



[Vous avez trouvé cette page utile, Vous pouvez l'obtenir ici au format kindle \(amazon\).](#)

La commande awk

- [Les variables prédéfinies](#)
- [Syntaxe du motif](#)
- [Syntaxe de l'action](#)
 - [Fonctions numériques](#)
 - [Les fonctions sur les chaines de caractères](#)
 - [Les opérations arithmétiques et les affectations](#)
 - [Les variables et expressions](#)
 - [Les opérations arithmétiques](#)
 - [Les variables de champs](#)
 - [concaténation de chaines de caractères](#)
 - [while](#)
 - [for](#)
 - [breack, continue](#)
 - [if, else](#)
 - [commentaire et action vide](#)
 - [next, exit](#)
 - [affichage](#)
 - [tableau](#)
 - [for et les tableau](#)
 - [simulations des tableau multidimensions](#)

Cette commande permet d'appliquer un certain nombre d'actions sur un fichier. La syntaxe est inspirée du C

syntaxe

awk [-Fs] [-v variable] [-f fichier de commandes] 'program' fichier

- F Spécifie les séparateurs de champs
- v Définit une variable utilisée à l'intérieur du programme.
- f Les commandes sont lu à partir d'un fichier.

principe de fonctionnement

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 champs est :

une chaîne de caractères séparée par un espace (ou par le caractère spécifié par l'option -F), en générale un mot.

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

Exemples

<code>awk -F ":" '{ \$2 = "" ; print \$0 }' /etc/passwd</code>	imprime chaque ligne du fichier /etc/passwd après avoir effacé le deuxième champs
<code>awk 'END {print NR}' fichier</code>	imprime le nombre total de lignes du fichiers
<code>awk '{print \$NF}' fichier</code>	imprime le dernier champs de chaque ligne
<code>who awk '{print \$1,\$5}'</code>	imprime le login et le temps de connexion.
<code>awk 'length(\$0)>75 {print}' fichier</code>	imprime les lignes de plus de 75 caractères. (print équivalent à print \$0)

Les variables prédéfinies

Variable	Signification	Valeur par défaut
ARGC	Nombre d'arguments de la ligne de commande	-
ARGV	tableau des arguments de la ligne de commande	-
FILENAME	nom du fichier sur lequel on applique les commandes	-
FNR	Nombre d'enregistrements du fichier	-
FS	séparateur de champs en entrée	" "
NF	nombre de champs de l'enregistrement courant	-
NR	nombre d'enregistrements déjà lu	-
OFMT	format de sortie des nombres	"%.6g"
OFS	séparateur de champs pour la sortie	" "
ORS	séparateur d'enregistrement pour la sortie	"\n"
RLENGTH	longueur de la chaîne trouvée	-
RS	séparateur d'enregistrement en entrée	"\n"
RSTART	début de la chaîne trouvée	-
SUBSEP	séparateur de subscript	"\034"

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/

- \$0 ~ /expression régulière/
- expression ~ /expression régulière/
- expression !~ /expression régulière/
- une expression BEGIN ou END
- une expression de comparaison: <, <=, ==, !=, >=, >
- une combinaison des trois (à l'aide des opérateurs booléens || *ou*, && *et*, ! *négation*)
- une caractérisation des lignes
motif1,motif2 : chaque ligne entre la première ligne correspondant au motif1 et la première ligne correspondant au motif2

exemples

```
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
```

Résultat :

Verification des UID et GID dans le fichier /etc/passwd
 UID erreur ligne 14 :
 clown:*.aaa:b:utilisateur en erreur:/home/clown:/bin/sh
 GID erreur ligne 14 :
 clown:*.aaa:b:utilisateur en erreur:/home/clown:/bin/sh
 Fin

```
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
```

Résultat :

Verification du fichier /etc/passwd pour ...
 - les utilisateurs avec UID = 0
 - les utilisateurs avec UID >= 60000
 UID 0 ligne 5 :
 root:*.0:b:administrateur:./bin/sh
 UID >= 60000 ligne 14 :
 clown:*.61000:b:utilisateur en erreur:/home/clown:/bin/sh
 Fin

```
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
```

Résultat :

Verification du fichier /etc/group
 le groupe 20 s'appelle t-il bien users ?
 groupe users a le GID 20 !

Fin

```
awk 'NR == 5 , NR == 10 {print NR" : " $0 }' fichier
```

Imprime de la ligne 5 à la ligne 10 , chaque ligne précédée par son numéro

Syntaxe de l'action

Une action transforme ou manipule des données. par défaut *print*
type des actions

- fonctions prédéfinies, numérique ou chaîne de caractères
- contrôle de flux
- affectation
- impression

Fonctions numériques

Nom des fonctions	signification
atan2(y,x)	arctangente de x/y en radians dans l'intervalle -pi pi
cos(x)	cosinus (en radians)
exp(x)	exponentielle e à la puissance x
int(x)	valeur entière
log(x)	logarithme naturel
rand()	nombre aléatoire entre 0 et 1
sin(x)	sinus (en radians)
sqrt(x)	racine carrée
srand(x)	reinitialiser le générateur de nombre aléatoire

Les fonctions sur les chaînes de caractères

Dans le tableau suivant :

s et t représente des chaînes de caractères

r une expression régulière

i et n des entiers

Nom des fonctions	signification
gsub(r,s,t)	sur la chaîne t, remplace toutes les occurrences de r par s
index(s,t)	retourne la position la plus à gauche de la chaîne t dans la chaîne s
length(s)	retourne la longueur de la chaîne s
match(s,r)	retourne l'index où s correspond à r et positionne RSTART et RLENGTH
split(s,a,fs)	split s dans le tableau a sur fs, retourne le nombre de champs
sprintf(fmt,liste expressions)	retourne la liste des expressions formatées suivant fmt

sub(r,s,t)	comme gsub, mais remplace uniquement la première occurrence
substr(s,i,n)	retourne la sous chaîne de s commençant en i et de taille n

Les variables et expressions

Les opérations et affectations arithmétiques

- Les opérateurs arithmétiques sont les opérations usuelles : + - * / % (reste division entière) et ^ (puissance). Tous les calculs sont effectués en virgule flottante.
- La syntaxe de l'affectation : var = expression
Vous pouvez aussi utiliser les opérateurs +=, -=, *=, /=, %= et ^= (x+=y équivaut à x=x+y)

Les variables de champs

Rappel : Les champs de la ligne courant sont : \$1, \$2, ..., \$NF

La ligne entière est \$0

Ces variables ont les mêmes propriétés que les autres variables. Elles peuvent être réaffectées. Quand \$0 est modifiée, les variables \$1,\$2 ... sont aussi modifiées ainsi que NF. Inversement si une des variables \$i est modifiée, \$0 est mise à jour.

Les champs peuvent être spécifiés par des expressions, comme \$(NF-1) pour l'avant dernier champs.

exemple

```
awk 'BEGIN { FS=":" ;
           OFS=":" }
     $NF != "/bin/ksh" { print $0 }
     $3 == "/bin/ksh" && NF == 7 { $7 = "/bin/posix/sh" ;
                                print $0 } '
```

/etc/passwd > /etc/passwd.new

Résultat :

On crée un nouveau fichier de mot de passe /etc/passwd.new en remplaçant le shell /bin/ksh 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.

exemples:

```
awk '{ print NR " : " $0 }' fichier
```

Résultat :

On numérote les lignes du fichier

```
awk 'BEGIN { FS=":" ;
           OFS=":" ;
```

```

    print " Run Level 2 : Liste des actions "
    $2 ~ /2/ { print "Keyword <<$3>>, \n Tache <<$4>>" }
    $2 == "" { print "Keyword <<$3>>, \n Tache <<$4>>" }
' /etc/inittab > /etc/passwd.new

```

Résultat :

Affiche les actions executées lors du passage à l'état 2

while

Syntaxe: while (condition) action

for

break, continue

Break: sortie de boucle

Continue: commence une nouvelle itération de la boucle.

if , else

Syntaxe:

if (expression) action

else action

commentaire et action vide

Le commentaire est précédé par #. tout ce qui est entre # et la fin de la ligne est ignoré par awk

Une action vide est représenté par ;

next, exit

Next: passe à l'enregistrement suivant. On reprend le script awk à son début

Exit: ignore le reste de l'entrée et execute les actions définie par END

affichage

print exp, exp ou print (exp , exp) affiche les expressions

print equivaut à print \$0

printf format , exp, exp ou printf (format,exp , exp) identique à print mais en utilisant un format (voir printf en C)

Un format est une chaine de caractères et des constructeurs commençant par %
specifieur signification

d nombre decimal

s chaine de caractères

specifieur signification

- expression justifiée à gauche

largeur largeur d'affichage

`.precision` longueur maximale d'une chaîne de caractères
ou nombre de décimales

Exemple:

La sortie d'un `print` ou d'un `printf` peut être redirigée dans un fichier ou sur un pipe
Les noms de fichiers doivent être entre guillemets sinon ils sont considérés comme des variables

Exemple:

`awk ' { print NR " :", $0 >` le fichier *fic.h.numerote* contient le fichier
`"fic.h.numerote" } ' fichier` *fichier* avec les lignes numérotées

`awk ' { printf "%3d : %s " , NR , $0` le fichier *fic.h.numerote* contient le fichier
`> "fic.h.numerote" } ' fichier` *fichier* avec les lignes numérotées sur 3 caractères

tableau

On peut utiliser des tableaux de chaînes de caractères et de nombres à une dimension

Il n'est pas nécessaire de les déclarer. La valeur par défaut est "" ou 0 .

Les indices sont des chaînes de caractères.

```
awk 'BEGIN { print "Mémorisation de votre fichier " FILENAME }
      {memfile [NR] = $0 }
      END   { for ( i = NR ; i >= 1 ; i-- ) {
                print i ":" memfile[i]
              }
              print "Fin"
            } ' fichier
```

Résultat :

Affiche le fichier en commençant par la dernière ligne

```
awk ' NF > 0 {
      for (i=1;i<=NF;i++) {
        if ( $i ~ /^[0-9a-zA-Z][0-9a-zA-Z]*$/ ) {
          index[$i] = index[$i] " " NR " " i " " ;
          index["total"]++ ;
        }
      }
      END   { x="total" ;
              printf("%s mots détectés = %d\n",x,index[x]);
            } ' fichier
```

Résultat :

Construction d'un index de cross reference

for et les tableaux

Comme les indices des tableaux sont des chaînes de caractères, on ne peut pas déterminer la taille d'un tableau

On doit donc utiliser la construction :

```
for (var in tableau)
action
```

```
awk ' NF > 0 {
      for (i=1;i<=NF;i++) {
```

```

        if ( $i ~ /^[0-9a-zA-Z][0-9a-zA-Z]*$/ ) {
            index[$i] = index[$i] ";" NR "," i " " ;
            index["total"]++ ;
        }
    }
END
{
    for ( x in index ) {
        if ( x != "total" )
            printf("%-20s\t%s\n",x,index[x]) | "sort -f "
        }
        x="total";
        printf("%s mots detectés = %d\n",x,index[x]);
    } ' fichier

```

Résultat :

Construction d'un index de cross reference

simulations des tableaux multidimensionnels

On ne peut pas utiliser des tableaux multidimensionnels.

On peut les simuler en concaténant des composants des sous chaînes avec le séparateur SUBSEP

```

awk 'BEGIN { print "Mémorisation de votre fichier " FILENAME
        SUBSEP=":"
    }
    {
        for ( i=1 ; i <=NF ; i++ ) {
            memfields[ NR , i ] = $i
        }
    }
END {
    for ( i in memfields ) {
        print i ":" memfields[i] | "sort -n -t: "
    }
    print "Fin"
} ' fichier

```

Résultat :

Affiche le fichier en commençant par la dernière ligne

[Obtenir l'aide mémoire Unix au format pdf](#)



© [Isabelle Vollant](http://www.shellunix.com) (<http://www.shellunix.com>)

mise à jour le 30/07/12