# Stor Command

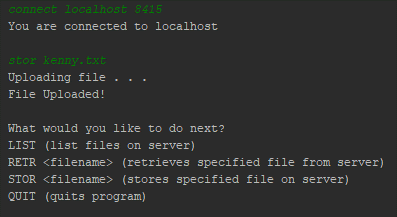


Figure 1 Client Output

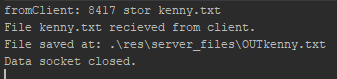


Figure 2 Server Output

**Overview of function logic**

This operation begins on the client side when the user enters the command: *STOR <filename>* on the command line. This command is passed to the stor function, where it uses a string tokenizer to get the filename, and ensure that the file exists in the client’s directory. Next the client creates a ServerSocket object to listen on. The client then uses the outputStream on the control connection to send the server a message consisting of the port on which the client will be listening, and the command that the user entered (in this case STOR <filename>).

Control now passes to the server side. On the server side the server first deletes it’s copy of the file if it already exists. Next it attempts to create a file to which it will save the content passed to it by the client. If this is successful it opens a data socket to the client. If it is unsuccessful it sends an error message on the connection port to the client, and both programs return. The server also creates a StringBuffer to which it will store the contents received from the client.

Back on the client, the client now opens the file to be transferred, reads it line by line and sends it over the data port to the server.

The server receives the data from the client, writes it to a string buffer, and then writes it to the file. If this operation is successful the server sends a message to the client informing it of success. The client then notifies the user of success, and the data connection is closed by the server.

**Challenges in writing this function**

While writing this function I encountered a few difficulties. When transferring lines with the .readUTF() and .writeUTF() functions I was having problems with line breaks not being sent. I overcame this by appending blank lines with System.getProperty("line.separator")) function. I also had difficulties with socket bind errors when two clients connected to the server. I realized that I had failed to close the data socket at the end of my functions. Another challenge was communicating to the client when the server failed to create a file, before the data socket was created. Originally, I had passed this information over the data socket, but it’s not possible to do this before the data socket is created. I overcame this by passing the relevant information over the control socket instead.