Samantha Rack
CSE 30264
Homework 5

1. Three approaches for servers handling multiple clients are the iterative method, the concurrent method, and the hybrid method. With the iterative method, the server handles only one client at a time. It takes the request from the client, performs it, and then closes the connection and handles the next client. If a second client (or more) tries to connect while the server is handling a different client, then the clients are queued and must wait for the server to be finished with earlier connecting clients. With the concurrent method, the server has a passive socket that is listening for client connections. When a client is trying to connect, the server creates a new active socket for each client and handles this connection with either a new thread or a new process. This allows the server to handle multiple clients at the same time, but the server must ensure that shared data is accessed safely. In the final approach using the select call, the server accepts clients and then is informed when the client wants to write data to the server or read from the server.

2. A thread is a "semi-process" that has its own call stack and executes on its own. It does not copy the entirety of a process when executing, but instead shares memory with other threads in the process. Because a thread does not require an entire copy of the program and memory elements of it, it is more efficient than forking. Threads are also preferred because it is easier to switch between and communicate between threads than it is between forked processes. Threads are important in network programming because most servers should be able to handle multiple clients at once. Threads allow a server to accept a connection from a client, spawn a thread to handle the client's requests, and continue waiting for more clients.

3. A mutex is a mutual exclusion that creates a lock on the memory used in the execution of a program. It protects the data in a critical region by not allowing other threads to access it until the mutex is unlocked. Mutexes are used to avoid data corruption when multiple threads are trying to access and operate on the same area of memory at the same time. They prevent race conditions. Mutexes are important in network programming because of the usage of threads in servers that can handle multiple clients. The server should use mutexes to protect data that may be access by multiple threads that are handling different clients, such as buffers for receiving or sending data.

4. The select call allows a single process and thread to manage multiple clients connected to it. With select, the state of each client connected to a server is maintained to and used to determine when a client has a request for the server and needs attention. Select gives the server the ability to multiplex the requests from clients as they come in. It blocks until there is a change in the status of one of the socket descriptors that the server is using for communication with one of the clients, and then wakes up and handles the request. Select is useful for network programming because it allows a server to simultaneously service multiple clients without the need for multiple processes or threads.

5. The four basic server types are iterative connectionless, iterative connection-oriented, concurrent connectionless, and concurrent connection-oriented. The two iterative server types handle

only one client at a time. If multiple clients send requests at the same time, or a second client requests while the server is still handling a first, then these clients must wait until the server is finished with the first client. In contrast, the concurrent server types are able to handle multiple clients by using a new thread or process for each client. The server has a passive socket that is listening for a connection, and it has a new socket descriptor when each client connects. The two server types that are connectionless use UDP protocol, where the server will listen, but has no connection established before it receives requests from its client. Connection-oriented servers, though, must establish a connection with the client before they can receive and respond to requests.