Manipulation de fichiers

Table des matières

Noms des fichiers et des répertoires

Les noms des fichiers et des répertoires sous Linux répondent à certaines règles d'écriture.

Description du contenu

En premier lieu, un nom de fichier doit fournir des indications sur son contenu ; ceci est valable sur tous les systèmes d'exploitation.

Pour un professeur, il est plus facile de retrouver les notes d'histoire de la classe 6e A du premier trimestre 2016 dans le fichier nommé notes.histoire.classe.6A.trim.1.an.2016 que dans le fichier qui porte le nom glop.

Syntaxe

La casse des caractères (majuscules/minuscules) est déterminante dans la syntaxe des noms de fichiers. Ainsi, les fichiers glop, Glop et GLOP sont distincts.

Seuls les caractères alphanumériques (de a à z, A à Z et 0 à 9) ainsi que quelques autres (_, ., @, -, +) devraient être utilisés dans les noms de fichiers.

Certains caractères ont une signification spéciale sur la ligne de commande. Par exemple, supprimer a* ne signifie pas effacer le fichier nommé a* mais tous les fichiers commençant par un a.

En fait, tous les caractères peuvent être utilisés dans les noms de fichiers à l'exception de /, caractère de séparation des noms de répertoires sur la ligne de commande.

Les signes - et + sont à éviter en début de nom de fichier car il peut y avoir ambiguïté avec des options sur la ligne de commande. Par exemple, sur la ligne suivante, -fic est-il un nom de fichier passé en argument à la commande, ou s'agit-il des options **f**, **i** et **c** ?

```
$ ls -fic
```

Enfin, le . (point) n'a pas de signification spéciale dans un nom de fichier, excepté en première position où il indique que ce fichier est caché. La seule particularité d'un fichier caché est de ne pas être affiché par défaut dans la liste du répertoire.

Il n'y a pas, sous Linux, de notion d'extension déterminante comme sous Windows. Ce n'est pas un nom finissant par .exe, .com ou .bat qui permet l'exécution d'un fichier mais les droits qui y sont associés ; les droits sont expliqués plus loin.

Longueur

La longueur maximale d’un nom de fichier (ou de répertoire) sous Linux est de 255 caractères.

Cette limite ne pose pas de problème mais certains outils n’acceptent pas des chemins de fichiers (voir ci-après) d’une taille infinie. C’est le cas de la commande **tar**, pour laquelle ces chemins ne peuvent excéder 1 024 caractères

Types de fichiers

On entend par fichier, sous Unix, la structure contenant des données utilisateur.

Les fichiers standards sont constitués d’une suite de caractères ou flux d’octets dont le format n’est pas imposé par le système mais par les applications.

L’analogie entre octet et caractère est due au fait qu’un caractère est codé avec un octet ; la taille d’un fichier peut donc être donnée indifféremment en octets ou en caractères.

Les répertoires sont des fichiers particuliers, pouvant contenir plusieurs autres fichiers (répertoires ou non). Cela permet d’organiser les fichiers de façon hiérarchique et arborescente.

Les termes "répertoire" et "fichier", utilisés sous Linux, sont équivalents à "dossier" et "document" employés habituellement sous Windows et Mac OS.

Il existe sept types de fichiers sous Linux, les trois plus importants étant :

- fichier standard ou ordinaire ;
- répertoire ;
- lien symbolique ou logique (soft link).

Les quatre autres types de fichiers, manipulés principalement en administration ou en programmation système, sont :

- fichier pointant vers un périphérique de type "bloc" (fichiers présents dans /dev) ;
- fichier pointant vers un périphérique de type "caractère" (fichiers présents dans /dev) ;

- fichier tampon ou tube nommé (named pipe) pour la communication entre processus ;
- fichier "socket", comme les tubes nommés mais généralement dans un contexte réseau.

Les fichiers sont stockés et référencés dans un système de fichiers. Ils peuvent porter plusieurs noms (références) grâce à la structure même des systèmes de fichiers Unix.

Chemins

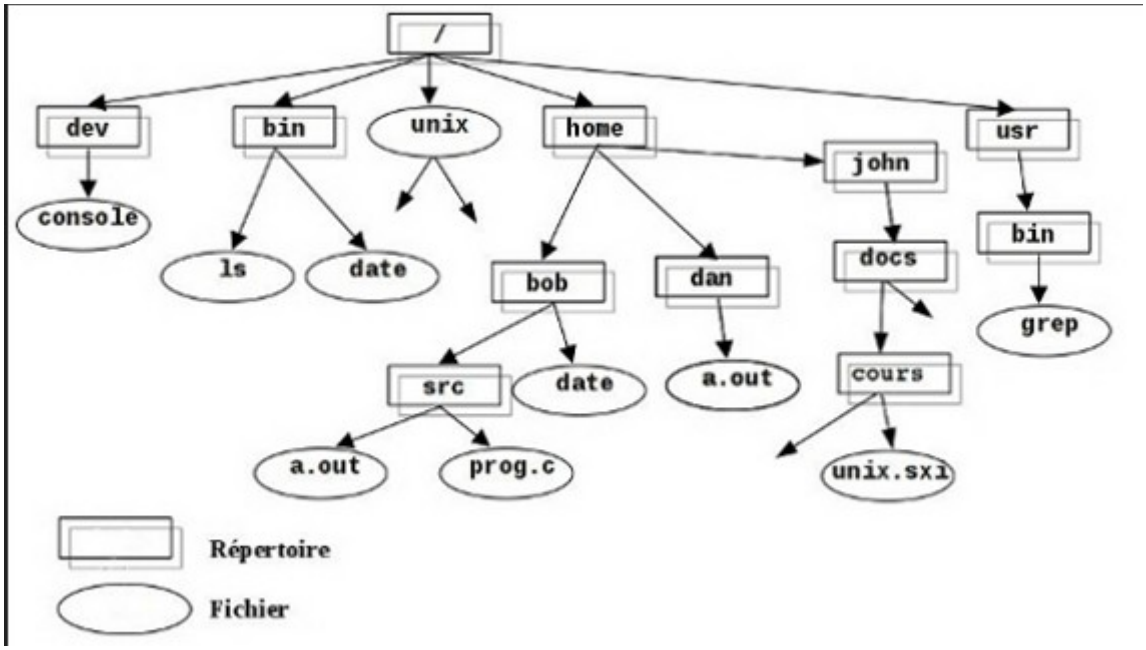
Connaître le nom d'un fichier (ou d'un répertoire) n'est pas suffisant pour accéder à celui-ci ; un même nom peut être en effet associé à plusieurs fichiers dans des répertoires différents.

La référence pleinement qualifiée d'un fichier est appelée "chemin" et indique le répertoire dans lequel figure le fichier. Les noms de répertoires et de fichiers sont séparés pas un / (slash) dans les chemins.

Attention, sur un système Windows, le séparateur entre les noms de fichiers et de répertoires est le \ ("antislash").

Il existe trois types de chemins : absolus, relatifs et personnels. Ils peuvent être utilisés indifféremment dans la façon de nommer des fichiers sur la ligne de commande.

Voici un exemple d'arborescence :



1. Chemins absolus

Un chemin absolu se base sur la racine de l'arborescence Linux. Tout chemin absolu commence donc par /.

Quel que soit l'endroit actuel où l'on se trouve, on pourra référencer le fichier notes de l'exemple précédent avec le chemin /home/driss/notes.

Par exemple :

```
[driss]$ ls /home/driss/date
/home/driss/date
```

2. Chemins relatifs

Les chemins relatifs dépendent du répertoire courant dans lequel se trouve l'utilisateur.

Sachant que chaque répertoire sur le système contient les fichiers . (point) et .. (point-point) référençant respectivement le répertoire courant (lui-même) et le répertoire parent, il existe plusieurs chemins relatifs désignant le fichier notes dans l'exemple :

Répertoire courant	Chemin relatif correspondant
/home	driss/notes
/home/driss	./notes
/home/driss/couleurs	../notes
/	home/driss/notes
/home	../tmp/./../home/./driss/notes

Pour un même répertoire courant (/home), il existe plusieurs chemins possibles, aussi absurdes soient-ils ; cette dernière manière de définir le chemin relatif au fichier notes pourra être employée dans des scripts shell, lors de la concaténation de plusieurs variables par exemple.

Par commodité, il est possible de supprimer le ./ au début d'un chemin relatif ; ainsi, le chemin relatif notes est équivalent à ./notes.

Par exemple :

```
[driss]$ pwd
/home/driss
[driss]$ ls date
date
[driss]$ ls ./date
```

./date
[driss]$ ls ../driss/date
../driss/notes

3. Chemins personnels

Cette troisième façon de définir le chemin d'un fichier se réfère au répertoire personnel d'un utilisateur ; on l'emploie donc principalement pour accéder aux fichiers présents dans les sous-répertoires de /home.

Un chemin personnel commence par le caractère ~ suivi du nom de l'utilisateur qui a pour répertoire personnel le répertoire contenu dans /home. Si le nom de l'utilisateur n'est pas spécifié, l'identité de l'utilisateur connecté est utilisée implicitement.

En considérant que les utilisateurs **gerard**, **nicolas** et **driss** ont respectivement pour répertoire personnel /home/gerard, /home/nicolas et /home/driss, on obtiendra :

- en tant que **gerard** :

[gerard]$ whoami
gerard
[gerard]$ ls ~/fichier1
/home/gerard/fichier1
[gerard]$ ls ~driss/formes/triangle
/home/driss/formes/triangle

- en tant que **nicolas** :

[nicolas]$ whoami
nicolas
[nicolas]$ ls ~gerard/fichier1
/home/gerard/fichier1
[nicolas]$ ls ~driss/notes
/home/driss/notes

- en tant que **driss** :

[driss]$ whoami
driss
[driss]$ ls ~/date
/home/driss/date
[driss]$ ls ~/couleurs/jaune
/home/driss/couleurs/jaune
[driss]$ ls ~gerard/fichier1
/home/gerard/fichier

Exploration de l’arborescence

1. pwd

La commande **pwd** (print working directory) affiche le chemin absolu du répertoire actuel dans lequel se trouve l’utilisateur :

```
[driss]$ pwd
/home/driss
```

2. cd

La commande **cd** (change directory) permet de changer de répertoire courant. Sur la ligne de commande, on indique en argument le répertoire vers lequel on désire aller :

```
[driss]$ pwd
/home/driss
[driss]$ cd /var/spool
[driss]$ pwd
/var/spool
[driss]$ cd ..
[driss]$ pwd
/var
```

Dans cet exemple la commande permettant de "remonter" vers le répertoire parent est **cd ..** et non **cd..** (sans espace) comme sous DOS : le caractère d’espacement est obligatoire entre la commande et son argument.

En outre, le chemin du fichier (ou du répertoire) en argument peut être absolu, relatif ou personnel ; ceci est valable pour toutes les commandes.

Appelée sans argument, la commande **cd** renvoie l’utilisateur dans son répertoire personnel :

```
[driss]$ pwd
/var
[driss]$ whoami
driss
[driss]$ cd
[driss]$ pwd
/home/driss
```

Enfin, la syntaxe particulière suivante permet de retourner dans le répertoire précédent :

```
[driss]$ pwd
/home/driss
```

```
[driss]$ cd -
/var
[driss]$ pwd
/var
```

Il ne faut pas confondre le répertoire "parent" (..), situé au-dessus du répertoire courant dans l'arborescence Linux, avec le répertoire "précédent", qui correspond au répertoire dans lequel se trouvait l'utilisateur précédemment.

3. ls

La commande **ls** permet de lister le contenu de répertoires. Sans option, l'affichage suit l'ordre alphabétique des noms de fichiers.

Arguments

Appelée sans argument, **ls** se contente d'afficher les fichiers présents dans le répertoire courant :

```
[driss]$ cd
[driss]$ pwd
/home/driss
[driss]$ ls
couleurs  formes  notes
```

Un ou plusieurs chemins peuvent être spécifiés sur la ligne de commande. Si le chemin correspond à un répertoire, les fichiers qu'il contient sont affichés ; si c'est un fichier, c'est le nom de celui-ci qui est affiché :

```
[driss]$ pwd
/home/driss
[driss]$ ls notes . couleurs /home/gerard ~nicolas/pasglop
ls: /home/nicolas/pasglop: No such file or directory
notes

.:
couleurs  formes  notes

couleurs:
bleu  jaune  rose

/home/gerard:
fichier1
```

Lorsque le chemin ne correspond à aucun fichier ou répertoire, la commande retourne une erreur.

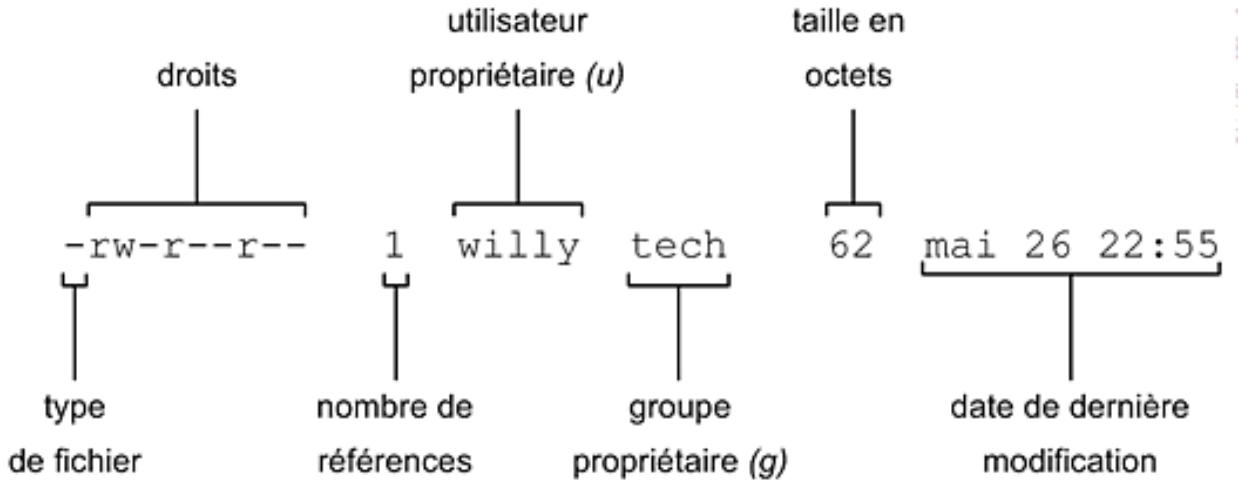
Option -l

L'option **-l** (long) produit un affichage détaillé avec une partie des informations contenues dans l'inode des fichiers (nous le verrons plus loin) :

```
[driss]$ ls -l
total 12
drwxr-xr-x  2 driss tech 4096 jun  1 10:42 couleurs
drwxr-xr-x  2 driss tech 4096 jun  1 10:43 formes
-rw-r--r--  1 driss tech   62 mai 26 22:55 notes
```

La première ligne **total 12** indique que la somme des blocs de données occupés par les fichiers du répertoire est égale à 12.

Les informations concernant le fichier notes dans le répertoire sont :



Le premier caractère indique le type de fichier ; les valeurs possibles sont:

- fichier ordinaire.

d répertoire (directory).

l lien symbolique (link).

b périphérique de type "bloc".

c périphérique de type "caractère".

p fichier tampon ou tube nommé (pipe).

s fichier "socket".

Les droits Linux sur les fichiers seront vu plus tard.

Le nombre de références correspond au nombre de noms de fichiers pointant vers le même inode.

L’utilisateur et le groupe propriétaires sont liés aux droits et seront abordés ultérieurement.

La taille du fichier est exprimée ici en nombre d’octets. Elle correspond à la quantité réelle de données stockées dans le fichier et non à l’espace disque occupé par le fichier en nombre de blocs.

La date affichée correspond à la date de dernière modification du fichier.

Le nom du fichier est, quant à lui, en fin de ligne.

Autres options

L’option **-a** (all) demande à **ls** d’afficher aussi les fichiers cachés (dont le nom commence par un point) :

```
[driss]$ ls
couleurs  formes  notes
[driss]$ ls -a
.  ..  .bash_history  .bash_profile  .bashrc  couleurs  formes  notes
```

-i (inode) affiche, en plus, le numéro d’inode de chaque fichier.

```
[driss]$ ls -i
15874 couleurs  15882 formes  15880 notes
[driss]$ ls -i /bin/gzip /bin/gunzip
32837 /bin/gunzip  32837 /bin/gzip
```

/bin/gzip et /bin/gunzip sont deux noms possibles pour un même fichier.

La notion d’inode sera détaillée plus tard.

-R ("récursif") donne une vue de l’arborescence d’un répertoire en parcourant ses sous-répertoires :

```
[driss]$ pwd
/home/driss
[driss]$ ls -R
.:
couleurs  formes  notes

./couleurs:
bleu  jaune  rose

./formes:
rond  triangle
```

Le terme "récursif" provient du type d’algorithme utilisé dans les programmes qui parcourent des données structurées hiérarchiquement de façon arborescente.

-d affiche le répertoire et non son contenu. Ceci est particulièrement pratique en combinaison avec l’option **-l** lorsqu’il s’agit d’afficher les caractéristiques d’un répertoire :

```
[driss]$ ls couleurs
bleu  jaune  rose
[driss]$ ls -d couleurs
couleurs
[driss]$ ls -ld couleurs
drwxr-xr-x  2 driss tech 4096 jun  1 10:42 couleurs
```

L'option **-t** trie les fichiers suivant leur date de dernière modification ; le dernier fichier modifié dans un répertoire apparaît en premier dans la liste.

Enfin, l'option **-r** inverse l'ordre d'affichage des fichiers.

Toutes ces options peuvent être combinées entre elles afin d'obtenir l'affichage désiré. Par exemple, il peut être utile d'afficher de façon détaillée les fichiers du répertoire /var/log triés du plus vieux au plus récent. On connaît alors le nom et les caractéristiques du dernier fichier journal modifié :

```
[driss]$ ls -ltr /var/log
total 2480
drwxr-xr-x   2 root     root      4096 Apr 25  2008 vbox
drwxr-xr-x   2 root     root      4096 Apr 25  2008 isdn
-rw-r--r--   1 root     root         0 Mar 18 20:43 uucp.log
-rw-r--r--   1 root     root         0 Mar 18 20:43 user.log
drwxr-sr-x   2 news     news      4096 Mar 18 20:43 news
-rw-r--r--   1 root     root         0 Mar 18 20:43 mail.warn
-rw-r--r--   1 root     root         0 Mar 18 20:43 mail.log
-rw-r--r--   1 root     root         0 Mar 18 20:43 mail.info
-rw-r--r--   1 root     root         0 Mar 18 20:43 mail.err
-rw-r--r--   1 root     root         0 Mar 18 22:06 lp-errs
-rw-r--r--   1 root     root         0 Mar 18 22:06 lp-acct
-rw-r--r--   1 root     root        41 Mar 18 22:06 lpr.log
-rw-r--r--   1 root     root     52261 Mar 18 23:09 installer.timings.1
-rw-r--r--   1 root     root    938374 Mar 18 23:09 installer.log.1
drwxrws---   3 root     adm       4096 Mar 18 23:21 webmin
-rw-r--r--   1 root     root      2927 Mar 19 15:27 aptitude
drwxr-xr-x   2 root     root      4096 Mar 22 06:26 apache
-rw-r-----   1 root     adm     465921 Mar 24 06:25 setuid.yesterday
drwxr-s---   2 mail     adm       4096 Mar 26 06:25 exim
-rw-r-----   1 root     adm     465921 Mar 26 06:25 setuid.today
-rw-r-----   1 root     adm        329 Mar 26 06:25 setuid.changes
-rw-r--r--   1 root     root     24024 Apr  3 01:04 faillog
-rw-r--r--   1 root     root      3818 Apr  3 02:17 daemon.log
drwxr-xr-x   2 root     root      4096 Apr 27 08:23 ksymoops
-rw-r--r--   1 root     root      5463 Apr 27 08:23 dmesg
-rw-r--r--   1 root     root     86972 Apr 27 08:23 kern.log
-rw-r--r--   1 root     root      3393 Apr 27 08:23 debug
-rw-r--r--   1 root     root     98415 Apr 27 09:43 messages
-rw-rw-r--   1 root     utmp    292292 Apr 27 09:52 lastlog
-rw-rw-r--   1 root     utmp    131712 Apr 27 09:52 wtmp
-rw-r-----   1 root     adm      41367 Apr 27 09:53 syslog
-rw-r-----   1 root     adm     121593 Apr 27 09:53 auth.log
```

4. file

La commande **ls -l** permet d'obtenir certaines informations sur les fichiers listés mais ne décrit pas leur contenu.

Étant donné que la notion d'extension n'existe pas sous Linux, il est hasardeux de se fier uniquement au nom du fichier pour déterminer le type de son contenu.

La commande **file** affiche cette information :

```
[driss]$ file /etc/passwd /home /bin/ls
/etc/passwd: ASCII text
/home:       directory
/bin/ls:     ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/
Linux 2.6.32, BuildID[sha1]=7f933a4b36714b454508c1ecd366b3bae3fd7e78,
stripped
```

Pour obtenir ces résultats, la commande **file** compare le contenu des fichiers passés en argument avec les descriptions renseignées dans sa base de données. La base de données de **file** se trouve dans le fichier /usr/share/file/magic (voir la page de manuel **man 5 magic**).

5. stat

De la même manière que la commande **ls -l**, l'outil **stat** affiche les informations contenues dans l'inode des fichiers passés en argument :

```
[driss]$ ls -ld /bin/ls /etc/passwd /tmp
-rwxr-xr-x   1 root root 82060 jan 26 19:38 /bin/ls
-rw-r--r--   1 root root  1364 mai 26 23:08 /etc/passwd
drwxrwxrwt  18 root root  4096 jun  1 10:53 /tmp
[driss]$ stat /bin/ls /etc/passwd /tmp
  File: `/bin/ls'
  Size: 82060           Blocks: 176    IO Block: 4096   fichier régulier
Device: 305h/773d       Inode: 32802   Links: 1
Access: (0755/-rwxr-xr-x)  Uid: (  0/  root)   Gid: (    0/    root)
Access: 2008-06-01 11:28:05.000000000 +0200
Modify: 2008-01-26 19:38:12.000000000 +0100
Change: 2008-05-26 07:43:30.000000000 +0200
  File: `/etc/passwd'
  Size: 1364            Blocks: 8      IO Block: 4096   fichier régulier
Device: 305h/773d       Inode: 557308      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2008-06-01 11:28:15.000000000 +0200
Modify: 2008-05-26 23:08:54.000000000 +0200
Change: 2008-05-26 23:08:54.000000000 +0200

  File: `/tmp'
  Size: 4096            Blocks: 8          IO Block: 4096   répertoire
Device: 305h/773d       Inode: 81921       Links: 18
Access: (1777/drwxrwxrwt)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2008-06-01 11:28:02.000000000 +0200
Modify: 2008-06-01 10:53:11.000000000 +0200
```

Change: 2008-06-01 10:53:11.000000000 +0200

Les informations affichées sont :

File	Nom du fichier.
Size	Taille réelle du contenu du fichier en nombre d'octets.
Blocks	Nombre de blocs occupés sur le disque par le fichier.
IO Block	Taille d'un bloc de données en octets sur la partition où se trouve le fichier.
type de fichier	(équivalent à ceux présentés pour la commande **ls -l**).
Access	Droits d'accès au fichier.
Uid	Identifiant de l'utilisateur propriétaire du fichier.
Gid	Identifiant du groupe propriétaire du fichier.
Access	Date de dernière consultation du fichier.
Modify	Date de dernière modification du contenu du fichier (c'est la date affichée avec **ls -l**).
Change	Date de dernière modification de l'inode du fichier ; lors d'un changement des droits associés au fichier par exemple.

Répertoires

1. mkdir

La commande **mkdir** (make directory) permet de créer des répertoires :

```
[driss]$ ls -l
total 12
drwxr-xr-x  2 driss tech 4096 jun  1 10:42 couleurs
```

```
drwxr-xr-x  2 driss tech 4096 jun  1 10:43 formes
-rw-r--r--  1 driss tech   62 mai 26 22:55 notes
[driss]$ mkdir fleurs
[driss]$ ls -l
total 16
drwxr-xr-x  2 driss tech 4096 jun  1 10:42 couleurs
drwxr-xr-x  2 driss tech 4096 jun  1 11:30 fleurs
drwxr-xr-x  2 driss tech 4096 jun  1 10:43 formes
-rw-r--r--  1 driss tech   62 mai 26 22:55 notes
```

Il est possible d'indiquer plusieurs arguments sur la ligne de commande :

```
[driss]$ ls
couleurs  fleurs  formes  notes
[driss]$ mkdir nombres fleurs/vivaces
[driss]$ ls . fleurs
.:
couleurs  fleurs  formes  nombres  notes

fleurs:
vivaces
```

Il faut ajouter l'option **-p** (parents) pour créer une série de répertoires imbriqués :

```
[driss]$ mkdir -p saisons/hiver/janvier
[driss]$ ls -R
.:
couleurs  fleurs  formes  nombres  notes  saisons

./couleurs:
bleu  jaune  rose

./fleurs:
vivaces

./fleurs/vivaces:

./formes:
rond  triangle

./nombres:
./saisons:
hiver

./saisons/hiver:
janvier

./saisons/hiver/janvier:
```

2. rmdir

rmdir (remove directory) permet de supprimer des répertoires. Comme pour la commande **mkdir**, l'option **-p** (parents) permet de supprimer une série de répertoires imbriqués.

L’exécution de **rmdir** est soumise à conditions :

- Le répertoire à supprimer doit être vide (ne comprend plus que les noms de fichiers . et ..).
- Le répertoire à supprimer ou un de ses sous-répertoires ne doivent pas être un répertoire courant de l’utilisateur (ou de tout autre processus) ; si tel est le cas, l’utilisateur se retrouve dans un endroit non défini où il ne peut plus créer de nouveaux fichiers et ce, jusqu’au prochain changement de répertoire avec la commande **cd**.

```
[driss]$ ls
couleurs  fleurs  formes  nombres  notes  saisons
[driss]$ rmdir nombres
[driss]$ ls
couleurs  fleurs  formes  notes  saisons
[driss]$ rmdir -p saisons/hiver/janvier
[driss]$ ls
couleurs  fleurs  formes  notes
[driss]$ rmdir fleurs
rmdir: `fleurs’: Directory not empty
[driss]$ rmdir fleurs/vivaces
[driss]$ rmdir fleurs
[driss]$ ls
couleurs  formes  notes
```

Fichiers

1. touch

La commande **touch** change les dates de dernier accès et de dernière modification d’un fichier existant :

```
[driss]$ ls -l notes
-rw-r--r--  1 driss tech 62 mai 26 22:55 notes
[driss]$ touch notes
[driss]$ ls -l notes
-rw-r--r--  1 driss tech 62 jun  1 11:37 notes
```

Plusieurs options, détaillées dans la page de manuel électronique (man touch), permettent de préciser le nouvel horodatage du fichier.

Lorsqu’on emploie cette commande avec un nom de fichier inexistant en argument, un fichier portant ce nom est créé et ne contient aucune donnée :

```
[driss]$ ls -l
total 12
drwxr-xr-x  2 driss tech 4096 jun  1 10:42 couleurs
drwxr-xr-x  2 driss tech 4096 jun  1 10:43 formes
-rw-r--r--  1 driss tech   62 jun  1 11:37 notes
[driss]$ touch nouvfic
[driss]$ ls -l
```

```
total 12
drwxr-xr-x  2 driss tech 4096 jun  1 10:42 couleurs
drwxr-xr-x  2 driss tech 4096 jun  1 10:43 formes
-rw-r--r--  1 driss tech   62 jun  1 11:37 notes
-rw-r--r--  1 driss tech    0 jun  1 11:38 nouvfic
```

2. cp

La copie de fichiers est effectuée avec le programme **cp** ; la syntaxe principale de cette commande est :

cp [-R] <source ...> <cible>

Il est possible de copier un fichier ordinaire dans un nouveau répertoire ou sous un nouveau nom, ou encore les deux en même temps ; par exemple :

```
[driss]$ ls -R
.:
couleurs  formes  notes  nouvfic

./couleurs:
bleu  jaune  rose
./formes:
rond  triangle
[driss]$ cp notes losange
[driss]$ cp couleurs/jaune formes
[driss]$ cp formes/triangle courbe
[driss]$ ls -R
.:
couleurs  courbe  formes  losange  notes  nouvfic

./couleurs:
bleu  jaune  rose

./formes:
jaune  rond  triangle
```

Attention, suivant la configuration du système (alias prédéfinis), l'écrasement d'un fichier existant lors d'une copie ne génère ni erreur, ni message d'alerte.

***cp** autorise la copie de plusieurs fichiers en même temps. Dans ce cas, la cible est obligatoirement un répertoire* :

```
[driss]$ cp notes losange formes
[driss]$ ls -R
.:
couleurs  courbe  formes  losange  notes  nouvfic

./couleurs:
bleu  jaune  rose

./formes:
jaune  losange  notes  rond  triangle
```

Enfin, l'option **-R** ("récursif") permet de spécifier un répertoire comme argument source ; sans celle-ci, **cp** ignore les répertoires lors de la copie :

```
[driss]$ cp notes formes couleurs
cp: omission du répertoire `formes'
[driss]$ ls -R
.:
couleurs  courbe  formes  losange  notes  nouvfic

./couleurs:
bleu  jaune  notes  rose

./formes:
jaune  losange  notes  rond  triangle
[driss]$ cp -R formes couleurs
[driss]$ ls -R couleurs
couleurs:
bleu  formes  jaune  notes  rose

couleurs/formes:
jaune  losange  notes  rond  triangle
```

Autres options

L'option **-i** (ou **--interactive**) interroge l'utilisateur avant d'écraser un fichier déjà existant. À l'inverse, l'option **-f** (ou **--force**) écrase les fichiers cibles sans poser de question. Si les deux options sont employées simultanément, **cp** écrase les fichiers cibles sans interroger l'utilisateur.

```
driss]$ ls notes couleurs
notes

couleurs:
bleu  formes  jaune  notes  rose
[driss]$ cp notes couleurs
[driss]$ cp -i notes couleurs
cp: écraser `couleurs/notes'?o
[driss]$ cp -if notes couleurs
```

-d (ou **--no-dereference**) permet de copier les liens en tant que tels plutôt que de copier les fichiers vers lesquels ils pointent.

Avec l'option **-p** (ou **--preserve**), la copie conserve le propriétaire, le groupe, les droits d'accès et les horodatages du fichier original. Cela n'est valable que si l'utilisateur est connecté en tant qu'administrateur (**root**).

```
[driss]$ ls -l notes
-rw-r--r--  1 driss tech 62 jun  1 11:37 notes
[driss]$ cp notes notes.2
[driss]$ ls -l notes notes.2
-rw-r--r--  1 driss tech 62 jun  1 11:37 notes
-rw-r--r--  1 driss tech 62 jun  1 12:06 notes.2
[driss]$ cp -p notes notes.3
[driss]$ ls -l notes notes.2 notes.3
-rw-r--r--  1 driss tech 62 jun  1 11:37 notes
-rw-r--r--  1 driss tech 62 jun  1 12:06 notes.2
```

```
-rw-r--r--  1 driss tech 62 jun  1 11:37 notes.3
```

Enfin, l'option **-a** (ou **--archive**) est équivalente aux options **-dpR**. Elle est principalement utilisée par l'administrateur pour sauvegarder, à l'identique, des répertoires.

3. rm

La commande **rm** (remove) efface les fichiers donnés en argument. Cette suppression est définitive. Il n'y a aucun mécanisme de corbeille mis en œuvre par défaut sous Linux.

```
[driss]$ ls -R
.:
couleurs  courbe  formes  losange  notes  notes.2  notes.3  nouvfic

./couleurs:
bleu  formes  jaune  notes  rose

./couleurs/formes:
jaune  losange  notes  rond  triangle

./formes:
jaune  losange  notes  rond  triangle
[driss]$ rm notes.2 notes.3 nouvfic couleurs/notes formes/notes
[driss]$ ls -R
.:
couleurs  courbe  formes  losange  notes

./couleurs:
bleu  formes  jaune  rose

./couleurs/formes:
jaune  losange  notes  rond  triangle

./formes:
jaune  losange  rond  triangle
```

À l'instar de **cp**, la commande **rm** requiert l'option **-R** (ou **-r**) pour la suppression de répertoires :

```
[driss]$ ls -l couleurs
total 16
-rw-r--r--  1 driss tech   30 jun  1 10:42 bleu
drwxr-xr-x  2 driss tech 4096 jun  1 12:03 formes
-rw-r--r--  1 driss tech  172 jun  1 10:42 jaune
-rw-r--r--  1 driss tech   54 jun  1 10:42 rose
[driss]$ rm couleurs/formes
rm: ne peut enlever `couleurs/formes': Is a directory
[driss]$ rm -R couleurs/formes
[driss]$ ls -l couleurs
total 12
-rw-r--r--  1 driss tech  30 jun  1 10:42 bleu
-rw-r--r--  1 driss tech 172 jun  1 10:42 jaune
-rw-r--r--  1 driss tech  54 jun  1 10:42 rose
```

Ainsi, il est maintenant possible de supprimer des répertoires non vides ; ce que ne sait pas faire la commande **rmdir**.

Options

Comme pour la commande **cp**, il existe les options **-i** (**--interactive**) et **-f** (**--force**) permettant respectivement d'interroger ou non l'utilisateur lors de l'effacement des fichiers et des répertoires.

4. mv

La commande **mv** (move) permet de déplacer et de renommer des fichiers ou répertoires.

À la différence de **cp** et **rm**, cette commande sait traiter des répertoires sans option supplémentaire.

La syntaxe de cette commande est :

mv <source ...> <cible>

La source peut être aussi bien un fichier ordinaire qu'un répertoire ; ce fichier :

- est déplacé dans le répertoire cible si celui-ci existe.
- est renommé sous le nom de la cible si elle n'existe pas.
- écrase le fichier cible si celui-ci existe.

```
[driss]$ ls -R
.:
couleurs  courbe  formes  losange  notes

./couleurs:
bleu  jaune  rose

./formes:
jaune  losange  rond  triangle
[driss]$ mv losange carre
[driss]$ mv notes couleurs/vert
[driss]$ mv formes/jaune couleurs
[driss]$ ls -R
.:
carre  couleurs  courbe  formes
./couleurs:
bleu  jaune  rose  vert

./formes:
losange  rond  triangle
```

Lorsqu'il y a plusieurs fichiers ou répertoires sources, la cible est alors le

répertoire qui les accueillera :

```
[driss]$ mv carre courbe formes
[driss]$ ls -R
.:
couleurs  formes

./couleurs:
bleu  jaune  rose  vert

./formes:
carre  courbe  losange  rond  triangle
```

Options

De nouveau, les options **-i** (**--interactive**) et **-f** (**--force**) permettent d'interroger ou non l'utilisateur lors de l'écrasement de fichiers.

Consultation de fichiers

1. cat

cat affiche à l'écran le contenu des fichiers passés en argument sur la ligne de commande :

```
[nicolas]$ cat /etc/motd
Bienvenue sous Linux
Vous êtes connecté sur la machine doe.reso.local
```

2. more, less

La commande Unix **more** permet aussi l'affichage du contenu de fichiers texte à l'écran. Par contre, son affichage se fait page par page ; les touches du clavier utiles avec **more** sont :

[Entr]	Fait défiler le texte ligne par ligne.
[Espace]	Fait défiler le texte page par page.
[q]	Quitte le programme.

Cet exemple montre l'indicateur présent en bas d'écran :

```
[nicolas]$ more /etc/services
ssh             22/udp                  # SSH Remote Login Protocol
telnet          23/tcp
telnet          23/udp                  # 24 - private mail system
smtp            25/tcp      mail
smtp            25/udp      mail
time            37/tcp      timserver
time            37/udp      timserver
```

```
rlp             39/tcp      resource    # resource location
rlp             39/udp      resource    # resource location
nameserver      42/tcp      name        # IEN 116
nameserver      42/udp      name        # IEN 116
nicname         43/tcp      whois
nicname         43/udp      whois
tacacs          49/tcp                  # Login Host Protocol (TACACS)
tacacs          49/udp                  # Login Host Protocol (TACACS)
re-mail-ck      50/tcp                  # Remote Mail Checking Protocol
re-mail-ck      50/udp                  # Remote Mail Checking Protocol
domain          53/tcp                  # name-domain server
domain          53/udp
whois++         63/tcp
--Encore--(10%)
```

Il existe, sous Linux, une commande similaire, plus évoluée que **more**, appelée **less**. En plus des touches précédentes, elle accepte aussi :

[Haut]	Remonte d'une ligne.
[Bas]	Descend d'une ligne.
[Page préc]	Remonte d'une page.
[Page suiv]	Descend d'une page.

L'emploi des commandes **more** et **less** est adapté à l'affichage de fichiers texte longs.

3. od, strings

La commande **od** (octal dump) affiche le contenu d'un fichier en octal ou sous d'autres formats suivant les options ; l'option **-x** produit une sortie en hexadécimal, par exemple :

```
[nicolas]$ cat /etc/motd
Bienvenue sous Linux
Vous êtes connecté sur la machine doe.reso.local
[nicolas]$ od /etc/motd
0000000 064502 067145 062566 072556 020145 067563 071565 046040
0000020 067151 074165 053012 072557 020163 072352 071545 061440
0000040 067157 062556 072143 020351 072563 020162 060554 066440
0000060 061541 064550 062556 062040 062557 071056 071545 027157
0000100 067554 060543 005154
0000106
[nicolas]$ od -x /etc/motd
0000000 6942 6e65 6576 756e 2065 6f73 7375 4c20
0000020 6e69 7875 560a 756f 2073 74ea 7365 6320
0000040 6e6f 656e 7463 20e9 7573 2072 616c 6d20
0000060 6361 6968 656e 6420 656f 722e 7365 2e6f
0000100 6f6c 6163 0a6c
0000106
```

Ce genre d’affichage est utilisé principalement par les programmeurs pour examiner le contenu de fichiers binaires.

Du même type, la commande **strings** affiche les chaînes de caractères lisibles contenues dans les fichiers binaires ; cela peut se révéler utile pour un programmeur ou un administrateur système en cas de débogage :

```
[nicolas]$ strings /bin/ls
/lib/ld-linux.so.2
libtermcap.so.2
_DYNAMIC
_init
tgetent
_fini
_GLOBAL_OFFSET_TABLE_
_Jv_RegisterClasses
tgetstr
...
```

Exercices

Exercice 1

Téléchargez une image de votre choix depuis Internet puis déplacez-la dans un sous-répertoire de /tmp.

Exercice 2

Supprimez le fichier et le sous-répertoire créés dans l’exercice précédent.

ASTUCES

La commande man permet d'accéder à la documentation de la commande passée en paramètre (ex : man ls). Elle permet d'obtenir une description de son fonctionnement et des arguments utilisables.

Les flèches haut et bas permettent de parcourir l'historique des commandes, et d'accéder ainsi aux commandes tapées précédemment.

La touche tabulation permet d'utiliser l'auto-complétion. Elle permet de compléter automatiquement les commandes et les chemins en ne tapant que les premières lettres.

Exercice 3

Lancer un émulateur de terminal, qui vous permettra d'accéder à l'interpréteur decommande bash.

1. Dans quel répertoire vous trouvez-vous ?

2. Listez tous les fichiers du répertoire personnel (y compris les fichiers caché) de l'utilisateur courant en affichant les informations telles que :

 le type du fichier ;

 les droits ;

 le nom du propriétaire ;

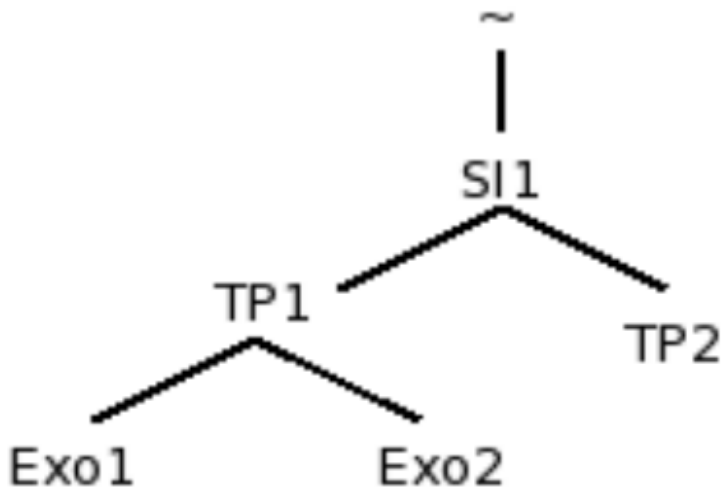
 la taille du fichier en Ko ;

3. Rendez vous dans le répertoire racine en utilisant un chemin relatif.

4. Listez le contenu du répertoire courant.

5. Retourner dans votre répertoire personnel en utilisant le caractère ~.

6. Reproduisez l'arborescence ci-contre :



SI1, TP1, Exo1 et Exo2 sont des répertoires.

7. Créez le fichier file.txt dans le répertoire Exo1

8. Utiliser la commande nano afin d'éditer le contenu de file.txt.

9. Saisir « Hello world » dans le fichier et enregistrez le avant de quitter.

10.Faites une copie de sauvegarde du fichier file.txt, nommée file.txt.bak

11.Créez 3 dossiers nommés test1,test2 et test3 dans le répertoire ~/TP1/Exo1.

12.Déplacez le fichier file.txt dans le répertoire test1 en utilisant un chemin absolu.

13.Copiez le fichier file.txt.bak dans le répertoire test2 en le renommant en file.txt.

14.Supprimez le dossier test3 en utilisant exclusivement la commande rmdir.

15.Supprimez les dossiers test1 et test2 en utilisant une seule commande.

EXERCICE 4

La commande find vous permet de recherchez des fichiers dans le système de fichiers. Elle permet aussi d'effectuer des actions sur les fichiers trouvés. (faites quelques recherches sur cette commande et ses options)

1. Cherchez l'ensemble des fichiers de votre répertoire personnel ayant l'extension « .h ».

2. Cherchez l'ensemble des fichiers de « /dev » ne se terminant pas par un chiffre.

3. Cherchez et supprimez maintenant l'ensemble des fichiers de votre répertoire personnel contenant « file » en demandant à l'utilisateur de confirmer chaque suppression (sans passer par rm)