Version : **2024.01**

Dernière mise-à-jour : 2024/03/11 09:17

LDF401 - Système de Fichiers

Contenu du Module

LAB #1 - Linux File Hierarchy System

Le système de fichiers de Linux est organisé autour d'une arborescence unique ayant un point de départ appelé la **racine**, représenté par le caractère **/**. En dessous de cette racine se trouvent des répertoires contenant fichiers et sous-répertoires. L'organisation des répertoires est conforme à un standard, appelé le **Linux File Hierarchy System**.

```
trainee@debian11:~$ cd /
```

```
trainee@debian11:/$ ls -l
total 60
lrwxrwxrwx   1 root root     7 Apr 25 06:26 bin -> usr/bin
drwxr-xr-x   3 root root  4096 Apr 25 06:54 boot
drwxr-xr-x  17 root root  3240 May 10 14:37 dev
drwxr-xr-x 112 root root  4096 May 10 14:37 etc
drwxr-xr-x   3 root root  4096 Apr 25 07:01 home
lrwxrwxrwx   1 root root    31 Apr 25 06:31 initrd.img -> boot/initrd.img-5.10.0-13-amd64
lrwxrwxrwx   1 root root    31 Apr 25 06:31 initrd.img.old -> boot/initrd.img-5.10.0-13-amd64
lrwxrwxrwx   1 root root     7 Apr 25 06:26 lib -> usr/lib
lrwxrwxrwx   1 root root     9 Apr 25 06:26 lib32 -> usr/lib32
lrwxrwxrwx   1 root root     9 Apr 25 06:26 lib64 -> usr/lib64
lrwxrwxrwx   1 root root    10 Apr 25 06:26 libx32 -> usr/libx32
drwx------   2 root root 16384 Apr 25 06:26 lost+found
drwxr-xr-x   3 root root  4096 Apr 25 06:26 media
drwxr-xr-x   2 root root  4096 Apr 25 06:27 mnt
drwxr-xr-x   2 root root  4096 Apr 25 06:27 opt
dr-xr-xr-x 166 root root     0 May 10 14:37 proc
drwx------   3 root root  4096 Apr 25 07:05 root
drwxr-xr-x  24 root root   660 May 10 14:37 run
lrwxrwxrwx   1 root root     8 Apr 25 06:26 sbin -> usr/sbin
drwxr-xr-x   2 root root  4096 Apr 25 06:27 srv
dr-xr-xr-x  13 root root     0 May 10 14:37 sys
drwxrwxrwt  10 root root  4096 May 10 14:37 tmp
drwxr-xr-x  14 root root  4096 Apr 25 06:27 usr
drwxr-xr-x  11 root root  4096 Apr 25 06:27 var
lrwxrwxrwx   1 root root    28 Apr 25 06:31 vmlinuz -> boot/vmlinuz-5.10.0-13-amd64
lrwxrwxrwx   1 root root    28 Apr 25 06:31 vmlinuz.old -> boot/vmlinuz-5.10.0-13-amd64
```

- **/bin** : est une abréviation de ***bin****ary* ou binaires. Il contient des programmes tels ls. Sous Debian 11 il s'agit d'un lien symbolique qui pointe vers /usr/bin.
- **/boot** : contient les fichiers nécessaires au démarrage du système.
- **/dev** : contient les nœuds utilisés pour accéder à tout type de matériel tel /dev/fd0 pour le lecteur de disquette. C'est le binaire *udev* qui se charge de créer et supprimer d'une manière dynamique les nœuds.

- **/etc** : contient des fichiers de configuration tels passwd pour les mots de passe et fstab qui est la liste des systèmes de fichiers à monter lors du démarrage du système.
- **/home** : contient les répertoires de chaque utilisateur sauf l'utilisateur root.
- **/lib** : contient les bibliothèques 32 bits communes utilisées par les programmes ainsi que les modules. Sous Debian 11 il s'agit d'un lien symbolique qui pointe vers /usr/lib.
- **/lib64** : contient les bibliothèques 64 bits communes utilisées par les programmes ainsi que les modules. Sous Debian 11 il s'agit d'un lien symbolique qui pointe vers /usr/lib64.
- **/media** : contient des répertoires pour chaque système de fichiers monté (accessible au système linux) tels floppy, cdrom etc.
- **/mnt** : contient des répertoires pour chaque système de fichiers monté temporairement par root.
- **/opt** : contient des applications optionnelles.
- **/proc** : contient un système de fichiers virtuel qui extrait de la mémoire les informations en cours de traitement. Le contenu des fichiers est créé dynamiquement lors de la consultation. Seul root peut consulter la totalité des informations dans le répertoire /proc.
- **/root** : le home de root, l'administrateur système.
- **/run** : remplace le répertoire /var/run.
- **/sbin** : contient des binaires, donc programmes, pour l'administration du système local. Sous Debian 11 il s'agit d'un lien symbolique qui pointe vers /usr/sbin.
- **/srv** : contient des données pour les **s**e**rv**ices hébergés par le système tels ftp, bases de données, web etc.
- **/sys** : contient un système de fichiers virtuel dont le rôle est de décrire le matériel pour udev.
- **/tmp** : stocke des fichiers temporaires créés par des programmes.
- **/usr** : contient des commandes des utilisateurs dans /usr/bin, les HOWTO dans /usr/share/doc, les manuels dans /usr/share/man ainsi que d'autres entrées majeures.
- **/var** : contient des fichiers de taille variable.

1.1 - Types de Fichiers

Il existe trois types majeurs de fichier sous le système Linux :

- les fichiers normaux (ordinary files)
- les répertoires (directories)
- les fichiers spéciaux (special files ou Devices)

Le fichiers normaux sont des fichiers textes, des tableaux ou des exécutables.

La longueur du nom de fichier est limité à 255 caractères.

Il y a une distinction entres les majuscules et le minuscules.

Si le nom d'un fichier commence par un **.**, le fichier devient caché.

1.2 - La Commande mount

Pour que Linux soit informé de la présence d'un système de fichiers, ce système doit être monté. Pour monter un système de fichiers, on utilise la commande **mount** :

```
# mount /dev/<fichier_spécial> /mnt/<répertoire_cible>
```

ou **/dev/<fichier_spécial>** est le périphérique à monter et **/mnt/<répertoire_cible>** est le répertoire qui servira comme «fenêtre» pour visionner le contenu du système de fichiers. Ce répertoire doit impérativement exister avant d'essayer de monter le système de fichiers.

A faire : Connectez-vous à votre machine virtuelle en tant que **trainee** avec le mot de passe **trainee**. Ouvrez un terminal via les menus **Applications > Favorites > Terminal**. Tapez la commande **su -** et appuyez sur la touche ↵ Entrée. Indiquez le mot de passe **fenestros**. Vous êtes maintenant connecté en tant que l'administrateur **root** et vous pouvez reproduire les exemples qui suivent.

Dans le cas où la commande **mount** est utilisée sans options, le système retourne une liste de tous les systèmes de fichiers actuellement montés :

```
trainee@debian11:/$ su -
Password: fenestros
root@debian11:~# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=1989872k,nr_inodes=497468,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,noexec,relatime,size=402560k,mode=755)
```

```
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
cgroup2 on /sys/fs/cgroup type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate,memory_recursiveprot)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
none on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs
(rw,relatime,fd=29,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=10999)
tracefs on /sys/kernel/tracing type tracefs (rw,nosuid,nodev,noexec,relatime)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,nosuid,nodev,noexec,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
configfs on /sys/kernel/config type configfs (rw,nosuid,nodev,noexec,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,nosuid,nodev,noexec,relatime)
tmpfs on /run/user/113 type tmpfs
(rw,nosuid,nodev,relatime,size=402556k,nr_inodes=100639,mode=700,uid=113,gid=121)
tmpfs on /run/user/1000 type tmpfs
(rw,nosuid,nodev,relatime,size=402556k,nr_inodes=100639,mode=700,uid=1000,gid=1000)
```

Cette information est stockée dans le fichier **/etc/mtab** :

```
root@debian11:~# cat /etc/mtab
sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
proc /proc proc rw,nosuid,nodev,noexec,relatime 0 0
udev /dev devtmpfs rw,nosuid,relatime,size=1989872k,nr_inodes=497468,mode=755 0 0
devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
tmpfs /run tmpfs rw,nosuid,nodev,noexec,relatime,size=402560k,mode=755 0 0
/dev/sda1 / ext4 rw,relatime,errors=remount-ro 0 0
securityfs /sys/kernel/security securityfs rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /dev/shm tmpfs rw,nosuid,nodev 0 0
tmpfs /run/lock tmpfs rw,nosuid,nodev,noexec,relatime,size=5120k 0 0
cgroup2 /sys/fs/cgroup cgroup2 rw,nosuid,nodev,noexec,relatime,nsdelegate,memory_recursiveprot 0 0
pstore /sys/fs/pstore pstore rw,nosuid,nodev,noexec,relatime 0 0
```

```
none /sys/fs/bpf bpf rw,nosuid,nodev,noexec,relatime,mode=700 0 0
systemd-1 /proc/sys/fs/binfmt_misc autofs
rw,relatime,fd=29,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=10999 0 0
tracefs /sys/kernel/tracing tracefs rw,nosuid,nodev,noexec,relatime 0 0
mqueue /dev/mqueue mqueue rw,nosuid,nodev,noexec,relatime 0 0
debugfs /sys/kernel/debug debugfs rw,nosuid,nodev,noexec,relatime 0 0
hugetlbfs /dev/hugepages hugetlbfs rw,relatime,pagesize=2M 0 0
configfs /sys/kernel/config configfs rw,nosuid,nodev,noexec,relatime 0 0
fusectl /sys/fs/fuse/connections fusectl rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /run/user/113 tmpfs rw,nosuid,nodev,relatime,size=402556k,nr_inodes=100639,mode=700,uid=113,gid=121 0 0
tmpfs /run/user/1000 tmpfs rw,nosuid,nodev,relatime,size=402556k,nr_inodes=100639,mode=700,uid=1000,gid=1000 0 0
```

Important : Notez que le système de fichiers de /dev/sda1 est **ext4**. La comparaison des systèmes de fichiers Linux est abordée dans le module **LDF504 - Gestion des Disques, des Systèmes de Fichiers et du Swap**.

Les options de la commande **mount** sont :

```
root@debian11:~# mount --help

Usage:
 mount [-lhV]
 mount -a [options]
 mount [options] [--source] <source> | [--target] <directory>
 mount [options] <source> <directory>
 mount <operation> <mountpoint> [<target>]

Mount a filesystem.

Options:
 -a, --all               mount all filesystems mentioned in fstab
```

```
 -c, --no-canonicalize   don't canonicalize paths
 -f, --fake              dry run; skip the mount(2) syscall
 -F, --fork              fork off for each device (use with -a)
 -T, --fstab <path>      alternative file to /etc/fstab
 -i, --internal-only     don't call the mount.<type> helpers
 -l, --show-labels       show also filesystem labels
 -n, --no-mtab           don't write to /etc/mtab
     --options-mode <mode>
                         what to do with options loaded from fstab
     --options-source <source>
                         mount options source
     --options-source-force
                         force use of options from fstab/mtab
 -o, --options <list>    comma-separated list of mount options
 -O, --test-opts <list>  limit the set of filesystems (use with -a)
 -r, --read-only         mount the filesystem read-only (same as -o ro)
 -t, --types <list>      limit the set of filesystem types
     --source <src>      explicitly specifies source (path, label, uuid)
     --target <target>   explicitly specifies mountpoint
     --target-prefix <path>
                         specifies path use for all mountpoints
 -v, --verbose           say what is being done
 -w, --rw, --read-write  mount the filesystem read-write (default)
 -N, --namespace <ns>    perform mount in another namespace

 -h, --help              display this help
 -V, --version           display version

Source:
 -L, --label <label>     synonym for LABEL=<label>
 -U, --uuid <uuid>       synonym for UUID=<uuid>
 LABEL=<label>           specifies device by filesystem label
 UUID=<uuid>             specifies device by filesystem UUID
 PARTLABEL=<label>       specifies device by partition label
```

```
 PARTUUID=<uuid>         specifies device by partition UUID
 ID=<id>                 specifies device by udev hardware ID
 <device>                specifies device by path
 <directory>             mountpoint for bind mounts (see --bind/rbind)
 <file>                  regular file for loopdev setup

Operations:
 -B, --bind              mount a subtree somewhere else (same as -o bind)
 -M, --move              move a subtree to some other place
 -R, --rbind             mount a subtree and all submounts somewhere else
 --make-shared           mark a subtree as shared
 --make-slave            mark a subtree as slave
 --make-private          mark a subtree as private
 --make-unbindable       mark a subtree as unbindable
 --make-rshared          recursively mark a whole subtree as shared
 --make-rslave           recursively mark a whole subtree as slave
 --make-rprivate         recursively mark a whole subtree as private
 --make-runbindable      recursively mark a whole subtree as unbindable

For more details see mount(8).
```

1.3 - La Commande umount

Pour démonter un système de fichiers, on utilise la commande umount :

```
# umount /mnt/<répertoire_cible>
```

ou

```
# umount /dev/cdrom
```

Les options de la commande **umount** sont :

```
root@debian11:~# umount --help

Usage:
 umount [-hV]
 umount -a [options]
 umount [options] <source> | <directory>

Unmount filesystems.

Options:
 -a, --all               unmount all filesystems
 -A, --all-targets       unmount all mountpoints for the given device in the
                           current namespace
 -c, --no-canonicalize   don't canonicalize paths
 -d, --detach-loop       if mounted loop device, also free this loop device
     --fake              dry run; skip the umount(2) syscall
 -f, --force             force unmount (in case of an unreachable NFS system)
 -i, --internal-only     don't call the umount.<type> helpers
 -n, --no-mtab           don't write to /etc/mtab
 -l, --lazy              detach the filesystem now, clean up things later
 -O, --test-opts <list>  limit the set of filesystems (use with -a)
 -R, --recursive         recursively unmount a target with all its children
 -r, --read-only         in case unmounting fails, try to remount read-only
 -t, --types <list>      limit the set of filesystem types
 -v, --verbose           say what is being done
 -q, --quiet             suppress 'not mounted' error messages
 -N, --namespace <ns>    perform umount in another namespace

 -h, --help              display this help
 -V, --version           display version

For more details see umount(8).
```

1.4 - Le Fichier /etc/fstab

Dans le cas où la commande **mount** est utilisée avec l'option **–a**, tous les systèmes de fichiers mentionnés dans un fichier spécial dénommé **/etc/fstab** seront montés en même temps.

```
root@debian11:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# systemd generates mount units based on this file, see systemd.mount(5).
# Please run 'systemctl daemon-reload' after making changes here.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=9887a74f-a680-4bde-8f04-db5ae9ea186e /               ext4    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=1f9439f5-4b19-49b1-b292-60c2c674cee9 none            swap    sw              0       0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0
```

Chaque ligne dans ce fichier contient 6 champs :

Champ 1	**Champ 2**	**Champ 3**	**Champ 4**	**Champ 5**	**Champ 6**
Fichier de bloc spécial ou UUID ou système de fichiers virtuel ou une étiquette	Point de montage	Type de système de fichiers	Options séparées par des virgules	Utilisé par *dump* (1 = à dumper, 0 ou vide = à ignorer)	L'ordre de vérification par *fsck* des systèmes de fichiers au moment du démarrage

L'**UUID** (*Universally Unique Identifier*) est une chaîne d'une longueur de 128 bits. Les UUID sont créés automatiquement et d'une manière aléatoire lors de la création du filesystem sur la partition. Ils peuvent être modifiés par l'administrateur.

Options de Montage

Les options de montage les plus importants sont :

Option	Systèmes de Fichier	Description	Valeur par Défaut
defaults	Tous	Egal à rw, suid, dev, exec, auto, nouser, async	S/O
auto/noauto	Tous	Montage automatique/pas de montage automatique lors de l'utilisation de la commande **mount -a**	auto
rw/ro	Tous	Montage en lecture-écriture/lecture seule	rw
suid/nosuid	Tous	Les bits SUID et SGID sont/ne sont pas pris en compte	suid
dev/nodev	Tous	Interprète/n'interprète pas les fichiers spéciaux de périphériques	dev
exec/noexec	Tous	Autorise:n'autorise pas l'exécution des programmes	exec
sync/async	Tous	Montage synchrone/asynchrone	async
user/nouser	Tous	Autorise/n'autorise pas un utilisateur à monter/démonter le système de fichier. Le point de montage est celui spécifié dans le fichier /etc/fstab. Seul l'utilisateur qui a monté le système de fichiers peut le démonter	S/O
users	Tous	Autorise tous les utilisateurs à monter/démonter le système de fichier	S/O
owner	Tous	Autorise le propriétaire du périphérique de le monter	S/O
atime/noatime	Norme POSIX	Inscrit/n'inscrit pas la date d'accès	atime
uid=valeur	Formats non-Linux	Spécifie le n° du propriétaire des fichiers pour les systèmes de fichiers non-Linux	root
gid=valeur	Formats non-Linux	Spécifie le n° du groupe propriétaire	S/O
umask=valeur	Formats non-Linux	Spécifie les permissions (droits d'accès/lecture/écriture)	S/O
dmask=valeur	Formats non-Linux	Spécifie les droits d'usage des dossiers (Obsolète, préférer dir_mode)	umask actuel
dir_mode=valeur	Formats non-Linux	Spécifie les droits d'usage des dossiers	umask actuel
fmask=valeur	Formats non-Linux	Spécifie les droits d'usage des fichiers (Obsolète, préférer file_mode)	umask actuel
file_mode=valeur	Formats non-Linux	Spécifie les droits d'usage des fichiers	umask actuel

LAB #2 - Système de Fichiers Unix

Chaque partition sous un système Unix peut héberger une des structures suivantes :

- superbloc
- inode
- bloc de données
- blocs d'indirection

2.1- Superbloc

Le superbloc contient :

- la taille des blocs
- la taille du système de fichiers
- le nombre de montages effectués pour ce système de fichiers
- un pointeur vers la racine du système de fichiers
- les pointeurs vers la liste des inodes libres
- les pointeurs vers la liste des blocs de données libres

Le Superbloc est dupliqué tous les 8 ou 16Mo sous ext3 et ext4. Pour réparer un système de fichiers en restaurant un Superbloc, utilisez la commande suivante :

```
# e2fsck -f -b 8193 /dev/sda1 [Enter]
```

Pour visualiser l'emplacement du Superbloc primaire et ses sauvegardes, utilisez la commande suivante :

```
root@debian11:~# dumpe2fs /dev/sda1 | grep -i superbloc
dumpe2fs 1.46.2 (28-Feb-2021)
  Primary superblock at 0, Group descriptors at 1-4
  Backup superblock at 32768, Group descriptors at 32769-32772
  Backup superblock at 98304, Group descriptors at 98305-98308
  Backup superblock at 163840, Group descriptors at 163841-163844
  Backup superblock at 229376, Group descriptors at 229377-229380
  Backup superblock at 294912, Group descriptors at 294913-294916
  Backup superblock at 819200, Group descriptors at 819201-819204
  Backup superblock at 884736, Group descriptors at 884737-884740
```

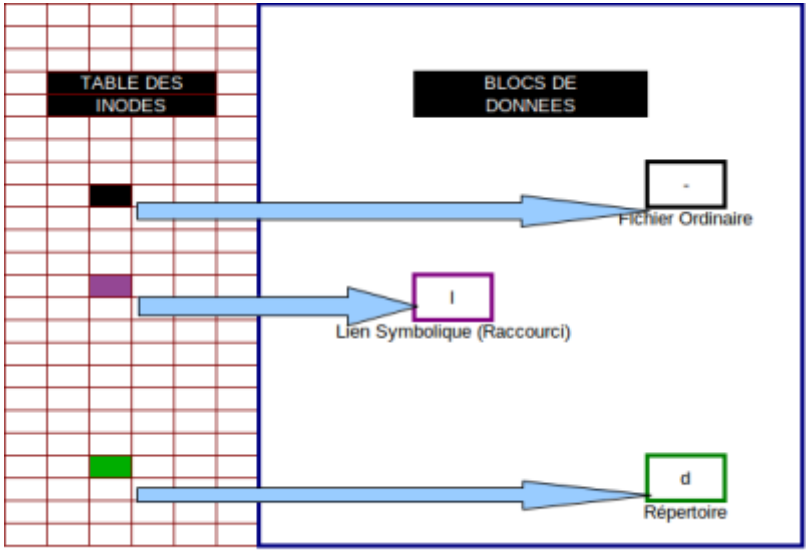
```
  Backup superblock at 1605632, Group descriptors at 1605633-1605636
  Backup superblock at 2654208, Group descriptors at 2654209-2654212
  Backup superblock at 4096000, Group descriptors at 4096001-4096004
  Backup superblock at 7962624, Group descriptors at 7962625-7962628
```

2.2 - Inodes

Chaque fichier est représenté par un **inode**. L'inode contient :

- le type de fichier, soit **-**, **d**, **l**, **b**, **c**, **p**, **s**
- les droits d'accès, par exemple **rwx rw- r–**
- le nombre de liens physiques soit le nombre de noms
- l'UID du créateur ou l'UID affecté par la commande **chown** s'il y a eu une modification
- le GID du processus créateur ou le GID affecté par la commande **chgrp**
- la taille du fichier en octets
- la date de dernière modification de l'inode, soit le **ctime**
- la date de dernière modification du fichier, soit le **mtime**
- la date du dernier accès, soit le **atime**
- les adresses qui pointent vers les blocs de données du fichier

Graphiquement, on peut schématiser cette organisation de la façon suivante :



Pour mieux comprendre, tapez la commande suivante :

```
root@debian11:~# ls -ld /dev/console /dev/sda /etc /etc/passwd
crw-------   1 root root 5, 1 May 10 14:37 /dev/console
brw-rw----   1 root disk 8, 0 May 10 14:37 /dev/sda
drwxr-xr-x 112 root root 4096 May 10 14:37 /etc
-rw-r--r--   1 root root 2007 Apr 25 07:04 /etc/passwd
```

Le premier caractère de chaque ligne peut être un des suivants :

- **-** - un fichier
- **d** - un répertoire
- **l** - un lien symbolique
- **b** - un périphérique du type bloc
- **c** - un périphérique du type caractère
- **p** - un tube nommé pour la communication entre processus
- **s** - un socket dans un contexte réseau

Pour visualiser le numéro d'inode, utilisez l'option **-i** :

```
root@debian11:~# ls -ldi /dev/console /dev/sda /etc /etc/passwd
    12 crw-------   1 root root 5, 1 May 10 14:37 /dev/console
   210 brw-rw----   1 root disk 8, 0 May 10 14:37 /dev/sda
784897 drwxr-xr-x 112 root root 4096 May 10 14:37 /etc
790923 -rw-r--r--   1 root root 2007 Apr 25 07:04 /etc/passwd
```

2.3 - Blocs de données

Les données sont stockées dans des blocs de données. Dans le cas d'un répertoire, le bloc de données contient une table qui référence les inodes et les noms des fichiers dans le répertoire. Cette table s'appelle une **table catalogue**.

Le nom d'un fichier n'est pas stocké dans l'inode mais dans une **table catalogue**. Cette particularité nous permet de donner deux noms différents au même fichier. Pour ajouter un nouveau nom à un fichier, il convient de créer un **lien physique**.

2.4 - Liens Physiques

Un lien physique se crée en utilisant la commande suivante :

- ln nom_du_fichier nom_supplémentaire

Pour illustrer ce point, tapez la ligne de commande suivante :

```
root@debian11:~# cd /tmp; mkdir inode; cd inode; touch file1; ls -ali
total 8
785012 drwxr-xr-x  2 root root 4096 May 10 16:01 .
654081 drwxrwxrwt 11 root root 4096 May 10 16:01 ..
790928 -rw-r--r--  1 root root    0 May 10 16:01 file1
```

Notez bien le numéro de l'inode du fichier **file1**. Notez aussi que le numéro dans le troisième champs de la ligne de file1 a la valeur **1** :

```
790928 -rw-r--r--  1 root root    0 May 10 16:01 file1
```

Créez maintenant un lien physique :

```
root@debian11:/tmp/inode# ln file1 file2

root@debian11:/tmp/inode# ls -ali
total 8
785012 drwxr-xr-x  2 root root 4096 May 10 16:02 .
654081 drwxrwxrwt 11 root root 4096 May 10 16:01 ..
790928 -rw-r--r--  2 root root    0 May 10 16:01 file1
790928 -rw-r--r--  2 root root    0 May 10 16:01 file2
```

Notez les deux lignes suivantes :

```
790928 -rw-r--r--  2 root root    0 May 10 16:01 file1
790928 -rw-r--r--  2 root root    0 May 10 16:01 file2
```

Les deux fichiers, file1 et file2, sont référencés par le même inode. Le nombre de liens est donc augmenté de 1 (le numéro dans le troisième champs).

Important : Un lien physique ne peut être créé que dans le cas où les deux fichiers se trouvent dans le même filesystem et que le fichier source existe.

2.5 - Liens Symboliques

Un lien symbolique est un **raccourci** vers un autre fichier ou répertoire. Un lien symbolique se crée en utilisant la commande suivante :

- ln -s nom_du_fichier nom_raccourci

Pour illustrer ce point, tapez la ligne de commande suivante :

```
root@debian11:/tmp/inode# ln -s file1 file3
```

```
root@debian11:/tmp/inode# ls -ali
total 8
785012 drwxr-xr-x  2 root root 4096 May 10 16:05 .
654081 drwxrwxrwt 11 root root 4096 May 10 16:01 ..
790928 -rw-r--r--  2 root root    0 May 10 16:01 file1
790928 -rw-r--r--  2 root root    0 May 10 16:01 file2
790931 lrwxrwxrwx  1 root root    5 May 10 16:05 file3 -> file1
```

Notez que le lien symbolique est référencé par un autre inode. Le lien symbolique pointe vers le file1.

Important : Un lien symbolique peut être créé même dans le cas où les deux fichiers se trouvent dans deux filesystems différents et même dans le cas où le fichier source n'existe pas.
