Informática I

Claudio Paz

claudiojpaz@gmail.com

Octubre 2019

Unidad 10

Manejo de archivos en C

Concepto de Flujo de datos (*streams*)

Concepto de Flujo de datos (*streams*)

La mayoría de los programas necesitan *entradas* (in) o *salidas* (out), o ambas.

Concepto de Flujo de datos (*streams*)

La mayoría de los programas necesitan *entradas* (in) o *salidas* (out), o ambas.

El lenguaje C NO dispone de sentencias de entrada/salida.

Concepto de Flujo de datos (*streams*)

La mayoría de los programas necesitan *entradas* (in) o *salidas* (out), o ambas.

El lenguaje C NO dispone de sentencias de entrada/salida.

Hay que recurrir a una biblioteca que tenga funciones para manejar las entrada/salidas.

Concepto de Flujo de datos (*streams*)

La mayoría de los programas necesitan *entradas* (in) o *salidas* (out), o ambas.

El lenguaje C NO dispone de sentencias de entrada/salida.

Hay que recurrir a una biblioteca que tenga funciones para manejar las entrada/salidas.

Las funciones de la biblioteca estándar están *declaradas* en el archivo de cabecera `stdio.h`

Concepto de Flujo de datos (*streams*)

Concepto de Flujo de datos (*streams*)

Todas las entradas y salidas son simplemente secuencias de bytes que van de un lado a otro, llamadas *streams*.

Concepto de Flujo de datos (*streams*)

Todas las entradas y salidas son simplemente secuencias de bytes que van de un lado a otro, llamadas *streams*.

Las entradas son bytes que van desde dispositivos (teclado, disco, red) hacia la memoria.

Concepto de Flujo de datos (*streams*)

Todas las entradas y salidas son simplemente secuencias de bytes que van de un lado a otro, llamadas *streams*.

Las entradas son bytes que van desde dispositivos (teclado, disco, red) hacia la memoria.

Las salidas son bytes que van desde la memoria a un dispositivo (pantalla, impresora, disco, red).

Concepto de Flujo de datos (*streams*)

Concepto de Flujo de datos (*streams*)

Cuando un programa comienza, ya tiene tres *streams* estándar predefinidos, listos para su uso...

Concepto de Flujo de datos (*streams*)

Cuando un programa comienza, ya tiene tres *streams* estándar predefinidos, listos para su uso...

...están declarados en `stdio.h`

Concepto de Flujo de datos (*streams*)

Cuando un programa comienza, ya tiene tres *streams* estándar predefinidos, listos para su uso...

...están declarados en `stdio.h`

`stdin` es la entrada y está *conectada* al teclado.

Concepto de Flujo de datos (*streams*)

Cuando un programa comienza, ya tiene tres *streams* estándar predefinidos, listos para su uso...

...están declarados en `stdio.h`

`stdin` es la entrada y está *conectada* al teclado.

`stdout` es la salida y está *conectada* a la pantalla.

Concepto de Flujo de datos (*streams*)

Cuando un programa comienza, ya tiene tres *streams* estándar predefinidos, listos para su uso...

...están declarados en `stdio.h`

`stdin` es la entrada y está *conectada* al teclado.

`stdout` es la salida y está *conectada* a la pantalla.

`stderr` es la salida de error y está *conectada* a la pantalla.

Concepto de Flujo de datos (*streams*)

Concepto de Flujo de datos (*streams*)

El SO puede permitir redireccionarlos a otros dispositivos.

Concepto de Flujo de datos (*streams*)

El SO puede permitir redireccionarlos a otros dispositivos.

Tanto `stdin`, `stdout` y `stderr` son de tipo `FILE *`

Concepto de Flujo de datos (*streams*)

El SO puede permitir redireccionarlos a otros dispositivos.

Tanto `stdin`, `stdout` y `stderr` son de tipo `FILE *`

`FILE *` es un tipo de datos usado para representar *streams*

Concepto de Flujo de datos (*streams*)

El SO puede permitir redireccionarlos a otros dispositivos.

Tanto `stdin`, `stdout` y `stderr` son de tipo `FILE *`

`FILE *` es un tipo de datos usado para representar *streams*

Es una estructura con información necesaria para el manejo en bajo nivel de los archivos.

Apertura y cierre de *streams*

Apertura y cierre de *streams*

Las funciones utilizadas para abrir y cerrar archivos son `fopen` y `fclose`

Apertura y cierre de *streams*

Las funciones utilizadas para abrir y cerrar archivos son `fopen` y `fclose`

`fopen`

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

Apertura y cierre de *streams*

Las funciones utilizadas para abrir y cerrar archivos son `fopen` y `fclose`

`fopen`

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

`fopen` recibe dos cadenas como argumentos: el nombre del archivo y el modo de apertura.

Apertura y cierre de *streams*

Las funciones utilizadas para abrir y cerrar archivos son `fopen` y `fclose`

`fopen`

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

`fopen` recibe dos cadenas como argumentos: el nombre del archivo y el modo de apertura.

Devuelve un `FILE *` que representa al *stream* asociado al archivo o `NULL` ante un error.

Apertura y cierre de *streams*

Las funciones utilizadas para abrir y cerrar archivos son `fopen` y `fclose`

Apertura y cierre de *streams*

Las funciones utilizadas para abrir y cerrar archivos son `fopen` y `fclose`

`fclose`

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

Apertura y cierre de *streams*

Las funciones utilizadas para abrir y cerrar archivos son `fopen` y `fclose`

`fclose`

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

`fclose` recibe el `FILE *` del *stream* asociado al archivo que se quiere cerrar.

Apertura y cierre de *streams*

Las funciones utilizadas para abrir y cerrar archivos son `fopen` y `fclose`

`fclose`

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

`fclose` recibe el `FILE *` del *stream* asociado al archivo que se quiere cerrar.

Devuelve un 0 (cero) si tuvo éxito, o un `EOF` si falló.

Apertura y cierre de *streams*

Apertura y cierre de *streams*

Según el estandar los archivos pueden ser abiertos en modo *texto* o en modo *binario*, para lectura, escritura o ambos.

Apertura y cierre de *streams*

Según el estandar los archivos pueden ser abiertos en modo *texto* o en modo *binario*, para lectura, escritura o ambos.

Los modos de apertura son:

- r para lectura en modo texto

Apertura y cierre de *streams*

Según el estandar los archivos pueden ser abiertos en modo *texto* o en modo *binario*, para lectura, escritura o ambos.

Los modos de apertura son:

- r para lectura en modo texto
- w para escritura en modo texto. Si no existe lo crea. Si existe lo sobreescribe.

Apertura y cierre de *streams*

Según el estandar los archivos pueden ser abiertos en modo *texto* o en modo *binario*, para lectura, escritura o ambos.

Los modos de apertura son:

- r para lectura en modo texto
- w para escritura en modo texto. Si no existe lo crea. Si existe lo sobreescribe.
- a para escritura en modo texto. Si no existe lo crea. Si existe escribe al final.

Apertura y cierre de *streams*

Apertura y cierre de *streams*

Otros modos de apertura son:

- r+ para lectura y escritura en modo texto. El archivo debe existir.

Apertura y cierre de *streams*

Otros modos de apertura son:

- r+ para lectura y escritura en modo texto. El archivo debe existir.
- w+ para lectura y escritura en modo texto. Si no existe lo crea. Si existe lo sobreescribe.

Apertura y cierre de *streams*

Otros modos de apertura son:

- r+ para lectura y escritura en modo texto. El archivo debe existir.
- w+ para lectura y escritura en modo texto. Si no existe lo crea. Si existe lo sobreescribe.
- a+ para lectura y escritura en modo texto. Si no existe lo crea. Siempre escribe al final.

Apertura y cierre de *streams*

Apertura y cierre de *streams*

En modo binario existen los mismos modos, agregando después de la letra inicial una b

Apertura y cierre de *streams*

En modo binario existen los mismos modos, agregando después de la letra inicial una b

- rb
- wb
- ab
- rb+
- wb+
- ab+

Apertura y cierre de *streams*

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

`fclose` cierra el *stream* estándar `stdout`

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

`fopen` abre el archivo `salida.txt` en modo escritura

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

`fopen` abre el archivo `salida.txt` en modo escritura, y el *stream* asociado es asignado en `stdout`. Si el archivo no existe, lo crea.

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

Luego de compilar, la ejecución no muestra nada por pantalla

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u10-redirect-stdout.c
$
```

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u10-redirect-stdout.c
$ ./a.out
$
```

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u10-redirect-stdout.c
$ ./a.out
$
```

pero si se revisa el contenido del archivo `salida.txt`...

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u10-redirect-stdout.c
$ ./a.out
$
```

Apertura y cierre de *streams*

Ejemplo de redirección de `stdout` a un archivo

```c
#include <stdio.h>

int main (void)
{
  fclose(stdout);
  stdout = fopen("salida.txt", "w");

  printf("Hola, mundo!\n");

  return 0;
}
```

```
$ gcc -Wall -std=c99 -pedantic-errors u10-redirect-stdout.c
$ ./a.out
$ cat salida.txt
Hola, mundo!
$
```

Lectura de *streams*

Lectura de *streams*

Descontando las funciones usadas para leer desde `stdin`, las funciones usadas para leer desde un archivo en general, disponibles en `stdio.h` según el estándar son:

Lectura de *streams*

Descontando las funciones usadas para leer desde `stdin`, las funciones usadas para leer desde un archivo en general, disponibles en `stdio.h` según el estándar son:

fgetc

```c
int fgetc ( FILE * stream );
```

Lectura de *streams*

Descontando las funciones usadas para leer desde `stdin`, las funciones usadas para leer desde un archivo en general, disponibles en `stdio.h` según el estándar son:

fgetc

```c
int fgetc ( FILE * stream );
```

Toma un caracter desde el archivo representado por el *stream* `stream` y lo devuelve como `int`

Lectura de *streams*

Descontando las funciones usadas para leer desde `stdin`, las funciones usadas para leer desde un archivo en general, disponibles en `stdio.h` según el estándar son:

Lectura de *streams*

Descontando las funciones usadas para leer desde `stdin`, las funciones usadas para leer desde un archivo en general, disponibles en `stdio.h` según el estándar son:

fgets

```
char *fgets(char *str, int n, FILE *stream);
```

Lectura de *streams*

Descontando las funciones usadas para leer desde `stdin`, las funciones usadas para leer desde un archivo en general, disponibles en `stdio.h` según el estándar son:

fgets

```
char *fgets(char *str, int n, FILE *stream);
```

Toma una cantidad `n` de caracteres desde el archivo representado por el *stream* `stream` y lo guarda en el arreglo apuntado por el puntero `str`

Lectura de *streams*

Descontando las funciones usadas para leer desde `stdin`, las funciones usadas para leer desde un archivo en general, disponibles en `stdio.h` según el estándar son:

Lectura de *streams*

Descontando las funciones usadas para leer desde `stdin`, las funciones usadas para leer desde un archivo en general, disponibles en `stdio.h` según el estándar son:

fscanf

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

Lectura de *streams*

Descontando las funciones usadas para leer desde `stdin`, las funciones usadas para leer desde un archivo en general, disponibles en `stdio.h` según el estándar son:

fscanf

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

Exactamente igual al `scanf` que se usa con `stdin` (desde el teclado) pero lleva como primer argumento el *stream* de donde se toman los valores

Lectura de *streams*

Lectura de *streams*

Si se tiene un archivo de texto llamado `frutas.txt`

Lectura de *streams*

Si se tiene un archivo de texto llamado `frutas.txt`

```
$ cat frutas.txt
manzana
banana
naranja
pera
$
```

Lectura de *streams*

Lectura de *streams*

```c
#include <stdio.h>

int main (void)
{
  FILE * fp;
  char cadena[80] = {0};

  fp = fopen("frutas.txt", "r");
  if ( fp == NULL ) {
    printf("No se pudo abrir...\n");
    return 1;
  }

  fgets(cadena, 80, fp);
  printf("%s\n", cadena);

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

Lectura de *streams*

```c
#include <stdio.h>

int main (void)
{
  FILE * fp;
  char cadena[80] = {0};

  fp = fopen("frutas.txt", "r");
  if ( fp == NULL ) {
    printf("No se pudo abrir...\n");
    return 1;
  }

  fgets(cadena, 80, fp);
  printf("%s\n", cadena);

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

- `fopen` devuelve `NULL` si el archivo no puede abrirse por algún motivo

Lectura de *streams*

```c
#include <stdio.h>

int main (void)
{
  FILE * fp;
  char cadena[80] = {0};

  fp = fopen("frutas.txt", "r");
  if ( fp == NULL ) {
    printf("No se pudo abrir...\n");
    return 1;
  }

  fgets(cadena, 80, fp);
  printf("%s\n", cadena);

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

Lectura de *streams*

```c
#include <stdio.h>

int main (void)
{
  FILE * fp;
  char cadena[80] = {0};

  fp = fopen("frutas.txt", "r");
  if ( fp == NULL ) {
    printf("No se pudo abrir...\n");
    return 1;
  }

  fgets(cadena, 80, fp);
  printf("%s\n", cadena);

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

Lectura de *streams*

```c
#include <stdio.h>

int main (void)
{
  FILE * fp;
  char cadena[80] = {0};

  fp = fopen("frutas.txt", "r");
  if ( fp == NULL ) {
    printf("No se pudo abrir...\n");
    return 1;
  }

  fgets(cadena, 80, fp);
  printf("%s\n", cadena);

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

- debe chequearse la correcta apertura para evitar errores en tiempo de ejecución

Lectura de *streams*

```c
#include <stdio.h>

int main (void)
{
  FILE * fp;
  char cadena[80] = {0};

  fp = fopen("frutas.txt", "r");
  if ( fp == NULL ) {
    printf("No se pudo abrir...\n");
    return 1;
  }

  fgets(cadena, 80, fp);
  printf("%s\n", cadena);

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

Lectura de *streams*

```c
#include <stdio.h>

int main (void)
{
  FILE * fp;
  char cadena[80] = {0};

  fp = fopen("frutas.txt", "r");
  if ( fp == NULL ) {
    printf("No se pudo abrir...\n");
    return 1;
  }

  fgets(cadena, 80, fp);
  printf("%s\n", cadena);

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

Lectura de *streams*

```c
#include <stdio.h>

int main (void)
{
  FILE * fp;
  char cadena[80] = {0};

  fp = fopen("frutas.txt", "r");
  if ( fp == NULL ) {
    printf("No se pudo abrir...\n");
    return 1;
  }

  fgets(cadena, 80, fp);
  printf("%s\n", cadena);

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

- `fgets` toma *hasta* 79 (80-1) caracteres. Si encuentra un retorno de línea (\n) termina

Lectura de *streams*

Lectura de *streams*

se puede reemplazar la línea

```
fgets(cadena, 80, fp);
```

Lectura de *streams*

se puede reemplazar la línea

```
fgets(cadena, 80, fp);
```

...por

```
fscanf(fp, "%80[^\n]s", cadena);
```

Lectura de *streams*

se puede reemplazar la línea

```
fgets(cadena, 80, fp);
```

...por

```
fscanf(fp, "%80[^\n]s", cadena);
```

La entrada `fscanf` es formateada, a diferencia de `fgets` que solo lee cadenas de texto.

Lectura de *streams*

se puede reemplazar la línea

```
fgets(cadena, 80, fp);
```

...por

```
fscanf(fp, "%80[^\n]s", cadena);
```

La entrada `fscanf` es formateada, a diferencia de `fgets` que solo lee cadenas de texto.

Con `fscanf` se pueden leer números enteros o reales directamente desde el archivo.