


Introduction aux systèmes d'exploitation Math-Info

TP n° 2 : organisation du système de fichiers

Pour ce TP vous devez télécharger l'archive `arborescence_TP2.tar` qui se trouve sur Moodle, dans les documents du cours IS1MI, puis le désarchiver dans votre répertoire `~/Cours/IS1` grâce à la commande « `tar xf arborescence_TP2.tar` ».

Modalités de rendu Au cours du TP, collectez dans un fichier texte appelé `reponses_TP2.txt` les réponses aux questions marquées par le symbole , ou les commandes utilisées pour y répondre (mais pas les résultats produits par les commandes). Vous insérerez dans ce fichier vos nom(s) et prénom(s) et vous le déposerez en fin de TP sur Moodle.

Un fichier peut être désigné de façon non ambiguë en utilisant deux types de *chemins* :




- un *chemin absolu* décrit un chemin dans l'arborescence du système de fichiers depuis sa racine (« / ») jusqu'au fichier ;
- un *chemin relatif* (au répertoire courant) décrit un tel chemin depuis le répertoire de travail courant de l'utilisateur.

Ces chemins sont représentés par la suite des noms (de base) des répertoires traversés, séparés par le symbole « / », et terminés par le nom de base du fichier.

Le répertoire parent d'un répertoire peut être indiqué par la notation « `..` ». Le répertoire lui-même peut être indiqué par « `.` ». Par exemple, le chemin « `/home/foo/enseignement` » est équivalent à « `/home/foo/./enseignement` », mais aussi, si ce répertoire contient un sous-répertoire « `islmi` », au chemin « `/home/foo/enseignement/islmi/..` ».

En outre, dans le *shell*, la tilde « `~` » utilisée comme première composante d'un chemin est interprétée comme le répertoire *home* de l'utilisateur ; par exemple, pour un utilisateur dont le *home* est « `/home/foo` », le chemin « `~/enseignement` » est interprété comme « `/home/foo/enseignement` ». De même, la syntaxe « `~foo` » est interprétée comme le *home* de l'utilisateur « `foo` ».

Exercice 1 – consulter d'un coup toute une arborescence

1.  Trouver une option de « `ls` » qui permet de lister les fichiers ordinaires et sous-répertoires contenus dans le répertoire `~/Cours/IS1/TP2` en déterminant leur type.
2.  Trouver l'option de « `ls` » qui permet d'afficher le répertoire `~/Cours/IS1/TP2` sans afficher son contenu.
3.  Trouver l'option de « `ls` » qui permet de lister, en une seule ligne de commande, tout le contenu de l'arborescence de racine `~/Cours/IS1/TP2`.

Exercice 2 – chemin relatif, chemin absolu

1. 🚧 En utilisant la commande « `cat` » depuis le répertoire `~/Cours/IS1`, afficher le contenu du (seul) fichier de l'arborescence TP2 dont le nom de base est `haddock` en utilisant trois chemins différents pour le désigner.
2. 🚧 Même question depuis le répertoire `~/Cours/IS1/TP2/Duck/CoffreDePicsou`.
3. 🚧 Déterminer tous les répertoires de travail dans lesquels la commande `cat ../Personnages/haddock` s'exécute sans erreur (et vérifier).
4. 🚧 Quelle commande permet de lister le contenu du répertoire `~/Cours/IS1/TP2/Tintin/Vehicules` depuis le répertoire `~/Cours/IS1/TP2/Toto` ?

Exercice 3 – fichiers cachés

Malgré les apparences, le répertoire `~/Cours/IS1/TP2/Duck/CoffreDePicsou` contient plusieurs fichiers.

🚧 Chercher l'option de « `ls` » qui permet de les afficher. Comment s'appellent-ils ?

Exercice 4 – références et addresses web (URL)

1. Ouvrir un navigateur (firefox, chrome, ...). Dans la barre d'URL, saisir « `file://` » puis la référence absolue de votre répertoire personnel. Qu'affiche le navigateur ? Essayer ensuite de faire de même avec le répertoire personnel de votre voisin. Quel est le résultat ?
2. 🚧 Considérer l'URL « `http://www.irif.fr/~gio/teaching/2020-21/is1mi` ». Quelle est la référence plus longue contenue dans cette URL ?

Fonctionnalités utiles du shell Il arrive parfois que l'on ait à utiliser une même commande plusieurs fois, ou que l'on souhaite corriger une commande tapée précédemment. À cette fin, un ensemble de « raccourcis » permet d'accéder à l'historique des commandes. En utilisant les flèches haut et bas, on peut faire défiler les commandes précédentes, de la plus récente à la plus ancienne. La combinaison de touches `Ctrl-r` permet de rechercher une commande dans l'historique qui contient un certain mot.

Avec les flèches gauche et droite, vous pouvez également corriger un détail de chaque ligne de commande. Les combinaisons de touches `Ctrl-a` et `Ctrl-e` permettent respectivement de repositionner le curseur au début ou à la fin de la ligne. `Ctrl-k` permet de supprimer la fin de la ligne (au-delà du curseur).

Autre point utile pour gagner du temps de saisie : la **touche de tabulation** permet de laisser le soin au shell de terminer les noms de fichiers lorsqu'il n'y a pas d'ambiguïté.

Exercice 5 – chercher dans l'historique

En utilisant l'historique, chercher la dernière fois que vous avez exécuté la commande « cat » pour afficher le contenu du fichier dont le nom de base est haddock.

Utilisez-la pour afficher à nouveau le contenu du fichier à partir de votre répertoire courant. Si nécessaire, vous éditez donc la commande en ligne, pour corriger la référence relative du fichier.

Exercice 6 – modifier d'un coup toute une arborescence

1. ➤ Copier, en une seule ligne de commande, le répertoire ~/Cours/IS1/TP2/Tintin (et toute l'arborescence en dessous) sous le nom ~/Cours/IS1/TP2/Sauvegarde.

« diff » (**differences**) compare les fichiers ou répertoires et affiche les différences ligne par ligne, respectivement fichier par fichier.

2. ➤ Vérifier que le fichier dont le nom de base est milou a bien été copié.
3. ➤ En utilisant une option de « diff », vérifier que l'arborescence de Sauvegarde est bien une copie de Tintin.
4. Supprimer le répertoire ~/Cours/IS1/TP2/Sauvegarde/Lieux/Reels et vérifier que « diff » détecte la modification.
5. Supprimer toute l'arborescence ~/Cours/IS1/TP2/Toto, toujours en une seule ligne de commande. Vérifier que ~/Cours/IS1/TP2/Toto n'existe plus.
6. ➤ Créer, grâce à une option de la commande mkdir, toujours en une seule ligne de commande, un répertoire ~/Cours/IS1/TP2/Toto/Tutu/Titi.

Manipuler des ensembles de noms de fichier

Important : ajouter la ligne suivante dans votre fichier ~/.bashrc :

```
export LC_COLLATE=C
```

les *jokers* ou *wildcards* permettent de décrire plusieurs chemins de fichiers en une seule expression : lorsque le shell rencontre un de ces caractères, il l'interprète de toutes les manières possibles correspondant à un nom (de base) de fichier existant.

- «*» représente n'importe quelle suite de zéro, un ou plusieurs symboles pouvant apparaître dans un nom de base (sauf le « . » en début de mot) ;
- «?» représente exactement un symbole quelconque (toujours sauf le « . » en début de mot) ;
- « [] » représente exactement un symbole parmi l'ensemble de symboles décrit entre les crochets :
 - soit par liste exhaustive de caractères, par exemple [abz] ;
 - soit par intervalle (au sens du code binaire) en séparant les bornes de l'intervalle par le symbole «-», par exemple [a-f] ou [2-7] ;
 - soit par union de listes et d'intervalles, par exemple [a-fA-FzZ] ;
 - soit par complément avec «^» ou «!», par exemple [^wx] ou [!a-z].

Exercice 7 – filtrer l'affichage

✎ A l'aide de «ls», afficher la liste de tous les noms de fichier dans le répertoire /usr/bin qui respectent les motifs suivants. Attention ! La commande «ls» sans option peut poser des problèmes, car elle affiche le contenu des répertoires à la place de leurs noms.

1. qui commencent par un k ;
2. qui contiennent un k ;
3. qui terminent par un k ;
4. qui commencent par un m et terminent par un e ;
5. dont la troisième lettre est un k ;
6. qui commencent par un k et contiennent exactement 6 caractères ;
7. qui commencent par un k et contiennent au moins 6 caractères ;
8. qui commencent par un k et contiennent entre 3 et 5 caractères ;
9. qui contiennent un j ou un y ;
10. qui contiennent un chiffre ;
11. qui contiennent un caractère non alphabétique ;
12. qui contiennent un k et un m ;
13. qui contiennent un k et commencent par une lettre parmi r, s, t, u, v, w, x ;
14. qui contiennent un k et ne commencent pas par un m ;
15. qui contiennent un k et ne commencent pas par une lettre parmi l, m, n, o, p.