Shell

TD2 - Expressions régulières, Grep, Sed

A. Krähenbühl

Telecom Nancy, Université de Lorraine

23 Avril 2014

Plan

Plan

Qu'est-ce que c'est ?

Les expressions régulières décrivent des **chaines de caractères** par des **motifs**.
Exemples :

```
1) [a-z]*.c      2) [^0-9]+.(c|o|java)
```

Lorqu'une chaine de caractère correspond à une expression régulière, on dit qu'elle **match**.
Exemple :

- *premierfichier*.*c* match avec 1)
- *premierFichier*.*c* ne match pas avec 1)

Remarque : « Expression régulière » est une mauvaise traduction de « Regular expression » qui signifie **expression rationnelle**.

À quoi ça sert ?

À manipuler facilement des données textuelles.

C'est utile par exemple pour :

- Rechercher des fichiers sur un disque dur
- Retrouver des informations dans fichier de données
- Trier, classer, séparer, fusionner des données

Les caractères uniques

c	« ce caractère »
*	« le caractère '*', pas le caractère spécial '*' »
.	« un seul caractère, n'importe lequel »
[**sncf**]	« l'un des quatre caractères »
[^**ratp**]	« n'importe quel caractère sauf ces quatres là »
[**a-m**]	« l'un des caractères compris dans l'intervalle »
[^**4-2**]	« n'importe quel caractère sauf dans cet intervalle »

Les combinaisons de caractères

.*	« Un nombre quelconque de fois le caractère »
.+	« Au moins une fois le caractère »
.?	« Zéro ou Une fois le caractère »
.\{n\}	« n fois le caractère »
.\{n_1,n_2\}	« entre n_1 et n_2 fois le caractère »

Les combinaisons de pattern

^er	« er en début de ligne »
er$	« er en fin de ligne »
$er_1 \mid er_2$	« l'une des deux er »
\(er\)*	« er répétée un nombre quelconque de fois »
\(er\)+	« er répétée au moins une fois »
\(er\)?	« er répétée Zéro ou Une fois »
\(er\)\{n\}	« er répétée n fois »
$\backslash(\mathbf{er}\backslash)\backslash\{n_1,n_2\backslash\}$	« er répétée n_1 à n_2 fois »

Les caractères spéciaux

\0 « la fin d'un fichier »(le caractère NULL)

\n « un retour à la ligne »(new-line)

\r « un retour à la ligne »(carriage-return)

\t « une tabulation »(tab)

Exemples

linux	le mot *linux*
linux*	les chaines *linu*, *linux*, *linuxx*, *linuxxx*, ...
(linux)*	la chaine vide ou *linux*, *linuxlinux*, ...
^linux	le mot *linux* en début de ligne
^linux$	une ligne ne contenant que le mot *linux*
^linux$	une ligne ne contenant que le mot *linux*
^$	la ligne vide
linux.*mieux	*linuxmieux*, *linux c'est bien mieux que windows*

Plan

Man

GREP signifie **G**lobal **R**egular **E**xpression **P**rint.

Elle permet d'**afficher les lignes qui match** une expression régulière.

Utilisation

```
grep [OPTIONS] PATTERN FICHIERS
```

Exemple d'utilisation :

```
grep ([Ee]rror)|([Ww]arning) logFile.txt
```

Quelques options

-v inverse le résultat → affiche les ligne qui ne contiennent PAS le motif

-l affiche le nom des fichiers contenant une occurence

-c affiche le nombre de lignes qui contiennent une occurence au moins

-i ignore la casse des caractères (a == A)

-m *n* s'arrête après *n* lignes qui match

Énoncé

Écrire une commande *isLog* permettant de tester si un utilisateur est connecté sur la machine hôte. Utilisez la commande *who*.

Exemple :

```
$ ./isLog dupont
dupont est connecte
$ ./isLog dupond
dupond n'est pas connecte
```

On suppose que la commande *who* afiiche les infos suivantes :

```
krahenbuhl    tty7     2013-04-18 10:01
dupont        tty3     2013-04-18 10:32
dupont        pts/0    2013-04-18 10:33
charlier      tty5     2012-09-01 08:00
```

Solution possible

```
#! /bin/bash

if [ $# -ne 1 ]
then
    echo usage : $0 nom_utilisateur
    exit 1
fi

who | grep -q "^$1"

if [ $? -eq 0 ]
then
    echo $1 est connecte
else
    echo $1 n'est pas connecte
fi

exit 0
```

Plan

Man

SED signifie **S**tream **ED**itor.

Il permet de modifier un **flux** ligne par ligne à partir d'expressions régulières.

Utilisation

```
sed [OPTIONS] COMMANDE FICHIER
```

Exemples d'utilisation :

```
sed 1d logFile.txt
sed -n 1p logFile.txt
sed s/Avril/04/g logFile.txt
```

Les commandes

- **d** Supprime les lignes qui match
- **p** Affiche les lignes qui match
- **s** Substitue la partie qui match par une partie de remplacement

La commande de suppression

d Supprime les lignes qui match

Exemples d'utilisation :

```
sed 5d file
sed 1,10d file
sed /^#/d file
```

La commande d'affichage

p Affiche les lignes qui match

-n Option de sed qui supprime l'affichage standard

Exemples d'utilisation :

```
sed -n 5p file
sed -n 1,10p file
sed -n /^#/p file
```

La commande de substitution

s Substitue la partie qui match par une partie de remplacement

Utilisation

```
sed s/<exp1>/<exp2>/<options> file
```

Exemples d'utilisation :

```
sed s/Avril/04/  file
sed s/Avril/04/g file
sed s/^#.*//g file
```

Énoncé

Écrire un script bash *substitution* qui remplace toutes les occurences de la chaine passée en 1er paramètre par la chaine passée en deuxième argument, cela dans le fichier passé en troisième argument. Le résultat sera stocké dans ce même fichier en troisième argument.

Restriction : les arguments ne contiennent pas d'espace.

Exemple :

```
$ ./substitution [cC]orbeau Yoda CorbeauEtRenard.txt
$ more CorbeauEtRenard.txt
Maitre Yoda, sur un arbre perche,
Tenait en son bec un fromage.
[...]
```

Solution possible

```
#! /bin/bash

if [ $# -ne 3 ]
then
    echo usage : $0 chaine1 chaine2 fichier
    exit 1
fi

sed "s/$1/$2/g" $3 > /tmp/$$
mv -f /tmp/$$ $3

exit 0
```

Plan

Man

Find est une commande de **recherche de fichiers**.
Elle effectue une recherche de façon **récursive**.

Utilisation

```
find CHEMIN [OPTIONS] EXPRESSION
     OU ?             QUI ? et QUOI ?
```

Exemples d'utilisation :

```
find $HOME -name "*.txt"
find /     \( -name "*.java" -o "*.class" /) -print
find /etc  -name "*.c" -exec ls -l {} \;
```

Expression : les tests (qui ?)

Liste non exhaustive → **RTFM** !

-name noms des fichiers à rechercher. Peut être une expression régulière.

-iname identique à -name, sans tenir compte de la casse.

-writable les fichiers doivent avoir un accès en écriture.

-perm fichiers dont les permissions sont exactements celles-ci.

-type fichiers de ce type (d=répertoires, f=fichiers régulier, l=lien symbolique, etc.)

Expression : les ordres (quoi ?)

-print afficher le résultat sur la sortie standard (ordre par défaut).

-exec exécuter une commande pour chaque fichier trouvé.

-delete supprime les fichiers → ATTENTION AUX DÉGATS !

Les options

-depth traiter d'abord les sous-répertoires

-maxdepth aller jusqu'à la profondeur donnée dans l'arborescence

-mindepth commencer seulement à la profondeur donnée dans l'arborescence

Plan

Man

Cut est une commande de **découpage d'une ligne**.

Utilisation

```
cut   OPTIONS   FICHIER
```

Exemples d'utilisation :

```
cut -c 4-7 fichier.txt
echo "abcdef" | cut -c 3
echo "un deux trois" | cut -d ' ' -f 2
```

Options

-c LIST selectionner les caractères de LIST

-f LIST selectionner les champs LIST

-d CHAR définir le caractère de séparation

LIST est un intervalle de la forme :

N Le N^{e} caractère ou champ

N- À partir du N^{e} caractère ou champ

N-M Du N^{e} au M^{e} caractère ou champ

-M Jusqu'au M^{e} caractère ou champ

Exemples

```
$ echo "abcdef" | cut -c 3
c

$ echo "un deux trois" | cut -d ' ' -f 2-
deux trois

$ echo "/home/etudiants/Shell/Tp5" | cut -d '/' -f 4
Shell
```

Tableau blanc non interactif