Fichero Aclaratorio

Pablo Rodriguez Guillen

Profesor: Enrique García Salcines

Abril 2019

Ejercicio 1

En primer lugar comprueba que los argumentos introducidos por el usuario son correctos. Si no lo son se imprime un mensaje de error en el que se especifica el formato del comando para ejecutar el script correctamente.

Si la ejecución por parte del usuario es correcta, se cuentan los ficheros con extensión ".c" y ".h" usando el comando find con la opción -name y se imprime la información obtenida por pantalla.

Posteriormente, haciendo uso de un bucle for se itera por cada uno de los ficheros del directorio especificado. En cada iteración se obtienen el número de líneas y de caracteres y se imprime por pantalla dicha información. Esta impresión se ordena por el número de caracteres de mayor a menor de cada fichero, haciendo uso del comando sort.

Una ejecución del script se mostraría así:

```
➜  Practice_3 git:(master) ejercicio1.sh ejemploCarpeta
Tenemos 4 ficheros con extensión .c y 5 ficheros con extensión .h
El fichero ejemploCarpeta/d1/audit.h tiene 463 líneas y 18221 caracteres
El fichero ejemploCarpeta/d1/auditBackup.h tiene 463 líneas y 18221 caracteres
El fichero ejemploCarpeta/d1/ecoc.h tiene 76 líneas y 2377 caracteres
El fichero ejemploCarpeta/d1/auto_fs.h tiene 72 líneas y 2288 caracteres
El fichero ejemploCarpeta/d1/compat.h tiene 17 líneas y 370 caracteres
El fichero ejemploCarpeta/d2/prueba.c tiene 14 líneas y 233 caracteres
El fichero ejemploCarpeta/d1/smo_routine.c tiene 8 líneas y 104 caracteres
El fichero ejemploCarpeta/d1/svm_model_matlab.c tiene 6 líneas y 99 caracteres
El fichero ejemploCarpeta/d1/d4/act_func.c tiene 5 líneas y 92 caracteres
➜  Practice_3 git:(master)
```

Figure 1: Ejecución ejercicio1.sh

Ejercicio 2

Al igual que en el ejercicio 1 y en el resto de los de la práctica. Se comprueba que los argumentos son correctos y si no lo son se imprime un mensaje de error. También se asigna un valor por defecto al argumento opcional.

En el bucle for, se recorre cada uno de los ficheros del directorio especificado. En cada iteración se crean dos variables locales: LongitudUsuario y Ejecutable. El resto de parámetros a mostrar tienen una obtención directa mediante opciones del comando stat.

Al igual que en el ejercicio anterior, se hace uso de una tubería para ordenar la salida por pantalla. Cabe mencionar que se hace uso de la opción -t para definir un carácter separador (";" en este caso). La salida se ordena por el tamaño en bytes del fichero de menor a mayor.

La ejecución del script se mostraría así:

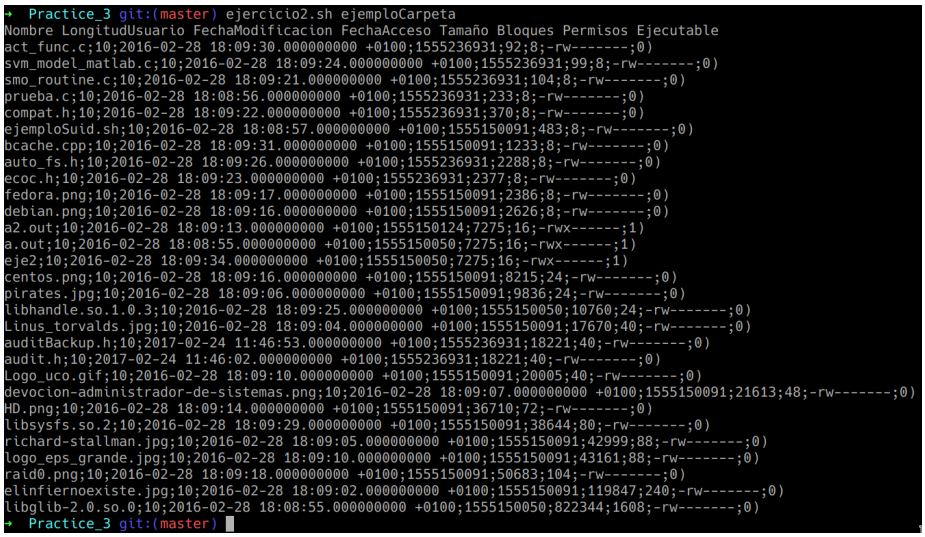


Figure 2: Ejecución ejercicio2.sh

Ejercicio 3

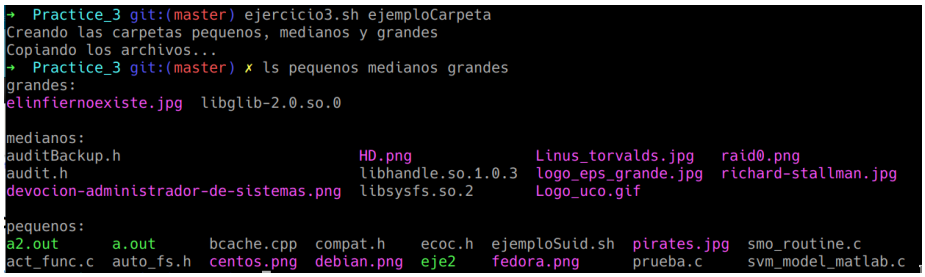
En primer lugar se realizan la comprobación ya mencionada de argumentos y la posterior asignación de valores por defecto a los argumentos opcionales. Estos, son los 2 números de bytes por los que se van a dividir los ficheros. Dado que uno debe ser mayor que el otro, se ha añadido un if que intercambie los valores

si el usuario introduce el mayor antes que el menor.

En segundo lugar, se procede a la creación de las carpetas "pequenos", "medianos" y "grandes" si estas carpetas no existían previamente. Si este fuera el caso, el script borra las carpetas y las vuelve a crear. Ambas acciones se imprimen por pantalla.

Por último, se hacen 3 bucles for que se diferencian por el uso de la opción -size del comando find en el que se copian a la carpeta que correspondan los ficheros que cumplan las condiciones del find.

Así se verían 2 ejecuciones del script, una con los valores por defecto y otra estableciendo el umbral inferior a 11000 bytes:



(a) Valores por defecto

(b) Umbral inferior 11000

Figure 3: Ejecución ejercicio3.sh

Ejercicio 4

Este script no cuenta con argumentos y utiliza como directorio sobre el que actuar, el directorio actual del usuario.

En primer lugar, usa el comando find con la opción -maxdepth=1 para contar el número de ficheros en el directorio actual. La información se imprime por pantalla. En segundo lugar, haciendo uso del comando awk, imprime los usuarios logueados en el sistema sin repetirlos (comando uniq).

Posteriormente, pide al usuario que introduzca un carácter. Si este no lo introduce en 5 segundos, se asigna a la variable char el carácter "a". El script, usando un bucle for y el comando find recorre todos los ficheros del directorio, buscando en su nombre el carácter introducido. Usando grep, wc y tuberías se obtiene el número de apariciones de un carácter. Se introducen los datos en un fichero temporal que se borra al finalizar el script porque no he conseguido guardar el valor en una variable.

A continuación se muestran un par de ejecuciones del script:

```
➜  ejemploCarpeta git:(master) ../ejercicio4.sh
El numero de ficheros en el directorio actual es 0
pablorg99
¿Qué carácter quieres contar?
El carácter a aparece 29 veces en nombres de ficheros o carpetas contenidos en la carpeta actual
➜  ejemploCarpeta git:(master) ../ejercicio4.sh
El numero de ficheros en el directorio actual es 0
pablorg99
¿Qué carácter quieres contar? d
El carácter d aparece 20 veces en nombres de ficheros o carpetas contenidos en la carpeta actual
```

Figure 4: Ejecuciones ejercicio4.sh

Ejercicio 5

Este script comprueba si existe la carpeta "Copia" en el directorio home del usuario. Si no existe la carpeta se crea.

En segundo lugar, recorre los ficheros de la carpeta "Copia" comparando la fecha del último acceso con la fecha actual, si la diferencia es superior a 200 segundos, se borra la copia en cuestión.

Por último, se crea una copia de seguridad de los ficheros que se pasan como argumento en el directorio citado usando el comando tar.

A continuación se muestran 2 ejecuciones del script:

```
➜  Practice_3 git:(master) ejercicio5.sh ejemploCarpeta
Creando el directorio ~/Copia
La copia de seguridad se ha creado correctamente.
➜  Practice_3 git:(master) ejercicio5.sh ejemploCarpeta
Borrando /home/pablorg99/Copia/copia-pablorg99-1555321303.tar.gz de 257 segundos...
La copia de seguridad se ha creado correctamente.
```

Figure 5: Ejecuciones ejercicio5.sh

Ejercicio 6

Tras comprobar que los argumentos son correctos, el script escribe en un fichero html el título y la cabecera del mismo. Cuando va a escribir el cuerpo, hace una llamada a la función listFile con el argumento pasado por el usuario.

En la función, se abre una enumeración html con la etiqueta <ul> y se recorren los ficheros del directorio pasado como argumento. Si el fichero de una iteración es un directorio, se escribe en negrita y se hace una llamada recursiva a listFile, si no, se escribe normal. Ambas escrituras se hacen con la etiqueta <li>. A continuación se muestran imágenes del fichero html creado tanto en el editor como en el navegador. El argumento pasado al script es "ejemploCarpeta":

```
<head>
<title>Listado directorios de ejemploCarpeta/</title>
</head>
<body>
<h1>Listado directorios de ejemploCarpeta/</h1>
<ul>
<li><strong>ejemploCarpeta/d3</strong></li>
<ul>
<li>ejemploCarpeta/d3/disco2.png</li>
<li>ejemploCarpeta/d3/eje2</li>
</ul>
<li><strong>ejemploCarpeta/images</strong></li>
<ul>
<li><strong>ejemploCarpeta/images/logos</strong></li>
<ul>
<li>ejemploCarpeta/images/logos/Logo_uco.gif</li>
<li>ejemploCarpeta/images/logos/logo_eps_grande.jpg</li>
</ul>
<li><strong>ejemploCarpeta/images/Carpeta1</strong></li>
<ul>
<li>ejemploCarpeta/images/Carpeta1/devocion-administrador-de-sistemas.png</li>
<li>ejemploCarpeta/images/Carpeta1/elinfiernoexiste.jpg</li>
```

Figure 6: Fichero html en editor

Listado directorios de ejemploCarpeta/

- **ejemploCarpeta/d3**
  - ejemploCarpeta/d3/disco2.png
  - ejemploCarpeta/d3/eje2
- **ejemploCarpeta/images**
  - **ejemploCarpeta/images/logos**
    - ejemploCarpeta/images/logos/Logo_uco.gif
    - ejemploCarpeta/images/logos/logo_eps_grande.jpg
  - **ejemploCarpeta/images/Carpeta1**
    - ejemploCarpeta/images/Carpeta1/devocion-administrador-de-sistemas.png
    - ejemploCarpeta/images/Carpeta1/elinfiernoexiste.jpg
    - ejemploCarpeta/images/Carpeta1/pirates.jpg
    - ejemploCarpeta/images/Carpeta1/Linus_torvalds.jpg
    - ejemploCarpeta/images/Carpeta1/richard-stallman.jpg
  - **ejemploCarpeta/images/Teoria**
    - ejemploCarpeta/images/Teoria/raid0.png
    - ejemploCarpeta/images/Teoria/fedora.png
    - ejemploCarpeta/images/Teoria/HD.png
    - ejemploCarpeta/images/Teoria/debian.png
    - ejemploCarpeta/images/Teoria/centos.png
    - ejemploCarpeta/images/Teoria/a2.out
- **ejemploCarpeta/d1**
  - ejemploCarpeta/d1/smo_routine.c
  - ejemploCarpeta/d1/svm_model_matlab.c
  - ejemploCarpeta/d1/audit.h
  - ejemploCarpeta/d1/auto_fs.h
  - ejemploCarpeta/d1/auditBackup.h
  - ejemploCarpeta/d1/compat.h
  - ejemploCarpeta/d1/ecoc.h
  - **ejemploCarpeta/d1/d4**
    - ejemploCarpeta/d1/d4/act_func.c
    - ejemploCarpeta/d1/d4/libsysfs.so.2
    - ejemploCarpeta/d1/d4/bcache.cpp
  - ejemploCarpeta/d1/libhandle.so.1.0.3
- **ejemploCarpeta/d2**
  - ejemploCarpeta/d2/a.out
  - ejemploCarpeta/d2/prueba.c
  - ejemploCarpeta/d2/libglib-2.0.so.0
  - ejemploCarpeta/d2/ejemploSuid.sh

Figure 7: Fichero html en navegador