Архитектура вычислительных систем.

Задание 5. Вариант 24.

Выполнила: Словягина Анна Михайловна

Студентка 2-го курса программной инженерии

группа БПИ207

Описание задания

Описание задания: разработать многопоточное приложение с использованием указанной парадигмы, библиотеки POSIX Threads языка программирования C или стандартной библиотеки языка C++. Программа должна быть написана на языке C или C++, компилироваться и выполняться в ОС Lnux. Стиль написания программы - свободный.

Условие: задача о привлекательной студентке. У одной очень привлекательной студентки есть N поклонников. Традиционно в день св. Валентина очень привлекательная студентка проводит романтический вечер с одним из поклонников. Счастливый избранник заранее не известен. С утра очень привлекательная студентка получает N «валентинок» с различными вариантами романтического вечера. Выбрав наиболее заманчивое предложение, студентка извещает счастливчика о своем согласии, а остальных - об отказе. Требуется создать многопоточное приложение, моделирующее поведение студентки. При решении использовать парадигму «клиент-сервер» с активным ожиданием.

Примечание: в данной задаче выбор поклонника происходит в зависимости от длины приглашения. У студентки богатый выбор, поэтому она выбирает тех, кто написал не менее 20-ти символов, но так как наша героиня очень занятая, у нее нет времени читать длинные приглашения, поэтому она также смотрит, чтобы в приглашении было не более 30-ти символов. Если ни одно из приглашений не пришлось ей по вкусу, то она выбирает последнее отправленное приглашение.

Парадигма «клиент-сервер»

Парадигма «клиент-сервер» является одной из наиболее распространенных моделей. В данной концепции участвуют две стороны: клиенты и сервер или серверы. Клиентский процесс запрашивает (возможно, неоднократно) данные у сервера и ожидает ответа, затем использует полученные данные по своему усмотрению. Серверный процесс ожидает запроса от клиента, далее в соответствии с поступившим запросом обрабатывает данные и возвращает запросившему их процессу- клиенту. Модель клиент-сервер предоставляет возможность разграничить поставленные задачи и работу над вычислениями между теми, кто заказывает услуги и теми, кто их поставляет.

Реализация модели «клиент-сервер» в приложении: в данной программе студентка является сервером, а ее поклонники – клиентами. В основной функции программы считывается количество поклонников и создается количество потоков равное этому числу. Каждый поток отображает собой одного клиента (то есть поклонника), помимо этого создается поток, который является «сервером». После инициализации потоков, каждый из них считывает валентинку из файла или вводит рандомно (зависит от выбора пользователя). Для того, чтобы все клиенты смогли правильно считать данные из файла, используется мьютекс, который блокирует другие потоки, пока один из них имеет доступ к файлу (помимо этого мьютекс также используется, когда какой-либо поток производит запись в файл или вывод в консоль). После считывания данных поток заполняет соответствующую ячейку массива и отправляет запрос серверу с номером ячейки. Для того, чтобы и клиентский и серверный поток не использовали одновременно одну и ту же область памяти, используются семафоры: сервер ждет пока счетчик семафора уменьшится и это и является сигналом того, что ячейка массива заполнена и он может к ней обращаться, после проверки приглашения сервер использует второй семафор, чтобы разрешить клиенту выполнять дальнейшие действия. После получения ответа от сервера клиент выводит данные в файл и на экран.

Используемые источники:

1. Грегори Р. Эндрюс. Основы многопоточного, параллельного и распределенного программирования. - М.: Издательский дом "Вильямс",2003 <https://l.wzm.me/_coder/custom/parallel.programming/main.htm>
2. О модели взаимодействия клиент-сервер простыми словами. Архитектура «клиент-сервер» с примерами - <https://zametkinapolyah.ru/servera-i-protokoly/o-modeli-vzaimodejstviya-klient-server-prostymi-slovami-arxitektura-klient-server-s-primerami.html>
3. Парадигмы параллельных приложений - <https://studref.com/702392/informatika/paradigmy_parallelnyh_prilozheniy>
4. Клиент-сервер - о технологии простыми словами - <https://galtsystems.com/blog/start/klient_server_o_tekhnologii_prostymi_slovami/>

Входные и выходные данные

Аргументы командной строки. пользователю необходимо ввести 3 аргумента в командной строке. Первый аргумент отвечает за то, будет ли ввод производиться из файла или он будет рандомным (“-f” – из файла, “-n” – рандомный). Если пользователь выбрал рандомный ввод, то следующим аргументом должно быть число поклонников (данное число может быть от 1 до 30 включительно), иначе пользователь должен ввести путь к файлу, из которого будет происходить ввод (например input.txt). Третьим аргументом должен быть выходной файл в обоих случаях. Собрать проект можно с помощью команды – g++-11 -std=c++20 main.cpp -o main -lpthread

Таким образом запуск программы может выглядеть вот так:

g++-11 -std=c++20 main.cpp -o main -lpthread

./main -n 5 Tests/outputR1.txt

./main -f Tests/input1.txt Tests/output1.txt

Входные данные: если ввод происходит из файла, то в первой строке файла должно быть указано число – количество поклонников. После этого идет n строк (n – число, указанное в первой строке файла), в которых первым аргументом написано имя поклонника и далее через пробел идет приглашение (обратите внимание, что само приглашение написано без пробелов, сплошным текстом). При рандомной генерации ничего дополнительно вводить не нужно, кроме аргументов командной строки.

Выходные данные: на каждом этапе выводится сообщение на экран в формате: Invitation number “n” from “name” with text – “invitation text” was

1. read – приглашение считано с файла
2. generated – приглашение сгенерировано рандомно
3. sent – приглашение отправлено на сервер
4. denied – сервер отказал
5. approved – сервер одобрил приглашение
6. answered negatively – клиенту ответили отрицательно и он делает об этом запись
7. answered positively – клиенту ответили положительно и он делает об этом запись

В выходном файле находится окончательный результат без промежуточных сообщений (перечислены все приглашения и указано, кому было отказано и кто был выбран).

По номеру приглашения можно понять, что программа действительно является многопоточным приложением. Рассмотрим пример:



Исходя из него видно, что как только поток захватывает управление, он не передает его другому потоку, пока не завершится (хотя передача управления другим потокам возможна, просто из-за мьютексов, другие потоки не успевают перехватить управление). Если мы посмотрим на номера приглашений, то увидим, что действительно порядок не определен и мы заранее не знаем какой поток первый перехватит управление.

Таким образом, в данной работе реализовано «клиент-серверное» многопоточное приложение, которое использует мьютексы и семафоры для синхронизации потоков.