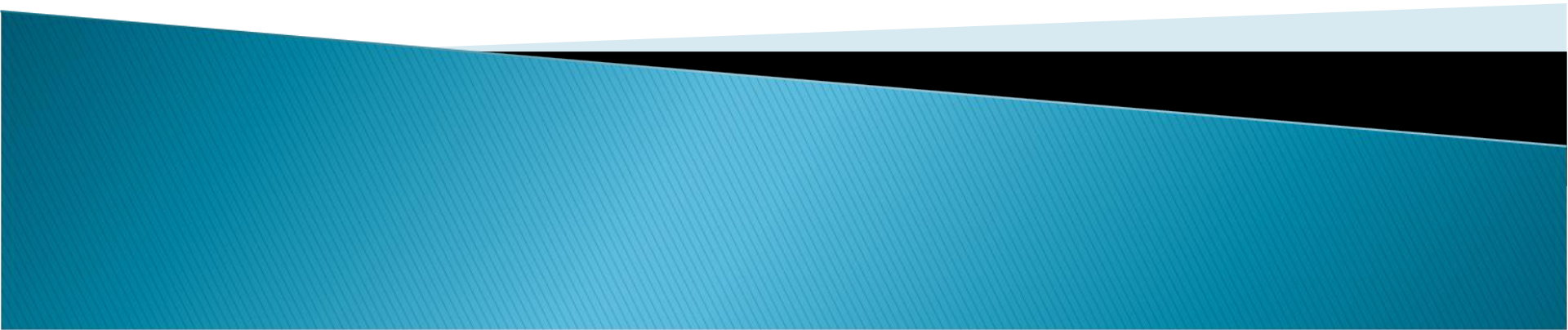
Archivos

M.C. Yolanda Moyao Martínez



Archivo

- **Cadena de bytes consecutivos terminada por una carácter especial llamado EOF (End Of File)**

Byte 1 Byte 2 Byte 3 ..... Byte n EOF

- **Desde el punto de vista del programador, los archivos hay que verlos como cadenas de bits (O o 1) agrupados en bytes consecutivos**

Tipos de archivos

- **Archivos de texto**
- **Archivos binarios**

Archivo de texto

- Secuencia de caracteres organizadas en líneas terminadas por un carácter de nueva línea.

- Todas las letras tienen el mismo formato y no hay palabras subrayadas, en negrita, o letras de distinto tamaño o ancho.

'H' 'O' 'L' 'A' ..... EOF

Archivo binario

- Contiene códigos y caracteres, los cuales sólo pueden ser utilizados para un tipo específico de software

- Por ejemplo Fotografías, imágenes, texto con formatos, archivos ejecutables (aplicaciones), etc.

Funciones de STDIO.H

- fopen() Abre un archivo.
- fclose() Cierra un archivo.
- fgets() Lee una cadena de un archivo.
- fputs() Escribe una cadena en un archivo
- fseek() Busca un byte especifico de un archivo.
- fprintf() Escribe una salida con formato en el archivo.
- fscanf() Lee una entrada con formato desde el archivo.
- feof() Devuelve cierto si se llega al final del archivo.
- ferror() Devuelve cierto si se produce un error.
- rewind() Coloca el localizador de posición del archivo al principio del mismo.
- remove() Borra un archivo.
- fflush() Vacía un archivo.

Declaración

- Un programa necesita utilizar punteros a archivos para leer o escribir en los mismos.

Sintaxis para su declaración:

- FILE *F;

Apertura de un archivo

- **fopen() abre una secuencia para que pueda ser utilizada y la asocia a un archivo.**

Prototipo:

FILE *fopen(const char nombre_archivo, cost charmodo);

- nombre_archivo: Puntero a una cadena de caracteres que representan un nombre valido del archivo y puede incluir una especificación del directorio.
- Valores permitidos para modo.

Modo

- r Abre un archivo de texto para lectura. Debe existir
- W Crea un archivo de texto para escritura. se crea si no existe o se sobreescribe
- a Abre un archivo de texto para añadir. si no existe se crea.
- rb Abre un archivo binario para lectura.
- wb Crea un archivo binario para escritura.
- ab Abre un archivo binario para añadir.
- r+ Abre un archivo de texto para lectura / escritura. Debe existir.
- w+ Crea un archivo de texto para lectura / escritura. se crea si no existe o se sobreescribe si existe.
- a+ Añade o crea un archivo de texto para lectura / escritura.
- r+b Abre un archivo binario para lectura / escritura.
- w+b Crea un archivo binario para lectura / escritura.
- a+b Añade o crea un archivo binario para lectura / escritura. al final del contenido

fclose

- Sirve para poder cerrar un fichero que se ha abierto.

Prototipo:

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

- Valor de retorno cero indica que el fichero ha sido correctamente cerrado
- Si el valor de retorno es la constante EOF es porque ha habido algún error .

Ejemplo: fichero.in en modo lectura

```
#include <stdio.h>

int main(int argc, char** argv)
{
   FILE *fp;
   fp = fopen ( "fichero.in", "r" );
   fclose ( fp );      /* si no existe ERROR*/

   return 0;
}
```

fgetc

Lee un caracter a la vez del archivo.

- Lectura sea exitosa devuelve el caracter leído
- En caso de que no lo sea o de encontrar el final del archivo devuelve EOF.

Prototipo:

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

- EOF: Fin de archivo

Ejemplo

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE *archivo;
    char caracter;

    archivo = fopen("prueba.txt","r");

    if (archivo == NULL)
       exit(1);
    printf("\nEl contenido del archivo de prueba es \n\n");

    while (feof(archivo) == 0)   /* verifica el fin de archivo */
    {
            caracter = fgetc(archivo);
            printf("%c",caracter);
    }
  fclose(archivo);
    return 0;
}
```

fputc

- Escribe un carácter a la vez

Prototipo:

```
int fputc(int carácter, FILE *archivo);
```

Ejemplo: escribir hasta dar enter

```
#include <stdio.h>

int main ( int argc, char **argv )
{
        FILE *fp;

        char caracter;

        fp = fopen ( "fichero.txt", “w+" );
         if (fp == NULL)
           exit(1);
        printf("\nIntrouce un texto al fichero: ");

        while((caracter = getchar()) != '\n')
             printf("%c", fputc(caracter, fp));
        fclose ( fp );

        return 0;
}
```

fgets

Leer cadenas de caracteres. Hasta EOF o hasta ln o hasta (n-1) caracteres

prototipo:

```
char *fgets(char *buffer, int tamaño, FILE *archivo);
```

Parámetros:

- Buffer: Puntero a un espacio de memoria del tipo char
- Tamaño: Cantidad de caracteres a leer
- archivo: Puntero del archivo

Ejemplo

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
        FILE *archivo;

        char caracteres[100];

        archivo = fopen("prueba.txt","r");

        if (archivo == NULL)
          exit(1);
        printf("\nEl contenido del archivo de prueba es \n\n");
        while (feof(archivo) == 0)
        {
                fgets(caracteres,100,archivo);
                printf("%s",caracteres);
        }
      fclose(archivo)
        return 0;
}
```

fputs

Escribe cadenas de caracteres que forman la cadena(salvo '\0')

prototipo:

```
int *fputs(char *buffer,  FILE *archivo);
```

Parámetros:

- Buffer: Puntero a un espacio de memoria del tipo char
- archivo: Puntero del archivo

Ejemplo

```
#include <stdio.h>
#include <stdlib.h>

int main()
    {
        FILE *archivo;

        char caracteres[]=Es una prueba;

        archivo = fopen("prueba.txt","r");
         if (archivo == NULL)
           exit(1);
         fputs(caracteres,archivo);
        printf("%s",caracteres);
        fclose(archivo)
        return 0;
    }
```

fread

Es capaz de leer desde un fichero uno o varios registros de la misma longitud y a partir de una dirección de memoria determinada

Prototipo es:

```
size_t fread (void *data, size_t size, size_t count, FILE *stream);
```

Parámetros:

- **Puntero: almacenaremos los datos leídos**
- **Tamaño de los datos a leer,**
- **cantidad de esos datos a leer**
- **apuntador al fichero.**

El valor de retorno es el número de registros leídos

Ejemplo: Lee los primeros 100 caracteres

```
#include <stdio.h>
#include <stdlib.h>

int main ( int argc, char **argv )
{
        FILE *fp;

        char buffer[100];

        fp = fopen ( "fichero.in", "r+" );    /*debe existir*/
        if (archivo == NULL)
           exit(1);

        fread ( buffer, sizeof ( char ), 100, fp );

        printf("%s", buffer);

        fclose ( fp );

        return 0;
}
```

fwrite

Capaz de escribir uno o varios registros de la misma longitud.

- El valor de retorno es el número de registros escritos.

Prototipo:

```
size_t fwrite(void *puntero, size_t tamano, size_t cantidad,
FILE *archivo);
```

Parámetros:

- Puntero a la zona donde se almacena
- Tamaño de cada registro
- Número de registros a escribir
- Puntero del fichero a escribir

Ejemplo: Escribe una cadena en el archivo

```
#include<stdio.h>
#include <stdlib.h>


int main ( int argc, char **argv )
{
 FILE *fp;
 char cadena[]=“fwrite en un fichero.\n";
   fp = fopen ( "fichero.txt", "r+" );
  if (fp == NULL)
        exit(1);
 fwrite(cadena, sizeof(char), sizeof(cadena),fp);
  fclose(fp);
return 0;
}
```

fscanf

Lee valores desde un archivo, siguiendo un cierto código de formato

Prototipo

```
int fscanf(FILE *fichero,char *formato,argumento, ...);
```

Parámetros:

- Identificador del fichero
- Cadena de formato
- Variable(s) en que se guardarán los datos.

Las variables deberán aparecer precedidas por "&" (excepto arreglos).

Devuelve la cantidad de datos leídos (0 si ninguno, EOF en caso de error

Ejemplo

```
#include<stdio.h>
#include <stdlib.h>
int main(int argc,char **argv )
{
 FILE *fp;
 char buffer[100];
 fp = fopen ( "fichero.txt", "r" );
 if (fp== NULL)
    exit(1);
 fscanf(fp, "%s" ,buffer);
 printf("%s",buffer);
 fclose(fp);
return 0;
}
```

fprintf

Escribe valores a un archivo, siguiendo un cierto código de formato

Prototipo

```
int fprintf(FILE *archivo, char *formato, argumento, ...);
```

Parámetros:

- Identificador del fichero
- Cadena de formato
- Variable(s) que se escribirán al archivo

Devuelve la cantidad de datos leídos (0 si ninguno, EOF en caso de error

Ejemplo

```
#include<stdio.h>
#include <stdlib.h>
int main( int argc, char **argv )
{
 FILE *fp;
 char buffer[100] = "es texto en fichero.";
  fp = fopen ( "fichero.txt", “w+" );
  if (fp == NULL)
      exit(1);
 fprintf(fp, "%s", "\n otro texto en fichero.");
 fclose(fp);
return 0;
}
```

Ejemplo

```
#include<stdio.h>
#include <stdlib.h>

int main( int argc, char **argv )
{
 FILE *fp;
 struct k
 {
    int edad;
    char nom[10];
 }todos, final;
 todos.edad=20;
 gets(todos.nom);
  fp = fopen ( "fic.txt", "w+" );
  fwrite(&todos, sizeof(todos),1,fp);
  rewind(fp);
  fread(&final, sizeof(struct k),1,fp);
  printf("%d",final.edad);
  puts(final.nom);
 fclose(fp);
 getchar();
 return 0;
}
```

Ejemplo

- Liga a un programa completo