Expressions régulières en AWK

Thomas MARTIN Florian LAPACHET

Définitions

Expressions régulières :

Aussi appelé regex ou regexp, est une séquence de caractères qui définit un modèle de recherche . Ce modèle est utilisé par des algorithmes de recherche de chaine pour des opérations “rechercher”.

AWK:

Langage de traitement de ligne, disponible sur Unix et windows. Il est principalement utilisé pour traiter des fichiers texte avec des opérations complexes de recherche, de remplacement et de conversion.

Syntaxe AWK

La syntaxe est inspirée du C :

```
awk [options] [programme] [fichier]
```

où la structure du programme est :

```
'motif1 { action1 } motif2 { action2 } …'
```

Quelques options :

- `-F séparateur` : permet de modifier le séparateur de champs ;
- `-f fichier` : lit le programme à partir d'un fichier.
- `-v awkVar=$shellVar` : Permet de facilement intégrer des variables du shell dans le code awk.

Les symboles

[ ] pour grouper des éléments
^ pour indiquer le début de chaîne
$ pour indiquer la fin de chaîne
| pour indiquer un choix possible
. pour indiquer n'importe quel caractère

Quelques expressions régulières simples

^B	chaîne qui commence par B
G$	chaîne qui finit par G
^.$	chaîne à un seul caractère
[AEIOU]	chaîne avec une seule voyelle majuscule
^[ABC]	chaîne qui commence par A ou B ou C
^[^a-z]$	chaîne à un seul caractère qui n'est pas une minuscule

Mise en application

1) Manipulation de champs

```
root@toto-VirtualBox:/home/toto/Documents# cat name.txt
1 xavier caen calvados
1 xavier caen calvados
2 pierre paris iledefrance
3 paul lyon rhone
root@toto-VirtualBox:/home/toto/Documents# cat name.txt
1 xavier caen calvados
1 xavier caen calvados
2 pierre paris iledefrance
3 paul lyon rhone
root@toto-VirtualBox:/home/toto/Documents# cat name.txt | awk '{print $1}'
1
1
2
3
root@toto-VirtualBox:/home/toto/Documents# cat name.txt | awk '{print $1$2}'
1xavier
1xavier
2pierre
3paul
root@toto-VirtualBox:/home/toto/Documents# cat name.txt | awk '{print $1 " + " $
2}'
1 + xavier
1 + xavier
2 + pierre
3 + paul
root@toto-VirtualBox:/home/toto/Documents#
```

2) Remplacer du caractère avec la commande GSUB

```
root@toto-VirtualBox:/home/toto/Documents# cat name.txt
1 xavier caen calvados
1 xavier caen calvados
2 pierre paris iledefrance
3 paul lyon rhone
root@toto-VirtualBox:/home/toto/Documents# cat name.txt | awk '{gsub("ier","ki",
$2) ; print $2}'
xavki
xavki
pkire
paul
root@toto-VirtualBox:/home/toto/Documents# cat name.txt | awk '{gsub("ier","ki",
$0) ; print $0}'
1 xavki caen calvados
1 xavki caen calvados
2 pkire paris iledefrance
3 paul lyon rhone
root@toto-VirtualBox:/home/toto/Documents#
```

3) Utiliser une variable bash dans du awk

```
root@toto-VirtualBox:/home/toto/Documents# jour=$(date +%d/%m/%Y) ; cat name.txt
 | awk -v mavariable="$jour" '{print mavariable " : " $0}'
27/01/2021 : 1 xavier caen calvados
27/01/2021 : 1 xavier caen calvados
27/01/2021 : 2 pierre paris iledefrance
27/01/2021 : 3 paul lyon rhone
root@toto-VirtualBox:/home/toto/Documents#
```

4) Supprimer les lignes en double dans un fichier

```
root@toto-VirtualBox:/home/toto/Documents# cat name.txt
1 xavier caen calvados
1 xavier caen calvados
2 pierre paris iledefrance
3 paul lyon rhone
root@toto-VirtualBox:/home/toto/Documents# cat name.txt | awk '{tab[$0]++} END {
for (line in tab) print line}'
1 xavier caen calvados
2 pierre paris iledefrance
3 paul lyon rhone
root@toto-VirtualBox:/home/toto/Documents#
```

FIN