PRÁCTICA 40

Repaso BASH
SOLUCIÓN PROPUESTA

1. Implementar un Shell-script que reciba dos números por parámetro. El programa devolverá todos los números enteros del intervalo formado por los dos números. El programa debe comprobar que se pasan sólo 2 números y que el segundo es mayor que el primero, en caso contrario mostrar mensaje de error.

```
#!\bin\bash
if [ $# -eq 2 ]
then
    if [ $1 -lt $2 ]
  then
    cont=$1
    while  [ $cont -le $2 ]
    do
      echo -n $cont
              if [ $cont -lt $2 ]
      then
              echo -n ","
      fi
          cont=`expr $cont + 1`
        done
    echo ""
  else
    echo "El segundo parámetro tiene que ser mayor que el primero"
  fi
else
  echo "Este Script espera 2 parametros enteros"
fi
```

2. Implementar un programa que dado un nombre de grupo determine si existe en el sistema, y si es así, presente su nombre, número de grupo (GID), y lista de usuarios que pertenecen a él.

```
#!/bin/bash
if [ $# -lt 1 ]
then
  echo "ERROR: Introduce nombre grupo"
  exit 1
fi

grep -E "^$1" /etc/group > /dev/null

if [ $? -eq 0 ]
then
  grupo=`cat /etc/group | grep -E "^$1"  | cut -f1 -d":"`
  gid=`cat /etc/group | grep -E "^$1"  | cut -f3 -d":"`
  usuarios=`cat /etc/group | grep -E "^$1"  | cut -f4 -d":"`
  echo $grupo $gid $usuarios
else
  echo "El grupo $1 no existe"
fi
```

3. Modifica el programa anterior haciendo que si el segundo parámetro es más pequeño que el primero, los intercambie dentro del script. Tendremos que comprobar que los números son distintos.

```
#!\bin\bash
if [ $# -eq 2 ]
then
  if [ $1 -ne $2 ]
  then
    if [ $1 -lt $2 ]
    then
      men=$1
      may=$2
    elif [ $1 -gt $2 ]
    then
      men=$2
      may=$1
    fi
    while  [ $men -le $may ]
    do
      echo -n $men
      if [ $men -lt $may ]
      then
        echo -n ","
      fi
      men=`expr $men + 1`
    done
    echo ""
  else
    echo "Los dos parámetros no pueden ser iguales"
  fi
else
  echo "Este Script espera 2 parametros enteros"
fi
```

4. Implementa un script que haga un empaquetado y compresión de uno o más directorios pasados como parámetros. El nombre del archivo de backup resultante será “backup” seguido de la fecha de creación y extensión “tgz”. Deberá comprobarse que los directorios existen y realizar el backup solo en el caso de que todos sean correctos.

```
#!/bin/bash
if test $# -lt 1
then
  echo "ERROR: falta nombre del directorio a comprimir"
  exit 1
fi

for dir in $*
do
  if test ! -d $dir
  then
    echo "Error: no existe $dir"
    exit 1
  fi
done

fecha=`date +%Y-%m-%d`

tar cfz backup$fecha.tgz $* 2> /dev/null
exit 0
```


Francisco Javier Cabrerizo (fjavier.cabrerizo@outlook.es)

5. Implementar un script que admita un número indeterminado de parámetros. Si el parámetro es un fichero y ocupa más de 50 bytes se borrará. Si es un directorio se pedirá confirmación antes de borrarlo. En cualquier otro caso se mostrará un mensaje de error “no se procesa el argumento: <argumento>”. Sólo se procesará el directorio de trabajo.

```
#/bin/bash

clear
if [ $# -ne 0 ]
then
  for i in $*
  do
    if [ -d $i ]
    then
      echo "Esta seguro de borrar el directorio $i (s o n) "
      read seguro
      if [ $seguro = "s" ]
      then
        rmdir $i
        echo "directorio $i borrado"
      fi
    elif [ -f $i ]
    then
      if [ -s $i > 50 ]
      then
        rm $i
        echo "fichero $i borrado"
      fi
    else
      echo "el argumento $i no se procesa"
    fi
  done
else
  echo "El script necesita al menos un parámetro"
fi
```

6. Implementar un programa que liste los nombres de login, el directorio propio del usuario y el intérprete invocado por defecto de todos los usuarios , ordenados alfabéticamente por nombre de login

```
#!/bin/bash
cat /etc/passwd | cut -f1,6,7 -d":" | sort
```

7. Diseña un script que reciba N argumentos correspondientes a nombres de fichero y/o directorios. Para cada uno de los argumentos comprobar si existe y si es un fichero o un directorio. Si es un fichero indicara también que tipo de fichero; en el caso de que el argumento se trate de un directorio indicará cuanto ocupa en total.

```
#!/bin/bash
for arg in $*
do
  if test ! -e $arg
  then
    echo "$arg no existe, no es ni un fichero ni un directorio."
  else
    if test -f $arg
    then
      tipo=`file $arg`
      echo "$arg es un fichero de tipo $tipo"
    fi

    if test -d $arg
    then
      tam=`du -sh $arg`
      echo "$arg es un directorio de tamaño $tam"
    fi
  fi
done
```


Francisco Javier Cabrerizo (fjavier.cabrerizo@outlook.es)

8. Diseña un Shell-script que muestre un menú que se repita indefinidamente en pantalla con 5 opciones:

1. Usuarios conectados al sistema.
2. Procesos que ejecuta un usuario
3. Uso de memoria que está haciendo el sistema.
4. Uso del espacio de disco.
5. Salir del menú.

```
#/bin/bash
while [ 0 -eq 0 ]
do
  clear
  echo "1. Usuarios conectados al sistema"
  echo "2. Procesos que ejecuta un usuario"
  echo "3. Uso de memoria"
  echo "4. Uso del espacio en disco"
  echo "5. Salir"
  read -p "Introduzca una opcion: " opcion

  case  $opcion  in
    "1")
      who
      read -p "Pulse INTRO para continuar"
      ;;
    "2")
      read -p "Introduzca login: " login
      ps aux | grep $login
      ;;
    "3")
      free
      read -p "Pulse INTRO para continuar"
      ;;
    "4")
      df
      read -p "Pulse INTRO para continuar"
      ;;
    "5")
      exit 0
  esac
done
```

9. Escribe un script llamado comando.sh que reciba al menos un argumento: una opción (letra c, r, v y h). Funcionamiento del script:
   - Si se introduce la opción “c” se copiarán los ficheros en el directorio. El segundo argumento tiene que ser obligatoriamente un nombre de directorio y los demás uno o varios nombres de fichero (necesita como mínimo 3 argumentos). *Ejemplo: ./ej09.sh c dir1 fich1 fich2 … -> Copia los ficheros fich1, fich2, … dentro de dir1*
   - Si se introduce la opción “r” se borraran los ficheros que se pasen como argumento (necesita como mínimo 2 argumentos). *Ejemplo: ./ej09.sh r fich1 fich2 fich3 fich4 …*
   - Si se introduce la opción “v” se visualizarán de forma paginada el contenido de los ficheros que se pase como argumento (necesita como mínimo 2 argumentos). *Ejemplo: ./ej09.sh v fich1 …*
   - Si se introduce la opcion “h” se mostrará una pequeña ayuda indicando cómo se usa el script. No necesita que existan más argumentos.

```
#!/bin/bash

if [ $# -lt 2 ] && [ "$1" != "h" ]
then
  echo "ERROR: Número de argumentos incorrecto"
  echo "USO: comando.sh [c|r|v|h] arg1 arg2 ..."
  exit 1
fi

if [ "$1" == "c" ]
then
  if [ $# -lt 3 ]
  then
    echo "ERROR: falta nombre de directorio y al menos un nombre de fichero"
    echo "USO: comando.sh c directorio fichero1 ..."
    exit 1
  fi

  if [ ! -d $2 ]
  then
    echo "ERROR: El segundo argumento deber ser un nombre de directorio"
    echo "USO: comando.sh c directorio fichero1 ..."
    exit 1
  fi
  shift

  dir=$1
  shift
  for fich in $*
  do
    if [ -f $fich ]
    then
      cp $fich $dir
    fi
  done
  exit 0
fi
```


Francisco Javier Cabrerizo (fjavier.cabrerizo@outlook.es)

```
if [ "$1" == "r" ]
then
  for fich in $*
  do
    if [ -f $fich ]
    then
      rm $fich
    fi
  done
  exit 0
fi

if [ "$1" == "v" ]
then
  for fich in $*
  do
    if [ -f $fich ]
    then
      cat $fich | more
    fi
  done
  exit 0
fi

if [ "$1" == "h" ]
then
  echo "ERROR: Número de argumentos incorrecto"
  echo "USO: comando.sh [c|r|v|h] arg1 arg2 ..."
  echo "donde c copia en el directorio 'arg1' los ficheros 'arg2 ...'"
  echo "  r borra los ficheros 'arg1 arg2 ...'"
  echo "  v visualiza los ficheros"
  echo "  h muestra ayuda"
  exit 1
fi
```