E/S con archivos

Cecilia Manzino

11/11/2016

Biblioteca de E/S estándar <stdio.h>

- Pasos a seguir para usar un archivo:

  1. Abrirlo.
  2. Utilizarlo
  3. Cerrarlo

- Para abrir un archivo:

```
FILE *fopen(const char *nombre, const char *modo);
```

  - valor de retorno: puntero a estructura que contiene información acerca del archivo.
  - nombre: nombre del archivo
  - modo: tipo de acceso ("r", "w", "a", etc).

Ejemplo fopen

```
FILE *file;
file = fopen("archivo.dat","r");
```

Es una buena práctica chequear si se pudo abrir el archivo:

```
if ( (file = fopen("archivo.txt","r")) == NULL )
{
        printf("No se pudo abrir %s\n","archivo.txt");
        exit(1);
}
```

Modos de acceso

"r" : (read) Para abrir un archivo existente.Lee desde el comienzo, si no existe da error.

"w" : (write) Para abrir un archivo para escritura. Si no existe lo crea, si existe elimina su contenido.

"a" : (append) Para abrir un archivo para escritura. Si no existe lo crea, sino escribe desde el final del mismo.

"r+" : Para abrir un archivo para lectura/escritura. Lee desde el comienzo, si no existe da error.

"w+" : Para abrir un archivo para lectura/escritura. Si no existe lo crea, si existe elimina su contenido.

"a+" : Para abrir un archivo para lectura/escritura. Si no existe lo crea, sino escribe desde el final del mismo.

Opcionalmente la bandera "b" es usada sólo en windows para abrir un archivo en modo binario.

funciones `fgetc` y `fputc`

Las funciones:

```
int fgetc( FILE *file);
int fputc( int ch, FILE *file );
```

permiten leer y escribir un carácter desde/hacia un archivo de texto.

- Ambas retornan el carácter leído/escrito o `EOF` si hubo un error.
- Notar que el caracter devuelto es de tipo `int` en lugar de `char`, esto es necesario para devolver `EOF` de tipo `int`.

Ejemplo

```c
#include <stdio.h>

int main()
{
    FILE* file;

    if ( (file = fopen("archivo.txt","r")) == NULL ){
        printf("No se pudo abrir %s\n","modos.txt");
        exit(1); // termina la ejecuci'on del programa
    }

    int c;
    while ((c = fgetc(file)) != EOF)
        putchar(c);

    if (ferror(file))
        puts("Error al leer el archivo");
    else if (feof(file))
        puts("El archivo se ley\'o exitosamente");

    fclose(file);
}
```

Funciones `fgets` y `fputs`

Permiten leer y escribir líneas (cadenas) desde/hacia un archivo de texto. Sintaxis:

```
int fputs( const char *string, FILE *file);
```

Retorna un valor no negativo si no falla, y EOF si hubo un error. Además setea el tipo de error.

```
char *fgets(char *linea, int maxlinea, FILE *file)
```

- Lee hasta `maxline-1` caracteres incluyendo el fin de línea.
- Retorna la string `linea`, o NULL si hay un error.
- Si el error fue causado por una condición end-of-file, setea el indicador eof de manera que feof() debería retorna un valor distinto de 0.
- Si hubo otro tipo de error, setea el indicador error.

Ejemplo fputs

```
int main()
{
    FILE* file;

    if ( (file = fopen("caperucita.txt","a")) == NULL )
        {
         printf("No se pudo abrir %s\n", "caperucita.txt
             ");
         exit(1);
    }

    int rc = fputs("Hola Mundo", file);

    if (rc == EOF)
        perror("Error en fputs()");

    fclose(file);
}
```

Ejemplo fgets

```
main () {
     FILE *file;
     char linea[100];

     file= fopen("elLobo.txt", "r");


     if ( file==NULL){
          printf("No se pudo abrir el archivo");
exit(1);
     }

      while ((fgets(linea, 100, file)) != NULL)
printf("%s", linea);

      if (feof(file))
        puts("\n Se alcanz'o el fin de archivo");

      fclose(file);
}
```

Más funciones

Para la E/S de archivos con formato también se pueden usar:

```
int fprintf(FILE *file, char *formato, ...)
int fscanf(FILE *file, char *formato, ...)
```

Se comportan como `printf` y `scanf` excepto que el primer argumento es un puntero a archivo que especifica el archivo donde se escribirá o leerá.

Práctica

1. Definir una función
   `void filecopy(FILE *origen, FILE *destino)` que copie el archivo `origen` en el archivo `destino`. Dar dos versiones, una usando `fgetc` y otro `fgets`.

2. Escribir un programa que compare dos archivos e imprima la primer línea en donde difieran.