

# Curso Básico de C

## Clase N° 8

*Manejo de Archivos*

*Prof. Patricia Guerrero*

# Archivos

---

- Definición:

Un ***archivo o fichero informático*** es una entidad lógica compuesta por una secuencia finita de bytes.

Se almacena en un sistema de archivos ubicada en la memoria secundaria del computador.

Se identifica por un nombre único.

# Archivos

---

- Definición:

Un ***archivo*** es un conjunto de ***datos estructurados*** en una colección de entidades elementales denominadas ***registros*** que son de igual tipo y constan a su vez de diferentes entidades de nivel más bajo denominadas ***campos***.

Colecciones de datos almacenados en memoria secundaria.

# Archivos

---

- Tipos de Archivo:

**De Texto:** Secuencia de caracteres organizadas en líneas terminadas por un carácter de nueva línea. Se caracterizan por ser planos, es decir, todas las letras tienen el mismo formato y no hay palabras subrayadas, en negrita, o letras de distinto tamaño o ancho.

En estos archivos se pueden almacenar ***canciones, fuentes de programas, base de datos simples.***

# Archivos

---

- Tipos de Archivo:

**Binarios:** Secuencia de bytes que tienen una correspondencia uno a uno con un dispositivo externo. El número de bytes escritos (leídos) será el mismo que los encontrados en el dispositivo externo.

Ejemplos de estos archivos son ***fotografías, imágenes, texto con formatos y archivos ejecutables.***

# Manejo de Archivos

- Apertura de Archivos:

Antes de abrir un fichero es necesario declarar un puntero de tipo **FILE**.

Para abrir el fichero se utiliza la función `fopen( )`

Su sintaxis es:

```
FILE *p;  
p = fopen (nombre-fichero,"modo-apertura");
```

Debe ir entre comillas.  
Es posible especificar la ruta  
del archivo.

# Manejo de Archivos

## Modo Texto:

- w** Crea un archivo de escritura.  
Si ya existe lo crea de nuevo.
- w+** Crea un archivo de lectura y escritura.  
Si ya existe lo crea de nuevo.
- a** Abre o crea un archivo para añadir datos al final del mismo.
- a+** Abre o crea un archivo para leer y añadir datos al final del mismo.
- r** Abre un archivo de lectura.
- r+** Abre un archivo de lectura y escritura.

Un archivo puede ser abierto en dos modos diferentes: **modo texto** o **modo binario**.

# Manejo de Archivos

## Modo Binario:

- wb** Crea un archivo de escritura.  
Si ya existe lo crea de nuevo.
- w+b** Crea un archivo de lectura y escritura.  
Si ya existe lo crea de nuevo.
- ab** Abre o crea un archivo para añadir datos al final del mismo.
- a+b** Abre o crea un archivo para leer y añadir datos al final del mismo.
- rb** Abre un archivo de lectura.
- r+b** Abre un archivo de lectura y escritura.

Un archivo puede ser abierto en dos modos diferentes: **modo texto** o **modo binario**.



# Manejo de Archivos

- Apertura de Archivos - Ejemplo:

```
FILE *puntero;
```

Abre el archivo en  
modo lectura.

```
puntero = fopen("datos.in","r");
```

```
puntero = fopen("C:\\Archivos\\saludo.txt","w");
```

Abre el archivo en  
modo escritura.

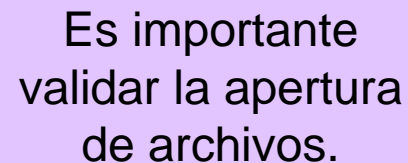
# Manejo de Archivos

- Apertura de Archivos:

Si se produce un error al intentar abrir un archivo la función **fopen** devolverá **NULL**.

```
FILE *pf;
```

```
pf = fopen("datos.txt", "r");  
if (pf == NULL)  
    printf("Error al abrir el archivo");
```



Es importante  
validar la apertura  
de archivos.

# Manejo de Archivos

## Función freopen( )

Cierra el archivo apuntado por el puntero y reasigna éste a un nuevo archivo que será abierto.

Su sintaxis es:

Nombre del nuevo  
archivo que será abierto.

**freopen (nombre-archivo, "modo-apertura", puntero);**

Puntero que será  
reassignado.

# Manejo de Archivos

- Cierre de Archivos:

Para cerrar un archivo se utiliza la función **fclose( )**.

```
FILE *pf;  
pf = fopen("Alumnos.txt", "r");  
if ( pf == NULL )  
    printf ("Error al abrir el archivo");  
else  
    fclose(pf);
```

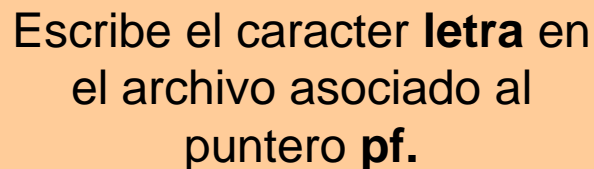
El número de archivos que pueden estar abiertos es limitado.

Cierra el archivo asociado al puntero **pf**, si el archivo se cierra con éxito devuelve **0**.

# Escritura y Lectura de Archivos

- **fputc (variable-caracter, puntero-archivo);**

```
FILE *pf;  
char letra = 'a';  
If ((pf = fopen("datos.txt", "w")) == NULL)  
{  
    printf("Error al abrir el archivo");  
    exit(0);  
}  
else  
{  
    fputc(letra, pf);  
    fclose(pf);  
}
```



Escribe el caracter **letra** en  
el archivo asociado al  
puntero **pf**.

# Escritura y Lectura de Archivos

- **fgetc (puntero-archivo);**

```
FILE *pf;  
char letra;  
if ((pf = fopen("datos.txt", "r")) == NULL)  
{  
    printf("Error al abrir el archivo");  
    exit(0);  
}  
else  
{  
    letra = fgetc(pf);  
    printf("%c", letra);  
    fclose(pf);  
}
```

Lee un caracter del archivo asociado al puntero **pf** y lo copia el **letra**.

# Escritura y Lectura de Archivos

---

- **putw (variable-entera, puntero-archivo);**

Escribe un número entero en formato binario en el archivo.

- **getw (puntero-archivo);**

Lee un número entero de un archivo, avanzando dos bytes después de cada lectura.

# Escritura y Lectura de Archivos

## **putw( ) - Ejemplo:**

```
FILE *pf;  
int num = 3;  
if ((pf = fopen("datos.txt", "wb")) == NULL)  
{  
    printf("Error al abrir el archivo");  
    exit(0);  
}  
else  
{  
    putw(num, pf); /* Equivalente a putw(3, pf); */  
    fclose(pf);  
}
```



# Escritura y Lectura de Archivos

## **getw( ) - Ejemplo:**

```
FILE *pf;  
int num;  
if ((pf = fopen("datos.txt", "rb")) == NULL)  
{  
    printf("Error al abrir el archivo");  
    exit(0);  
}  
else  
{  
    num = getw(pf);  
    printf("%d", num);  
    fclose(pf);  
}
```

# Escritura y Lectura de Archivos

- **char \*fputs (char \*str, FILE \*f);**

Escribe la cadena **str** en el archivo asociado al puntero **f**.

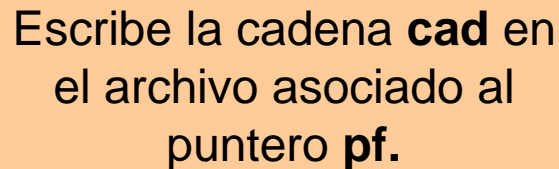
- **char \*fgets (char \*str, int long, FILE \*f);**

Lee una cadena desde el archivo asociado al puntero **f** y la copia en **str** hasta que lee un carácter de nueva línea o hasta **long-1** caracteres.

# Escritura y Lectura de Archivos

## fputs( ) - Ejemplo:

```
FILE *pf;  
char cad = "Aprendemos manejo de archivos en C";  
if ((pf = fopen("datos.txt", "w")) == NULL)  
{  
    printf("Error al abrir el archivo");  
    exit(0);  
}  
else  
{  
    fputs(cad, pf);  
    fclose(pf);  
}
```



Escribe la cadena **cad** en  
el archivo asociado al  
puntero **pf**.

# Escritura y Lectura de Archivos

## fgets - Ejemplo:

```
FILE *pf;
char cad[80];
if ((pf = fopen("datos.txt", "r")) == NULL)
{
    printf("Error al abrir el archivo");
    exit(0);
}
else
{
    fgets(cad, 80, pf);
    printf("%s", cad);
    fclose(pf);
}
```

Lee del archivo asociado a **pf** una cadena de longitud max 80-1 caracteres y la copia en **cad**.

# Escritura y Lectura de Archivos

---

- **fprintf (puntero-archivo, formato, argumentos);**

Escribe los argumentos en un archivo.

- **fscanf (puntero-archivo, formato, argumentos);**

Lee los argumentos desde un archivo. Al igual que un **scanf( )**, es necesario indicar la dirección de memoria de los argumentos con el símbolo **&**.

# Escritura y Lectura de Archivos

## **fprintf( ) - Ejemplo:**

```
FILE *pf;
char nombre[20] = "Xiomara";
int edad = 49;
if (!(pf = fopen("datos.txt", "w")))
{
    printf("Error al abrir el archivo");
    exit(0);
}
else
{
    fprintf(pf, "Nombre: %20s, Edad: %2d\n", nombre, edad);
    fclose(pf);
}
```

# Escritura y Lectura de Archivos

## **fscanf( ) - Ejemplo:**

```
FILE *pf;
char nombre[20];
int edad;
if (!(pf = fopen("datos.txt", "r")))
{
    printf("Error al abrir el archivo");
    exit(0);
}
else
{
    fscanf(pf, "%s %d", nombre, &edad);
    printf("Nombre: %s, Edad: %d", nombre, edad);
    fclose(pf);
}
```

# Otras Funciones

- **rewind( )**

Inicializa el indicador de posición, al principio del archivo. Su prototipo es: **void rewind(FILE \*F);**

- **ferror()**

Determina si ha ocurrido un error en alguna operación sobre el archivo. Su prototipo es: **int ferror(FILE \*F);**

Devuelve verdadero si se produjo un error durante la última operación sobre el archivo. En caso contrario, devuelve falso.



# Otras Funciones

---

- **remove( )**

Borra el archivo especificado. Su prototipo es:

**int remove(char \*nombre-archivo);**

Devuelve cero si tiene éxito. En caso contrario, devuelve un valor distinto de cero.