Sujet 103
Commandes GNU & Unix

Commandes GNU & Unix

- 103.1 : Travail en ligne de commande (Val. 4)
- 103.2 : Traitement de flux de type texte avec des filtres (Val. 3)
- 103.3 : Gestion élémentaire des fichiers (Val. 4)
- 103.4 : Utilisation des flux, des tubes et des redirections (Val. 4)
- 103.5 : Création, contrôle et interruption des processus (Val. 4)
- 103.6 : Modification des priorités des processus (Val. 2)
- 103.7 : Recherche dans des fichiers texte avec les expressions rationnelles (Val. 2)
- 103.8 : Édition de fichiers texte avec vi (Val. 3)

103.3
Gestion élémentaire des fichiers
(Val. 4)

Gestion élémentaire des fichiers

- Description : Les candidats doivent être en mesure d'utiliser les commandes Linux de base pour gérer les fichiers et les répertoires.
- Termes, fichiers et utilitaires utilisés pour cet objectif :
  - cp
  - find
  - mkdir
  - mv
  - ls
  - rm
  - rmdir
  - touch
  - tar
  - cpio
  - dd
  - file
  - gzip
  - gunzip
  - bzip2
  - xz
  - Motifs de fichiers

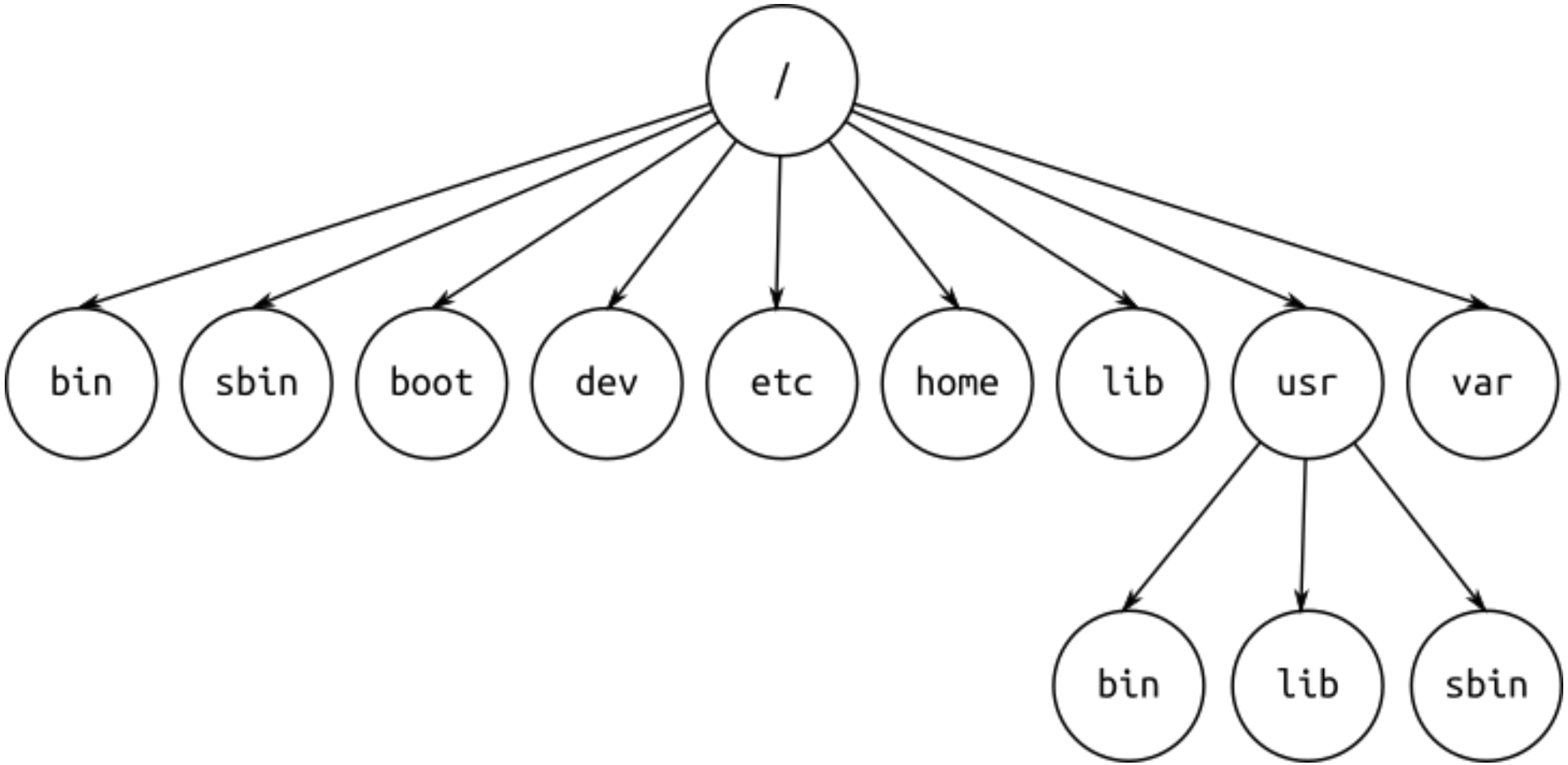
Système de fichiers

- File System ou FS, définit :
  - L'organisation des données sur un support de stockage.
  - Comment sont gérés et organisés les fichiers par le S.E.
- GNU/Linux est un S.E. entièrement orienté fichiers. Tout est représenté par des fichiers :
  - Données ;
  - Périphériques (terminaux, souris, clavier, carte son, etc.) ;
  - Moyens de communication (sockets, tubes nommés, etc.)
- Le S.F. est hiérarchique. Il décrit une arborescence de répertoires et de sous-répertoires, en partant d'un répertoire de base appelé la racine ou root directory.

Système de fichiers GNU/Linux

- L'extended file system ou ext, est le premier S.F. créé en avril 1992 spécifiquement pour GNU/Linux.
- 1993 : Version 2, ext2, évolution de ext.
- 2001 : Version 3, ext3, ajoute la journalisation à ext2. La journalisation améliore la récupération des données en cas d'arrêt brutal du système.
- 2008 : Version 4, ext4, amélioration de ext3.
- Futur : BtrFS (B-tree FS) : offre des améliorations en termes d'évolutivité, de fiabilité et de facilité de gestion.

Arborescence standard (I)



Arborescence standard (II)

- / : Répertoire racine
- /bin : Fichiers programmes exécutables binaires (commandes "essentielles").
- /sbin : commandes pour l'administration du système.
- /boot : Noyau du S.E. et les fichiers de démarrage.
- /dev : Fichiers spéciaux qui servent comme canaux de communication avec les périphériques (disques, adaptateur réseau, cartes son, etc.)
- /etc : Fichiers de configuration du S.E. et les principaux scripts de paramétrage.
- /home : Répertoires personnels des utilisateurs.
- /lib : Bibliothèques et modules du noyau.

Arborescence standard (III)

- /mnt : Points de montage des S.F. périphériques.
- /media : Points de montage des S.F. externes (CD, USB, etc.).
- /opt : Installation des applications supplémentaires.
- /root : Répertoire personnel du super-utilisateur root.
- /tmp : Stockage des fichiers temporaires.
- /usr : Programmes accessibles à tout utilisateur; sa structure reproduit celle de la racine /
- /var : Données variables liées à la machine (fichiers d'impression, traces de connexions http, smb .. dans /var/log)
- /proc : Contient une "image" du système (/proc/kcore est l'image de la RAM)

Types de fichiers

- Fichiers ordinaires ou réguliers :
  - Fichiers qui contiennent des données (texte, image, audio, etc.)
- Fichiers répertoires :
  - Un répertoire n'est rien d'autre qu'un fichier particulier contenant la liste des fichiers présents dans ce répertoire.
- Fichiers spéciaux :
  - Principalement des fichiers servant d'interface pour les divers périphériques. Se trouvent dans le répertoire /dev

Chemins (I)

- Permettent de définir et identifier un emplacement au sein du S.F.
- Chemin absolu représente l'arborescence complète à partir de la racine /
- Chemin relatif représente l'arborescence depuis une position dans l'arborescence, généralement le répertoire courant (répertoire de travail).
  - . Répertoire courant.
  - .. Répertoire parent du répertoire courant.
- Un chemin relatif peut commencer par :
  - ~ Répertoire personnel de l'utilisateur (/home/user_login)

Chemins (II)

- Exemples
  - **/root**
  - **/bin/bash**
  - **~/bin**
  - **/~**
  - **../home**
  - **~/../home**
  - **Documents/LPIC1**
  - **./Documents/LPIC1**
  - **/usr/local/../bin**

Nomenclature des fichiers

- On ne peut pas donner n'importe quel nom à un fichier.
- Longueur du nom jusqu'à 255 caractères (l'éventuelle extension comprise).
- GNU/Linux fait la distinction entre les minuscules et les majuscules.
- Quelques caractères sont à éviter car ils ont une signification particulière dans le Shell

  : & ( ) ~ <espace> \ / | ` ? - (en début de nom)

pwd

- Afficher le chemin absolu du répertoire de travail actuel (*Print Working Directory*).

pwd

cd

- Changer le répertoire de travail (*Change Directory*).

`cd chemin`

- Exemples :
  - **`cd /tmp`**
  - **`cd Documents/LPIC1`**
  - **`cd .`**
  - **`cd ..`**
  - **`cd ~`**
  - **`cd`**

ls

- Lister le contenu d'un répertoire.

`ls [options] chemin`

- Exemples :
  - **`ls`**
  - **`ls .`**
  - **`ls ~`**
  - **`ls Documents/LPIC1`**
  - **`ls /`**

ls – options

Option	Signification
`-l`	Afficher les informations détaillées pour chaque fichier et répertoire.
`-a`	Afficher les fichiers cachés (ils commencent par un point).
`-h`	Afficher les tailles dans un format plus lisible (42K, 74M, ...).
`-d`	Sur un répertoire, afficher le répertoire lui-même et non pas son contenu.
`-R`	Afficher le contenue du répertoire de manière récursive.
`-t`	La sortie est triée par date de modification du plus récent au plus ancien.
`-r`	L'ordre de sortie est inversé.
`-i`	Afficher l'**inode** du fichier.

ls – Affichage détaillé

```
- rw-r--r-- 1 kerkeni lpic1 465 déc.9 13:42 hello.c
1|    2    |3|   4   |  5  | 6 |     7     |   8
```

1. Type (**-** : ordinaire, **d** : répertoire, **l** : lien symbolique, etc.)
2. Droits d'accès pour le propriétaire, le groupe et les autres.
3. Nombre de liens.
4. Propriétaire, généralement celui qui l'a créé.
5. Groupe au quel appartient le fichier.
6. Taille en octets.
7. Date de la dernière modification.
8. Nom.

file

- Déterminer le type d'un fichier

  `file chemin`

- Exemples :
  - **`file /tmp`**
  - **`file /etc/passwd`**
  - **`file /dev/cdrom`**
  - **`file /dev/sda1`**
  - **`file /dev/tty1`**
  - **`file /bin/bash`**

touch

- Créer un fichier vide.
- Changer la date de la dernière modification.

`touch chemin`

- Exemples :
  - **`touch ./testFile.txt`** **`# Création`**
  - **`touch ./testFile.txt`** **`# Modification date`**

mkdir

- Créer un répertoire (*make directory*).
- Permet de créer plusieurs répertoires ou une arborescence.

`mkdir [-p] chemin[s]`

- Exemples :
  - **`mkdir rep1`**
  - **`mkdir /tmp/rep1 /tmp/rep2 rep3`**
  - **`mkdir rep2/rep3`**
  - **`mkdir -p rep2/rep3`**

rmdir

- Supprimer un répertoire (*remove directory*). Le répertoire doit être vide.

`rmdir chemin[s]`

- Exemples :
  - **`rmdir rep1`**
  - **`rmdir /tmp/rep1 /tmp/rep2 rep3`**
  - **`rmdir rep2`**
  - **`rmdir rep2/rep3`**

rm

- Supprimer un fichier ou une arborescence (*remove*).

`rm [options] chemin[s]`

- Options :
  - **`i`** : demander une confirmation avant toute suppression.
  - **`r`** : supprimer une arborescence (récursivement). Le chemin est généralement un répertoire.
  - **`f`** : forcer la suppression.
- Exemples :
  - **`rm file1.txt file2.txt`**
  - **`rm -r /tmp/rep1`**
  - **`rm -rf rep1  # ATTENTION : Combinaison dangereuse`**

cp (I)

- Copier un fichier vers un autre fichier ou vers un répertoire (*copy*).
- Permet de copier plusieurs fichiers ou une arborescence.

```
cp [options] chemin_source chemin_destination
```

- Options :
  - **i** : demander une confirmation avant toute copie.
  - **r** : copier une arborescence (récursivement). Le chemin source est généralement un répertoire.
  - **f** : forcer la copie.
  - **p** : préserver les permissions et les dates.

cp (II)

- Le comportement de la commande varie selon le type de la source et la destination.
- Exemples :
  - **cp file1.txt file2.txt**
  - **cp file1.txt file2.txt file3.txt**
  - **cp file1.txt file2.txt file3.txt rep1**
  - **cp file1.txt rep1/file4.txt**
  - **cp rep1 rep2**
  - **cp -r rep1 rep2**
  - **cp -r rep1 rep2**
  - **cp -r rep1 rep2/rep3**

mv (I)

- Déplacer ou renommer un fichier (*move*).
- Permet de déplacer plusieurs fichiers ou une arborescence.

`mv [options] chemin_source chemin_destination`

- Options :
  - **`i`** : demander une confirmation avant toute opération.
  - **`f`** : forcer l'opération.

mv (II)

- Le comportement de la commande varie selon le type de la source et la destination.
- Exemples :
  - **mv file1.txt file2.txt**
  - **mv file1.txt file2.txt file3.txt**
  - **mv file1.txt file2.txt file3.txt rep1**
  - **mv file1.txt rep1/file4.txt**
  - **mv rep1 rep2 # rep2 n'existe pas**
  - **mv rep1 rep2 # rep2 existe**
  - **mv rep1 rep2/rep3 # rep3 n'existe pas**

Caractères de substitution (I)

Caractères	Rôle
*	Remplace une chaîne de longueur variable, même vide.
?	Remplace un caractère unique quelconque.
[...]	Une série ou une plage de caractères.
[a-z]	Un caractère parmi la plage indiquée (de a à z inclus).
[!...]	Inversion de la recherche.
[^...]	Inversion de la recherche.

Caractères de substitution (II)

- Exemples :
  - **ls a***
  - **ls a???**
  - **ls b??***
  - **ls *[12]**
  - **ls *[!3]**
  - **ls [A-Z]*[0-9]**
  - **ls [a-c]?*[12]**

Caractères de substitution (III)

- Le shell interprète et remplace les caractères de substitution avant de passer les paramètres à la commande.
- Exemple :
  - **`touch file1.txt file2.txt file3.txt`**
  - **`cp file?.txt rep1`**

Caractères de substitution (IV)

- Il est possible de verrouiller des caractères spéciaux pour interdire leurs interprétation pas le shell.
  - **** : antislash, permet de verrouiller un caractère unique.
  - **"…"** : guillemets, permettent de verrouiller tout les caractères spéciaux sauf **$ \ et `**.
  - **'…'** : apostrophes, permettent de verrouiller tout les caractères spéciaux.
- Exemples :
  - **touch my file.txt**
  - **touch my\ file.txt**
  - **touch "Hello World.c"**
  - **cp 'my file.txt' rep1**

find

- Rechercher récursivement dans un chemin des fichiers à l'aide de critères et permet de faire des opérations sur les résultats.

`find chemin critères opérations`

- Par défaut :
  - **`chemin`** : le répertoire courant → **`'.'`**
  - **`opération`** : afficher sur l'écran → **`'-print'`**
- Exemples :
  - **`find`**
  - **`find . -print`**

find – critère : name

- Rechercher selon le nom du fichier.
- Il est possible d'utiliser les caractères de substitution.
- Il est recommandé de mettre le nom du fichier entre guillemets "…"
- Exemple :
  - **`find Documents -name "*.pdf"`**

find – critère : type

- Rechercher selon le type du fichier.

Code	Type de fichier
`f`	Fichier ordinaire.
`d`	Fichier répertoire.
`l`	Fichier lien symbolique.
`b`	Fichier spécial en mode bloc.
`c`	Fichier spécial en mode caractère.
`p`	Tube nommé (pipe).
`s`	Socket (connexion réseau).

- Exemples :
  - **`find . -type l`**
  - **`find . -name "rep*" -type d`**

find – critères : user et group

- Rechercher selon le propriétaire ou le groupe du fichier.
- Il est possible d'utiliser soit les noms (utilisateur, groupe) soit les identifiants (UID, GID).
- Exemples :
  - **`find /etc -user 0`**
  - **`find . -type d -user 1000 -group LPIC1`**

find – critère : size

- Rechercher selon la taille du fichier.

Code	Signification
`b`	Taille en bloc (512 o). Par défaut.
`c`	Taille en caractère (1 o).
`w`	Taille en mot (2 o).
`k, M, G`	Taille en Ko, Mo, et Go

Exemples :

- **`-size 5 # Fichiers de taille 5 blocs.`**
- **`-size 42c # Fichiers de taille 42 octets.`**
- **`-size -100k # Fichiers de taille < à 100 Ko.`**
- **`find /var/log -size +10M # Fichiers de taille > à 10 Mo.`**
- Le critère **`-empty`** peut remplacer **`-size 0`**

find – critères : atime, mtime et ctime

- Rechercher selon :
  - atime (*access time*) : La date du dernier accès.
  - mtime (*modification time*) : La date de la dernière modification.
  - ctime (*change time*) : La date du changement des informations liées à l'inode (nom, droits, etc).
- Travaillent avec le nombre de jours.
- Exemples :
  - **`find /var/log -mtime 1 # Fichiers modifiés hier (entre 24 et 48 heures).`**
  - **`find /var/log -mtime -3 # Fichiers modifiés il y a moins de 3 jours (moins de 72 heures).`**
  - **`Find /var/log -atime +4 # Fichiers accédés il y a plus de 4 jours (plus de 96 heures).`**

find – critère : perm

- Rechercher selon les droits, permissions, du fichier.
- Les droits doivent être précisés complets et en octale.
- Exemple :
  - **find /var/log -type f -perm 644 # Fichiers avec les droits 644**
  - **find /var/log -type f -perm -644 # Fichiers avec au moins les droits 644**
  - **Find /var/log -type f -perm /644 # Fichiers avec l'un des droits 6 ou 4 ou 4**

find – critères : links et inum

- Rechercher selon :
  - links (*hard links*) : Le nombre de liens physiques.
  - inum (*inode number*) : Le numéro d'inode.
- Exemples :
  - **`find /var/log -links +2 # Fichiers avec plus de 2 liens → généralement les répertoires.`**
  - **`find /var/log -links -2 # Fichiers avec moins de 2 liens → généralement les fichiers ordinaires.`**
  - **`Find . -inum  422142 # Fichier avec le numéro d'inode 422142.`**

find – combinaison de critères

- Il est possible de combiner les critères de recherche.

Critère	Action
`a, and`	ET logique. **Par défaut**.
`o, or`	OU logique.
`!`	NOT, négation.

Exemples :

- **`find . ! -name "*.c" -print`**
- **`find . \( -type f -o -type d \) -ls`**
- **`find . ! \( -type f -o -type d \) -ls`**

find – opération : print

- Afficher les noms des fichiers trouvés.
- Opération par défaut.
- Exemples :
  - **`find . -links +2 -print`**
  - **`find . -type l -print`**
  - **`find . -perm 777 -print`**

find – opération : ls

- Afficher des informations détaillées sur les fichiers trouvés.
- Détails de la commande **`ls`** avec les options **`d`**, **`i`**, **`l`** et **`s`**.
- Exemples :
  - **`find . -links +2 -ls`**
  - **`find . -type l -ls`**
  - **`find . -perm 777 -ls`**

find – opération : exec

- Exécuter une commande pour chaque fichier trouvé.
  - Doit être la dernière opération dans le commande **find**.
  - La commande à exécuter doit se terminer par un « **;** ».
  - Pour passer le fichier trouvé par **find** comme paramètre à la commende il faut utiliser **{}** → substitution de fichier.
- Exemple :
  - **find . -type f -name "*.tmp" -exec rm -f {} \;**

find – opération : ok

- Opération identique à l'opération exec mais, demande une confirmation de l'utilisateur pour chaque fichier trouvé.

- Exemple :
  - **`find . -type f -name "*.tmp" -ok rm -f {} \;`**

tar (I)

- Créer des archives de fichiers et d'arborescence de fichiers.

`tar options archive chemins`

- La première lettre dans les options doit indiquer l'opération :
  - c : créer l'archive.
  - x : extraire (restaurer) l'archive.
  - t : lister le contenu de l'archive.
- Options :
  - v : afficher les détails.
  - z : compresser l'archive au format gzip.
  - j : compresser l'archive au format bzip2.
  - f : le paramètre suivant est le nom de l'archive.

tar (II)

- Exemples :
  - **tar cvf mesDocuments.tar mesDocuments**
  - **tar tvf mesDocuments.tar**
  - **tar xvf mesDocuments.tar**
  - **tar czvf mesDocuments.tar.gz mesDocuments**
  - **tar cjvf mesDocuments.tar.bz2 mesDocuments**

cpio (I)

- Sauvegarder sur la sortie standard les fichiers dont on saisit les noms sur l'entrée standard.
- Il est nécessaire d'utiliser les redirections pour créer le fichier de sauvegarde.
- Options :
  - o : création de la sauvegarde en sortie (output).
  - i : lecture de l'archive en entrée (input).
  - v : afficher les détails.
  - t : lister le contenu de l'archive.
  - d : reconstruire l'arborescence des répertoires manquants.

cpio (II)

- Exemples :
  - `find . -type f -name "*.c" | cpio -ov > mesProg.cpio`
  - `cpio -it < mesProg.cpio`
  - `cpio -iv < mesProg.cpio`

gzip & gunzip

- Compresser (gzip) ou décompresser (gunzip) des fichiers.
- La commande gzip supprime les fichiers compressés.
- Exemples :
  - **`gzip prog.c # Création de prog.c.gz et suppression de prog.c`**
  - **`gzip -c prog.c > prog.c.gz # Forcer le résultat à passer par la sortie standard.`**
  - **`gunzip prog.c.gz`**

bzip2 & bunzip2

- Compresser (bzip2) ou décompresser (bunzip2) des fichiers.
- La commande bzip2 supprime les fichiers compressés.
- Exemples :
  - **bzip2 prog.c # Création de prog.c.bz2 et suppression de prog.c**
  - **bzip2 -k prog.c # Création de prog.c.bz2 et sans suppression de prog.c**
  - **bunzip2 prog.c.bz2**
  - **bzip2 -d prog.c.bz2**

xz & unxz

- Compresser (xz) ou décompresser (unxz) des fichiers.
- La commande xz supprime les fichiers compressés.
- Exemples :
  - **`xz prog.c # Création de prog.c.xz et suppression de prog.c`**
  - **`xz -k prog.c # Création de prog.c.xz et sans suppression de prog.c`**
  - **`unxz prog.c.xz`**
  - **`xz -d prog.c.xz`**

dd

- Copier physiquement, bloc par bloc, un fichier périphérique vers un fichier périphérique ou un fichier ordinaire (*device to device*).

Argument	Rôle
`if=fichier`	Nom du fichier en entrée.
`of=fichier`	Nom du fichier en sortie.
`bs=n`	Taille du bloc en octet.
`count=n`	Nombre de bloc à copier.
`skip=n`	Nombre de bloc à sauter au début du fichier d'entrée.

- Exemple :
  - **`dd if=/dev/zero of=vide bs=1024 count=1024`**