12-5-2020

david ros y álvaro fraidias

PRÁCTICA 2

Programación en Shell Script

**Ejercicio 1. Analiza las siguientes órdenes y describe su uso, al igual que todos aquellos argumentos que consideres interesantes.**

* **sort 🡪** ordena el contenido de un archivo de texto.

sort [OPTION] file

Las principales opciones son:

* -b 🡪 ignora los espacios en blanco principales.
* -d 🡪 ordena ignorando todos los caracteres salvo letras, números y espacios.
* -n 🡪 ordena según el valor numérico de la cadena
* -k n1,[n2] 🡪 especifica un campo como clave de ordenación, comienza en n1 y acaba en n2.
* -r 🡪 invertirá el orden.
* **Split 🡪** divide un archivo en partes.

Split [OPTION] [INPUT [PREFIX]]

Las principales opciones son:

* -a N🡪 usa sufijos de longitud N.
* -b TAMAÑO 🡪 escribe TAMAÑO bytes por archivo de salida.
* -c TAMAÑO 🡪 escribe como máximo TAMAÑO bytes de líneas por archivo de salida.
* -d 🡪 usar sufijos numéricos en vez de alfabéticos.
* **kill 🡪** finalizar procesos manualmente
* **ps 🡪** visualizar los procesos de nuestro sistema y obtener información sobre ellos.

ps [OPTION]

Las principales opciones son:

* -l 🡪 formato largo.
* -u 🡪 formato usuario: muestra el usuario y l ahora de inicio.
* -a 🡪 muestra también los procesos de otros usuarios.
* -S 🡪 añade tiempo de CPU y fallos de página de los hijos.
* -r 🡪 muestra solo procesos que se están ejecutando.
* **pstree 🡪** visualizar el árbol de procesos, mostrando la relación padre-hijo

pstree [OPTION]

Las principales opciones son:

* -p 🡪 muestra el PID de cada proceso.
* -u 🡪 muestra las transiciones UID. Cada vez que el UID de un proceso difiere desde el UID de su padre, el nuevo UID se muestra en paréntesis después del nombre del proceso.
* -a 🡪 muestra argumentos de la línea de comandos.
* -s 🡪 muestra el ID de seguridad para cada proceso.
* -n 🡪 ordenar procesos con el mismo ancestro por PID en lugar de por nombre.
* **renice 🡪** altera la prioridad de programación de una o más ejecuciones de procesos.

renice [-n] priority [-g | -p | -u ] identifier

Sus principales opciones son:

* -n 🡪 especificar la prioridad de programación que se utilizará para el proceso, el grupo de procesos o el usuario.
* -g 🡪 interpretar los argumentos siguientes como ID de grupo de procesos.
* -p 🡪 interpretar los argumentos siguientes como ID de proceso (el valor prdeterminado).
* -u 🡪 interpretar los argumentos siguientes como nombres de usuario o UID.
* **uptime 🡪** ofrece una visualización de una línea con la siguiente información: la hora actual, cuánto tiempo ha estado funcionando el sistema, cuántos usuarios están actualmente conectados y los promedios de carga del sistema durante los últimos 1, 5 y 15 minutos.

uptime [options]

Las principales opciones son:

* -p 🡪 mostrar el tiempo de funcionamiento en “pretty format”.
* -s 🡪 mostrar la fecha y la hora desde que el sistema está funcionando.
* -h 🡪 muestra un mensaje de ayuda y sale.
* -V 🡪 muestra información de la versión y la salida.
* **hostname 🡪** mostrar el host actual, el dominio o el nombre de nodo del sistema.

Sus principales opciones son:

* -a 🡪 muestra el nombre de alias del host.
* -d 🡪 muestra el nombre del dominio DNS.
* -F 🡪 lee el nuevo nombre de host del archivo especificado.
* -f 🡪 muestra el FQDN (Nombre de dominio completo). Consta de un nombre que incluye el dominio DNS.
* -i 🡪 muestra las direcciones IP del host.
* **shutdown 🡪** puede usarse para detener, apagar o reiniciar la máquina.

shutdown [OPTIONS] [TIME] [WALL]

La cadena de tiempo especificará la hora a la que se ejecutará el apagado, especificado en formato de reloj de 24 h.

Después del tiempo le podrá seguir un mensaje de muro que se enviará a todos los usuarios registrados antes de dejar de funcionar.

Sus principales opciones son:

* -H 🡪 detendrá la máquina.
* -P 🡪 apaga la máquina.
* -k 🡪 no se detenga, apague, reinicie, solo escriba un mensaje en el muro.
* **uname 🡪** imprime cierta información del sistema.

uname [OPTION]

Sus principales opciones son:

* -a 🡪 muestra toda la información.
* -s 🡪 muestra el nombre del kernel.
* -n 🡪 muestra el nombre de host del nodo de red.
* -i 🡪 muestra la plataforma del hardware.
* -o 🡪 muestra el sistema operativo.

**Ejercicio 2. Escribir un script donde el usuario pasará un conjunto de valores por consola y automáticamente devolverá todos los valores sin ordenar por consola y posteriormente ordenados.**

#!/bin/bash

function burbuja {

lista=$1

tam=${#lista[@]}

for i in $(seq 1 $[$tam-1]); do

for j in $(seq 0 $[$tam - $i - 1]); do

if [ ${lista[$j]} -gt ${lista[$j+1]} ]; then

k=${lista[$[$j+1]]}

lista[$j+1]=${lista[$j]}

lista[$j]=$k

fi

done

done

}

lista=($\*)

echo "La secuencia introducida es: $\*"

burbuja $lista

echo "La secuencia ordenada es:"

for i in ${lista[@]}; do

echo $i;

done

**Ejercicio 3. Desarrollar un script llamado “configuración.sh” que nos muestre diferentes configuraciones del sistema:**

#!/bin/bash

mostrarUsuarios()

{

echo EJER3

echo 1. Usuarios conectados y su logname

echo $(who | cut -d' ' -f1 | sort | uniq)

}

mostrarShell()

{

echo 2. El shell actual y los shells disponibles

echo Shell actual:

echo $SHELL

echo Shells disponibles:

more /etc/shells

}

mostrarDirectorio()

{

echo 3. El directorio home y el PATH actual

echo Directorio home:

more $HOME

echo PATH actual:

echo $PATH

}

mostrarUsuariosNoConectados()

{

echo 4. Numero y nombre de usuarios que no estan conectados, pero pertenecen al sistema

sort /etc/passwd | cut -d: -f1

}

versionSO()

{

echo 5. Version del sistema operativo

lsb\_release -a

}

configuracionRaton()

{

echo 6. Configuracion del raton instalado

cat /proc/devices

}

informacionCPU()

{

echo 7. Informacion de la CPU

cat /proc/cpuinfo

}

informacionMemoria()

{

echo 8. Informacion de la memoria

cat /proc/meminfo

}

informacionDiscosDuros()

{

echo 9. Informacion de los diferentes discos duros del sitema

sudo sfdisk -l

}

mostrarFicherosMontados()

{

echo 10. Sistema de ficheros montados

cat /proc/mounts

}

mostrarUsuarios

mostrarShell

mostrarDirectorio

mostrarUsuariosNoConectados

versionSO

configuracionRaton

informacionCPU

informacionMemoria

informacionDiscosDuros

mostrarFicherosMontados

**Ejercicio 4. Desarrollar un script que nos muestre por consola información relevante del identificador de proceso que el usuario inserte por consola.**

#!/bin/bash

#Muestra la información del proceso pasado como parámetro

mostrarInfoProceso()

{

proceso=$1

ps -l|tr -s ' ' |grep ^$proceso|cut -d " " -f 3,4,5,6,7,9

}

mostrarInfoProceso

**Ejercicio 5. Desarrollar un script para que busque a aquellos procesos con una prioridad relativa superior a 10 y les incremente su prioridad en 5 unidades; posteriormente deberá mostrar que procesos han sido susceptibles de modificación junto con la nueva prioridad asignada.**

#!/bin/bash

#Función que aumenta la prioridad relativa del proceso pasado si su prioridad es mayor de 10

prioridad()

{

PID=($(ps -l|awk '$8 > 10 && $4 {print $4}'))

NI=($(ps -l|awk '$8 > 10 && $4 {print $8}'))

longitud=${#PID[@]}

if [[ $longitud = 1 ]]

then

echo "No se han encontrado procesos con una prioridad relativa superior a 10"

else

for ((i = 1;i <$tam;i++)) do

NI[$i]=$((${lista[$i]} + 5))

renice -n ${NI[$i]} -p ${PID[$i]}

done

fi

}

prioridad

**Ejercicio 6. Realizar un shell script que genere cuentas de usuario.**

Para comprobar que el usuario ha introducido 3 parámetros hacemos una función, pero en los demás casos no porque la ejecución del script era diferente y el resultado era erróneo en comparación con la ejecución sin funciones.

#!/bin/bash

#Comprobamos que el usuario haya introducido 3 parámetros

comprobarTresParametros()

{

if ! [ "$#"=3 ]

then

echo "Numero de parametros incorrecto"

exit 0

fi

}

comprobarTresParametros

#Comprobamos que el login del usuario está comprendido entre 3 y 8 caracteres y comprobamos que sólo contenga letras minúsculas

login=$1

longitud=${#login}

if [ $longitud -lt 3 ] || [ $longitud -gt 8 ]

then

echo "Login incorrecto: el login deberá tener entre 3 y 8 caracteres"

exit 1

else

sudo useradd -d /HOME -s /bin/bash $login

fi

#Comprobamos si el tercer parámetro es contabilidad,finanzas o estadísitca

grupo=$3

if [ "$grupo" = contabilidad ] || [ "$grupo" = finanzas ] || [ "$grupo" = estadistica ]

then

echo "Grupo correcto"

else

echo "Grupo incorrecto: el grupo tiene que ser contabilidad, finanzas, estadística"

exit 2

fi

#Comprobamos si el grupo seleccionado ya existe

columnaGrupo=$(cut -d ":" -f1 /etc/group|grep "$grupo")

if [ "$columnaGrupo" ]

then

usermod -g $grupo $login

else

#Creamos el grupo y se lo asignamos al usuario creado anteriormente

groupadd -r $grupo

usermod -g $grupo $login

fi

**Ejercicio 7. Generar un script que nos permita generar ficheros tar, inicialmente el script nos mostrará un menú con las siguientes opciones:**

#!/bin/bash

echo "a. Generación fichero tar"

echo "b. Extracción fichero tar"

echo "c. Visualización de la información del fichero tar"

echo "d. Listado de todos los archivos incluidos en el fichero tar"

configurarFicheroTar()

{

read -p "Seleccione una opción: " opcion

case $opcion in

a) read -p "Introduzca el nombre del archivo a generar: " archivo

tar -cvf $archivo.tar ./$archivo;;

b) read -p "Introduzca el nombre del archivo a extraer: " archivo

tar -xvf $archivo.tar;;

c) read -p "Introduzca el nombre del archivo a visualizar: " archivo

stat $archivo;;

d) read -p "Introduzca el nombre del archivo: " archivo

tar -tvf $archivo.tar;;

\*) echo "Opción incorrecta";;

esac

}

configurarFicheroTar

**Ejercicio 8. No hemos sabido resolverlo.**

**Ejercicio 9. Realizar un script que nos genere password cifrados de 10 caracteres aleatorios, siendo dichos caracteres los que a continuación se describen dentro de un vector:**

#!/bin/bash

VECTOR="0123456789ABCDEFGHIJKLMNÑOPQRSTUVWXYZabcdefghijklmnñopqrstuvwxyz"

crearContrasenia()

{

#Creamos una contraseña aleatoria de 10 carácteres

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

aleatorio=$(echo "$RANDOM")

#Módulo 63 porque hay 64 psibilidades en el vector

num=$((aleatorio%63))

contra+=$(echo ${VECTOR:$num:1})

done

cifrada=$(echo "$contraseña"|md5sum)

aleatorio=$(echo "$RANDOM")

#caracter por caracter se le aplica la formula caracter = (ASCII+RANDOM)&256

for((i=0;i<10;i+=1))

do

car=$(echo ${contra:$i:1})

car=$(printf "%d" \'"$car")

car=$((car+aleatorio))

car=$((car%256))

car=$(echo "$car"|xxd -p -r)

cifradocar+=$(echo "$car")

done

echo "Contraseña: $contra"

echo "Contraseña cifrada con md5: $cifrada"

echo "Contraseña cifrada caracter a caracter: $cifradocar"

}

crearContrasenia