All methods and implementations for this project have been completed and should be fully functioning. The hardest part of the project was setting it up. We faced a lot of problems trying to create thread and mutex locks. However after researching more on how multithreading works, we were able to successfully implement these.

- For multithreading, we made sure to thread the portion where the server is actively listening to any requests from the client and will continue to do so until a signal interruption.
- For mutex locks, we made sure to apply mutex locks before and after any reading and writing portions of our code.

Some header files we used helped us significantly. For example, sha256.h, ftp.h, and mtp.h all came in handy when implementing our functions.

sha256.h

- This header file made sure that we would properly make hashcode in sha256 encryption.
- It was important for us to use this in our manifest files to compare older and new files between server and client.

ftp.h

- this file allowed us to transfer single files between the server and client. We had an encodeFile which was in the format of…
    - <fileName>:<content>
- for decodeFile, we used strtok to make sure that we were able to tokenize the file name and content.

mtp.h

- this method allowed us to send important messages between server and client.
- in these message interactions, we made sure to send and receive the bytes of each message before receiving the content. This would prevent us from losing any data or gathering too much.
- we also have a sendCompress method which allowed us to **complete the extra credit portion of the project.**

With these three methods, we were able to communicate with the server and client what command the client wants allowing the server to provide with a proper output to the client.

We were able to finish all the methods besides rollback and push. We had a working commit, however, in the rush of merging the last few parts of our code together, we replaced the code and lost most of the code for commit

Some big methods

Checkout was one of the more challenging ones since we decided to aim for the extra credit opportunity. We were able to recycle the -r flag from our filecompressor (Asst2) as a template to gather all files and directories into a zipped file. After we were able to do this, we sent the entire zipped file over to the client where the file is unzipped and remade on the client side.

rollback was somewhat challenging in that many flags were implemented for our project. Despite potentially not needing some of them, we thought it was necessary for us to do so. These flags prevented the previous versions to overlap into the next push. Therefore, we are able to provide an accurate manifest and rollback file each time we call it. For this method, we had a single for loop that looked for the same version that we wish to roll back to. Should the version not be there, our flag would not be changed in value resulting in a failure message sent to the client. After finding the version, we mark the flag which allows us to make sure we are in the correct version.

upgrade was challenging in the aspect where i had to tokenize and compare the manifest with the upgrade file. We struggled a lot trying to figure out the algorithm with the looping in order to make sure to ignore D tags and include A and M tags.

update was hard to understand but once we were able to get some basic logic and error cases down, it was very smooth. We first made two arrays of the manifests and compared their values using nested for loops. Next, we added them to the .update based on the conditions given in the instructions.

commit is very similar to update. However, the fail cases for this one where much harder. We tried our best to catch every possible scenario. We first made two arrays of the manifests and compared their values using nested for loops. Next, we added them to the .update based on the conditions given in the instructions.

push was very complicated and the only method we did not finish. Our push ends with creating a manifest with doubles and does not replace files. We have commented out version of the rollback addition in the file, however, it did not print right and quite often caused there to be segfaults. Therefore, we decided to comment it out. Our code is able to change manifest correctly but prints out doubles.