Documentación tarea 1, elo 330

Pascal Sigel - Oscar Silva

Sep-22-2014

1 Introducción:

En el presente documento se comentarán los hechos más importantes de la tarea, se hará un análisis de la solución propuesta y además se explicarán los problemas que tomaron más tiempo y se explicará cómo fueron solucionados.

En la siguiente sección se mostrará un bosquejo de la solución que se propone.

2 Solución:

La solución se basa esencialmente en 3 bloques,

1. el primer bloque es un bloque de ayuda donde se programa lo que el programa debe responder ante una entrada -h.
2. el segundo bloque del programa corresponde a una etapa de procesamiento de la información de entrada donde se toman las decisiones de qué hacer dependiendo de la entrada por teclado que el usuario ingrese.
3. El tercer bloque corresponde al procesamiento de los nombres, en caso que el usuario desee listar esto simplemente al encontrar un nombre lo lista en caso que el usuario desee efectuar un cambio de los nombres entonces el proceso corresponde a recursivamente cambiar el nombre de los archivos pero desde el extremo hasta la raíz, o sea primero se cambian los nombres de los archivos que se encuentran a más profundidad de búsqueda y en caso de la existencia de otro archivo con el mismo nombre que el que se propone a cambiar entonces se le agrega un “_1” y por robustez en caso que existiera uno con el posible nombre y otro con el agregado “_1” entonces se le agrega un “_2” y así sucesivamente.

La siguiente imagen ilustra los procesos del programa:

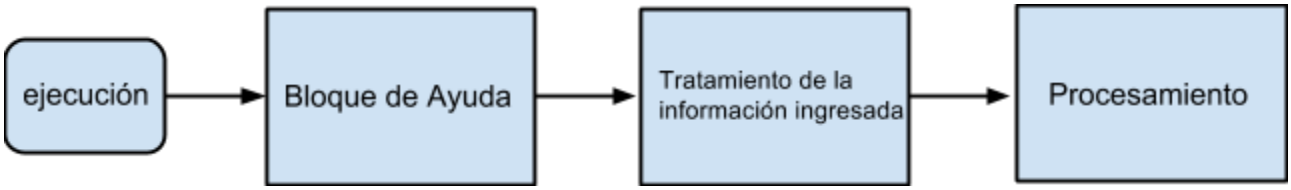

Fig. 1 procesos del programa

En la sección de anexos se expone el código.

Además se realizó un manual para que los usuarios que deseen usar el programa puedan ver.

Para la ejecución del programa se debe leer el readme.

3 Problemas que existieron:

En esta sección se expondrán los problemas que surgieron y sus soluciones.

Estructura de la solución con el comando find:

Como se puede ver en el anexo “código” la solución presentada posee una estructura que principalmente usa el comando “for” para la búsqueda recursiva de las carpetas y el posterior tratamiento de ellos, pero antes de usar “for” se trabajó con el comando find para tratar la información las conclusiones son las siguientes:

1. Si se intenta solucionar con “find -name * | while read line {...}” se tendrá dificultad en resolver la tarea dado que find envía todos los nombres por el pipe antes de que el while procese cada nombre, como ejemplo si una carpeta con ñ tiene dentro un archivo con ñ este archivo al ser leído supondrá que la carpeta tiene ñ aunque esta ya hubiera sido cambiada.
2. No se puede ejecutar una función creada en el mismo documento, por ejemplo funcion1, a cada resultado del comando “find” como sí se puede con comandos:
    a. funcion1 find -name * (no funcionará bien)
    b. rm find -name * (sí funcionará, no se recomienda probarlo)
3. usar find -name * -exec comando {}, es una alternativa pero al exec se le debe pasar como comando un comando del sistema como rm, mv, etc el problema es que la implementación por lo mismo es mucho más compleja que usar simplemente el for y una función escrita en el mismo programa; para usarlo habría que hacer un comando que sea aceptable por el exec y exterior al código.

Obviamente la solución fue cambiar la estructura a una más sencilla para solucionar la tarea como lo fue con “for”.

Autoaprendizaje:

La mayor parte del tiempo fue dedicada a la comprensión de la programación bash tanto la estructura como cada comando en particular, si bien no se puede decir que con la tarea se logra un dominio en el tema es cierto que si se logra una comprensión más profunda. Esencialmente la solución a este problema es la lectura proporcionada por lo manuales de linux (man comando) y también por la ayuda proporcionada en internet, un punteo de los comandos que fueron usados es el siguiente:

1. **sed**, este comando es usado para cambiar el nombre de las carpetas para luego poder hacer un mv (comando típico de linux para cambiar de nombre de una carpeta ó archivo) con el nombre que debe tener el archivo formateado tipo unix, el funcionamiento es el siguiente:
    a. comandoDePruebas | sed "s/${“ñ”}/${“n”}/g"` : la salida del comandoDePruebas es pasada como entrada del comando sed y esta cambia todas las ñ por n.

Anexos, Código fuente.

El siguiente es el código fuente del programa bash.

```
#! /bin/bash
#Función que proporciona un mensaje indicando como acceder a la información de ayuda.
help0()
{
	echo "Pruebe './fn2u -h' para más información"
	exit 0
}

#Función que proporciona una descripción del script, su forma de uso y algunos ejemplos.
help1()
{
cat <<HELP
fn2u -- Busca dentro de un directorio específico todos los directorios o archivos que posean dentro de sus nombres caracteres acentuados, espacios o caracteres ñ minusculas y mayusculas.

USO:
	fn2u [OPCION] DIRECTORIO ARGUMENTO

OPCION:
	-h
		Muestra el texto de ayuda y salir

ARGUMENTO:
	1
		Esta opción sólo indica el nombre de los directorios y archivos encontrados dentro del directorio especificado.
	2
		Esta opción renombra todos los directorios y archivos encontrados.

EJEMPLO:
	-Buscar y renombrar directorios y archivos dentro del directorio 'mi_directorio' que contengan los caracteres nombrados anteriormente.

		fn2u mi_directorio/ 2

	-Buscar y nombrar directorios y archivos dentro del directorio 'mi_directorio' que contenga los caracteres nombrados anteriormente.

		fn2u mi_directorio/ 1
```

```
HELP
  exit 0
}

#Función que permite identificar y modificar un determinado directorio
#o archivo si posee alguno de los caracteres nombrados anteriormente.
#Recibe como argumentos:
#1) Archivo o directorio a analizar.
#2) Opción proporcionada por el usuario.
change_name()
{
        #Definición de la extensión adicional necesaria para renombrar un directorio o archivo
        #con un nombre ya existente.
        AD="_"
        ADD=1
        #Definición de los caracteres que hay que sustituir y sus sustitutos.
        old=( "ñ" "Ñ" " " "á" "Á" "é" "É" "í" "Í" "ó" "Ó" "ú" "Ú" )
        new=( "n" "N" "_" "a" "A" "e" "E" "i" "I" "o" "O" "u" "U" )
        #Bandera que determina si el directorio o archivo posee alguno de los caracteres
indicados anteriormente.
        flag=0

        #Se almacena el path del directorio o archivo analizado en la variable path.
        path="$( dirname "$1" )/"
        #Se almacena el nombre del directorio o archivo analizado en la variable file.
        file=$( basename "$1" )
        #Se separa el path y el nombre del directorio o archivo con el fin de no modificar
        #algun directorio que sea parte del path proporcionado inicialmente. El objetivo
        #es solo modificar el directorio y sus subdirectorios y archivos.

        newfile=$file

        #Se comprueba si el directorio o archivo analizado posee alguno de los caracteres
        #mencionados anteriormente y se reemplazan los caracteres encontrados por sus
substitutos.
        for i in ${!old[*]} ; do
                if [[ "$file" == *${old[i]}* ]] ; then
                        #Con el comando sed se hace la sustitución de los caracteres
respectivos.
                        newfile=`echo "$newfile" | sed "s/${old[i]}/${new[i]}/g"`
                        #Se levanta la bandera que indica que el directorio o archivo fue
modificado.
                        flag=1
                fi
        done

        #Si la bandera esta en alto el programa continua, si no el programa se detiene.
```

```
        [[ $flag == 1 ]] || return 0

        #Dependiendo de la opción indicada por el usuario se aplican las siguientes rutinas:
        case $2 in
        1)
                #Si la opción indicada por el usuario es la número 1, se muestra el nombre del
directorio
                #o archivo que posee alguno de los caracteres indicados anteriormente.
                echo "$1" ;;
        2)
                #Si la opción indicada por el usuario es el número 2, se renombra el dorectorio
o archivo
                #que posea alguno de los caracteres indicados anteriormente.

                #Si se trata de un archivo se procede de la siguiente manera:
                if [ -f "$1" ] ; then
                        #Si el nombre modificado propuesto no existe se procede a modificarlo
directamente.
                        if [ ! -f "$path$newfile" ] ; then
                                echo "renombrando $1 como $path$newfile ..."
                        mv "$1" "$path$newfile"
                   #Pero si este ya existe      se le agrega al final una extensión adicional.

                        else
                                #Se separa del nombre del archivo su nombre y extensión
propia.
                                extension=".${newfile##*.}"
                                newfilename="${newfile%.*}"
                                #Se busca una extensión apropiada hasta que el nuevo nombre
propuesto no exista previamente.
                                while [ -f "$path$newfilename$AD$ADD$extension" ] ; do
                                        ADD=$(($ADD + 1))
                                done
                        echo "renombrando $1 como $path$newfilename$AD$ADD$extensión
..."
                            mv "$1" "$path$newfilename$AD$ADD$extension"
                        fi
                #Si se trata de un directorio se procede de la siguiente manera:
                elif [ -d "$1" ] ; then
                        #Si el nombre modificado propuesto no existe se procede a modificarlo
directamente.
                        if [ ! -d "$path$newfile" ] ; then
                                echo "renombrando $1 como $path$newfile ..."
                            mv "$1" "$path$newfile"
                        #Pero si este ya existe         se le agrega al final una extensión
adicional.
                        else
```

```
                                #Se busca una extensión apropiada hasta que el nuevo nombre propuesto no exista previamente.
                                while [ -d "$path$newfile$AD$ADD" ] ; do
                                        ADD=$(($ADD + 1))
                                done
                        echo "renombrando $1 como $path$newfile$AD$ADD ..."
                           mv "$1" "$path$newfile$AD$ADD"
                        fi
                fi ;;
        *)
                #Si la opción indicada no es ninguna de las dos anteriores la función arroja un mensaje de advertencia.
                echo "change_name: No se puede actuar usando <<$2>>: La opción indicada no es válida" ;;
        esac
}

#Dependiendo de los argumentos entregados por el usuario al script se aplican las siguientes rutinas:
case $# in
0)
        #Si no se entregan argumentos el script listará los directorios o archivos con caracteres
        #indicados anteriormente del directorio en el cual se esta ejecutando el bash.
        dir=.
        n=1 ;;
1)
        #Si se proporciona un solo argumento, debe ser un directorio o la opción '-h'.
        #Si se tratase de un directorio el script tomará por defecto la opción de listar
        #los directorios y archivos dentro del directorio indicado que posean alguno
        #de los caracteres indicados anteriormente.
        if [ -d "$1" ] ; then
                dir=$1
                n=1
        #Si se tratase de la opción '-h', el script desplegará un mensaje de ayuda y se cerrará
        elif [[ "$1" == "-h" ]] ; then
                help1
        #Si no fuese ninguno de los dos casos anteriores, el script mostrará una advertencia
        #y además recomendará al usuario ejecutar el script con la opción '-h' para recibir ayuda.
        else
                echo "$0\: no se puede actuar sobre <<$1>>: No existe el directorio"
                help0
        fi ;;
2)
        #Si se proporcionan dos argumentos, el primero debe ser un directorio y el segundo una
```

```
        #de las opciones indicadas anteriormente en el script y en el texto de ayuda.

        #Si el primer argumento efectivamente se tratase de un directorio se ejecuta la
siguiente rutina:
        if [ -d "$1" ] ; then
                dir=$1
                #Si el segundo argumento corresponde a una de las opciones indicadas
anteriormente en el
                #script y en el texto de ayuda, el script prosigue su ejecución.
                if [[ $2 == 1 || $2 == 2 ]] ; then
                        n=$2
                #De lo contrario el script mostrará una advertencia y además recomendará
                #al usuario ejecutar el script con la opción '-h' para recibir ayuda.
                else
                        echo "$0: No se puede actuar usando <<$2>>: El argumento indicado
no es valido"
                        help0
                fi
        #Si el primer argumento entregado por el usuario no corresponde a un directorio
        #el script mostrará una advertencia y además recomendará al usuario ejecutar
        #el script con la opción '-h' para recibir ayuda.
        else
                echo "$0\: no se puede actuar sobre <<$1>>: No existe el directorio"
                help0
        fi ;;
*)
        #Si la cantidad de argumentos fueran distintos a los indicados anteriormente el
        #script mostrará una advertencia y además recomendará al usuario ejecutar el
        #script con la opción '-h' para recibir ayuda.
        echo "$0: La cantidad de parámetros indicados no es válida"
        help0
esac

#Se recorrerá recursivamente el directorio que se desee analizar
for i in $dir/*
do
        #Si se encontrará un directorio, se llamará recursivamente al mismo script.
        if test -d "$i"
        then
                #Se llama al mismo script con el directorio encontrado.
                $0 "$i" $n
                #terminado el script anterior se procederá a actuar con la función
change_name
                change_name "$i" $n
        #De lo contrario se asumirá que se trata de un archivo y se procederá a actuar con la
función change_name
        else
```

```
                change_name "$i" $n
        fi
done
```