



# Preparador Informática

[www.preparadorinformatica.com](http://www.preparadorinformatica.com)

## PRÁCTICA 17 SHELL SCRIPTS

OPOSICIONES SAI 2018  
CASTILLA Y LEÓN

## EJERCICIOS

El siguiente ejercicio sobre Shell script está sacado del examen de PTFP SAI de las oposiciones de 2018 en Castilla y León. Lo hemos utilizado para los simulacros del mes 8 de Preparación de:

- PES INFORMÁTICA CANTABRÍA/COM. VALENCIANA/ARAGÓN /LA RIOJA/MADRID \*
- PTFP SAI ANDALUCÍA \*/ARAGÓN/CANTABRIA/CASTILLA Y LEÓN/ LA RIOJA / COM. VALENCIANA/MADRID \*/

\* En estas comunidades solo se ha utilizado el ejercicio 5

1. Indicar el comando Linux que debe de emplearse si se quiere listar el contenido únicamente de los directorios que se llamen casa, cama, cara, cata.
2. Se tiene un directorio *dir1* en el directorio actual y se quiere tener un enlace del mismo directorio que se llame *enlacedir1* en el directorio padre del actual. Indicar el comando Linux a utilizar.
3. Se supone que en el directorio actual se encuentra el Shell script *borrarArchivo.sh*. Hacer que se pueda ejecutar desde cualquier sitio utilizando un comando Linux en segundo plano y que redirija la salida estándar al archivo *salida.txt* y el error estándar al archivo *log.txt*
4. Indicar el comando Linux y máscara que se ha de utilizar para que al crear directorios otorgue por defecto los permisos *rwx r-x r—*. Solamente se contará la respuesta si está explicada razonadamente.
5. Crear un shell script en bash que permita crear de forma automática usuarios en Linux. Para ello, el script:
  - Recibe como parámetro el nombre de un archivo. Si el número de parámetros es incorrecto, o si el parámetro no corresponde al nombre de un archivo, se mostrará un mensaje de error.
  - El archivo tendrá una línea para cada grupo, con el nombre del grupo y el número de alumnos separados por el carácter dos puntos (:).  
*grupo1:número1*  
*grupo2:número2*  
...
  - El script va recorriendo las líneas del archivo y:
    - Crea el grupo. Al crear el grupo mostrará "*Se ha creado el grupo .....*" y si ya existe el grupo, no mostrará nada.
    - Crea los usuarios con el nombre *grupo1-1*, *grupo1-2*, etc. hasta el número de usuarios del grupo y como grupo principal el grupo correspondiente. Para cada usuario creado correctamente, mostrará el mensaje "*Se ha creado el usuario ...*". Por si estos usuarios existiesen los eliminará previamente al crearlos junto con su directorio personal.

Ejemplo. Si el archivo contiene:

*asir2:3*  
*smr1:2*

Crear los usuarios *asir2-1*, *asir2-2*, hasta *asir2-3*; y los usuarios *smr1-1*, *smr1-2*.



## SOLUCIÓN PROPUESTA EJERCICIOS DEL 1 AL 4

1. Indicar el comando Linux que debe de emplearse si se quiere listar el contenido únicamente de los directorios que se llamen casa, cama, cara, cata.

```
ls ca[smrt]a
```

2. Se tiene un directorio *dir1* en el directorio actual y se quiere tener un enlace del mismo directorio que se llame *enlacedir1* en el directorio padre del actual. Indicar el comando Linux a utilizar.

```
ln -s dir1/ ../enlacedir1
```

3. Se supone que en el directorio actual se encuentra el Shell script *borrarArchivo.sh*. Hacer que se pueda ejecutar desde cualquier sitio utilizando un comando Linux en segundo plano y que redirija la salida estándar al archivo *salida.txt* y el error estándar al archivo *log.txt*

1º Darle permisos de ejecución:

```
chmod +x borrarArchivo.sh
```

2º Incluir el script en el PATH para que pueda ser ejecutado desde cualquier sitio:

```
mv borrarArchivo.sh /usr/bin
```

3º Ejecuta el script en segundo plano y redirige la salida estándar al archivo *salida.txt* y la salida de error al archivo *log.txt*

```
borrarArchivo.sh > salida.txt 2> log.txt &
```

4. Indicar el comando Linux y máscara que se ha de utilizar para que al crear directorios otorgue por defecto los permisos *rw-r--r--*. Solamente se contará la respuesta si está explicada razonadamente.

```
umask 023
```

Por lo general, cada vez que se utiliza una cuenta de usuario limitado para crear un archivo o un directorio, se establecen por defecto los permisos *rw-rw-r--* (octal 664) y *rw-rw-r-x* (octal 775), respectivamente. Si lo hacemos como root, los permisos son *rw-r--r--* (octal 644) y *rw-r--r-x* (octal 755). Dichos permisos por defecto pueden modificarse con el comando *umask*.

Con *umask* podemos definir la máscara de permisos, cuyo valor original es 022. El permiso por defecto será el resultado de restar del permiso original, el valor de la máscara. Por ejemplo: Si deseamos que los archivos se creen con permisos 644 (lo más habitual), pondremos máscara 022 ya que  $666-022=644$ .

En el caso de las carpetas, el permiso efectivo será 755 ya que  $777-022=755$ . Si analizamos el valor de la máscara en binario, cada bit a '1' desactiva un permiso y cada bit a '0' lo activa, es decir, si tiene un valor 022 (000 010 010) cuando creamos una carpeta, tendrá permisos *rw-r--r-x* y cuando creamos un archivo tendrá permisos *rw-r--r--*.

**NOTA:** La modificación con *umask* de la máscara por defecto no afecta a los archivos y carpetas existentes sino solo a los nuevos que cree ese usuario a partir de ese momento.



## SOLUCIÓN PROPUESTA EJERCICIO 5 (VERSIÓN 1)

```
#!/bin/bash

#####INICIO DE DECLARACIÓN DE FUNCIONES#####

#Funcion que crea los grupos.
#La función recibe como parámetro el fichero que contiene los grupos
#Parámetros
#$1: el fichero con los grupos
function crearGrupos
{
    lineas=`cat $1`
    for linea in $lineas
    do
        GRUPO=$(echo $linea | cut -d: -f1)
        NALUMNOS=$(echo $linea | cut -d: -f2)
        groupadd $GRUPO &> /dev/null

        #Si el grupo se ha creado correctamente
        if [ $? -eq 0 ]
        then
            echo " Se ha creado el grupo $GRUPO"
            #Se llama a la función crearUsuarios para
            #crear los usuarios del grupo
            crearUsuarios $GRUPO $NALUMNOS
        fi
    done
}

#Funcion que crea los usuarios.
#La función recibe el nombre de grupo y el número de alumnos
#Parámetros
#$1: el grupo
#$2: el número de alumnos que tiene el grupo
function crearUsuarios
{
    #Se utiliza la estructura for para crear el número de alumnos indicado por parámetro
    for (( nAlumno=1; nAlumno<=$2; nAlumno++ ))
    do
        nombreUsuario="$1-$nAlumno";
        #Se comprueba si el usuario ya existe en el sistema
        #El comando grep devolverá alguna línea si existe un usuario con dicho nombre
        grep "^$nombreUsuario" /etc/passwd &> /dev/null

        #Si $? devuelve un valor distinto de 0 significa que el comando anterior (grep)
        #ha fallado al ejecutarse y significa que no existe el usuario en el sistema
        if [ $? -ne 0 ]
        then
            useradd -g $1 -d "/home/$nombreUsuario" -m $nombreUsuario 2> /dev/null &&
            echo " Se ha creado el usuario $nombreUsuario";

            #Si $? devuelve un valor igual a 0 significa que el comando anterior (grep)
            #se ha ejecutado correctamente y significa que ya existe el usuario en el sistema,
            #por lo que se procede previamente a su eliminación y posteriormente se crea.
            else
                userdel -r $nombreUsuario 2> /dev/null
                useradd -g $1 -d "/home/$nombreUsuario" -m $nombreUsuario 2> /dev/null &&
                echo " Se ha creado el usuario $nombreUsuario";
            fi
        done
    }
}
```



```
#####FIN DE DECLARACIÓN DE FUNCIONES#####

#####INICIO PROGRAMA PRINCIPAL#####

#Se comprueba si el usuario que ejecuta el script tiene permisos de root
if [ $UID -ne 0 ]
then
    echo "ERROR 1: EL script debe ser ejecutado por root para permitir el alta de usuarios"
    exit 1
fi

#Se comprueba que se pasa exactamente un parámetro al script
if [ $# -ne 1 ]
then
    echo "ERROR 2: El número de parámetros es incorrecto"
    exit 2
fi

#Se comprueba si el fichero debe ser regular y contr al menos una linea
if [ ! -f $1 ]
then
    echo "ERROR 3: El parámetro introducido debe ser un fichero regular"
    exit 3
fi

FICHERO=$1
crearGrupos $FICHERO
```



Preparador Informática



## SOLUCIÓN PROPUESTA EJERCICIO 5 (VERSIÓN 2)

```
#!/bin/bash

#####INICIO DE DECLARACIÓN DE FUNCIONES#####

#Funcion que crea los grupos.
#La función recibe como parámetro el fichero que contiene los grupos
#Parámetros
#$1: el fichero con los grupos
function crearGrupos
{
    #Se establece el separador del sistema al carácter :
    AUX=$IFS
    IFS=":";
    while read GRUPO NALUMNOS
    do
        groupadd $GRUPO &> /dev/null

        #Si el grupo se ha creado correctamente
        if [ $? -eq 0 ]
        then
            echo " Se ha creado el grupo $GRUPO"
        fi

        #Se llama a la función crearUsuarios para
        #crear los usuarios del grupo
        crearUsuarios $GRUPO $NALUMNOS

    done < $1

    # Se reestable el separador del sistema a su valor por defecto
    IFS=$AUX;
}

#Funcion que crea los usuarios.
#La función recibe el nombre de grupo y el número de alumnos
#Parámetros
#$1: el grupo
#$2: el número de alumnos que tiene el grupo
function crearUsuarios
{
    #Se utiliza la estructura for para crear el número de alumnos indicado por parámetro
    for (( nAlumno=1; nAlumno<=$2; nAlumno++ ))
    do
        nombreUsuario="$1-$nAlumno";
        #Se comprueba si el usuario ya existe en el sistema
        #El comando grep devolverá alguna línea si existe un usuario con dicho nombre
        grep "^$nombreUsuario" /etc/passwd &> /dev/null

        #Si $? devuelve un valor distinto de 0 significa que el comando anterior (grep)
        #ha fallado al ejecutarse y significa que no existe el usuario en el sistema
        if [ $? -ne 0 ]
        then
            useradd -g $1 -d "/home/$nombreUsuario" -m $nombreUsuario 2> /dev/null &&
            echo " Se ha creado el usuario $nombreUsuario";
        fi
    done
}
```



```
#Si $? devuelve un valor igual a 0 significa que el comando anterior (grep)
#se ha ejecutado correctamente y significa que ya existe el usuario en el sistema,
#por lo que se procede previamente a su eliminación y posteriormente se crea.
else
    userdel -r $nombreUsuario 2> /dev/null
    useradd -g $1 -d "/home/$nombreUsuario" -m $nombreUsuario 2> /dev/null  &&
    echo " Se ha creado el usuario $nombreUsuario";
fi
done
}

#####FIN DE DECLARACIÓN DE FUNCIONES#####

#####INICIO PROGRAMA PRINCIPAL#####

#Se comprueba si el usuario que ejecuta el script tiene permisos de root
if [ $UID -ne 0 ]
then
    echo "ERROR 1: EL script debe ser ejecutado por root para permitir el alta de usuarios"
    exit 1
fi

#Se comprueba que se pasa exactamente un parámetro al script
if [ $# -ne 1 ]
then
    echo "ERROR 2: El número de parámetros es incorrecto"
    exit 2
fi

#Se comprueba si el fichero debe ser regular y contr al menos una linea
if [ ! -f $1 ]
then
    echo "ERROR 3: El parámetro introducido debe ser un fichero regular"
    exit 3
fi

FICHERO=$1
crearGrupos $FICHERO
```