Instantly share code, notes, and snippets.

Create a gist now

ivanzugnoni / **sistemas-operativos-ejercicios-bash.md**

Last active a year ago

sistemas-operativos-ejercicios-bash.md

Ejercicio 1

```
# Realizar un script utilizando una estructura repetitiva, que recorra un directorio
# cualquiera pasado por parametro y devuelva el nombre del archivo comun (no directorio)
# que tiene mayor tamaño (devolver nombre y tamaño). A su vez, validar que el parametro
# no falta y sea realmente un directorio.

#!/bin/bash

if [ ! $1 ]; then
        echo "El script solicita 1 parametro"
        exit
fi

if [ ! -d $1 ]; then
        echo "El parametro debe ser un directorio"
        exit
fi

for i in $1/*; do
        if [ -f $i ]; then
                TAM=$(du -s $i | cut -d "/" -f1)
                echo $TAM $i >> archivos
        fi
done

sort -nr archivos | head -n1
```

Ejercicio 2

```
# Dado un directorio pasado como parametro:
# 1) Verificar que no falte el parametro y que realmente sea un directorio
# 2) Buscar todos los archivos .JPG y .PNG y copiarlos a una carmeta MIS_IMAGENES
# 3) Empaquetar y comprimir la carpeta MIS_IMAGENES

#!/bin/bash

if [ ! $1 ]; then
        echo "El script solicita 1 parametro"
        exit
fi

if [ ! -d $1 ]; then
        echo "El parametro debe ser un directorio"
        exit
fi

mkdir $HOME/MIS_IMAGENES

find $1 -type f -name "*.jpg" -o -name "*.png" -exec cp {} $HOME/MIS_IMAGENES \;

tar -czf $HOME/MIS_IMAGENES.tar MIS_IMAGENES
```

Ejercicio 3

```
# Dado un archivo con 10 palabras desordenadas (una palabra por linea) enviado como parametro,
# imprimir por pantalla el top 3 de las palabras con mas apariciones, ordenado de forma descendiente.
# Validar que no falte el parametro.


#!/bin/bash

if [ ! $1 ]; then
        echo "El script solicita 1 parametro"
        exit
fi

if [ ! -f $1 ]; then
        echo "El parametro debe ser un archivo"
        exit
fi


sort $1 | uniq -c | sort -nr | head -n3
```

Ejercicio 4

```
# Dado un nombre de usuario recibido por parametro:
# 1) Validar que no falte enviar el parametro
# 2) Decir si ese usuario se encuentra actualmente conectado al sistema o no
# 3) Indicar la cantidad total de usuarios de la com53 conectados actualmente al sistema
# 4) Guardar la salida del script en un archivo llamado usuarios.txt


#!/bin/bash

if [ ! $1 ]; then
        echo "El script solicita 1 parametro"
        exit
fi

ESTA_CONECTADO=$(who | grep $1)

if [ "$ESTA_CONECTADO" ];then
        echo "El usuario $1 esta conectado al sistema" >> usuarios.txt
else
        echo "El usuario $1 NO esta conectado al sistema" >> usuarios.txt
fi

CANTIDAD_CONECTADOS=$(who | cut -d " " -f1 | sort -u | wc -l)

echo "La cantidad de usuarios conectados es $CANTIDAD_CONECTADOS" >> usuarios.txt

cat usuarios.txt
```

Ejercicio 5

```
# Guardar en un fichero un listado de usuarios y la cantidad de procesos que esta
# ejecutando cada uno. Ordenar de mayor a menor por cantidad de procesos.

#!/bin/bash

for i in $(cat /etc/passwd | cut -d ":" -f1); do
        NOM=$i
        CANT=$(ps -u $i | wc -l)
        let CANT=$CANT-1
        echo $CANT $NOM >> usuarios
done

sort -nr usuarios
```

Ejercicio 6

```
# Hacer un script que cree un directorio usuarios y dentro del mismo
# cree un fichero por cada usuario que hay en el sistema (el nombre de cada
# fichero debe ser el nombre de usuario) que contenga la frase: Hola <nombreusuario>
# y cuyo dueño sea el mismo usuario.


#!/bin/bash

# NOTA: Para cambiar dueño con chown se necesitan permisos especiales.

mkdir usuarios

for i in $(cat /etc/passwd | cut -d ":" -f1); do
        echo "Hola $i" > usuarios/$i
        chown $i usuarios/$i
done
```

Ejercicio 7

```
# Guardar en un fichero un listado de usuarios y espacio en disco ocupado por
# los ficheros que tenga cada uno en su carpeta personal.
# Ordenar de mayor a menor por espacio ocupado

#!/bin/bash

# NOTA: Para acceder a los directorios de algunos usuarios se
# necesitan permisos especiales.

for i in $(cat /etc/passwd | cut -d ":" -f1); do
        NOM=$i
        TAM=$(du -s /home/$i | cut -d "t" -f1)
        echo $TAM $NOM > usuarios
done

sort -nr usuarios
```

Ejercicio 8

```
# Hacer un script al cual se le pase como parametro un directorio y una
# palabra. El script debe enviar a un archivo llamado matched el nombre
# de cada archivo que se encuentre dentro del directorio y que contenga
# la palabra indicada
# Comprimir el archivo matched en matched.gz

#!/bin/bash

DIRECTORIO=$1
PALABRA=$2

for i in $DIRECTORIO/*; do
        if [ -f $i ]; then
                CONTIENE=$(cat $i | grep $PALABRA)
                if [ $CONTIENE ]; then
                        echo $i >> matched
                fi
        fi
done

gzip matched
```

Ejercicio 9

```
#!/bin/bash

# Hacer un script que realice los siquientes pasos:
```

```
# Copiar el archivo passwd a su home de trabajo con nombre user
cp /etc/passwd $HOME/user
# Limpiar la pantalla
clear
# Del archivo user sacar todos los usuarios de la com54 y mandarlos
# al archivo solo54
grep "com54" /etc/passwd > solo54
# Del archivo solo54 cortar nombre de usuario, UID, GID y Shell y guardarlo
# en el archivo sindescrip
cat solo54 | cut -d ":" -f1,3,4,7 > sindescrip
# Cambiar los permisos de los tres archivos para que pueda ser MODIFICADO por
# OTROS (el resto dejarlos igual)
chmod o+w $HOME/user solo54 sindescrip
# Comprimir y empaquetar el archivo solo54, sindescrip y user como usu54.tar.gz
tar -czf usu54.tar.gz $HOME/user solo54 sindescrip
```

Ejercicio 10

```
#!/bin/bash

# Hacer un script que realice los siguientes pasos:

# Crear un archivo con nombre process (usando comando de creacion de archivos)
touch process
# En el archivo process agregar todos los procesos que se estan ejecutando en
# el sistema
ps aux > process
# Del archivo process cortar el nombre de usuario y mandarlo al archivo solousu
cat process | cut -d " " -f1 > solousu
# Cambiar los permisos de process y solousu para que OTROS no lo pueda VER
# (el resto de los permisos dejarlos igual)
chmod o-r process solousu
# Comprimir y empaquetar el archivo process y solousu como proce2014.tar.gz
tar -czf proce2014.tar.gz process solousu
```

Ejercicio 11

```
#!/bin/bash

# Hacer un script que reciba como parametro un archivo y un string y me informe si dicho string
# esta dentro del archivo. A su vez chequear la existencia del archivo y los parametros.

if [ ! $# -eq 2 ];then
    echo "Debe pasar dos parametros"
    exit
fi

if [ ! -f $1 ];then
    echo "El primer parametro debe ser un archivo"
    exit
fi

ARCHIVO=$1
STRING=$2

ESTA=$(grep "$STRING" $ARCHIVO)

if [ "$ESTA" = "" ];then
    echo "El string NO se encuentra dentro del archivo"
    else
    echo "El string se encuentra dentro del archivo"
fi
```

Ejercicio 12

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

```
# Realizar un script que evalue 2 archivos pasados como argumentos indicando adecuadamante entre
# ambos cual posee mas cantidad de caracteres. Dejar la informacion especificando el nombre del
# archivo y cantidad, en el archivo cant-caracteres. Comprobar que los parametros no sean pasados
# en blanco

if [ ! $# -eq 2 ];then
        echo "Debe pasar dos parametros"
        exit
fi

if [ ! -f $1 ] || [ ! -f $2 ];then
        echo "Ambos parametros deben ser archivos"
        exit
fi

CANT1=$(wc -m $1 | cut -d " " -f1)
CANT2=$(wc -m $2 | cut -d " " -f1)

if [ $CANT1 -gt $CANT2 ];then
        echo "El archivo $1 posee mas cantidad de caracteres"
        echo "Nombre: $1 - Cantidad de caracteres: $CANT1" > cant-caracteres
        else
        echo "El archivo $2 posee mas cantidad de caracteres"
    echo "Nombre: $2 - Cantidad de caracteres: $CANT2" > cant-caracteres
fi
```

Ejercicio 13

```
#!/bin/bash

# Hacer un script que realice los siguientes pasos:

# Generar un archivo llamado listagrupo en el que conste solamente nombre de grupo y GID de todos
# los usuarios de la com64 definidos en el sistema. Usar el archivo de grupos para resolverlo
grep "com64" /etc/group | cut -d":" -f1,3 > listagrupo
# Cambiar los permisos a listagrupo para que puedan ser MODIFICADOS por el GRUPO y OTROS
# (No dar permisos de mas, solamente los que se solicitan)
chmod go+w listagrupo
# Comprimir el archivo listagrupo (dice comprimir, no empaquetar)
gzip listagrupo
```

Ejercicio 14

```
#!/bin/bash

# Realizar un script el cual solicite un numero y responda mostrando los 6 siguientes.
# Los 6 numeros deben quedar guardados en orden inverso en el archivo numeros

NUM=$1
rm numeros

for i in $(seq 6); do
    let NUM=$NUM+1
    echo $NUM >> numeros
done

sort -nr numeros
```

Ejercicio 15

```
#!/bin/bash

# Realizar un script donde por parametro se ingresa una palabra y esta se imprima 10 veces por pantalla
# y a su vez se guarde en un archivo llamado word. Usar una sentencia iterativa para resolverlo, a su vez
# chequear que el parametro no se pase en blanco cuando se ejecuta el script. Comprimir el archivo word.

if [ ! $# -eq 1 ];then
```

```
        echo "Debe pasar un parametro"
        exit
fi

if [ "$1" = " " ];then
        echo "La palabra no puede estar en blanco"
        exit
fi

PALABRA=$1

for i in $(seq 10); do
        echo $PALABRA >> word
done

cat word
gzip word
```

Ejercicio 16

```
#!/bin/bash

# Realizar un script que indique si un usuario pasado como argumento tiene como shell el bash,
# de lo contrario decir que tipo de shell tiene.

if [ ! $# -eq 1 ];then
        echo "Debe pasar un usuario como parametro"
        exit
fi

USUARIO=$1
SHELL=$(grep $USUARIO /etc/passwd | cut -d":" -f7)

if [ "$SHELL" = "/bin/bash" ];then
        echo "El usuario $USUARIO tiene como shell BASH"
        else
        echo "El usuario $USUARIO tiene como shell $SHELL"
fi
```

Ejercicio 17

```
#!/bin/bash

# Crear un script al cual se le pase como argumento un nombre de usuario y muestre los procesos
# que esta ejecutando ese usuario. En caso de que no se pase ningun argumento, debe mostrar
# todos los procesos en ejecucion y en caso de que el usuario pasado como argumento sea root,
# mostrar un mensaje de error

USUARIO=$1

if [ $# -eq 0 ];then
        # No pasaron argumento, entonces muestro todos los procesos
        ps aux
        else
        if [ "$USUARIO" = "root" ];then
                # Usuario es root, entonces muestro mensaje de error
                echo "ERROR!!!"
                else
                # Muestro los procesos de ese usuario
                ps aux | grep "$USUARIO"
        fi
fi
```

Ejercicio 18

```
#!/bin/bash

# Realizar un script que al pasarle un archivo como parametro nos devuelva el tamaño del mismo.
# A su vez, chequear que el parametro no se pase en blanco y que el archivo exista o sea
# del tipo ordinario.

if [ ! $# -eq 1 ];then
        echo "Debe pasar un parametro"
        exit
fi

if [ ! -f $1 ];then
        echo "El parametro debe ser un archivo"
        exit
fi

TAM=$(du $1 | cut -d " " -f1)

echo "Tamaño de $1: $TAM"
```

Ejercicio 19

```
#!/bin/bash

# Realizar un script en bash que busque todos los archivos mp3 de todo el directorio home (todos los usua
# y los mueva a un directorio cualquiera llamado basura. Ademas informar sobre el tamaño total en Mbyte d
# todos los archivos encontrados

mkdir basura

find / -type f -name "*.mp3" -exec mv {} ./basura \;
du -m basura
```

Ejercicio 20

```
#!/bin/bash

# Realizar un script utilizando la sentencia while, que lea un archivo cualquiera linea por linea
# y guarde cada una de ellas en un archivo llamado copia, a su vez vaya mostrando cada linea
# por terminal con un retardo de 3 segundos.

ARCHIVO=$1

cat $ARCHIVO | while read linea; do
        echo $linea >> copia
        echo $linea
        sleep 3
done
```

Ejercicio 21

```
#!/bin/bash

# Hacer un script que lea un directorio cualquiera por parametro y guarde solamente el nombre de los arch
# en solo-archi, y contabilice los mismos. Dicho contador guardarlo al final del archivo solo-archi. Usar
# iterativa para resolverlo y chequear que el parametro no se pase en blanco

if [ ! $# -eq 1 ];then
        echo "Debe pasar un parametro"
        exit
fi

if [ ! -d $1 ];then
        echo "El parametro debe ser un directorio"
        exit
```

```
fi

CONT=0

for i in $1/*; do
        if [ -f $i ];then
                echo $i >> solo-archi
                let CONT=$CONT+1
        fi
done

echo "Cantidad de archivos: $CONT" >> solo-archi
```

Ejercicio 22

```
#!/bin/bash

# Hacer un script que arme un menu de 3 opciones:
# 1) Guarde una frase ingresada por teclado en un archivo
# 2) Muestre la frase ingresada en la opcion 1 si es que existe
# 3) Salir del menu

clear
RESP=0
rm frase.txt

function mostrar_menu {
    clear
    echo "MENU"
    echo "1) Ingresar frase"
    echo "2) Mostrar frase"
    echo "3) Salir"
}

while [ $RESP != 3 ]; do
        mostrar_menu
        read RESP
        if [ $RESP = 1 ]; then
                clear
                echo "Ingrese frase: "
                read FRASE
                echo $FRASE > frase.txt
                clear
                echo "La frase fue ingresada con exito!"
                echo "Aguarde un momento"
                sleep 5
                mostrar_menu
        fi

        if [ $RESP = 2 ]; then
                if [ -e frase.txt ]; then
                        clear
                        echo "Frase: $(cat frase.txt)"
                        sleep 5
                        mostrar_menu
                        else
                        clear
                        echo "Primero debe cargar una frase."
                        echo "Aguarde un momento"
                        sleep 5
                        mostrar_menu
                fi
        fi

        if [ $RESP = 3 ]; then
            echo "Chau!"
            exit
        fi
done
```

Ejercicio 23

```
#!/bin/bash

# Hacer un script que guarde en el archivo perm todos los archivos que tienen permisos igual a 755
# y en el archivo exten todos los que terminan con extension .conf de un directorio cualquiera
# pasado como parametro. A su vez, chequear que el parametro exista y sea realmente un directorio.

if [ ! $# -eq 1 ]; then
        echo "Debe pasar un parametro"
        exit
fi

if [ ! -d $1 ]; then
        echo "El parametro debe ser un directorio"
        exit
fi

find $1 -type f -perm 755 -exec echo {} >> perm \;
find $1 -type f -name "*.conf" -exec echo {} >> exten \;
```

Ejercicio 24

```
#!/bin/bash

# Hacer un script que guarde en el archivo conectados todos los usuarios conectados al sistema y
# en el archivo procesos todos los procesos que se estan ejecutando en el sistema. Ademas, si al
# script le paso como parametro un usuario cualquiera me diga si esta conectado o no.

who | cut -d " " -f1 | sort --unique > conectados
ps aux > procesos

if [ $# -eq 1 ]; then
        if [ ! -d $1 ] && [ ! -f $1 ]; then
                ESTA=$(grep $1 conectados)
                if [ "$ESTA" = "" ]; then
                        echo "El usuario $1 NO esta conectado al sistema"
                        else
                        echo "El usuario $1 esta conectado al sistema"
                fi
        fi
fi
```

Ejercicio 25

```
#!/bin/bash

# Hacer un script que recorra un directorio cualquier pasado por parametro y borre el archivo mas grande
# en el 1er nivel de este directorio. Ademas guardar como informacion el nombre y tamaño del archivo borr
# en un archivo llamado archivo-borrado

if [ ! $# -eq 1 ]; then
        echo "Debe pasar un parametro"
        exit
fi

if [ ! -d $1 ]; then
        echo "El parametro debe ser un directorio"
        exit
fi

MAYOR_TAM=0

for i in $1/*; do
    TAM=$(stat $i | grep "Tam" | cut -d " " -f4)
    if [ $TAM -gt $MAYOR_TAM ]; then
        echo "Archivo: $i Tamaño: $TAM" > archivo-borrado
        ARCHIVO=$i
        let MAYOR_TAM=$TAM
    fi
done
```

```
rm $ARCHIVO
```

Ejercicio 26

```
#!/bin/bash

# Hacer un script que reciba como parametro un directorio cualquiera y nos devuelva el tamaño del mismo,
# expresado en la unidad correspondiente (Byte, KByte, MByte, etc); chequear que el parametro no se pase
# en blanco y que el directorio pasado exista. Guardar el tamaño del mismo en un archivo cualquiera y
# ademas mostrar por pantalla la leyenda: El directorio XX tiene un tamaño de X bytes

if [ ! $# -eq 1 ]; then
    echo "Debe pasar un parametro"
    exit
fi

if [ ! -d $1 ]; then
    echo "El parametro debe ser un directorio"
    exit
fi

TAM=$(du -sb $1 | cut -d " " -f1)
echo $TAM > archivo
echo "El directorio $1 tiene un tam de $TAM bytes"
```

Ejercicio 27

```
# Realizar un script que recibe un directorio y una palabra como argumentos,
# y retorna el top 3 de los archivos en dicho directorio que contienen
# la palabra mayor cantidad de veces.


#!/bin/bash

for file in $(ls $1); do
        if [ ! -f $1/$file ]; then
                continue
        fi
        counter=$(grep $2 $file --only-matching | wc -l)
        echo "$counter  $file" >> /tmp/output
done
cat /tmp/output | sort -nr | head -n3
rm /tmp/output
```

Ejercicio 28

```
# Se solicita realizar un script para averiguar que alumno realizó mayor cantidad
# de ejercicios previo al parcial.
# Para ello, se recibirá como argumento una lista (no se sabe exactamente cuantos)
# de directorios. El script debe validar que dichos directorios sean validos,
# y retornar el nombre del directorio que contiene mayor cantidad de arhivos
# con la extensión ".sh" cuyo dueño contenga permisos de ejecución.


#!/bin/bash

MAX=0
MAX_NAME=""
for dir in $@; do
        if [ ! -d $dir ]; then
                continue
        fi
        counter=$(find . -name "*.sh" -perm /u+x | wc -l)
        if [ $counter -gt $MAX ]; then
```

```
                MAX=$counter
                MAX_NAME=$dir
        fi
done
echo "$MAX_NAME contiene mayor cantidad de scripts: $MAX"
```

Ejercicio 29

```
# Se desea realizar un informe detallado de los usuarios conectados actualmente
# al sistema. Para ellos se debe crear un directorio "usuarios_conectados",
# que contenga un archivo por cada uno de los usuarios conectados al momento de
# ejecución del script. Cada archivo tendrá la siguiente estructura de
# nombre: "datos_<nombre-usuario>.txt". Dentro de cada archivo, detallar en lineas
# independientes: id del usuario, nombre completo, ruta a tu home y shell configurado
# por defecto.
# Finalmente empaquetar el directorio "usuarios_conectados".


#!/bin/bash

if [ ! -d "usuarios_conectados" ]; then
        mkdir usuarios_conectados
fi

for usuario in $(who | cut -d" " -f1 | sort -u); do
        datos=$(cat /etc/passwd | grep $usuario)

        id=$(echo $datos | cut -d":" -f 3)
        echo "id: $id"  >> usuarios_conectados/datos_$usuario.txt

        nombre=$(echo $datos | cut -d":" -f 5)
        echo "nombre: $nombre"  >> usuarios_conectados/datos_$usuario.txt

        home=$(echo $datos | cut -d":" -f 6)
        echo "home: $home"  >> usuarios_conectados/datos_$usuario.txt

        shell=$(echo $datos | cut -d":" -f 7)
        echo "shell: $shell"  >> usuarios_conectados/datos_$usuario.txt
done
```

Ejercicio 30

```
# Realizar un script utilizando una estructura repetitiva "for", que recorra
# un directorio cualquiera pasado como argumento, y devuelva el top 3 de los
# archivos (no directorios) con nombre mas largo encontrados en dicho directorio.
# El script debe devolver el top 3 en orden descendente, incluyendo
# longitud y nombre del archivo.


#!/bin/bash

if [ ! $1 ]; then
        echo "Falta parametro"
        exit
fi

for file in $(ls $1); do
        if [ ! -f $1/$file ]; then
                continue
        fi
        length=$(echo $file | wc -m)

        echo "$length  $file" >> /tmp/output
done
cat /tmp/output | sort -nr | head -n3
rm /tmp/output
```

Ejercicio 31

```
# Realizar un script en el cual se reciba un directorio como parámetro.
# Copiar todos los archivos ejecutables dentro del directorio a otro directorio llamado "directorio_nuevo
# y eliminarle la primera y la última linea a cada uno de los archivos encontrados.
# Mostrar por pantalla la cantidad de archivos copiados de esta manera.

#!/bin/bash

if [ ! $1 ]; then
        echo "El script solicita 1 parametro"
        exit
fi

if [ ! -d $1 ]; then
        echo "El parametro debe ser un directorio"
        exit
fi

mkdir directorio_nuevo
find $1 -type f -perm /u+x -exec cp {} ./directorio_nuevo \;

for ejecutable in $(ls directorio_nuevo); do
    sed -i '1d;$d' directorio_nuevo/$ejecutable
done

copiados=$(ls directorio_nuevo | wc -l)
echo "Fueron copiados $copiados archivos"
```

jaanauati commented on 2 Jul 2015

Ejercicio 1: al usar expansión de esa forma tener en cuenta que no expande a nombres de archivos ocultos (.soyDeBocayEstoyEscondido).

```
for i in $1/*;
```

a no ser que previamente hagas algo así (no lindo):

```
shopt -s dotglob
```

Quizá otra forma para hacer lo que piden:

```
find eldirectorio -type f -depth 1 -exec du \{\} \; | sort -nr| head -n1 |awk '{ print $2, $1 }'
```

mxlian commented on 3 Jul 2015

Ejercicio 23: el `-exec echo {}` no es necesario si no se modifica el contenido de {}. Es mas limpio y eficiente simplemente guardar el output the find:

```
    find $1 -type f -perm 755 > perm
```

Performance test en un directorio con muchos archivos (~46000):

```
# Original command
$  time find $1 -type f -perm 755 -exec echo {} >> perm \;
real    0m15.127s
user    0m6.107s
sys 0m8.941s

# Optimized command
$  time find $1 -type f -perm 755 > perm2
real    0m0.478s
user    0m0.169s
sys 0m0.270s

# How many entries (count lines)
```

```
$ cat perm | wc -l
45918
```