DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN
ISIS1304
FUNDAMENTOS DE INFRAESTRUCTURA TECNOLÓGICA

Proyecto 1 - 2020-1

Este proyecto vale 10% de la nota del curso.
Debe ser elaborado en grupo (3 integrantes).
No se permite ningún tipo de consulta entre grupos.
Se debe entregar por Sicua+ a más tardar el 26 de abril de 2020

A. OBJETIVOS

- Practicar el lenguaje C y desarrollar un programa de complejidad pequeña.
- Conocer las operaciones de C para el manejo de bits.
- Aplicar lo anterior en un traductor de octal a binario.

B. DESCRIPCIÓN DEL PROBLEMA

El objetivo de este proyecto es desarrollar un traductor de octal a binario. Este traductor permitirá leer un archivo de texto compuesto únicamente por dígitos octales y guardar su traducción en un archivo binario (de un tipo cualquiera).

Es parecido a lo que ocurre cuando teclea en el editor hexa, pero aquí trabajaremos base 8.

La forma en que se hará la traducción será explicada en la sección de codificación.

Codificación

El sistema de numeración octal es un sistema de numeración en base 8. Como es una base que es potencia exacta de 2 ($8=2^3$), la traducción de cualquier dígito se puede representar con 3 bits. De esta manera, se espera de su programa que, a partir del archivo inicial, traduzca cada dígito octal a binario, representando cada uno con tres bits, para después reportar su valor en un nuevo archivo. A continuación se muestra un ejemplo. Se tendrá la siguiente cadena de **caracteres** representando el mensaje:

Archivo de entrada (un archivo de texto pero solo puede contener caracteres entre '0' y '7'):

2026005

Notese que en el archivo de entrada se representa cada dígito con su ASCII respectivo, es decir: ‘2’,’0’, ‘2’,’0’, ‘4’, ‘0’, ‘5’.

Es decir, en un editor hexa este caso en particular se vería representado como:

32 30 32 30 34 30 35

El programa debe hacer la traducción a binario de cada dígito octal (se separa cada octal con “-”):

010 - 000 - 010 - 000 - 100 - 000 - 101

Lo cual agrupado en bytes es:

0100 0001	0000 1000	0010 1**000**

Note que, como se explica más adelante, fue necesario completar el último byte agregando tres ceros (**000**)

En un editor hexa este ejemplo en particular se vería representado como:

41 08 28

Note que el archivo de entrada tiene 7 bytes, pero el de salida tiene 3 porque convertimos caracteres que representan dígitos octales (de 8 bits) a verdaderos valores octales de 3 bits.

Formato del archivo

En este ejercicio los archivos de entrada solo contendrán caracteres del '0' al '7' que representan los dígitos octales. Sin embargo, el archivo de salida será un archivo binario, así que no necesariamente contendrá caracteres ASCII.

Restricciones y consideraciones adicionales

El programa debe tener en tener en cuenta y controlar que si el byte al final del archivo queda incompleto, se deberá completar los dígitos faltantes con 0 en los bits (razón por la cual fue necesario agregar **000** en el ejemplo dado).

El programa

En el archivo adjunto ("`main.c`"), encuentra el esquema del programa. El programa se invoca por línea de comando.

El programa ya tiene definidos los procedimientos `main (), readFile (Datos *, char *) y writeFile (int, Datos * archivoEnBinario, char *).` El procedimiento `main` se encarga de preguntar qué archivo se desea traducir, cargar sus datos en memoria principal y preguntar el nombre del archivo para guardar la respuesta. Por otro lado, la función `readFile` se encarga de abrir un archivo y leerlo en el vector de datos y `writeFile` se encarga de escribir un archivo a partir del vector de datos. Estas funciones ya se encuentran desarrolladas, por lo cual es muy importante que **no sean modificadas.**

- Para traducir el archivo, el procedimiento `main` recibe por parámetro la ruta completa del archivo de origen que se da como argumento del comando, así como la ruta completa del archivo de destino.
- El procedimiento de lectura recibe el nombre del archivo y un vector de tipo `unsigned char`; el procedimiento lee el archivo y lo carga en el vector. Retorna cuántos bytes leyó.
- El procedimiento de escritura recibe el nombre del archivo donde se va a escribir la traducción y el vector de tipo `unsigned char`, y escribe el vector de datos (ya traducido a binario) en el archivo.

El procedimiento que el grupo debe completar es el siguiente:

- **`convertirABinario (Datos * datosOct, Datos * datosBin)`**: Esta función se encarga de convertir a binario y escribir cada uno de los bytes generados a partir de la traducción del archivo. Si queda algún byte incompleto, la función

debe completar los bits faltantes con ceros (0). El parámetro **`datosOct`** es un apuntador a una estructura donde se encuentran los datos de entrada y el parámetro **`datosBin`** es un apuntador a una estructura donde se encuentran los datos de salida.

Para desarrollar el programa pueden crear las funciones adicionales que necesiten (recomendado), las cuales deben estar debidamente comentadas y documentadas.

C. ESPECIFICACIONES

- Los programas se deben escribir en C.

  Nota importante: los programas se calificarán únicamente usando el ambiente de visual de las máquinas virtuales. Si el programa no compila en este ambiente, se considerará que no corre (así compile en otros ambientes). Refierase a los videos tutoriales publicados en Sicua para compilar sus programas.

- Legibilidad del programa: indentar el programa; **escribir comentarios** explicando el código.
- Debe respetar la estructura del código entregado. En particular, debe usar los procedimientos y variables del esqueleto.

D. CONDICIONES DE ENTREGA

- Entregar el código fuente junto con el ejecutable en un archivo *.zip. **Al comienzo del archivo fuente escriba los nombres de los miembros, sus códigos y correos, de lo contrario no será evaluado (ver esqueleto)**. Si su programa no funciona o si su solución tiene particularidades, puede enviar un archivo `.docx` o `.pdf` explicando por qué cree que no funciona o qué fue lo que hizo.
- El trabajo se realiza en grupos de **3** personas. No debe haber consultas entre grupos.
- El grupo responde solidariamente por el contenido de todo el trabajo, y lo elabora conjuntamente (no es trabajo en grupo repartirse puntos o trabajos diferentes).
- Se puede solicitar una sustentación a cualquier miembro del grupo sobre cualquier parte del trabajo. Dicha sustentación puede afectar la nota de todos los miembros.
- El proyecto debe ser entregado por Sicua+ por uno solo de los integrantes.
- **Entregar por Sicua+ a más tardar el 26 de abril de 2020 hasta las 11:50 pm.**

E. CASOS DE PRUEBA

Adjuntos al proyecto encontrarán tres archivos de ejemplo con los valores de entrada y sus respectivos archivos esperados de salida. Pueden utilizar los archivos de entrada para verificar que sus programas estén funcionando correctamente, pero es altamente recomendado que generen sus propias pruebas.

F. CRITERIOS DE CALIFICACIÓN PARA LOS PROGRAMAS

La calificación consta de dos partes:

- Ejecución (50%). Para las funciones propuestas se harán 5 pruebas: los 3 casos de prueba entregados y otros 2 nuevos. Para cada caso, se revisará si la salida es correcta o no según los requerimientos establecidos en el enunciado. Cada prueba vale 10%.
- Inspección del código (50%). Se consideran tres aspectos:
  - 10% - legibilidad (nombres dicientes para variables, comentarios e indentación)
  - 20% - manejo de bits (uso de los operadores de bits de C: >>, &, etc.)
  - 20% - manejo de la estructura de datos (recorrido, manejo de los elementos).

G. RECOMENDACIONES

- Recuerden que los trabajos hechos en grupo y entregados individualmente (o en grupos diferentes al original) son una forma de fraude académico.