Sistemas Operativos
UTN-FRLP
Scripts

Alumnas:
Maria Sol Bruschini - Godoy Agustina

1. Crear un script que imprima “Estoy aprendiendo lenguaje scripting”

```
#!/bin/bash
  Echo “Estoy aprendiendo lenguaje scripting”
```

2. Hacer un script que realice los siguientes pasos:

- Copiar el archivo fstab como dispositivos
- Limpiar la pantalla
- Del archivo dispositivos listar la información del CDROM.
- Crear un directorio tp1. Comprimir y empaquetar este directorio.

```
#!/bin/bash
   Cp /etc/fstab dispositivos
   Clear
   Grep “CDROM” dispositivos
    tar -czvf  $HOME/practicas/tp2/tp1.tar.gz $HOME/ practicandoScripts /ParteB
```

3. Realizar un script que evalúe 2 archivos pasados como argumentos indicando adecuadamente entre ambos cual posee más cantidad de líneas.

```
#!/bin/bash

   if [  ! -f $1 ]; then
     echo ‘tiene que ser un archivo’
     exit
   fi
   if [ ! -f $2 ]; then
```

```
    echo 'tiene que ser un archivo'
    exit
  fi
  a=$( cat $1 | wc -l )
  b=$( cat $2 | wc -l)
  if [ $a -eq $b ]; then
    echo 'los archivos tienen la misma cantidad de líneas'
  else
    if [ $a -gt $b ]; then
       echo 'el primer archivo tiene mas líneas'
    else
       echo 'el segundo archivo tiene mas líneas'
    fi
  fi
  echo 'la cantidad del primer archivo es $a'
  echo 'la cantidad del segundo archivo es $b'
```

4. Hacer un script que indique la cantidad de archivos y directorios que hay en un directorio cualquiera pasado como argumento, usando estructuras repetitivas.

```
#!/bin/bash
If [ -d $1 ]; then
    cont_D=0
    cont_F=0
    for elemento in $(ls -a $1); do
        if [ -d $elemento ]; then
           cont_D=$(($cont_D + 1))
        else
           if [ -f $elemento ]; then
              cont_F=$(($cont_F + 1))
           fi
        fi
         Done
            echo 'la cantidad de directorios es $cont_D'
            echo 'la cantidad de archivos es $cont_F'
 else
     echo 'no existe el directorio'
 fi
```

5. Hacer un script que realice los siguientes pasos:

- Descomprimir el directorio prácticas en un directorio llamado nuevo
- Del archivo cmd.txt sacar las líneas que corresponden al comando ls y guardarlas en el archivo listado
- Comprimir el archivo listado.

```
#!/bin/bash
    Mkdir $HOME/practicas/tp2/nuevo
    Cd nuevo
    tar -xzvf $HOME/practicas.tar.gz
    touch listado.txt
    cat $HOME/practicas/tp1/ hist.txt | grep ls >listado.txt
    tar -czvf listar.tar.gz listado.txt
```

6. Hacer un script que realice los siguientes pasos: - Copiar el archivo solo-archivos.txt de la práctica anterior como solo-archivo. - Cambiar los permisos del archivo solo-archivo para que pueda ser modificado por otros - Adicionar al archivo anterior el contenido del archivo perfil.txt.

```
#!/bin/bash
 Cp -r $HOME/practicas/tp1/solo-archivos.txt $HOME/practicas/tp2/casa/solo-archivo
 Chmod 646 solo-archivo
 Cp -r $HOME/practicas/tp1/perfil.txt solo-archivo
```

7. Implementar con un script el punto 6 de la parte B de la practica 1

```
#!/bin/bash
 Cat /etc/passwd | grep agodoy | cut -d":" -f 1,3,4,7 >> $HOME/practicas/tp2/perfil.txt
```

8. Crear un archivo de nombre informa en el directorio tp2 donde se describa lo siguiente.
   - Cantidad de usuarios conectados.
   - Cantidad de puertos abiertos.
   - Guardar los procesos que se están ejecutando en el sistema.
   - Comprimir el archivo informa

```
#!/bin/bash
    touch informa
    echo "cantidad de usuarios conectados" >>informa
    who wc -l >> informa
    echo "cantidad de puertos abierto" >> informa
    echo "procesos que se están ejecutando en el sistema" >>informa
    Ps >>informa
    cat informa
    tar -czvf $HOME/practicas/tp2/casa/informa.tar.gz
```

9. Implementar un script usando la sentencia while, en donde se lea el archivo passwd línea por línea y se imprima el mismo por Terminal.

```
#!/bin/bash
   archivo=/etc/passwd
   while read l
    do
      echo $line
   done <$archivo
```

10. Tomar el archivo passwd copiado en el tp1 y utilizarlo en un script que permita simular dar de alta un usuario en términos de: Info01:x:uid:guid: :/home/info01:/bin/bash El usuario info01 debe ser pasado como argumento.

```
#!/bin/bash
    id=$(tail -l $HOME/practicas/tp1/passws | cut -d" " f1)
    ci=$(tail -l $HOME/practicas/tp1/passwd | cut -d" " f1)
    let id=$id+1
    let ci=$ci+1
    echo $1:x:$id:$ci:JuanPerez >>$HOME/practicas/tp1/passwd
```

11. Implementar un script que cambie los permisos de todos los ejercicios (scripts) de la practica 2 que se encuentran en #HOME/practicas/tp2/ para que puedan ser ejecutados por otros y modificados por el grupo.

```
#!/bin/bash
    for i in $(ls $HOME/practicas/tp2/*); do
      if [ -f $i ]; then
        if [ -x $i ]; then
          chmod 765 $i
          echo $i
        fi
      fi
    done
```

12. Guardar solo los nombres de aquellos archivos que tengan permisos igual a 644 pertenecientes al directorio raíz de su perfil de usuario

```
#!/bin/bash
    for i in $(find $HOME/practicas/tp2/* -maxdepth 1 -type f -perm 644); do
        echo $i >> solo-archivos
    done
```

13. Usando el comando find, generar un scrip que al pasarle un directorio cualquiera pasado como parámetro:

a) Guarde en el archivo name los nombres de archivos que comienzan con la letra a.

b) Guarde en el archivo extend los nombres de archivos que tienen extensión .txt c) Guarde en el archivo perm los nombre de archivos que tienen permisos iguales a 644

d) Guarde en el archivo tam los nombres de archivos que tienen tamaño mayor a 5K

```
#!/bin/bash
    if [ -d $1 ];then
       for i in $(find $1/* -maxdepth 1 -type f -name “a*”); do
          echo $1 >> name
       done
       for i in $(find $1/* -maxdepth 1 -type f -name “*.txt”); do
          echo $1 >>  extend
       done
       for i in $(find $1/* -maxdepth 1 -type f -perm 644); do
          echo $1 >> perm
       done
       for i in $(find $1/* -maxdepth 1 -type f -size 5k); do
```

```
            echo $1 >> tam
        done
    else
        echo “tiene que pasar un directorio como parámetro”
    fi
```

14. Usando una estructura repetitiva recorrer un directorio cualquiera pasado por parámetro y determinar que archivos fueron modificados en un mes determinado. Dicho mes también pasarlo por parámetro.

```
#!/bin/bash
    for i in $1/*; do
        c=$(stat –format %y $1 | cut -d- -f2)
        if [ $c -eq $2 ];then
            echo “el archivo $i fue modificado el mes $2”
        else
            echo “el archivo $i no fue modificado el mes $2”
        fi
    done
```

15. Hacer un informe de un directorio cualquiera pasado por parámetro que indique:
a) Que archivos han sido modificados en los últimos 30 minutos
b) Que archivos han sido accedidos en los últimos 60 minutos.
c) Que archivos han sido modificados en los últimos 5 dias
d) Que archivos han sido modificados en los últimos 10 dias

```
#!/bin/bash
    echo >> $HOME/practicas/tp3/casa/informe
    echo “Archivos modificados los últimos 30 minutos”>>$HOME/practicas/tp3
        /casa /informe
    for i in $(find $1/* -maxdepth 1 -type f -mmin -31); do
        echo $i >> $HOME/practicas/tp3/casa/informe
    done
    echo >> $HOME/practicas/tp3/casa/informe
    echo “Archivos modificados los últimos 60 minutos”>>$HOME/practicas/tp3
        /casa/informe
    for i in $(find $1/* -maxdepth 1 -type f -amin -61); do
        echo $i >> $HOME/practicas/tp3/casa/informe
    done
    echo >> $HOME/practicas/tp3/casa/informe
```

```
echo “Archivos modificados los últimos 5 dias”>>$HOME/practicas/tp3
        /casa/informe
for i in $(find $1/* -maxdepth 1 -type f -mtime  -5); do
    echo $i >> $HOME/practicas/tp3/casa/informe
done
echo >> $HOME/practicas/tp3/casa/informe
echo “Archivos modificados los últimos 10 dias” >>$HOME/practicas/tp3
        /casa/informe
for i in $(find $1/* -maxdepth 1 -type f -mtime  -10); do
    echo $i >> $HOME/practicas/tp3/casa/informe
done
```

16. Hacer un script que luego de ejecutarse pida al usuario 2 números y después presente la suma, resta, producto y división de los mismos

```
#!/bin/bash
    echo “Ingrese el primer numero a operar”
    read a
    echo “ingrese el segundo numero a operar”
    read b
    resultado =$(expr $a + $b)
    echo “$a + $b = $resultado”
    resultado =$(expr $a - $b)
    echo “$a - $b = $resultado”
    resultado =$(expr $a \* $b)
    echo “$a * $b = $resultado”
    resultado =$(expr $a / $b)
    echo “$a / $b = $resultado”
```

17. Compruebe si un directorio cualquiera pasado como argumento existe, en tal caso contabilizar la cantidad de archivos y directorios, guardar ambos contadores en un archivo. Usar una estructura repetitiva para resolverlo.

```
#!/bin/bash
    if [ ! -d $1 ];then
      echo “el directorio no existe”
    else
      for i in $1/* ;do
        if [ -f $i ];then
          Let ar=$ar+1
        else
          if [ -d $i ];then
            Let di=$di+1
```

```
            fi
          fi
        done
        echo “cantidad de directorios de $1” >> contadores
        echo $di >> contadores
        echo “cantidad de archivos de $1” >> contadores
        echo $ar >> contadores
      fi
```

18. Muestre los números naturales del 1-20

```
#!/bin/bash
    num=0
    while [ $num -lt 20 ]; do
        let num=$num+1
        echo $num
    done
```

19. Hacer un script que visualice un menú de tres opciones, la primera borra un fichero cualquiera, la segunda visualiza un fichero, la tercera copia un archivo al directorio actual y la cuarta sale del script.

```
#!/bin/bash
   op=0
   While [ $op -ne 4 ];do
       echo ---------- Menu de Opciones----------
       echo 1. Borrar archivo
       echo 2. Visualizar archivo
       echo 3. Copiar archivo
       echo 4. Salir del programa
       read op
       if [ $op -eq 1  ]; then
           echo “el archive $1 fue borrado”
           rm $1
exit
       else
           if [  $op -eq 2 ]; then
```

```
            cat $1
        else
          if [ $op -eq 3  ];then
          cp $1 $HOME/practicas/tp3/casa
           else
              if [ $op -eq 4 ];then
                      Echo “usted ha salido del programita”
                      Exit
              else
                      echo “Esa opción es incorrecta”
              fi
          fi
        fi
    done
```

20. Hacer un script que visualice un menú de tres opciones, la primera borra un fichero cualquiera, la segunda visualiza un fichero, la tercera copia un archivo al directorio actual y la cuarta sale del script.

```
#!/bin/bash
   if [¡ -f $1 ];then
      echo “El archive no existe”
   else
        while read L; do
          echo $L | tr [[:lower:]] [[:upper:]]
        done < $1
   fi
```

21. Hacer un script que pida continuamente una palabra clave, si la palabra introducida es “secreto” que nos muestre un mensaje de Bienvenida.

```
#!/bin/bash
    op=0
```

```
while [ $op -ne 1 ]; do
        echo “ingrese la palabra clave ”
        read op
        if [ $op = “secreto” ];then
           echo “Bienvenido, persona de buen corazon”
           op=1
        else
           echo “Estas intentando ingresar a algo a lo cuando no tenes acceso,
                    por favor vete de aquí”
        fi
done
```

22. Realizar un script que permita copiar un archivo pasado como parámetro en un directorio cualquiera también pasado como parámetro, antes de copiar comprobar que el archivo y directorio existan y que el directorio este vacío

```
#!/bin/bash
    if [ -d $2  ];then
      if [ -f   $1 ];then
        c=$(ls $2 | wc -l)
        if [  $c -eq “0” ];then
            cp $1 $2
            echo “El archive se ha copiado al directorio $2”
        else
            echo “el archivo no está vacio”
        fi
      else
        echo “El primer parámetro tiene que ser un archivo”
      fi
```

```
else
  echo “El segundo parámetro tiene que ser un directorio”
fi
```

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

```
#!/bin/bash
    if [ ! $1 ]; then
        echo "El script solicita 1 parámetro"
        exit
    fi
    if [ ! -d $1 ]; then
         echo "El parámetro debe ser un directorio"
         exit
    fi
    for i in $1/*; do
         if [ -f $i ]; then
            TAM=$(du -5 $i | cut -d "/" -f1)
            echo $TAM $i >> archivos
         fi
    done
    sort -nr archivos | head - n1
```

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

```
#!/bin/bash
    if [ ! $1 ]; then
        echo "El script solicita i parámetro"
        exit
    fi
    if [ ! -d $1 ]; then
        echo "El parámetro 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
```

25. Dado un archivo con 10 palabras desordenadas (una palabra por línea) enviado como parámetro, imprimir por pantalla el top 3 de las palabras con más apariciones, ordenado de forma descendiente. Validar que no falte el parámetro.

```
#!/bin/bash
    if [ ! $1 ]; then
        echo "El script solicita 1 parámetro"
        exit
    fi
    if [ ! -f $1 ]; then
        echo "El parámetro debe ser un archivo"
        exit
    fi
    sort $1 | uniq -c | sort -nr | head -n3
```

26. Dado un nombre de usuario recibido por parámetro:

1) Validar que no falte enviar el parámetro
2) Decir si ese usuario se encuentra actualmente conectado al sistema o no
3) Indicar la cantidad total de usuarios de la com15 conectados actualmente al sistema
4) Guardar la salida del script en un archivo llamado usuarios.txt

```
#!/bin/bash
    if [  ! $1 ]; then
        echo "El script solicita 1 parámetro"
        exit
    fi
    ESTA_CONECTADO=$(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 -1)
    echo "La cantidad de usuarios conectados es
                $CANTIDAD_CONECTADOS">> usuarios.txt
    cat usuarios.txt
```

27. Guardar en un fichero un listado de usuarios y la cantidad de procesos que está 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 $i | WC -1)
        let CANT=$CANT-1
        echo $CANT $NOM >> usuarios
    done
    sort -nr usuarios
```

28. 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 <nombreusuarios> y cuyo dueño sea el mismo usuario. Para cambiar dueño con chown se necesitan permisos especiales.

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

29. 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. Para acceder a los directorios de algunos usuarios se necesitan permisos especiales.

```
#!/bin/bash
    for i in $(cat /etc/passwd | cut -d ":" --1); do
        NOM=$i
        TAM=$(du -s /home/$i | cut -d "t" -f1)
        echo $TAM $NOM > usuarios
    done
    sort -nr usuarios
```

30. Hacer un script al cual se le pase como parámetro 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
```

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

```
#!/bin/bash
    if [  $# -eq 2  ]; then
      echo "Debe pasar dos parametros"
      exit
    fi
    if [  !  -f  $1  ]; then
      echo "El primer parámetro 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
```

32. Realizar un script que evalue 2 archivos pasados como argumentos indicando adecuadamente entre ambos cual posee más cantidad de caracteres. Dejar la información especificando el nombre del archivo y cantidad, en el archivo cant-caracteres. Comprobar que los parámetros no sean pasados en blanco.

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

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

```
#!/bin/bash
  NUM=$1 rm numeros
  for i in $(seq 6); do
      let NUM=$NUM+1
      echo $NUM >> numeros
  done
  sort -nr números
```

34. Realizar un script donde por parámetro 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 parámetro no se pase en blanco cuando se ejecuta el script. Comprimir el archivo word.

```
#!/bin/bash
    if [ !$# -eq 1 ]; then
      echo "Debe pasar un parámetro"
      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
```

35. 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

```
#!/bin/bash
    if [ ! $# -eq 1 ]; then
       echo "Debe pasar un usuario como parámetro"
       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
```

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

```
#!/bin/bash
    USUARIO=$1
    if [ $# -eq 0 ]; then
```

```
    ps aux
  else
    if [ "$USUARIO" = "root" ]; then
       echo "ERROR!!!"
    else
       ps aux | grep "$USUARIO"
    fi
  fi
```

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

```
#!/bin/bash
    if [ ! $# -eq 1 ]; then
          echo "Debe pasar un parámetro"
          exit
    fi
    if [ ! -f $1 ]; then
          echo "El parámetro debe ser un archivo"
          exit
    fi
   TAM=$(du $1 | cut -d " " -f1)
    echo "Tamaño de $1: $TAM"
```

38. 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. Además informar sobre el tamaño total en Mbyte d # todos los archivos encontrados

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

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

```
#!/bin/bash
    ARCHIVO=$1
    cat $ARCHIVO | while read linea; do
                              echo $línea >> copia
```

```
        echo $línea
        sleep 3
    done
```

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

```
#!/bin/bash
    if [ ! $# -eq 1 ]; then
        echo "Debe pasar un parámetro"
        exit
    fi
    if [ ! -d $1 ]; then
        echo "El parámetro 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
```

41. Hacer un script que arme un menú de 3 opciones:

1) Guarde una frase ingresada por teclado en un archivo
2) Muestre la frase ingresada en la opción 1 si es que existe
3) Salir del menú

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

42. 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 extensión .conf de un directorio cualquiera pasado como parámetro. A su vez, chequear que el parámetro exista y sea realmente un directorio.

```
#!/bin/bash
    if [ ! $# -eq 1 ]; then
        echo "Debe pasar un parámetro"
        exit
    fi
    if [ ! -d $1 ]; then
        echo "El parámetro debe ser un directorio"
        exit
    fi
    find $1 -type f -perm 755 -exec echo {} >> perm \;
    find $1 -type f -name "*. conf" -exec echo {} >> exten \;
```

43. 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 están ejecutando en el sistema. Además, si al script le paso como parámetro un usuario cualquiera me diga si está conectado o no.

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

44. Hacer un script que recorra un directorio cualquier pasado por parámetro y borre el archivo más grande en el primer nivel de este directorio. Además guardar como información el nombre y tamaño del archivo borrado en un archivo llamado archivo-borrado.

```
#!/bin/bash
    if [ ! $# -eq 1 ]; then
        echo "Debe pasar un parámetro"
        exit
    fi
    if [ ! -d $1 ]; then
        echo "El parámetro 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
```

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

```
#!/bin/bash
    if [ ! $# -eq 1 ]; then
        echo "Debe pasar un parámetro"
        exit
    fi
    if [ ! -d $1 ]; then
        echo "El parámetro debe ser un directorio"
        exit
    fi
    TAM=$(du -sb $1 | cut -d " " -F1)
    echo $TAM > archivo
    echo "El directorio $1 tiene un tam de $TAM bytes"
```

46. 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 $file --only-matching | wc -1)
        echo "$counter $file" >> /tmp/output
    done
    cat /tmp/output | sort -nr | head -n3
    rm /tmp/output
```

47. 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 válidos, y retornar el nombre del directorio que contiene mayor cantidad de archivos 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 -1)
         if [ $counter -gt $MAX ]; then
              MAX=$counter
              MAX_NAME=$dir
         fi
     done
     echo "$MAX_NAME contiene mayor cantidad de scripts: $MAX"
```

48. 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 líneas 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 $(who | cut -d" " --1 | sort -u); do
        datos=$(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
```

49. 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 más 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
```

50. 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 línea 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 parámetro"
        exit
    fi
    if [ ! -d $1 ]; then
        echo "El parámetro 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 'id; $d' directorio_nuevo/$ejecutable
    done
    copiados=$(15 directorio_nuevo | wc -1)
    echo "Fueron copiados $copiados archivos"
```

51. Crear cuatro nuevos directorios llamados dira, dirb, dirc, y dird bajo el directorio actual.

```
$mkdir dira dirb dirc dird
```

52. Comprobar los permisos de acceso de los directorios recién creados para comprobar el funcionamiento del comando umask.

```
$ls -l
```

53. Crear el fichero uno . Quitarle todos los permisos de lectura. Comprobarlo. Intentar borrar dicho fichero.

```
$touch uno
$chmod a-r uno
```

$ls -l
$rm uno

54. Quitarle todos los permisos de paso al directorio dir2 y otorgarle todos los demás.

$chmod = dir2
$chmod o=rwx dir2

55. Crear en el directorio propio:

- El directorio carpeta1 con los tres permisos para el propietario, dentro de él fich1 con lectura y escritura para todos y fich2 con lectura y escritura para el propietario y solo lectura para el resto.
- El directorio carpeta2 con todos los permisos para el propietario y lectura y ejecución para los del mismo grupo. Dentro file1 con lectura y escritura para el propietario y los del grupo y file2 con los mismos para el propietario y solo lectura para el grupo.

$mkdir carpeta1 carpeta2
$chmod u=rwx,g=,o= carpeta1
$chmod u=rwx,g=rx,o= carpeta
$ls -l
$touch carpeta1/{fich1,fich2}
$chmod = carpeta1/{fich1,fich2}
$chmod o=rw carpeta1/fich1
$ls -l carpeta1
$touch carpeta2/{file1,file2}
$chmod = carpeta2/{file1,file2}
$chmod u=rw,g=rw carpeta2/file1
$chmod u=rw,g=r carpeta2/file2
$ls -l carpeta2

56. Desde otro usuario probar todas las operaciones que se pueden hacer en los ficheros y directorios creados.

$su us3rlinux
Contraseña:
carpeta1
prueba de acceso
$us3rlinux@equipo1:/home/usuario1/PRUEBA$ cd carpeta1

$bash: cd: carpeta1: Permiso denegado
prueba de lectura
$us3rlinux@equipo1:/home/usuario1/PRUEBA$ ls carpeta1
$ls: no se puede abrir el directorio carpeta1: Permiso denegado
carpeta2
prueba de acceso
$us3rlinux@equipo1:/home/usuario1/PRUEBA$ cd carpeta2
prueba de lectura
$us3rlinux@equipo1:/home/usuario1/PRUEBA/carpeta2$ ls -l
$total 0
$-rw-rw---- 1 usuario1 usuario1 0 2009-12-08 09:41 file1
$-rw-r----- 1 usuario1 usuario1 0 2009-12-08 09:41 file2
prueba de lectura
$us3rlinux@equipo1:/home/usuario1/PRUEBA/carpeta2$ cat file1
$us3rlinux@equipo1:/home/usuario1/PRUEBA/carpeta2$ cat file2
prueba de escritura
$us3rlinux@equipo1:/home/usuario1/PRUEBA/carpeta2$ echo 'hola' > file1
$us3rlinux@equipo1:/home/usuario1/PRUEBA/carpeta2$ echo 'hola' > file2
$bash: file2: Permiso denegado
$exit
$us3rlinux@equipo1:/home/usuario1/PRUEBA$ whoami
$us3rlinux
$us3rlinux@equipo1:/home/usuario1/PRUEBA$ exit
$exit
$usuario1@equipo1:~/PRUEBA$ whoami
$usuario1
$usuario1@equipo1:~/PRUEBA$

57. Visualizar la trayectoria completa del directorio actual. Crear dos directorios llamados correo y fuentes debajo del directorio actual.

$pwd
$/home/usuario1/PRUEBA
$mkdir correo fuentes

58. Posicionarse en el directorio fuentes y crear los directorios dir1, dir2, dir3.

$cd fuentes
$mkdir dir1 dir2

59. Crear el directorio menus bajo correo sin moverse del directorio actual.
$mkdir ../correo/menus

60. Posicionarse en el directorio HOME. Borrar los directorios que cuelgan de fuentes que acaben en un número que no sea el 1.

$cd $HOME
$find PRUEBA/fuentes -type d -name "*1" -exec rm -r {} \;

61. Ver si existe el archivo tty2 en el directorio dev. En caso de que exista, ver su fecha de creación o actualización.

$find PRUEBA/fuentes/* -type d -regex ".*[0,2,3,4,5,6,7,8,9]" -exec rm -r {} \;
$find PRUEBA/fuentes/* -type d -regex ".*[^1]" -exec rm -r {} \;

62. Ver los permisos que tienen los archivos que empiecen por tt del directorio /dev.

$ls -l /dev/tt*

63. Visualizar la lista de los archivos ordinarios que están en el directorio /usr/bin.

$find /usr/bin -type f

64. Visualizar la lista de todos los directorios que cuelgan del raíz.

$ls /
$find / -maxdepth 1 -type d

65. Visualizar la lista de todos los ficheros que pertenezcan a root.

$find / -user root -type f

66. Visualizar la lista de todos los ficheros .h del directorio /usr/include.

$find /usr/include -type f -regex ".*.h"

67. Ejecutar todos los comandos que empiecen por ls del directorio /bin.

$ls /bin/ls*

68. Visualizar de qué tipo son todos y cada uno de ficheros de todo el árbol del sistema propiedad de un usuario conocido.

```
$find /home/us3rlinux -exec file --mime-type -0 '{}' \;
```

69. Crear el directorio uno en el directorio HOME con permiso de escritura y paso para el propietario, de lectura y paso para los usuarios de su mismo grupo y ningún permiso para el resto de usuarios.

```
$mkdir uno
$chmod u=rw,g=rw,o= uno
$ls -ld uno
```

70. Crear el directorio uno1 dentro del directorio creado en el ejercicio anterior con todos lo permisos para el usuario, ninguno para los usuarios del grupo y permiso de escritura para el resto de usuarios.

```
$chmod u=rwx,g=rwx,o= uno
$mkdir uno/uno1
$chmod u=rwx,g=,o=w uno/uno1
$ls -ld uno/uno1
```

71. Copiar todos los ficheros propiedad de un usuario conocido que acaben en un número en el directorio menus.

```
$find /home/usuario1 -type f -regex ".*[0-9]" -exec cp -r '{}'
PRUEBA/correo/menus/ \;
```

72. Crea un archivo de tamaño 0

```
$touch archivo_tamaño_cero
```

73. Visualiza el archivo /etc/motd, que contiene el "mensaje del día".

```
$cat /etc/motd
0 packages can be updated.
0 updates are security updates.
```

74. Utilizando de entrada la información de los usuarios conectados al sistema, guardar, ordenadas por el campo hora, las líneas correspondientes al usuario que se desee en el archivo persona.

```
#!/bin/bash
    who | grep $USER | sort -k 4 > persona
```

75. Crear el directorio carpeta debajo del directorio PRUEBA. Quitarle todos los permisos de lectura. A continuación, buscar todos los directorios que cuelguen del directorio propio y guardarlos en el archivo direc.

```
#!/bin/bash
    mkdir carpeta
    chmod a-r carpeta
    find ~ -type d > direc
```

76. Volver a realizar la segunda parte del ejercicio anterior, pero redireccionando los errores al fichero malos. Comprobar la información del fichero malos.

```
#!/bin/bash
    find ~ -type d 2> malo
```

77. Añadir al fichero direc la lista de todos los ficheros ordinarios que cuelguen de /etc.

```
#!/bin/bash
    find /etc -type f >> direc
```

78. Añadir al archivo nuevalista el/los nombre/s de el/los fichero/s del directorio PRUEBA que contengan en su nombre la cadena "ai", añadiendo el posible error al fichero malos.

```
#!/bin/bash
    find ./ -type f -not -iname *ai* 1> nuevalista 2> malos
    find ./ -type f -iname *ai* 1> nuevalista 2> malos
```

79. Sacar por pantalla únicamente el tiempo (buscar comando time) que tarda en ejecutarse el comando who.

```
$time `sleep 3`
$time who -p %e
```

80. Sacar por pantalla un listado completo (buscar comando ps) de los procesos que está realizando el usuario root.

```
$ps -U root -u root u
```

81. Crear el archivo proceso con los procesos que no tienen ninguna terminal asignado.

```
$ps -U root -u root u | grep -v "`ls /dev`"
```

82. Añadir al fichero anterior la fecha actual y la trayectoria completa del directorio actual.

```
#!/bin/bash
    echo "`date +"%A %D"` - `pwd`" >>nuevalista
```

83. Sacar por pantalla el listado de todos los usuarios conectados ordenados por número de proceso asignado.

```
$ps axu
```

84. Averiguar cuál es la actividad actual del sistema. Para ello visualice un listado completo del estado de todos los procesos que se están ejecutando en el sistema.

```
$top -d .1 -n 10
```

85. Mostrar cuantos usuarios tiene registrados el sistema (el registro de usuarios está en el archivo /etc/passwd)

```
$cat /etc/passwd | wc -l
```

86. Mostrar cuántos usuarios tiene registrados el sistema y que utilizan el intérprete bash (debe aparecer al final de la línea /bin/bash o similar)

$cat /etc/passwd | grep bash

87. Mostrar cuantos usuarios hay conectados

```
$who -q
```

88. Mostrar las líneas, de un archivo de texto, empiecen por L (mayúscula o minúscula)

```
#!/bin/bash
    man gcc > gcc.man_page
    cat gcc.man_page | sed -e 's/ //g' > file.filled
    cat file.filled | grep ^[Ll]
```

89. Contar las líneas, del ejemplo anterior

```
$cat file.filled | grep ^[Ll] | wc -l
```

90. Extraer los nombres de usuario (primer campo) del sistema

```
$cat /etc/passwd | cut -d ':' -f 1
```

91. Extraer los nombres de usuario y el shell que utilizan (último campo)

```
$gawk -F: '{print $1, $7}' /etc/passwd
```

92. Cambiar la fecha de creación de un archivo ya previamente creado

```
$touch -t 9910011101 good
$ls -l good
```

93. Calcular la firma md5 de un archivo

```
$md5sum good
```

94. Modificar la firma md5 y detectar que se ha cambiado (revisión de firma)

```
#!/bin/bash
    md5sum good > good.MD5
    echo hola >> good
    md5sum -c good.MD5
    md5sum good
```

95. Monitorear la ocupación de las particiones en los discos

```
$df -lh
```

96. ¿Cual es el proceso que más carga el procesador?

```
#!/bin/bash
  for x in `seq 1 10`; do
     ps -eo pid,pcpu,pmem,user,args | sort -r -k 2 | head -n 2;
     sleep 3;
  done
```

97. ¿Está corriendo el proceso bash?

```
#!/bin/bash
   ps -eo pid,pcpu,pmem,user,args | grep bash
   ps a | grep bash
```

98. ¿Cuántos procesos que empiecen por k están corriendo?

```
$ps -eo args | cut -d ' ' -f 1 | grep ^g | wc -l
```

99. Realizar un script llamado 'usuarioconectado' que retorna un SI si el primer parámetro coincide con algún usuario conectado o NO en caso contrario.

```
#!/bin/bash
    # si número de parámetros distinto 1
    if [ $# ne 1 ]; then
        echo "El número de parámetros debe de igual a 1"
        exit 1
    fi
    ESTA_CONECTADO= who | grep $1`
    if [ z "$ESTA_CONECTADO" ]; then
        echo "NO"
    else
        echo "SI"
    fi
```

100. Realizar un script llamado 'usuariosistema' que retorna un SI si el primer parámetro coincide con algún usuario del sistema o NO en caso contrario.

```
#!/bin/bash
    if [ $# ne 1 ]; then
      echo "El número de parámetros debe de igual a 1"
      ayuda
      exit 1
    fi
    ESTA_EN_SISTEMA=`grep E ^$1: /etc/passwd`
    if [ z "$ESTA_EN_SISTEMA" ]; then
      echo "NO"
    else
      echo "SI"
    fi
```

101. Realizar un script llamado 'suma' que realice la suma de 2 parámetros introducidos (tendrá que poder sumar números decimales, como 2.2 + 3).

```
#!/bin/bash
   function comprobarQueNoEsNumero() {
    if [ n "$1" \
          a "$1" != "0" \
          a "`echo $1 | awk '{ print $1*1 }'`" != "$1" ]; then
               echo "El parámetro '$1' no es un número"
               exit 2
    fi
    }

    if [ $# ne 2 ]; then
        echo "El número de parámetros debe de ser igual a 2"
        exit 1
    fi
    comprobarQueNoEsNumero $1
    comprobarQueNoEsNumero $2
    echo $1 $2 | awk '{ print $1 + $2 }'
```

102. Realizar un script llamado 'resta' que realice la resta de 2 parámetros introducidos (tendrá que poder sumar números decimales, como 2.2 – 3)

```
#!/bin/bash
        function comprobarQueNoEsNumero() {
           if [ n "$1" \
           a "$1" != "0" \
           a "`echo $1 | awk '{ print $1*1 }'`" != "$1" ]; then
                echo "El parámetro '$1' no es un número"
                exit 2
           fi
        }

           if [ $# ne 2 ]; then
                echo "El número de parámetros debe de ser igual a 2"
                exit 1
           fi
           comprobarQueNoEsNumero $1
           comprobarQueNoEsNumero $2
           echo $1 $2 | awk '{ print $1  $2 }'
```

103. Factorial de un número pasado por parámetro:

```
#!/bin/bash
   function factorial(){
     fact=1;
     cont=1;
     while test $1 -ge $cont
      do
        fact=`expr $fact \* $cont`;
        cont=`expr $cont + 1`;
     done
     echo "El factorial de $1 es $fact";
   }

        factorial $1;
```

104. Buscar archivos con una determinada extensión que son pasados con varios parámetros:

```
#!/bin/bash
   echo "Numero de parametros pasados: $#";
   for param in "$@"
    do
      find / -name *.param;
    done
```

105. Realizar la tabla de multiplicar de un número pasado por parámetro:

```
#!/bin/bash
   n=0
   while test $n -le 10
    do
      result= `expr $n \* $1`
      echo $1*$n = [$result]
      n= `expr $x +1`
      echo $x
```

106. Mostrar números del 1 a 10:

```
#!/bin/bash
   for x in `seq 1 10`
    do
      echo [$x];
    done
```

107. Listar todos los archivos del directorio bin también listar todos los archivos del directorio tmp.

```
$ ls /bin
$ls /tmp
```

108. Mostrar por pantalla los archivos ordinarios del directorio HOME y sus subdirectorios.

```
$ ls -R $HOME
```

109. Listar todos los archivos del directorio etc que empiecen por t en orden inverso.

```
 $ ls -dr /etc/t*
```

110. Listar todos los archivos del directorio dev que empiecen por tty y tengan 5 caracteres.

```
 $ ls /dev/tty??
```

111. Listar todos los archivos del directorio dev que empiecen por tty y acaben en 1,2,3 ó 4.

$ ls /dev/tty*[1-4]

112. Listar todos los archivos del directorio dev que empiecen por t y acaben en C1.

$ ls /dev/t*C1

113. Listar todos los archivos, incluidos los ocultos, del directorio raíz.

$ ls -a /

114. Listar todos los archivos del directorio etc que no empiecen por t.

$ ls -d /etc/[^t]*

115. Listar todos los archivos del directorio usr y sus subdirectorios.

$ ls -R /usr

116. Cambiarse al directorio tmp y verificar que el directorio actual ha cambiado.

$ cd /tmp
$ pwd

117. Con un solo comando posicionarse en el directorio $HOME.

$ cd /HOME

118. Mostrar el día y la hora actual.

$ date +"%A %D - %r
#o simplemente DATE

119. Verificar que se está en él.

$ pwd

120. Listar todos los ficheros del directorio HOME mostrando su número de i-nodo.

echo $ ls -i

```
$ rm -rf PRUEBA/*
```

121. Crear los directorios dir1, dir2 y dir3 en el directorio PRUEBA. Dentro de dir1 crear el directorio dir11. Dentro del directorio dir3 crear el directorio dir31. Dentro del directorio dir31, crear los directorios dir311 y dir312.

```
$ mkdir PRUEBA/{ dir1,\
dir1/dir11,\
 dir2,\
dir3,\
dir3/dir31,\
 dir3/dir31/dir311,\
 dir3/dir31/dir312}
```

122. Crear el directorio dir2 y dir3 en el directorio PRUEBA ¿Cuáles son los actuales permisos del directorio dir2?

```
ls PRUEBA
 $ mkdir dir1 dir2
```

123. Copiar el archivo /etc/motd a un archivo llamado mensaje de vuestro directorio PRUEBA.

```
$ cp /etc/motd ./PRUEBA
```

124. Copiar mensaje en dir1, dir2 y dir3.

```
$ cd PRUEBA
$ cp mensaje dir1/mensaje && cp mensaje dir2/mensaje && cp mensaje
dir3/mensaje
```

125. Comprobar el ejercicio anterior mediante un solo comando.

```
$ ls -R PRUEBA
```

126. Copiar los archivos del directorio rc.d que se encuentra en /etc al directorio dir31.

```
$ cp -r /etc/rc.d dir3
```

127. Copiar en el directorio dir311 los archivos de /bin que tengan una a como segunda letra y su nombre tenga cuatro letras.

$ cp -r /bin/?a?? PRUEBA/dir3/dir31/dir311

128. Copiar el directorio de otro usuario y sus subdirectorios debajo de dir11 (incluido el propio directorio).

$ sudo cp -r ../user_other PRUEBA/dir1/dir11
$ cp -r ../user PRUEBA/dir1/dir11

129. Mover el directorio dir31 y sus subdirectorios debajo de dir2
.

$ mv PRUEBA/dir3/dir31 PRUEBA/dir2

130. Ocultar el archivo mensaje del directorio dir3.

$ mv PRUEBA/dir3/mensaje PRUEBA/dir3/.mensaje

131. Copiar al directorio dir312 los ficheros del directorio /dev que empiecen por t, acaben en una letra que vaya de la a a la b y tengan cinco letras en su nombre.

$ ls /dev/t???[a*b]

132. Borrar los archivos de dir312 que no acaben en b y tengan una “q” como cuarta letra.

$ find dir312 -type f -regex ".*???q[^b$]" -exec rm -r {} \;

133. Mover el directorio dir312 debajo de dir3.

$ mv PRUEBA/dir2/dir31/dir312 PRUEBA/dir3

134. Crear un enlace simbólico al directorio dir1 dentro del directorio dir3 llamado enlacedir1.

$ ln -s /home/usuario1/PRUEBA/dir1 PRUEBA/dir3/enlacedir1

135. Posicionarse en dir3 y, empleando el enlace creado en el ejercicio anterior, crear el directorio nuevo1 dentro de dir1.

$ cd PRUEBA/dir3 $ mkdir enlacedir1/nuevo1

136. Utilizando el enlace creado copiar los archivos que empiecen por u del directorio /bin en directorio nuevo1.

$ cp -r /bin/u* enlacedir1/nuevo1/

137. Crear dos enlaces duros del fichero fich1, llamarlo enlace, en los directorios dir1 y dir2.

$ ln fich1 dir1/enlace
$ ln fich1 dir2/enlace

138. Borrar el archivo fich1 y copiar enlace en dir3.

$ rm fich1 $ cp dir1/enlace dir3/ $ ln -s /home/usuario1/PRUEBA/dir2/enlace /home/usuario1/PRUEBA/dir1/enlafich1

139. Crear un enlace simbólico (llamado enlafich1) al fichero enlace de dir2 en dir1.

$ ln -s dir2/enlace dir1/enlafich1

140. Posicionarse en dir1 y, mediante el enlace creado, copiar el archivo fichl dentro de dir311.

$ cd dir1
dir1 $ cp enlafich1 ../dir2/dir31/dir311/fich1

141. Seguir en dir1 y, mediante el enlace creado, sacar por pantalla las líneas que tiene el archivo fich1.

dir1$ cat enlafich1

142. Borrar el fichero fich1 de dir2

PRUEBA$ rm dir2/fich1

143. Borrar los archivos y directorios de dir1, incluido el propio directorio.

$ rm -rf PRUEBA/dir1

144. Borrar todos los archivos y directorios creados durante los ejercicios.

$ rm -r *

145. ¿Cuáles son ahora los permisos asociados a dir2?
$ ls -la ./dir2

146. Crear bajo dir2, un directorio llamado dir2l.

ls dir2
mkdir dir21

147. ¿Cuáles son los valores por omisión asignados a los archivos?

$ touch dir1/{file1,file2,file3}
PRUEBA$ ls -l dir1

148. Cambiar el directorio actual al directorio dir3. Imprimir su trayectoria completa para verificar el cambio.

$ ls
dir1 dir2 dir3
$ mv dir1 dir3/
$ ls -lR.:
./dir2:
./dir2/dir21:
./dir3:
./dir3/dir1:

149. ¿Cuáles son los permisos asignados en su momento a este directorio?

./dir3:

150. Establecer mediante el comando umask (buscar este comando) los siguientes valores por omisión: rwxr--r-- para los directorios y rw-r--r-- para los archivos ordinarios.

umask 0033

151. Utilizando la notación simbólica, eliminar todos los permisos de escritura (propietario, grupo, otros) del directorio dir2
. .
$ chmod = dir1

152. Utilizando la notación octal, eliminar el permiso de lectura del directorio dir2, al resto de los usuarios.

$ chmod 751 dir2

153. Concederse a sí mismo permiso de escritura en el directorio dir2 e intentar de nuevo el paso anterior.

$ chmod 200 dir1
$ ls -l $ mkdir dir1/dir21
mkdir: no se puede crear el directorio «dir1/dir21»: Permiso denegado