Licence d'Informatique – S4 – 2009/2010

TP d'AL

mars 2010

Manipulation d'expressions régulières avec `egrep`

1 Introduction

Le système d'exploitation `unix`, dont `linux` que vous utilisez en TP est une implantation, comprend de nombreuses commandes prenant des expressions rationnelles comme arguments ; sous `unix` on parle plutôt d'expressions régulières et c'est le terme que nous utiliserons dans ce TP.

Les commandes `unix` manipulant ou utilisant des expressions régulières peuvent être regroupées en trois familles :

1. les outils d'édition de texte (`ed`, `vi`, `emacs`,...)
2. les outils d'analyse lexicale (`lex`, `flex`, `bison`,...)
3. les outils d'extraction d'information (`sed`, `grep`,...)

Nous utiliserons pour ce TP l'outil `grep` dans sa version améliorée `egrep` qui permet l'utilisation d'expressions régulières plus expressives. Cette commande est disponible dans toutes les versions d'`unix` et dans toutes les distributions de `linux`
`egrep` est un acronyme pour `E`xtended `G`et `R`egular `E`xpression `P`attern, alors que `grep` est un acronyme pour `G`lobally - `R`egular `E`xpression - `P`rint qui sont les trois options de recherche dans un texte disponibles dans l'editeur *historique* `ed` sous `unix`. Par extension, `grep` désigne tout programme chargé de retrouver un fichier sur un disque contenant une chaîne de caractères donnée.

La syntaxe d'appel (Taper `man egrep` dans un terminal pour une description complète des options) de la commande `egrep` est

```
egrep [options] motif fichiers
```

Le résultat de l'invocation de la commande est la recherche dans les fichiers d'entrée indiqués et l'affichage des lignes correspondant au motif fourni. Parmi les options possibles, citons :

`-c` Afficher le nombre de lignes en correspondance pour chaque fichier d'entrée plutôt que les lignes elles-mêmes.

`-e motif` Utiliser le motif indiqué. Ceci permet d'utiliser des motifs commencant par -.

`-h` Ne pas afficher le nom des fichiers dans les résultats lorsque plusieurs fichiers sont parcourus.

`-i` Ignorer les différences majuscules/minuscules aussi bien dans le motif que dans les fichiers d'entrée.

`-l` Afficher uniquement le nom des fichiers pour lesquels des résultats positifs auraient été affichés.

`-n` Ajouter à chaque ligne de sortie son numéro d'ordre dans son fichier d'entrée.

`-o` N'afficher que la partie d'une ligne qui correspond au motif.

`-v` Inverser la mise en correspondance : affiche les lignes ne correspondant pas au motif fourni.

L'utilisation d'`egrep` peut aussi se faire en enchaînement de commandes, à l'aide du symbole `|`, c'est alors l'entrée standard qui est analysée.

Exemple 1 *Sous* `unix`*, la commande* `ps` *affiche une liste des processus en cours pour l'utilisateur courant, et qui ont été lancés dans le terminal courant. Pour avoir une liste de tous les processus, on tape* `ps aux`*. Vérifiez par vous-même l'effet de cette commande en la tapant dans un terminal. Si on veut maintenant n'afficher que les processus dont le propriétaire est* `root` *(C'est le nom de l'utilisateur ayant les droits d'administrateur sous* `unix`*), on peut taper :* `ps aux | egrep root` *pour obtenir un* filtrage *de l'affichage et ne garder que les lignes contenant le mot* `root`*.*

Dans l'exemple précédent, vous pouvez constater que la commande affiche aussi la ligne correspondant au processus `egrep` que vous avez lancé et qui n'est pas un processus de `root`. Il faut pouvoir préciser qu'on veut sélectionner les lignes pour lesquelles `root` apparaît uniquement au tout début de la ligne. Ce genre d'information est possible dans l'expression régulière définissant le motif. La section suivante décrit la syntaxe des expressions régulières utilisées dans `egrep`.

2 La syntaxe des expressions régulières

Cette syntaxe suit à peu-près celle vue en cours d'AL pour les expressions rationnelles : la concaténation est obtenue par juxtaposition des expressions régulières, l'étoile de Kleene utilise bien le caractère `*` et on peut utiliser des parenthèses pour isoler des sous-expressions. Par contre, l'union (le `+` binaire des expressions rationnelles vues en cours) dans les expressions régulières utilise le symbole `|`. On peut utiliser le symbole `+`, opérateur unaire, qui veut dire *une ou plusieurs occurrences de*. Rappelons que $L^+ = L.L^*$. On dispose aussi d'un certains nombres d'opérateurs supplémentaires.

Certains caractères ont une signification particulières dans les expressions, on les appelle des *méta-caractères*. En voici la liste :

```
"  \  {  }  [  ]  ^  $  <  >  ?  .  *  +  |  (  )  /
```

Il permettent de définir des opérateurs dont la signification est donnée dans la figure 1, où les caractères `x`, `y` et `z` représentent des caractères quelconques qui ne sont pas des méta-caractères et n et m représentent des entiers positifs ou nuls.

Expression	Signification
`x`	une occurrence du caractère x
`"x"`	le caractère x, même s'il s'agit d'un méta-caractère
`\x`	le caractère x, même s'il s'agit d'un méta-caractère
`^x`	un x au début d'une ligne
`x$`	un x à la fin d'une ligne
`x+`	une ou plusieurs occurrences de x
`x*`	zéro ou une ou plusieurs occurences de x
`x?`	une occurence optionnelle de x
`x{n}`	exactement n occurrences de x
`x{n,m}`	au moins n occurrences et au plus m occurrences de x
`(x)`	un x
`.`	n'importe quel caractère
`x|y`	un x ou un y
`[xy]`	n'importe quel caractère dans la liste xy (donc le caractère x ou le caractère y)
`[x-z]`	n'importe quel caractère dans l'intervalle de x à z (donc x, y ou z)
`[X-Zx-z]`	une occurrence de caractère X, Y, Z, x, y ou z
`[-+0-9]`	le signe `-` ou le signe `+` ou n'importe quel chiffre
`[^x]`	n'importe quel caractere sauf x
`[^x-z]`	n'importe quel caractere sauf x, y ou z

FIGURE 1 – La syntaxe des expressions régulières

Les opérateurs de plus forte priorité sont `"" [] ()` puis viennent `+ * ?` puis la concaténation et enfin l'opérateur `|` qui a la plus faible priorité.

Finalement, il existe certaines classes de caractères prédéfinies. Leurs noms sont assez explicites et sont donnés figure 2.

Nom symbolique	Classe correspondante
`[:alnum:]`	les caractères alpha-numériques
`[:alpha:]`	les caractères alphabétiques
`[:cntrl:]`	les caractères de contrôle
`[:digit:]`	les caractères chiffres
`[:lower:]`	les lettres minuscules
`[:punct:]`	les caractères de ponctuation
`[:space:]`	les caractères espace
`[:upper:]`	les lettres majuscules

FIGURE 2 – Les classes de caractères

✔*Les crochets font partie du nom symbolique et ne représentent pas les* opérateurs *intervenant pour les listes et intervalles. Par exemple, pour obtenir l'équivalent de* `[0-9A-Za-z]` *il faut utiliser* `[[:alnum:]]` *et non* `[:alnum:]`.

✔*Sous* `unix`*, quand on tape une commande, certains caractères comme* `*`*,* `(` *ou* `?` *sont interprétés par l'interpréteur de commandes (qu'on appelle le* shell*). Pour éviter leur interprétation, il faut encadrer le motif par des guillemets ou des apostrophes. Par exemple* `egrep 'a?b.*' *.java`.

Exemple 2 *Quelques exemples :*
- *l'expression à utiliser pour l'exemple 1 était* `^root`
- *à l'expression* `[0-9][A-Za-z0-9]*` *correspond toute chaîne composée de caractères aphanumériques et commençant par un chiffre. On aurait pu aussi utiliser* `[[:digit:]][[:alnum:]]*`
- *aux expressions* `Automate` *ou* `"Automate"` *correspond la chaîne* Automate.
- *à l'expression* `\"Automate\"` *correspond la chaîne* "Automate".
- *à l'expression* `(ab|cd+)?(ef)*` *correspond l'expression rationnelle (entre autres) en notation usuelle :* $(ef)^* + ab(ef)^* + cd^+(ef)^*$.

3 Exercices

Exercice 3.1 : Récupérez l'archive `motif.tgz` sur le portail AL et décompressez la en tapant dans votre répertoire de travail la commande `tar -xvzf motif.tgz`. Placez vous ensuite dans le répertoire `motif`

Question 3.1.1 : Utilisez `egrep` pour afficher toutes les lignes des fichiers `java` du répertoire commencant par un caractère `/`

Question 3.1.2 : Utilisez `egrep` pour afficher toutes les lignes des fichiers `java` du répertoire commençant par un caractère `p` éventuellement précédé d'espaces et finissant par un caractère `{` éventuellement suivi d'espaces.

Question 3.1.3 : Utilisez `egrep` pour afficher toutes les lignes des fichiers `java` du répertoire contenant au moins une occurrence de la chaîne `int`

Question 3.1.4 : Utilisez `egrep` pour afficher toutes les lignes des fichiers `java` du répertoire contenant au moins une occurrence du mot `int`

En `pascal`, les identificateurs doivent commencer par une lettre, puis se poursuivre éventuellement par des lettres ou des chiffres.

Question 3.1.5 : Utilisez `egrep` pour afficher tous les mots suivant la syntaxe des identificateurs `pascal` des fichiers `java` du répertoire.

En `java`, l'usage est d'écrire les constantes de classe en majuscule.

Question 3.1.6 : Utilisez `egrep` pour afficher tous les identificateurs des fichiers `java` du répertoire. ne contenant que des majuscules. Essayez de ne pas afficher ce qui fait partie d'un contenu `HTML`.

On définit une notion de littéral chaîne de caractères de la façon suivante : une chaîne de caractères commence par un caractère guillemet et se termine, sur la même ligne, au prochain caractère guillemet. Pour pouvoir inclure des occurrences de guillemets dans ces chaînes, on double les caractères guillemets à inclure. Par exemple, `a"abc""cab"a` contient une seule chaîne de caractères dont le contenu est `abc"cab`. En revanche `a"abc" "cab"a` contient deux chaînes de caractères dont les contenus sont respectivement `abc` et `cab`.

Question 3.1.7 : Utilisez `egrep` pour afficher toutes les occurrences de littéral chaîne des fichiers du répertoire.

Exercice 3.2 : Récupérez l'archive `html.tgz` sur le portail AL et décompressez la en tapant dans votre répertoire de travail la commande `tar -xvzf html.tgz`. Placez vous ensuite dans le répertoire `html`

Question 3.2.1 : Affichez la liste de toutes les balises ouvrantes `HTML` des documents du répertoire.

Question 3.2.2 : Essayez d'extraire tous les numéros de téléphone apparaissant dans les documents du répertoire.

Les éléments que vous aurez à traiter dans la suite sont les éléments `A` [1] et `IMG`. Ces éléments permettent respectivement de définir un hyperlien (ou ancre) et d'inclure une image dans une page `HTML`. Les références correspondantes (cible de l'hyperlien ou source de l'image) sont données par l'intermédiaire de l'attribut `HREF` pour l'élément `A` et de l'attribut `SRC` pour l'élément `IMG`. Voici quelques exemples :

```
<A HREF="http://www.cr-npdc.fr">Nord-Pas-de-Calais</A>
<A HREF="/LIFL_ANG/index.Ang.html">Site overview</A>
<a id=1a class=q href="/imghp?hl=fr" onClick="return c('/imghp?hl=fr');">
  <font size=-1>Images</font>
</a>
<IMG BORDER=0  ALT="USTL" SRC="/LOGOS/mini-logo-ustl.gif" USEMAP="#ustl">
```

Question 3.2.3 : Affichez la liste de toutes les lignes des documents du répertoire contenant des balises `A` et `IMG` qui disposent sur la même ligne respectivement d'un attribut `HREF` et d'un attribut `SRC`.

1. Les noms d'éléments peuvent être donnés en majuscule ou minuscule