Práctica 3.
Juan Cruz de Urquiza.
Introducción a los Sistemas Operativos.
2022.
Facultad de Informática, Universidad Nacional de La Plata.

1. Un Shell Script es un programa que está creado con instrucciones que son ejecutadas por un Shell de Unix o Linux. El código no es compilado ni precompilado, se va ejecutando línea por línea efectuando lo que cada instrucción le indica. Necesita un programa que entienda los comandos y estructuras que contiene y esto se suele poner en la primera línea del programa. Se suele usar la extensión .sh para identificar qué contiene el fichero, pero no es necesario.
   Sirve para automatizar tareas y para realizar procesos más complejos de los que un solo comando puede efectuar. El propio sistema de Linux tiene programadas multitud de tareas con sus Script del sistema, desde la rotación de logs, actualización del arranque, gestión de servidores, niveles de ejecución etc.
2. El comando echo de Linux repite lo que se le encarga que repita. La función es sencilla pero necesaria. Sin echo no se podrían mostrar visiblemente los scripts de Shell.
   Por ejemplo, si se quiere imprimir el String “Esto es un ejemplo”, se ingresa en la terminal: echo esto es un ejemplo
   La variable read permite leer una entrada y almacenarla en una variable. Por ejemplo, si se quiere guardar algo en la variable var1, se ejecuta
   read var1
   texto que se guardará en var1
   a. Los comentarios dentro de un script se indican con el signo #, seguido del comentario a introducir.
   b. Para crear una variable, se indica el nombre de la variable, seguido de = y sin espacios luego de este se indica el valor que tendrá la variable.
      Por ejemplo:
      edad=20
      Se creó una variable llamada edad que tiene el valor 20.
      Para acceder a una variable, se utiliza el comando echo seguido del nombre de la variable con un $ adelante de éste.
      echo $edad # imprimirá el valor de la variable edad(en este caso,20).
3.
   a. chmod 777
      /home/iso/Documentos/Facultad/practicashellscript/mostrar.sh
   b. *Resuelto en máquina.*
   c. Al ejecutar mostrar.sh, imprime “Introduzca su nombre y apellido:” y se puede leer un string donde se introduce el nombre y el apellido. Terminado esto, imprime la fecha y hora actual, el apellido y nombre

del usuario, el usuario que está usando y “Su directorio actual es:” seguido de nada.

d. Los backquotes `` dejan ejecutar comandos dentro de un archivo shell script.

e. #!/bin/bash
```
# Comentarios acerca de lo que hace el script
# Siempre comento mis scripts, si no hoy lo hago
# y mañana ya no me acuerdo de lo que quise hacer
echo "Introduzca su nombre y apellido:"
read nombre1 nombre2 apellido1 apellido2
echo "Fecha y hora actual:"
date
echo "Su apellido y nombre es:
echo "$apellido1 $apellido2 $nombre1 $nombre2"
echo "Su usuario es: `whoami`"
echo "Su directorio actual es: `pwd`"
echo “Su directorio personal es: $HOME”
echo “El contenido de `pwd` es: `ls`”
echo “El espacio libre en el disco es de: `df -h`”
```

4. Al momento de su invocación, los parámetros son accedidos mediante $1, $2,...,$n, n siendo el número del último parámetro recibido por el script.
$# contiene la cantidad de argumentos recibidos.
$* contiene la lista de todos los argumentos.
#? contiene en todo momento el valor de retorno del último comando ejecutado.
$HOME contiene la dirección del directorio personal del usuario ejecutando el script.

5. La función exit, causa la terminación de un script. Puede devolver cualquier valor entre 0 y 255: el valor 0 indica que el script se ejecutó de forma exitosa, y un valor distinto indica un código de error. Se puede consultar el estado de exit imprimiendo la variable $?.

6. Se pueden realizar las operaciones de suma(+), resta(-), multiplicación(*) y división(/). Es importante que haya espacio entre los operandos y la operación.

7. Al comando test se le pueden pasar diferentes parámetros, que devuelven true o false:

-b verdad si el fichero existe y es un fichero especial de bloques.

-c verdad si el fichero existe y es un fichero especial de caracteres.

-d verdad si el fichero existe y es un directorio.

-e verdad si el fichero existe.

-f verdad si el fichero existe y es un fichero regular.

-g verdad si el fichero existe y tiene el bit SGID.

-p verdad si el fichero existe y es una tubería con nombre (FIFO).

-r verdad si el fichero existe y se puede leer.

-s verdad si el fichero existe y tiene un tamaño mayor que cero.

-u verdad si el fichero existe y tiene el bit SUID.

-w verdad si el fichero existe y se puede modificar.

-x verdad si el fichero existe y es ejecutable.

-L verdad si el fichero existe y es un enlace simbólico o blando.

-z verdad si la longitud de una cadena es cero.

-n verdad si la longitud de una cadena no es cero.

8. La estructura del if se compone de:

```
if [condición]
then
        bloque
fi
```

La estructura del case se compone de:

```
case $variable in
        “valor 1”)
                bloque
        ;;
        “valor 2”)
                bloque
        ;;
        *)
                bloque
        ;;
esac
```

La estructura del while se compone de:

```
while [condición]
do
        bloque
done
```

La estructura del for tiene 2 estilos: para una lista de valores y el estilo de iteración (basado en C):

Estilo de iteración:

```
for ((i=0; i<10;i++))
do
        bloque
```

done
Estilo para lista de valores:
for i In value1 value2 value3 valueN;
do
	bloque
done
La estructura del select se compone de:
select variable in opcion1 opcion2 opcionN
do
	bloque
done

9. Dentro de un break, la sentencia break recibiendo el parámetro [n] corta la ejecución de n niveles del loop. La sentencia continue [n] salta a la siguiente iteración del loop número n que contiene esta instrucción.
10. Bash soporta variables de tipo string y arrays. Por defecto son globales, y son reemplazadas por el valor 0 o nulo dependiendo del contexto. Bash no es fuertemente tipado, ya que no hace falta declarar de qué tipo son las variables para darle un valor. Sí, se pueden definir arreglos en bash:
    Definición de un arreglo vacío: arreglo1=()
    Definición de un arreglo con valores dentro: arreglo2=(2 4 10 11 23 85)
11. Sí, se pueden definir funciones dentro de un script. Se pueden declarar de 2 formas: function nombre { código } ó nombre {código}.
    Las variables de entorno son heredadas por los procesos hijos. Para exponer una variable global a los procesos hijos se usa el comando export.
12. *Resuelto en máquina.*
13. *Resuelto en máquina.*
14. *Resuelto en máquina.*
15. El comando cut extrae las partes seleccionadas de cada fichero en la salida estándar. Es decir, de la salida de un script o función, con el comando cut se puede decidir cuál es la información que se quiere mostrar. Con el parámetro -b se pueden elegir los bytes a mostrar, con -c los caracteres a mostrar, -f para delimitar los campos, --complement para complementar la salida seleccionada, entre otros.
    Por ejemplo, si se tiene un archivo.txt que contiene:
    Prueba del comando cut
    Esta es la segunda línea
    Si se ejecuta cut -c 4 archivo.txt la salida de la terminal será:
    e
    a
    ya que ese es el cuarto carácter de ambas líneas.
    Si se ejecuta cut -c 1-3,5-8 archivo.txt la salida de la terminal será:
    Por ejemp
    Est es l

Ya que esos son los caracteres que van del 1 al 3 y del 5 al 8 en las 2 líneas del archivo.
Si se ejecuta cut -- complement -c 4 archivo.txt la salida de la terminal será:
Pruba del comando cut
ya que esos son todos los caracteres a excepción del 4.

16. *Resuelto en máquina.*
17. *Resuelto en máquina.*
18. *Resuelto en máquina.*
19. *Resuelto en máquina.*
20. *Resuelto en máquina.*
21. *Resuelto en máquina.*
22. *Resuelto en máquina.*
23. *Resuelto en máquina.*
24. *Resuelto en máquina.*
25. *Resuelto en máquina.*
26. *Resuelto en máquina.*
27. *Resuelto en máquina.*
28. *Resuelto en máquina.*
29. *Resuelto en máquina.*
30. *Resuelto en máquina.*