TP1 : Le système de gestion de fichiers d'Unix

1 Rappel sur l'organisation des données

Il existe différents types de fichiers, dont les principaux sont :
- les fichiers ordinaires, qui contiennent des données (texte brut, données formatées pour être utilisables par certaines applications, fichiers exécutables...),
- les répertoires : ils permettent d'organiser l'espace mémoire disponible. Ils contiennent des fichiers de données, des fichiers spéciaux ou encore des sous-répertoires. On obtient ainsi une structure hiérarchique,
- les fichiers spéciaux /dev/ : ils permettent de gérer les périphériques (imprimante, lecteur de disquette...).

Le rôle des répertoires est primordial puisqu'ils permettent de « ranger » les fichiers : on peut comparer les répertoires à des boîtes dans lesquelles on peut classer des objets, i.e les fichiers, et aussi y stocker d'autres boîtes dans lesquelles sont classés d'autres objets. On obtient ainsi une *structure arborescente*, dans laquelle il est primordial de pouvoir désigner précisément les fichiers. Leur simple nom peut être insuffisant, puisque deux répertoires distincts peuvent contenir deux fichiers portant le même nom sans pour autant être identiques. Un fichier est donc désigné par son *chemin d'accès*, c'est-à-dire la succession de répertoires à parcourir pour parvenir jusqu'à son emplacement
- à partir de la racine de l'arborescence, `/`, pour le chemin *absolu*,
- à partir du répertoire courant pour le chemin *relatif*.

Dans l'énoncé du chemin, les répertoires successifs sont séparés par des `/`. Le répertoire courant est désigné par `.`, son père par `..`, et le répertoire par défaut de l'utilisateur par `~`. Par exemple, lorsque le répertoire courant est `/usr/local`, le chemin absolu décrivant le fichier `toto` du sous-répertoire `games` est `/usr/local/games/toto`. Mais ce fichier peut également être nommé de façon relative `games/toto`, ou `./games/toto`, ou encore `../local/games/toto`.

2 Parcourir l'arborescence

Quelques commandes de base :
- `pwd` indique quel est le répertoire courant.
- `ls [`*`rep`*`]` permet de lister le contenu du répertoire *rep* (ou, par défaut, du répertoire courant), en donnant plus ou moins d'indications selon les options choisies.
- `cd [`*`rep`*`]` change le répertoire courant. Sans argument, retourne à votre répertoire par défaut.
- `cat [`*`fichier`*`]` affiche le contenu de *fichier*.

Remarque : la mention d'un argument entre crochets (`[`*`rep`*`]` par exemple) signifie que cet argument est optionnel. Utilisez `man` *commande* ou `info` *commande* pour obtenir les pages de description de ces commandes.

Essayez de repérer l'emplacement de votre répertoire par défaut dans l'arborescence à l'aide de `pwd`. Regardez ensuite ce qui se trouve dans le répertoire père et dans le répertoire racine. Consulter également le contenu de votre répertoire principal.

3 Créer sa propre sous-arborescence

- `mkdir` ***rep*** permet de créer un répertoire ***rep***, repéré par son chemin relatif ou absolu.
- `rmdir` ***rep*** permet de supprimer un répertoire vide.

Exercice :

- Créez un répertoire `IF241`.
- En restant dans votre répertoire principal, créez, dans le répertoire `IF241`, un sous-répertoire `Unix`.
- Placez-vous dans le répertoire `Unix`.
- Dans le répertoire `Unix`, créez un répertoire `TD`.
- En restant dans le répertoire `Unix`, créez un répertoire `CPP` et un répertoire `Java` dans le répertoire `IF241`.
- Placez-vous dans le répertoire `Java` et affichez précisément tout ce qu'il contient (il devrait contenir deux répertoires).
- Placez-vous dans le deuxième de ces deux répertoires et déterminez le répertoire courant.
- Replacez-vous dans le répertoire `Java`, puis allez dans le premier des deux répertoires de la liste établie précédemment.
- Déterminez le répertoire courant.
- Revenez directement dans votre répertoire principal.
- De là, effacez le répertoire `Java`.

4 Manipulation simple des fichiers

- `cp` ***toto tutu*** crée un fichier ***tutu*** identique au fichier ***toto***.
- `mv` ***toto tutu*** renomme (déplace) le fichier ***toto*** en ***tutu***.
- `rm` ***toto*** efface le fichier ***toto***.

Lisez les pages de description de ces commandes. Faites particulièrement attention à l'option `-i`, fortement recommendée.

Exercice :

- Placez-vous dans le répertoire `/etc` et copiez le fichier `fstab` dans votre répertoire `IF241`.
- Revenez dans le répertoire `IF241` et renommez le fichier `fstab` en `table`.
- Déplacez `table` dans le répertoire `TD` (tout en restant dans `IF241`).
- Déplacez le répertoire `TD` dans le répertoire courant.
- Affichez le contenu du répertoire `IF241` et de tous ses sous-répertoires.

- Renommez le répertoire `TD` en `TP`.
- Faites une copie de `table` dans `IF241`.
- Effacez `table` de votre répertoire `TP` sans changer de répertoire courant.
- Effacez le répertoire `TP`.

5 Les droits d'accès

Ils sont de trois types : lecture (`r`), écriture (`w`) et exécution (`x`). Les utilisateurs sont répartis en trois niveaux, le propriétaire du fichier (`u`), son groupe (`g`), et les autres (`o`). L'option `-l` de `ls` indique, entre autres choses, quels sont les droits pour chaque fichier. Ces droits peuvent être modifiés à l'aide de la commande `chmod`.

- Déterminez à quel groupe vous appartenez.
- Créez un fichier `mode` quelconque, par exemple à l'aide de l'éditeur de textes `emacs`.
- Déterminez quels sont, par défaut, les droits sur un de vos fichiers.
- Changez les droits d'accès sur le fichier `mode` afin d'obtenir les droits suivants : `rw-rwx--x`.
- Effectuez les mêmes manipulations sur votre répertoire `IF241`. Que se passe-t-il alors ? Mettez ensuite les droits : `-wx------`. Qu'en concluez-vous ? Remettez maintenant les droits initiaux.
- Que se passe-t-il si sur un fichier vous avez les droits d'écriture et non pas les droits de lecture ? Faites des essais et voyez ce que vous pouvez faire.
- Essayez de modifier les droits sur le répertoire /dev. Que se passe-t-il ? Pourquoi ?

6 Redirection d'entrées – sorties.

Redirection de sortie : le symbole `>`

- Exécutez `ls` dans une fenêtre de terminal. Taper ensuite `ls > fichier`, puis réexécutez `ls`. Vous remarquez qu'un nouveau fichier appelé `fichier` a été créé. **Lisez son contenu**.
- Exécutez maintenant la commande `ls -l > fichier`. Consultez à nouveau le contenu du fichier `fichier`.
- Enfin, exécutez la commande `ls >> fichier`. Consultez `fichier`.

Concluez sur le rôle de `>` et `>>`.

Redirection d'une entrée : le symbole `<` Placez vous dans votre répertoire `IF241`. Exécutez `wc table` ; que fait cette commande ? Exécutez maintenant `wc < table`. Quelle est la différence ?

Exercice :

Comment mettre le résultat de la commande `wc < table` dans un fichier nommé `sortie` ?

Le canal d'erreur Essayez d'exécuter la commande `wc trtrer`. Ce fichier n'existe pas!!! Ce message est affiché non sur le canal de sortie mais sur le *canal d'erreur*. Pour voir la différence, essayez de rediriger ce message dans un fichier nommé `erreur` par la commande `wc trtrer > erreur`. Consultez le contenu du fichier `erreur`. Puis exécutez la commande `wc trtrer >& erreur` et observez le résultat.

7 Les liens

La commande `ln` ***toto tutu*** permet de créer un lien appelé ***tutu*** vers le fichier ***toto***. L'option `-s` permet de créer un lien symbolique.

L'option `-i` de `ls` permet d'afficher les numéros d'*i*node des fichiers. Quant à l'option `-l`, elle affiche, entre autres choses, le nombre de liens pointant sur chaque fichier.

Exercice :

- Créez un fichier `test` contenant la ligne `Ceci est un fichier de test`. Créez une copie de ce fichier, `test1`, puis un lien `test2` et un lien symbolique `test3` sur `test`.
- Comparez les numéros d'*i*node de ces quatre fichiers. Est-ce cohérent ?
- Effacez `test`, puis afficher `test1`, `test2` et `test3`. L'action de `rm` consiste-t-elle réellement à effacer les fichiers ?
- Créez un nouveau fichier `test` contenant le texte `Deuxième fichier de test`. Afficher à nouveau le contenu des quatre fichiers. Tout vous semble-t-il normal ?