Sarwaan Ansari

CNT4007C

20 February 2019

Project 1 Report

qclient:

qclient initializes socket programming by using the arguments to get the hostname and port and creates a TCP connection to the server it connects to. If the program successfully connects, the program goes into an endless loop until the server is closed (‘k’), quit interrupt, or by typing the message ‘q’ to quit the client. The infinite loop calls a printfull command that can print out strings that are larger than the packet buffer (which is set to 80 characters), this ensures the large questions are able to be called in the program. The infinite loop can call two different type of interfaces: choicesinterface, questioninterface, and getquestioninterface. The choicesinterface allows choices to be entered until a ‘.\n.\n’ is entered. The question interface allows a question to be entered until a ‘.\n’ is entered. The getquestioninterface ensures that the full size of a ‘g’ (get question) command is returned.

qserver:

qserver initializes a socket to connect to a qclient with the hostname and port number it prints out. After initializing the TCP connection, qserver goes in an endless loop until a quit hardware interrupt (control-c) on either the qserver or qclient, or a ‘k’ command issued by the qclient it is connected to. The endless loop reads the input from the qclient program and compares it to

the list of commands it has. If it obtains a command, it calls the necessary function to execute the command.

q: This implies that the client wants to quit the connection. The qclient is closed and the server is now looking for another connection with the same init method to initially connect to the qclient.

p: This command implies that the client wants to create a question. This calls a series of functions in order to fulfill the command. The createQuestion method obtains the number of questions and creates/appends a questions.txt file, the file that holds all of the questions. The client obtains the question tag, question body, question answer choices, and question answer individually to the server and the server adds each of the components will be appended to the questions.txt file with a tag at the front of each component (e.g. <ca> implies question answers).

h: Simply prints out the commands that the server can accept.

g [number]: Obtains the question with the supplied number from the question database. qserver will call getQuestion, which simply reads all the lines in questions.txt and looks for the question number after the given tag (e.g. question 1 will be on a line <st>1). Then prints out the rest of the lines after the question line until it reaches an '<end>' tag. All periods are discarded.

c: Obtains a question number and answer and uses a similar method to read a file as g, however when the question is found, it only looks for the correct answer tag and compares it to the input of qclient. If the answers match, then it prints out that the answer is correct to the qclient, otherwise it prints out that the answer is wrong to the qclient.

d: Deletes a question from questions.txt. The deleteques() function reads through the file and writes a temporary file except when '<st>#' comes up, then the delete function waits until the next '<end>' tag comes up and continues reading. The function then deletes questions.txt and renames the temporary file to questions.txt.

r: Obtains a random question. A random number is generated, and It calls getQuestion(sockfd, random number, 1). The 1 signifies that it is a random question and it should omit the correct answer. It then obtains the qclient's input for the question answer and checks if it is correct.

k: Simply calls exit(0).

If no commands is inputted or the wrong command is inputted, then the program asks the client to try again.