Buffering

- Leer de archivos es lento
- Comparado con leer la memoria del computador
- Leer 1 byte toma el mismo tiempo que leer 8 KB
- La estructura FILE maneja un *buffer* de 8 KB
- Buffer: memoria para almacenar datos temporalmente
- Aunque se lean pocos bytes, internamente se lee el buffer completo
- Los próximos bytes no se leen del archivo, si no que del buffer
- Cuando se acaba el buffer, se vuelve a leer el buffer completo

Buffering

- Escribir archivos es lento
- Cuando se escriben pocos bytes, se escriben en el buffer
- Solo se escribe en disco
  - cuando se llena el buffer
  - cuando se cierra el archivo con:

    fclose(stream);
  - Cuando se invoca:

    fflush(stream);

Manejo de errores

- Practicamente todas las funciones estándares tienen una manera de indicar un error
- Fopen retorna NULL
- Fread retorna número negativo
- La siguiente función permite explicar el error al usuario

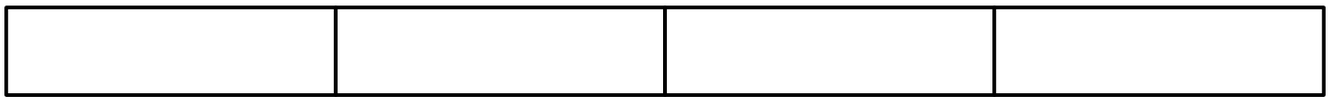
  void perror(const char *s);

- Ejemplo:

```
FILE *in= fopen(argv[1], "r");
if (in==NULL) {
  perror(argv[1]);
  exit(1);
}
```

Acceso directo: fseek

- En la estructura FILE se mantiene un cursor hacia la próxima posición en el archivo que se debe leer o escribir
- Normalmente los archivos se leen y escriben secuencialmente
- La función fseek permite leer o escribir directamente cualquier parte del archivo sin tener que haber leído o escrito lo que venía antes
- Ejemplo: considere el archivo de personas de la clase pasada



$Persona_0$ $Persona_1$ $Persona_2$ ...

- Podemos ver ese archivo como un arreglo. Se necesita una función que lea la i-ésima de persona
- Problema: programar la función leerPers que lee la i-ésima persona en archPers

```
void leerPers(FILE *archPers, Persona *pers, int i);
```

- Leer i-ésima persona del archivo

```
void leerPers(FILE *archPers, Persona *pers, int i) {
  if (fseek(archPers, i*sizeof(Persona), SEEK_SET)!=0) {
    perror("leerPers");
    exit(1);
  }
  if (fread(pers, sizeof(Persona), 1, archPers)!=1) {
    perror("leerPers");
    exit(1);
  }
}
```

- Escribir i-ésima persona del archivo:
  lo mismo, pero cambiar *fread* por *fwrite*

- Determinar tamaño del archivo

```
// Posicionar cursor al final del archivo
fseek(archPers, 0, SEEK_END);
int sizeBytes= ftell(archPers); // Solicitar posición
int numPers= sizeArch / sizeof(Persona);
```

- Ordenar el archivo de personas

```
sort(archPers, 0, numPers-1, cmpPers, swapPers);
```

- Técnicamente es O(n log(n))
- En la práctica es ineficiente porque el acceso directo en discos es lento (la constante es muy grande)
- Es pedagógico porque se puede hacer, pero no lo hagan