Programación y Administración de Sistemas
Práctica 1. Programación de *shell scripts*

Convocatoria de junio (curso 2021/2022)

Víctor Manuel Vargas

2 de marzo de 2022

Resumen

Esta serie de ejercicios se os entregan para que podáis practicar y profundicéis vuestros conocimientos de *bash* de cara al examen de prácticas. Estos ejercicios no se entregan, la evaluación de la práctica 1 se realizará mediante ejercicios similares a los expuestos en este guion. Para evitar problemas al ejecutar tus ejercicios de cara al examen, asegúrate de que todos los scripts que realices funcionen correctamente en los ordenadores de la UCO o conectándote por `ssh` al `ts.uco.es`. Para cualquier duda de los ejercicios, por favor, escribid en el foro del moodle o enviad un correo a la dirección vvargas@uco.es

1. `ejercicio1.sh`

Desarrollar un *script* que permita generar un directorio con ficheros y subdirectorios de ejemplo que podrás utilizar en futuros ejercicios de esta práctica. El *script* recibirá 3 argumentos:

1. Ruta del nuevo directorio que se va a crear.
2. Número de subdirectorios que se crearán dentro del directorio principal.
3. Longitud de los nombres de los ficheros (sin extensión) y subdirectorios.

Al ejecutarlo, deberá crear un directorio principal en la ruta que se haya especificado en el primer parámetro. Dentro de ese directorio, se deberán crear N subdirectorios (indicado por el segundo parámetro) con nombres aleatorios de la longitud especificada por el tercer parámetro. Por último, dentro de cada uno de estos subdirectorios, se crearán 4 ficheros (vacíos) con nombres aleatorios y las extensiones `.sh`, `.html`, `.key` y `.txt`. Por ejemplo, si el número de subdirectorios es 2 y la longitud del nombre es 5, se podrían generar de la siguiente forma:

- carpetadeejemplo
  - aXsh2
    - 3fdSS.sh
    - 38xzs.html
    - pLk32.key
    - hHg8M.txt
  - oLkJs
    - aZZ11.sh
    - uJgbM.html
    - 2wZxa.key
    - mMghk.txt

En el caso de que se indique el nombre de una carpeta que ya existe, se deberá pedir confirmación para eliminarla antes de crear la nueva carpeta.

Para generar nombres aleatorios, puedes hacerlo de la siguiente forma:

```
tr -dc A-Za-z0-9 < /dev/urandom | head -c LONGITUD
```

Esta línea accede al fichero `/dev/urandom`, que proporciona caracteres aleatorios de manera infinita, y elimina todos aquellos que no coincidan con A-Za-z0-9 (para obtener solo cadenas alfanuméricas). Por último, con `head`, indicamos que sólo se obtenga el número de caracteres que deseamos. Ya que es posible que debas utilizar esta línea varias veces en tu *script* , sería recomendable que hagas una función que te devuelva una cadena aleatoria de los caracteres que le indiques como argumento. Recuerda realizar los controles de errores oportunos (argumentos de entrada, ...).

Nota: en Moodle tendrás disponible la carpeta de ejemplo que se ha utilizado en los siguientes ejercicios por si quieres utilizar la misma para comparar salidas.

2. `ejercicio2.sh`

Desarrolla un *script* que permita configurar los permisos de los ficheros y subdirectorios de una determinada carpeta de la siguiente forma:

- El directorio y todos los subdirectorios deberán tener todos los permisos para el usuario, lectura y ejecución para el grupo y ninguno para otros.
- Los archivos cuya extensión sea `.sh` deberán recibir permisos de ejecución para el usuario.
- Los ficheros con extensión `.key` deberán asegurarse, restringiendo los permisos de manera que sólo el usuario propietario pueda acceder a ellos.

A continuación, se muestra un ejemplo de ejecución sobre una carpeta generada con el *script* del ejercicio 1 con los parámetros `example 5 5`:

```
i42vayuv@NEWTS:~/PAS$ ./ejercicio2.sh example/
Cambiando permisos de directorios...

example/
example/68YJr
example/eCgdi
example/GhMRl
example/cJzgY
example/CIdd8


Añadiendo permisos de ejecución a scripts...

example/68YJr/knUWV.sh
example/eCgdi/MaFGx.sh
example/GhMRl/cGm45.sh
example/cJzgY/YOMfz.sh
example/CIdd8/Us8gy.sh


Restringiendo permisos de ficheros de claves...

example/68YJr/NkMak.key
example/eCgdi/pxGov.key
example/GhMRl/KDm0B.key
example/cJzgY/iTKbd.key
example/CIdd8/RccQA.key
```

3. `ejercicio3.sh`

Desarrolla un *script* que permita realizar una copia de seguridad de un determinado directorio y almacenarla en un fichero comprimido. El programa deberá recibir dos argumentos:

1. Directorio que se va a copiar.

2. Directorio donde se almacenará la copia comprimida.

El nombre del fichero de copia resultante deberá seguir el formato: `nombredirectorioooriginal_AñoMesDia.tar.gz`.

Por ejemplo, si se hace una copia del directorio `ejemplo` el día 20 de marzo de 2022, el fichero resultante se llamará `ejemplo_20220320.tar.gz`. Para comprimir el fichero, deberás utilizar la herramienta `tar`. Consulta los argumentos necesarios para comprimir un directorio.

Si se intenta realizar una copia de un directorio que ya ha sido copiado en ese mismo día, se deberá mostrar un mensaje y no hacer nada. Si el directorio de destino de la copia no existe, deberás crearlo. Además, deberás realizar los controles de errores que estimes oportunos.

```
i42vayuv@NEWTS:~/PAS$ ./ejercicio3.sh example/
Argumentos incorrectos. Uso: ./ejercicio3.sh <directorio_origen> <directorio_destino>
i42vayuv@NEWTS:~/PAS$ ./ejercicio3.sh example/ backups
Copia realizada en backups/example_20220203.tar.gz.
i42vayuv@NEWTS:~/PAS$ ./ejercicio3.sh example/ backups
Ya se ha realizado esta copia hoy (backups/example_20220203.tar.gz).
i42vayuv@NEWTS:~/PAS$ ./ejercicio3.sh noexiste backups
noexiste no existe.
i42vayuv@NEWTS:~/PAS$ ls backups/
example_20220203.tar.gz
```

4. `ejercicio4.sh`

Desarrolla un *script* que permita listar todos los ficheros de un directorio sin mostrar los subdirectorios pero incluyendo los ficheros que estos puedan contener. El nombre del fichero deberá mostrar sin su ruta, solo incluyendo el nombre. Además, se deberá añadir un número que indicará el orden de cada fichero y también otro número que indicará el número de caracteres del mismo, tal y como se muestra en el siguiente ejemplo:

```
i42vayuv@NEWTS:~/PAS$ ./ejercicio4.sh example/
1                   07uFT.txt           10
2                   cGm45.sh            9
3                   Dl5Hg.html          11
4                   eEtJQ.txt           10
5                   gjOZF.html          11
6                   IjLpH.txt           10
7                   iTKbd.key           10
8                   j5W4s.html          11
9                   KDm0B.key           10
10                  knUWV.sh            9
11                  MaFGx.sh            9
12                  NkMak.key           10
13                  p9gwB.txt           10
14                  pxGov.key           10
15                  RccQA.key           10
16                  Us8gy.sh            9
17                  Wzs4s.html          11
18                  xsAf9.html          11
19                  yKQv0.txt           10
20                  YOMfz.sh            9
```

Nota: la herramienta `nl` puede servirte de utilidad para numerar las líneas.

5. `ejercicio5.sh`

Desarrollar un *script* que reciba como parámetros la ruta de un directorio y un número entero N de horas, y liste todos los ficheros que se encuentren dentro de dicho directorio que hayan sido modificados en las N horas anteriores. Recuerda realizar los controles de errores oportunos.

Una ejecución del *script* debe producir una salida similar a:

```
i42vayuv@NEWTS:~/PAS$ ./ejercicio5.sh
Argumentos incorrectos. Uso: ./ejercicio5.sh <ruta_directorio> <num_horas>
i42vayuv@NEWTS:~/PAS$ ./ejercicio5.sh example/ 1
example/68YJr/knUWV.sh
example/68YJr/gjOZF.html
example/68YJr/IjLpH.txt
example/68YJr/NkMak.key
example/eCgdi/pxGov.key
example/eCgdi/07uFT.txt
example/eCgdi/xsAf9.html
example/eCgdi/MaFGx.sh
example/GhMRl/cGm45.sh
example/GhMRl/j5W4s.html
example/GhMRl/p9gwB.txt
example/GhMRl/KDm0B.key
example/cJzgY/iTKbd.key
example/cJzgY/yKQv0.txt
example/cJzgY/Dl5Hg.html
example/cJzgY/YOMfz.sh
example/CIdd8/RccQA.key
example/CIdd8/eEtJQ.txt
example/CIdd8/Us8gy.sh
example/CIdd8/Wzs4s.html
```

6. `ejercicio6.sh`

Desarrolla un *script* que simule la creación de nuevos usuarios. La gestión de usuarios es una tarea muy común para un administrador de sistemas. Sin embargo, en los servidores de la universidad no tenemos la posibilidad de crear o eliminar usuarios. Por ello, en este ejercicio se pretende crear un sistema de usuarios, muy sencillo, que simule el sistema utilizado en Linux.

El sistema constará de un fichero (por ejemplo `users.txt`) que almacenará los usuarios existentes. Por otro lado, cada usuario tendrá su home dentro de un directorio (por ejemplo dentro de `./homes`). Además, habrá un directorio que contendrá los ficheros por defecto que se añaden al home de un usuario al crearlo (por ejemplo `./skel`).

Dentro de nuestro *script* deberemos implementar una función `crear_usuario` que se encargue de añadir un nuevo usuario al sistema con el nombre indicado en su primer argumento. Al crearlo, lo añadirá al fichero de texto, le creará su home y meterá los archivos por defecto que se encuentren en el directorio `skel`. Si se intenta crear un usuario que ya existe, no deberá volver a crearlo.

Una vez creada dicha función, el script deberá llamarla utilizando como nombre el primer argumento con el que se invoque el *script* . Recuerda realizar los controles de errores oportunos.

A continuación se muestra un ejemplo de ejecución del *script* :

```
i42vayuv@NEWTS:~/PAS$ mkdir skel
i42vayuv@NEWTS:~/PAS$ touch skel/file1
i42vayuv@NEWTS:~/PAS$ touch skel/file2
i42vayuv@NEWTS:~/PAS$ touch skel/file3
i42vayuv@NEWTS:~/PAS$ ./ejercicio6.sh prueba
Se ha creado el usuario prueba.
i42vayuv@NEWTS:~/PAS$ cat users.txt
prueba
i42vayuv@NEWTS:~/PAS$ ls homes/prueba/
file1  file2  file3
i42vayuv@NEWTS:~/PAS$ ./ejercicio6.sh nuevo
Se ha creado el usuario nuevo.
i42vayuv@NEWTS:~/PAS$ cat users.txt
prueba
nuevo
i42vayuv@NEWTS:~/PAS$ ls homes/nuevo/
file1  file2  file3
```

A. Conexión en remoto a la UCO

Para poder trabajar en remoto en el servidor de la UCO, necesitaremos, por un lado, conectarnos a una sesión de `ssh` para poder tener una terminal remota y, por otro lado, conectarnos por `sftp` para poder transferir archivos.

A.1. Conexión SSH

A.1.1. Linux

Para conectarnos por `ssh` desde Linux, basta con abrir una terminal y escribir:

```
ssh usuarioUCO@ts.uco.es
```

En caso de que no tengamos el cliente de `ssh` instalado, deberemos instalarlo con

```
apt install openssh-client
```

A.1.2. Windows 10+

En primer lugar es necesario habilitar el cliente `ssh`, que viene deshabilitado por defecto. Para ello, puedes seguir estas instrucciones [1]. Una vez habilitado, basta con abrir una ventana de `cmd` y escribir el comando:

```
ssh usuarioUCO@ts.uco.es
```

A.2. Conexión SFTP

A.2.1. Linux

Para conectarnos por `sftp` desde Linux, basta con abrir una ventana del explorador de archivos y en la barra de direcciones escribir:

```
sftp://usuarioUCO@ts.uco.es/home/usuarioUCO
```

A.2.2. Windows 10+

En el caso de Windows deberemos instalar algún cliente de `sftp` como por ejemplo WinSCP. El usuario de conexión será nuestro usuario de la UCO y la dirección del servidor `ts.uco.es`.

[1] `https://www.howtogeek.com/336775/how-to-enable-and-use-windows-10s-built-in-ssh-commands/`