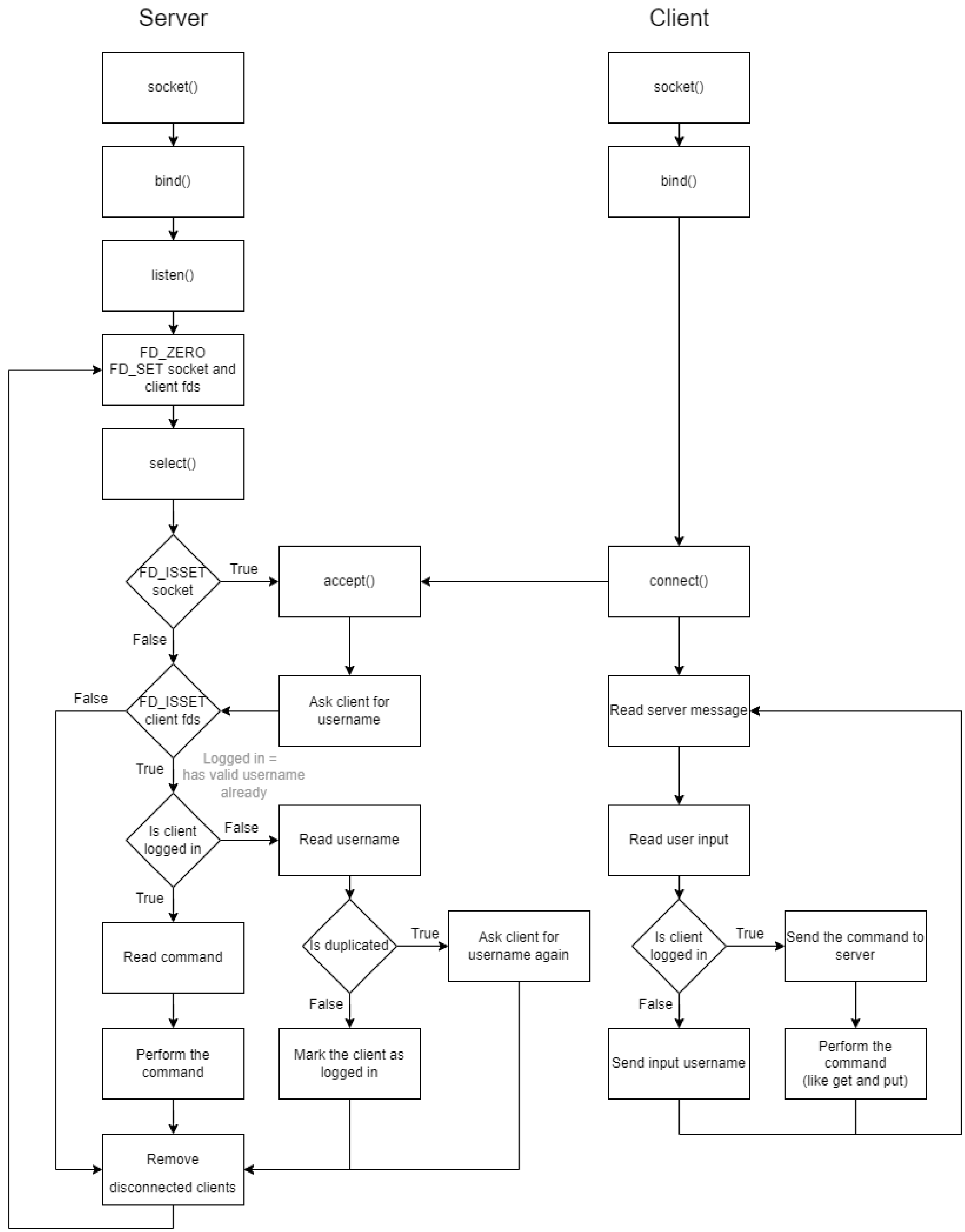
Connection:

I use `select()` to handle multiple clients.

For details, please refer to the graph below:

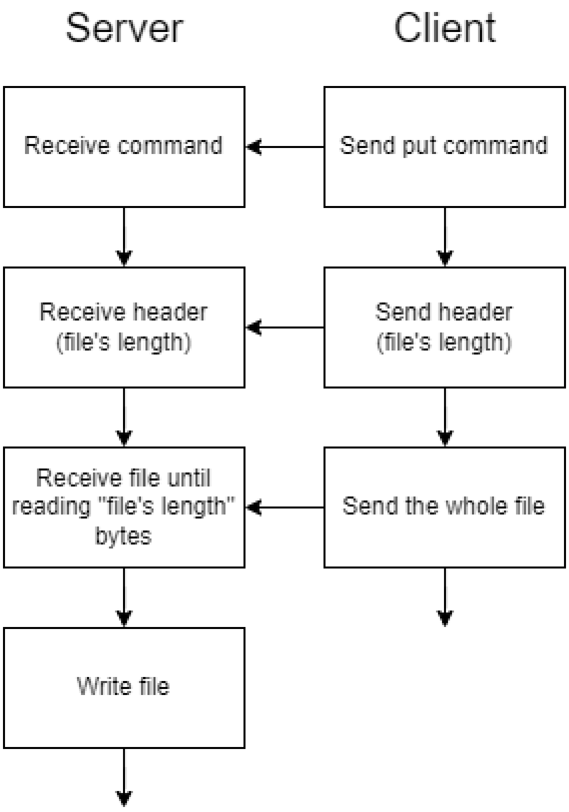


File transferring:

There are two types of implementation to complete file transferring. I use “put” command for the explaination below, but “get” command can be explained in the same way.

1.

The first approach is that when every time a client wants to put a file to server, it directly put the whole file at the first time. To complete this, the client has to send a header indicating the file’s length, so that the server can know how many bytes to read in advance.


Server
Client
Receive command
Send put command
Receive header (file's length)
Send header (file's length)
Receive file until reading "file's length" bytes
Send the whole file
Write file

The drawback of this approach is that the server can’t serve other client when the file is transferring. Especially when the file is large, other clients may notice a long delay. I use this implementation for the assignment, since it’s easier and the spec doesn’t tell us to handle this problem.

2.

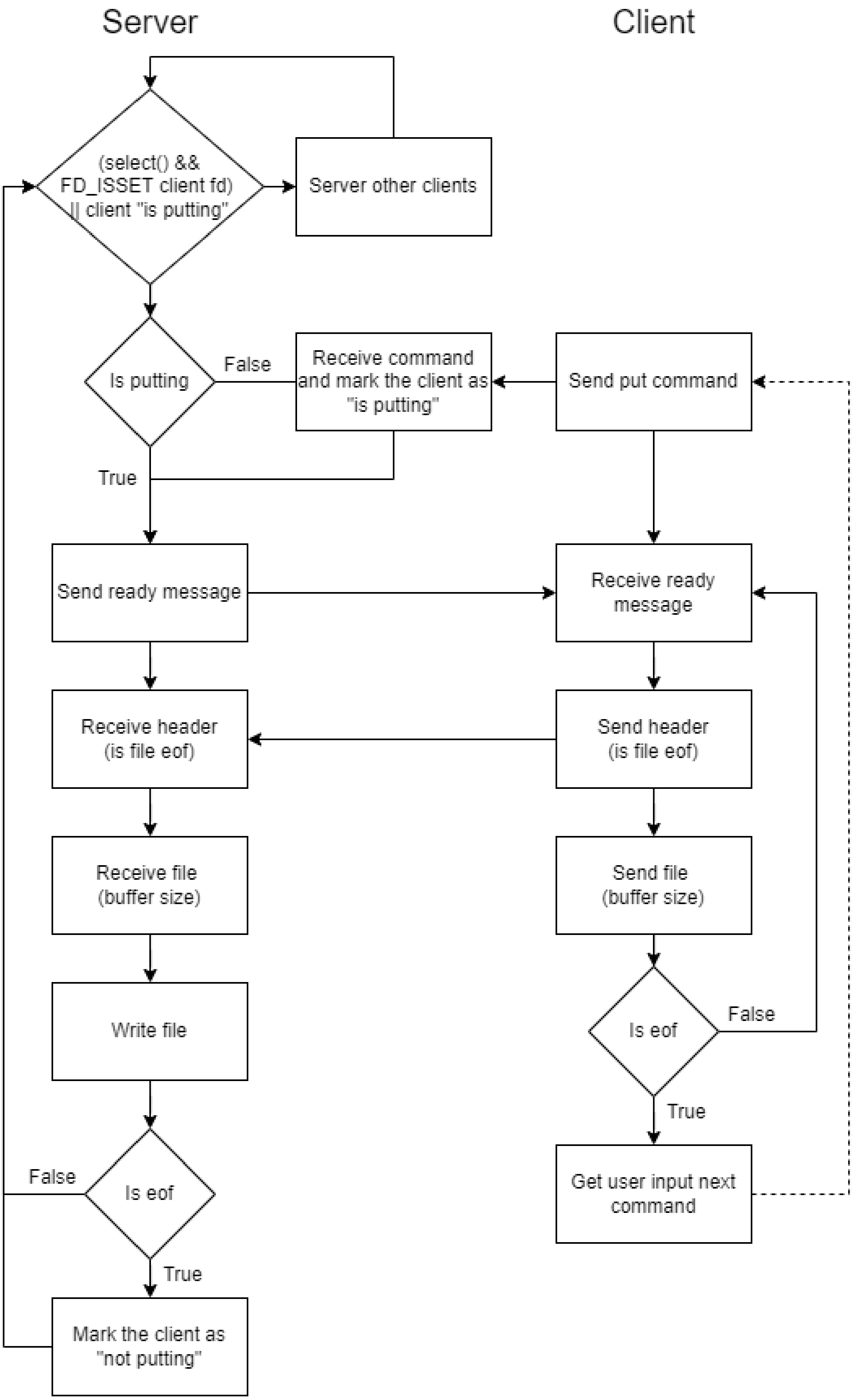
The second one is sending the file part by part. When a client wants to put a file to server, it first waits for a ready message of the server.

After receiving the ready message, the client sends a header that indicates if the file will reach eof in this transfer or not.

Next, the client send BLOCK_SIZE bytes of the file to the server and wait for the next ready message if the file hasn’t reached eof.

In the meantime of every BLOCK_SIZE file transferring, the server can serve other clients. When the server is ready to serve this file transferring again, it sends a ready message and the file transfer continues.

For details, please refer to the graph below:



The drawback of this approach is that there are more overhead of the file transferring and it is harder to implement.

SIGPIPE

When the server is trying to send a message to the client, if the client has already terminated or disconnected, then the server will receive a SIGPIPE signal. If the server didn’t handle this signal, it will termainate as well.

In this assignment, if we terminate the client when it is transferring the file, it is possible for this situation occurs.

To solve this, we can use `sigaction()` to catch SIGPIPE signal and ignore it.