Sequential Files

11-1-2002

Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignment?

Motivation

- So far we have only done input and output to and from the "standard" input and "standard" output. When we wanted to work with files we could redirect them, but that meant that we either couldn't input from the keyboard, or we couldn't see the output, or both.
- We'd like to be able to access files for reading and writing and still have keyboard input and screen output.

File Streams

- The way that we do this in C is with file streams. These are part of the stdio.h library that you have been using all along, we just haven't used this part of it.
- These are called streams because there is a parallel between data flowing through them and water in a stream. The data goes by once. You can't tell it to give you back data that already went by.

FILE Pointers

- The way we keep track of file stream in C is with FILE pointers. FILE is another type in C that is defined in stdio.h. It is a structure, something we will talk more about later.
- All the functions we will use work with pointers to instances of the FILE type. In your code you will declare FILE* variables to represent the streams.

Opening Files

- Before we can do anything with a file be have to have it opened. For this we use the fopen function. Do "man fopen" for details.
- This function takes two strings and returns a FILE*. The first string is the file name and the second one specifies what we want to do with it in a 2-3 character string.
  - First character is 'r' for read, 'w' for write, or 'a' for append..
  - The second character is 't' for text or 'b' for binary. We will only do text now.
  - A '+' can follow saying it is read and write.

Opening Details

- When you open a file with the 'w' option it will create a new file with that name. If one already existed, it will be deleted in the process.
- Using 'r' it won't create a file, but the file must exist. The function will return NULL if it doesn't exist already.
- The 'a' option opens for writing, but what is written is added to the end of the file.

File Input and Output

- Once you have a FILE* you want to be able to read from it or write to it. To do this you use the fprintf and fscanf functions.
- These functions work just like printf and scanf, but they take an additional first argument of a FILE* telling them what file to read from or write to.
- The stdin and stdout global variables represent standard input and output.

Closing Files

- After we are done using a file, either writing to it or reading from it, that file needs to be closed. This is done with the fclose function.
- The man pages can give you details, but this function is quite simple. It take a FILE* as the only argument and closes that file.

Code

- Now we will write code that uses files to demonstrate the usage of what we just talked about.

Minute Essay

- Write code that opens a file for writing and prints the numbers 1 through 10 to it.