des données, elle permet de trier le contenu d'un fichier.
- **wc:** compter le nombre de lignes et compter le nombre de caractères.

8. Flux, pipe et redirection

C'est simplement une entré et une sortie d'un programme sous forme de texte.

Il existe trois types de flux :

1. entrée standard (stdin)
2. sortie standard (stdout)
3. erreur standard (stderr)



> : redirige dans un fichier et l'écrase s'il existe déjà.

sort etudiants.txt > resultat.txt

>> : redirige à la fin d'un fichier et le crée s'il n'existe pas.

sort etudiants.txt >> resultat.txt

2>, 2>> et 2>&1 : rediriger les erreurs

Rediriger les erreurs dans un fichier à part

sort fichier_inexistant.txt > resultat.txt 2> erreurs.log

> ***resultat.txt*** *:* contient le résultat de la commande (sauf les erreurs).

2>erreurs.log: contient les éventuelles erreurs.(Notez qu'il est aussi possible d'utiliser **2>>**)

Fusionner les sorties : il est possible de fusionner les sorties dans un seul et même fichier.

sort fichier_inexistant.txt > resultat.txt 2>&1

< et << : lire depuis un fichier ou le clavier

< : lire depuis un fichier

cat < notes.csv

<< : lire depuis le clavier progressivement

sort -n << FIN

wc -m << FIN

sort -n << FIN > nombres_tries.txt 2>&1

| : chaîner les commandes

le pipe | permet connecter la sortie d'une commande à l'entrée d'une autre commande.

Exemples:

- ls | grep toto
- ls -l | less

- du | sort –nr
- ls | wc –l

9. Recherche de fichiers

find : Rechercher des fichiers sur le système en fonction de paramètres spécifiques

- Cherchez un fichier par son nom de fichier:

  find -iname "nomdefichier"

- Faites démarrer la recherche dans le dossier « root »:
  find / -iname "nomdefichier"
- Utilisez le caractère de remplacement .* pour chercher quelque chose qui correspond à ce morceau de requête:

  find /home/pat -iname "*.conf"

- Cherchez des types spécifiques de résultats:

  find / -type f -iname "nomdefichier"

- Filtrez vos résultats de recherche par taille:

  find / -size +50M -iname "nomdefichier"

- Utilisez des opérateurs booléens pour combiner des filtres de recherche.
  find /photosdevacances -type f -size +200k -not -iname "*2015*“

locate : rechercher un fichier dans un index des fichiers du système

1. Installez la fonctionnalité : La commande locate fonctionne généralement beaucoup plus rapidement que find, car elle fonctionne à l'aide d'une base de données issue de la structure de vos fichiers. Toutes les distributions Linux ne viennent pas d'emblée avec la fonctionnalité locate installée dessus, donc tapez les commandes suivantes pour la faire installer :
   - Tapez **sudo apt-get update** et appuyez sur Entrée.
   - Tapez **sudo apt-get install mlocate** et appuyez sur Entrée.
2. Mettez à jour votre base de données .locate. La commande locate ne pourra rien trouver tant que sa base de données n'aura pas été construite et mise à jour. Cela se fait automatiquement tous les jours, mais vous pouvez lancer une mise à jour manuellement également. Vous devrez faire cela si vous voulez commencer à utiliser locate immédiatement.
   - Tapez **sudo updatedb** et appuyez sur Entrée.
3. Utilisez **locate** pour faire des recherches simples. La commande locateest rapide, mais elle ne dispose pas d'autant d'options que la commande find. Vous pouvez lacer une recherche de fichier basique plus ou moins de la même manière qu'avec la commande find.
   - **locate -i "*.jpg"**
   - Limitez vos résultats de recherche.
     locate -n 20 -i "*.jpg"

grep : rechercher un fichier à partir de son contenu

- Utilisez la commande **grep** pour chercher des suites de caractères au sein de fichiers.
  grep -r -i "requête" /chemin/vers/répertoire/
- Enlevez le texte supplémentaire.
  grep -r -i "requête" /chemin/vers/répertoire/ | cut -d: -f1
- Recherche sans tenir compte de la casse :
  grep -i "toto" *
- Les options suivantes sont utilisées mais sont facultatives :
  - L'option **n** permet d'afficher le numéro de ligne ou apparait le texte recherché ;
  - L'option **r** permet d'effectuer cette recherche récursivement ;

  grep -rn "It works" /var/www/

which : La commande which affiche le chemin complet des commandes (shell).

Description

La commande which peut prendre un ou plusieurs arguments. Pour chacun d'entre eux elle affiche sur la sortie standard (stdout) le chemin complet des exécutables qui auraient pu être saisis directement depuis le prompt Shell. La commande which recherche l'exécutable dans les différents répertoires listés dans la variable d'environnement PATH (et utilise le même algorithme que BASH).

10. Archivage et compression

Compression : Réduire la taille d'un fichier par algorithme de compression.

Archivage : Placer un ensemble de fichiers et/ou de dossiers dans un seul fichier.

- Compression sans archivage : gzip/gunzip, bzip2/bunzip2
- Archivage avec ou sans compression : tar, star

Compression gzip/gunzip

- L'utilitaire gzip est basé sur l'algorithme Deflate (combinaison des algorithmes LZ77 et Huffman). C'est la méthode de compression la plus populaire sous GNU/Linux.
- Compresser un fichier (le fichier est remplacé par son format compressé) :
  gzip mon_fichier
- Décompresser un fichier gzippé :
  gunzip mon_fichier_compresse.gz

  ou bien **gzip -d mon_fichier_compresse.gz**
- Compresser un fichier de façon optimisée :
  gzip -9 mon_fichier
- Compresser plusieurs fichiers en un :

gzip -c mon_fichier1 mon_fichier2 > mon_fichier_compresse.gz

Compression bzip2/bunzip2

- **bzip2** est à la fois le nom d'un algorithme de compression de données et d'un logiciel libre développé par Julian Seward entre 1996 et 2000 qui l'implémente.
- L'algorithme **bzip2** utilise la transformée de Burrows-Wheeler avec le codage de Huffman. Le taux de compression est la plupart du temps meilleur que celui de l'outil classique **gzip**.
- **bzip2** est une alternative à **gzip**, plus efficace mais moins rapide.
- Compresser un fichier :

  bzip2 mon_fichier

- Décompresser un fichier bzippé :

  bunzip2 mon_fichier_compresse.bz2

Commande tar

- Tar (« tape archiver », en français « archiveur pour bande », son rôle à l'origine) est le programme d'archivage de fichiers le plus populaire sous GNU/Linux et les systèmes Unix.
- **tar :** archivage sans compression
  - **tar cvf mon_archive.tar fichier1 fichier2**
  - **tar cvf mon_archive.tar dossier1/**

  Pour extraire une archive **tar**:
  - tar xvf mon_archive.tar
    - -c : signifie créer une archive tar ;
    - -v : signifie afficher le détail des opérations ;
    - -f : signifie assembler l'archive dans un fichier.

tar : archivage avec compression

- Tar peut archiver en utilisant des algorithmes de compression, afin d'avoir des archives moins volumineuses.
  - z : compression Gunzip
  - j : compression Bzip2

  Pour archiver et compresser un dossier avec Gunzip :
  - **tar cvzf mon_archive.tar.gz dossier1/**

  Pour extraire une archive **tar.gz**, tapez :
  - **tar xvzf mon_archive.tar.gz**
- De même pour Bzip2 :
  - **tar cvjf mon_archive.tar.bz2 dossier1/**
  - **tar xvjf mon_archive.tar.bz2**

En résume:

- Pour regrouper plusieurs fichiers et dossiers au sein d'un même fichier (appelé *archive*), on utilise le programme tar. Celui-ci ne compresse pas les fichiers par défaut, contrairement à zip.
- Il est possible de compresser une archive tar avec le programme **gzip** (très couramment utilisé) ou **bzip2** (meilleure compression mais plus lente).
- Les archives non compressées ont l'extension .tar, les archives compressées ont l'extension **.tar.gz** (pour **gzip**) ou **.tar.bz2** (pour **bzip2**).
- On utilise peu les formats de compression zip et rar sous Linux, mais il est possible de décompresser ces types de fichiers avec les programmes unzip et unrar. Ceux-ci ne sont en général pas installés par défaut.