Below is the information pertaining to the contributions made by each team member and the test cases ran on the client and server.

**Overview:**

For this project, both team members decided to work on their own versions of the TFTP server code implementation with the plan to identify the better of the two codes to use for the submission. Both team members were successful in developing and implementing their code with successful compilation of their codes. As a team, we decided to use Neha’s TFTP server code for the submission. Attached you will find the TFTP server code, ChatGPT associated codes, and this README file. For the rest of the project, James completed all of the test runs of the code, developed the README file and the *makefile*. Additionally, James implemented the WRQ functionality.

Precursor:

Prior to running any test cases, the team needed to verify that all code was working correctly and would compile. The *makefile* is the created for the original source code file called *tftp\_server.c*; therefore, it isn’t reflected in the below example.

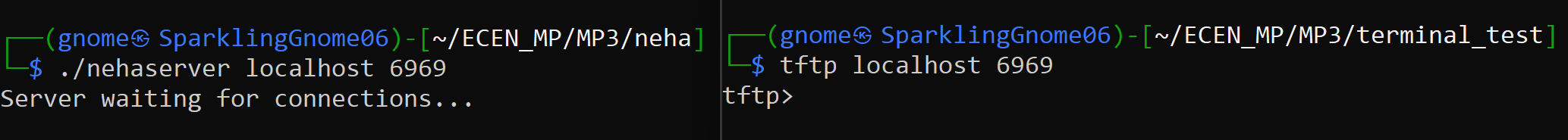
A computer screen with white text and green text

Description automatically generated

As shown above, an executable file called *nehaserver* was created, which will be used throughout the duration of the test cases. The code is identical to the original source file called *tftp\_server.c*.

Beginning Stages:

Overall syntax for running the executable: *./<executable> <IP address> <Port Number>*



Test Cases:

**Test Case #1: Transfer a binary file of 2048 bytes and check that it matches the source file**

A black screen with white text

Description automatically generatedServer: Client:

A screen shot of a computer

Description automatically generated

**Test Case #2: Transfer a binary file of 2047 bytes and check that it matches the source file**

Server: Client:

A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

**Test Case #3: Transfer a netascii file that includes two CR’s and check that the resulting file matches the input file**

Server: Client:

A computer screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

**Test Case #4: Transfer a binary file of 34 MB and see if block number wrap-around works**

Server: Client:

A computer screen shot of a black screen

Description automatically generated

A black background with white text

Description automatically generated

**Test Case #5: Check that you receive an error message if you try to transfer a file that does not exist and that your server cleans up and the child process exits.**

Server: Client:

A screen shot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

**Test Case #6: Connect to the TFTP server with three clients simultaneously and test that the transfers work correctly (you will probably need a big file to have them all running at the same time)**

A screenshot of a computer program

Description automatically generatedServer: Client(s):

A screen shot of a computer

Description automatically generated

**Test Case #7: Terminate the TFTP client in the middle of a transfer and see if your TFTP server recognizes after 10 timeouts that the client is no longer there (you will need a big file)**

A screen shot of a computer

Description automatically generatedServer: Client:

A black background with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

As you can see in the image on the left, the server recognized that the client left the session and was unsuccessful in transferring the file.

**Test Case #8: BONUS – Implement WRQ function to write files to the TFTP Server:**

Below is a screenshot of the WRQ implementation between one client terminal and the server.

Previous Directory:

A screen shot of a computer

Description automatically generated

Server Start / connected:

A screen shot of a computer

Description automatically generated

Client Terminal / Transfer:

A black screen with white text

Description automatically generated

Updated directory after the transfer:

A black background with white text

Description automatically generated

Even though the transfer “timed out” the file was still created in the server working directory. The new file that was created by the server during the transfer is “*WRQ\_test.txt*”.