À l'issue de ce chapitre, le lecteur sera capable de se déplacer dans l'arborescence Linux ainsi que de créer, déplacer et supprimer des fichiers et des répertoires.

A. 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 2004 dans le fichier nommé *notes.histoire.classe.6A.trim.1.an.2004* 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 ; ceux-ci seront détaillés dans le chapitre consacré au shell Bash. 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 dans cet ouvrage.

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.

B. 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.

C. Chemins

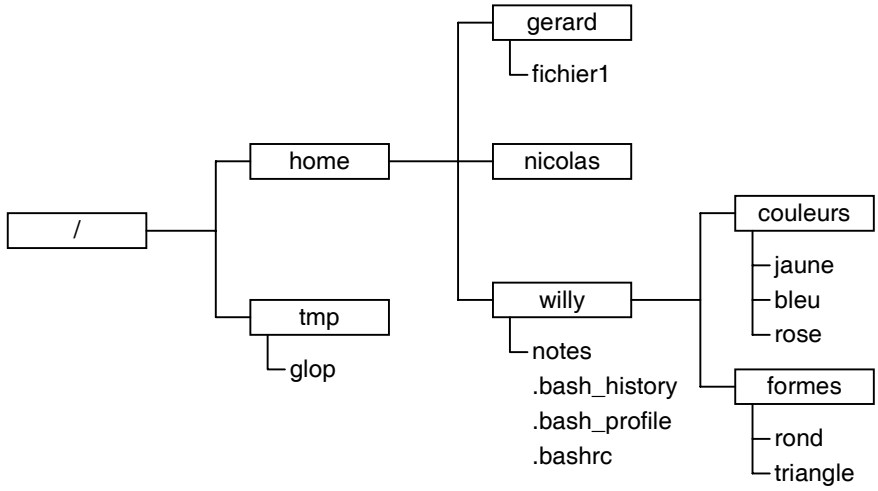
Connaître le nom d'un fichier (ou d'un répertoire) n'est pas suffisant pour accéder à celui-ci ; un même nom peut être en effet associé à plusieurs fichiers dans des répertoires différents.

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/willy/notes`.

Par exemple :

```
[willy]$ ls /home/willy/notes
/home/willy/notes
```

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	willy/notes
/home/willy	./notes
/home/willy/couleurs	*../notes*
/	home/willy/notes
/home	*../tmp/./../home/./willy/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 :

```
[willy]$ pwd
/home/willy
[willy]$ ls notes
notes                                                    .../...
```