Introducción a Sistemas Operativos: Usando el shell

Clips xxx
Francisco J Ballesteros

1. Líneas y campos

Tenemos toda una plétora de comandos que saben trabajar con ficheros suponiendo que tienen líneas y que cada una tiene una serie de campos. Muchos de los comandos que trabajan con líneas no requieren siquiera que tengan campos o cualquier otra estructura predeterminada. Vamos a continuar nuestro script de ejemplo centrándonos ahora en este tipo de tarea: procesar líneas y campos.

Teníamos ya, para nuestro fichero

```
shell /bin/sh
libdir /usr/lib
bindir /usr/bin
logdir /log
```

un comando capaz de escribir las líneas que corresponden a directorios:

```
unix$
unix$ egrep '^[a-z]+dir ' config
libdir /usr/lib
bindir /usr/bin
logdir /log
unix$
```

Ahora nos gustaría tener la lista de directorios mencionados en dichas líneas. Lo primero que tenemos que hacer es pensar que la entrada que tenemos es un fichero con líneas que tienen dos campos separados por blanco. En tal caso basta con utilizar un programa capaz de escribir el segundo campo. Hay muchas formas de hacer esto. Una es utilizar *cut(1)*. El flag `-f` de `cut` admite como argumento el número de campo que se desea, comenzando a contar desde `1`.¡Pero atención!

```
unix$ egrep '^[a-z]+dir ' config | cut -f2
libdir /usr/lib
bindir /usr/bin
logdir /log
unix$
```

Por omisión `cut` utiliza el tabulador como carácter separador de campos y, en nuestro caso, teníamos un espacio en blanco y no un tabulador separando los campos. En realidad queremos

```
unix$ egrep '^[a-z]+dir ' config | cut '-d ' -f2
/usr/lib
/usr/bin
/log
unix$
```

que utiliza la opción `-d` de `cut` para indicarle que el carácter escrito tras la opción es el que deseamos utilizar como separador. ¡Ya tenemos una lista de directorios!

No obstante, `cut` no es la mejor opción en la mayoría de los casos. Pensemos en obtener la lista de dueños de los ficheros del directorio actual... Sabiendo que podemos usar un listado largo en `ls` como en

```
unix$ ls -l config
-rw-r--r--  1 nemo  staff  58 Aug 28 11:27 config
```

podríamos intentar usar algo como

```
unix$ ls -l config | cut -f3
-rw-r--r--  1 nemo  staff  58 Aug 28 11:27 config
```

¡Pero `cut` escribiría todos los campos! El problema de nuevo es el separador entre un campo y otro.

El programa que menos problemas suele dar para seleccionar campos es *awk(1)*. Es un lenguaje de programación para manipular ficheros con aspecto de tabla, pero tiene muchos programas de una sóla línea que resultan útiles. En este caso,

```
unix$ ls -l config | awk '{print $3}'
nemo
```

consigue el efecto que buscamos. El programa significa "*en todas las líneas, imprime el tercer campo*".

Lo bueno de `awk` es que utiliza una expresión regular como separador de campos. Y puedes cambiarla escribiendo la que quieras como argumento de la opción `-F`:

```
awk '-F[: ]+' '{print $3}'
```

imprime el tercer campo suponiendo que un campo está separado de otro por uno o mas caracteres que sean bien un espacio en blanco o bien dos puntos.

Volviendo a nuestro problema, una vez tenemos los directorios mencionados en nuestro fichero de configuración, estaría bien tenerlos ordenados y sin tener duplicados en la lista. Para conseguirlo podemos utilizar el comando *sort(1)* que sabe ordenar líneas utilizando campos como clave (o la línea entera). Otro comando relacionado es `uniq(1)`, que elimina duplicados de una entrada ya ordenada. Así pues,

```
unix$ egrep '^[a-z]+dir ' config | awk '{print $1}' | sort | uniq
bindir
libdir
logdir
unix$
```

escribe los directorios ordenados y sin duplicados.

Esto resulta útil también al recolectar nombres o valores que aparecen en cualquier otro sitio. Por ejemplo, para obtener la lista de dueños de ficheros en el directorio actual:

```
unix$ ls -l * | awk '{print $3}' | sort -u
nemo
unix$
```

El flag `-u` de `sort` hace lo mismo que `uniq`. ¡Ya tenemos algo que podríamos incluso guardar en una variable de entorno!

```
unix$ owners=‘ls -l * | awk '{print $3}' | sort -u‘
unix$ echo $owners
nemo
unix$
```

De hecho, vamos a hacer justo con nuestros directorios, y ya podemos ver si existen o no...

```
unix$ dirs=‘egrep ’^[a-z]+dir ’ config | awk ’{print $1}’ | sort | uniq‘
unix$ for d in $dirs ; do
>    if test ! -d $d ; then
>        echo no dir $d
>    fi
>    done
unix$
```

¡Bien, todos los directorios existen!, ¡Problema resuelto!

Recuerda que es el shell el que escribe los "`>`" para indicarnos que es preciso escribir más líneas para completar el comando. Hemos usado el bucle `for` que vimos anteriormente y que ejecuta los comandos entre `do` y `done` para cada palabra que sigue a `in`. En cada iteración, la variable cuyo nombre precede a `in` toma como valor cada una de las palabras que siguen a `in`.

También hemos utilizado el comando `if` que ejecuta en este caso

```
test ! -d $d
```

como condición y, de ser cierta, ejecuta los comandos en el `then`. Aunque no lo hemos necesitado, es posible escribir

```
if ....
then
    ...
else
    ...
fi
```

como comando.

El comando `sort` merece que le dediquemos algo más de tiempo. Sabe ordenar la entrada asumiendo que son cadenas de texto o bien ordenarla segun su valor numérico (y de algunas otras formas). Por ejemplo,

```
unix$ seq 10 | sort
1
10
2
3
4
5
6
7
8
9
```

no produce el efecto que podrías esperar. Ahora bien...

```
unix$ seq 10 | sort -n
1
2
3
4
5
6
7
8
9
10
```

con la opción `-n`, `sort` ordena numéricamente.

Sea numérica o alfabéticamente, podemos pedir una ordenación en orden inverso:

```
unix$ seq 5 | sort -nr
5
4
3
2
1
unix$
```

Aquí utilizamos ambos flags para indicar que se desea una ordenación numérica y además en orden inverso.

Además podemos pedir a `sort` que utilice como clave para ordenar sólo alguno de los campos, por ejemplo

```
sort -k3,5
```

ordena utilizando los campos del 3 al 5 (contando desde uno). Consulta su página de manual en lugar de recordar todo esto.

¿Recuerdas el comando *du(1)*? !Podemos ver dónde estamos gastando el espacio en disco!

```
unix$ du -s * | sort -nr | sed 5q
23824    zx-spe
6088    wr_pic1.eps
6088    inkdump.eps
6088    clive_pic3.eps
5768    zx
```

Primero, hacemos que `du` liste el total (opción `-s`, *summary*) para todos los ficheros y directorios. Ordenamos entonces su salida numéricamente en orden inverso y nos quedamos con las 5 primeras líneas. Son los 5 ficheros o directorios en que estamos usando más disco. Ahora podemos pensar qué hacemos con ellos si necesitamos espacio...

¿Y si queremos los que menos espacio usan? Podríamos quedarnos con las últimas 5 líneas, por ejemplo:

```
unix$ du -s * | sort -nr | tail -5
```

utiliza *tail(1)* que escribe las últimas *n* líneas de un fichero. Aunque habría sido mejor no invertir la ordenación en sort:

```
unix$ du -s * | sort -n | sed 5q
```

El comando `tail` tiene otro uso para imprimir las últimas líneas pero empezando a contar desde el principio. Esto es útil, por ejemplo, para eliminar las primeras líneas de un fichero:

```
unix$ seq 5 | tail +3
3
4
5
unix$
```

También es posible seleccionar determinados campos y/o reordenarlos. Por ejemplo, con la opción `-l`, `ps` produce un listado largo...

```
unix$ man ps
...
     -l      Display information associated with the following keywords:
             uid, pid, ppid, flags, cpu, pri, nice, vsz=SZ, rss, wchan,
             state=S, paddr=ADDR, tty, time, and command=CMD.
...
```

Para cada item, `ps` produce una columna separada de las demás por espacio en blanco. Así pues podemos utilizar `awk` para imprimir el pid, tamaño de la memoria física (*resident set size*, o rss) y nombre de los procesos utilizando

```
unix$ ps -l | awk '{printf("%s\t%s\t%s\n", $2, $15, $9);}'
PID    CMD    RSS
448    -bash    372
519    acme    28
526    (fontsrv)    0
527    acme    104
528    acme    5576
534    9pserve    24
...
```

Como puedes ver, `awk` dispone de `printf`. Dicha función se utiliza prácticamente como la de C. En nuestro caso optamos por imprimir las columnas 2, 15, y 9 en ese orden. Utilizar `cut` sería más complicado dado que hay múltiples blancos entre un campo y el siguiente.

¿Qué proceso está utilizando más memoria? Bastar ordenar numéricamente por el tercer campo, de mayor a menor y quedarse con la primera línea.

```
unix$ ps -l | awk '{printf("%s\t%s\t%s\n", $2, $15, $9);}' | \
>    sort -nr -k3 | sed 1q
91136    acme    52920
unix$
```

Parece que `acme`, con el pid 91136.