Brent Yelle

CSCI 3240

Project 4 Documentation

Problem-Solving Approach

After my dissatisfaction with the organization of my client-side program for Project 3, I rewrote the client so that it can be in a consistent loop of *read → write → read → write* and ad infinitum until the user decides to quit. I achieved this through these major changes:

1. The server will only ever send a newline character `'\n'` in a message at the conclusion of a series of messages to the client, when it is about to ask for input from the client.
2. In all other cases, when the server wants the client to print a newline client-side but doesn't want to ask for input, it instead sends a tab character `'\t'`.
3. When the client receives a message, it reads through every character in the received string, and if that character is a tab `'\t'`, it replaces it with a newline `'\n'` before printing it client-side.
4. The client looks at its own message that is sent to the server, scanning it to see if it starts with a digit. If it does, it reads in that digit, and if it is equal to 2 (the "quit" code for Project 4), it exits the *read-write-read-write* loop and closes its connection to the server. (This has an unintended side effect of killing the server thread when the server thread is not, since the server thread is not anticipating a break in connection. At the very least, this is safer than the alternative of leaving the server thread going forever.)

For implementing the thread-based concurrency, I followed the pattern established in slides 13 and 14 of the Lecture 18-19 slides, except that instead of merely passing a pointer the connection file descriptor, I passed a pointer to a `struct` called `ClientInfo` containing said file descriptor in addition to the client's hostname and port name.

Data Structures

In addition to what was defined for Project 3, I defined a new struct (mentioned above) called `ClientInfo`, which was dynamically allocated in `main()` and filled with the information of any new client.

It was then passed to the `thread(…)` function by address, and there its `connfd` was used to read and write from that specific client. Finally, when all the communication with the client was finished, it used the hostname and port name to print a message saying that the connection was closed, and finally the object was de-allocated using `free(…)` to avoid a memory leak.

Algorithms

No algorithms are changed between this program and Project 3's programs.

User-Defined Functions

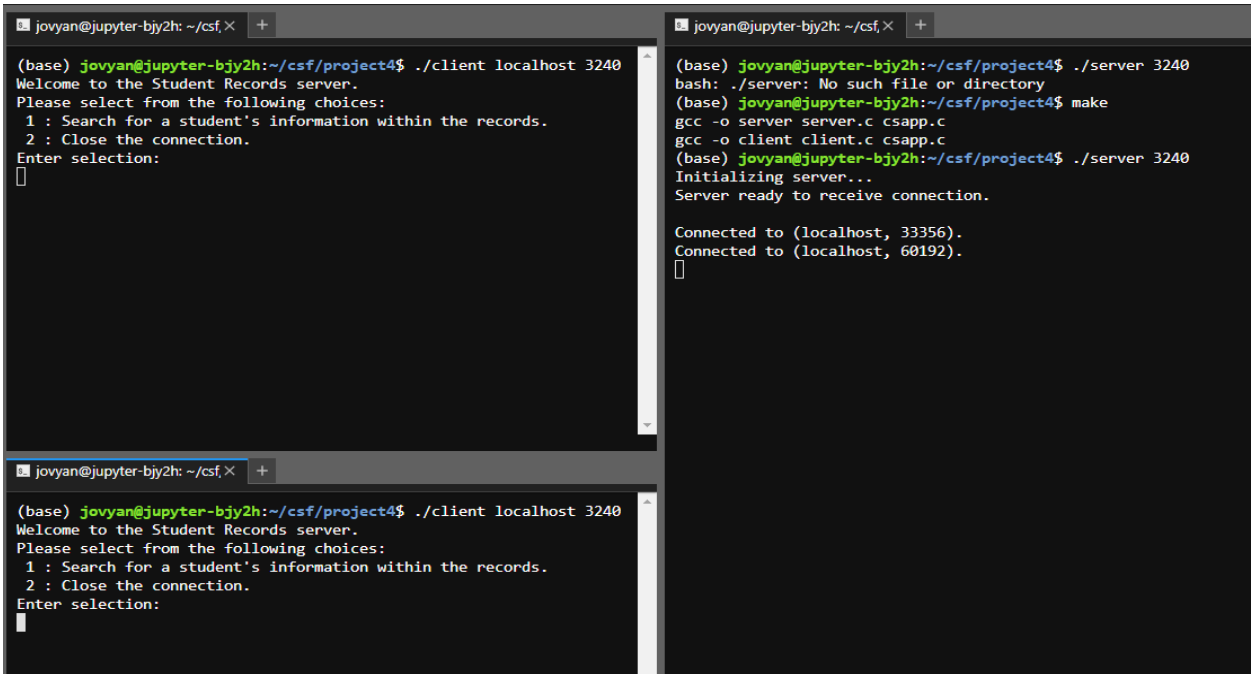
Only one new function was defined for Project 4 that was not present in Project 3: the `thread(…)` function in `server.c`, which performs the server's service. Instead of running the root of the service in the `while`-loop of `main()`, it is called as a new thread using `pthread_create(…)` whenever a new connection is made to a client.

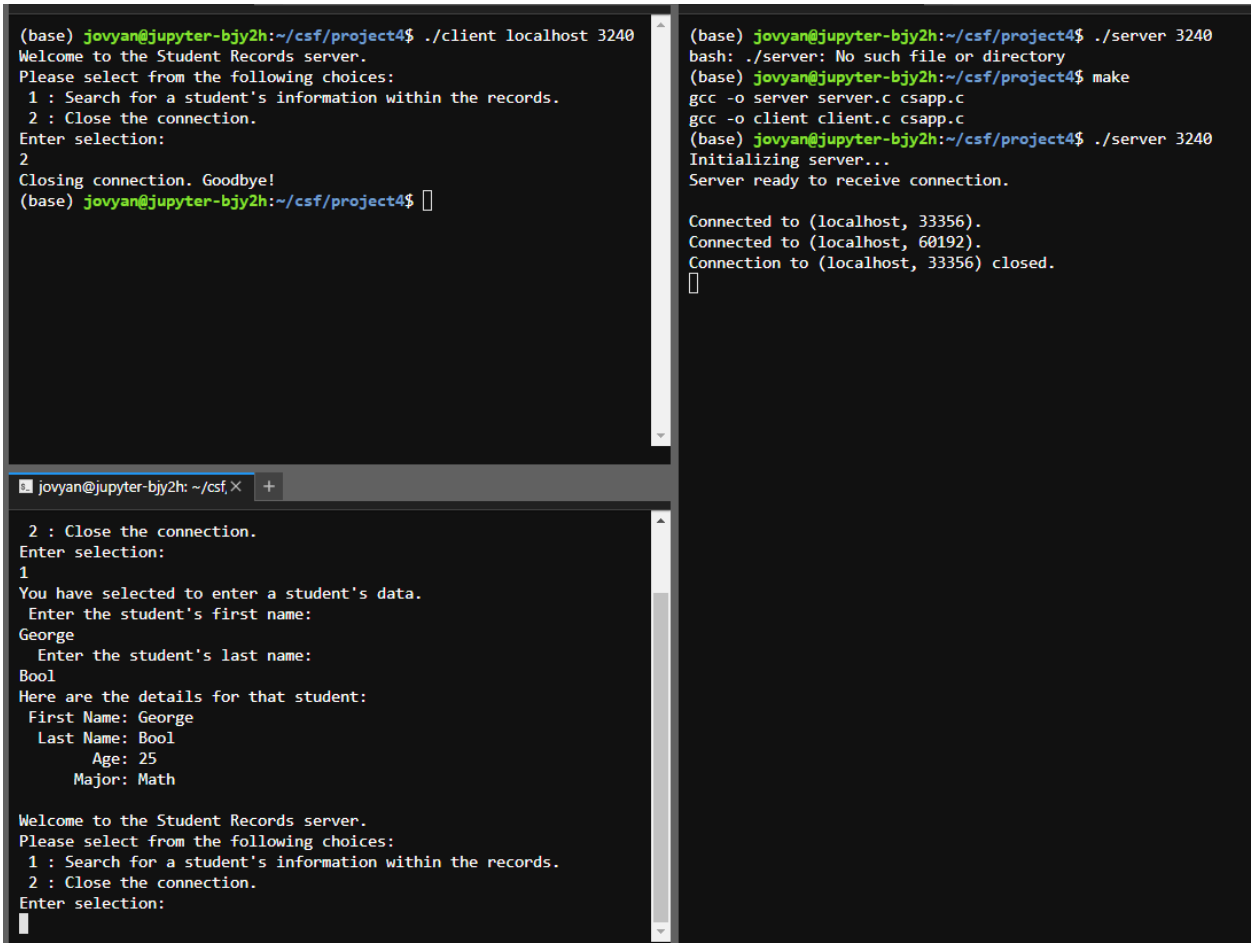
In addition, the various functions and statements related to printing things on the server-side terminal or to writing new data to `studentRecords.txt` were either commented out or left unused in the code.

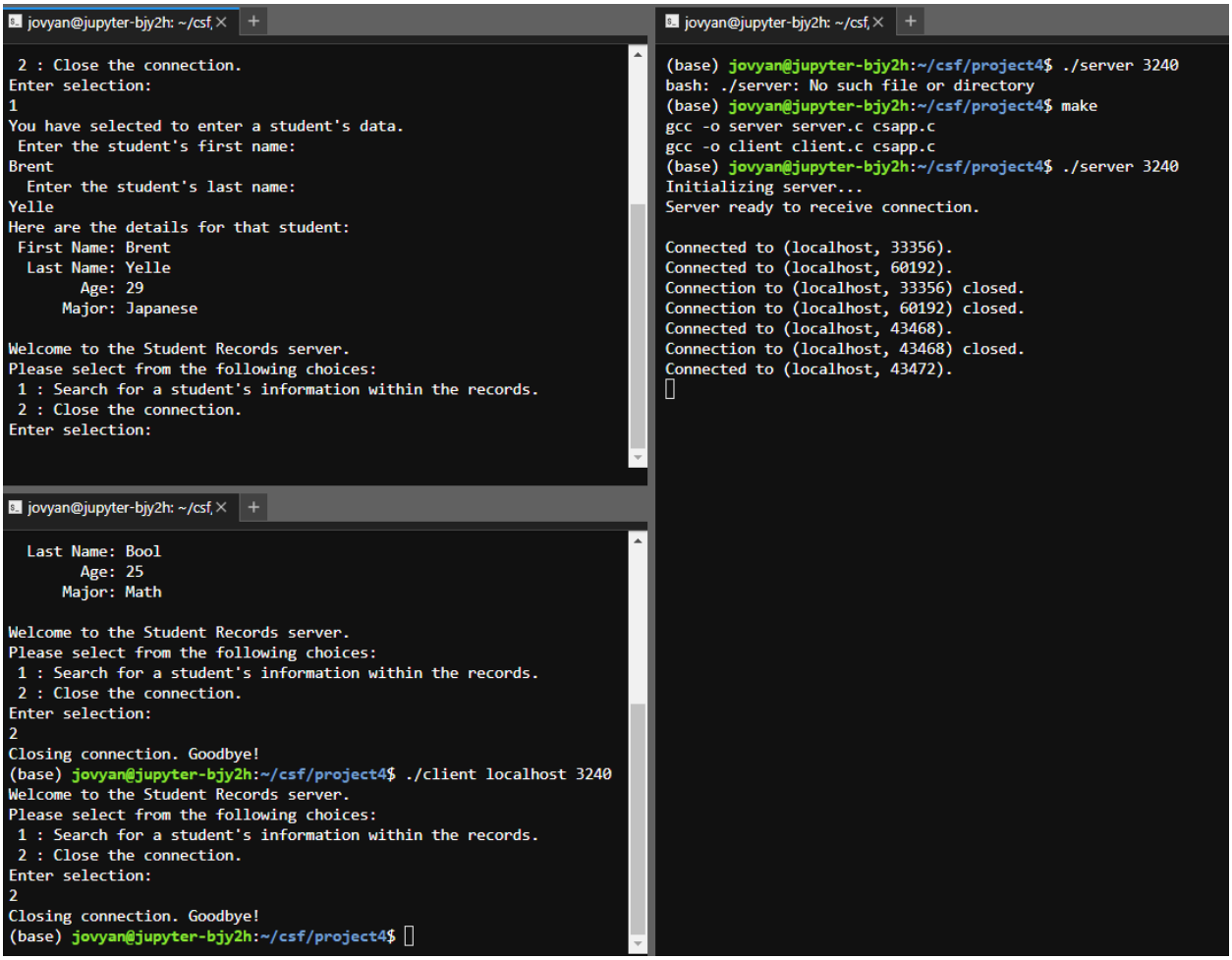
Challenges

Thanks to the Lecture 18-19 slide notes, it took me under an hour to implement the project with very little difficulty, and `make` successfully compiled the code every time, which felt amazing.

Screenshots:






jovyan@jupyter-bjy2h: ~/csf
2 : Close the connection.
Enter selection:
1
You have selected to enter a student's data.
Enter the student's first name:
Brent
Enter the student's last name:
Yelle
Here are the details for that student:
First Name: Brent
Last Name: Yelle
Age: 29
Major: Japanese
Welcome to the Student Records server.
Please select from the following choices:
1 : Search for a student's information within the records.
2 : Close the connection.
Enter selection:
jovyan@jupyter-bjy2h: ~/csf
Last Name: Bool
Age: 25
Major: Math
Welcome to the Student Records server.
Please select from the following choices:
1 : Search for a student's information within the records.
2 : Close the connection.
Enter selection:
2
Closing connection. Goodbye!
(base) jovyan@jupyter-bjy2h:~/csf/project4$ ./client localhost 3240
Welcome to the Student Records server.
Please select from the following choices:
1 : Search for a student's information within the records.
2 : Close the connection.
Enter selection:
2
Closing connection. Goodbye!
(base) jovyan@jupyter-bjy2h:~/csf/project4$
jovyan@jupyter-bjy2h: ~/csf
(base) jovyan@jupyter-bjy2h:~/csf/project4$ ./server 3240
bash: ./server: No such file or directory
(base) jovyan@jupyter-bjy2h:~/csf/project4$ make
gcc -o server server.c csapp.c
gcc -o client client.c csapp.c
(base) jovyan@jupyter-bjy2h:~/csf/project4$ ./server 3240
Initializing server...
Server ready to receive connection.
Connected to (localhost, 33356).
Connected to (localhost, 60192).
Connection to (localhost, 33356) closed.
Connection to (localhost, 60192) closed.
Connected to (localhost, 43468).
Connection to (localhost, 43468) closed.
Connected to (localhost, 43472).