CS 180 Notes - 2/8/16

Operating System API
- The interface between a user's program and the operating system is a series of system calls provided by the operation system.
  - -- Realistically, a user is limited to the system calls in the operating system.
  - -- System calls are made from any language that the operating system can support (and most languages can depreciate back into C)

POSIX Support
- In the list of Mostly POSIX-compliant systems, notice that Windows does not fall into this list. However, this does not mean that Windows completely rejects POSIX support (as many programs wouldn't be cross-platform).

System Calls
- A system call can often handle more than one library call
- In the case of ssize_t read(int fd, void *buf, size_t nbyte), fd usually starts as '3' (as '0' is stdin, '1' is stdout, and '2' is stderr).
- Parameters are pushed on the stack in reverse order

- All programs will call an exit(status) system call (Basically, returning to whoever called Main)

Strace Program
- To see what system calls are being made, we can use strace.
- It is also particularly useful for debugging in some scenarios
- strace should already be installed, and it defaults to stderr

Example Strace output
- Notice the amount of '3' in the file, as that is the first available file descriptor.
- Multiple library calls means it is possible to see multiple duplicate system calls

There is a program, ltrace, that lets you see library calls (and with -s, both system and library calls)

Where is the call to printf?
- The compiler likely changed it to puts, as no formatting was done.

Speed of programs
- The larger the buffer, the more effecient a program is
- Reduces the number of read/writes the system has to do.
- Some library calls (like fread) will ignore prohibitably small buffers
- For large number of calls, system call overhead will take its toll on the speed of the program.