Fancy Quiz Server

OS Final Homework/Project

Alibek Manabayev

## Objectives

* Improve coding skills in C
* Apply theoretical knowledge in threading and multitasking
* In particular, use advantages of the multiplexing and threading in specific task

## Plan

The main plan is “divide and conquer". In particular, try to divide task between different threads, but keeping in mind flexibility and easy implementability. There will be 3 types of the threads: main loop, “hub” thread, and “quiz-group” thread. The first, main loop will be responsible only for accepting new clients and forwarding them to correct thread. Next “hub” thread is for the clients that just arrived and for those who hasn’t joined any group. Let’s call them “free clients”. And finally, there will be separate assigned thread for each quiz-group, whose task is simply handle quiz process: wait for the clients, send questions to them and keep scoreboard.

## General Structure

Generally, project can be divided to 3 parts independent parts, one part for each thread type: Main function, Hub and quiz-group threads.

First part is the simplest, code from the previous homework can be used here.

For the second part, I’m going to use multiplexing for the smoother handling of the clients. If this part is done in the right way, last quiz section will be simple. This part will create new threads for the new quiz-groups and forward clients to the dedicated thread.

Last part is the best, more information will be known in the future reports. :D

P.S. All ideas and the concepts are subject to change :)

## Structure of the data

The following block contains list of the structures, main global variables and function prototypes that will be used in the project.

|  |
| --- |
| 2. struct question { 3. char \* str; 4. int q\_len; 5. char \* ans; 6. int a\_len; 7. } 9. struct client { 10. int id; 11. int ssock; 12. char \* name; 13. int n\_len; 14. int point; 15. quiz\_group \* group; 16. } 18. struct quiz\_group { 19. int id; 20. char \* name; 21. int n\_len; 22. int current\_size; 23. int dedicated\_size; 24. client \* admin; 25. client \* members; *// array for the members of the group* 26. } 28. *// 0 is for the "hub".* 29. *// NULL, if there is no such* 30. quiz\_group all\_groups[33]; 31. *// Mutex for the each group, so object can be safely used* 32. pthread\_mutex\_t groups\_mutex[33]; 33. pthread\_t threads[33]; 34. int last\_thread 36. *// there will be at most 1020 clients* 37. client all\_clients[1024]; 39. *// In order to normalize string = remove trailing \r\n* 40. int normalize(char \* str, int len); 41. int normalize(char \* str); 42. *// Thread-safe parsing of the arguments by "|"* 43. int parse\_args(char \* str, char \*\* args); 44. *// Sort members by their points* 45. void sort\_members(quiz\_group \* q\_group); 47. *// reads message from client terminated with CRLF* 48. *// returns status of read* 49. *// 0 - ok* 50. *// 1 - error, occured* 51. int read\_msg\_cr(client \* cl, char \* str); 52. *// reads message from client with given size in the message* 53. *// returns status of read* 54. int read\_msg\_size(client \* cl, char \* str); 56. *// Safely (mutex) adds and removes member to the group* 57. *// returns status of the operation:* 58. *// 0 - ok* 59. *// add: 1 - no such group, 2 - full group, 3 - already there* 60. *// remove: 1 - no such group, 2 - no such client* 61. int add\_member(char \* group\_name, clinet \* cl); 62. int remove\_member(char \* group\_name, clinet \* cl); 64. *// Creates client, checks if there is already* 65. client \* create\_member(int ssock); 66. *// Close connection, remove from groups, and all lists* 67. void close\_connection(client \* cl); 69. *// Hub function* 70. void \* hub(void \* args); 71. *// quiz-group function* 72. void \* quiz\_group(void \* args); 73. *// main function :D* 74. int main(int argc, char \*argv[]); |