

# Laboratorio de Computación II

Archivos

# Archivos de datos

Un registro de información de una determinada entidad debe cumplir una serie de reglas para poder ser almacenado en un archivo



Archivo de clientes

- Registros deben tener longitud fija.
- Registros deben ser identificados por un valor único e irrepetible.
- El archivo, en consecuencia al ítem anterior, no debiera admitir registros duplicados.

# Archivos de datos

Imaginemos que tenemos un cassette de audio. En él, cada canción dura exactamente dos minutos y se encuentran una inmediatamente después de la otra.



Aclaración para centennials: La imagen corresponde a un cassette de audio

# Archivos de datos

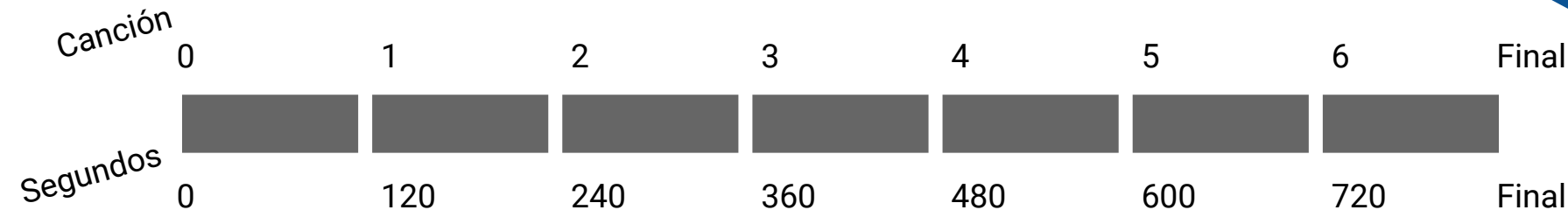
Imaginemos que tenemos un cassette de audio. En él, cada canción dura exactamente dos minutos y se encuentran una inmediatamente después de la otra.



Aclaración para centennials: La imagen corresponde a un cassette de audio

No nos es posible ir fácilmente de la canción 2 a la canción 5. Pero sabemos que la canción 1 comienza en el segundo 0. La canción 2 en el segundo 120. Por lo tanto, la canción 5 en el segundo 480.

# Archivos de datos



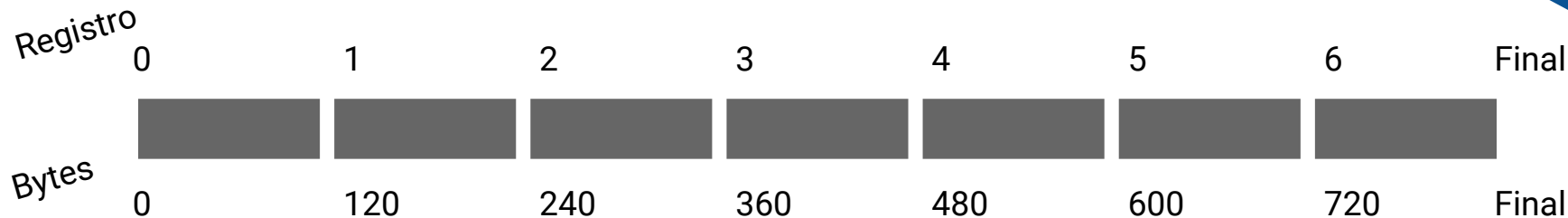
- Para ir del inicio de la canción 0 al inicio de la canción 3. No puedo decir "Adelantar 3 canciones desde la canción 0" pero sí puedo decir "Adelantar 360 segundos desde el segundo 0".
- Si estoy ubicado en la canción 3 y quisiera ir a la canción 1 podría "Retroceder 240 segundos desde el segundo 360".

1 canción



= 120 segundos

# Archivos de datos



- Pensemos la misma idea pero con archivos. Reemplacemos Canción por Registro y Segundos por Bytes. En lugar de música tenemos bits.
- Ya entendieron cómo se almacena información en un archivo binario.

1 registro  = 120 bytes

# fseek

La función que nos permite adelantar o retroceder el cursor de un archivo se llama fseek y recibe los siguientes parámetros.

```
fseek (FILE *archivo, long desplazamiento, int origen);
```

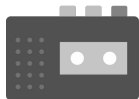
**archivo** - Es el puntero file al archivo que queremos manipular. Debe estar abierto previamente.

**desplazamiento** - Es la cantidad de bytes que queremos desplazarnos.

**origen** - Es una bandera que indica desde donde queremos desplazarnos la cantidad de bytes indicada por desplazamiento.

| Valor | Alias    | Descripción                 |
|-------|----------|-----------------------------|
| 0     | SEEK_SET | Inicio del archivo          |
| 1     | SEEK_CUR | Posición actual del archivo |
| 2     | SEEK_END | Final del archivo           |





# Ejemplos de fseek

```
fseek(p, 500, SEEK_SET);
```

Se desplaza 500 bytes desde el inicio del archivo p

```
fseek(p, 500, SEEK_CUR);
```

Se desplaza 500 bytes desde la posición actual del archivo p

```
fseek(p, 0, SEEK_END);
```

Se desplaza 0 bytes desde el final del archivo p

```
fseek(p, 4 * sizeof(XX), 0);
```

Se desplaza 4 veces el tamaño del tipo **XX** desde el inicio del archivo p

```
fseek(p, pos * sizeof(XX), 0);
```

Se desplaza **pos** veces el tamaño del tipo **XX** desde el inicio del archivo p



# ftell

Devuelve la cantidad de bytes desde el inicio del archivo a la posición donde se encuentre el cursor al momento de ejecutar la función.

```
fseek(p, 0, SEEK_END);  
  
t = ftell(p);  
  
cr = t / sizeof(XX);
```



¿Qué tiene la variable t?

¿Qué tiene la variable cr?

¿Para qué nos sirve?

# Ejemplos en C/C++