Data & Communication Networks Project 1
Peter Fabinski (pnf9945)

Question 1

Concurrency was implemented using the `pthreads` library. The server starts as a single-threaded application to set up the listening socket, but then performs an infinite loop which calls `accept`, and creates a new server thread given the socket file descriptor for that individual client connection. The thread is then detached, and manages its own closing after the client has finished making requests.

By passing in the connection file descriptor for each unique client to a new pthread, they are all able to operate independently of one another, while the main server thread keeps the listening socket, creating new threads if necessary.

The project was done mostly using research from the Open Group specification, as well as "Beej's Guide to Network Programming" at `beej.us/guide/bgnet` for a higher level overview of the process.

Question 2

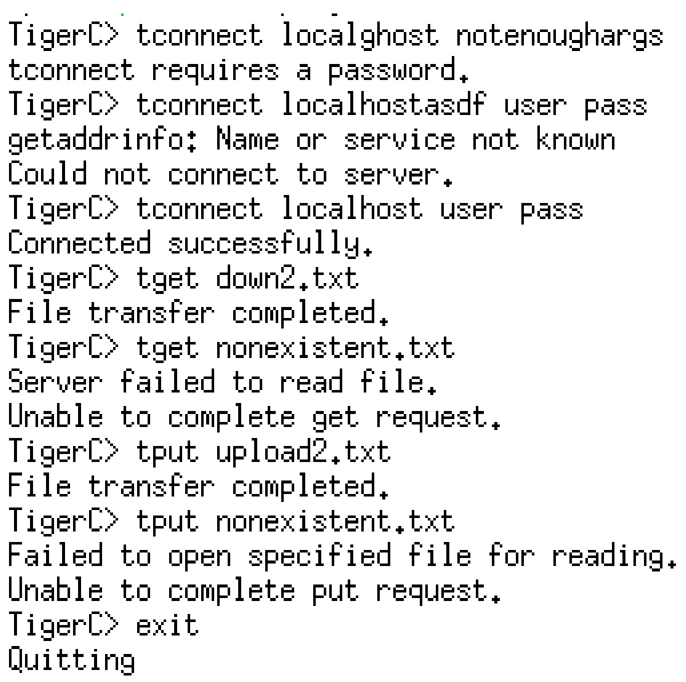
The files `client_output.txt` and `server_output.txt` show the output from the client and server, respectively, when running the provided test script which makes 100 simultaneous connections. As seen in the client output file, all the connections are made before any of the file transfers in either direction finish, indicating that they all are working at the same time. This can also be observed in the server output.

Question 3

The number of concurrent clients that the server is able to serve at once is determined by the maximum number of ports that are available at any one time. The TCP/IP spec has port numbers as 16 bits, so the maximum value here is 65535. However, there are also additional factors which make this number smaller in practice - for example, the ports less than 1024 being limited-access for privileged programs. The number of concurrent threads could also be an issue on some systems; however, on my machine, `cat /proc/sys/kernel/threads-max` results in 255196, which makes the port numbers the limiting factor.

Question 4

Figure 1 shows a sample interactive run of the program, demonstrating success and failure messages for each command (except exit, of course).



```
TigerC> tconnect localghost notenoughargs
tconnect requires a password.
TigerC> tconnect localhostasdf user pass
getaddrinfo: Name or service not known
Could not connect to server.
TigerC> tconnect localhost user pass
Connected successfully.
TigerC> tget down2.txt
File transfer completed.
TigerC> tget nonexistent.txt
Server failed to read file.
Unable to complete get request.
TigerC> tput upload2.txt
File transfer completed.
TigerC> tput nonexistent.txt
Failed to open specified file for reading.
Unable to complete put request.
TigerC> exit
Quitting
```

Figure 1: Sample client output.