Système de fichiers

Principes généraux

Sous UNIX quasiment tout est accessible grâce à la notion de fichier : des données stockées sur le disque dur jusqu'aux périphériques de l'ordinateur. La notion de fichier est donc centrale dans le fonctionnement du système d'exploitation.

Unix expose différents *types* de fichier à l'utilisateur parmi lesquels on retient notamment :

- un **répertoire** [1] (*directory* en anglais) contient d'autres fichiers,
- un **lien symbolique** (*symbolic link*) est un raccourci vers un autre fichier,
- un **fichier régulier** (*regular file*) contient des données dont la nature n'est, a priori, pas spécifiée ou connue par le système.

En imposant que tout fichier soit contenu dans un répertoire, Unix offre une vue arborescente du rangement des données. Le répertoire racine (*root* en anglais) de cette hiérarchie est identifiée par le nom `/`.

Identification par les chemins

Tous les fichiers sont identifiables par un chemin (*path*).

Les fichiers autres que le répertoire `/` sont **identifiés de manière canonique**, par la liste, la plus courte possible, des répertoires à traverser depuis la racine pour les atteindre suivie d'un nom. Cet identifiant est nommé le **chemin absolu** (*absolute path*) du fichier. Il est unique.

Unix permet cependant à un même fichier d'avoir plusieurs noms, dans des répertoires quelconques. Un répertoire, par exemple, contient toujours au moins deux autres fichiers : `.` qui est un surnom pour le répertoire lui-même, `..` qui est un surnom du répertoire qui le contient. Il est ainsi possible d'identifier un fichier par un **chemin relatif**. Pour cela on n'utilise pas la liste des répertoires à traverser depuis la racine mais la liste des répertoires à traverser depuis un répertoire quelconque (qui peut contenir les surnoms `..` par exemple). Cette liste peut, par ailleurs, être vide.

Dans un chemin :

- les différents répertoires traversés, s'il y en a, sont séparés les uns des autres par le caractère `/` ;
- on distingue la partie répertoire (*dirname* : tout ce qui est avant le dernier `/`) du nom de base (*basename* : ce qui est après le dernier `/`) ;
- les noms de bases peuvent généralement comporter entre 1 et 1024 caractères quelconque, sauf le caractère `/`. Unix fait une différence entre les lettres en capitales et les lettres en minuscules, de telle sorte que `toto` et `Toto` représentent deux chemins vers des fichiers différents.

À tout moment il existe un répertoire de travail courant (*current working directory*) dans lequel Unix considère être positionné.

Il est possible de spécifier un chemin en ne précisant que le nom de base d'un fichier. Dans ce cas Unix considère le chemin relatif à ce répertoire courant.

Conventions

La plupart des commandes considèrent que les fichiers dont le nom de base débute par le caractère `.` sont cachés et ne les traitent pas directement.

Un lien symbolique est un fichier qui **contient** un chemin vers un autre fichier. Quand elles accèdent à un tel fichier, la plupart des commandes ne traitent pas le fichier lui même mais le fichier identifié par le chemin qu'il contient. Pour être valide un lien symbolique doit donc contenir un chemin absolu ou un chemin **relatif au répertoire dans lequel il est contenu**.

De nombreuses commandes permettent de manipuler les fichiers via le langage de commandes (*shell*). Elles utilisent, et attendent donc, des chemins pour identifier les fichiers qu'elles doivent gérer. Le tableau 1 donne une liste de certaines de ces commandes. Les pages du manuel en ligne vont en donneront la documentation.

1. On dit aussi *catalogue* ou *dossier*

Commandes	Utilité
`pwd`	afficher le chemin absolu du répertoire de travail courant
`cd`	changer le répertoire de travail courant
`ls`	lister des fichiers
`rm`	supprimer des fichiers
`mkdir`	créer des répertoires
`rmdir`	supprimer des répertoires vides
`od`	afficher les octets d'un fichier sous différents formats
`cat`	afficher le contenu de fichiers
`stat`	afficher les caractéristiques de fichiers
`file`	déterminer la convention de structure du contenu du fichier (son type *applicatif*)
`cp`	copier (dupliquer) des fichiers
`mv`	déplacer (renommer) des fichiers
`ln`	surnommer ou créer un *lien symbolique* vers un fichier

TABLE 1 – Commandes de manipulations du système de fichiers

Exercices

Dans les exercices de ce TP
- ▷ **respectez scrupuleusement les consignes qui vous sont données,**
- ▷ **utilisez uniquement `vi` pour créer ou éditer des fichiers réguliers,**
- ▷ **n'utilisez la commande `cd` que lorsque cela est explicitement autorisé par ⇐ `cd` en marge de la question.**

Exercice 1 : Manipulation du système de fichiers

Q 1. Si ça n'est pas déjà fait, créez un répertoire `asr/tp6` dans votre répertoire principal.

Q 2. Déplacez-vous dans le répertoire créé. ⇐ `cd`

Q 3. Depuis ce répertoire créez la hiérarchie de répertoires présentée ci-dessous en considérant que

- tous les fichiers dont le *basename* est doté d'une majuscule (initiale, capitale) sont des fichiers réguliers,
- les fichiers réguliers ont comme contenu le *basename* du fichier écrit en caractères minuscules.

```
tp6
'-- fs
    |-- a-faire
    |   |-- Algo2
    |   |-- Anglais
    |   '-- Math
    '-- fait
        |-- Algo
        '-- Gestion
```

Q 4. Mettez dans un fichier nommé `reponses` dans le répertoire `asr/tp6` les réponses (**une par ligne**) aux questions suivantes :

1. placez-vous dans le répertoire `a-faire` et donnez la commande que vous avez exécutée pour cela, ⇐ `cd`
2. donnez le chemin absolu du fichier `Gestion`,
3. donnez le chemin relatif le plus court vers `Algo`,
4. donnez le chemin relatif le plus court vers `Anglais`.

Q 5. Vérifiez que les chemins que vous avez spécifiés sont corrects en utilisant la commande `cat` pour visualiser leurs contenus.

Q 6. Réalisez la suite d'opérations demandées en vous assurant :

- de ne faire qu'une seule commande par opération demandée,
- d'ajouter la commande exécutée sur une ligne (**et une seule**) dans le fichier `reponses`,
- de ne jamais vous déplacer (**ne jamais utiliser la commande** `cd`).

1. copiez le fichier `Algo2` dans le fichier `Algorithmique`,
2. copiez le fichier `Algo2` dans le fichier `Algo` du répertoire `fait`,
3. renommez le fichier `Anglais` en `English`,
4. déplacez le fichier `English` dans le répertoire `fait`,
5. faites en sorte que le fichier `Math` s'appelle également `Abandon` dans le répertoire `fait` (on le surnomme),
6. éditez le fichier `Abandon` et y **ajouter** le mot `regret` sur sa seconde ligne,
7. visualisez le contenu du fichier `Math`,
8. créez un lien **symbolique** nommé `Persevere` pointant sur le fichier `Abandon`,
9. visualisez le contenu du fichier `Persevere`,
10. supprimez le fichier `Abandon`,
11. affichez le contenu du fichier `Persevere`,
12. affichez le contenu du fichier `Math`,
13. créez le répertoire `plustard`.

Q 7. Quelle commande permet de faire en sorte que le fichier `Math` s'appelle également (*i.e.* soit surnommé) `Sauvegarde` dans le répertoire `/tmp`. Ajoutez cette commande dans votre fichier `reponses`.

Q 8. Expliquez le problème posé par la question précédente puis proposez une autre solution. Ajoutez cette commande à votre fichier `reponses`.

Q 9. Déterminez le nombre de noms du répertoire `a-faire` et à quoi ils correspondent.