



Instituto Politécnico Nacional
La Técnica al Servicio de la Patria

**Unidad Profesional Interdisciplinaria de
Ingeniería Campus Tlaxcala UPIIT**

Fundamentos de Programación

Esaú Eliezer Escobar Juárez

E/S para archivos

- Las funciones y tipos están definidos en `<stdio.h>`
- FILE
 - Estructura que define un descriptor de archivo
- EOF
 - Constante para detectar el fin del archivo

Apertura de flujos

- **fopen**

- Abre un archivo para su uso

- `FILE* fopen(char* nombre, char* modo);`

Devuelve el descriptor
del archivo para su uso posterior.
NULL en caso de error

Nombre del archivo a abrir

Modo de apertura
(lectura, escritura, etc.)

Apertura de flujos

r	Abrir para lectura
w	Abrir para escritura
a	Abrir para añadir datos al final
rb	Abrir para lectura binaria
wb	Abrir para escritura binaria
ab	Abrir para añadir datos binarios
r+	Abrir para lectura/escritura
w+	Crear archivo para lectura/escritura
a+	Abre o crea para añadir datos
r+b	Abre para lectura/escritura binaria
w+b	Crea para lectura/escritura binaria
a+b	Abre o crea para añadir datos binarios


Cierre de flujos

- `fclose`

- Cierra un archivo previamente abierto, liberando los recursos asociados al programa.

- `int fclose(FILE* f);`

Éxito de la operación
(0 en caso de éxito)



Flujo a cerrar



Apertura y cierre

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

```
int main(){
```

```
    FILE* archivo;
```

```
    archivo = fopen("testt.txt", "r");
```

```
    if(archivo!=NULL) {
```

```
        if(fclose(archivo) !=EOF)
```

```
            printf("Ok!\n");
```

```
        else
```

```
            printf("Error al cerrar!\n");
```

```
    }else
```

```
        printf("Error al abrir!\n");
```

```
}
```

Manejo de errores

- En C, muchas funciones modifican una variable global cuando ocurre un error.
- Esta variable puede ser consultada para saber más acerca del error.
- La variable global se llama “`errno`”.
 - Se define en `<errno.h>`
- La función “`strerror(int e)`” entrega una descripción de un código de error.
 - Se define en `<string.h>`

Manejo de errores

```
#include <stdio.h>

int main() {

    FILE* archivo;
    archivo = fopen("testt.txt", "r");
    if (archivo != NULL) {
        if (fclose(archivo) != EOF)
            printf("Ok!\n");
        else
            printf("Error al cerrar!\n");
    } else
        printf("Error al abrir: %s\n", strerror(errno));
}
```


Lectura de caracteres

- **fgetc**

- Lee un carácter desde un archivo abierto para lectura.

- `int fgetc(FILE* f);`

Devuelve el carácter leído como un entero.
En caso de error, devuelve EOF

Descriptor de archivo

Lectura de caracteres

- **fgets**

- Lee desde un archivo abierto para lectura hasta un largo fijo o el fin de línea.

- `fgets(char* cadena, int longitud, FILE* f);`

Arreglo de caracteres donde guardar la cadena leída

Cantidad máxima de caracteres a leer

Descriptor de archivo


Lectura de caracteres

```
int main() {
    FILE* archivo;
    archivo = fopen("test.txt", "r");
    if (archivo != NULL) {
        char c;
        do{
            c = fgetc(archivo);
            printf("%c", c);
        } while (c != EOF);
        fclose(archivo);
    }
}
```

Lectura de caracteres

```
int main(){
    FILE* archivo;
    archivo = fopen("test.txt","r");
    if(archivo!=NULL){
        char* res;
        char cadena[128];
        do{
            res=fgets(cadena,128,archivo);
            if(res!=NULL) printf("%s",cadena);
        }while(res!=NULL);

        fclose(archivo);
    }
}
```



Lectura de caracteres

- Una función muy útil es “feof”, quien detecta si se ha alcanzado o no el fin de archivo.

- feof

- Devuelve verdadero si se ha alcanzado el fin de archivo

- `int feof(FILE* f);`

1: Fin de archivo
0: otro caso

Descriptor de archivo

Lectura de caracteres

```
int main() {  
    FILE* archivo;  
    archivo = fopen("test.txt", "r");  
    if (archivo != NULL) {  
        char cadena[128];  
        while (!feof(archivo)) {  
            fgets(cadena, 128, archivo);  
            printf("%s", cadena);  
        }  
        fclose(archivo);  
    }  
}
```

Lectura de caracteres

- Existe una version de scanf para archivos

• `int fscanf(FILE* f, char* fmt, ...);`



Numero de conversiones
realizadas con éxito

The diagram consists of three arrows pointing upwards from descriptive text to the parameters of the `fscanf` function signature. The first arrow points from 'Numero de conversiones realizadas con éxito' to the `int` return type. The second arrow points from 'Descriptor de archivo' to the `FILE* f` parameter. The third arrow points from 'Variables a Modificar (referencias!)' to the `char* fmt` parameter.

Descriptor de
archivo

Variables a
Modificar (referencias!)

Lectura de caracteres

```
int main() {  
    FILE* archivo;  
    archivo = fopen("test.txt", "r");  
    if (archivo != NULL) {  
        char c;  
        while( fscanf(archivo, "%c", &c) == 1 )  
            printf("%c", c);  
        fclose(archivo);  
    }  
}
```


Escritura de caracteres

- Cada función de lectura tiene su pareja
- `int fputc(int c , FILE* f);`
 - EOF en caso de error
- `int fputs(char* cadena, FILE* f);`
 - EOF en caso de error
- `int fprintf(FILE* f, char* fmt, ...);`
 - Devuelve el numero de transformaciones realizadas con éxito.

Escritura de caracteres

```
archivo2 = fopen("test2.txt", "w+");  
...  
do{  
    c = fgetc(archivo);  
    if (c!=EOF) {  
        fputc(c, archivo2);  
        printf("%c", c);  
    }  
}while (c!=EOF);
```

Escritura de caracteres

```
do {  
    res=fgets (cadena,128,archivo);  
    if (res!=NULL) {  
        printf ("%s",cadena);  
        fputs (cadena,archivo2);  
    }while (res!=NULL);
```

Escritura de caracteres

```
while( fscanf(archivo, "%c", &c) == 1 ) {  
    printf("%c", c);  
    fprintf(archivo2, "%c", c);  
}
```

Escritura de caracteres

- **fflush**

- Vacía el buffer de escritura

- `int fflush(FILE* f);`

Devuelve EOF en caso
de error



Descriptor de archivo

