

# Laboratorio de Computación

II

Archivos

# Archivos

- Conjunto de bytes que son almacenados en un dispositivo.
  - Archivos de sistema: Utilizados por el sistema operativo y que no suelen ser accesibles al usuario estándar.
  - Archivos ejecutables: Programas o aplicaciones destinados a resolver una tarea específica. Desde comandos básicos de un SO hasta procesadores de texto y juegos.
  - Archivos de datos: Datos con un formato y/o estructura específica que son interpretados por programas específicos para poder crearlos, visualizarlos, editarlos y/o borrarlos. Por ejemplo, una planilla de cálculo y la aplicación Microsoft Excel o un conjunto de registros de información de Alumnos.

# Archivos de datos

- Un registro de información de una determinada entidad debe cumplir una serie de reglas para poder ser almacenado en un archivo:
  - Los registros deben ser de longitud fija.
  - Los registros debieran ser identificados con un valor único e irrepetible.
  - El archivo, en consecuencia al ítem anterior, no debiera admitir registros duplicados.

# Ejemplo de Archivo de datos: Escritura

```
struct Alumno{  
    int legajo;  
    char apellido[50];  
    char nombres[50];  
    int edad;  
};
```

Memoria

Alumno reg;

```
bool guardarAlumno(Alumno reg);
```

Archivo: alumnos.dat

#1	#2	#3	#4	#5	#6	#7	#8	#9
----	----	----	----	----	----	----	----	----

# Ejemplo de Archivo de datos: Lectura

alumnos.dat

ID	Apellido	Nombre	Edad
100	Santos	Mario	40
200	Medina	Gabriel	48
300	Ravenna	Emilio	44
400	Lamponne	Pablo	47

→ Alumno leerAlumno(int nroRegistro);

Memoria

200	Medina	Gabriel	48
-----	--------	---------	----

## Pero....

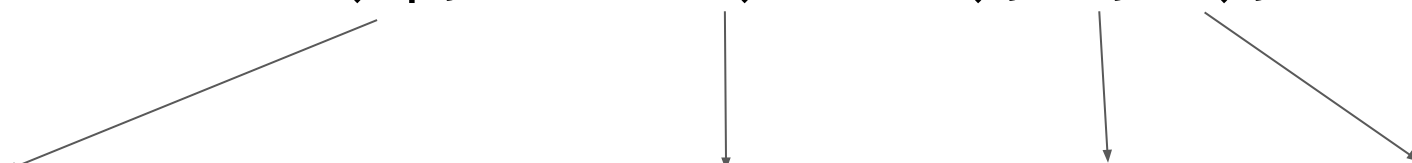
- Las funciones de archivos no admiten trabajar con "números de registros".
- Las funciones de archivos trabajan con cantidad de bytes a escribir o leer.

# Lectura

- Las funciones de archivos trabajan con cantidad de bytes a escribir o leer.

Lee del inicio del archivo apuntado por `f`, una cantidad de 120 bytes (el tamaño de un registro de `Alumno`) almacénalos en la variable `p` de tipo `Alumno`.

```
fread(&p, sizeof(Alumno), 1, f);
```



Registro de tipo `Alumno` donde guardar la información proveniente del archivo

Cantidad de bytes de cada registro a leer. Hipotéticamente 120 bytes por registro.

Cantidad registros a leer a la vez. Generalmente será de a uno por vez.

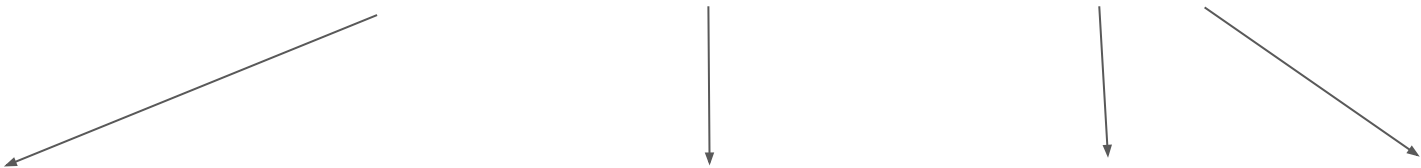
Puntero `FILE`. Tipo de dato que "apuntará" a un archivo en disco

# Escritura

- Las funciones de archivos trabajan con cantidad de bytes a escribir o leer.

Escribí al final del archivo apuntado por `f`, un registro de tamaño 120 bytes (el tamaño de un registro de `Alumno`) proveniente desde la dirección de `p`.

```
fwrite(&p, sizeof(Alumno), 1, f);
```



Registro de tipo `Alumno` de cual proviene la información a guardar

Cantidad de bytes de cada registro a escribir. Hipotéticamente 120 bytes por registro.

Cantidad registros a escribir a la vez. Generalmente será de a uno por vez.

Puntero `FILE`. Tipo de dato que "apuntará" a un archivo en disco



# Puntero FILE

- Un tipo de dato que nos permitirá "conectar" nuestro programa a un archivo en el dispositivo que indiquemos.

```
FILE *p;  
p = fopen("archivo.dat", "rb");  
  
/* Todas las acciones a realizar con  
el archivo */  
  
fclose(p);
```

# fopen

Una función que nos permite conectar un archivo de datos a un puntero FILE. Además, le indicamos el modo de apertura.

```
FILE *p = fopen("miarchivo.dat", "rb");
```

Modo	Nombre	Descripción
wb	write-binary	Destruye el archivo a 0 bytes y lo abre como escritura. Si no existe el archivo, lo crea.
rb	read-binary	Abre el archivo como lectura desde el byte 0. No admite escritura.
ab	append-binary	Abre el archivo desde el final y lo abre como escritura. No admite modificar lo existente, sólo agregar. Si no existe el archivo, lo crea.
rb+	read-binary plus overwrite	Abre el archivo desde el byte 0 y lo abre como escritura. Admite modificar registros existentes.

# **fclose**

Una función que nos permite liberar la conexión de un archivo de datos a un puntero FILE.

```
fclose(p);
```

- Es clave porque no se puede abrir el mismo archivo más de una vez simultáneamente.
- Confirma los datos en aperturas de escritura.

Ejemplo en C/C++