**Introduction**

This document outlines the System Architecture, Data Management, and additional relevant information about the design of the QuizServer C program. This program is mixture between multiplexing and multithreading server architecture. The process of connecting clients unto the server is handled via multiplexing methods, whilst activities are run through threads.

**System Architecture**

Clients connecting into the server is handled through the ‘main’ thread. The thread creates the main socket in which new clients will connect through. When a client connects, the socket in which the client connected through is added to the list of files descriptors. Once connected, the server will listen to each of the clients using the ‘select’ method and checking through each file descriptor. This thread handles a user creating a group and cancelling its creation, joining a group, and leaving a group before the activity starts.

So that the client is able to switch between threads, the file descriptor set for the main thread has been declared as a global variable ‘afds\_main’. Additionally, a Group structure array is declared globally in order to keep track of and store the existing groups.

Once the maximum number of players is reached by a group, the activity can start. At this point, all of the client’s file descriptor within the group is removed from the main thread and added unto another, quiz thread. A quiz thread is created for each instance of a started activity. Each thread handles the distribution of questions to the players within a group, and processes the answers given by them.

Once the activity ends, the file descriptors of each of the remaining group members is added back unto the main thread, and the quiz thread is destroyed.

Three additional functions have been created for the purpose of helping handle the server:

* get\_quiz()
* process\_answer()
* remove\_group()

The ‘get\_quiz` function has as argument the string ‘quiztext’, which contains the entire text of a single quiz and returns a Quiz data structure. This function separates the text into an array of questions and their potential answers, and an array of the correct answers.

The ‘process\_answer’ function is used to determine whether a given answer is correct. It takes as argument the user’s answer as a char, and the expected correct answer to the question. Returns 1 if the two match and 0 if not. If the user provides an invalid option, it will return -1.

The ‘remove\_group’ function is used to remove a group from the list of open groups in the server. It takes the file descriptor of the createor of the group to be removed as an argument, removes, and then shifts the entire global array of groups to accommodate for the missing group. The global variable ‘group\_index’ is decreased by one.

**Data Structures**

For the purposes of data maintenance, this program makes use of 3 main data structures, defined as follows

* Player
* Quiz
* Group

The Player structure contains information regarding client and potential player of the Quiz Server. Once said player joins a group and a quiz starts, their score is stored within this data structure. Lastly, the socket in which the client is connected to is stored in order to identify each unique user

Table

Description automatically generated

The Quiz structure contains the relevant information regarding a quiz. The number of questions is stored as an int. The questions and its possible answers are stored in a string array. The correct answers for each question are stored in a char array.

Text

Description automatically generated

The Group structure contains the information regarding an existing group. The name is an unique identifier for each specific group. The maximum size and number current players are stored in integer variables. The file descriptor for the creator of the Group is stored for the purpose of identifying the admin. Finally, the Quiz the specified group will use for its activity is stored in its own Quiz structure instance.

Text

Description automatically generated with medium confidence