Introducción a AWK

Grupo de usuarios de Linux

Univ. Carlos III de Madrid

8 de abril de 2003

Contenido de la charla

- Introducción
- ¿Como funciona?
- Una linea en AWK
- Variables predefinidas
- Procesando los datos
- Expresiones
- La salida
- Sentencias de control
- Imitando otros programas
- Bibliografía
- Dudas

Introducción

- ¿Quien?
  AWK debe su nombre a sus diseñadores: Alfred. V. Aho, Peter J. Weinberger y Brian W. Kernighan.

- ¿Cuándo? ¿Dónde?
  La versión original fue escrita en 1977 en los laboratorios de AT&T
  En 1985 apareció una nueva versión mas potente con soporte de expresiones regulares, funciones definidas por el usuario y múltiples streams de entrada.
  Un año después apareció la versión GNU implementada por Paul Robin y Jay Fenlason aconsejados por Richard Stallman.

¿Como funciona?

Cada linea de código de AWK consta de dos partes. El patrón de búsqueda y las acciones. Las acciones solo se ejecutaran si el patrón es cierto para esa linea.

patrón{acción}

AWK lee linea a linea un archivo o la entrada estándar y lleva a cabo cada una de sus lineas de código sobre él.

$$
\begin{gathered}
linea_1 \rightarrow CodigoAwk \rightarrow Salida_1 \\
linea_2 \rightarrow CodigoAwk \rightarrow Salida_2 \\
\vdots \\
linea_n \rightarrow CodigoAwk \rightarrow Salida_n
\end{gathered}
$$

Tipos de patrones

Hay 5 formas de utilizar los patrones, si estos coinciden para el registro que se procesa se llevan a cabo las acciones. Estos patrones son los de la siguiente tabla.

/expresión regular/	Comprueba que la expresión regular encaje en la linea.
expresión	Comprueba que la expresión sea verdadera.
pat1,pat2	Acepta como correcto, todo lo que se encuentre entre el patrón 1 y el patrón 2, esto se repite hasta el final del texto.
BEGIN/END	Son patrones especiales que se ejecutan solo una vez, al principio y al final del script respectivamente.
null	Entiende todos los registros como correctos.

Explicación de una linea AWK

```
/alumnos.uc3m.es/{print $0} mails.de.mis.amigos
```

/alumnos.uc3m.es/: Cuando se encuentre este patrón se ejecutaran las acciones delimitadas entre { y }. Admite expresiones regulares.

{print $0}: Ejecuta las acciones, en este caso "print".

$0: El 0 simboliza la linea completa y los números 1,2,...,n simbolizan cada uno de los campos que encuentra separados por cierto caracter o caracteres, por defecto es el caracter de espacio.

mails.de.mis.amigos: Archivo que se filtra, también puede sustituirse este archivo por la entrada estándar.

AWK en una sola linea

En muchas ocasiones solo se necesita hacer algo puntual que se puede expresar en una única linea de código, en estos casos se puede hacer directamente en nuestra consola de este modo:

`awk 'patrón{acción}' archivo.a.filtrar`

Ej 1: `awk '/alumnos.uc3m.es/{print $0 }' mails.de.mis.amigos`

En este ejemplo se imprimirán todas las lineas que contenga el patrón.

Ej 2: `ls -l | awk '$8 == "curso.dvi"{print $0}'`

En este ejemplo se imprimirán solo las lineas en que el 8º campo sea exactamente `"curso.dvi"`.

Variables predefinidas

Las variables predefinidas, son aquellas que AWK define al principio de la ejecución, y con las que trabaja. Es útil e importante saber manejar estas variables y saber que información almacena cada una.

FS	Separador de campo.
RS	Separador de registro.
IGNORECASE	Controlador de sensibilidad a mayúsculas.
OFS	Separador de campos de salida.
ORS	Separador de registro de salida.
ARGC	Contador de parámetros.
ARGV	Contenedor de parámetros.
FNR	Numero de registro en el fichero actual.
NF	Numero de campo del registro actual.
NR	Numero de registro.

Procesando los datos

La entrada se procesa por registros. Que se separan unos de otros por un separador de registro, declarado internamente como RS"(Record Separator). Por defecto el valor de RS es el salto de linea. El registro actual se almacena en la variable $0

A su vez, los registros son procesados en campos. Que se separan por el separador de campo, declarado internamente como "FS"(Field Separator). Por defecto el valor de FS es un numero indefinido de espacios o el tabulador. Cada campo del registro actual se almacena en la variable $1,$2,...,$n según su posición.

Los registros son procesados uno a uno, y todo el código AWK es ejecutado en cada registro. Es importante tener esto en cuenta para predecir la salida.

Existen dos excepciones, las lineas de código con el patrón BEGIN y END, que solo se ejecutan una vez, al principio y al final del programa respectivamente.

Getline

Hemos visto como procesar datos desde la entrada estándar y desde un archivo, pero a veces esto no es suficiente, y para eso esta la función getline, que permite leer archivos o salidas de comandos durante la ejecución del programa.

El getline, debido a su complejidad no se explicara en esta charla pero podrá encontrarse documentada en las lecturas sugeridas al final del libro.

Operadores de Aritméticos

Estos operadores se usan para realizar las operaciones de calculo aritmético básico

x + y	Suma x e y
`x - y`	Resta x e y
`x * y`	Multiplica x e y
`x / y`	Divide x e y
`x % y`	Haya el resto de x entre y
`x ^ y`	Eleva x a y
`x ** y`	Eleva x a y

Operadores de Asignación

Estos operadores se usan para asignar valores a las variables.

`x = y`	Asigna a la variable x el valor de y
x += y	Asigna a la variable x el valor de y + x
`x -= y`	Asigna a la variable x el valor de y - x
`x *= y`	Asigna a la variable x el valor de y * x
`x /= y`	Asigna a la variable x el valor de y / x
`x %= y`	Asigna a la variable x el valor de y % x
`x ^= y` `x **= y`	Asigna a la variable x el valor de x ** y Asigna a la variable x el valor de x ** y

Operadores de comparación

Estos operadores, como su nombre indica se usan para comparación, y devuelven TRUE en caso de ser cierto y FALSE en caso de no serlo.

`x < y`	Verdadero si x es menor que y
`x <= y`	Verdadero si x es menor o igual que y
`x > y`	Verdadero si x es mayor que y
`x >= y`	Verdadero si x es mayor o igual que y
`x == y`	Verdadero si x es igual a y
`x != y`	Verdadero si x es diferente de y
`x ~ y`	Verdadero si en x encaja la exp. regular y
`x !~ y`	Verdadero si en x no encaja la exp. regular y

Operadores Booleanos

Estos operadores se usan para construir expresiones booleanas que dan como resultado TRUE o FALSE. Se pueden convinar entre ellos para conseguir los resultados deseados.

`exp1 && exp2`	Solo es verdadero si ambas expresiones son verdaderas
`exp1 || exp2`	Es verdadero si alguna de las expresiones es verdadera
`!exp1`	Es verdadero si la expresión es falsa

Imprimiendo la Salida

Una de las funciones principales de AWK es hacer informes a partir de una cantidad enorme de datos. Por ello es muy importante la salida de los programas. Las principales funciones son:

- print: Imprime las cadenas que se le pasan como parámetro separadas por el "FS". (Se pueden concatenar cadenas escribiéndolas juntas)

- printf: Imprime las cadenas que se le pasan como parámetro con un formato especificado. El formato es el primer parámetro, todos los demás serán cadenas.

print

Esta es una función bastante sencilla, simplemente imprime las cadenas que se le pasan como parámetro.

Ej:

```
awk '/alumnos.uc3m.es/{print NR,
$1,$2,"Universidad Carlos III"}' mails.de.mis.amigos
```

Salida:

```
 1 200032565@alumnos.uc3m.es Carlos Universidad Carlos III
 3 200078541@alumnos.uc3m.es Pedro Universidad Carlos III
 5 200012345@alumnos.uc3m.es Maria Universidad Carlos III
 8 200054321@alumnos.uc3m.es Benito Universidad Carlos III
```

printf

Esta función trata de dar un formato mas elegante al texto de salida.

Ej:

```
awk '/alumnos.uc3m.es/{printf "%3.3d %-27s %10s %-s",
NR,$1,$2,"Universidad Carlos III\n"}' mails.de.mis.amigos
```

Salida:

```
 001 200032565@alumnos.uc3m.es       Carlos Universidad Carlos III
 003 200078541@alumnos.uc3m.es        Pedro Universidad Carlos III
 005 200012345@alumnos.uc3m.es        María Universidad Carlos III
 008 200054321@alumnos.uc3m.es       Benito Universidad Carlos III
```

Control de formato

El formato de salida de la función printf se controla a través de las letras precedidas por % y sus modificadores, del primer parametro como lo refleja esta tabla:

c	imprime un caracter
d,i	imprime un entero decimal
e	imprime en notación científica
f	imprime en notación punto flotante
g	imprime en notación científica o punto flotante, escoje la mas corta
o	imprime entero octal sin signo
s	imprime una cadena de caracteres
x	imprime un entero hexadecimal sin signo (letras mayúsculas)
X	imprime un entero hexadecimal con signo (letras mayúsculas)

Modificadores de formato

Dependiendo del tipo de formato los modificadores trabajan de una manera u otra, los modificadores son los siguientes:

-	Justifica el texto a la izquierda de la anchura especificada. Por defecto, el texto se justifica a la derecha
'ancho'	Es un numero que se coloca justo detrás del caracter % y establece el ancho minimo que ocupara el parámetro correspondiente
'.precisión'	Este numero establece la precisión con la que se debe imprimir un numero, y para una cadena, el numero máximo de caracteres que ocupará.

Redireccionar la salida

A veces se desea redireccionar la salida de las funciones print y printf a un archivo o a un comando, para esto se usan los operadores de redireccion. Estos operadores son:

>	Vuelca la salida a un archivo, si existe, borra el contenido y escribe la salida en él.
>>	Vuelca la salida un archivo, si existe, añade la salida el final del archivo.
|	Vuelca la salida a un comando.

Hay veces que conviene cerrar un archivo o un pipe a comando, para ello se utiliza la función close.

Sentencias de control

Como no, en AWK también existen las sentencias de control, que son las encargadas de realizar las iteraciones y las bifurcaciones. Estas son:

`while(exp)`	Crea un bucle que se reitera mientras exp == TRUE.
`do..while(exp)`	Es un bucle while que se ejecuta por lo menos una vez.
`for(exp1;exp2;exp3)`	Es un bucle que ejecuta la exp1 en su primera iteración, la exp3 al final de cada iteración y se repite mientras exp2 == TRUE;
`if(exp)..else`	Es una bifurcación, si exp == TRUE se ejecuta las acciones, si no, se ejecutan las acciones del "else", si existe.

Funciones implícitas

Son funciones definidas en el lenguaje que permiten gran variedad de operaciones, pero aunque las operaciones que pueden realizar estas funciones no fueran suficiente para cubrir las necesidades del problema, siempre se puede definir nuevas funciones para el programa awk.

Funciones definidas

Las funciones definidas por el usuario permiten dar un aspecto mas modular y organizado al programa AWK, y en la mayoría de los casos te permite ahorrar unas cuantas lineas. La definición de una función es tan simple como:

```
function nombre_de_la_funcion(parametro1,parametro2,...,
parametroN){accion1;accion2;...;accionN}
```

Y a la hora de ejecutar la función, basta con escribir el nombre de la función seguido de sus parámetros, de este modo:

```
nombre_de_la_funcion(parametro1,parametro2,...,parametroN);
```

Imitando otros programas

AWK es capaz de imitar a otros programas como grep, tr, nl, cat, sed... y un largo etcetera. Aquí van unos cuantos ejemplos simples de imitación.

grep	awk '/patrón/'
nl	awk '{print NR,$0}'
cat	awk '{print}'

Bibliografía

El lenguaje de programación AWK/GAWK. De Jesús Alberto Vidal Cortez.

- http://inicia.es/de/chube
- http://www.lawebdelprogramador.com

Curso de Linux de Ciberdroide

- http://www.ciberdroide.com/misc/novato/curso/awk1.html

Agradecimientos