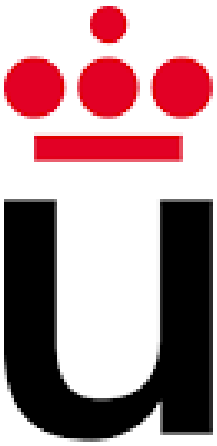

Universidad
Rey Juan Carlos

Práctica I - Shell Scripting

Carlos Chamizo Cano
Sistemas Operativos | 2ºIngeniería de Computadores
15-03-2021

ÍNDICE

Descripción

Elementos principales

En el script se puede distinguir una función principal (copy) y dos funciones adicionales (errorF y countF).

Función copy

Esta función recibe un único argumento que corresponde con la extensión de los archivos que se quieren copiar.

La dinámica de la función es la siguiente. Mediante un bucle “FOR IN” se busca en todos los directorios y subdirectorios del directorio origen, esta búsqueda se consigue gracias a que la variable dentro del FOR ha recibido como argumentos la salida del mandato find -type -d el cual lista todos los directorios en los que se debe buscar.
Antes de realizar ninguna operación dentro del for, se accede al directorio en el cual queremos buscar y mediante una variable auxiliar y el mandato find -type f -name "*$1" se puede saber si a partir de ese directorio existe algún fichero que se deba copiar, si existe se creará el directorio en la carpeta destino (en caso de que no esté ya creada) y se copiaran aquellos ficheros que coincidan con la extensión. En caso de no existir ficheros que coincidan con la extensión a partir de ese directorio, se pasará al siguiente directorio en la lista.

Función errorF

Esta función comprueba el valor del último mandato (con $?) ejecutado para comprobar si terminó con éxito o hubo un fallo. Este valor es almacenado en la variable ERRCODE.

La variable ERRCODE va cambiando su valor a medida que se ejecuta el script, en función del error que se produzca al ejecutar los distintos mandatos. Esta es devuelta al final, y muestra el número correspondiente al último error realizado. Los números asignados a cada error son:

cd -> 3 | mkdir -> 4 | cp -> 2 | find -> 6 | wc -> 7 | grep -> 8 | date -> 9 | pwd -> 10

Función countF

Esta función hace una búsqueda en el directorio en el que nos encontremos para saber cuantos ficheros se deben copiar y de esta forma incrementa este número a la variable de FICHEROS donde se lleva la cuenta de los ficheros que han sido copiados. Esta función se usa dentro de la función copy.

FUNCIONAMIENTO

Variables declaradas

Las variables declaradas son las siguientes
IFS -> “La variable de entorno IFS que significa Internal Field Separator (separador de campos internos), sirve para indicar que valor se usa como separador.”
(Iñaki Martínez ,2008)
Esta variable, básicamente se usa para que el script pueda detectar correctamente los espacios en los directorios y no haya problemas de ejecución.
ERRCODE-> En esta variable se almacena el código de error de los distintos mandatos.
FICHEROS-> Almacena la cuenta de ficheros que han sido copiados.
DIRECTORIOS-> Almacena la cuenta de Directorios que han sido creados.

Comprobación de argumentos

Lo primero en ejecutarse es la comprobación de argumentos en esta se comprueba que los datos introducidos por el usuarios son válidos para la ejecución del script. para ello se contemplan 3 posibles casos:

- **Número insuficiente de parámetros:** es decir, el usuario ha introducido menos de 3 argumentos y por lo tanto no se puede ejecutar el script, por lo que se termina la ejecución del script con exit 2.
- **Alguno de los directorios origen o destino no es un directorio existente:** es decir, que el parámetro introducido por el usuario o bien no es un directorio como tal o no corresponde con ningún directorio existente, por lo que se termina la ejecución del script con exit 2.
- **Parámetros correctos:** en este caso, se continúa la ejecución del script, en cuyo caso, se almacenarán los 2 primeros parámetros (directorios origen y destino) en sus correspondientes variables. Para obtener la ruta completa en caso de que la introducida sea una ruta relativa, se entrará en la ruta indicada y mediante el mandato pwd se obtendrá la ruta completa.

Núcleo del script

Finalmente queda el núcleo del script donde se pone en marcha toda la maquinaria del script.
Antes de realizar cualquier tipo de procesamiento, lo primero que se hace es imprimir por pantalla los datos del usuario, la fecha y hora (mandato date) y la versión del BASH.
Para la ejecución de las funciones principales del script, se utiliza un bucle FOR IN que recorre todos los argumentos (omitiendo los 2 primeros) gracias a la variable $@. Por cada iteración del bucle, se llama a la función copy con el argumento correspondiente. Este se encargará de copiar todos los ficheros con la extensión indicada en la ubicación destino.
Finalmente, en caso de que no se haya copiado ningún fichero se mostrará un mensaje por pantalla indicandolo y finalizando la ejecución con un exit 2 ya que no se copió ningún fichero. Por lo contrario, si se copió algún fichero se mostrará un mensaje con el número de directorios creados y el número de ficheros copiados y se termina el script con un “exit” que devuelve el valor contenido en ERRCODE.

Comentarios personales

Problemas encontrados

El principal problema que he encontrado es la manera de recorrer los directorios recursivamente, ya que tras muchos intentos llegué a la conclusión de que la mejor manera era utilizar un mandato que me listara todos los subdirectorios y el segundo principal problema fue la de solucionar la gestión de espacios en el nombre de los directorios. Además de mi desconocimiento sobre muchas de las opciones de los mandatos en ocasiones complicando el código con alternativas que hacen menos comprensible el código.

Crítica constructiva

Podría haber hecho un código menos lioso y quizá más fácil de comprender si hubiera modularizado mejor el código y tuviera más conocimientos sobre los mandatos en Linux.

Propuesta de mejoras

- Ya que como tal no tengo quejas sobre esta práctica ya que las propuestas del año pasado se cumplieron, quizá propondría que se deje utilizar para el examen de ejercicios una chuleta como la que proporcionais en el aula virtual, ya que como programadores (al menos en mi experiencia realizando la práctica) me han surgido en ocasiones dudas sobre qué opciones dispone un mandato o cual mandato funciona mejor en diferentes situaciones y las cuales he solventado muy rápido mirando la documentación.

Evaluación del tiempo dedicado

Estimo que para realizar toda la práctica he necesitado alrededor de 2 tardes, es decir unas 8-10 horas, es decir 5 horas a la creación del script y aproximadamente 3 a la memoria dejando el resto del tiempo para pruebas.