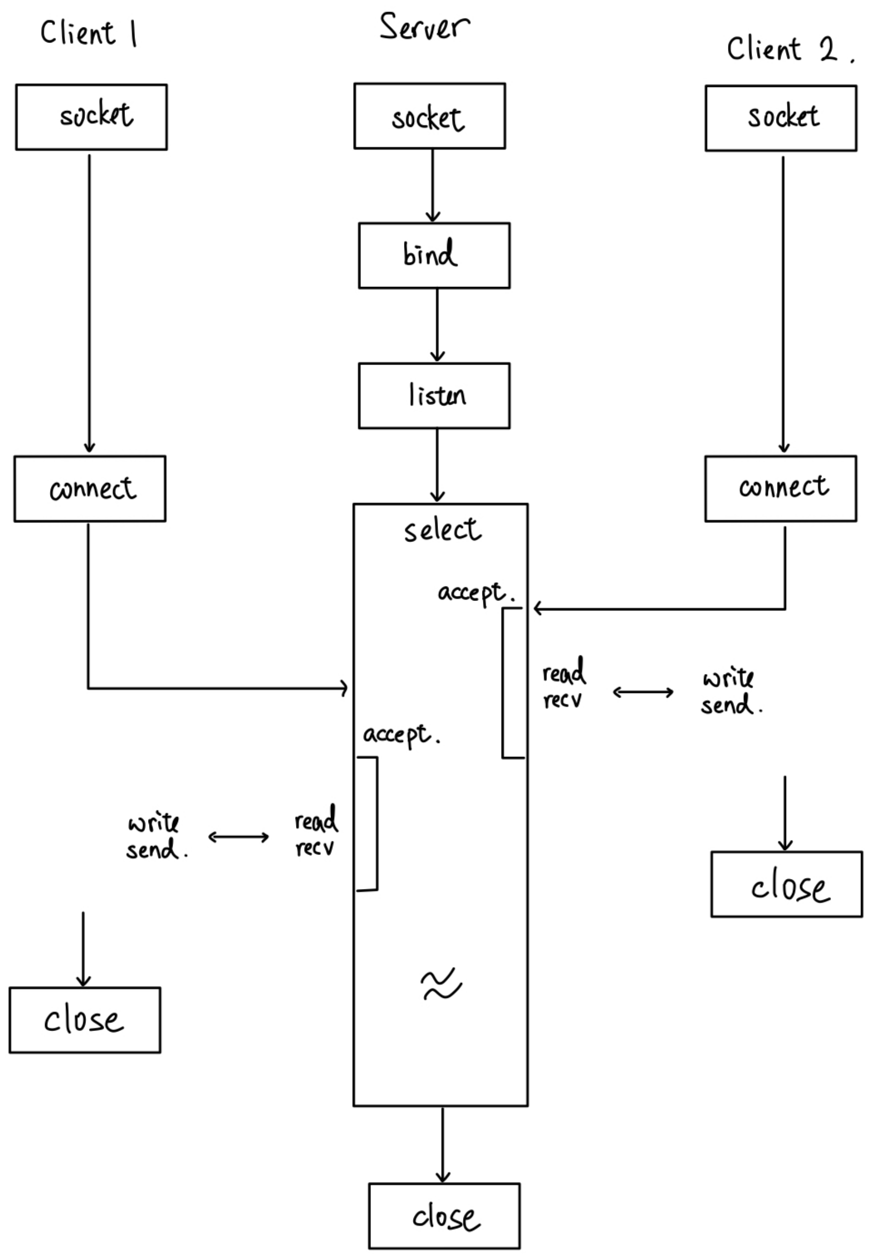
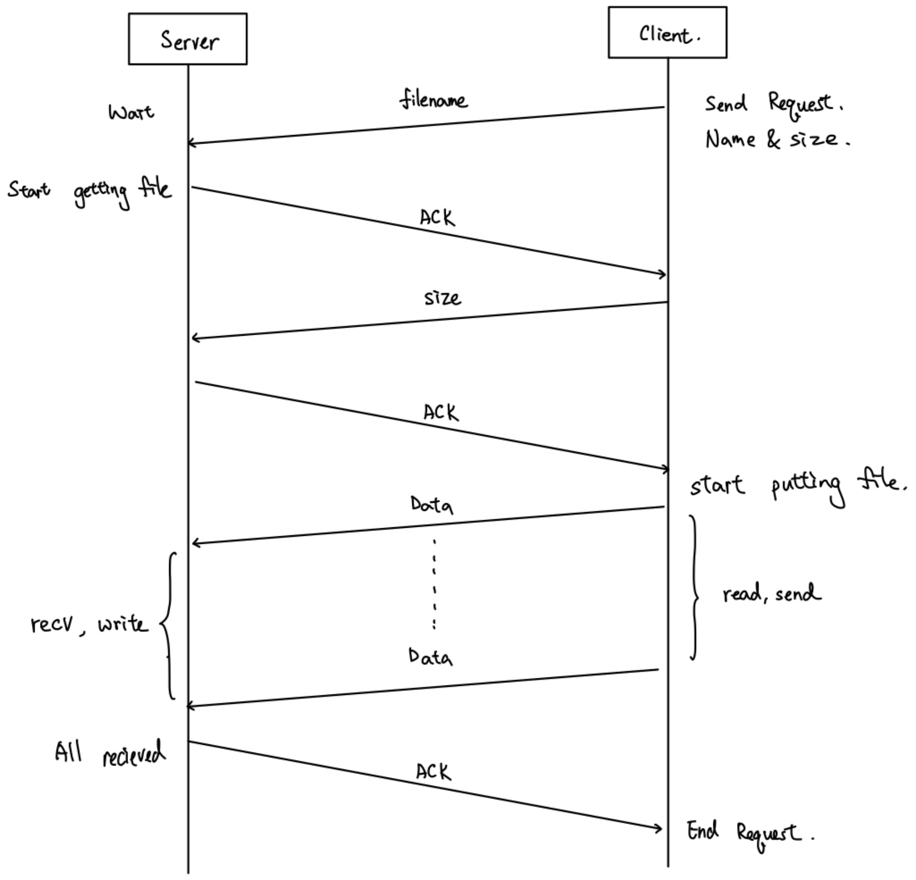
Flowchart of the connection:



After setting up server’s socket (*socket, bind, listen*), the socket called *select* to wait for the client to send request. When the client writes something to the socket, the server will know by the function *select’s read file descriptor.* This will allow I/O multiplexing. Also, when the client first make the connection, it will use the *svr.listen_fd* socket. Then calling *accept* will assign the connection a new socket for communication until the whole client process end. Later on, every application between the accepted client will communicate with the server with the assigned socket.

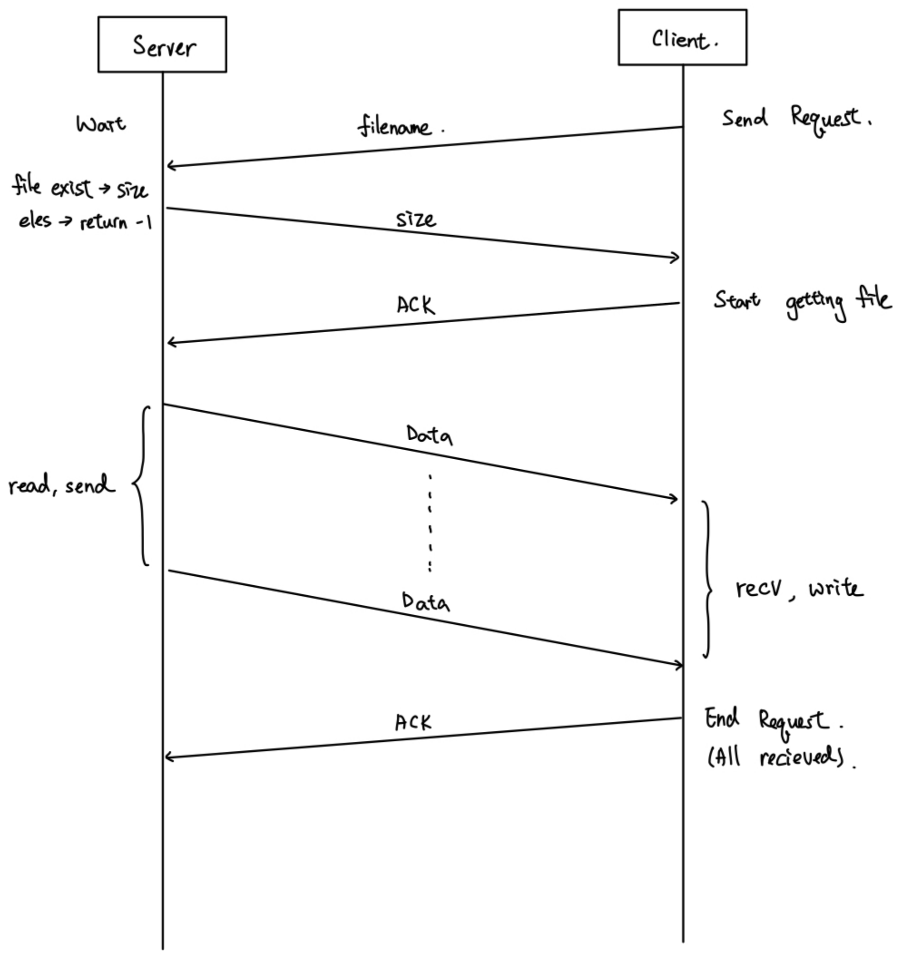
Also, multiple user is handled using different socket (one client one socket), and by select, we can perform I/O multiplexing. Select will tell which client is ready to be read, then further file transmitted or list can be performed.

Flowchart of client's put file:



Client will send filename to the server and server will send back ACK after receive filename. The story is the same for size. Then, the data will be transmitted to server through read and send from client and recv and write to server. At last, server will send an ACK to tell client that the file is transmitted successfully.

Flowchart of client’s get file:



Client will send filename to the server and server will send back the size of the file. Then, the data will be transmitted to client through read and send from server and recv and write to client. At last, client will send an ACK to tell server that the file is transmitted successfully.

What is SIGPIPE? It is possible to happen to your code? If so, how do you handle it?

SIGPIPE is a broken pipe signal and happened when someone write or read to broken or ended pipe (or socket).

When the client ends the process (etc. ^c), the server will receive a SIGPIPE signal and terminate. Since the signal routine for SIGPIPE will kill the process.

So, in order to avoid server being closed by client, I register and signal handler that disconnect the socket and unset the file descriptor (FD_CLR) with system call *signal*.