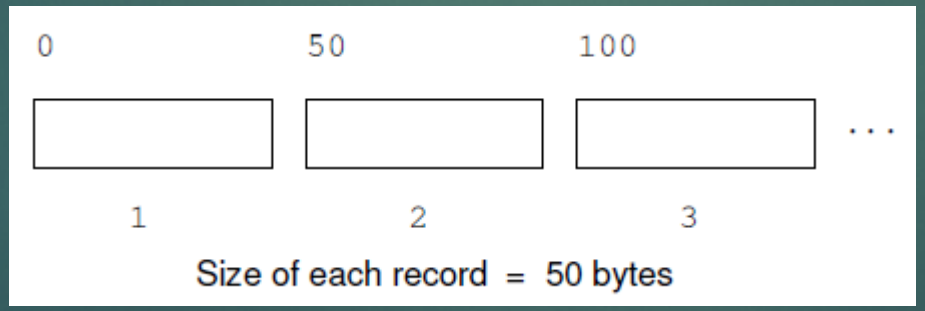
CSI 402 Systems Programming

LECTURE 16| HACKATHON

Let's Revisit Random Access Files

- No explicit support
- Functions *fread* and *fwrite* are used
  - How does this affect access time?
  - Is access time independent of position?
- Common solution: make all "records" to be of the same size



- In this example, starting position of record i = (i − 1) × size of record

Unformatted Files

- Also called **binary** files
- They cannot be viewed/edited using standard text editors
- Can be produced by a C program using "**unformatted write**" (i.e., using ***fwrite***)
- **Prototype:** *size_t fwrite(const void *ptr, size_t size, size_t nent, FILE *fp)*
  - Writes bytes from memory to a file
  - *ptr* gives the starting address in memory
  - *size* denotes the size in bytes of each element to be written
  - *nent* is the number of elements, each one with a size of *size* bytes
- Returns the number of entries written
  - If this value is less than nent, it is an indication of error
- Note: **Always** check the return value of *fwrite* to ensure that no errors occurred

fwrite Function Example

- **Prototype:** *size_t fwrite(const void *ptr, size_t size, size_t nent, FILE *fp)*
- Example:

```
1   FILE *ofp; int num = -25;
```

- How to call *fwrite*?

```
fwrite((const void *) &num, sizeof(num), 1, ofp);
```

- Why?
  - &num: starting address of num (Type: int *)
  - (const void *) &num: type casts address to const void *
  - sizeof(num): size of the entry (i.e., no. of bytes) to be written
  - 1: number of entries to be written
  - ofp: pointer to the output file

Function fread

- **Protorype:** *size_t fread (void *p, size_t size, size_t nent, FILE *fp);*
  - Reads bytes from file into memory
  - *p* provides the starting address for reading into memory
  - *size* indicates the size (i.e., number of bytes) of each entry to be read
  - *nent* denotes the number of entries to be read
  - *fp* is a pointer to the input file
- Returns the number of entries read
  - If this value is less than nent, it is an indication of error

Hands-on Exercise

- Let us store flight data in a random access file (i.e., "flights.bin")
- Each flight record is of the form:
  - AirlineCode FlightNumber OriginAirportCode DestinationAirportCode DepartureDate
  - e.g., AA43 DFW DTW Wed Jan 6 11:00 2016
- Tasks:
  - (a) Create a binary file to store info about up to 100 flights from a txt file
  - (b) Read input from user (i.e., create a menu)
  - (c) Read and print info about all flight records in the file
  - (d) Count number of airlines for a given airport
  - (e) Print the number of inbound flights for airport x
  - (f) Print the number of inbound flights for each airport
  - (g) Print a sorted list of origin airports based on the number of outbound flights
  - (h) Print a list of origin airports that have at least 2 flights that have a departure time earlier than noon
- Requirements: Use a struct to read and write data from/to the bin file