Sistemas Operativos

Curso 2018

TRABAJO PRÁCTICO LABORATORIO Nº 1: Shell scripts

Comandos Básicos en Linux

1) Explique en no más de un renglón cada uno de los siguientes comandos:
    a. Relacionados con archivos: ls, cat, more, less, cp, rm, mkdir, cd, pwd, file, df, grep.
    b. Relacionados con la documentación del sistema Linux: whatis, apropos, info, man.
    c. passwd, mount, time, date, vi, gcc, exit, reboot, halt, poweroff
    d. Relacionados con procesos y memoria: ps, kill, top
    e. Relacionados con redes: ifconfig, ping, ip, traceroute, ssh, telnet, ftp

2) Mencione al menos cinco comandos equivalentes en Windows, por ejemplo: ls ≈ dir. Intente preferentemente con los relacionados a redes.

3) Explique (con la ayuda del manual en línea) y pruebe el comando chmod.
    a. ¿Cómo invocaría chmod para asignar permisos de lectura, escritura y ejecución al propietario, lectura y ejecución para el resto de los usuarios del sistema? Suponga que estos permisos deben asignarse a todos los archivos contenidos en una carpeta de nombre *scripts* (sin entrar a la carpeta).
    b. ¿Cómo invocaría chmod para asignar permisos de lectura y escritura (sin ejecución) al propietario y solo lectura para el resto de los usuarios del sistema? Suponga que estos permisos deben asignarse a todos los archivos con extensión *.txt* en el directorio actual.

Shell Scripts

Resolver los siguientes ejercicios en Linux

1. Dado el siguiente script:

```
clear
while :
do
        echo -e "\033[H"
        ps
done
```

   a. Describir qué hace.
   b. Modificar el script de tal forma que ejecute cualquier comando en forma reiterada, en vez de hacerlo solo con ps.

2. Construir un shellscript que reciba argumentos numéricos e imprima cada uno en líneas separadas utilizando la estructura de repetición while. Mostrar al final, la suma de todos los números pasados como parámetros.

3. Agregar al script anterior la condición de que solo se ejecute si la cantidad de argumentos está entre 5 y 7 inclusive y todos los argumentos ingresados son números. En caso que la cantidad de argumentos sea diferente y algún argumento no sea un número, se debe mostrar un mensaje de uso y de error respectivamente.

4. Rehacer el script del punto 1, utilizando un for en lugar de while.

5. Escribir un shellscript que solicite al usuario presionar una tecla y luego imprima un mensaje indicando si se presionó una letra, un dígito u otro símbolo. La instrucción para tomar información del teclado dentro de un shellscript es "*read <nombre_variable>*". Utilizar un case para resolver el ejercicio

6. Crear un shellscript que permita listar los tipos de archivos que contiene un directorio pasado como parámetro. Efectuar las correspondientes validaciones.

7. Crear un script que al pasarle un nombre de archivo como primer argumento, ordene el mismo de forma ascendente o descendente, de acuerdo a si el segundo parámetro es ‘-A’ o ‘-Z’. Verificar que el script reciba los 2 argumentos, sino mostrar la forma de uso.

$> ./nombre_script.sh [archivo] [-A|-Z]

8. Dado el siguiente shellscript:

```
01    #!/bin/bash
02    #
03    # prog5
04
05    if [ $# -gt 0 ]; then
06          if [ $# -ne 1 -o $1 != "-h" ]; then
07                echo "uso: prog5 [-h]"
08                exit 1
09          fi
10          echo "Texto explicativo."
11          exit 0
12    fi
13
14    for i in *
15    do
16          lfn=`echo $i | gawk ' {str = tolower($0);print str }'`
17          if [ $lfn != $i ]; then
18                mv $i $lfn
19          fi
20    done
```

Mediante inspección del código impreso, describir que es lo que hace. Luego transcribir en Linux y probar.
NOTA: En la línea 16 se asigna a la variable lfn el contenido de la variable i transformado a minúsculas.

9. Dado el siguiente ShellScript:

```
01    #!/bin/bash
02    # shscript
03    toggle="A"
04    if [ $1 == "-p" ]; then
05          toggle="B"
06    fi
07    while read line
08    do
09          if [ $toggle == "A" ]; then
10                echo $line
11                toggle="B"
12          else
13                toggle="A"
14          fi
15    done
```

a. Describa qué hace el shellscript.
b. Probar redireccionando en la entrada un archivo con texto.

10. El siguiente shellscript está pensado para recibir uno o dos parámetros en línea de comandos, ni más ni menos:

```
01    #! /bin/bash
02    # shscript
03    if [ "$1" = "-" ]
04    then
05          for i in *
06          do
07                grep "$2" $i > /dev/null || echo $i
08          done
09    else
10          for i in *
11          do
12                grep "$1" $i > /dev/null && echo $i
13          done
14    fi
```

a. Describa qué hace este shellscript.
b. Transcriba y pruebe en Linux.

11. Dado el siguiente Sellscript:

```
01    #!/bin/bash
02    #
03    # shellscript1
04
05    if [ $# -ne 1 ]; then
06          echo "uso: shellscript1 <patron> | -h"
07          exit 1
08    fi
09
10    if [ $1 = "-h" ]; then
11          echo "Texto explicativo."
12          exit 0
13    fi
14
15    PATRON=$1
16
17    for i in `find ./ -name “*.[hc]”`
18    do
19          echo "----------"
20          echo $i
21          grep $PATRON $i
22    done
```

a. Mediante inspección del código impreso, comente qué hace.

b. Cuál sería el contenido del archivo "salida.txt" si se ejecutara el siguiente comando:

```
./shellscript1 shmget > salida.txt
```

Suponiendo que el directorio actual es /usr/src. (Directorio donde normalmente está el código fuente del Sistema Operativo Linux, distribuido a su vez en varios subdirectorios.)

c. Modificar el shellscript de tal forma que no se impriman los nombres de los archivos que no contienen el patrón. Puede ser útil el uso del operador &&.

12. Buscar un servicio JSON en Internet que entregue la latitud y la longitud de una ciudad. En base a esto construir un shell script que reciba el nombre de una ciudad y muestre la latitud y longitud de la misma. Seguramente el servicio entregará más datos que tendrán que ser desestimados. Para este script es posible que necesite usar: wget, curl, sed, awk, etc.