WTF Thread Design

cy287, zu9

Setup

For each connection (client) to the server, we created a thread for the connection to run on so multithreaded can be performed. In addition, we stored all of these threads in to a linked list struct to perform pthread_join in the event of errors or abrupt closing of the server. Furthermore, we used a mutex to add these threads to the global linked list to prevent race conditions from occurring and ensuring every thread we created will be stored. On closing, we loop through the linked list and pthread_join any threads still in the linked list that were not already pthread_exit.

Commands

Every server command requires access to a project and either does reading or modification to the project. Therefore for every server command there must be a mutex locked to prevent the desired project from being accessed while it is either being read or modified and unlocked when the operation is done. For instance, if client1 sent currentversion of project1 and client2 sent destroy project1, we do not want a race condition to occur where client1 recieves the message the server found the project and is able to send the contents and then client2's thread proceeds to delete it and then client1 resumes and errors out since the project no longer exists. In addition, we have a global file struct linked list that receives a list of files from the client and prepares the files to be sent to the client which needs a mutex to prevent other projects from adding to this list when it is being used by another project. In addition, for commit, push, and destroy, we have a global active commit linked list that needs to be locked. So in the event destroy destroys all the active commits for that project, push will then search the active commits linked list to find it is gone rather than the race condition between destroy and commit/push.

Extra Credit

We implemented the extra credit regarding compressing old versions of the project in rollback.