TCP Sockets programming

Summary of what we've Learned

- The socket is created by itself. It has to be bound to the port we want for it to send and receive anything. To let it monitor incoming connection requests it has to be set to listen.
- All ip addresses must be changed to the proper binary endian format for the sockaddr_in struct. INADDR_ANY is actually all 0s, meaning the socket will listen on any interface of the system instead of only for one specific ip address.
- The kernel has a timeout period of when a socket can bind to a specific port. The socket needs to be configured to allow port reuse, otherwise a new socket can't bind to that port if the server was quit and restarted quickly.
- For connections that will be short and infrequent an iterative server is fine, for longer or frequent connection requests a concurrent server is better
- The accept function creates another socket id that should be used for actually sending and receiving. Sockets should be closed after they are done to prevent conflict.
- Read and write are similar to send and receive. They have fewer arguments needing to be passed into them, which makes them easier to use.
- Uptime only returns info for the system its run on. It returns what time it is now, how long the system has been running, how many users are logged in, and some load averages.

Capturing uptime output

1) system("argument") will send an argument as a bash command to bash. This can be used to run the uptime program and print the output to a text file. Next a file stream can be made to open the output text file with fopen("output.txt", "r"). As long as the uptime output is always the first line of the text file, the uptime can be read in with fgets() and printed.

2) popen("argument","r") will open a pipeline that is treated as a filestream. The pipeline will send an argument as a bash command to bash and r means it can be read using fgets() and printed.

Nina: 50%
Drew: 50%