**#!/bin/bash**

#simula la ejecución de un proceso escribiendo puntos

#se introduce el intervalo entre punto y punto, si no se introduce será 5

#programa principal

**read** -p "introduce intervalo: " intervalo

**while true;do echo** -n "." **sleep** $intervalo

**done**

#!/bin/bash

#CLACULADORA DE OPERACIONES SOBRE 2 NÚMEROS PASADOS COMO PARÁMETROS

#COMPROBAR PARÁMETROS

#LLAMAR A LAS FUNCIONES DE LAS OPERACIONES #VISUALIZACIÓN DE RESULTADOS

menu ()**{**

**echo** "Elige opción del siguiente menú:"

**echo** "

1. suma
2. resta
3. producto
4. división
5. resto"

**}**

#perfecto con números enteros. Fallos con reales solucionado con () suma ()**{**

resultado=`echo "($1)+($2)" **|** bc -l` #resultado=$(($1+$2)) #sólo para enteros

**}**

resta()**{**

resultado=`echo "($1)-($2)" **|** bc -l` #resultado=$(($1-$2))

**}**

producto()**{**

resultado=`echo "($1)\*($2)" **|** bc -l` #resultado=$(($1\*$2))

**}**

division()**{**

**if [** "$2" != "0" **];then** #para reales -ne es para enteros resultado=`echo "scale=2;($1)/($2)" **|** bc -l`

**else**

**echo** "División por 0"

**fi**

**}**

resto()**{**

**if [** $2 -ne 0 **];then**

resultado=**$((**$1%$2**))** #sólo para enteros no para reales

**else**

**echo** "División por 0"

**fi**

**}**

#PROGRAMA PRINCIPAL

**if [** $# -eq 2 **];then**

#comprobación de números enteros y reales

**echo** $1 **| egrep** -q '^\-?[0-9 .,]+$'

**if [** $? -ne 0 **];then**

**echo** "El parámetro $1 no es número"

**exit fi**

**echo** $2 **| egrep** -q '^\-?[0-9 .,]+$'

**if [** $? -ne 0 **];then**

**echo** "El parámetro $2 no es número"

**exit**

**fi**

**echo** "Se procede a realizar las operaciones que elijas del siguiente menú" menu #llamada a función

**read** -p "Introduce opción del menú: " resp

**echo** $resp **| egrep** -q '^?[0-9 .,]+$' #nos aseguramos números

**while [** $? -eq 0 **] && [** $resp -ge 1 **];do case** $resp **in**

1)operacion=" + "**; echo** "Se va a realizar $operacion "**;**suma $1 $2**;;** 2)operacion=" - "**;echo** "Se va a realizar $operacion "**;**resta $1 $2**;;** 3)operacion=" \* "**;echo** "Se va a realizar $operacion "**;**producto $1 $2**;;** 4)operacion=" / "**;echo** "Se va a realizar $operacion "**;**division $1 $2**;;** 5)operacion=" % "**;echo** "Se va a realizar $operacion "**;**resto $1 $2**;;**

\*)**echo** "opción $resp no válida y se termina"**;break;;** #sale

**esac**

**echo** "$1 $operacion $2 = $resultado"

**read** -p "Introduce opción del menú: " resp

**done else**

**echo** "Debe introducir 2 números"

**fi**

#|/bin/bash

#SE PASA UN DIRECTORIO

#LO COMPRIME CON EL NOMBRE yyyy-mm-dd nombre-direc.tar.gz

#comprobar el parámetro directorio #indicar el nombre

#empaquetar y comprimir con ese nombre

**if [** $# -eq 1 **];then if [** -d $@ **];then**

#indicar nombre fecha=`date +%F`

**tar** -cf $fecha$@.tar $@

**gzip** $fecha$@.tar

**else**

**echo** "El parámetro $@ no es directorio"

**fi else**

**echo** "Debe introducir un parámetro"

**fi**

#!/bin/bash

#averiguar el pid del script

#indicar si el pid es mayor o menor que el número introducido #cuando lo adivine debe mostar el número de intentos

#visualización del pid para comprobar el funcionamiento valor=$$

**echo** "El pid es $valor" intentos=0

**read** -p "Introduce el pid que creas: " num**;**intentos=**$((**$intentos+1**))** #comprobar número

**echo** $num **| egrep** -q '?[0-9]+$'

**if [** $? -eq 0 **];then**

**while [** $num -ne $valor **];do if [** $num -lt $valor **];then**

**echo** "El pid es mayor, vuelve a intentarlo" intentos=**$((**$intentos+1**))**

**elif [** $num -gt $valor **];then**

**echo** "El pid es menor, vuelve a intentarlo" intentos=**$((**$intentos+1**))**

**fi**

**read** -p "Introduce el pid que creas: " num

**echo** $num **| egrep** -q '?[0-9]+$'

**if [** $? -eq 0 **];then continue**

**else**

**echo** "no has introducido un número"**;exit fi**

**done**

**echo** "Enhorabuena has acertado el pid del script con $intentos intentos"

**else**

**echo** " $num no es número"

**fi**

#!/bin/bash

#crear archivo listaetc con los ficheros con permiso de lectura de /tmp #creación de archivo

**touch** listaetc

#comprobar que existe /tmp

**if [** -d /etc **];then**

#recorrer e introducir en lista etc

**for** i **in** `ls **/**etc**/**\*`**;do**

**if [** -f $i -a -r $i **];then**

**basename** $i **>>** listaetc #para que sólo salga el nombre

**fi done**

**else**

**echo** "El directorio /tmp no existe"

**fi**

#muestra los ficheros que contiene services

**echo** "Los ficheros que contiene service de listaetc son:"

**cat** listaetc **| grep** -w "services" #!/bin/bash

#cuenta el número de ficheros y directorios de un directorio pasado

#comprobación de directorio

**if [** $# -eq 1 **];then if [** -d $@ **];then**

nd=-1**;**nf=0

**for** i **in** `du -a $@`**;do** #du cuenta el directorio indicado

**if [** -d $i **];then**

nd=**$((**$nd+1**)) elif [** -f $i **];then**

nf=**$((**$nf+1**)) fi**

**done**

**echo** "directorios hay $nd" **echo** "ficheros hay $nf" **else**

**echo** "$@ no es directorio"

**fi else**

**echo** "Debe introducir 1 parámetro"

**fi**

#!/bin/bash

#comprueba que directorio pasado está en directorio activo y si está vacío

#comprobación de parámetro

**if [** $# -eq 1 **];then if [** -d $@ **];then**

#comprobación en directorio activo **ls** -R **| grep** -w "$@" **1>**/dev/null **if [** $? -eq 0 **];then**

**echo** "$@ está en directorio activo"

**else**

**echo** "$@ está en directorio activo"

**fi**

#comprobación de vacío contenido=`ls $@ **|** wc -l`

**if [** $contenido -eq 0 **];then echo** "$@ está vacío"

**else**

**echo** "$@ está no vacío"

**fi else**

**echo** "El parámetro $@ no es directorio"

**fi else**

**echo** "Debe introducir un parámetro"

**fi**

#!/bin/bash

#copia todos los ficheros del directorio actual a cgs #no dice nada de recursividad

#si el directorio no está debe crearse

**if [** -d ./cgs **];then**

**echo** "El directorio cgs ya existe"

**else**

**mkdir** cgs

**fi**

#copia todos los ficheros del directorio actual a cgs

**for** i **in** `ls`**;do if [** -f $i **];then**

**cp** $i cgs

**fi done**

#comprobación de la copia

**echo** "Este es el resultado de la copia"

**ls** cgs #!/bin/bash

#dice el fichero del directorio actual que tiene más líneas lineas\_mayor=0

fichero=""

**for** i **in** `ls`**;do if [** -f $i **];then**

num\_lin\_actual=`cat "$i" **|** wc -l`

**echo** $i $num\_lin\_actual # para comprobar resultados

**if [** $num\_lin\_actual -gt $lineas\_mayor **];then**

lineas\_mayor=$num\_lin\_actual fichero="$i"

**fi fi**

**done**

#visualización de resultados

**echo** "El fichero que más lineas contien del directorio actual es $fichero que contiene

$lineas\_mayor" #!/bin/bash

#comprobar si el fichero pasado tiene permisos de lectura

#si tiene permiso de lectura mostrar contenido en modo paginado

**if [** $# -eq 1 **];then**

#hay que buscalo en el directorio de trabajo

fi=`du -a **|** grep -w "$@" **|** cut -f2` #por si está en subdirectorio del directorio de trabajo #du para obtener la ruta

**if [** -f $fi **];then if [** -r $fi **];then**

**echo** "$@ tiene permisos de lectura y su contenido es:"

**more** $fi

**else**

**echo** "$@ no tiene permisos de lectura"

**fi else**

**echo** "$@ no es fichero"

**fi else**

**echo** "Debe introducir 1 parámetro"

**fi** #!/bin/bash #crea menús

#CUENTA USUARIOS CONECTADOS

#CUENTA USUARIOS CON DIRECTORIO HOME Y LOS ALMACENA EN all.users, #LISTA IDENTIFICADORES DE USUARIOS CONECTADOS AL SISTEMA,

#LISTA CUENTAS DE USUARIOS ORDENADOR POR NOMBRE E ID #DIA MES AÑO EN VARIABLE MOSTRADA EN PANTALLA

#CAMBIA A MAYÚSCULAS LOS NOMBRES DE FICHEROS PASADOS POR PARÁMETRO

#menú de opciones

menu()**{ echo** "

1. CUENTA USUARIOS CONECTADOS
2. CUENTA USUARIOS CON DIRECTORIO HOME Y LOS ALMACENA EN all.users,
3. LISTA IDENTIFICADORES DE USUARIOS CONECTADOS AL SISTEMA,
4. LISTA CUENTAS DE USUARIOS ORDENADOR POR NOMBRE E ID
5. DIA MES AÑO EN VARIABLE MOSTRADA EN PANTALLA
6. CAMBIA A MAYÚSCULAS LOS NOMBRES DE FICHEROS PASADOS POR PARÁMETRO

"

**}**

uno()**{**

**echo** "El número de usuarios conectados es: "**;**users **| wc** -w

**}**

dos()**{**

**echo** "El número de usuarios con directorio home es: "**;ls** /home **| wc** -l #almacenamiento de usuarios en allusers

**ls** /home **>** allusers

**echo** "comprobación de allusers"

**cat** allusers

**}**

tres()**{**

**echo** "Lista de usuarios conectados al sistema: "**;**users

**}**

cuatro()**{**

**echo** "---Cuentas de usuarios ordenadas por nombre:--- "

**cat** /etc/passwd **| cut** -d: -f1,3 **| sort** -t: -k1 **echo** "---Cuentas de usuarios ordenadas por id: " **cat** /etc/passwd **| cut** -d: -f1,3 **| sort** -t: -k2 -n

**}**

cinco()**{**

**echo** "Dia-més-año: "`date +%d-%m-%Y`

**}**

seis()**{**

#comprobación de parámetros

**if [** $# -ne 0 **];then for** i **in** $\***;do**

**if [** -f $i **];then** # los ficheros son del directorio de trabajo #traducción del nombre

nuevo=`echo "$i" **|** tr [:lower:] [:upper:]` #sustitución del archivo

**mv** $i $nuevo

**echo** "Comprobación de cambio de nombres de ficheros"

**ls | grep** $nuevo

**else**

**echo** "El parámetro $i no es fichero"

**fi done**

**else**

**echo** "No ha introducido nombres de ficheros a cambiar el nombre"

**fi**

**}**

#PROGRAMA PRINCIPAL

#llamadas a funciones usando menú y eligiendo opciones de menú

**while true;do** #repite menú mientras introduce opción válida menu

**read** -p "Introduce opción del menú anterior: " opc

**case** $opc **in**

* 1. **clear;**uno**;;**
  2. **clear;**dos**;;** 3)**clear;**tres**;;**

4)**clear;**cuatro**;;** 5)**clear;**cinco**;;** 6)**clear;**seis $@**;;**

\*)**echo** "Ha introducido una opción no válida"**;break;; esac**

**done** #!/bin/bash #menú 2

#visualización de particiones, memoria libre, espacio ocupado por directorios, versión completa del sistema mediante menú de opciones

menu()**{ echo** "

1)Visualización de particiones 2)Visualización de memoria libre

1. Visualización de espacio ocupado por directorios 4)Visualización de versión completa del sistema

"

**}**

uno()**{**

#para las particiones debe ser root usuario=`id -u`

**if [** $usuario -eq 0 **];then**

**echo** "Las particiones del sistema son:" fdisk -l

**else**

**echo** "Debe ser root para conocer las particiones"

**fi**

**}**

dos()**{**

**echo** "La memoria libre es:"

**free** -h

**}**

tres()**{**

**echo** "El espacio ocupado por directorios es:"

**du** -a

**}**

cuatro()**{**

**echo** "La versión completa del sistema es:"

**uname** -a

**}**

#programa principal

**while true;do**

menu

**read** -p "Introduce una opción del menú anterior: " opc

**case** $opc **in**

* 1. **clear;**uno**;;**
  2. **clear;**dos**;;** 3)**clear;**tres**;;** 4)**clear;**cuatro**;;**

\*)**echo** "Debe introducir una opción válida"**;break;; esac**

**done**

#!/bin/bash

#Mostrar la fecha del sistema.

#Mostrar información sobre qué usuarios han iniciado sesión y qué están haciendo. Para ello utilice el comando w.

#Mostrar los 10 procesos que consumen más memoria. Para ello utilice el comando ps. #Mostrar los 10 procesos que consumen más CPU. Para ello utilice el comando ps. #Mostrar el estado de la red. Para ello utilice el comando netstat.

#Salir del menú.

menu()**{ echo** "

1. MOSTRAR FECHA
2. MOSTRAR USUARIOS CONECTADOS Y QUÉ HACEN
3. MOSTRAR 10 PROCESOS QUE CONSUMEN MÁS MEMORIA
4. MOSTRAR 10 PROCESOS QUE CONSUMEN MÁS CPU
5. MOSTRAR EL ESTADO DE LA RED
6. SALIR

"

**}**

uno()**{**

**echo** "LA FECHA DEL SISTEMA ES: "`date +%d-%m-%Y`

**}**

dos()**{**

**echo** "LOS USUARIOS CONECTADOS Y LO QUE HACEN SON:"**; w**

**}**

tres()**{**

**echo** "LOS 10 PROCESOS QUE CONSUMEN MÁS MEMORIA"

**ps** -auxf **| sort** -r -k4 **| head** -10

**}**

cuatro()**{**

**echo** "LOS 10 PROCESOS QUE CONSUMEN MÁS CPU"

**ps** -auxf **| sort** -r -k3 **| head** -10

**}**

cinco()**{**

**echo** "EL ESTADO DE LA RED ES:"**;**netstat -s

**}**

#PROGRAMA PRINCIPAL

**while true;do**

menu

**read** -p "INTRODUCE OPCIÓN DEL MENÚ ANTERIOR: " opc

**case** $opc **in**

* 1. **clear;**uno**;;**
  2. **clear;**dos**;;** 3)**clear;**tres**;;** 4)**clear;**cuatro**;;** 5)**clear;**cinco**;;** 6)**exit;;**

\*)**echo** "OPCIÓN NO VÁLIDA"**;break;; esac**

**done**

#!/bin/bash

#Menú con 4 opciones que actúe sobre un fichero que se le pase como argumento.

#1 Buscar el fichero y mostrar su camino absoluto si existe o decir que no se ha encontrado.

#2 Cambiar los permisos al fichero. Hay que pedir los nuevos permisos y verificar que los permisos se han actualizado o decir porque no se han podido cambiar.

#3 Buscar una cadena en el fichero. Hay que pedir la cadena a buscar y mostrar las líneas en las que aparece, o decir que no se ha podido encontrar.

#4 Salir menu()**{**

**echo** "

1. BUSCAR FICHERO Y MOSTRAR CAMINO ABSOLUTO
2. CAMBIAR PERMISOS AL FICHERO
3. BUSCAR CADENA EN FICHERO
4. SALIR

"

**}**

uno()**{**

#supongo que está en directorio actual o subdirectorios

**du** -a **| grep** -w "$@" **1>**/dev/null **if [** $? -eq 0 **]; then**

**echo** "EL FICHERO SE HA ENCONTRADO "

**du** -a **| grep** -w "$@" #supongo que está en directorio actual o subdirectorios

**echo** "Y SU CAMINO ABSOLUTO ES: "`readlink -f $@`

**else**

**echo** "El fichero $@ no se ha encontrado"

**fi**

**}**

dos()**{**

**echo** "LOS PERMISOS ACTUALES DEL FICHERO $@ son "

**ls** -lR **| grep** -w "$@"

**echo** "INDICA LOS PERMISOS QUE QUIERES CAMBIAR EN EN VALOR OCTAL"

**read** -p "Permiso de usuario " perus **read** -p "Permiso de grupo " pergr **read** -p "Permiso de otros " perot **chmod** $perus$pergr$perot $@

**echo** "VERIFICACIÓN DE CAMBIO DE PERMISOS"

**ls** -lR **| grep** -w "$@"

**}**

tres()**{**

**read** -p "INTRODUCE LA CADENA A BUSCAR EN EL FICHERO " cad

**grep** -w "$cad" $@ **1>**/dev/null **if [** $? -eq 0 **];then**

**echo** "LA CADENA $cad SE HA ENCONTRADO en $@"

**else**

**echo** "LA CADENA $cad NO SE HA ENCONTRADO EN $@ "

**fi**

**}**

#PROGRAMA PRINCIPAL

#comprobación de parámetro

**if [** $# -eq 1 **];then**

**if [** -f $@ **];then** #sólo funciona para ficheros del directorio actual #llamada a menú mientras introduzca un opción válida

**while true;do**

menu

**read** -p "Introduce un opción del menú " opc

**case** $opc **in** 1)**clear;**uno $@**;;** 2)**clear;**dos $@**;;** 3)**clear;**tres $@**;;** 4)**exit;;**

\*)**echo** "Ha introducido un opción no válida"**;break esac**

**done else**

**echo** "El parámetro $@ no es fichero"

**fi else**

**echo** "Debe introducir un parámetro"

**fi**

#!/bin/bash

#MODIFICACIÓN PARA QUE SEA DEL DIRECTORIO DE TRABAJO Y SUBDIRECTORIO

#Menú con 4 opciones que actúe sobre un fichero que se le pase como argumento.

#1 Buscar el fichero y mostrar su camino absoluto si existe o decir que no se ha encontrado.

#2 Cambiar los permisos al fichero. Hay que pedir los nuevos permisos y verificar que los permisos se han actualizado o decir porque no se han podido cambiar.

#3 Buscar una cadena en el fichero. Hay que pedir la cadena a buscar y mostrar las líneas en las que aparece, o decir que no se ha podido encontrar.

#4 Salir menu()**{**

**echo** "

1. BUSCAR FICHERO Y MOSTRAR CAMINO ABSOLUTO
2. CAMBIAR PERMISOS AL FICHERO
3. BUSCAR CADENA EN FICHERO
4. SALIR

"

**}**

uno()**{**

#supongo que está en directorio actual o subdirectorios

**du** -a **| grep** -w "$@" **1>**/dev/null **if [** $? -eq 0 **]; then**

**echo** "EL FICHERO SE HA ENCONTRADO "

**du** -a **| grep** -w "$@"

**echo** "Y SU CAMINO ABSOLUTO ES: "`readlink -f $@`

**else**

**echo** "El fichero $@ no se ha encontrado"

**fi**

**}**

dos()**{**

**echo** "LOS PERMISOS ACTUALES DEL FICHERO $@ son "

nombre=`basename $@`

**ls** -lR **| grep** -w "$nombre" # aquí necesita nombre

**echo** "INDICA LOS PERMISOS QUE QUIERES CAMBIAR EN EN VALOR OCTAL"

**read** -p "Permiso de usuario " perus **read** -p "Permiso de grupo " pergr **read** -p "Permiso de otros " perot

**chmod** $perus$pergr$perot $@ #aquí necesita la ruta

**echo** "VERIFICACIÓN DE CAMBIO DE PERMISOS"

**ls** -lR **| grep** -w $nombre

**}**

tres()**{**

**read** -p "INTRODUCE LA CADENA A BUSCAR EN EL FICHERO " cad

**grep** -w "$cad" $@ **1>**/dev/null **if [** $? -eq 0 **];then**

**echo** "LA CADENA $cad SE HA ENCONTRADO en $@"

**else**

**echo** "LA CADENA $cad NO SE HA ENCONTRADO EN $@ "

**fi**

**}**

#PROGRAMA PRINCIPAL

#comprobación de parámetro

**if [** $# -eq 1 **];then**

#busco en directorio de trabajo y subdirectorios fich=`du -a **|** grep -w "$@" **|** cut -f2`

**if [** -f $fich **];then**

#llamada a menú mientras introduzca un opción válida

**while true;do**

menu

**read** -p "Introduce un opción del menú " opc

**case** $opc **in** 1)**clear;**uno $fich**;;** 2)**clear;**dos $fich**;;** 3)**clear;**tres $fich**;;** 4)**exit;;**

\*)**echo** "Ha introducido un opción no válida"**;break esac**

**done else**

**echo** "El parámetro $@ no es fichero o no está en directorio trabajo"

**fi else**

**echo** "Debe introducir un parámetro"

**fi**

#!/bin/bash

#INFORME DE FICHEROS DE UN DIRECTORIO PASADO

#RECIBE 3 PARÁMETROS: fichero donde se guarda el informe, tipo de ficheros a incluir en informe, directorio donde están los ficheros

#El informe deberá contener la siguiente información y estructura (por líneas): #La fecha y la hora actual, (la de generación del informe).

#El directorio que contiene los ficheros del informe.

#listado de los ficheros del tipo indicado del directorio indicado #Una línea en blanco

#Informe realizado por “el nombre del usuario que ejecuta el script"

#Se presentarán informes en pantalla de los posibles errores relativos a: #los parámetros pasados o a la falta de ellos.

#Si el fichero informe ya existe, se advertirá al usuario y se le dará la opción de poder sobrescribirlo o abandonar la ejecución del script.

#Si se ha de sobrescribir el fichero, deben tenerse en cuenta la posibles trabas, e informar de ellas.

#si realmente no se puede modificar el fichero ya existente (p.e.: está en un CD ROM). #si no pudiera crearse el fichero.

#generación de informe informe()**{**

**echo** " "

**echo** "FECHA Y HORA ACTUAL: "`date +%d-%m-%Y" "%T` **echo** "EL DIRECTORIO DONDE ESTÁN LOS ARCHIVOS ES: $2" **echo** "LISTADO DE FICHEROS CONTENIDOS EN EL INFORME: "

**find** $2 -type "$1" -print

**echo** "EL USUARIO QUE HA REALIZADO EL INFORME ES: "`whoami`

**echo** " "

**}**

#PROGRAMA PRINCIPAL CON COMPROBACIONES Y LLAMADA A informe

#comprobación de parámetros

**if [** $# -eq 3 **];then**

#comprobación de $2:tipo de fichero

**case** $2 **in**

"f")**echo** "Se va a realizar informe de ficheros"**;;** "d")**echo** "Se va a realizar informe de directorios"**;;** "p")**echo** "Se va a realizar informe de pipes"**;;**

"h")**echo** "Se va a realizar informe de enlaces simbólicos"**;;** "b")**echo** "Se va a realizar informe de archivos de bloques"**;;** "c")**echo** "Se va a realizar informe de fucheros de caracteres"**;;**

\*)**echo** "tipo de fichero $2 no admitido"**;exit;; esac**

#comprobación de $3

**if [** -d $3 **];then**

#comprobación de $1 en directorio de trabajo #podría mejorarse buscándolo en otro lugar **if [** -f $1 **];then**

**echo** "$1 ya existe"

#mejora:no salir hasta que no de respuesta correcta

**read** -p "¿Quiere modificarlo? s/n: " resp

**case** $resp **in**

s**|**S)

**echo** "Se va a modificar el informe"

#comprobación de sobreescritura = si tenemos permisos

**echo** "Se comprueba permisos de escritura:"

**if [** -w $1 **];then**

**echo** "Si tiene permisos de escritura y se hace informe" #se hace informe con parámetros

informe $2 $3 **>** $1

**else**

**echo** "El usuario no dispone de permisos de escritura y no se puede realizar

informe"

**fi**

**;;**

n**|**N)**echo** "No se modifica el informe y salimos "**;exit;;**

\*)**echo** "Debe introducir una respuesta correcta"**;; esac**

**else**

#antes de crearlo comprobamos si está en otro lugar fich=`find . -name "$1"`

**if [** "$fich" != "" **];then**

#se hace el informe con parámetros

informe $2 $3 **>**$fich **else**

**touch** $1

#comprobación de creación de informe

**if [** $? -ne 0 **];then**

**echo** "Error en la creación del fichero informe: no se puede generar el informe"

**else**

#se hace el informe con parámetros informe $2 $3 **>**$1

**fi fi**

**fi else**

**echo** "El parámetro $3 no es directorio y no se puede realizar informe"

**fi else**

**echo** "Debe introducir 3 parámetros"

**fi** #!/bin/bash #menú

#1)pide ruta de directorio busca enlaces duros (INODOS) #2)listado de alumnos y número

menu()**{ echo** "

1. busca enlaces duros en un la ruta de un directorio 2)lsita de alumnos y número

"

**}**

uno()**{**

**echo** "VISUALIZACIÓN DE ENLACES DUROS DE DIRECTORIO INDICADO"

**read** -p "Introduce el la ruta de un directorio: " ruta

**if [** -d $ruta **];then**

**echo** "inodos del directorio $ruta"

**ls** -i -R $ruta

**else**

**echo** "$ruta no es directorio o no se encuentra en el directorio de trabajo"

**fi**

**}**

dos()**{**

**echo** "LISTADO DE ALUMNOS Y NÚMERO"

#comprobación de existencia de fichero lista en directorio de trabajo y subdirectorios listab=`find . -name "lista"`

**if [** -f $listab **];then**

#llenado de nombre apellido1 apellido2 en el fichero

**echo** "LISTA YA EXISTE"

**read** -p "Desea borrar su contenido s/n: " resp

**case** $resp **in** s**|**S)borrado=1**;;** n**|**N)borrado=0**;;**

\*)**echo** "Respuesta no válida"**;exit;; esac**

llenado $borrado $listab

**else**

**echo** "El fichero lista no existe y procedemos a crearlo"

**touch** lista

**if [** $? -ne 0 **];then**

**echo** "Error al crear el fichero lista y salimos"**;exit else**

borrado=0

llenado $borrado lista

**fi fi**

**}**

llenado()**{**

**if [** $1 -eq 1 **];then**

**echo** "" **>** $2 #borrado de lista primero=0

**else**

primero=`cat $2 **|** tail -1 **|** tr -s " " " " **|** cut -d" " -f1`

**fi**

**read** -p "Introduzca el número de alumnos a insertar: " num

**for** i **in** `seq 1 $num`**;do**

**read** -p "Introduce nombre: " nombre **read** -p "Introduce apellido1: " apell1 **read** -p "Introduce apellido2: " apell2 orden=**$((**$i+$primero**))**

**echo** -e $orden "\t" $nombre "\t" $apell1 "\t" $apell2 **>>** $2

**done**

#comprobación de lista **echo** "Comprobación" **cat** $2

**}**

#PROGRAMA PRINCIPAL

**while true;do**

menu

**read** -p "Introduce opción del menú anterior: " opc

**case** $opc **in**

* 1. **clear;**uno**;;**
  2. **clear;**dos**;;**

\*)**echo** "Opción no válida"**;exit esac**

**done**

#!/bin/bash

#convierte a octal los permisos del archivo solicitado convertir()**{**

v1=0**;**v2=0**;**v3=0**;**v4=0 #necesario cuando se llaman varias veces

**for** i **in** `seq $2 $3`**;do** vu=`echo $1 **|** cut -c $i` **case** "$vu" **in**

"r")v1=4**;**v4=0**;;**

"w")v2=2**;**v4=0**;;**

"x")v3=1**;**v4=0**;;**

"-")v4=0**;;**

**esac done**

vu\_octal=**$((**$v1+$v2+$v3+$v4**))**

**echo** "Los permisos de $4 son: $vu\_octal"

**}**

**read** -p "introduce fichero " fich perm=`ls -l $fich **|** cut -d" " -f1`  **echo** "los permisos de $fich son: $perm" **if [** -f $fich **];then**

convertir $perm 2 4 usuario

convertir $perm 5 7 grupo

convertir $perm 8 10 otros

**else**

**echo** "El dato introducido no es fichero"

**fi**

#!/bin/bash

#cambiar permisos en octal y con caracteres #progrma principal

cambio\_octal()**{**

**echo** "cambio en octal" #introducir valores

**read** -p "introduce permisos para usuario " vu **read** -p "introduce permisos para grupo " vg **read** -p "introduce permisos para otros " vo #comprobar

**if [** $vu -ge 0 -a $vu -le 7 -a $vg -ge 0 -a $vg -le 7 -a $vo -ge 0 -a $vo -le 7 **];then**

#cambiar

**chmod** $vu$vg$vo $1

**echo** "Los permisos nuevos de $1 son: "`ls -l **|** grep $1 **|** cut -d" " -f1`

**else**

**echo** "Valores no válidos "

**fi**

**}**

validar()**{ case** $1 **in**

"+r"**|**"-r"**|**"+w"**|**"-w"**|**"+x"**|**"-x")valido=1**;;**

\*)valido=0**;; esac**

**}**

cambio\_caracteres()**{**

**echo** "cambio en caracteres +-rwx"

**read** -p "introduce permisos para usuario " vu **read** -p "introduce permisos para grupo " vg **read** -p "introduce permisos para otros " vo #comprobar

validar $vu validar $vg validar $vo

**if [** $valido -eq 1 **] ;then**

**echo** "valores válidos para cambiar permisos"

#considero una forma de cambiar los permisos pero la más completa es chmod uuugggooo+r+w+x+r+w+x+r+w+x o bien chmod u+r+w+x chmod g+r+w+x chmod o+r+w+x

**chmod** "ugo"$vu$vg$vo $1

**echo** "Los permisos nuevos de $1 son: "`ls -l **|** grep $1 **|** cut -d" " -f1`

**else**

**echo** "valores no válidos para cambiar permisos"

**fi**

#cambiar

**}**

cambiar()**{**

**echo** "Se van a cambiar los permisos"

**read** -p "Cómo prefiere cambiar octal(o)/caracteres(c) " resp

**case** $resp **in** o**|**O)cambio\_octal $1**;;** c**|**C)cambio\_caracteres $1**;;**

\*)**echo** "Opción no válida"**;exit;; esac**

**}**

#programa principal: funciona en directorio de trabajo no fuera

**read** -p "Desea cambiar los permisos del fichero o directorio que prefiera: " valor #comprobación de fichero o directorio

**if [** -f $valor -o -d $valor **];then**

**echo** "Los permisos actuales de $valor son: "`ls -l **|** grep $valor **|** cut -d" " -f1`

**read** -p "Desea cambiarlos s/n? " resp

**case** $resp **in**

s**|**S)cambiar $valor**;;**

n**|**N)**echo** "Ha elegido no cambiar y salimos"**;exit;;**

\*)**echo** "Opción no válida"

**esac else**

**echo** "Debe introducir un fichero o directorio"

**fi**

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

#EXAMEN DE PARÁMETROS Y OPERACIONES SOBRE FICHEROS

#EL SCRIPT SE EJECUTA nombre.sh [opciones][argumentos] #opciones es el parámetro1

#sin parámetro muestra el contenido del propio script

#-m comprueba que argumentos siguientes son ficheros y muestra contenido

#-x comprueba que los argumentos siguientes son programas ejecutables y los ejecuta #-p muestra propietarios de ficheros que recibe en argumentos

#FUNCIONES DE CADA OPCIÓN

uno()**{**

**echo** "SE EJECUTA LA OPCIÓN -m"

**shift**

**for** i **in** $\***;do**

**if [** -f $i -a -r $i **];then** #debe tener permiso

**echo** " "

**echo** "EL CONTENIDO DE $i"

**cat** $i

**echo** " "

**else**

**echo** "$i NO ES FICHERO O NO DISPONE DE PERMISOS"

**fi done**

**}**

dos()**{**

**echo** "SE EJECUTA LA OPCIÓN -x"

**shift**

**for** i **in** $\***;do**

**if [** -x $i **];then**

**echo** " "

**echo** "SE EJECUTA $i"

**sh** $i

**echo** " "

**else**

**echo** "$i NO ES EJECUTABLE"

**fi done**

**}**

tres()**{**

**echo** "SE EJECUTA LA OPCIÓN -p"

**shift**

**for** i **in** $\***;do**

**if [** -f $i **];then**

**echo** "EL PROPIETARIO DE $i ES "`ls -l $i **|** cut -d" " -f3`

**else**

**echo** "$i NO ES FICHERO"

**fi done**

**}**

#PROGRAMA PRINCIPAL

#COMPROBACIÓN DEL NÚMERO DE PARÁMETROS Y LLAMADA A LAS FUNCIONES

**if [** $# -eq 0 **];then**

**echo** "CONTENIDO DEL SCRIPT $0 ES:"

**echo** " "

**cat** $0

**echo** " "

**else**

#comprobación de argumentos

**if [** $# -ge 2 **];then case** $1 **in**

"-m")uno $\***;;**

"-x")dos $\***;;**

"-p")tres $\***;;**

\*)**echo** "Opción no válida"**;exit;; esac**

**else**

**echo** "Se necesitan como mínimo 2 parámetros"

**fi fi**

#!/bin/bash

#menú de tratamiento de usuarios debe ser root #1)crear usuario

#2)cambiar contraseña a usuario #3)crear grupo

#4)añadir usuario a grupo

#5)ver datos de usuario #6)borrar usuario #7)borrar grupo #0)salir

#FUNCIONES

menu()**{ echo** "

1. crear usuario
2. cambiar contraseña a usuario 3)crear grupo

4)añadir usuario a grupo 5)ver datos de usuario 6)borrar usuario 7)borrar grupo

1. salir "

**}**

uno ()**{**

**echo** "CREACIÓN DE USUARIO"

**read** -p "Introduce el nombre del usuario a crear: " usu1

**grep** -w "$usu1" /etc/passwd **1>**/dev/null **if [** $? -eq 0 **];then**

**echo** "El usuario $usu1 ya existe"

**else**

adduser $usu #adduser con directorio

**fi**

**}**

dos ()**{**

**echo** "CAMBIAR CONTRASEÑA A USUARIO"

**read** -p "Introduce el nombre del usuario a cambiar la contraseña: " usu2

**grep** -w "$usu2" /etc/passwd **1>**/dev/null **if [** $? -eq 0 **];then**

**passwd** $usu2

**else**

**echo** "El usuario $usu no existe"

**fi**

**}**

tres ()**{**

**echo** "CREACIÓN DE GRUPO"

**read** -p "Introduce el nombre del grupo a crear: " grupo

**grep** -w "$grupo" /etc/group **1>**/dev/null **if [** $? -eq 0 **];then**

**echo** "El grupo $grupo ya existe"

**else**

addgroup $grupo #adduser con directorio

**fi**

**}**

cuatro ()**{**

**echo** "AÑADIR USUARIO A GRUPO"

#el usuario y el grupo deben existir

**read** -p "Introduce el nombre del usuario: " usu4 **read** -p "Introduce el nombre del grupo: " grup4 usuario4=`grep -w "$usu" **/**etc**/**passwd` grupo4=`grep -w "$grupo" **/**etc**/**group`

**if [** "$usuario4" != "" **] && [** "$grupo4" != "" **];then**

adduser $usu4 $grup4

**else**

**echo** "No se puede añadir $usu4 a $grupo4 por que no están en el sistema"

**fi**

**}**

cinco ()**{**

**echo** "VER DATOS DE USUARIO"

**read** -p "Introduce el nombre del usuario a ver los datos: " usu5

**grep** -w "$usu5" /etc/passwd **1>**/dev/null

**if [** $? -eq 0 **];then**

**cat** /etc/passwd **| grep** $usu5 **| cut** -d: -f5

**else**

**echo** "El usuario $usu5 no existe"

**fi**

**}**

seis()**{**

**echo** "BORRAR USUARIO"

**read** -p "Introduce el nombre del usuario a borrar: " usu6

**grep** -w "$usu6" /etc/passwd **1>**/dev/null **if [** $? -eq 0 **];then**

deluser $usu6

**else**

**echo** "El usuario $usu6 no existe"

**fi**

**}**

siete ()**{**

**echo** "BORRAR GRUPO"

**read** -p "Introduce el nombre del grupo a borrar: " grup7

**grep** -w "$grup7" /etc/group **1>**/dev/null **if [** $? -eq 0 **];then**

delgroup $grup7

**else**

**echo** "El grupo $grup7 no existe"

**fi**

**}**

#PROGRAMA PRINCIPAL

#comprobación de root usu=`id -u`

**if [** $usu -eq 0 **];then while true;do**

menu

**read** -p "ELIGE OPCIÓN DEL MENÚ ANTERIOR: " resp

**case** $resp **in**

1. **clear;**uno**;;**
2. **clear;**dos**;;** 3)**clear;**tres**;;** 4)**clear;**cuatro**;;** 5)**clear;**cinco**;;** 6)**clear;**seis**;;** 7)**clear;**siete**;;** 8)**exit;;**

\*)**echo** "Opción no válida"**;exit;; esac**

**done**

**else**

**echo** "NO SE PUEDE EJECUTAR SI NO ES ROOT"**;exit**

**fi**

#!/bin/bash

#examen 3:genera informe del estado del sistema #admite parámetros

#-u usuario para indicar el usuario del que mostrar el sistema. Si no se indica realizará el informe del usuario actual

#-a para generar informe de todos los usuarios del sistema (distinto de usuarios conectados who)

#el informe mostrará #nombre de usuario

#número de procesos en ejecución, proceso más antiguo del usuario, listado de procesos del usuario

#número de directorios del usuario, número de ficheros regulares

#tamaño ocupado en disco por el usuario y porcentaje que este representa sobre el total

informe()**{**

#recibe el usuario a generar informe

**echo** " "

**echo** "GENERACIÓN DE INFORME DEL USUARIO: $1"

**echo** -n "Nº PROCESOS EN EJECUCIÓN: "

**ps** -f -u $1 --sort=start\_time **| sed** 1d **| wc** -l

**echo** "PROCESO MÁS ANTIGUO: "

**ps** -f -u $1 --sort=start\_time **| sed** 1d **| head** -1

**echo** "LISTADO DE PROCESOS DEL USUARIO:"

**ps** -f -u $1 --sort=start\_time

**echo** "NÚMERO DE DIRECTORIOS DEL USUARIO: "`ls -lR **/**home**/**$1 **|** grep ^d **|** wc -l `

**echo** "NÚMERO DE FICHEROS REGULARES: "`ls -lR **/**home**/**$1 **|** grep ^- **|** wc -l `

tamano=`du -s **/**home**/**$1 **|** cut -f1`

**echo** "TAMAÑO OCUPADO EN DISCO POR EL USUARIO: " $tamano

total=`df **/ |** tr -s " " " " **|** cut -d" " -f3 **|** sed 1d ` # sed borra la cabecera porcentaje=`echo "scale=2;$tamano/$total\*100" **|** bc -l`

**echo** "PORCENTAJE QUE EL TAMAÑO DEL USUARIO $tamano REPRESENTA SOBRE EL TOTAL $total:

$porcentaje %"

**echo** " "

**}**

#programa principal

#usuario que ejecuta el script usu=`whoami`

#comprobar parámetros

**if [** $# -eq 0 **];then**

informe $usu

**else**

**case** $1 **in**

"-u")

#debe haber un segundo parámetro con el nombre del usuario

**if [** $2 **];then**

#comprobación de existencia de $2 us=`grep -w "$2" **/**etc**/**passwd`

**if [** "$us" != "" **];then**

informe $2 **else** informe $usu

**fi else**

**echo** "No ha proporcionado el usuario y se genera informe del usuario actual" informe $usu

**fi**

**;;**

"-a") #todos los usuarios del sistema

**for** i **in** `cat **/**etc**/**passwd**|** cut -d: -f1`**;do** #solamente para usuarios con id >1000

us=`id -u $i`

**if [** $us -ge 500 **];then**

informe $i

**fi done**

**;;**

\*)**echo** "parámetro no válido"**;exit;; esac**

**fi**

#!/bin/bash #simula tree con +-

#tree [opción] directorio

#MUESTRA POR PANTALLA EL ÁRBOL DE SUBDIRECTORIOS DEL DIRECTORIO PASADO

#OPCIÓN -f MUESTRA FICHEROS, MARCANDO CON + LOS DIRECTORIOS Y - **TODO** LO QUE NO SEA DIRECTORIO

#AL FINAL DEL ÁRBOL DE SUBDIRECTORIOS INFORMA DE CUÁNTOS DIRECTORIOS Y FICHEROS REGULARES HA MOSTRADO

arbol()**{**

**if [** $1 -eq 1 **];then**

md='+'**;** mf='-'

**else**

md=" "**;** mf=" "

**fi**

**for** i **in** $2**/**\***;do** #recorrido recursivo nivel=`echo $i **|** tr -s '/' ' ' **|** wc -w` #nivel #tabula hasta el nivel

**if [** -f $i **];then**

**for** j **in** `seq 1 $nivel`**;do** #es necesario repetir por errores

**echo** -n " "

**done**

**echo** $mf`basename $i` cf=**$((**$cf+1**))**

**elif [** -d $i **];then**

**for** j **in** `seq 1 $nivel`**;do echo** -n " "

**done**

**echo** $md`basename $i` cd=**$((**$cd+1**))**

arbol $marca $i #recursividad

**fi done**

**}**

#comprobación de parámetros marca=0 # flag de -f cf=0**;**cd=0

**if [** $# -eq 1 -o $# -eq 2 **];then for** i **in** $\***;do**

**if [** "$i" == "-f" **];then**

marca=1

**shift**

**elif [** -d $i **];then**

**if [** $marca -eq 1 **];then echo** '+'$i

**else**

**echo** $i

**fi**

arbol $marca $i #visualización de contadores

**echo** "Se han encontrado: $cf ficheros"

**echo** "Se han encontrado: $cd directorios"

**else**

**echo** "$i no válido"

**fi done**

**else**

**echo** "Debe introducir un o dos parámetros"

**fi**

#!/bin/bash

#apartado a) del examen de Murcia de 2002 ¡¡PERFECTO!!

#copia de seguridad del directorio home del usuario pasado por parámetro #puede que haya usuarios que no tengan directorio personal (hay que verificar

#el nombre del fichero copia tendrá el formato /var/copias login\_aammdd[.extensión] (extensión corresponde al método utilizado para realizar la copia

#la copia podrá realizarse con rutas absolutas(a) o relativas (r) dependiendo de lo que se elija como primer parámetro

#SINTAXIS DE EJECUCIÓN copia\_seg opción usuario

#control de errores: número de parámetros, opción no válida, usuario no existente, usuario sin directorio personal

#función de realización de copia copia()**{**

**echo** "Se realiza la copia del usuario $2 usando la opción $1 " #comprobación de existencia de /var/copias

exis=`ls **/**var **|** grep -w "copias"`

**if [** "$exis" = "" **];then**

**mkdir** /var/copias #necesitamos permisos

**fi**

fecha=`date +%y%m%d`

**if [** "$1" = "a" **];then**

**echo** "copia con ruta absoluta"

**tar** -cvf /var/copias/$2\_$fecha.tar /home/$2 **1>**/dev/null **2>**/dev/null #para eliminar mensaje de /

**gzip** -9 /var/copias/$2\_$fecha.tar

**else**

**echo** "copia con ruta relativa"

da=`pwd` #guardo el directorio actual para poder volver después

**cd** /var/copias

**tar** -cvf $2\_$fecha.tar /home/$2 **1>**/dev/null **2>**/dev/null #tar no pone extensiones

**gzip** -9 /var/copias/$2\_$fecha.tar #gzip pone extensió

**cd** $da

**fi**

**}**

#programa principal #comprobación de usuario root usu=`id -u`

**if [** $usu -ne 0 **];then**

**echo** "Se requieren privilegios de root para realizar el programa"**;exit fi**

#comprobación de parámetros

**if [** $# -eq 2 **];then**

**if [** "$1" = "a" **] || [** "$1" = "r" **];then** #comprobación de usuario en /etc/home usu=`grep -w "$2" **/**etc**/**passwd`

**if [** "$usu" != "" **];then**

#comprobación de directorio personal

usu1=`find **/**home -name "$2"` # también ls /home | grep $2

**if [** "$usu1" != "" **];then**

copia $1 $2

**else**

**echo** "El usuario $2 no dispone de directorio personal"

**fi else**

**echo** "Usuario $2 no existente en del sistema"

**fi else**

**echo** "$1 no válido, debe ser a o r"

**fi else**

**echo** "Debe introducir 2 parámetros"

**fi**

#!/bin/bash

#SIMULACIÓN DE UNA PAPELERA CON MENÚ:EXAMEN DE VALENCIA

#1.Eliminar archivo proporcionando ruta completa

#2.Restaurar archivo a su ubicación original indicando el nombre de archivo #3.Vaciar la papelera

#4.Mostrar el contenido de la papelera #5.Salir

#FUNCIONES DEL MENÚ

menu()**{**

**echo** " 1.Eliminar archivo proporcionando ruta completa

2.Restaurar archivo a su ubicación original indicando el nombre de archivo 3.Vaciar la papelera

1. Mostrar el contenido de la papelera 5.Salir

"

**}**

uno()**{**

**echo** "Eliminar archivo proporcionando ruta completa"

**read** -p "Introduce el nombre del archivo con ruta absoluta: " ruta

**if [** -f $ruta **];then** #comprobación de existencia

**echo** $ruta **>>** archivo\_rutas #muevo las rutas al archivo\_rutas

**mv** $ruta papelera

**echo** "El archivo $ruta se ha movido a la papelera"

**echo** "comprobación" `ls papelera`

**else**

**echo** "El archivo $ruta no existe"

**fi**

**}**

dos()**{**

**echo** "Restaurar archivo a su ubicación original indicando el nombre de archivo"

**read** -p "Introduce el nombre del archivo a restaurar " nom

nombre=`find papelera -name $nom` #necesario porque nom no está en directorio de trabajo

**if [** "$nombre" = "" **];then**

**echo** "El archivo $nom no se encuentra en la papelera"

**else**

ruta=`grep -w "$nom" archivo\_rutas`

**mv** $nombre $ruta

**echo** "Se ha restaurado el archivo $nom a su ubicación $ruta"

**echo** "Se procede a eliminar su ruta del archivo\_rutas"

**sed** -i /$nom/d archivo\_rutas

**echo** "Comprobación de salida de papelera" `ls papelera **|** grep -w "$nom"` **echo** "Comprobación de vuelta al directorio de trabajo" **;ls | grep** -w "$nom" **echo** "Comprobación de borrado de ruta"**;grep** -w "$nom" archivo\_rutas

**fi**

**}**

tres()**{**

**echo** "Vaciar la papelera"

**rm** -r papelera/\*

**echo** "Comprobación:"`ls papelera`

**}**

cuatro()**{**

**echo** "Mostrar el contenido de la papelera "

**ls** -R papelera

**}**

#PROGRAMA PRINCIPAL DE LLAMADA A MENÚ

#creación del directorio papelera si no existe

**if [** ! -d papelera **];then**

**mkdir** papelera #no dice donde se crea

**fi**

**while true;do**

menu

**read** -p "Introduce opción del menú anterior " resp

**case** $resp **in**

* 1. **clear;**uno**;;**
  2. **clear;**dos**;;** 3)**clear;**tres**;;** 4)**clear;**cuatro**;;** 5)**exit;;**

\*)**echo** "Opción no válida"**;exit;; esac**

**done**

#!/bah/bash

#EXAMEN ANDALUCÍA 2010

#Implementa un shell-script que se detalla a continuación:

#Sintaxis: realiza [-{cbmpe}] fichero

#• Si se utiliza alguna de las opciones, no se podrán utilizar de forma conjunta. Solo se podrá elegir una de ellas.

#• Si se ejecuta sin ninguna opción se visualizará el fichero pasado por parámetro. #• Opción -c: copiará fichero a un directorio que se pedirá por teclado.

#• Opción -b: borrará fichero del directorio donde esté ubicado.

#• Opción -m: moverá fichero a un directorio que se pedirá por teclado.

#• Opción -p: ejecutará fichero de forma programada. Se pedirá el día, mes, hora y minuto en el que se deberá ejecutar. La salida la enviará a un fichero situado en el directorio hogar, denominado log. PROBLEMAS CON EL CRON AL PASA A LOG

#• Opción -e: Creará un shell-script, realiza2, y lo llamará. En realiza2 se ejecuta el fichero pasado a realiza. El resultado de ejecutar fichero se enviará por mail al usuario que indique la variable de entorno USERDEST="realiza\_root" que se ha de crear en el shell- script realiza y que utiliza realiza2. MUY HIPOTÉTICO

#• Cada una de las opciones enumeradas anteriormente, así como la visualización del fichero en ausencia de opciones, se realizarán utilizando funciones.

#• Se comprobará la existencia o no de los directorios que se pidan por teclado. Si no existen se interrumpirá la función correspondiente.

#• Se comprobará la existencia o no del fichero pasado por parámetro. Si no existe se pedirá uno por teclado hasta que este exista. Asumiremos que siempre se introduce algún nombre de fichero.

#• En el directorio hogar, tendremos un fichero denominado nejecs que contendrá el n° de veces que se ha ejecutado realiza.

#FUNCIONES

opciones()**{**

**echo** "Usted a introducido la opción $1 y procedemos a ejecutarla"

**case** $1 **in**

"-c")uno $1 $2**;;**

"-b")dos $1 $2**;;**

"-m")tres $1 $2**;;**

"-p")cuatro $1 $2**;;**

"-e")cinco $1 $2 **;;**

\*)**echo** "Opción no válida"**;break;; esac**

**}**

uno()**{**

**echo** "Copia fichero $2 a un directorio que se pide por teclado"

**}**

dos()**{**

**echo** "Borra fichero $2 del directorio donde esté ubicado"

**}**

tres()**{**

**echo** "Mueve fichero $2 a un directorio que se pedirá por teclado"

**}**

cuatro()**{**

**echo** "Ejecuta fichero $2 de forma programada"

**}**

cinco()**{**

**echo** "Crea un shell-script, realiza2, y lo llama"

**}**

visualizacion()**{**

**echo** "Visualización del fichero $1 con ausencia de opciones"

**cat** $1

**}**

fichero()**{**

**echo** "Comprobación del fichero $1"

**if [** -f $1 **];then echo** "$1 es fichero"

**else**

**read** -p "Introduzca fichero " fich

**while [** ! -f $fich **];do**

**read** -p "Introduzca fichero " fich

**done fi**

**return** $fich

**}**

#PROGRAMA PRINCIPAL

#comprobación del número de parámetros y del fichero #se asume que siempre se introduce algún fichero **case** $# **in**

1. **echo** "Ha introducido 1 parámetro"

**echo** "Comprobación del fichero $1"

**if [** -f $1 **];then echo** "$1 es fichero"

**else**

**read** -p "Introduzca fichero " fich

**while [** ! -f $fich **];do**

**read** -p "Introduzca fichero " fich

**done fi**

visualizacion $fich

**;;**

1. **echo** "Ha introducido 2 parámetros" **echo** "Comprobación del fichero $2" **if [** -f $2 **];then**

**echo** "$2 es fichero"

**else**

**read** -p "Introduzca fichero " fich

**while [** ! -f $fich **];do**

**read** -p "Introduzca fichero " fich

**done**

**fi**

opciones $1 $fich

**;;**

\*)**echo** "Sólo se permite 1 o 2 parámetros"**;break;; esac**

#!/bah/bash

#EXAMEN ANDALUCÍA 2010 ¡¡¡MUY BIEN!!!!!!!!!!!!!!!!!!!!!!!!!!!

#Implementa un shell-script que se detalla a continuación:

#Sintaxis: realiza [-{cbmpe}] fichero

#• Si se utiliza alguna de las opciones, no se podrán utilizar de forma conjunta. Solo se podrá elegir una de ellas.

#• Si se ejecuta sin ninguna opción se visualizará el fichero pasado por parámetro. #• Opción -c: copiará fichero a un directorio que se pedirá por teclado.

#• Opción -b: borrará fichero del directorio donde esté ubicado.

#• Opción -m: moverá fichero a un directorio que se pedirá por teclado.

#• Opción -p: ejecutará fichero de forma programada. Se pedirá el día, mes, hora y minuto en el que se deberá ejecutar. La salida la enviará a un fichero situado en el directorio hogar, denominado log. PROBLEMAS CON EL CRON AL PASA A LOG

#• Opción -e: Creará un shell-script, realiza2, y lo llamará. En realiza2 se ejecuta el fichero pasado a realiza. El resultado de ejecutar fichero se enviará por mail al usuario que indique la variable de entorno USERDEST="realiza\_root" que se ha de crear en el shell- script realiza y que utiliza realiza2. MUY HIPOTÉTICO

#• Cada una de las opciones enumeradas anteriormente, así como la visualización del fichero en ausencia de opciones, se realizarán utilizando funciones.

#• Se comprobará la existencia o no de los directorios que se pidan por teclado. Si no existen se interrumpirá la función correspondiente.

#• Se comprobará la existencia o no del fichero pasado por parámetro. Si no existe se pedirá uno por teclado hasta que este exista. Asumiremos que siempre se introduce algún nombre de fichero.

#• En el directorio hogar, tendremos un fichero denominado nejecs que contendrá el n° de veces que se ha ejecutado realiza.

#FUNCIONES

opciones()**{**

**echo** "Usted a introducido la opción $1 y procedemos a ejecutarla"

**case** $1 **in**

"-c")uno $2**;;**

"-b")dos $2**;;**

"-m")tres $2**;;**

"-p")cuatro $2**;;**

"-e")cinco $2 **;;**

\*)**echo** "Opción no válida"**;break;; esac**

**}**

uno()**{**

**echo** "Copia fichero $1 a un directorio que se pide por teclado" **read** -p "Introduce el nombre del directorio donde copiar " di1 **if [** -d $di1 **];then**

**cp** $1 $di1

**else**

**echo** "$di1 no es válido"

**fi**

**}**

dos()**{**

fi2=`readlink -f $1`

**echo** "Borra fichero $1 de la ruta $fi2 donde esté ubicado"

**rm** $fi2

**}**

tres()**{**

**echo** "Mueve fichero $1 a un directorio que se pedirá por teclado" **read** -p "Introduce el nombre del directorio donde copiar " di3  **if [** -d $di3 **];then**

**mv** $1 $di3

**else**

**echo** "$di3 no es válido"

**fi**

**}**

cuatro()**{** #probar con fichero ejecutable prueba1.sh

#Se pedirá el día, mes, hora y minuto en el que se deberá ejecutar.

#La salida la enviará a un fichero situado en el directorio hogar, denominado log

**echo** "Ejecuta fichero $1 de forma programada"

**if [** -x $1 **];then**

**if [** ! -f hogar/log **];then touch** hogar/log

**fi**

#ejecutar de forma programada **read** -p "Introduce minuto " min **read** -p "Introduce hora " hor **read** -p "Introduce dia " dia **read** -p "Introduce mes " mes

##¡¡¡¡¡MUY IMPORTANTE SUDO Y RUTAS ABSOLUTAS!!!!

**sudo** echo "$min $hor $dia $mes \* ana /home/ana/Documentos/$1>/home/ana/Documentos/‐

hogar/log" **>>** /etc/crontab

**else**

**echo** "$1 no es fichero ejecutable"

**fi**

**}**

cinco()**{** #probar con fichero ejecutable prueba1.sh #En realiza2 se ejecuta el fichero pasado a realiza.

#El resultado de ejecutar fichero se enviará por mail al usuario que indique la variable de entorno USERDEST="realiza\_root" que se ha de crear en el shell-script realiza y que utiliza realiza2.

**echo** "Crea un shell-script, realiza2, y lo llama"

**if [** -x $1 **];then**

**echo** "sh $1 | mail to $USERDEST" **>** realiza2.sh # se necesita tener instaldo mail (el buzón está en /var/spool/mail/usuario

**chmod** 777 realiza2.sh**; sh** realiza2.sh #comprobar en el buzón del usuario realiza\_root

**else**

**echo** "$1 no es fichero ejecutable"

**fi**

**}**

visualizacion()**{**

**echo** "Visualización del fichero $1 con ausencia de opciones"

**cat** $1

**}**

fichero()**{**

**echo** "Comprobación del fichero $1"

**if [** -f $1 **];then echo** "$1 es fichero" valido=$1

**else**

**read** -p "Introduzca fichero " valido

**while [** ! -f $valido **];do**

**read** -p "Introduzca fichero " valido

**done fi**

**}**

#PROGRAMA PRINCIPAL

**if [** ! -d hogar **];then mkdir** hogar**;**

**if [** ! -f hogar/nejecs **];then touch** hogar/nejecs

**fi**

**fi**

# variable de entorno USERDEST="realiza\_root" que se ha de crear en el shell-script realiza

#comprobación de existencia de usuario realiza\_root (se deduce que se crea usuario si no existe)

usu=`grep -w "realiza\_root" **/**etc**/**passwd` **if [** "$usu" = "" **];then**

**sudo** adduser realiza\_root

**fi**

USERDEST="realiza\_root" **;export** USERDEST

#comprobación del número de parámetros y del fichero #se asume que siempre se introduce algún fichero **case** $# **in**

1. **echo** "Ha introducido 1 parámetro"**;** fichero $1

visualizacion $valido

**;;**

1. **echo** "Ha introducido 2 parámetros" fichero $2

opciones $1 $valido

**;;**

\*)**echo** "Sólo se permite 1 o 2 parámetros"**;break;; esac**

#En el directorio hogar, tendremos un fichero denominado nejecs que contendrá el n° de veces que se ha ejecutado realiza(es el script principal aquí examen71.sh)

**echo** "Se ejecuta el script examen71.sh" **>>** hogar/nejecs

**echo** "El script examen71.sh se ha ejecutado " `cat hogar**/**nejecs **|** wc -l` "veces"

#!/bin/bash

#AUTOMATIZACIÓN DE COPIAS Y RESTAURACIÓN DE SEGURIDAD DE CUENTAS DE USUARIOS >1000

#ADMITE 2 PARÁMETROS acción directorio

#acción podrá ser -c (crea copia de seguridad en directorio), -r (restaura copia de seguridad en directorio)

#la copia de seguridad consistirá en un fichero llamado usuario.tgz comprimido y empaquetado del directorio home de cada usuario del sistema con id >1000

#También se generará el fichero usuarios que tendrá la siguiente estructura

#usuario;nombre\_completo;clave encriptada;directorio home;shell

#-c generará el ficheros usuarios con la estructura anterior. Localizará el directotorio de trabajo de cada usuario y generará los ficheros xxxx.tgz

#-r restaurará la copia de seguridad que se encuentra en el directorio y dejará el sistema tal como se encontraba en el momento de realizar la copia

#el script verificará los posibles errores:permisos, no directorio, parámetros #FUNCIONES

fichero()**{**

#crear archivo usuarios\_1 con id >1000(1000 es usuario ana no quiero afecte copia) #usuario;nombre\_completo;clave encriptada;directorio home;shell

**for** i **in** `ls **/**home`**;do** #problemas si cogemos los usuarios de /etc/passwd nusu=`id -u $i`

**if [** $nusu -gt 1000 **];then**

usu=`grep $i **/**etc**/**passwd **|** cut -d: -f1`

nom=`grep $i **/**etc**/**passwd**|** cut -d: -f5 **|** cut -d',' -f1`

#para clave encripatada en /etc/shadow en /etc/passwd aparece x clave=`cat **/**etc**/**shadow **|** grep $i **|** cut -d: -f2`

dih=`grep $i **/**etc**/**passwd **|** cut -d: -f6` inter=`grep $i **/**etc**/**passwd **|** cut -d: -f7`

**echo** "$usu;$nom;$clave;$dih;$inter" **>>** usuarios\_1

**fi done**

**}**

copia()**{**

**echo** "Se procede a realizar la copia en $1" #crear copias y guardar en $1

**for** i **in** `ls **/**home`**;do**

nusu=`id -u $i`

**if [** $nusu -gt 1000 **];then**

#-P para evitar mensaje eliminando / inicial de los nombres #1>/dev/null para evitar mensajes en pantalla

**tar** -czvf $i.tgz /home/$i -P **1>**/dev/null **mv** $i.tgz $1

**fi done**

**}**

restaura()**{**

**echo** "Se procede a realizar la restauración"

**for** i **in** `ls $1`**;do** usu=`echo $i **|** cut -d. -f1`

#ha que poner $1 porque si no no lo encuentra

**tar** -xzvf $1/$i /home/$usu -P **1>**/dev/null **done**

**}**

#PROGRAMA PRINCIPAL

#comprobaciones de parámetros y permisos usu=`id -u`

**if [** $# -eq 2 **];then**

**if [** $usu -eq 0 **];then if [** -d $2 **];then**

**case** $1 **in**

"-c")

fichero copia $2**;;**

"-r")restaura $2**;;**

\*)**echo** "Opción $1 no válida"**;exit;; esac**

**else**

**echo** "$2 no es directorio"

**fi else**

**echo** "El usuario no es root y no puede ejecutar este programa"

**fi else**

**echo** "Debe introducir 2 parámetros"

**fi**

#MANDAR AL CRON ¡¡perfecto!!

**echo** "30 15 6 1 \* /home/ana/Documentos/$0 -c /home/ana/Documentos/carprueba" **>>** /etc/‐ crontab

**echo** "40 15 6 1 \* /home/ana/Documentos/$0 -r /home/ana/Documentos/carprueba" **>>** /etc/‐ crontab

#!/bin/bash ¡¡¡PERFECTO!!!

#Ejecutado Por Root, Recibe Un Usuario Registrado En Sistema,

#Crea En Directorio De Trabajo del usuario recibido Un Directorio Con Nombre CopiaSeguridad,

#El directorio deberá pertenecer al usuario y a su grupo.

#Solo el propietario tendrá permiso de lectura, escritura y ejecución.

#En el directorio se deberán copiar todos los ficheros y directorios que haya en el directorio de trabajo del usuario dentro de un directorio con el nombre AAMMDDMMSS

#función copia copia ()**{**

#comprobación de directorio de trabajo del usuario recibido di=`ls **/**home **|** grep -w "$1"`

**if [** "$di" != "" **];then**

#creación de directorio CopiaSeguridad

**if [** ! -d /home/$1/CopiaSeguridad **];then mkdir** /home/$1/CopiaSeguridad

**fi**

**chown** $1:$1 /home/$1/CopiaSeguridad

**chmod** 700 /home/$1/CopiaSeguridad

#creación de directorio AAMMDDHHMMSS nombre=`date +%y%m%d-%H%M%S`

**if [** ! -d /home/$1/CopiaSeguridad/$nombre **];then mkdir** /home/$1/CopiaSeguridad/$nombre

**fi**

**echo** "realización de copia de /home/$1"

# no se puede copiar sobre sí mismo(primero cp en directorio de trabajo actual que es root y luego mv)

#otra posibilidad es con tar

**cp** -r -a /home/$1 $nombre

**mv** $nombre /home/$1/CopiaSeguridad

**else**

**echo** "El usuario $1 no dispone de directorio de trabajo"

**fi**

**}**

#PROGRAMA PRINCIPAL

#Comprobación de root y del parámetro usu=`id -u`

**if [** $usu -eq 0 **];then if [** $# -eq 1 **];then**

dato=`grep -w "$1" **/**etc**/**passwd` **if [** "$dato" != "" **];then**

**echo** "Se procede a realizar la copia de seguridad del usuario $1" copia $1

**else**

**echo** "El usuario $1 no está en el sistema"

**fi else**

**echo** "Debe introducir 1 parámetro"

**fi else**

**echo** "El programa debe ser ejecutado por root"

**fi**

#!/bin/bash

#PARÁMETROS DIVERSOS Y CAMBIAR NOMBRES A MAYÚSCULAS, RECURSIVIDAD(LLAMANDO AL PROPIO SCRIPT CON **TODO** EL CONTENIDO DEL DIRECTORIO)

#Cambiar a mayúsculas los nombres de los archivos o directorios pasados por parámetros

#se puede pasar -r directorio cambio recursivo -d directorio para cambiar en el directorio indicado

cambio()**{**

ruta=`sudo find . -name $1` **2>/**dev**/**null **if [** "$ruta" != "" **];then**

di=`dirname $ruta`

nom1=`echo "$1" **|** tr [:lower:] [:upper:]`

**mv** $ruta $di/$nom1

**fi**

**}**

#PROGRAMA PRINCIPAL

#comprobación de parámetros

**if [** $# -ne 0 **];then for** i **in** $\***;do**

**case** "$i" **in**

"-r")**shift;**recursivo=1**;continue;;** #necesario continue "-d")**shift;**recursivo=0**;continue;;**

\*)

**if [** -f $i **];then**

cambio $i

**fi**

**if [** -d $i **] && [** $recursivo -eq 1 **];then cd** $i

**for** j **in** `ls -R`**;do** cambio $j

**done cd** ..

cambio $i

**fi**

**if [** -d $i **] && [** $recursivo -eq 0 **];then**

cambio $i

**fi**

**;;**

**esac done**

**else**

**echo** "No ha indicado ningún archivo o directorio"

**fi**

#!/bin/bash #¡¡¡PERFECTO!!!

#se parte de un fichero (pasado por parámetro) que contiene lista de equipos y dirección ip (cada línea tiene formato host:ip

#averiguar qué equipos hacen ping

#se reciben los parámetros -i -h ip o host. Si no se indican los parámetros se solicitará en el script el método para hacer ping

#comprobación de conexión conexion()**{**

**echo** "Se procede a comprobar la conexión del fichero $1 con la opción $2"

**for** i **in** `cat $1`**;do**

**if [** "$2" = "-h" **];then**

tipo=`echo $i **|** cut -d: -f1`

**elif [** "$2" = "-i" **];then**

tipo=`echo $i **|** cut -d: -f2`

**fi**

#2>/dev/null para las ip o maq que no encuentre

con=`ping -c1 $tipo **|** grep -w "0% packet loss"` **2>/**dev**/**null **if [** "$con" != "" **];then**

**echo** "La máquina $tipo responde"

**else**

**echo** "La máquina $tipo no responde"

**fi done**

**}**

comprobacion\_fichero()**{ if [** ! -f $1 **];then**

**echo** "$1 no es fichero y no se puede ejecutar el programa"**;exit fi**

**}**

comprobacion\_opcion()**{**

**if [** "$1" = "-i" -o "$1" = "-h" **];then**

op=$1

**else**

**echo** "$1 no es válida"**;exit fi**

**}**

#comprobación de parámetros

**case** $# **in**

0)

**echo** "Debe introducir como mínimo el archivo"**;;** 1)

comprobacion\_fichero $1

**read** -p "Introduce opción -i o -h " resp**;** comprobacion\_opcion $resp

conexion $1 $op

**;;**

2)

comprobacion\_fichero $1 comprobacion\_opcion $2 conexion $1 $2

**;;**

\*)**echo** "Debe introducir 1 o 2 parámetros"**;; esac**

#!/bin/bash

#configuración de red estática usando netplan para ubuntu20.04

#copia del fichero inicial (ya realizado fuera del script con cp sudo cp /etc/netplan/01- network-manager-all.yaml /etc/netplan/01-network-manager-all.yaml.bak

#CAMBIO ADAPTADOR PUENTE ¡¡¡¡PERFECTO CONECTA CON EQUIPOS Y A INTERNET!!!

validacion ()**{**

#comprobar formato ipv4 y valores ok=0

**for** i **in** $\***;do**

num1=`echo $i **|** tr -s . " " **|** wc -w`

**if [** $num1 -eq 4 **];then for** j **in** `seq 1 4`**;do**

num2=`echo $i **|** cut -d. -f$j`

**if [** $num2 -lt 0 -o $num2 -gt 255 **];then echo** $i "no es correcto y salimos"**;exit**

**fi done**

**else**

**echo** $i "no es correcto y salimos"**;exit fi**

**done**

**echo** "Comprobación de direcciones correcta"**;**ok=1

**}**

#PROGRAMA PRINCIPAL

#comprobación de root usu=`id -u`

**if [** $usu -ne 0 **];then**

**echo** "El usuario no es root y salimos "**;exit fi**

#comprobar la interfaz de red a utilizar

**echo** "Elige una interfaz de red de las que aparecen a continuación " ip -0 -o address **| cut** -d: -f2 # 0 para interfaces -o solo una línea #obligo a elegir una

nic=""

**while [** "$nic" = "" **];do**

**read** -p "Debes elegir nic " nic

**done**

#petición de datos ip , máscara, puerta de enlace y dns

**read** -p "Introduce ip " direcc

**read** -p "Introduce máscara de subres en formato /nº " mas

**read** -p "Introduce puerta de enlace " enla **read** -p "Introduce dns principal " ser\_dns1 **read** -p "Introduce dns secundario " ser\_dns2 #validación de datos

validacion $direcc $enla $ser\_dns1 $ser\_dns2 #validación de máscara

**if [** $mas -lt 8 -o $mas -gt 30 **];then**

**echo** "$mas no es una máscara correcta y salimos"**;exit**

**fi**

**if [** $ok -eq 1 **];then**

#modificación de archivo /etc/netplan/01-network-manager-all.yaml #cuidado con las tabulaciones

**echo** "

ethernets:

$nic:

dhcp4: no

addresses: [$direcc/$mas] gateway4: $enla nameservers:

search: [local]

addresses: [$ser\_dns1,$ser\_dns2] "**>>** /etc/netplan/01-network-manager-all.yaml #actualización de netplan:

**sudo** netplan apply

**fi**

#comprobación de modificación de ip con : ip address

**echo** "Comprobación de cambio"**;** ip address

#comprobar conectividad con máquina real: ping -c1 ip