
Programación I

Teoría : Entrada/Salida - Archivos

<http://proguno.unsl.edu.ar>

proguno@unsl.edu.ar

Entrada/Salida

Interacción del programa con su ambiente para leer (***entrada***) y escribir (***salida***) datos.

Modelo de Entrada/Salida en C

- Las entradas o salidas en C, sin importar de dónde vienen o hacia dónde van, tratan con ***streams (flujos) de bytes***.
- En las operaciones de entrada, los bytes *fluyen* desde un dispositivo de entrada a la memoria principal.
- En las operaciones de salida, los bytes *fluyen* desde la memoria principal a un dispositivo de salida.

Entrada/Salida

- Hasta ahora hemos trabajado sólo:
 - Entrada estándar: teclado
 - getchar: lee un caracter
 - scanf: lee entradas con formato
 - Salida estándar: pantalla
 - printf: imprime datos con formato
- Existen otras formas ...

Entrada y Salida Estándar

- Entrada estándar (o por defecto): generalmente es el teclado
- Salida estándar (o por defecto): generalmente es la pantalla
- Ambos pueden ser ***redireccionados*** a archivos.

Redireccionamiento de la Entrada y la Salida Estándar

- Ej: programa sumador:

```
#include <stdio.h>
main() {
    float sum, x;
    sum = 0;
    while (scanf("%f", &x) == 1)
        sum = sum + x;
    printf("La suma de los numeros
        ingresados es %.2f\n", sum);
}
```

Redireccionamiento de la Entrada y la Salida Estándar

- Al ejecutar `sumador` en la línea de comandos escribiremos:

Prompt del sistema | `$ sumador` | Nombre del programa a ejecutar

- Podemos redireccionar el flujo de entrada a un archivo, desde la línea de comandos:

`$ sumador <input.txt` | Nombre del archivo de entrada

Redireccionamiento de Entrada y Salida Estándar

- Igualmente, podemos redireccionar la salida a un archivo:

```
$ sumador >output.txt
```

Nombre del
archivo de
salida

- o ambos:

```
$ sumador <input.txt >output.txt
```

Acceso a Archivos

- Hasta ahora nuestros programas leen y escriben sólo en la entrada y salida estándar.
- Queremos poder leer/escribir en archivos.

Apertura de Archivos

- Para poder leer/escribir un archivo, este debe encontrarse **abierto**:

```
FILE *fopen(char *name, char *mode);
```

Puntero a archivo.
Usado en las subsecuentes lecturas y/o escrituras del archivo.
Retorna NULL si no pudo realizar la operación.

Nombre del archivo a abrir

Modo de apertura del archivo

Apertura de Archivos – Modos de Apertura

Mode	Description
r	Open an existing file for reading.
w	Create a file for writing. If the file already exists, discard the current contents.
a	Append; open or create a file for writing at the end of the file.
r+	Open an existing file for update (reading and writing).
w+	Create a file for update. If the file already exists, discard the current contents.
a+	Append: open or create a file for update; writing is done at the end of the file.
rb	Open an existing file for reading in binary mode.
wb	Create a file for writing in binary mode. If the file already exists, discard the current contents.
ab	Append; open or create a file for writing at the end of the file in binary mode.
rb+	Open an existing file for update (reading and writing) in binary mode.
wb+	Create a file for update in binary mode. If the file already exists, discard the current contents.
ab+	Append: open or create a file for update in binary mode; writing is done at the end of the file.

Apertura de Archivos - Ejemplos

```
FILE *fp1, *fp2, *fp3;
```

```
fp1 = fopen("fechas.txt", "r"); /*  
    abre archivo fechas.txt para lectura  
    */
```

```
fp2 = fopen("empleados.txt", "a"); /*  
    abre archivo empleados.txt para  
    añadir */
```

```
fp3 = fopen("master.dat", "w+"); /*  
    crea archivo master.dat para lectura  
    o escritura */
```

Acceso a Archivos

Entrada, Salida y Error Estándar

- Cuando un programa C es iniciado, el sistema operativo es responsable de abrir tres flujos y de proveer los correspondientes punteros a **FILE**:
 - ❑ la entrada estándar (**stdin**)
 - ❑ la salida estándar (**stdout**)
 - ❑ el error estándar (**stderr**)

Algunas Funciones de Entrada/Salida

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

- Lee datos desde un flujo de entrada, los interpreta de acuerdo a **un formato dado** y los almacena en variables.
- Retorna la cantidad de entradas leídas o EOF si hubo error antes de la 1ra lectura.
- (`fscanf` usado con `stdin` \equiv `scanf`)

Ejemplo lectura archivo

```
FILE *fp;  
int i;  
fp = fopen("numeros.txt", "r");  
if (fp==NULL)  
    printf ("error");  
else  
    fscanf (fp, "%d", &i);  
fclose(fp);
```

Ejemplo lectura del archivo

```
FILE *fp1;
Persona p; //tipo persona

if ((fp1=fopen("personas.txt", "r")) == NULL) )
    exit(1);
fscanf (fp1, "%s", p.nombre);
fscanf (fp1, "%d", &p.edad);
fclose(fp1);
```

Algunas Funciones de Entrada/Salida

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

- Imprime datos en un flujo de salida con **el formato** dado.
- Retorna el nro. de caracteres escritos o un valor negativo si hubo error.
- (`fprintf` usado con `stdout` \equiv `printf`)

Ejemplo escritura en archivo

```
FILE *fp;
int i;
fp = fopen("numeros.txt", "w");
if (fp==NULL)
    printf ("error");
else
    fprintf (fp, "%d", i);
fclose(fp);
```

Ejemplo escritura en archivo

```
FILE *fp1;  
persona p; //tipo persona  
  
if ((fp1=fopen("personas.txt", "w")) == NULL) )  
    exit(1);  
fprintf (fp1, "%s", p.nombre);  
fprintf (fp1, "%d", p.edad);  
fclose(fp1);
```

Algunas Funciones de Entrada/Salida

```
fread(buffer, tam, n, fp)
```

- Lee del archivo apuntado por `fp`, `n` componentes de tamaño `tam` y los almacena a partir de la dirección apuntada por `buffer`
- Lee datos desde un flujo de entrada, **sin formato** y los almacena en variables.
- Retorna el nro. de caracteres leídos o un valor negativo si hubo error.

Ejemplo de lectura desde archivo

```
FILE *fp1;
persona p[max]; //tipo persona
if ((fp1=fopen("personas.txt","r")==NULL))
    exit(1);
fread(p,sizeof(persona),2,fp1);
fclose(fp1);
```

Algunas Funciones de Entrada/Salida

```
fwrite(buffer, tam, n, fp)
```

- Escribe en el archivo apuntado por `fp`, `n` componentes de tamaño `tam` que están almacenados a partir de la dirección apuntada por `buffer`
- Escribe datos desde un flujo de entrada, **sin formato** y los almacena en variables.
- Retorna el nro. de caracteres escritos o un valor negativo si hubo error.

Ejemplo de escritura en archivo

```
FILE *fp1;  
persona p[max]; //tipo persona  
if ((fp1=fopen("personas.txt","w")==NULL))  
    exit(1);  
fwrite(p,sizeof(persona),2,fp1);  
fclose(fp1);
```

Algunas Funciones de Entrada/Salida

void rewind(FILE *stream)

- Coloca el apuntador al comienzo del archivo de un flujo de datos.

int fclose(FILE *stream)

- Cierra la conexión entre un archivo y su apuntador, dejando libre al apuntador para ser usado, si fuera necesario, con un nuevo archivo.

int feof(FILE *stream)

- Retorna 0 si NO llego al final del archivo. Valor distinto de 0 si llego a fin de archivo.

```
#include <stdlib.h>
int main() {
    char str1[10], str2[10];
    int year;
    FILE * fp1, *fp2;

    fp1 = fopen ("file1.txt", "w+");
    fp2 = fopen ("file2.txt", "w");
    fputs("Estamos en 2018", fp1);
    rewind(fp1);
    fscanf(fp1, "%s %s %d", str1, str2, &year);
    fprintf(fp2, "String1 leido |%s|\n", str1 );
    fprintf(fp2, "String2 leido |%s|\n", str2 );
    fprintf(fp2, "Entero leido |%d|\n", year );
    fclose(fp1);
    fclose(fp2);
    return(0);
}
```

Algunas Funciones de Entrada/Salida

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

```
int getc(FILE *stream)
```

- Lee y retorna el próximo caracter en un flujo de entrada avanzando el apuntador al próximo carácter en el flujo de entrada.
- (getc/fgetc usado con stdin \equiv getchar)

Algunas Funciones de Entrada/Salida

```
int fputc(int char, FILE *stream)
```

```
int putc(int char, FILE *stream)
```

- Escribe un caracter especificado por el argumento **char** en el flujo especificado y avanza la posición del apuntador en el flujo.
- Retorna el carácter escrito si no hubo error, sino EOF.
- (putc/fputc usado con stdout \equiv putchar)

Ejemplo

```
/* filecopy: copia archivo ifp a
   archivo ofp */
void filecopy(FILE *ifp, FILE *ofp) {
    int c;
    while ((c = getc(ifp)) != EOF)
        putc(c, ofp);
}
```

Algunas Funciones de Entrada/Salida

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

- Lee una línea de un flujo y la almacena en el string apuntado por **str**. Para cuando leyó (**n-1**) caracteres o leyó `'\n'` o se alcanzó el fin de archivo (lo que ocurra primero).
- Retorna la línea leída; si lee el fin del archivo u ocurre algún error durante la lectura retorna `NULL`.
- (`fgets` usado con `stdin` \approx `gets`)

Algunas Funciones de Entrada/Salida

```
int fputs(const char *str, FILE
          *stream)
```

- Escribe un string, sin `\0`, en un flujo de salida.
- Retorna un entero positivo, o EOF si hubo error.
- (`fputs` usado con `stdout` \approx `puts`)

```
int main() {
    FILE *fp;
    char str[60];

    fp = fopen("file2.txt", "r");
    if (fp == NULL) {
        perror("Error al abrir file2.txt");
        return(-1);
    }
    while (fgets(str, 60, fp) != NULL)
        fputs(str, stdout);
    fclose(fp);
    return(0);
}
```

Parámetros de un Programa

- Muchas veces, resulta necesario al invocar un programa para su ejecución, desde la línea de comandos, pasarle argumentos de entrada.
- Ejemplo: Programa `echo` que reproduce sus argumentos de la línea de comandos, en una línea aparte, separados por blancos:

```
$ echo hello, world
```

imprime en la pantalla:

```
hello, world
```

- ¿Cómo lo hacemos? Por medio de los parámetros del programa

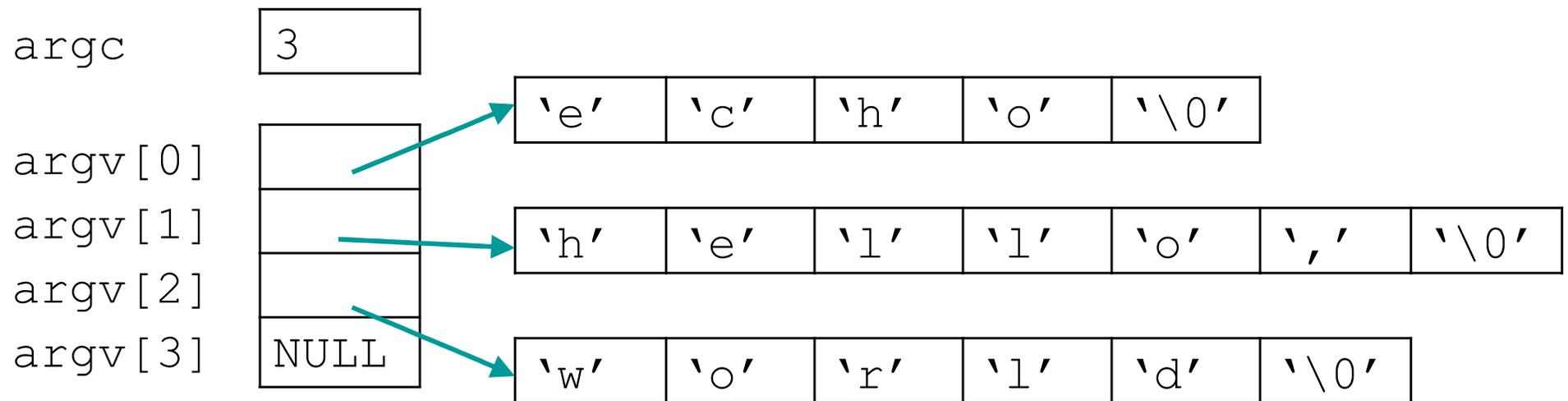
```
int main(int argc, char *argv[]);
```



Parámetros de un Programa

Ejemplo

```
$ echo hello, world
```



Parámetros de un Programa

Ejemplo

```
#include <stdio.h>
/* comando echo */
main(int argc, char *argv[]) {
    int i;
    for (i = 1; i < argc; i++) {
        printf("%s", argv[i]);
        if (i < argc-1) printf(" ");
    }
    printf("\n");
    return 0;
}
```