Outils pour la Manipulation et l'Extraction de Données

Notes de cours

Cours 1

10 mars 2021

AWK : une brève introduction

AWK : historique

`awk` est un outil pour la manipulation de fichiers (au format texte).

Il a été conçu en 1977 par Alfred Aho, Peter Weinberger et Brian Kernighan (Laboratoires Bell) et intégré dans Unix V7 en 1979.

`awk` est aujourd'hui dans la norme POSIX (donc intégré dans toutes les distributions Unix, Linux, etc.)

Il existe de nombreuses versions dérivées (Gawk, Mawk, Tawk, etc.)

Le langage de script de awk

```
awk -f script.awk fichier.txt
```

Un script awk est une suite d'actions gardées de la forme :

```
motif { action }
```

- Un script awk lit un fichier ligne par ligne.
- Chaque ligne est découpée en mots selon un séparateur.
- Les actions gardées sont exécutées sur chaque ligne si le motif spécifié dans la garde existe dans la ligne

Dans la terminologie de awk, une ligne est un enregistrement et un mot est un champs.

La syntaxe des scripts est proche du langage C.

Quelques variables préféfinies

La ligne courante est `$0`

Le `i`ème mot se nomme `$i`

- `FS` : séparateurs de mots (espace ou tabulation par défaut)
- `OFS` : séparateurs de mots pour la sortie
- `NS` : séparateurs de lignes (retour-chariot par défaut)
- `NR` : numéro de la ligne courante
- `NF` : nombre de mots dans la ligne courante
- `FILENAME` : nom du fichier

Un premier exemple

Une action sans motif :

```
{ print "La moyenne de " $1 " est de " ($2 +$3)/2 }
```

A lancer sur le fichier :

```
Joe     18 5
Ludovic 1 11
Steph    8 4
Alex    16 8
Julie    16 18
```

Les motifs

Les motifs `awk` peuvent être soit :

- Des expressions régulières (entre *slash*) `/ regexp /`
- l'opération *match* (`~`), par exemple `$1 ~ /regexp/`
- l'opération *ne match pas* (`!~`), par exemple `$0 !~ /regexp/`
- Les mots-clés `BEGIN` ou `END`
- Une expression `exp` op `exp`, avec $op \in \{$`<`, `<=`, `==`, `!=`, `>=`, `>`$\}$
- Une combinaison booléenne avec `||`, `&&` et `!`
- `motif1,motif2` les lignes entre `motif1` et `motif2`

Expressions régulières

`.`	n'importe quel caractère
`^`	le début de la chaîne
`$`	la fin de la chaîne
`*`	zero ou plus
`+`	au moins une fois
`?`	zero ou une fois
`[abcd]`	une liste
`[a-z]`	un intervalle
`[[:upper:]]`	la classe des lettres majuscules
	(autres classes `[:alnum:]`, `[:alpha:]`, `[:blank:]` `[:digit:]`, `[:lower:]`, `[:space:]`)

Exemples

```
awk ‘$0 ~ /^[a-d]/’ fichier
awk ‘$0 !~ /^[a-d]/’ fichier
```

Structure d'un script

Un script est composé de trois parties :

- La section initiale dont les actions sont effectuées *avant* la lecture du fichier

  ```
  BEGIN { ... }
  ```

- La section terminale qui est traitée *après* que le fichier soit fermé (et traité)

  ```
  END { ... }
  ```

- La section des commandes gardées

  ```
  motif1 { ... }
  motif2 { ... }
  ...
  ```

Exécution d'un script

1. Exécuter les actions initiales
2. Pour chaque ligne du fichier, exécuter toutes les actions gardées dont la garde est vraie.
3. Exécuter les actions finales

Variables

Il y a trois types de variables dans `awk`

- système : non modifiables
- champs : désignent les champs d'un enregistrement
- utilisateurs : définies par l'utilisateur

Les variables non initialisées contiennent `0` (pour les variables numériques) et `""` pour les chaînes de caractères.

Instructions

Alternatives	Boucle while	Boucle for

`if (condition)`
`{ ... }`
`else`
`{ ... }`
	`while (condition)`
	`{ ... }`
		for (init ;cond ;incr)
		`{ ... }`

Affectations =, +=, etc.

Affichage

```
print exp exp ...
printf (format,exp,exp,...)
```

Tableaux

Tableaux unidimensionnels

Toutes les cases contiennent `0` par défaut (ou `""`)

`t[i]` pour accéder au `i`ème élément d'un tableau `t`

`delete(t)` pour supprimer un tableau

`delete(t[i])` pour supprimer un élément de tableau

`awk` permet également de manipuler des tableaux associatifs dont les indexs sont des chaînes de caractères

```
age["toto"] = 12

for (nom in age)
{ ... }
```

Quelques fonctions de traitement de chaînes

`gsub(r,s,t)`	sur la chaine t, remplace toutes les occurrences de r par s
`length(s)`	longueur de la chaine s
split(s,a,fs)	split s dans le tableau a avec le délimiteur fs, retourne le nombre de champs
sub(r,s,t)	comme gsub, mais remplace uniquement la première occurence
substr(s,i,n)	sous chaine de s commencant en i et de taille n

TP1

http://www.lri.fr/~conchon/OMED/1/tp/