README - Shubham Rustagi

The source code for client and server are respectively in the src/client and src/server folders. There is also a src/common folder which contains helper methods used throughout the client-server interaction. The helpers.c file in src/common is where most of the code exists and comments above each method indicate the purpose for that particular function. The file has been compartmentalized into sections. There are sections dealing with reading/writing files, working with .Manifest files, working with .Commit files, working with .Update files, etc. For the sake of brevity, I have not listed the functions or their purpose in the README since it is all contained within the src/common/helpers.c file and would be too extensive to reiterate.

Thread synchronization:

Multithreading the server occurs in server/main.c in the "handle_connection" function. I have created a linked list of structs called project_t. Each struct has an associated project name and a respective lock. When a command is called from the client, a lock is placed on the respective project. Thus, no other client can interact with this project until the project is unlocked. The command then executes and only after execution is the project unlocked, allowing another thread to access the project. This is an efficient way to do multithreading because the entire repository on the server isn't being locked - only a single project the user is trying to interact with is locked. Thus, multiple clients can connect with the server at the same time and be able to execute commands on unique projects.

Another feature of multithreading the server occurs in the function "handle_connection". The "handle_connection" function first places a lock on the linked list of project_t structs, calls the function "get_proj_info", and then unlocks it. The reason this is done is because "get_proj_info" will insert in a project name if it's not found in the linked list. Thus, since duplicate project names aren't desired, each thread is only able to access this linked list one at a time.

Testing:

- All tests done were done using bash scripts which can be found in tests/scripts/ and on success, the Makefile prints a "PASS" for the corresponding test and on fail, a "FAIL" is printed.

- More details on testing can be found in testcases.txt and testplan.txt.

- To run the suite of test cases, enter the bash command: "make clean; make test".

- Note: You must run "make clean" each time before the test cases can proceed.

How to run:

- Simply enter the bash command: "make" to generate two executables, "WTF" and "WTFserver" located in the bin folder.

- Then run the server from one terminal and the client from another terminal and enter in appropriate commands.

- If you are unsure on how to use a particular command, please see one of the shell scripts for the respective command in tests/scripts.

Extra Credit:

- All old versions of the projects are kept at the repository level and are compressed. (10 pts)

- All files sent between the client and server only involve 1 file being sent/received. This was done by using the tar command, invoked from system. (20 pts)