Departamento de Ingeniería de Sistemas Telemáticos
COMPUTACIÓN EN RED
3º curso 2016/2017

Segunda práctica

Fecha límite de entrega:
22 de Marzo de 2017

ENUNCIADO

Practica 3.2

En esta práctica los alumnos realizarán un ejercicio de consolidación y refuerzo práctico del aprendizaje en relación con los contenidos del tema 3 sobre el soporte de sistema operativo UNIX. En particular, sobre el uso interactivo de la Shell y la programación básica de comandos de Shell.

Los ejercicios que se proponen pueden realizarse en cualquier distribución o versión de sistema operativo UNIX, preferiblemente con bash como Shell de usuario. La respuesta del ejercicio es un Shell script ejecutable que se editará en un fichero de texto simple. Preferiblemente se utilizarán los editores de texto Vi o Sublime Text 3. El fichero se entregará a través de Moodle como respuesta a la actividad configurada en Moodle por los profesores de la asignatura. El trabajo se realizará preferentemente por grupos de 2 alumnos.

El fichero de respuesta incluirá una cabecera con un título y el nombre completo y DNI de los autores del informe.

Se valorará, además de la correcta ejecución funcional, la buena estructuración, legibilidad y documentación de los Shell scripts.

Ejercicio

El comando Unix `cal` muestra un calendario del mes o año especificado, por defecto el mes actual. La sintaxis simplificada del comando es

- `cal [ [mes] año]`

Es decir, si sólo se proporciona un argumento, se entiende que es el año y si se proporcionan dos, se entiende que el primero es el mes y el segundo el año. Si no se proporciona ningún argumento, el comando ofrece el calendario del mes en curso. El mes puede especificarse como un número entre uno y doce o por su nombre o cualquier abreviatura de este en inglés (Feb, february, etc.)

En este ejercicio se propone una implementación en forma de *shell script* del comando `cal`, que llamaremos `mcal`, con algunas variaciones:

La sintaxis del nuevo comando será:

- `mcal [mes] [año]`

Es decir, se puede proporcionar sólo el mes, sólo el año o ambos. El comando debe ser suficientemente "inteligente" para discernir si el valor proporcionado en un único argumento se refiere a un mes o a un año. Por ejemplo, la salida del comando `mcal 2`, debe proporcionar el calendario de febrero de 2013, que parece más adecuado que la ejecución del comando `cal 2`, que produce el calendario completo del año 2 (es improbable que esa sea la intención del usuario).

Además de la funcionalidad anterior, el comando debe tener las siguientes características:

- El comando ha de implementar una opción de ayuda (`-help`) para mostrar el uso básico del comando
- El comando ha de implementar un control básico de errores: número incorrecto de argumentos, mes o año incorrectamente especificados, etc.
- Los meses pueden especificarse mediante un número entre 1 y 12 y por su nombre o abreviatura en inglés y en español
- Documente el script con líneas de comentario para que resulte legible y fácilmente mantenible
- Añada también mediante líneas de comentario una cabecera que incluya el nombre y DNI de los autores
- Se valorará cualquier funcionalidad adicional útil y razonable

Nota orientativa

Puede ser útil extraer la fecha del sistema para determinar cuál es el mes o año en curso si el usuario no lo proporciona como argumento de `mcal`. Esto puede obtenerse mediante el comando `date`, que proporciona la fecha y hora actuales.

También es útil conocer que el siguiente comando:

- `set string1 string2 string3 …`

Asigna las cadenas *string1*, *string2*, *string3*, …, respectivamente, a los parámetros posicionales `$1`, `$2`, `$3`, …

Con todo ello, una posible forma de manejar como argumentos del Shell script los componentes de fecha, tal y como se obtiene del sistema, puede ser:

- `set \`date\``

Ejercicio resuelto

Se incluye aquí a modo de ejemplo la especificación de la funcionalidad de un Shell script y su solución.

Estudie el ejemplo básico de Shell script de la transparencia número 119 a través de las páginas de manual de cada uno de los comandos utilizados en el ejemplo y su ejecución manual por separado y en combinación. Siga las instrucciones que se enumeran a continuación:

1. Edite un fichero de nombre *gordos* con el contenido que se muestra en el ejemplo de la transparencia. Guárdelo, añádale permiso de ejecución y ejecute el comando de las siguientes formas
   a. `sh –x gordos $HOME`
   b. `gordos $HOME`
2. Complete el Shell script para que reúna las siguientes características
   a. Sintaxis: `gordos [-n k] [DIR]`. Muestra un listado de los k subdirectorios más ocupados por debajo del directorio DIR. Si no se proporciona DIR, el directorio de referencia será el directorio HOME del usuario que invoca el script. Si no se utiliza la opción –n, se listan los 10 subdirectorios más ocupados
   b. Implemente un control básico de errores: El script no puede tener más de una opción (-n k) ni más de un argumento (DIR) y si se le proporciona uno ha de ser un directorio existente.
   c. Si el comando se utiliza mal, se imprimirá un mensaje de error a través de la salida estándar de error informando sobre el error concreto: número de argumentos incorrecto, directorio no existe, el fichero no es un directorio, uso de una opción no contemplada, k no es un número en el rango contemplado, etc.
   d. El script debe devolver al entorno un valor indicativo del éxito o error de ejecución
   e. Documente el script con líneas de comentario para que resulte legible y fácilmente mantenible

```
#!/bin/bash
# gordos [-n k] [DIR]
# Genera un listado con los k subdirectorios más voluminosos bajo el  directorio
DIR
# Por defecto lista 10 directorios bajo el directorio $HOME

#Codigos de error para exit
E_NODIR=21
E_NONUM=22
E_NOOP=23
E_MASTRES=24

# Otras variables
MISCRIPT=`basename $0` # Nombre del script sin path

case $# in # Comprobamos el numero de argumentos
0) dir=$HOME; num=10;; # Comando sin argumentos
1) if [ -d $1 ] # Comando con un argumento -> DIR
   then
      dir=$1
   else
      echo "$1 no existe o no es un directorio" >&2; exit ${E_NODIR}
   fi;;
2) case $1 in # Comando con dos argumentos -> -n k
   -n)
      case $2 in
      [1-9]|[1-9][0-9]) num=$2; dir=$HOME;; # patron de numero entre 1 y 99
      *) echo "Uso: ${MISCRIPT} [-n k] [DIR], 0<k<100" >&2; exit ${E_NONUM};;
      esac;;
   *) echo "$1: opcion desconocida" >&2; exit ${E_NOOP};;
   esac;;
3) case $1 in # Comando con tres argumentos -> -n k DIR
   -n)
      case $2 in
      [1-9]|[1-9][0-9]) # patron de numero entre 1 y 99
      num=$2;
      if [ -d $3 ] # Si existe y es un directorio
      then
         dir=$3
      else
         echo "$3 no existe o no es un directorio" >&2; exit ${E_NODIR}
      fi;;
      *) echo "Uso: ${MISCRIPT} [-n k] [DIR], 0<k<100" >&2; exit ${E_NONUM};;
      esac;;
   *) echo "$1: opcion desconocida" >&2; exit ${E_NOOP};;
   esac;;
*) # Mas de tres argumentos
   echo "Uso: ${MISCRIPT} [-n k] [DIR], 0<k<100" >&2; exit ${E_MASTRES};;
esac

du $dir | sort -n | tail -$num
exit $? # Devuelve el valor de retorno del ultimo comando ejecutado
```