Área de Arquitectura y Tecnología de Computadores
Universidad Carlos III de Madrid

uc3m | Universidad Carlos III de Madrid

SISTEMAS OPERATIVOS

Práctica opcional: bash scripts

Grado en Ingeniería en Informática
Doble Grado en Ingeniería en Informática y ADE
Curso 2018/2019

Javier López Gómez <jalopezg@inf.uc3m.es>

1. Introducción

Un shell script es un programa escrito usando mandatos del shell. En éstos está permitido el uso de variables y estructuras de control.

Su principal ventaja es que pueden ser portados de una máquina UNIX a otra siempre que sólo usen mandatos *IEEE POSIX 1003.2 —Shell and Utilities—*, mientras que los programas compilados (escritos en C, Pascal, etc.) deberían ser recompilados para otro sistema operativo/arquitectura. Otra ventaja es la comodidad de escritura.

El principal inconveniente es la lentitud de ejecución, que en muchos casos puede paliarse usando funciones incluidas en el intérprete (*builtins*) en lugar de ejecutar mandatos externos.

Un script puede ser ejecutado de la siguiente forma: `$ bash script.sh`.

Con el fin de hacer mantenible el script, es común encontrar comentarios[1] tanto en la cabecera como en diferentes partes del mismo. Típicamente, verá que la primera línea de un script es un comentario parecido a:

```
1 #!/bin/bash
```

En este caso, la ejecución de este fichero se tratará de forma diferente: si no es un binario soportado (ELF, COFF, a.out, etc.) y los dos primeros caracteres son `#!`, el kernel asume que es un script que será interpretado. La ruta del intérprete debería seguir después de la secuencia `#!` y hasta el fin de línea. En la jerga UNIX, esto se conoce como *shebang* y permite ejecutar su script simplemente como `$ ./script.sh`. Recuerde que en este caso, además del permiso de lectura, es necesario tener permiso de ejecución (ver manual de `chmod`), e.g. `$ chmod +x script.sh`.

En UNIX, un proceso que termina con la llamada a sistema `exit()` devuelve un entero de 8 bits que puede ser recogido por el padre (ver página de manual de `wait()`, macro `WEXITSTATUS(wstatus)`). Ésto se conoce como exit status o "estado de salida". Por convención, se devuelve 0 si la ejecución terminó con éxito y un número distinto de 0 si hubo un error. Consulte la sección EXIT STATUS de la página de manual de cada mandato para obtener más detalles acerca de los posibles valores devueltos.

Use la utilidad `man` para ver la página de manual de un mandato, e.g. `$ man grep`. Cuando vea algo como grep(1), se le está indicando que revise la página de manual de *grep* en la sección 1 del manual, i.e. `$ man 1 grep`. La mayoría de las veces puede omitir la sección.

Por último, se recomienda la lectura de la página de manual de `bash`, ya que además de documentación, podrá encontrar cosas que podrá usar a diario para ser más productivo (alias, history expansion, etc.).

1.1. Otras consideraciones

- Para depurar un script puede pasar la opción `-v` a bash:

```
1 #!/bin/bash -v
```

[1] El intérprete ignorará desde que encuentra el caracter # hasta el final de línea.

Alternativamente, puede usar `set -x` (ver manual de bash(1), sección SHELL BUILTIN COMMANDS).

- Tenga en cuenta que `bash` parte una línea de comando en palabras (o tokens); la primera palabra especifica el comando que será ejecutado. Por tanto, los espacios son significativos en muchos casos, e.g.

  esta asignación funcionará: `STRLEN=`echo $STR | wc -c``,

  pero ésta no: `STRLEN␣=`echo $STR | wc -c``, ya que en este caso, `STRLEN` se interpretará como el nombre de un mandato externo.

- En el caso de las sentencias de control, la forma correcta es usar ; para separar las partes (ver manual de bash(1), sección SHELL GRAMMAR, subsección Compound Commands), e.g.

```
1 if [ $# -gt 0 ]; then
2   ARGV1=$1;
3 fi
```

2. Condiciones de entrega

- La práctica se realizará en grupos de tres personas (máximo). La entrega se hará a través de Aula Global antes del **26 de Abril de 2019 a las 23:55**.

- Se entregará un único fichero, "`4_bash-scripts-1718_100xxxxxx.tar.gz`", conteniendo la solución a los ejercicios propuestos. Deberá reemplazar `100xxxxxx` por el NIA de la persona que realice la entrega. Sólo uno de los miembros del grupo deberá entregar el fichero.

  El tarball deberá contener los ficheros: `exercise1_a.sh`, `exercise1_b.sh`, `exercise2_b.sh`, `exercise3.sh`, `exercise4.sh`, `exercise5_c.sh` y `memoir.pdf`.

  La memoria (`memoir.pdf`) deberá incluir, al menos, la respuesta a los apartados 2.a, 5.a y 5.b. No olvide incluir el nombre y NIAs de los miembros del grupo.

- ¡Comente su código! Un script que no incluya comentarios obtendrá una calificación de 0.

¡NOTA IMPORTANTE! Puede hacer la práctica en su máquina local, en las aulas del Laboratorio de Informática o trabajar en `guernika.lab.inf.uc3m.es` (SSH). Las máquinas del Laboratorio de Informática tienen instalados los paquetes requeridos.

Ejercicio 1

En un sistema UNIX, las cuentas de usuario locales pueden encontrarse en el fichero `/etc/passwd`[2]. Este fichero contiene una línea por cada cuenta de usuario, con siete campos delimitados por ``:''. Para más información ver la página de manual passwd(5).

Se pide:

(a) Escriba un script que imprima por salida estándar el intérprete de comandos de cada usuario pasado como parámetro. La salida producida debería ser similar a:

```
$ ./exercise1_a.sh root nobody foo dbus
root: /bin/bash
nobody: /usr/bin/nologin
-ERROR- foo: no such user
dbus: /sbin/nologin
```

(b) Teniendo en cuenta que los grupos de usuarios locales están definidos en `/etc/group` (ver manual de group(5)), imprimir por salida estándar los miembros de un grupo[3] dado su GID (numeric group ID), e.g.

```
$ ./exercise1_b.sh 1
root
bin
daemon
```

Ejercicio 2

Tenga en cuenta el código copiado a continuación y responda a las preguntas de abajo.

```
1 #!/bin/bash
2 tr -c [:alnum:] [\\n\*] < $1 | sort | uniq -c | sort -nr | head -$2
```

(a) Describa cuál es el propósito del script y de cada una de las etapas del pipeline.

(b) Extienda el script para que éste compruebe que se ejecuta con dos argumentos, y en caso contrario, muestre un mensaje de instrucciones de uso, e.g.

```
$ ./exercise2_b.sh FOO
  Usage: ./exercise2_b.sh FILE NUM
```

Ejercicio 3

Escriba un script que imprima un listado de los números primos en el intervalo $[A, B]$. A y B serán pasados como parámetros, e.g.

[2]Típicamente, el campo de contraseña contiene una ``x''; en este caso, las contraseñas se almacenan en el fichero `/etc/shadow`.

[3]Tenga en cuenta que la salida podría variar dependiendo de su sistema.

```
$ ./exercise3.sh A B
```

Puede comprobar si un número es primo analizando la salida de la utilidad `factor` incluida en el paquete GNU coreutils (que ya debería estar instalado en su sistema). Dicha utilidad factoriza números enteros, e.g.

```
$ factor 2018
2018: 2 1009
```

A continuación se copia la salida esperada para una ejecución del script para el intervalo $[5, 14]$.

```
$ ./exercise3.sh 5 14
5
7
11
13
```

Ejercicio 4

Típicamente, un script de shell automatiza tareas repetitivas o de administración de sistemas. En algunos casos, es necesaria la ejecución de binarios externos que no están cubiertos por la norma *IEEE POSIX 1003.2 —Shell and Utilities—*, lo que limita la portabilidad de los scripts, y requiere la instalación previa de paquetes (dependencias).

En este ejercicio, se requerirá ImageMagick, un conjunto de herramientas para automatizar el trabajo con imágenes. Normalmente, estas herramientas ya se encuentran instaladas en la mayoría de distribuciones GNU/Linux como dependencia de algún otro paquete. Puede comprobar la disponibilidad de la utilidad `convert` incluida en el paquete ImageMagick, ejecutando: `$ convert -version`. Si el comando anterior no está disponible, instale ImageMagick de la siguiente manera[4]: `$ sudo apt-get install imagemagick`.

La utilidad `convert` puede usarse para convertir entre formatos de imágen, opcionalmente aplicando algún operador (cambiar tamaño, recortar, girar, etc.) antes de generar la salida. La guía rápida puede encontrarse en la página de manual de convert(1); la documentación completa está accesible en `http://www.imagemagick.org/`. Puede probar lo siguiente para convertir una imágen PNG a GIF, aplicando un filtro de desenfoque:

```
$ convert /usr/share/icons/gnome/256x256/apps/terminal.png -blur 2x2 \
 /tmp/terminal.gif
```

Mientras que `convert` escribe a un fichero diferente, `mogrify` reemplaza el fichero original, e.g.

```
$ mogrify -resize 128x128 /tmp/terminal.gif
```

Se pide escribir el script `exercise4.sh`, que cambiará el ancho de las imágenes JPEG (sufijo `.jpg`) a 720 pixels (alto proporcional) para aquellos ficheros cuyo tamaño sea mayor a 1 MiB y que estén contenidos en el directorio pasado como parámetro. Además, se deberá imprimir por salida estándar la ruta de cada fichero tocado, e.g.

[4] Válido para Debian GNU/Linux y distribuciones derivadas (Ubuntu, etc.).

```
$ cp -R /usr/share/backgrounds/gnome /tmp
$ ./exercise4.sh /tmp/gnome
/tmp/gnome/Flowerbed.jpg
/tmp/gnome/Godafoss_Iceland.jpg
/tmp/gnome/Signpost_of_the_Shadows.jpg
/tmp/gnome/Stones.jpg
```

Ejercicio 5

Lo habitual es que un script de shell interactúe con el usuario usando la entrada y salida estándar. Sin embargo, con `dialog` es posible presentar al usuario interfaces en modo texto (TUI) sencillas. Al igual que en el ejercicio anterior, `dialog` es una dependencia externa. Es posible que ya tenga `dialog` instalado en su sistema como dependencia de algún otro paquete; pruebe lo siguiente: `$ dialog --version`. Si el comando anterior no está disponible, instale `dialog` con: `$ sudo apt-get install dialog`. `dialog` permite presentar o solicitar información al usuario desde un script bash. Pruebe lo siguiente:

```
$ dialog --yesno '¿Le parece interesante?\n\n' 0 0
```

Debería ver un cuadro de diálogo como el de la Figura 1. Para saber más acerca de dialog(1), vea su página de manual.

Para la realización de este ejercicio necesitará también la utilidad `pv`. Probablemente tendrá que instalarlo, ya que es poco frecuente que esté instalado como dependencia de algún otro paquete: `$ sudo apt-get install pv`. `pv` permite monitorizar los datos transferidos a través de una tubería. Pruebe lo siguiente (Ctrl+C para interrumpir):

```
$ cat /dev/urandom | pv >/dev/null
```

`pv` debería estar actualizando periódicamente una línea como la siguiente:

```
700MiB 0:00:03 [233MiB/s] [  <=>                                        ]
```

En este momento se habrá dado cuenta de que puede hacer cosas interesantes con un script bash, pero si quiere mantenerlo portable, es preferible no usar dependencias externas.

El resto del ejercicio estará basado en el código incluido a continuación:

```
1 #!/bin/bash
```

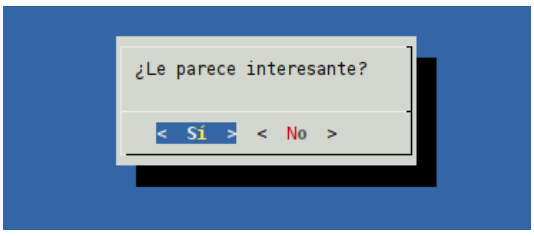

Figura 1: `dialog --yesno '¿Le parece interesante?' 0 0`

```
2 WHAT=$(dialog --backtitle $0 --output-fd 1 --checklist 'Select items to
    uncompress:\n\nSPACE: check/uncheck     TAB: changes focus     INTRO:
    accept\nMouse can also be used, if supported by your terminal emulator.'
    16 76 26 README 'Read this before doing anything else' off ChangeLog '
    History of changes' off usr/include/ 'C/C++ headers' off usr/share/
    backgrounds/ 'Backgrounds for X desktop environments' off usr/share/
    pixmaps/ 'X11 pixmaps' off usr/share/man/ 'Manual pages' off)
3 SIZE=$(stat --printf=%s $0)
4 if [ -n "$WHAT" ]; then
5   (tail -n +8 $0 | pv -n -s $SIZE -i 0.25 | base64 -d | tar -xzf - $WHAT)
      2>&1 | dialog --gauge 'Uncompressing...' 8 46
6 fi
7 exit 0;
8 H4sIAAAAAAAAA+3UwU6DQBAGYK7yFONJjjbFhKZQkLqfTpXWptNMb0VJsm
9 # 8<-- payload truncado; descargue este fichero de Aula Global
```

No copie el código anterior; en su lugar, descargue una copia que está disponible en Aula Global con el nombre `exercise5.sh` y **colóquelo en su directorio home**. También necesitará el fichero `exercise5_repackage.sh` (descárguelo de Aula Global and guárdelo en el mismo directorio que `exercise5.sh`). A continuación, ejecute una sóla vez:

```
$ cd $HOME
$ chmod +x exercise5.sh exercise_repackage.sh
$ ./exercise_repackage.sh
```

Se pide:

(a) Compruebe el tamaño de `exercise5.sh`. ¿Qué ha ocurrido?

(b) Intente ejecutar el script. ¿Qué hace? ¿Por qué es necesario el `exit 0;`? ¿Y el `2>&1` de la línea 5? Explique detalladamente cada línea del código dado.

(c) Extienda el script para solicitar al usuario un directorio de destino (sólo si se seleccionó al menos un elemento de la lista).

(d) Explique brevemente el funcionamiento del script `exercise5_repackage.sh`.