TDM 2 - Automates et Langages

septembre 2018

Manipulation d'expressions régulières (2ème partie : avec **egrep**)

1 Introduction

egrep est un outil UNIX permettant une recherche de **motif** (expression régulière) dans un ou plusieurs fichiers. Dans son fonctionnement « normal » :

```
egrep motif fichiers
```

En pratique, on protège le motif par des quotes afin d'empêcher son interprétation par le shell Unix. Ces quotes ne font bien évidemment pas partie du motif

```
egrep 'motif' fichiers
```

1.1 Recherche de sous-chaînes, ligne par ligne

egrep recherche, **séparément dans chaque ligne du fichier**, un texte correspondant au motif. Il n'est pas nécessaire que la ligne entière corresponde au motif : egrep cherche une (ou plusieurs) sous-chaîne(s) qui conviennent. Voici quelques exemples :

motif	ligne (occ. trouvées en rouge)	nb
`à\s[a-z]+`	`Que j'aime à faire apprendre ce nombre utile aux sages`	1
`re`	`Que j'aime à faire apprendre ce nombre utile aux sages`	4
`.+`	`Que j'aime à faire apprendre ce nombre utile aux sages`	1
`[0-9]+`	`Que j'aime à faire apprendre ce nombre utile aux sages`	0
`\s.+`	`Que j'aime à faire apprendre ce nombre utile aux sages`	1

Par défaut, egrep recherche toujours la plus longue chaîne possible correspondant au motif (pour le premier exemple, la recherche ne s'arrête pas à « `à f` », qui convient pourtant au motif, mais se poursuit pour trouver « `à faire` »)

1.2 Fonctionnement et options

egrep affiche chaque ligne du ou des fichiers dans laquelle le motif a pu être trouvé.
Par exemple : `egrep [A-Z][0-9] *.txt` affichera toutes les lignes des fichiers `.txt` contenant une majuscule suivie d'un chiffre décimal (quoi qu'il se trouve avant ou après). Voici quelques options qui modifient le fonctionnement de egrep :

Modifient la forme du résultat	
–color	affiche en couleur les sous-chaînes trouvées
-n	préfixe chaque ligne par son numéro
-c	affiche uniquement le nombre de lignes trouvées, pas les lignes elles-même
-h	n'affiche pas le nom du fichier en début de ligne
-o	n'affiche que la partie de la ligne qui correspond au motif
Modifient les critères de recherche	
-i	ignore la casse majuscules/minuscules
-x	ne cherche que les lignes qui correspondent en totalité (et pas les facteurs propres)
Modifient la syntaxe ou la source des motifs	
-P	utilise la syntaxe Perl pour les motifs
-f nom de fichier	lit le motif dans le fichier et non sur la ligne de commande

2 Syntaxe des expressions

En plus de ce que nous avons vu lors de la séance précédente, voici quelques fonctionnalités disponibles dans les expressions (avec egrep).

2.1 Classes de caractères prédéfinies

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

Par exemple, pour désigner une lettre on pourra écrire `[[:alpha:]]`
Pour désigner une lettre ou un @, ou pourra écrire `[[:alpha:]@]`
Pour egrep, `\s` ne désigne pas un espace (utiliser `[[:space:]]`)
Attention : quand le motif est entré sur la ligne de commande (donc dans la plupart des cas) les caractères spéciaux Unix doivent être échappés. Il est conseillé de mettre l'expression régulière entre deux signes '

2.2 Assertions

Une assertion est un élément de la syntaxe des expressions régulière qui fixe une condition portant sur le contexte dans lequel le motif est recherché. Les deux assertions les plus connues sont ^ et $

^	début de ligne
$	fin de ligne

Exemples

motif	ligne (occ. trouvées en rouge)	nb
`[0-9]+`	ab cd 452	1
`^[0-9]+`	ab cd 452	0
`[0-9]+$`	ab cd 452	1
`[0-9]+$`	ab cd 452x	0
`^[0-9]+$`	ab cd 452	0
`^[0-9]+$`	452	1

Un mot « usuel » est une suite de caractères qui peuvent être une lettre de l'alphabet usuel, un chiffre ou l'underscore. Des assertions permettent de vérifier si le motif apparaît au début ou en fin d'un mot usuel.

\<	début d'un mot usuel
\>	fin d'un mot usuel
\b	début ou fin d'un mot usuel

Exemples

motif	ligne (occ. trouvées en rouge)	nb
`\<[abc]+`	axaa xbb ccxcy	2
`\b[abc]+`	axaa xbb ccxcy	2
`[abc]+\>`	axaa xbb ccxcy	2
`[abc]+\b`	axaa xbb ccxcy	2
`\b[0-9]+\b`	a99b 1234 x56 a78	1
`\<[0-9]+\>`	a99b 1234 x56 a78	1

3 Exercices

Exercice 1 :
Cyrano
Vous utiliserez le fichier Cyrano.txt
Q 1 . Affichez toutes les lignes du fichier contenant la chaîne « nez ». En utilisant l'option `--color=auto` vous visualiserez tous les facteurs du texte correspondant au motif cherché.
NB : 8 lignes conviennent
Q 2 . Affichez toutes les lignes du fichier contenant un mot ou un portion de phrase entre parenthèses.
NB : 7 lignes conviennent
Q 3 . On considèrera que les mots sont composés uniquement de lettres.
Un mot (au sens littéraire du terme) est une suite de ces lettres délimitée avant ou après par un autre caractère ou une extrémité de ligne.

Affichez toutes les lignes comportant un mot de longueur 5 exactement. Là aussi, l'option `--color` permettra de visualiser les mots trouvés.
Vérifiez que vous prenez bien en compte les mots de 5 lettres figurant en début ou en fin de ligne. (indication : utilisez les prédicats)
NB : 25 lignes conviennent
Q 4 . Toutes les exemples de style (agressif, amical, descriptif,...) de la célèbre « tirade du nez » commencent par le même motif. Observez le texte pour trouver ce motif et déduisez-en une commande grep qui affiche tous les vers de cette tirade commençant par un nom de style.
NB : 19 lignes conviennent
À rendre pour cet exercice : un script shell (bash) comportant les commandes egrep de cet exercice. Chaque commande sera précédée de l'affichage (echo) bien visible du numéro de question précédé de `**`, et d'un espace de façon à obtenir à l'exécution un résultat comme ceci :
(Votre script doit produire un texte qui respectera **exactement** le format de sortie ci-dessous. La validité sera vérifiée automatquement par le correcteur).

```
** Q1
Vous...vous avez un nez...heu...un nez...très grand.
En variant le ton, par exemple, tenez:
...
** Q2
(De Guiche)
....
** Q3
Vous...vous avez un nez...heu...un nez...très grand.
...
** Q4
Agressif : " Moi, monsieur, si j'avais un tel nez,
Amical : " Mais il doit tremper dans votre tasse!
...
```

Exercice 2 :
Vous utiliserez les fichiers du sous-répertoire html.
Q 1 . En reprenant et adaptant ce que vous avez fait pour le premier TP, écrivez un script shell qui définit une variable `valeurAttribut` contenant le motif des valeurs d'attributs XML.
Ajoutez à votre script une commande egrep affichant (avec colorisation) toutes les lignes qui contiennent ce motif dans les fichiers du répertoire html
Q 2 . Définissez de même des variables `nomXML` et `refEntite` Vous pourrez ensuite définir une variable `baliseOuvrante` en utilisant les variables précédentes.
Testez cette expression en ajoutant une commande egrep affichant avec colorisation toutes les balises ouvrantes tenant sur une seule ligne dans les fichiers du dossier html.
Q 3 . Observez quelles formes (il y en a plusieurs) prennent les numéros de téléphone figurant dans les fichiers du dossier `html`. Essayez d'extraire avec une seule commande grep tous les numéros de téléphone apparaissant dans les documents du répertoire. Le résultat affichera le nom du fichier parcouru puis les numéros qu'il contient. (utilisez la commande -o de egrep).

```
** Q3
html/contact.html:03.20.43.45.09
03.20.33.62.31
03.20.43.44.94
03.20.33.72.84
html/contact.html:+33 (0) 3.20.43.44.94
html/fil.html:+33 (0) 3.20.43.44.94
```

À rendre pour cet exercice : comme précédemment un script shell (bash) comportant les commandes egrep de cet exercice. Pour les questions 1 et 2 **vous limiterez à 10** le nombre de lignes affichées (option `-m 10` de egrep).

Exercice 3 :

Le fichier `bano-59009.csv` est un fichier CSV (« comma separated values »). Un fichier CSV représente une table. Chaque ligne du fichier est une ligne de la table. Les données des différentes cellules (ou colonnes) d'une même ligne sont séparées par une virgule.

Le fichier fourni représente une table à 8 colonnes (donc chaque ligne comporte exactement 7 virgules : il n'y a pas de virgule après la dernière cellule). Il contient une base d'adresses géolocalisées de Villeneuve d'Ascq, à raison d'une adresse par ligne.

Q 1 . La deuxième colonne contient le numéro (au sein de la voie). En utilisant egrep, affichez exactement les adresses ayant un numéro qui comporte le chiffre 4 et qui se termine par BIS ou par TER (ex : 40BIS, 341TER, ...)

NB : 80 lignes conviennent

Q 2 . La troisième colonne contient le nom de la voie. Affichez exactement les adresses dont **le nom de voie** comporte le mot « Ascq »

NB : 113 lignes conviennent

Q 3 . Sélectionnez les adresses dont le **nom de voie** ne comporte aucune lettre minuscule.

NB : 639 lignes conviennent

À rendre pour cet exercice : comme précédemment un script shell (bash) comportant les commandes egrep de cet exercice.