ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
Universitat Rovira i Virgili

[DΣIM]

GESTOR DE CÓPIAS DE SEGURIDAD

FUNDAMENTOS DE SISTEMAS OPERATIVOS

Estudiante: Joan David Frent Frent

Profesor/a: Stéphane Salaet

Fecha Entrega: 11 de junio de 2024

Grupo de Laboratorio: L3

ÍNDICE

1. Decisiones de diseño

Para empezar, para cada script se ha creado una cabecera explicativa de lo que hace el script pasado y los argumentos de entrada necesarios para hacer funcionar bien el script.

Además, para evitar explicaciones redundantes se ha de precisar que se ha creado las banderas *kFlag* y *DFlag* para controlar si se han activado las opciones *-k* y *-D* (duplicado de ficheros y *DryRun* respectivamente) y, si se ha introducido una opción errónea como *-K* y *-d* terminamos el script e informamos del porqué se ha terminado.

También se ha creado una función para controlar si se ha introducido un número de argumentos menor al esperado.

Para cada operación que se haga (modificar, crear, eliminar, …) se comprobará si está activo el *DryRun* para solo haber de informar.

1.1. Copia de seguridad

Estructura principal del comando a ejecutar:

Copia [-k] [-D] *fitxer1 fitxer2 ... fitxer_tgz*

Para empezar, se ha decidido crear una variable que contendrá el nombre del comprimido y otro que contendrá todos los ficheros a añadir en el comprimido.

Claramente se creará un bucle que comprobará si existen todos los ficheros pasados por parámetro para asegurarnos que no habrá problemas de ningún tipo, si es el caso que algún fichero no existe, abortamos e informamos del problema.

Si se crea el comprimido des de cero, es decir si no existía anteriormente simplemente creamos el comprimido con todos los ficheros.

Para trabajar con un comprimido que ya existe, se usará un directorio temporal para hacer las comprobaciones necesarias.

Si se hace la compresión con un comprimido existente, comprobamos si está activo la opción *-k*, si es el caso, todos los ficheros que se repitan se creará un duplicado con la fecha al final de nombre.

Por ejemplo:

- hay dos ficheros/directorios iguales llamados abc.txt
- Crearemos dos versiones: abc.txt – abc20240523.txt

Si no está activo *-k* informamos de que no se puede hacer la recompresión por temas de seguridad.

1.2. Recuperación de un directorio o fichero a otro directorio

Estructura principal del comando a ejecutar:

Recupera [-D] *fitxer_tgz fitxer1 fitxer2 ... directori_destí*

Para empezar, se ha decidido crear una variable que contendrá el nombre del comprimido a descomprimir, otra con los ficheros a recuperar y otra con el nombre del directorio destino para guardar los ficheros recuperados.

Para trabajar usaremos un directorio temporal en el que se descomprimirá el contenido del comprimido.

A continuación, necesitaremos verificar si el fichero pasado por parámetro existe en el directorio temporal, si no es el caso, buscaremos el mismo fichero, pero con fecha incluida, si se encuentra, recuperamos dicho fichero y continuamos hasta el final.

Claramente, en cada operación se comprobará si la opción *-D* (*DryRun*) está activado para solo informar de cada paso hecho.

Para liberar espacio eliminaremos el directorio temporal.

1.3. Automatización con *crontab*

Estructura principal del comando a ejecutar:

Configura *dia hora script parametres*

Para empezar, se han creado diversas variables para guardar el *día*, *hora*, *script* y *parámetros* para agilizar una lectura ordenada y correcta de cada argumento.

Para las variables *día* y *hora* se ha creado un comprobante para ver si se ha introducido un número o no y si está dentro del rango establecido.

Luego creamos una variable llamada *comandoCrontab* que contendrá el comando para poder ejecutar crontab correctamente.

Finalmente, tendremos un comprobante para verificar si crontab se ha ejecutado o no correctamente.

2. Juego de Pruebas

El juego de pruebas servirá para registrar todos los posibles casos de error que puede aparecer al ejecutar el script y así abarcar todas las posibilidades posibles.

Copia de Seguridad		
PRUEBAS	***RESULTADO ESPERADO***	***¿OK?***
Opción **-k** y/o **-D** incluida.	Mensaje de confirmación de opción.	
Opción **-k** y **-D** no incluida.	Mensaje de confirmación.	
Opción **-K** y/o **-d.**	Mensaje de error y termina.	
Argumentos pasados < 2 (sin opciones -k o -D).	Informa y termina.	
Argumentos pasados no existen (excepto *fichero_tgz*).	Informa y termina al primer inexistente.	
Los ficheros o directorios no existen.	Informa y termina el script.	
No existe el nombre del comprimido.	Crea el comprimido con los ficheros a guardar.	
Comprimimos ficheros o directorios repetidos [con **-k**].	Se crea un duplicado con la fecha incluida en el fichero copiado.	
Comprimimos ficheros o directorios repetidos [sin **-k**].	Simplemente no hace duplicado. No informa nada.	

Extracción de un fichero comprimido		
PRUEBAS	***RESULTADO ESPERADO***	***¿OK?***
No se introduce suficientes parámetros.	Informa y termina el script.	
Se introduce **-D** o **-d**.	Informa y continua o termina el script, respectivamente.	
No existe el comprimido para extraer.	Informa y termina el script.	
Insertamos ficheros que no existen en el comprimido.	Informa y no recupera el comprimido que no existe.	
Insertamos un directorio destino que no existe.	Crea el directorio destino.	
Insertamos un fichero que no existe, pero hay uno con fecha.	Recuperamos la versión con fecha.	
Insertamos un fichero que existe para recuperar.	Recupera el fichero dentro del directorio destino.	
Activamos la opción *-D.*	Informamos en cada momento los pasos realizados.	

Automatización con CRONTAB		
PRUEBAS	***RESULTADO ESPERADO***	***¿OK?***
Introducimos un día incorrecto de los límites del rango [0-6].	Muestra un mensaje de error y abortamos.	
Introducimos una hora incorrecta de los límites del rango [0-23].	Muestra un mensaje de error y abortamos.	
Comprobamos que se recoge bien el nombre del script a ejecutar.	Muestra el nombre del script y continuamos.	
Comprobamos que se recoge bien los parámetros del script a ejecutar.	Muestra los parámetros del script y continuamos.	
Crontab se ejecuta según los parámetros pasados (con CopiaSeguridad.sh)	Aparece un comprimido con los argumentos pasados.	

3. Código

3.1. CopiaSeguridad.sh

```
#================================================================
=========
#                        1. COPIA DE SEGURIDAD
#
# Comando: ./script [-k] [-D] fich1 fich2 ... fich.tgz(o tar.gz)
#
#   "fich1 fich2 ,,,": ficheros para realizar la copia de
seguridad.
#
#   "fich.tgz(o tar.gz)": nombre del comprimido
#
# Claramente ha de existir los ficheros o directorios a copiar
# Las carpetas que se hayan introducido se ha de copiar TODO el
#   contenido que tenga dentro correctamente sin modificar.
#
# Opción -k: (por temas de precisión solo se aceptará la k
minúscula)
#  Si al añadir un fichero al comprimido ya existe, conservar las
dos
#   y, la mas reciente, ponerle la fecha AAAAMMDD al final del
nombre
#  Ej: ak.c(existente) == ak.c(añadido) --> ak.c y ak20240309.c
#
# Opción -D: (por temas de precisión solo se aceptará la D
MAYÚSCULA)
#  Va informando al usuario que hace el script (comprobando,
añadiendo...)
#    sin ejecutar nada
#
# Nota Adicional: ha de permitir tanto rutas absolutas como
relativas
#  en la entrada y/o salida
#
#================================================================
=========
#!/bin/bash

# Funcion que muestra un mensaje de error si no se
#  llama al script correctamente
function mssgError () {
   echo " <<< Comando Erroneo. ABORTAMOS >>>"
   echo "$0 [-k] [-D] fich1 fich2 ... nombreComprimido"
```

```
    exit 1
}

# Definimos banderas para el -k y -D para saber si se han
introducido
kFlag=0
DFlag=0 # Ambos desactivados inicialmente

# Iteramos sobre los primeros dos argumentos
#  para comprobar si hemos activado alguna opción
for arg in "$1" "$2"; do
    case "$arg" in
        -k) kFlag=1 # Actualizamos la bandera
            shift # Desplazamos argumentos para leer el resto
            echo "Opcion -k activado" ;;

        -D) DFlag=1 # Actualizamos bandera
            shift # Desplazamos argumentos para leer el resto
            echo "Opcion -D activado" ;;

        -K) echo "-- ABORTANDO -- Opcion -K incorrecta. Debe ser -
k"
            exit 1 ;;

        -d) echo "-- ABORTANDO -- Opción -d incorrecta. Debe ser -
D"
            exit 1 ;;
    esac
done

# Mostramos si no se han activado ningun flag
#if [ $kFlag -eq 0 ] && [ $DFlag -eq 0 ]; then
#   echo "Ninguna opcion activada (-k y -D)."
#fi

# Comprobamos los argumentos después de mirar las opciones
if [ $# -lt 2 ]; then
    mssgError
fi

# Obtenemos el nombre del comprimido (ultimo argumento)
dirDest="${@: -1}"

# Obtenemos el resto de argumentos excepto el ultimo y -k/-D
ficheros="${@:1:$#-1}"

# Bucle que recorre los ficheros a hacer copia y comprueba si
existen
```

```
for arg in ${ficheros[@]}; do
    # Miramos si no existe y, si es el caso, mostramos mensaje de
error y abortamos
    if [ ! -e $arg ]; then
        echo "-- Arch/Dir no existe: '$arg' ABORTAMOS--"
        exit 1
    fi

    # Indicamos que existe el archivo o directorio insertado
    echo "-- Arch/Dir existe: '$arg' --"
done

if [ -e "$dirDest" ]; then # Verificar si el comprimido ya existe

    if [ $kFlag -eq 1 ]; then # Verificamos si -k está activado

        # Creamos directorio temporal
        dirTmp=$(mktemp -d)

        # Descomprimimos el comprimido existente en el dir temporal
        tar -xzf $dirDest -C $dirTmp

        for arg in ${ficheros[@]}; do # Bucle que itera sobre cada
argumento de los ficheros
            baseName=$(basename "$arg") # Obtenemos el nombre del
fichero
            fichDest="$dirTmp/$baseName" # Creamos ruta con el
nombre del fichero

            # Verificamos si el archivo ya existe
            if [ -e $fichDest ]; then
                # Obtenemos fecha de modificación del archivo
                dateMod=$(stat -c %y "$fichDest" | cut -d ' ' -f1)
                # Elimina los guiones. P.Ej: 2024-05-23 -> 20240523
                dateMod=$(echo "$dateMod" | tr -d '-')

                # Cambiar el nombre del archivo nuevo agregando la
fecha
                nuevoFich="${baseName}.${dateMod}"
                cp -r "$arg" "$dirTmp/$nuevoFich" # Copia recursiva
del original al original+fecha

            else # Si el archivo no existe, simplemente lo copiamos
                cp -r "$arg" "$dirTmp"
            fi
        done

        # Volvemos a comprimir los archivos
```

```
        if [ $DFlag -eq 1 ]; then
            echo "Añadiendo nuevo ficheros al comprimido <<
$dirDest >>"
            find "$dirTmp" -type f
        else # Hacemos comprimido del temporal al destino con todo
el contenido
            tar -czf $dirDest -C $dirTmp .
        fi

        rm -rf "$dirTmp" # Eliminamos el directorio temporal

    else # Si el comprimido ya existe y -k desactivado
        echo "El comprimido << $dirDest >> ya existe y no se ha
activado -k"
        exit 1
    fi

else # No existe el comprimido a crear
    if [ $DFlag -eq 1 ]; then # Si DryRun está activado
        echo "Creando el comprimido << $dirDest >> con los
siguientes ficheros:"

        # Recorrido por todos los argumentos de ficheros
        for arg in ${ficheros[@]}; do
            echo "$arg"
        done

    else # Simplemente creamos el nuevo comprimido
        tar -zcf $dirDest $ficheros
    fi
fi

echo "Copia de seguridad completada"
```

3.1. Recuperación.sh

```
#================================================================
=========
#   1. Recuperación de un directorio o fichero a otro directorio
#
# Comando: ./script [-D] fich_tgz fich1 fich2 ... dir_destino
#
# Recupera los ficheros y directorios que se le pasan por parámetro
des de un archivo
#  comprimido (.tgz o .tar.gz).
#
# "fich_tgz": comprimido a extraer
#
# "fich1 fich2 ...": ficheros para extraer
#
# "dir_destino": ubicación de la extracción de ficheros del
comprimido
#
# Opción -D: (por temas de precisión solo se aceptará la D
MAYÚSCULA)
#  Va informando al usuario que hace el script sin ejecutar nada
#
#================================================================
=========
#!/bin/bash

# Funcion que sirve para mostrar un mensaje de error de comando
function mssgError () {
    echo "Comando introducido erroneo"
  echo "$0 [-D] fich_tgz fich1 fich2 ... dir_destino"
  exit 1
}

# Para saber si -D está activado
DFlag=0

# Comprobamos si -D está activado
for arg in "$1"; do
  case "$arg" in
    -D) DFlag=1 # Actualizamos bandera
        shift # Desplazamos argumentos para leer el resto
        echo "Opcion -D activado" ;;

    -d) echo "-- ABORTANDO -- Opción -d incorrecta. Debe ser -D."
        exit 1 ;;
  esac
```

```
done

# Comprobamos que se han isertado suficientes argumentos
#  Mínimo: nombreComprimido ficheroAExtraer DirectorioDestino
if [ $# -lt 3 ]; then
  mssgError
fi

# Obtenemos el comprimido a leer
dirComprimido="$1"
shift # Desplazamos argumentos para leer el resto

# Obtenemos los ficheros a recuperar excepto el ultimo
ficheros="${@:1:$#-1}"

# Obtenemos el directorio a guardar
dirGuardar="${@: -1}"

# Miramos si existe el comprimido
if [ ! -e $dirComprimido ]; then
    echo "El comprimido a leer no existe: << $dirComprimido >>"
    exit 1
fi

dirTmp=$(mktemp -d) # Creamos directorio temporal
tar -xzf $dirComprimido -C $dirTmp # Descomprimimos en la temporal

# Iteramos sobre los archivos a recuperar
for arg in ${ficheros[@]}; do
    baseName=$(basename "$arg") # obtenemos el nombre base del
archivo

    if [ -e "$dirTmp/$basename" ]; then # Verificamos si el archivo
existe en el temporal
        if [ $DFlag -eq 1 ]; then
            echo "Recuperamos << $baseName >> a << $dirGuardar >>"
        else
            cp -r "$dirTmp/$baseName" "$dirGuardar" # Copiamos al
directorio destino
        fi
    else # Si no existe en el temporal, buscamos el mismo fichero
con fecha
        fileMod=$(echo "$baseName" | grep -oE '[0-9]{8}$')

        if [ -n $fileMod ]; then # Miramos si se encontro un
fichero con fecha
            # Reemplazamos el fichero eliminadno la fecha
            baseName=$(echo "$baseName" | sed "s/.$fileMod//")
```

```
            if [ $DFlag -eq 1 ]; then
                echo "Recuperamos << $baseName.$fileMod >> a <<
$dirGuardar >>"

            else # Copiamos del temporal al directorio destino
                cp -r "$dirTmp/$baseName.$fileMod" "$dirGuardar"
            fi
        fi
    fi
done

# Eliminamos el directorio temporal
echo "Eliminamos el directorio temporal << $dirTmp >>"
rm -rf "$dirTmp"

echo "Recuperacion completada"
```

3.2. Crontab.sh

```
#=====================================================================
=========
#                    1. Automatización con CRONTAB
#
# Comando: ./script dia hora script parametros
#
# Configuracion de una copia de seguridad automática de forma
periódica
#  usando "crontab".
#
# "dia": dia para ir haciendo la copia de seguridad
#
# "hora": hora para ir haciendo la copia de seguridad
#
# "script": script que ejecutar
#
# "parametros": parametros necesarios para hacer funcionar al
script
#
# El script ha de ser programado para que ejecute el comando
especificado
#  todos los días a la hora especificada.
#
# NOTA: "shift" se usará para desplazar los argumentos pasados
hacia la
#   izquierda
#
#=====================================================================
=========
#!/bin/bash

# Funcion que muestra un mensaje de error si se usa mal el comando
function mssgError () {
    echo "<<< Comando incorrecto >>>"
    echo "$0 dia hora script parametros"
    exit 1
}

# Comprobamos si hay suficientes argumentos de entrada
if [  $# -lt 4 ]; then
    mssgError
fi

dia=$1 # Obtenemos el dia
hora=$2 # Obtenemos la hora
```

```
# Comprobamos que se introduzca un numero
if ! [ $dia =~ ^[0-9]+$ ]; then
    echo "<< Error: 'dia' debe ser un numero. >>"
    exit 1
fi

# Comprobamos que sea un día de la semana
if [ $dia -gt 6 ] || [ $dia -lt 0 ]; then
    echo "<< Error: 'dia' esta fuera del rango esperado: [0-6]. >>"
    echo " < 0-Domingo; 1-Lunes; 2-Martes; 3-Miercoles; 4-Jueves;
5-Viernes; 6-Sabado >"
    exit 1
fi
shift

# Comprobamos que se introduzca un numero
if ! [ $hora =~ ^[0-9]+$ ]; then
    echo "<< Error: 'hora' debe ser un numero. >>"
    exit 1
fi

# Comprobamos que la hora sea la correcta
if [ $hora -gt 23 ] || [ $hora -lt 0 ]; then
    echo "<< Error: 'hora' not inside the expected range: [0-23].
>>"
    exit 1
fi
shift

nScript=$1 # Obtenemos el nombre del script
shift

argScript=$@ # Obtenemos los argumentos de los script

comandoCrontab="* ${hora} * * ${dia} ${nScript} ${argScript}"

echo "-- Comando preparado para ejecutar --"
echo "$comandoCrontab" | crontab -

# Miramos si crontab se ejecuta correctamente
if [ $? -eq 0 ]; then
    echo "Crontab configurado correctamente."
else
    echo "-- ABORTANDO -- Hubo un error al configurar el crontab."
fi
```