Introduction aux systèmes d'exploitation (IS1)

TP 9 : Manipulations textuelles

1 Sélectionner des lignes

1.1 La commande `grep`

La commande `grep` est un filtre qui sélectionne les lignes contenant un motif donné en paramètre, par exemple la chaîne 'bonjour' ou bien encore la chaîne 'ab34'.

Exercice 1 – *Recherche de méthodes*

Récupérez l'archive `programmes.tgz` sur `didel` et désarchivez-la avec la commande `tar` vue au premier TP, grâce à la ligne de commande `tar xzf programmes.tgz`.

1. En utilisant `grep`, déterminez le ou les fichiers qui contiennent un `main`.
2. En utilisant `grep`, trouvez toutes les méthodes statiques (commençant par le mot-clef `static`) des fichiers.

Correction. `grep 'main' *`

1.2 Expressions régulières

La commande `grep` permet de rechercher des chaînes de caractères fixées, mais aussi des motifs plus complexes, les *expressions régulières*, construites avec les opérateurs de la table 1 (à consulter page 2).

Puisque certains caractères sont aussi expansés par le shell, nous vous recommandons de mettre vos expressions régulières entre guillemets simples.

Remarquez que les opérateurs pour fabriquer des expressions régulières et les règles d'expansion vues au TP8 présentent des similarités mais sont différents.

Par exemple, si le fichier `liste-mots.txt` contient les mots {matou, ours, oublier, pousse, patio, reussie, oubli, rue, ruse, russe, ruine, salutation, saturation, savon, station, agio} séparés par des retours à la ligne, on a :

```
[] grep 'rus\+e' liste-mots.txt
ruse
russe
[] grep 'a.*a' liste-mots.txt
salutation
saturation
[] grep '^ou' liste-mots.txt
ours
oublier
```

Exercice 2 – *Utilisation des expressions régulières*

Téléchargez le fichier `liste-mots.txt` sur didel. A l'aide de `grep`, affichez les mots qui respectent les conditions suivantes :

1. contiennent un n (`ruine, salutation, saturation, savon, station`)
2. contiennent un r puis un s (`ours, reussie, ruse, russe`)

TAB. 1 – Quelques opérateurs pour fabriquer des expressions régulières

`.`	un seul caractère quelconque
`\.`	le caractère '.'
`\?`	l'élément précédent est facultatif `'ab\?c'` représente l'ensemble `{ac,abc}`
`*`	l'élément précédent est répété 0 ou plusieurs fois `'ab*'` représente l'ensemble `{a,ab,abb,abbb,...}`
`\+`	l'élément précédent est répété au moins 1 fois `'ab\+'` représente l'ensemble `{ab,abb,abbb,...}`
`\{n\}`	l'élément précédent est répété exactement n fois `'ab\{3\}c'` représente `abbbc`
`\{n,\}`	l'élément précédent est répété au moins n fois `'ab\{3,\}c'` représente l'ensemble `{abbbc,abbbbc,abbbbbc,...}`
`\{n,m\}`	l'élément précédent est répété entre n et m fois `'ab\{3,5\}c'` représente l'ensemble `{abbbc,abbbbc,abbbbbc}`
`^`	début de ligne `'^a'` représente toutes les lignes qui commencent par `a`
`$`	fin de ligne `'a$'` représente toutes les lignes qui finissent par `a`
`[...]`	liste ou intervalle de caractères recherchés `'[abcd]'` (ou `'[a-d]'`) représente l'ensemble `{a,b,c,d}`
`'[^ab]'`	listes des caractères interdits

3. contiennent un r puis deux s collés (`reussie, russe`)
4. terminent par io (`patio, agio`)

Correction.

1. `grep 'n' liste-mots.txt`
2. `grep 'r.*s' liste-mots.txt`
3. `grep 'r.*ss' liste-mots.txt`
4. `grep 'io\$' liste-mots.txt`

Exercice 3 – *Options de `grep`*

1. A l'aide du manuel, déterminez à quoi servent les options `-i`, `-H`, `-n`, `-r` et `-v` de `grep`.
2. En utilisant le fichier `liste-mots.txt` et des options de grep, affichez à l'écran la liste de mots qui ne contiennent pas un `s` (`matou, oublier, patio, rue, ruine, agio`)
3. A l'aide de la commande `grep`, déterminez la liste des fichiers contenant le mot `static` de votre répertoire `IF1`, et le numéro de ligne où ce mot apparaît.

Correction.

1. `-i` ignore la casse (majuscules / minuscules), `-v` affiche le contraire, `-n` affiche le numéro de ligne et `-r` faire une recherche récursive dans les dossiers, `-H` affiche le nom du fichier où le motif apparaît
2. `grep -v 's' liste-mots.txt`
3. `grep -r 'static' ~/IF1/`

1.3 Applications de `grep` (facultatif)

Exercice 4 – *Tri d'e-mails*

Téléchargez le fichier `is1box` sur didel. Ce fichier est le contenu d'une boîte aux lettres.

1. Affichez les sujets de tous les messages reçus. Il doit y en avoir 5.
2. Affichez maintenant les sujets des messages qui contiennent la chaîne `IS1`. Il doit y en avoir 2.
3. Affichez toutes les lignes susceptibles de contenir un numéro de salle de la Halle aux farines (i.e. 3 chiffres et 1 lettre majuscule)

Correction.

1. `grep '^Subject:' is1box`
2. `grep '^Subject:.*IS1' is1box` ou `grep '^Subject:' is1box | grep 'IS1'`
3. `grep ' [0-9]\{3\}[A-Z] ' is1box`

Exercice 5 – *Recherche dans une arborescence*

L'option `-r` permet à `grep` de rechercher dans tout les sous-dossier d'un dossier, mais pas d'en sélectionner uniquement certains. Pour faire cela, il faut utiliser `grep` en combinaison avec `find`.

1. A l'aide de la commande `find`, affichez la liste des fichiers `.java` contenus dans l'ensemble de vos répertoires.
2. Affichez la liste des fichiers `.java` de vos répertoires où sont déclarées des fonctions `moutons`. Comment faire pour avoir uniquement les définitions, et non pas les appels à cette fonction ?
3. Affichez la liste de tous les appels à la fonction `System.out.println` dans les fichiers `.java` de vos répertoires.

Correction.

1. `find ~ -name '*.java'`
2. `find ~ -name '*.java' -exec grep -H 'moutons' {} \;`
3. `find ~ -name '*.java' -exec grep -H 'System.out.println' {} \;}`

   Pour avoir uniquement la définition, on peut lancer le grep sur tout ou une partie du prototype de la fonction.

2 Remplacement dans une ligne

2.1 Commandes de base de `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.

Les 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 la forme `sed -e cmd1 -e cmd2 -e cmd3` où `cmd1`, `cmd2`, `cmd3` sont des commandes, ou `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 celle de substitution. Elle s'écrit `s/expr/remplacement/`, qui sert à remplacer l'expression régulière `expr` et la remplace par l'expression `remplacement`. Les expressions régulières sont décrites avec la même syntaxe que pour `grep`, donc le tableau de la page 2 reste valable.

Par exemple, la commande `sed 's/bonjour/Bonjour/'` lira du texte sur l'entrée standard, et remplacera la premiere occurence de `bonjour` de 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, taper `sed 's:bonjour:Bonjour:'` est équivalent à taper `sed 's/bonjour/Bonjour/'`.

Comme pour `grep`, sed lit par défaut sur l'entrée standard mais peut lire dans un fichier donné en argument, et il est conseillé de mettre les commandes entre guillemets simples pour éviter les expansions.

Exercice 6 – *Commenter du code java*

1. Ecrivez un script `comment.sh` qui prend en argument un fichier java et qui commente toutes les lignes du fichier, c'est à dire qui rajoute la chaîne `//` au début de chaque ligne du fichier. Testez votre script.
2. Ecrivez un script `uncomment.sh` qui prend en argument un fichier java et enlève les commentaires faits avec `//` en début de ligne. Testez votre script.

Correction.

1. `sed -e 's:^://:' $1`
2. `sed -e 's:^//::' $1`

Par défaut, une substitution ne se fait que sur *la première occurence* du motif. Si l'on souhaite le faire sur tout les motifs de la ligne, il faut rajouter la lettre `g` après la substitution. Par exemple, la commande `sed -e 's/[bB]onjour/Salut/g'` remplacera *toutes* les occurences des mots `bonjour` et `Bonjour` d'une ligne par le mot `Salut`.

Exercice 7 – *Remplacements de base*

Téléchargez le fichier `lettre.txt` sur Didel, puis effectuez les remplacements suivants :

1. Affichez le contenu de `lettre.txt` où les chaînes `\'e` ont été remplacées par le caractère `é`.
2. Affichez le contenu de `lettre.txt` où les chaînes `` \`e `` ont été remplacées par le caractère `è`.
3. Affichez le contenu de `lettre.txt` où les chaînes `` \`a `` ont été remplacées par le caractère `à`.
4. Regardez le rôle de l'option `-i` de `sed`. Corrigez les accents de `lettre.txt` à l'aide de `sed`.

Correction.

1. `sed -e "s/\\'e/é/g" lettre.txt`
2. `sed -e 's/\\`e/è/g' lettre.txt`
3. `sed -e 's/\\`a/à/g' lettre.txt`
4. `sed -i  -e "s/\\'e/é/g" -e 's/\\`e/è/g' -e 's/\\`a/à/g' lettre.txt`

2.2 Remplacements ciblés et effacements

Lorsque l'on met plusieurs commandes, elles sont exécutées sur toutes les lignes. 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 `expr` est un motif, pour appliquer la commande uniquement à une ligne 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`.

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

- `d` empêche l'affichage de la ligne à l'écran
- `p` affiche la ligne à l'écran
- `i\ unpeudetexte` écrit la ligne `unpeudetexte` à l'écran

Exercice 8 – *Manipulation d'affichages en java*

1. Ecrivez un script `unprint.sh` qui prend en argument un fichier `.java` et qui l'affiche à l'écran en enlevant les lignes dans lesquelles sont faites un appel à la fonction `System.out.println`. Testez votre script.
2. Ecrivez un script `signalPrint.sh` qui prend en argument un fichier `.java` et qui écrit la phrase `// appel a println` avant chaque appel à la fonction `System.out.println`. Testez votre script.
3. Regardez ce que fait l'option `-n` de `sed`. A l'aide de la commande `sed` uniquement, affichez toutes les lignes d'un fichier `.java` contenant un appel à la fonction `System.out.println`.

Correction.

1. `sed /System.out.println/d $1`
2. `sed /System.out.println/i\ ‘‘// appel a println’’ $1`
3. `sed -n /System.out.println/p`

2.3 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 `\(` et `\)`. On les rappelle avec les expressions `\1`, `\2`, etc ..., 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 entrée.

Exercice 9 – *Plan de site internet*

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

1. Testez la commande en tapant `curl www.wikipedia.org`
2. Ecrivez un script `titre.sh` qui prend en argument une adresse internet, et qui affiche le titre de la page à l'adresse donnée. On indique que le titre d'une page HTML est entre des balises `<title>` et `</title>`.
3. Ecrivez un script `plan.sh` qui prend en argument une adresse internet, affiche le titre de la page à l'adresse donnée ainsi que le texte contenu entre les balises `<h1>` et `</h1>` et `<h2>` et `</h2>`.

Correction. `curl $1 | sed -n -e 's|^.*<title>\([^<]*\)<\/title>.*$|\1|p'`

Exercice 10 – *Mise en forme de texte*

Récupérez le fichier `liste` sur Didel. On rappelle que celui-ci contient une liste de nom, nom de groupe, jour et ville, séparés par des doubles-points.

1. Affichez le contenu du fichier de manière lisible, c'est-à-dire sous la forme

```
Etudiant : Yago
Groupe : GR1
date : lundi
ville : Paris
```

On rappelle que le caractère de fin de ligne est `\n`.

Correction. `sed -n 's/\(.*\):\(.*\):\(.*\):\(.*\)/Nom : \1\nGroupe : \2\ndate : \3\nville : \4\n/p' liste`