UNT – FACEyT - DEEC

Programación I

Año 2016

Unidad 5: Archivos en Lenguaje C
Introducción

- Un archivo es un dato estructurado dinámico.
- Un **archivo** es un conjunto de bits almacenados en un dispositivo.
- En lenguaje C, un archivo es un concepto lógico.
- El archivo es el destino (escritura) o fuente (lectura) del flujo de datos (bytes) que se están tratando.
- El destino o fuente del flujo de datos es siempre un dispositivo físico: disco, terminal, monitor, impresora, etc.

Unidad 5: Archivos en Lenguaje C
Introducción

- Hasta ahora, todos los datos que se procesaron en un programa y/o función residían en la memoria del computador.
- El almacenamiento de datos en la memoria es volátil. Los datos se pierden cuando finaliza la ejecución del programa.
- La persistencia de los datos se puede lograr con el uso de archivos.
- El flujo de información (bytes) se escribe en el dispositivo físico (disco) donde los datos se mantienen inalterables aún después de finalizada la ejecución del programa.

Unidad 5: Archivos en Lenguaje C
Introducción

- Aún trabajando con archivos, la memoria principal es el lugar donde se ejecuta el programa y residen temporalmente los datos.
- Para vincular el dispositivo físico, donde se guardarán o leerán los datos, con la memoria se debe realizar la operación de "***apertura***" del archivo (destino y/o fuente).
- Si la apertura del archivo se realizó satisfactoriamente se habilita el flujo de datos entre la memoria del computador y el dispositivo físico.

Unidad 5: Archivos en Lenguaje C
Introducción

- Las operaciones de entrada y salida de datos (E/S) se realizan por medio de funciones de la biblioteca estándar de C. Estas están definidas en el archivo cabecera o *header*: `stdio.h`
- En el tratamiento de archivos está establecido que el punto de referencia es la unidad central de proceso:
  - ✓ La entrada de datos es el proceso de lectura de datos desde un archivo.
  - ✓ La salida de datos corresponde al proceso de escritura de estos en un archivo.

Unidad 5: Archivos en Lenguaje C
La estructura FILE

El puntero a FILE

- Un puntero a FILE es la variable que utiliza el programa para crear el vínculo entre la unidad central de proceso y el dispositivo físico.
- Este puntero apunta a una estructura estándar de C que contiene la información del archivo que se vinculará con el programa.

Unidad 5: Archivos en Lenguaje C
La estructura FILE

Atributos de la estructura

Los atributos más significativos de la estructura FILE son:

- Posición de puntero de archivo
- Posición del inicio del archivo
- "***File handle***" o "***inode***": identificador de archivo de S.O. Windows / Unix
- "Flags": banderas que indican el estado del archivo (read, write, r/w).
- Tamaño del archivo en bytes.

Unidad 5: Archivos en Lenguaje C
Puntero a `FILE`

Declaración

`FILE` ***nombre_del_puntero**;

En esta declaración ***nombre_del_puntero** toma la dirección de memoria del bloque asignado para la estructura `FILE`.

Los atributos toman valores cuando se "abre" el archivo de forma exitosa.

Unidad 5: Archivos en Lenguaje C
Apertura de un archivo

Para abrir un archivo se utiliza la función

```
FILE* fopen("nombre del archivo", "modo de apertura");
```

Si la apertura se realizó de manera exitosa la función `fopen()` devuelve un puntero a la estructura `FILE` que contiene los datos del archivo físico ("nombre del archivo"). Caso contrario devuelve `NULL`.

Unidad 5: Archivos en Lenguaje C

Apertura de un archivo

nombre del archivo: cadena con el nombre de un archivo válido. Si el archivo no se encuentra en el directorio de trabajo se debe especificar el camino.

modo de apertura: Especifica las operaciones E/S que se podrán realizar sobre el archivo.

modo	Descripción
`"r"`	Sólo lectura. Si el archivo no existe `fopen()` devuelve NULL.
`"w"`	Sólo escritura. Si no existe lo crea, si existe lo sobrescribe.
`"a"`	Sólo escritura. Abre un archivo de texto para añadir . Si no existe lo crea.
`"r+"`	Lectura/escritura. Si no existe no lo crea, devuelve NULL.
`"w+"`	Lectura/escritura. Si no existe lo crea, si existe lo sobrescribe.
`"a+"`	Lectura/escritura. Abre un archivo de texto para añadir. Si no existe lo crea.

Unidad 5: Archivos en Lenguaje C

Cierre de un archivo

Así como para operar sobre el archivo se lo debe abrir. Cuando se deja de operar sobre un archivo abierto este debe "cerrarse".

La acción de cerrar un archivo es importante, sobre todo, si se realizaron modificaciones en él, puesto que con esta instrucción se actualizan los parámetros del archivo en el "sistema de archivos" (File System) del dispositivo de almacenamiento.

Unidad 5: Archivos en Lenguaje C

Cierre de un archivo

La función para cerrar archivos es:

```
fclose(FILE *);
```

Si el cierre del archivo apuntado por el puntero a `FILE` fue exitoso la función devuelve **`0`.**

Los archivos se cierran, por defecto, cuando encuentran la sentencia `return` o se finaliza la ejecución de la función donde se realizó la apertura.

Unidad 5: Archivos en Lenguaje C

Tipos básicos de archivos

Archivos de texto (acceso secuencial)

Contienen secuencias de caracteres. Para acceder a cada elemento se debe pasar por todos los anteriores. Se utilizan para almacenar texto plano, fuentes de programas, base de datos simples, etc.

Archivos binarios (acceso directo)

Los archivos binarios están compuestos por una secuencia de bytes. Se escriben y se leen bytes. No hay traducción de caracteres.

Unidad 5: Archivos en Lenguaje C

Tipos básicos de archivos

La utilización de archivos binarios o de archivos de texto depende de las necesidades o requerimientos de la aplicación.

Los archivos de texto tienen la facilidad de la lectura, es posible leerlos con cualquier editor de texto, cosa que no ocurre con los archivos binarios.

En los archivos de texto sólo se pueden escribir tipos primitivos de datos y cadenas de caracteres. Para guardar registros deben guardarse cada uno los campos (datos primitivos), que lo componen, de forma separada.

En los archivos binarios, además de los tipos de datos primitivos, también se pueden escribir datos definidos por el usuario (arreglos, registros, etc.).

Unidad 5: Archivos en Lenguaje C

Tipos básicos de archivos

La ventaja de los archivos binarios se basa en la facilidad para la búsqueda, ordenamiento, modificación , etc. Estos procesos son dificultosos en un archivo de texto.

La diferencia (binario o texto) se establece según las funciones utilizadas para la escritura.

En algunos sistemas operativos (antiguos) se debe especificar la naturaleza del archivo en la sentencia de apertura agregando al modo:

t : si es archivo de texto

b : si es archivo binario.

Por ejemplo:

"wb": binario sólo escritura

"r+t": texto para lectura / escritura.

Unidad 5: Archivos en Lenguaje C
Fin de archivo (EOF)

- Los archivos tienen internamente un puntero de posición.
- Cuando se abre un archivo este puntero se ubica al inicio.
- A medida que se escribe, o se lee, el puntero se ubica, automáticamente, en la posición inmediata siguiente.
- Todo archivo tiene al final una "marca de fin de archivo" que se conoce como **EOF**(End Of File) . Esta marca se establece cuando se cierra un archivo.

Unidad 5: Archivos en Lenguaje C
Fin de archivo (EOF)

- Cuando se lee en un archivo, este proceso se debe realizar, desde el inicio, hasta que se encuentre la marca de **EOF**.
- Si el proceso de lectura de un archivo se realiza luego de la marca de "fin de archivo" (**EOF**) se producirá un error.
- La función, en C, para detectar el fin de archivo es:

```
feof(FILE*);
```

Unidad 5: Archivos en Lenguaje C

Funciones para escritura en un archivo de texto

`fprintf`(`FILE *`, "<cadena de control>" [,<lista de argumentos>]);

Escribe la "cadena de control" con los valores de los argumentos indicados en el destino apuntado por el puntero a `FILE`.

- Su uso es similar al de la función `printf()`. Sólo que tiene 3 argumentos
- El primer argumento especifica el "*flujo*" o "*stream*" de salida apuntado `FILE*`. Si la salida es el monitor, el flujo estándar definido en C es **`stdout`**

Unidad 5: Archivos en Lenguaje C

Funciones para escritura en un archivo de texto

```
fputs(char *,FILE *);
```

Lee la cadena de caracteres que está en la dirección de memoria del bloque apuntado por `char` y la escribe en el destino apuntado por `FILE*`

```
fputc( char,FILE *);
```

Lee el caracter guardado en la dirección de memoria de la variable identificada por char y lo escribe en el destino apuntado por `FILE*`

Unidad 5: Archivos en Lenguaje C

Apertura de archivos verificando el modo

Si se ejecuta la sentencia de abrir un archivo con el modo `"r"` o `"r+"` y el archivo no existe, la apertura no se realiza en forma exitosa. El puntero a `FILE` toma el valor `NULL`, pero no hay una notificación que indique la anomalía o que el archivo no existe.

Para asegurar que la apertura/creación de un archivo sea exitosa se puede utilizar el siguiente conjunto de sentencias:

```
ptrFILE = fopen( nombre, "r+");
       if (ptrFILE == NULL) {
         printf("\n El archivo no existe se creara ");
         ptrFILE = fopen( nombre, "w+");
         if (ptrFILE == NULL)              {
              printf("\n El archivo no se pudo crear ");
              exit(1);
         }
       }
```

Unidad 5: Archivos en Lenguaje C

Posicionamiento del puntero interno de archivo

- La función:

```
ftell( FILE *);
```

devuelve un valor (`long int`), en bytes, que indica la posición actual del puntero interno del archivo respecto al inicio del mismo.

- Si se finalizó la escritura sobre el archivo y se quieren leer los datos guardados hay que tener en cuenta que el puntero de posición se encuentra al final.
- Para posicionar el puntero interno de un archivo al inicio de este se utiliza la función:

```
rewind(FILE*);
```

Unidad 5: Archivos en Lenguaje C

Posicionamiento del puntero interno de archivo

Otra función para posicionar el puntero interno es:

```
fseek( FILE * , long int offset, int origen);
```

donde:

offset: desplazamiento (en bytes) del puntero de posición respecto al origen tomado como referencia.

origen: referencia desde donde se realiza el desplazamiento:

- **SEEK_SET** ó **0** : inicio del archivo
- **SEEK_CUR** ó **1** : posición actual
- **SEEK_END** ó **2** : fin del archivo

Unidad 5: Archivos en Lenguaje C

Funciones para lectura de un archivo de texto

`fscanf``( FILE *, "<cadena de control>", <dirección>);`

Lee en el archivo (*"flujo"* o *"stream"*) apuntado por `FILE` el dato descripto por el especificador de datos contenido en la cadena de control y lo guarda en la dirección de memoria indicada.

- Su uso es similar al de la función `scanf()`, sólo que tiene 3 argumentos
- El primer argumento especifica el "*flujo"* o "*stream"* de entrada apuntado `FILE*`. Si la entrada es el teclado, el flujo estándar definido en C es **`stdin`**

Unidad 5: Archivos en Lenguaje C

Funciones para lectura de un archivo de texto

`fgets``(char *`***`cad`***`,int `***`n`***`,FILE*);`

Lee del archivo (*"flujo"* o *"stream"*) apuntado por `FILE` una cadena de caracteres de ***n*** bytes (o caracteres) y la guarda en el bloque de memoria de la variable ***cad*** .

`char` **`fgetc`**`(FILE*);`

Devuelve el caracter que lee en el archivo apuntado por `FILE.`

Unidad 5: Archivos en Lenguaje C

Ejemplos de tratamiento de archivos de texto

1. Lectura de archivos de texto utilizando las funciones `fgetc(…)` y `fgets(…)`
   (Los caractres válidos corresponden al juego 1 de ASCII. Por esta razón los caractres con acento y las ñ aparecen como símbolos)

   Unidad-5-Ej-11.c

2. Lectura de archivos de texto utilizando las funciones `fgetc(…)`, `fgets(…)`y `fscanf(…)`

   Unidad-5-Ej-12.c

3. Cambiar a mayúscula la primera letra de cada palabra.

   Unidad-5-Ej-13.c

4. Reemplazar un signo de puntuación por salto de línea

   Unidad-5-Ej-14.c

Unidad 5: Archivos en Lenguaje C
Ejemplos tratamiento de archivos de texto

5. Copia de archivos — Unidad-5-Ej-15.c
6. Ingresar el nombre del archivo en la línea de órdenes — Unidad-5-Ej-16.c
7. Cargar datos en un archivo de texto
   (Se utilizan funciones de registros) — Unidad-5-Ej-17.c
8. Conociendo que, en un archivo de texto, cada línea guarda lo siguiente: — Unidad-5-Ej-18.c

   Apellido;Nombre;CX;Edad

   El programa permite: a) recuperar el contenido del archivo y presentarlo por pantalla, b) agregar contenido al archivo respetando el formato y c) usar registros para recuperar los datos y para la lectura de nuevas líneas a ser almacenadas.

Unidad 5: Archivos en Lenguaje C

Funciones útiles para el tratamiento de archivos

Existe, en lenguaje C, una definición de tipo de dato:

`fpos_t`

que se utiliza para especificar la posición del puntero interno de un archivo o fichero.

Según el compilador puede estar implementado por:

- **`long int`**, ó
- **`unsigned long int`**

Hay funciones para el tratamiento de archivos que utilizan sólo este tipo de dato para especificar la posición dentro del mismo.

Unidad 5: Archivos en Lenguaje C

Funciones útiles para el tratamiento de archivos

Dos funciones que utilizan como argumentos el tipo de dato **`fpos_t`** son:

- **`int fgetpos`** `( FILE*, &posicion);`

  guarda el valor actual del indicador de posición del archivo en la variable posicion. Esta es del tipo **`fpos_t.`** Si la operación se realizó con éxito devuelve `0.`

- **`int fsetpos`** `( FILE*, &posicion);`

  ubica al indicador de posición del archivo en el valor de la variable posicion. Esta es del tipo **`fpos_t`**. Si la operación se realizó con éxito devuelve `0.`

Unidad 5: Archivos en Lenguaje C
Archivos binarios (de acceso directo)

Un archivo binario, es un archivo cuyo contenido usa exactamente la misma representación que utiliza el computador internamente para representar la información. Se escribe en el dispositivo externo de la misma forma que la información se escribe en memoria.

El contenido de un archivo binario no está pensado para ser entendible por humanos. Si es abierto con un editor de texto, generalmente, se verán sólo símbolos ininteligibles.

Unidad 5: Archivos en Lenguaje C
Archivos binarios (de acceso directo)

Las operaciones de apertura y cierre para archivos binarios son exactamente las mismas que las vistas para archivos de texto se efectúan con las funciones :

```
FILE* fopen ("nombre del archivo", "modo de apertura");

int fclose(FILE *);
```

Los modos de apertura que se utilizan son:

`r , w , r+ y w+`

No se utiliza `a` ni `a+` por ser exclusivamente para archivos de texto.

Unidad 5: Archivos en Lenguaje C
Archivos binarios (de acceso directo)

El cambio fundamental en el tratamiento de archivos binarios respecto a los de texto radica en las funciones que se utilizan para lectura y escritura. Estas son:

`fwrite(&var, size, n, FILE *);`

escribe en el flujo especificado por el puntero a `FILE`, **n** datos del tamaño **size** que están guardados en la dirección **&var.**

`fread(&var, size, n, FILE *);`

lee en el flujo apuntado por `FILE` **n** datos de tamaño **size** y los guarda en la dirección de memoria de **var.**

Unidad-5-Ej-20.c

Unidad-5-Ej-21.c

Unidad-5-Ej-22.c