Sistemas Operativos
2do Semestre 2010

awk

Wenceslao Palma M.
<wenceslao.palma@ucv.cl>

Aho, **W**einberger y **K**ernighan

Lenguaje de procesamiento y manejo de patrones

La estructura básica de un programa contiene patrones y acciones.

$awk 'programa' archivo(s)

Un programa tiene la forma:

patrón {acción}
patrón {acción}
.....

awk toma como entrada las líneas (una por vez) que componen un archivo.

Cada línea que concuerde con algún patrón es procesada de acuerdo a lo especificado en {acción}

El patrón o la acción son opcionales.

Si falta la acción por defecto se imprimen todas las líneas que concuerdan con el patrón.

Si se omite el patrón, la acción se realiza para cada linea de la entrada. Por ejemplo:

```
$awk '{print}' /etc/passwd
```

Lo anterior es equivalente (pero más lento) a:

```
$cat /etc/passwd
```

El 'programa' puede ser ejecutado a partir del contenido de un archivo.

```
$awk -f mi_programa archivo(s)
```

Campos

awk procesa cada línea dividiéndola en campos. Un campo es una cadena de caracteres que no sean blancos, separados por blancos o tabuladores.

El separador de campos se puede cambiar utilizando -F seguido del nuevo separador.

Los campos generados se nombran como $1, $2, $3.....$NF, donde éste último contiene la cantidad de campos.

```
$awf -F: '/\/home\// {print $1}' /etc/passwd

$awk -F: '{print $7}' /etc/passwd
```

La salida de awk se puede utilizar en pipes.

```
$awk -F: '{print $1}' /etc/passwd | sort
```

Impresion

Cada línea procesada tiene asociado un número, el cual se puede utilizar mediante la variable predefinida NR

```
$who | sort | awk '{print  NR,$1}'
```

Se puede obtener un control más completo sobre la salida con el uso de printf

```
$who | sort | awk '{printf “%d----%s\n”, NR, $1}'

$df | awk '{printf “%s\t%s\t%s\n”, $1,$5,$6}'
```

Patrones

$awk -F: '**$7== “/sbin/nologin”** {print $0}' /etc/passwd

Existen varias alternativas para saber si un campo es una cadena vacía

$2 == “”

$2 ~ /^$/

$2 !~/./

length($2) == 0

BEGIN y END

Son patrones especiales.

Las acciones de BEGIN se realizan antes de procesar la primera línea.

Las acciones de END se realizan luego de ser procesada la última línea.

```
$awk 'END {print NR}' /etc/passwd #es equivalente a
                                  # $cat /etc/passwd | wc -l


$awk '{s=s+$1} END { print s}' archivo
```

Variables predefinidas

FILENAME : nombre del archivo de entrada
FS : caracter separador de campo
NF : número de campos por línea
NR : número de la línea procesada
OFS : cadena separadora de campo de salida (blanco por defecto)
ORS : cadena separadora de línea de salida (nueva línea por defecto)
RS : caracter separador de línea de entrada (nueva línea por defecto)

Control de flujo

Buscando palabras repetidas contiguas en un archivo de texto:

```
#!/usr/bin/bash
awk ' NF>0 { for(i=1;i<NF;i++) if ($i==$(i+1)) print NR,$i}' $1
```

La estructura de control for tiene la siguiente forma:

```
for(expresión1;condición;expresión2)
        proposición
```

También existe una estructura while equivalente:

```
expresión1
while (condición){
        proposición
        expresión2
}
```

Y una estructura condicional:

```
if (condición)
    proposición1
else
    proposición2
```

Operadores de awk

= += -= *= /= %=
||
&&
!
> >= < <= == != ~ !~
+ -
* / %
++ --

Funciones predefinidas

cos(expr)	
exp(expr)	e^{expr}
getline()	lee la siguiente línea de entrada; 0 si es EOF, 1 eoc
index(s1,s2)	posición del string s2 en s1, 0 si no está.
int(expr)	parte entera.
length(s)	largo de s
log(expr)	log natural
sin(expr)	
substr(s,m,n)	substring de n caracteres de s, comienza en posición m.

Arreglos asociativos

En awk cualquier valor puede servir como un subíndice .

Considerando el sgte archivo:

```
pepe    200
maria   100
bianca  500
pablo   300
maria   200
pablo   600
bianca  400
```

el total por cada nombre será:

```
awk '{suma[$1] +=$2}
        END {for (name in suma) print name,suma[$name] }'
```

Paso de variables desde un shell script

Se utiliza la opción -v.

Por ejemplo:

```
#!/bin/bash

        username=”pepe”;
        awk -v nombre=$username -F: '{$1==nombre print $5,$6}' /etc/passwd;
```

Para retornar la salida de awk hacia un shell script::

```
#!/bin/bash

        username=”pepe”;
        salida=”$(awk -v nombre=$username -F: '{$1==nombre print $5,$6}' /etc/passwd)”;
        echo $salida;
```