Сушкова Дарья Сергеевна, БПИ204, ДЗ-5, вариант 26.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА.

Описание задачи:

26. Задача про экзамен. Преподаватель проводит экзамен у группы студентов. Каждый студент получает свой билет, сообщает его номер и готовит письменный ответ. Подготовив ответ, он передает его преподавателю. Преподаватель просматривает ответ и сообщает студенту оценку. Требуется создать многопоточное приложение, моделирующее действия преподавателя и студентов. При решении использовать парадигму «клиент-сервер».

Структура проекта:

exam/

CMakeLists.txt

main.cpp

Student.h

Student.cpp

Teacher.h

Teacher.cpp

TicketsDeck.h

TicketsDeck.cpp

Основные характеристики программы:

- Число модульных файлов: 3
- Число файлов реализации: 4
- Размер исходных текстов: 9.22 Кб

Инструментальные средства:

- Виртуальная машина Oracle VM VirtualBox (на Windows 10):
  - ОС: Linux (Ubuntu (64-bit));
  - Кол-во процессоров виртуальной машины: 2;
  - Оперативная память: 2048 МБ;
- Языки программирования: C/ C++;
- Библиотеки: iostream, fstream, string, unistd, pthread, sstream, random, utility, unordered_map, set, semaphore;
- Интегрированная среда разработки: CLion;
- Средства сборки проектов: CMake.

Принцип работы:

В задании было указано, что необходимо использовать парадигму «Клиент-Сервер». Данное описание модели: это способ взаимодействия неравноправных потоков. Клиентский поток запрашивает сервер и ждет ответа. Серверный поток ожидает запроса от клиента, затем действует в соответствии с поступившим запросом.

В задаче про экзамен в качестве сервера примем преподавателя, а клиенты – отдельные потоки – студенты. В ходе «экзамена» преподаватель ожидает запросы от студентов – получение билета/ ответ на полученный вопрос. В зависимости от того, какой запрос поступил, преподаватель выбирает вариант действия – отдача билета/ оценивание ответа.

Конкретнее с учетом именований в программе: Teacher (в роли сервера) раздает билеты, принимает ответы и журналирует их с оценками (журнал в конце работы программы записывается в выходной файл); каждый Student (в роли клиента), получает билет, отвечает, получает оценку (в общем, взаимодействует с сервером). Каждый Student – отдельный поток, Teacher – в главном потоке (main). Разделяемый ресурс – стопка билетов (TicketsDeck) – доступ к ней синхронизируется с помощью семафора и мьютекса.

В ходе программы фактически:

- Создается Учитель, который будет принимать экзамен и у которого есть стопка из К билетов.
- Создаются М студентов, которые запрашивают у Учителя билеты.
- Если билеты кончились (К < М), оставшиеся студенты ждут, пока кто-нибудь не сдаст экзамен и не вернет билет.
- Как только в колоде появляются билеты, их разбирают оставшиеся студенты (и так пока все студенты не получат свой билет).
- Билет студент получает рандомный из оставшихся.
- Учитель ставит оценку рандомно от 2 до 5.

Учитывая выше изложенное, модель «Клиент-Сервер» подходит для данной задаче лучше всего.

Ввод/ вывод:

- Ввод данных осуществляется двумя способами:
  - Командная строка: -s*count-of-students* -t*count-of-tickets*;
  - Файл: два числа через пробел – количество студентов и количество билетов;
- В командную строку данные можно передать тремя способами (описание – ниже в примерах, а также в самой программе при выводе инструкции);
- Если программа запускается без аргументов, то предполагается, что она использует файлы ввода (input.txt) и вывода (output.txt) по умолчанию;

Примеры запуска программы:

- exam -s35 -t20 - запускает программу с аргументами "35 студентов" и "20 билетов", вывод в output.txt;
- exam (запуск без аргументов) - запускает программу, которая читает два числа из input.txt, вывод в output.txt по умолчанию;
- exam -o file.txt - запускает программу, которая читает два числа из input.txt, вывод в file.txt.

В выводе программы – журнал преподавателя с порядком оценивания и всеми данными.

Примеры файлов input.txt и output.txt лежат в папке проекта.