Ejercitación – Serie Q

Archivos

Conceptos que se agregan en esta serie: Manejo de archivos en bajo nivel (no buffereados).

Está prohibido el uso de variables globales.

Se deben implementar las aplicaciones y funciones complementarias que permita visualizar el resultado de las funciones realizadas.

Ejercicio 1:

1.a.-

Implemente una función que agregue información al inicio de un archivo (manteniendo la información original).

El prototipo de la función es el siguiente:

```
int agregarInicio (const char *nombre, const void *data, int size);
```

En donde:

- **`nombre`**: Puntero al nombre del archivo
- **`data`**: puntero a la información a agregar al inicio del archivo
- **`size`**: tamaño del bloque **`data`**.

La función retorna la cantidad de bytes escritos en el archivo, o un número negativo en caso de error.

1.b.-

Agregar a lo realizado en **1.a.** los siguientes condicionales:

- Verificar que los parámetros de ingreso sean válidos:
- Si el archivo no existe:
  - Este debe ser creado.
  - Si no es posible su creación, debe retornar el valor ERR_AI_CRE (-4).
- Si el archivo existe
  - Este debe ser modificado.
  - Si no es posible su modificación, debe retornar el valor ERR_AI_MOD (-2).
- Si se produce un error de escritura o cierre, durante la manipulación del archivo, debe retornar el valor ERR_AI_WT (-3).
- Cualquier otro error que pudiese surgir, debe retornar el valor ERR_AI_UNKOWN (-1).

1.c.-

Modificar **1.b.** de manera qué si se debe modificar el archivo, se utilice un archivo temporal.

Una vez finalizado el proceso, se deben eliminar los archivos temporales.

Agregar los tipos de error que considere necesario.

Ejercicio 2:

Implemente una función que copie un archivo en otro, pero a la inversa, es decir que el último byte del archivo original, sea el primer byte del nuevo archivo.

El prototipo de la función es el siguiente:

```
int invertirArchivo (const char *fileIn, const char *fileOut);
```

En donde:

- **`fileIn:`** Puntero al nombre del archivo origen
- **`fileOut:`** Puntero al nombre del archivo destino a crear

Casos de error a contemplar:

- **`-1:`** Archivo origen no existe o no puede ser abierto
- **`-2:`** Archivo destino ya existe
- **`-3`**: No es posible crear el archivo destino
- **`-4`**: Error durante el proceso de lectura del archivo origen
- **`-5`**: Error durante el proceso de escritura/cierre del archivo destino
- **`-6`**: otros errores

Opciones de implementación:

- A: Sin usar memoria dinámica.
- B: Utilizando memoria dinámica.
  Consulta: *como haría si el archivo es más grande que el mayor bloque de memoria que puede obtener*.

Ejercicio 3: Código EAN 13

Como podrá advertir, hay una serie de posibles errores que se pueden generar en el siguiente ejercicio. `

Se solicita analizar estos posibles errores y generar una lista de estos asignándoles valores negativos mediante una enumeración (**`enum`**).

Incluir una función que convierta ese número de error en texto (similar a **`errno`** y **`strerror`**)

3.a.-

Realizar una aplicación que genere un archivo de texto, en donde cada línea del archivo este compuesta por 12 dígitos numéricos.

La cantidad de líneas a generar y nombre de archivo a crear se reciben por argumento del main.

Ninguna de las líneas debe iniciar con el número 0 (cero)

3.b.-

Se cuenta con un archivo de texto, en donde cada línea posee un código EAN13, pero sin el dígito verificador.

El código EAN 13 es un estándar internacional que está compuesto por 12 dígitos numéricos que identifican a un determinado producto, más un 13avo dígito que es el dígito verificador.

Proceso para determinar el dígito verificador (EAN13).

Supongamos el siguiente número

Posición:	0	1	2	3	4	5	6	7	8	9	10	11	DC
Número:	8	4	1	2	5	8	4	5	1	2	5	4	?

1. Sumamos todos los dígitos que ocupan las posiciones pares: **8+1+5+4+1+5 = 24** (pares)
2. Sumamos todos los dígitos que ocupan las posiciones impares: **4+2+8+5+2+4 = 25** (impares)
3. Multiplicamos por 3 el valor obtenido en la suma de los dígitos impares: **25*3 = 75**
4. Sumamos al valor obtenido anteriormente, la suma de los números pares: **24+ 75 = 99**
5. Redondeamos el valor obtenido a la decena inmediatamente superior, en este caso **100**
6. El dígito de control es el valor obtenido del redondeo de decenas menos la suma total del punto 4: **100 – 99 = 1**

- Implemente una función que le agregue el dígito verificador a cada una de estas líneas
- Genere una aplicación que agregue a un archivo EAN13 sin dígito verificador, el dígito verificador correspondiente y lo almacene en un nuevo archivo.

3.c.-

Se cuenta con un archivo de texto, en donde cada línea posee un código EAN13, con el dígito verificador.

Realizar una aplicación que reciba el nombre del archivo y verifique que todos los códigos están bien conformados (se entiende por verificar que el número esta conformados por 13 dígitos y que el dígito verificador es correcto).

Utilice como referencia los archivos:

- **`List-ean01.txt`** – todos los dígitos están correctos
- **`List-ean02.txt`** – las líneas 3,7,9 y 11 del total de 12 líneas poseen error
- **`List-ean03.txt`** – las líneas 2,5 y 11 del total de 12 líneas poseen error

La primera línea del archivo la enumeramos con 1 (uno)