Week 11 - Friday

CS222

Last time

- What did we talk about last time?
- Exam 2!
- Before that
  - Review
  - Trees
  - File I/O

Questions?

Project 5

Quotes

The key to performance is elegance, not battalions of special cases. The terrible temptation to tweak should be resisted unless the payoff is really noticeable.

Jon Bently and M. Douglas McIlroy
Computer Scientists at Bell Labs

Binary Files

What is a binary file?

- Technically, **all** files are binary files
  - They all carry data stored in binary
- But some of those binary files are called **text files** because they are filled with human readable text
- When most people talk about binary files, they mean files with data that is only computer readable

Why use binary files?

- Wouldn't it be easier to use all human readable files?
- Binary files can be more efficient
  - In binary, all **`int`** values are the same size, usually 4 bytes
- You can also load a chunk of memory (like a WAV header) into memory with one function call

Integer	Bytes in text representation
`0`	1
`92`	2
`789`	3
`4551`	4
`10890999`	8
`204471262`	9
`-2000000000`	11

Changes to fopen()

- To specify that a file should be opened in binary mode, append a **b** to the mode string

```
FILE* file = fopen("output.dat", "wb");
```

```
FILE* file = fopen("input.dat", "rb");
```

- On some systems, the **b** has no effect
- On others, it changes how some characters are interpreted

`fread()`

- The **`fread()`** function allows you to read binary data from a file and drop it directly into memory
- It takes
  - A pointer to the memory you want to fill
  - The size of each element
  - The number of elements
  - The file pointer

```
double data[100];
FILE* file = fopen("input.dat", "rb");
fread(data, sizeof(double), 100, file);
fclose(file);
```

fwrite()

- The **fwrite()** function allows for binary writing
- It can drop an arbitrarily large chunk of data into memory at once
- It takes
  - A pointer to the memory you want to write
  - The size of each element
  - The number of elements
  - The file pointer

```
short values[50];
FILE* file = NULL;
//fill values with data
file = fopen("output.dat", "wb");
fwrite(values, sizeof(short), 50, file);
fclose(file);
```

Seeking

- Binary files can be treated almost like a big chunk of memory
- It is useful to move the location of reading or writing inside the file
  - Some file formats have header information that says where in the file you need to jump to for data
- **`fseek()`** lets you do this
- Seeking in text files is possible but much less common

fseek()

- The **`fseek()`** function takes
  - The file pointer
  - The offset to move the stream pointer (positive or negative)
  - The location the offset is relative to
- Legal locations are

`SEEK_SET`	From the beginning of the file
`SEEK_CUR`	From the current location
`SEEK_END`	From the end of the file (not always supported)

```
FILE* file = fopen("input.dat", "rb");
int offset;
fread(&offset,sizeof(int),1,file); //get offset
fseek(file, offset, SEEK_SET);
```

Example 3

- Write a program that prompts the user for an integer ***n*** and a file name
- Open the file for writing in binary
- Write the value ***n*** in binary
- Then, write the ***n*** random numbers in binary
- Close the file

Example 4

- Write a program that reads the file generated in the previous example and finds the average of the numbers
- Open the file for reading
- Read the value ***n*** in binary so you know how many numbers to read
- Read the ***n*** random numbers in binary
- Compute the average and print it out
- Close the file

Lab 11

Upcoming

Next time…

- Bitfields
- Unions

Reminders

- Keep working on Project 5
  - Due next Friday