Fundamentos del Software. Ejercicios Parte 2 – Prácticas 6 a la 9 [10 puntos]

Ejercicio 1. [2 puntos] Escriba un guion que acepte el nombre de un directorio como argumento, haga un recorrido por los archivos que contenga y muestre la contabilidad siguiente:

- Número de archivos totales que contiene.
- Número de archivos sin permisos de ejecución que contiene.
- Número de directorios que contiene.

Debe realizar la correspondiente comprobación del argumento empleando para ello una función. Deberá comprobar que se aporta dicho argumento, se trata de un directorio y posee permisos de lectura.

Ejercicio 1. [4 puntos] Construya un guion en bash para gestionar un archivo que contiene los datos del inventario de cartuchos de tinta. El archivo que contiene los datos dispone de una línea de datos por cada artículo y cada línea dispone de cuatro campos separados uno del otro por un tabulador organizado de la siguiente forma:

`<NombreCartucho><tab><CódigoArtículo><tab><NumExistencias><tab><Ubicación>`

- El primer campo contiene el nombre del cartucho.
- El segundo campo contiene el código del artículo.
- El tercer campo es un número que representa la cantidad de cartuchos existentes de ese artículo.
- El cuarto y último campo contiene una clave de la ubicación indicada por cuatro caracteres alfanuméricos, donde la letra `E` representa el número de estantería y la `P` el número de apartado.

Un ejemplo de archivo de cartuchos de tinta sería:

```
$ cat stockImpres
EPS200cCyan E2cc  13    E1A1
EPS200cBlue E2cb  20    E1A2
EPS200cBla1 E2b1  12    E1A3
EPS200cBla2 E2b2  09    E1A3
…
```

El guion solicitado deberá realizar lo siguiente:

(a) **[0.5]** El guion admitirá como argumento un nombre de archivo que se corresponderá con el archivo cartuchos que deseamos usar (en nuestro caso será `stockImpres`). Cuando se ejecute el mismo, deberá comprobar que dicho archivo es de texto (o regular) y existe en el directorio en el que nos encontramos. En caso contrario, deberá informar con un mensaje de error y regresar al indicador del Shell.

(b) **[0.5]** Mostrar un menú con tres opciones con el siguiente aspecto:

```
Seleccione opción:
1) Localizar cartucho
2) Mostrar cartuchos de estantería o apartado
3) Salir
Elija la opción deseada:
```

(c) **[1.0]** La opción 1, debe solicitar por teclado un artículo a través de su nombre o su código y deberá imprimir las existencias disponibles del mismo (por ejemplo, si se introduce `Cyan`, deberá imprimir `13`).

(d) **[1.0]** La opción 2, debe solicitar por teclado la estantería o apartado de la misma y mostrar todos aquellos artículos ubicados allí (por ejemplo, si se introduce `A3`, debería imprimir `EPS200cBla1` y `EPS200cBla2`).

(e) **[0.5]** La opción 3, debe permitir salir del guion y volver al indicador del Shell. Mientras no se pulse este valor, siempre se mostraría el menú.

(f) **[0.5]** Crear una función denominada `_CompruebaArgumentos` para comprobar la validez del argumento aportado e incluir en ella las órdenes realizadas para dar solución al apartado (a).

Fundamentos del Software. Ejercicios Parte 2 – Prácticas 6 a la 9 [10 puntos]

El guion se considerará nulo si en la ejecución del mismo aparece algún error sintáctico, con independencia de que pudiera funcionar alguno de sus apartados.

Ejercicio 3. [2 puntos] Crear un archivo `makefile` con los archivos fuente `funciones.hpp`, `funciones.cpp` y `main.cpp` que formar parte del archivo comprimido con los siguientes requisitos:

(a) **[0.5]** Debe crearse el ejecutable denominado `invertir` a partir de los archivos objeto compilados `funciones.o` y `main.o`. Cada archivo fuente `.cpp` debe compilarse para obtener el objeto correspondiente. Esta sería la opción por defecto del archivo makefile.

(b) **[0.5]** Debe compilar cada archivo fuente con la opción de depuración `-g`.

(c) **[0.5]** Si se modifica sólo uno de los archivos fuentes, la ejecución de la orden `make` debe compilar únicamente el archivo fuente modificado. Del mismo modo, si se modifica el archivo `funciones.hpp` deberá también compilar aquellos archivos fuentes donde esté incluido.

(d) **[0.5]** Crear una opción llamada "tocar" que haga la función de modificar todos los archivos fuentes y otra llamada "limpiar" que borrar los archivos objeto y el ejecutable.

El archivo makefile se considerará nulo si en la ejecución de la orden make aparece algún error sintáctico, con independencia de que pudiera funcionar en alguno de sus apartados.

2. [2 puntos] Una vez generado el archivo ejecutable con la opción de depuración, la ejecución del mismo muestra un error de un valor incorrecto en la visualización de los datos. Realizar la depuración del código en los siguientes apartados.

Cuestionario:

(a) **[0.5]** Indicar de qué forma se situaría un punto de ruptura en la línea `39` del archivo `funciones.cpp`.

(b) **[0.5]** Indicar qué orden es necesaria para realizar la ejecución del programa introduciendo los valores 1, 2, 3, 4, 5, 6, 7, 8, 9 y 10 y, cada vez que se detenga en el punto de ruptura indicado en el apartado anterior, ¿qué orden habría que indicar para mostrar en cada iteración el valor de la variable "`i+1`" con la idea de identificar si el acceso al vector se hace fuera del rango permitido?

(c) **[0.5]** ¿En qué momento de la ejecución del programa se detecta la impresión por pantalla de un dato incorrecto?

(d) **[0.5]** ¿Cómo se corrige el error que presenta este código?