Lectura de *streams*

Lectura de *streams*

Para leer más líneas se pueden usar sucesivos `fscanf` o utilizar estructuras repetitivas

```
fgets(cadena, 80, fp);
while (!feof(fp)) {
  printf("%s", cadena);
  fgets(cadena, 80, fp);
}
```

Lectura de *streams*

Para leer más líneas se pueden usar sucesivos `fscanf` o utilizar estructuras repetitivas

```
fgets(cadena, 80, fp);
while (!feof(fp)) {
  printf("%s", cadena);
  fgets(cadena, 80, fp);
}
```

La función `feof` devuelve 0 (cero) si no se alcanzó el final del archivo.

Lectura de *streams*

Para leer más líneas se pueden usar sucesivos `fscanf` o utilizar estructuras repetitivas

```
fgets(cadena, 80, fp);
while (!feof(fp)) {
  printf("%s", cadena);
  fgets(cadena, 80, fp);
}
```

La función `feof` devuelve 0 (cero) si no se alcanzó el final del archivo.

Devuelve 1 (uno) luego de intentar leer datos del archivo sin lograrlo por estar al final del mismo.

Escritura de *streams*

Escritura de *streams*

Las funciones usadas para escribir en un archivo de texto, disponibles en stdio.h según el estándar son:

Escritura de *streams*

Las funciones usadas para escribir en un archivo de texto, disponibles en stdio.h según el estándar son:

fputc

```c
int  fputc(int data, FILE * stream);
```

Escritura de *streams*

Las funciones usadas para escribir en un archivo de texto, disponibles en stdio.h según el estándar son:

fputc

```
int  fputc(int data, FILE * stream);
```

Toma el dato `data`, lo convierte a `unsigned char` y lo escribe en la salida apuntada por `stream`. En caso de éxito devuelve el caracter escrito, si no, devuelve `EOF`.

Escritura de *streams*

Las funciones usadas para escribir en un archivo de texto, disponibles en stdio.h según el estándar son:

Escritura de *streams*

Las funciones usadas para escribir en un archivo de texto, disponibles en stdio.h según el estándar son:

fputs

```c
int fputs(const char * ps, FILE * stream);
```

Escritura de *streams*

Las funciones usadas para escribir en un archivo de texto, disponibles en stdio.h según el estándar son:

fputs

```c
int fputs(const char * ps, FILE * stream);
```

Toma la cadena apuntada por el puntero `ps`, y la escribe en la salida apuntada por `stream` hasta que encuentra un caracter nulo. En caso de éxito devuelve un entero mayor a cero, si no, devuelve `EOF`.

Escritura de *streams*

Las funciones usadas para escribir en un archivo de texto, disponibles en stdio.h según el estándar son:

Escritura de *streams*

Las funciones usadas para escribir en un archivo de texto, disponibles en stdio.h según el estándar son:

fprintf

```c
int fprintf(FILE * stream, const char * format, ...);
```

Escritura de *streams*

Las funciones usadas para escribir en un archivo de texto, disponibles en stdio.h según el estándar son:

fprintf

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

Exactamente igual al `printf` que se usa con `stdout` (a la pantalla) pero lleva como primer argumento el *stream* donde se colocan los valores. Tiene la ventaja de dar formato a la salida.

Ejemplo Lectura/Escritura de *streams*

Ejemplo Lectura/Escritura de *streams*

```c
#include <stdio.h>

void agregar(FILE *);
void mostrar(FILE *);
void menu (FILE *);
int menu_option (void);

int main (void) {
  FILE *fp;

  fp = fopen("db.txt","a+");

  menu(fp);

  fclose(fp);
  return 0;
}
```

Ejemplo Lectura/Escritura de *streams*

```c
#include <stdio.h>

void agregar(FILE *);
void mostrar(FILE *);
void menu (FILE *);
int menu_option (void);

int main (void) {
  FILE *fp;

  fp = fopen("db.txt","a+");

  menu(fp);

  fclose(fp);
  return 0;
}
```

Ejemplo Lectura/Escritura de *streams*

```c
#include <stdio.h>

void agregar(FILE *);
void mostrar(FILE *);
void menu (FILE *);
int menu_option (void);

int main (void) {
  FILE *fp;

  fp = fopen("db.txt","a+");

  menu(fp);

  fclose(fp);
  return 0;
}
```

El *modo* `"a+"` permite agregar registros al final del archivo y al mismo tiempo leer.

Ejemplo Lectura/Escritura de *streams*

Ejemplo Lectura/Escritura de *streams*

```c
void menu (FILE *fp)
{
  int salir = 0;
  int op;

  do {
    op = menu_option();
    switch (op) {
      case 1:
        agregar(fp);
        break;
      case 2:
        mostrar(fp);
        break;
      case 3:
        salir = 1;
        break;
      default:
        printf("Opción no válida\n");
    }
  } while ( salir == 0 );
}
```

Ejemplo Lectura/Escritura de *streams*

Ejemplo Lectura/Escritura de *streams*

```c
int menu_option (void)
{
  int op;

  printf("1-Agregar\n");
  printf("2-Mostrar\n");
  printf("3-Salir\n");
  printf("Que quiere? "); scanf("%d", &op);

  return op;
}
```

Ejemplo Lectura/Escritura de *streams*

Ejemplo Lectura/Escritura de *streams*

```c
void agregar(FILE *fp)
{
  int legajo;
  char nombre[80];
  char apellido[80];

  printf("Ingrese el nombre: "); scanf(" %80[^\n]s", nombre);
  printf("Ingrese el apellido: "); scanf(" %80[^\n]s", apellido);
  printf("Ingrese el legajo: "); scanf("%d", &legajo);

  fprintf(fp, "%d %s, %s\n", legajo, apellido, nombre);
}
```

Ejemplo Lectura/Escritura de *streams*

```c
void agregar(FILE *fp)
{
  int legajo;
  char nombre[80];
  char apellido[80];

  printf("Ingrese el nombre: "); scanf(" %80[^\n]s", nombre);
  printf("Ingrese el apellido: "); scanf(" %80[^\n]s", apellido);
  printf("Ingrese el legajo: "); scanf("%d", &legajo);

  fprintf(fp, "%d %s, %s\n", legajo, apellido, nombre);
}
```

Ejemplo Lectura/Escritura de *streams*

```c
void agregar(FILE *fp)
{
  int legajo;
  char nombre[80];
  char apellido[80];

  printf("Ingrese el nombre: "); scanf(" %80[^\n]s", nombre);
  printf("Ingrese el apellido: "); scanf(" %80[^\n]s", apellido);
  printf("Ingrese el legajo: "); scanf("%d", &legajo);

  fprintf(fp, "%d %s, %s\n", legajo, apellido, nombre);
}
```

La función `fprintf` agrega el registro al final (debido al a+ del `fopen`)

Ejemplo Lectura/Escritura de *streams*

Ejemplo Lectura/Escritura de *streams*

```c
void mostrar(FILE *fp)
{
  char linea[160];

  rewind(fp);

  fgets(linea, 160, fp);
  while (!feof(fp)) {
    fputs(linea, stdout);
    fgets(linea, 160, fp);
  }

}
```

Ejemplo Lectura/Escritura de *streams*

```c
void mostrar(FILE *fp)
{
  char linea[160];

  rewind(fp);

  fgets(linea, 160, fp);
  while (!feof(fp)) {
    fputs(linea, stdout);
    fgets(linea, 160, fp);
  }

}
```

Ejemplo Lectura/Escritura de *streams*

```c
void mostrar(FILE *fp)
{
  char linea[160];

  rewind(fp);

  fgets(linea, 160, fp);
  while (!feof(fp)) {
    fputs(linea, stdout);
    fgets(linea, 160, fp);
  }

}
```

La función `rewind` vuelve al inicio al puntero que indica cual es la posición que se lee o escribe en el archivo.

Archivos binarios

Archivos binarios

La diferencia entre los archivos de texto y los binarios es que los primeros siempre usan caracteres para representar la información, aunque sean números.

Archivos binarios

La diferencia entre los archivos de texto y los binarios es que los primeros siempre usan caracteres para representar la información, aunque sean números.

Los binarios escriben en los archivos con bits, sin importar lo que representen esos bits.

Archivos binarios

La diferencia entre los archivos de texto y los binarios es que los primeros siempre usan caracteres para representar la información, aunque sean números.

Los binarios escriben en los archivos con bits, sin importar lo que representen esos bits.

En cuestiones de espacio, *a veces* puede ser una ventaja

Archivos binarios

Archivos binarios

La función usada para escribir en un archivo binario, disponible en stdio.h según el estándar es:

Archivos binarios

La función usada para escribir en un archivo binario, disponible en stdio.h según el estándar es:

fwrite

```
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
```

Archivos binarios

La función usada para escribir en un archivo binario, disponible en stdio.h según el estándar es:

fwrite

```
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
```

Escribe en el *stream* apuntado por `stream`, `nmemb` datos de tamaño `size` comenzando por la posición de memoria apuntada por `ptr`.

Archivos binarios

Archivos binarios

```
FILE * fpt;
FILE * fpb;
int numero = 4000000;

fpt = fopen("archivo.txt", "w");
fpb = fopen("archivo.bin", "wb");

fprintf(fpt, "%d", numero);
fwrite(&numero, sizeof(numero), 1, fpb);

fclose(fpb);
fclose(fpt);
```

Archivos binarios

```
FILE * fpt;
FILE * fpb;
int numero = 4000000;

fpt = fopen("archivo.txt", "w");
fpb = fopen("archivo.bin", "wb");

fprintf(fpt, "%d", numero);
fwrite(&numero, sizeof(numero), 1, fpb);

fclose(fpb);
fclose(fpt);
```

Archivos binarios

```c
FILE * fpt;
FILE * fpb;
int numero = 4000000;

fpt = fopen("archivo.txt", "w");
fpb = fopen("archivo.bin", "wb");

fprintf(fpt, "%d", numero);
fwrite(&numero, sizeof(numero), 1, fpb);

fclose(fpb);
fclose(fpt);
```

si se graba en `archivo.txt` en modo texto la variable `numero`, se graban siete caracteres (un 4 y seis 0) lo que corresponden a siete bytes.

Archivos binarios

```
FILE * fpt;
FILE * fpb;
int numero = 4000000;

fpt = fopen("archivo.txt", "w");
fpb = fopen("archivo.bin", "wb");

fprintf(fpt, "%d", numero);
fwrite(&numero, sizeof(numero), 1, fpb);

fclose(fpb);
fclose(fpt);
```

Archivos binarios

```c
FILE * fpt;
FILE * fpb;
int numero = 4000000;

fpt = fopen("archivo.txt", "w");
fpb = fopen("archivo.bin", "wb");

fprintf(fpt, "%d", numero);
fwrite(&numero, sizeof(numero), 1, fpb);

fclose(fpb);
fclose(fpt);
```

Archivos binarios

```c
FILE * fpt;
FILE * fpb;
int numero = 4000000;

fpt = fopen("archivo.txt", "w");
fpb = fopen("archivo.bin", "wb");

fprintf(fpt, "%d", numero);
fwrite(&numero, sizeof(numero), 1, fpb);

fclose(fpb);
fclose(fpt);
```

si se graba en `archivo.bin` en modo binario la variable `numero`, se graban cuatro bytes que corresponden a un `int`

Archivos binarios

```
FILE * fpt;
FILE * fpb;
int numero = 4000000;

fpt = fopen("archivo.txt", "w");
fpb = fopen("archivo.bin", "wb");

fprintf(fpt, "%d", numero);
fwrite(&numero, sizeof(numero), 1, fpb);

fclose(fpb);
fclose(fpt);
```

Archivos binarios

```c
  FILE * fpt;
  FILE * fpb;
  int numero = 4000000;

  fpt = fopen("archivo.txt", "w");
  fpb = fopen("archivo.bin", "wb");

  fprintf(fpt, "%d", numero);
  fwrite(&numero, sizeof(numero), 1, fpb);

  fclose(fpb);
  fclose(fpt);
```

```
$ ls -lh archivo.*
-rw-rw-r-- 1 claudiojpaz claudiojpaz 4 oct 20 19:23 archivo.bin
-rw-rw-r-- 1 claudiojpaz claudiojpaz 7 oct 20 19:23 archivo.txt
$
```

Archivos binarios

```c
FILE * fpt;
FILE * fpb;
int numero = 4000000;

fpt = fopen("archivo.txt", "w");
fpb = fopen("archivo.bin", "wb");

fprintf(fpt, "%d", numero);
fwrite(&numero, sizeof(numero), 1, fpb);

fclose(fpb);
fclose(fpt);
```

```
$ ls -lh archivo.*
-rw-rw-r-- 1 claudiojpaz claudiojpaz 4 oct 20 19:23 archivo.bin
-rw-rw-r-- 1 claudiojpaz claudiojpaz 7 oct 20 19:23 archivo.txt
$
```

Archivos binarios

Archivos binarios

```
FILE * fpt;
FILE * fpb;
char cadena[80] = "Texto Vs Binario";

fpt = fopen("archivo.txt", "w");
fpb = fopen("archivo.bin", "wb");

fprintf(fpt, "%s", cadena);
fwrite(cadena, sizeof(cadena), 1, fpb);

fclose(fpb);
fclose(fpt);
```

Archivos binarios

```
  FILE * fpt;
  FILE * fpb;
  char cadena[80] = "Texto Vs Binario";

  fpt = fopen("archivo.txt", "w");
  fpb = fopen("archivo.bin", "wb");

  fprintf(fpt, "%s", cadena);
  fwrite(cadena, sizeof(cadena), 1, fpb);

  fclose(fpb);
  fclose(fpt);
```

```
$ ls -lh archivo.*
-rw-rw-r-- 1 claudiojpaz claudiojpaz 80 oct 20 23:37 archivo.bin
-rw-rw-r-- 1 claudiojpaz claudiojpaz 16 oct 20 23:37 archivo.txt
$
```

Archivos binarios

```
  FILE * fpt;
  FILE * fpb;
  char cadena[80] = "Texto Vs Binario";

  fpt = fopen("archivo.txt", "w");
  fpb = fopen("archivo.bin", "wb");

  fprintf(fpt, "%s", cadena);
  fwrite(cadena, sizeof(cadena), 1, fpb);

  fclose(fpb);
  fclose(fpt);
```

```
$ ls -lh archivo.*
-rw-rw-r-- 1 claudiojpaz claudiojpaz 80 oct 20 23:37 archivo.bin
-rw-rw-r-- 1 claudiojpaz claudiojpaz 16 oct 20 23:37 archivo.txt
$
```

Archivos binarios

```c
FILE * fpt;
FILE * fpb;
char cadena[80] = "Texto Vs Binario";

fpt = fopen("archivo.txt", "w");
fpb = fopen("archivo.bin", "wb");

fprintf(fpt, "%s", cadena);
fwrite(cadena, sizeof(cadena), 1, fpb);

fclose(fpb);
fclose(fpt);
```

```
$ ls -lh archivo.*
-rw-rw-r-- 1 claudiojpaz claudiojpaz 80 oct 20 23:37 archivo.bin
-rw-rw-r-- 1 claudiojpaz claudiojpaz 16 oct 20 23:37 archivo.txt
$
```

El archivo binario almacena la cadena completa

Archivos binarios

Archivos binarios

La ventaja principal del uso de archivos en modo binario es que cada registro ocupa largo fijo.

Archivos binarios

La ventaja principal del uso de archivos en modo binario es que cada registro ocupa largo fijo.

Entonces se puede accesar a los datos del archivo de manera aleatoria, porque se puede calcular donde comienza cada dato.

Archivos binarios

La ventaja principal del uso de archivos en modo binario es que cada registro ocupa largo fijo.

Entonces se puede accesar a los datos del archivo de manera aleatoria, porque se puede calcular donde comienza cada dato.

Además como la información se almacena byte a byte, se pueden guardar datos más complejos, como estructuras de datos.

Archivos binarios

Archivos binarios

La función usada para leer de un archivo binario, disponible en stdio.h según el estándar es:

Archivos binarios

La función usada para leer de un archivo binario, disponible en stdio.h según el estándar es:

fread

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
```

Archivos binarios

La función usada para leer de un archivo binario, disponible en stdio.h según el estándar es:

fread

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
```

Lee del *stream* apuntado por `stream`, `nmemb` datos de tamaño `size` y los pone en la posición de memoria apuntada por `ptr`.

Archivos binarios

Archivos binarios

La otra función más usada en un archivo binario, disponible en stdio.h según el estándar es:

Archivos binarios

La otra función más usada en un archivo binario, disponible en stdio.h según el estándar es:

fseek

```
int fseek(FILE *stream, long int offset, int whence);
```

Archivos binarios

La otra función más usada en un archivo binario, disponible en stdio.h según el estándar es:

fseek

```
int fseek(FILE *stream, long int offset, int whence);
```

Mueve el puntero de posición dentro del archivo apuntado por `stream` una distancia dada por `offset`. El argumento `whence` indica desde donde se hace el movimiento del offset.

Archivos binarios

La otra función más usada en un archivo binario, disponible en stdio.h según el estándar es:

fseek

```
int fseek(FILE *stream, long int offset, int whence);
```

Archivos binarios

La otra función más usada en un archivo binario, disponible en stdio.h según el estándar es:

fseek

```
int fseek(FILE *stream, long int offset, int whence);
```

SEEK_SET desde el principio del archivo

SEEK_CUR desde la posición actual

SEEK_END desde el final