Домашнее задание №5. Многопоточность

Давыдов Вячеслав Олегович

БПИ 208

1 Вариант 11

AVS_Davydov_BPI208_291_11_21_HW5

Карфаген должен быть разрушен

Хочеца сказать что я все корни этих потоков вспомнил

Архитектура ВС. Многопоточность

Давыдов Вячеслав Олегович, вариант 11

Описание задания

Задача о гостинице - 1. В гостинице 30 номеров, клиенты гостиницы снимают номер на одну ночь, если в гостинице нет свободных номеров, клиенты устраиваются на ночлег рядом с гостиницей и ждут, пока любой номер не освободится. Создать многопоточное приложение, моделирующее работу гостиниц

Описание реализации

Программа написана на C++17 в объектно-ориентированном стиле. На ubuntu. Многопоточность реализована с использованием средств стандартной библиотеки C++ вперремешку с POSIX. Такой мулат был выпущен мною поскольку с плюсовыми мутексами я никак не мог совладать. Вообще я весь код хотел на ++ написать, вот и потоки я начал писать по ++ библиотеке, но мутексы из нее и слип работали как-то совсем непонятно. Вы можете увидеть мои закомменченные, так и нереализованные заготовки в коде. В целом буду рад если вы скажите в чем причина того, что эти плюсовые мутексы не функционируют. А еще я замучился жесть с передачей в структуру ofstream-а. Это вообще возможно? Я над этим минимум 3 часа сидел. В итоге просто создал поле в мейне.

Использована система сборки CMake. Исходные коды в папке 'src'. Программа реагирует на ошибки ввода, выводя в стандартный поток вывода соответствующее сообщение.

Программа полностью комментирует происходящее с потоками.

Модель реализации

Использовалась реализация парадигмы параллельного программирования, называемая "Портфель задач". Реализуется с помощью разделяемой переменной, количестве мест в отеле, ну иными словами ресепшн. Доступ к нему в один момент времени имеет только один поток(клиент).

Есть условная переменная и мьютекс. Обращение к условной переменной происходит только в критических секция. При переполненности гостиницы будет вызываться метод pthread_cond_wait, а при освобождении места - pthread_cond_broadcast(типо notify all как я понял), который зазывает всех спящих на улице клиентов в гостиницу. А вообще планировалось использовать unique_lock и его wait с предикатом, но звезды не сошлись этой ночью.

Когда клиент приходит на ресепшн, он синхронизируется. Поэтому ни один другой клиент не может его вытолкнуть. То есть заселение гостей происходит в порядке хоть и не упорядоченной очереди, но никто хотя бы не спорит. После бронирования номера, клиент рассинхронизируется с ресепшеном, освобождая место следущему и идет спать. Время сна реализовано в виде usleep на определенное количество времени, которое задается входным параметром. В это время другий клиенты выполняют ту же последовательность действий. Сразу проснувшись клиент не идет курить на балкон, а идет на ресеппшн, съезжать. И опять там выстроится очередь выспавшихся посетителей с одной целью - съехать. Поэтому нам снова на помощь приходит синхронизация! Синхронизируемся на ресепшене и дальше все аналогично бронированию, разве что перед тем как покинуть стойку регистрации, хостес идет на улицу и кричит в микрофон всем бедным уличным бродягам, которым не хватило места (если таковые имеются разумеется в силу заполненности гостиницы), о том что место освободилось. Но так как ресепшн у нас один, то сначала все кто стоит в очереди съедут, и только потом, в какое-нибудь окно, заселится другой.

При всем при этом надо понимать, что есть еще клиенты, которые не успели еще доехать до гостиницы, и разумеется они будут конкурировать за места с бродягами.

Программа закончит свое выполнение, когда каждый желающий провести свою ночь в гостиничном номере удовлетвориться.

Формат ввода и вывода

Входные параметры:

1. Количество клиентов.
2. Доина ночи в милисекундах.
3. Файл вывода

Пример ввода:

Ввод из файла условно выглядит так: ./main -f input.txt output.txt

Если через консоль, то: ./main -n 100 5000 output.txt

Пример вывода:

```
 Client 85: book room for the night in the hotel.
Hotel: full for 27/30 rooms.

Client 87: book room for the night in the hotel.
Hotel: full for 28/30 rooms.

Client 91: book room for the night in the hotel.
Hotel: full for 29/30 rooms.

Client 83: book room for the night in the hotel.
Hotel: full for 30/30 rooms.

Client 82: wait outside due to hotel is full.

Client 74: wait outside due to hotel is full.

Client 80: wait outside due to hotel is full.

Client 96: wait outside due to hotel is full.

Client 77: wait outside due to hotel is full.

Client 86: wait outside due to hotel is full.

Client 78: wait outside due to hotel is full.

Client 98: leaves hotel.
Hotel: full for 29/30 rooms.

Client 77: book room for the night in the hotel.
Hotel: full for 30/30 rooms.

Client 38: leaves hotel.
Hotel: full for 29/30 rooms.
```

P.S.

Я так понял, что вы в этой задаче не требуете каких-то данных о времени работы программы и не простие составлять статистику. Тем не менее, если вам это будет интересно, я добавил функционал получения времени, которое проработает программа, перед тем, как гостиница станет свободной.