Introduction aux systèmes d'exploitation (IS1)

TP 9 – les commandes « `find` » et « `sed` »

Travailler sur une arborescence avec « `find` »

La syntaxe de la commande « `find` » est la suivante :

```
find [options] [chemins] [expression]
```

Cette commande parcourt les arborescences commençant en chaque chemin mentionné, en évaluant l'expression fournie pour chaque fichier rencontré. Le premier argument (après les options) commençant par `-`, `(`, `)`, `,`, ou `!` est considéré comme le début de l'expression. Si aucun chemin n'est mentionné, la recherche part du répertoire courant ; si aucune expression n'est fournie, « `find` » utilise l'expression `-print` par défaut.

L'expression est constituée de tests et d'actions, séparés par des opérateurs logiques ; quand un opérateur est manquant, `-and` est appliqué par défaut.

Les tests De très nombreux tests sont possibles, consulter « `man` » pour plus d'informations. Par exemple, `-name`, `-empty`, `-inum`, `-newer`, `-size`, `-type`...

Les actions

- l'action `-print` affiche sur la sortie standard la liste des fichiers trouvés.
- l'action `-delete` supprime les fichiers trouvés.
- l'action `-exec cmd` ; permet plus généralement d'exécuter n'importe quelle commande cmd pour chaque fichier trouvé ; si cmd utilise ce fichier, il est désigné par `{}` ; la fin de la commande est signalée par `;` (point-virgule).

Par exemple, pour connaître le numéro d'i-nœud de tous les fichiers de son arborescence dont le nom contient le motif `toto`, on peut écrire :

```
find ~/ -name '*toto*' -exec ls -i \{\} \;
```

Attention, le motif décrivant le nom des fichiers doit être protégé pour éviter l'expansion immédiate par le shell, de même que les caractères spéciaux {, } et ;

Exercice 1 – recherche par nom

Lister tous vos fichiers :

1. d'extension `.java`,
2. dont le nom est de longueur 6,
3. dont le nom possède au moins deux `'.'` (point),
4. dont la référence absolue possède au moins cinq `'.'` (point).

L'utilisation fréquente de « `xemacs` » crée de nombreux fichiers de sauvegarde, qui finissent par encombrer l'arborescence. On les reconnaît à leur suffixe `'~'`.

5. Écrire une ligne de commande permettant de supprimer de votre arborescence tous ces fichiers.

Attention ! Pour éviter les mauvaises surprises, il est plus que recommandé de ***tester votre ligne de commande sur une sous-arborescence construite pour l'occasion***, et d'utiliser l'option `-i` de « `rm` ».

Exercice 2 – autres options de recherche

Pour résoudre les questions qui suivent, il sera utile de feuilleter la page de manuel de « `find` » (et éventuellement de créer des fichiers satisfaisant les critères demandés).

1. Lister tous vos liens symboliques.
2. Lister tous vos répertoires vides.
3. Lister tous vos répertoires ayant 12 liens.
4. Lister tous vos fichiers ordinaires ayant 2 ou 3 liens.
5. Lister tous vos fichiers sur lesquels vous avez tous les droits, et votre groupe et les autres n'ont aucun droit.
6. Lister avec « `ls -l` » tous les fichiers se trouvant dans votre arborescence personnelle mais qui ne vous appartiennent pas.
7. Compresser avec « `gzip` » tous vos fichiers de plus d'un mégaoctet.

Faire des remplacements avec « `sed` »

La commande « `sed` » lit sur l'entrée standard une ligne de texte, lui applique une série de commandes puis l'affiche sur la sortie standard. Ces commandes peuvent être données soit directement via l'option `-e`, soit dans un fichier externe via l'option `-f`. Un appel à « `sed` » a donc soit la forme :

`sed -e` cmd1 -e cmd2 -e cmd3 (où cmd1, cmd2, cmd3 sont des commandes),

soit la forme :

`sed -f` sed_script (où sed_script est un fichier contenant une série de commandes).

Si on ne souhaite effectuer qu'une commande, l'option `-e` est facultative.

La commande de base de `sed` est la substitution. Elle s'écrit `s/expr/rplcmt/` et sert à remplacer l'expression régulière expr par l'expression rplcmt. Les expressions régulières sont décrites avec la même syntaxe que pour « `grep` ». Par exemple, la commande

```
sed 's/bonjour/Bonjour/'
```

lit du texte sur l'entrée standard, et remplace la premiere occurrence de `bonjour` dans chaque ligne par la chaîne de caractères `Bonjour`. L'utilisation du caractère `/` comme séparateur n'est pas obligatoire et on peut utiliser le caractère que l'on souhaite. Par exemple, la commande

```
sed 's:bonjour:Bonjour:'
```

est équivalente à la commande précédente.

Comme « `grep` », « `sed` » peut également lire dans un fichier donné en argument, et il est conseillé de mettre les commandes entre guillemets simples pour éviter les expansions.

Exercice 3 – commenter du code java

1. Écrire un script « `comment.sh` » prenant en argument un fichier `.java` et commentant toutes les lignes du fichier, c'est-à-dire rajoutant la chaîne `//` au début de chaque ligne du fichier. Tester ce script.
2. Écrire un script « `uncomment.sh` » prenant en argument un fichier java et enlevant les commentaires faits avec `//` en début de ligne. Tester ce script.

Par défaut, une substitution ne se fait que sur la première occurrence du motif. Si l'on souhaite le faire sur tous les motifs de la ligne, il faut rajouter la lettre `g` (pour global) après la substitution. Par exemple, la commande

```
sed -e 's/[bB]onjour/Salut/g'
```

remplace toutes les occurrences des mots `bonjour` ou `Bonjour` d'une ligne par le mot `Salut`.

Exercice 4 – remplacements de base

Télécharger le fichier `lettre.txt` sur Didel, puis effectuer les remplacements suivants :

1. Afficher `lettre.txt` après remplacement de toutes les chaînes `n'e` par le caractère `é`.
2. Afficher `lettre.txt` après remplacement de toutes les chaînes `n‘e` par le caractère `è`.
3. Affichez `lettre.txt` après remplacement de toutes les chaînes `n‘a` par le caractère `à`.
4. À l'aide de l'option `-i` de « `sed` », corriger tous les accents de `lettre.txt`.

Remplacements ciblés et effacements

Par défaut, toutes les lignes sont concernées par les remplacements. Si l'on souhaite ne manipuler que certaines lignes, on peut ***préfixer*** une commande par :

- un numéro de ligne, pour n'appliquer la commande que sur cette ligne,
- deux numéros de lignes séparés par une virgule, pour appliquer la commande sur toutes les lignes entre ces deux numéros,
- une expression de la forme `/expr/` où cmdargexpr est un motif, pour appliquer la commande uniquement aux lignes où le motif est présent.

Par exemple, la commande `sed -e '2,3s/bonjour/BONJOUR/'` remplace le mot `bonjour` par le mot `BONJOUR` sur les lignes 2 et 3 de l'entrée.

De la même manière, la commande `sed -e '/Marc/s/bonjour/BONJOUR/'` remplace le mot `bonjour` par le mot `BONJOUR` sur toutes les lignes qui contiennent le motif `Marc`.

Parmi les autres commandes `sed` dans le manuel, on notera les commandes `d`, `p` et `i\` :

— `d` (delete) empêche l'affichage de la ligne
— `p` (print) affiche la ligne
— `i\ unpeudetexte` écrit (insert) la chaîne `unpeudetexte`

Exercice 5 – manipulation d'affichages en java

1. Écrire un script « `unprint.sh` » qui prend en argument un fichier `.java` etl'affiche à l'écran en enlevant les lignes dans lesquelles est fait un appel à la fonction `System.out.println`. Tester ce script.
2. Écrire un script « `signalPrint.sh` » qui prend en argument un fichier `.java` et y insère

   ```
   // appel a println
   ```

   avant chaque appel à la fonction `System.out.println`. Tester ce script.
3. À l'aide de l'option `-n` de la commande `sed`, afficher les lignes d'un fichier `.java` contenant un appel à la fonction `System.out.println`.

Manipuler des lignes

La commande s permet non seulement de faire des substitutions, mais aussi de manipuler des blocs de texte. Pour sélectionner un bloc de texte dans l'expression d'origine, il suffit de le mettre entre les symboles `\(` et `\)`. Ces blocs sont ensuite rappelés avec les expressions `\1`, `\2`, ..., où le numéro correspond à leur ordre d'apparition.
Par exemple, la commande

```
sed -e 's/\([a-zA-Z]\+\)  \([a-zA-Z]\+\)/\2 \1'
```

inverse les deux premiers mots de chaque ligne.

Exercice 6 – plan de site internet

La commande « `curl` » suivie d'une adresse internet affiche le code source de la page à l'adresse indiquée.

1. Tester la commande en tapant `curl www.wikipedia.org`.
2. Écrire un script « `titre.sh` » qui prend en argument une adresse internet, et affiche le titre de la page située à l'adresse donnée (rappel éventuel : le titre d'une page HTML est entouré des balises `<title>` et `</title>`).
3. Écrire un script « `plan.sh` » qui prend en argument une adresse internet, puis affiche le titre de la page correspondante ainsi que le texte entouré des balises `<h1>` et `</h1>` et `<h2>` et `</h2>`.

Exercice 7 – mise en forme de texte

Récupérer le fichier `liste` sur Didel. On rappelle que celui-ci contient une liste d'entrées, chacune formée d'un nom, d'un nom de groupe, d'un jour et d'une ville, séparés par des doubles-points.

1. Afficher le contenu du fichier de manière lisible, par exemple sous la forme :

   ```
   Étudiant : Yago
   Groupe : GR1
   Date : lundi
   Ville : Paris
   ```

   On rappelle que le caractère de fin de ligne est `\n`.
2. Afficher le fichier trié selon le groupe des étudiants.