Le filtre AWK

Fonctionnalités et origine

Le nom de l'utilitaire awk vient de ses 3 auteurs Aho, Weinberger et Kernighan. Il peut être classé dans la catégorie des filtres. Mais c'est beaucoup plus qu'un utilitaire de gestion des fichiers textes, il intègre un langage interprété très voisin du C.
La version disponible sur Linux s'appelle gawk =GNU awk; il s'agit d'une version plus performante.
Cet utilitaire s'applique à un fichier, ou à un flot de données provenant de son entrée, si le fichier est absent.

awk lit un texte dans un fichier, le modifie et envoie le résultat vers la sortie standard

Il décompose chaque ligne (enregistrement) en champs, les champs étant des chaînes de caractères séparées les unes des autres par un caractère particulier appelé le séparateur.
Ce caractère de séparation peut être fourni explicitement, sous la forme de l'option **-F**, par exemple (-F":"), ou invoqué dans le motif d'une expression rationnelle.

Le programme awk est une suite d'action de la forme : motif { action } , le motif permet de déterminer sur quels enregistrements est appliquée l'action.

Un **enregistrement** est :
une chaîne de caractères séparée par un retour chariot, en général une ligne.
Un **champ** est :
une chaîne de caractères séparée par un espace (ou par le caractère spécifie par l'option -F), en générale un mot.

On accède à chaque champ de l'enregistrement courant par la variable $1, $2, ... $NF.
$0 correspond à l'enregistrement complet. La variable NF contient le nombre de champ de l'enregistrement courant, la variable $NF correspond donc au dernier champ.

Syntaxe

awk [-Fs] 'motif { action } ' fichier

Les variables prédéfinies

FNR Numéro de l' enregistrements courant dans le fichier
FS séparateur de champs en entrée, "" par défaut
NF nombre de champs de l'enregistrement courant
NR Numéro de l' enregistrements courant dans l'entrée standard
OFS séparateur de champs pour la sortie , "" par défaut

ORS séparateur d'enregistrements pour la sortie , "\n" par défaut
RS séparateur d'enregistrements en entrée , "\n" par défaut

Syntaxe du motif

Si le **motif** existe dans l'enregistrement, l'action sera appliquée à la ligne .
Le motif peut être :

Une expression régulière :

/expression régulière/
L'action est appliquée pour chaque ligne contenant une chaîne satisfaisant à l'expression régulière.

Exemple:
```
awk  '/bonjour/ {print $0}' file
```
La ligne n'est affichée que si elle contient la chaîne bonjour.

```
awk –F":" '/^r/ {print $0}' file
```
Affiche les lignes qui commencent par r

```
$n  ~/expression régulière/
```
L'action est appliquée si le champ **n** satisfait l'expression régulière.

```
$<n> !~/expression régulière/
```
L'action est appliquée si le champ **n** ne satisfait pas l'expression régulière.

Exemples:
`$1 ~/bonjour/ {print $0}` La ligne est affichée si $1 contient bonjour

`$1 !~/bonjour/ {print $0}` La ligne est affichée si $1 ne contient pas bonjour

Section BEGIN

La section BEGIN est introduite par le critère BEGIN, l'action associée n'est exécutée qu'une fois avant de traiter le 1er enregistrement.

Section END

La section END est introduite par le critère END, l'action associée n'est exécutée qu'une fois après avoir traiter le dernier enregistrement.

`Awk 'END {print NR}' /etc/group` affiche le nombre de lignes de /etc/group

Une expression de comparaison: <, <=, == , !=, >=, >

Une combinaison à l'aide des opérateurs booléens || ou, && et, ! négation)

exemples

1) Soit le fichier adresse suivant (nom, nuémro_de_téléphone_domicile, numéro_de_portable, numéro_quelconque):

Samir | 011 33 33 33 | 069 12 34 56 | 25
Said | 012 22 22 22 | 077 12 34 56 | 70
Kamal | 033 11 11 11 | 088 12 34 56 | 40
Souad | 044 00 00 00 | 099 12 34 56 | 67

On vérifie que dans le fichier le numéro_de_téléphone_domicile (champ 2) et le numéro_de_portable sont bien des nombres :

```
awk 'BEGIN { print " On vérifie les numéros de téléphone"; FS="|"}
$2 !~ /^[0-9][0-9]*$/ { print "Erreur sur le numéro de téléphone domicile, ligne
"NR": \n" $0} $3 !~ /^[0-9][0-9]*$/ { print "Erreur sur le numéro de téléphone
portable, ligne "NR": \n" $0}
END { print "Vérification terminée" } '  adresse
```

2)

```
 awk  'BEGIN { print "Verification des UID et GID dans le fichier /etc/passwd"; FS=":"}
   $3 !~ /^[0-9][0-9]*$/ {print "UID erreur ligne "NR" :\n"$0 }
   $4 !~ /^[0-9][0-9]*$/ {print "GID erreur ligne "NR" :\n"$0 }
   END { print "Fin" } '    /etc/passwd
```

3)

```
awk '
         BEGIN { print "Verification du fichier /etc/passwd pour ...";
         print "- les utilisateurs avec UID = 0 " ;
         print "- les utilisateurs avec UID >= 60000" ;
         FS=":"}
     $3 == 0 { print "UID 0 ligne "NR" :\n"$0 }
     $3 >= 60000 { print "UID >= 60000 ligne "NR" :\n"$0 }
     END { print "Fin" }
' /etc/passwd
```

4)

```
awk 'BEGIN { print "Verification du fichier /etc/group";
          print "le groupe 20 s'appelle t-il bien users ? " ;
          FS=":"}
     $1 == "users" && $3 ==20 { print "groupe "$1" a le GID "$3" !" }
     END { print "Fin" }
' /etc/group
```

Syntaxe de l'action

Une action transforme ou manipule des données. par défaut print

type des actions

affectation, impression ou concaténation de chaînes de caractères

```
awk –F":" '{OFS=":"} /^s/ {$4= "33"; print $0}' /etc/passwd
```
Affiche les lignes commençant par s après avoir avoir affecté la valeur 33 au champs 4.

```
ls –l| awk   '/^d/   { $9="Repertoire: "  $9  ;  print $9, $1, $3}'
```
Afficher le nom, les autorisations et le propriétaire des sous-répertoires du répertoire courant

```
awk –F":" 'BEGIN {OFS=":"} $7!= "/bin/false" {print $0} $7 =="/bin/false"
{$7="/bin/posix/sh"; print $0 }' /etc/passwd >file
```

On crée un nouveau fichier de mot de passe file en remplacant le shell /bin/false par /bin/ posix/sh

concaténation de chaînes de caractères

Il n'y a pas d'opérateur de concaténation, il faut simplement lister les chaînes à concaténer.

```
awk '{ print NR " : " $0 }' fichier
```
On numérote les lignes du fichier