Buffers, Files, and Devices

Section 5.2, Section 5.4, and Section 5.5

Buffers

- A **buffer** is temporary storage between sender and receiver of bytes in a stream
  - Sender might be sending bytes faster than receiver can handle
  - Receiver might be doing something else and unable to handle bytes at the moment
  - Buffer stores up the bytes until they're ready to be received

- **Flushing** a buffer is sending that data in it
  - Some types of buffers wait until they have a certain amount of data before flushing
    - Called block buffering
    - Common for transferring large files
  - Other types wait until a newline sent
    - Think of typing something and it sends when press enter
    - Called line buffering

- Sometimes printing to the screen is buffered
  - Might not print until a certain amount of characters are buffered or a newline character is buffered
- **fflush()** can explicitly force the buffer to flush
  - **fflush(stdout);**

- **buffered.c**
  - Demonstrates buffering is being used

Files

- Uses files on computer all the time
- Learned how to read and write files
- But what are files really?
- Can think of as a single dimensional array of bytes

File Pointer

- A **file pointer** is like a marker used to keep track of which byte currently on in reading or writing to a file
  - Pointer is used figuratively here, not literally a pointer data type, though this file pointer is contained in a FILE structure typically
- As we call **fscanf()** or **fread()** on the file, the file pointer is automatically advanced

- Can explicitly set the file pointer using **fseek()**
  - **fseek(*FILE_pointer*, *byte*, *base_value*);**
  - Base value is a constant that indicates where the *byte* offset is in relation to
    - SEEK_CUR, SEEK_SET, SEEK_END
- **ftell()** tells where file pointer currently at
  - **long int where = ftell(*FILE_pointer*);**

- **file_pointer.c**
  - Gets file as command line argument
  - Repeatedly asks user which byte to move to
  - Displays that byte

File Attributes

- A file itself really only contains its data
- Where is all the other information given by **ls -l**, like date modified, kept?
- Actually stored in tables managed by the operating system

- Can access the data using the **stat()** function
  - Need **#include <sys/stat.h>**
- Fills a structure with info stored by operating system on the file
- Most of the data encoded as numbers
  - Permissions stored as bits
  - Dates as seconds since 1 January 1970 UTC

- **file_attributes.c**
  - Calls **stat()** to fill the stat structure with information about the file
  - Displays information about the file

Directories

- **Directories** are used to group files together
- Technically, a **directory** is a list of file names
- Operating system manages directories similar to how it manages files

- Several functions and data types for working with directories
  - Most need **#include <dirent.h>**
- **DIR** data type to represent a directory
- **struct dirent** to represent an entry in a directory (either a file name or another directory)

- **opendir()** used to open a directory
  - Input is string of a directory name
  - Returns a DIR pointer
- **closedir()**
  - Input is a DIR pointer

- Once a directory has been opened, **readdir()** is used to retrieve and entry from the list of the directory's contents
  - Input is a DIR pointer
  - Returns a struct dirent pointer
- A struct dirent contains information about the entry, such as its name in its **d_name** member

- **directory.c**
  - Opens a directory with opendir()
  - Repeatedly reads entries with readdir()
  - Prints the file name of each entry
  - Closes the directory with closedir()

Devices

- A **device** is usually a peripheral or piece of hardware in the computer system
- In Unix, devices are treated like files
- So getting data from or giving data to a device is like reading from and writing to a file
- Most devices have a file representing them in **/dev**

- **tty.c**
  - Opens file to the current terminal
  - Writes text to the terminal
- **mouse.c**
  - Opens file for the mouse
  - Reads data from the mouse

Device Drivers

- A **device driver** is low-level software that interacts with the hardware
  - Typically implements functions like open(), close(), read(), write(), etc. for working with the specific hardware
- Driver needs to be compiled into the kernel or dynamically loaded module