Programación II

Archivos en C

Lic. Federico Severino Guimpel
LCC - FCEIA - UNR

Archivos

En C, al igual que en Python se pueden manipular archivos. La forma de hacerlo es similar. En esta presentación vamos a ver cómo hacerlo.

Archivos

En C, se usa la función `fopen` para crear un archivo nuevo o abrir uno existente. Esta función, retorna un objeto del tipo FILE. Este contiene toda la información necesaria para controlar la entrada/salida del archivo.

El prototipo de esta llamada a función es:

```
FILE *fopen( const char * filename, const char * mode );
```

Archivos

El primer argumento es, como podemos imaginar, el nombre del archivo (con el path si así se requiere).

El segundo argumento es el modo de apertura. Los mismos son similares a los que vimos en Python.

Modo	Descripción
r	Abre un archivo existente para lectura
w	Abre un archivo para escritura. Si no existe, un nuevo archivo es creado. Se comienza a escribir desde el inicio del archivo, pisando la información existente, si es que tenía alguna.
a	Abre un archivo para escritura en formato append. Si no existe, un nuevo archivo es creado. Aquí el programa comienza a escribir desde el final del contenido que el archivo tenga, si es que tiene alguno.
r+	Abre un archivo para lectura y escritura.
w+	Abre un archivo para lectura y escritura. Si el archivo existe entonces pisa todo su contenido y, si no existe lo crea.
a+	Abre un archivo para lectura y escritura. Crea el archivo si no existe. La lectura va a comenzar del comienzo del archivo pero, la escritura puede ser sólo en formato append, es decir, al final del archivo.

Para cerrar un archivo se usa, al igual que en Pyhton, la función `close`. Su comportamiento es similar:

```
int fclose( FILE *fp );
```

La función `close` retorna cero si pudo cerrarse adecuadamente o EOF si existe un error al tratar de cerrar el archivo. Esta función es la que, al llamarse, vuelca el buffer con todos los datos pendientes al archivo, lo cierra y, libera la memoria usada por el archivo. La constante EOF es definida en la librería stdio.h

Para escribir en un archivo tenemos tres opciones. La primera consiste es escribir caracter a caracter usando la función `fputc`, la función `fputs` que imprime un string y, finalmente la función `fprintf`. Vamos a ver cada una de ellas.

La función `fputc` escribe el caracter representado por el entero c al archivo al que referencia fp. Esta retorna EOF si la función tuvo algún error y, el entero que se quiso escribir si pudo realizar la operación.

```
int fputc( int c, FILE *fp );
```

Archivos

La función `fputs` escribe el string s al archivo al que referencia fp. Esta retorna EOF si la función tuvo algún error y, un número negativo si pudo realizar la operación.

```
int fputs( const char *s, FILE *fp );
```

La función `fprintf` tiene el mismo comportamiento que la función `printf`. Sólo cambia que tiene un argumento más que es el archivo en el que se va a escribir.

```
int fprintf(FILE *fp,const char *format, ...)
```

Vamos a ver un ejemplo del uso de estas funciones:

```c
#include <stdio.h>

int main() {
    FILE* fp;
    fp = fopen("test.txt", "w+");
    fprintf(fp, "Esta es una pruueba para fprintf...\n");
    fputs("Esta es una prueba para fprint...\n", fp);
    fclose(fp);
}
```

Para leer también contamos con tres opciones, cada una de ellas es la contrapartida de una función para escribir: `fgetc`, `fgets` y `fscanf`.

La función `fgetc` lee de a un caracter. Esta retorna EOF si la función tuvo algún error y sino, el caracter leído.

```
int fgetc( FILE * fp );
```

La función `fgets` lee hasta n−1 caracteres del el archivo referenciado por fp. Este copia el string leido en el array buf, agregando un caracter '\0' al final.

Si, durante la lectura, se encuentra un salto de línea, un '\n' o el final del archivo entonces, se leen la máxima cantidad de caracteres y, se retornan sólo los caracteres leídos hasta ese punto incluyendo el caracter de salto de línea.

```
char *fgets( char *buf, int n, FILE *fp );
```

La función `fscanf` es similar a la función `scanf`. Debemos recordar que esta función lee hasta encontrar un separador el cual puede ser un espacio en blanco, una tabulación, un salto de línea o el final del archivo.

```
int fscanf(FILE *fp, const char *format, ...)
```

Archivos

Un ejemplo del uso de estas funciones sería:

```c
#include <stdio.h>

int main() {
    FILE* fp;
    char buff[255];
    fp = fopen("test.txt", "r");
    fscanf(fp, "%s", buff);
    printf("1 : %s\n", buff );
    fgets(buff, 255, fp);
    printf("2: %s\n", buff );
    fgets(buff, 255, fp);
    printf("3: %s\n", buff );
    fclose(fp);
    return 1;
}
```