

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>

- o FILE
  - Estructura que define un descriptor de archivo
- EOF
  - Constante para detectar el fin del archivo

# Apertura de flujos

- o 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

Modo de apertura (lectura, escritura, etc.)

Nombre del archivo a abrir

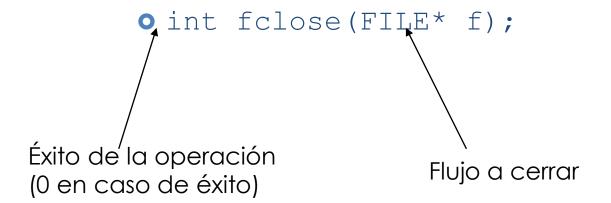
# Apertura de flujos

r	Abrir para lectura
W	Abrir para escritura
а	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.



# Apertura y cierre

# 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 descripcion de un código de error.
  - Se define en <string.h>

# Manejo de errores

- fgetc
  - Lee un carácter desde un archivo abierto para lectura.
  - o int fgetc(FILE\* f);

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

Descriptor de archivo

- fgets
  - Lee desde un archivo abierto para lectura hasta un largo fijo o el fin de línea.
  - o fgets(char\* cadena, int longitud, FILE\*
    f);

Arreglo de caracteres donde guardar la cadena leída

Cantidad máxima de carácteres a leer

Descriptor de archivo

```
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);
```

 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

o int feof(FILE\* f);

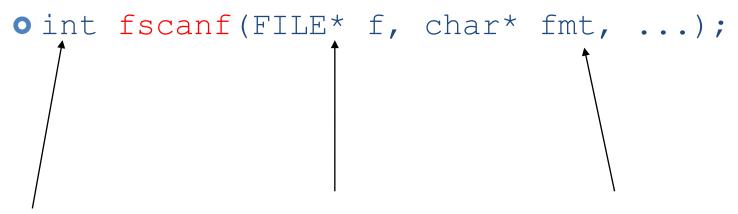
1: Fin de archivo

0: otro caso

Descriptor de archivo

```
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);
    }
}
```

• Existe una version de scanf para archivos



Numero de conversiones realizadas con éxito

Descriptor de archivo

Variables a Modificar (referencias!)

```
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);
    }
}
```

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.
```

```
archivo2 = fopen("test2.txt","w+");

do{
    c = fgetc(archivo);
    if(c!=EOF) {
        fputc(c,archivo2);
        printf("%c",c);
    }
} while(c!=EOF);
```

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

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

- fflush
  - Vacia el buffer de escritura

