Пояснительная записка к домашнему заданию №5 по архитектуре вычислительных систем

Выполнила: Куликова Татьяна Дмитриевна, БПИ-206

Вариант 11

11. 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 гостиницы.

1. Краткое описание работы программы

Требование к входным данным:

В качестве входного параметра пользователю необходимо в командной строке указать одно значение – количество посетителей гостиницы. Данное значение должно быть больше 0 и не больше 100. При некорректном вводе аргументов командной строки программа выведет сообщение об ошибке.

Далее в процессе работы программы будут выводиться сообщения о происходящих процессах. Сообщения двух видов: 1 – “Visitor number <visitor number> goes to room number [room number] at <время>” – сообщение о поселении конкретного клиента в конкретный номер гостиницы, 2 – “Room number <room number>: visitor number <visitor number> goes out at <время>” – сообщение о выселении клиента из его номера.

2. Описание модели, которая использовалась при решении задачи>

Для решения задачи была выбрана модель “Производитель – потребитель”.

Ее суть состоит в том, что процесс-производитель выполняет вычисления и выводит поток результатов. Процесс-потребитель вводит и анализирует поток значений. Многие программы в той или иной форме являются производителями и/или потребителями. Сочетание становится особенно интересным, если производители и потребители объединены в *конвейер* — последовательность процессов, в которой каждый из них потребляет данные выхода предшественника и производит данные для последующего процесса. Классическим примером являются конвейеры в операционной системе Unix. Процесс-производитель ожидает (при необходимости), пока в буфере появится свободное место, затем добавляет в конец буфера новую строку. Процесс-потребитель ожидает (при необходимости), пока в буфере не появится строка данных, затем забирает ее. (информация взята из книги **“Основы многопоточного, параллельного и распределенного программирования” Грегори Эндрюса**).

Теперь рассмотрим данную модель применительно к задаче про гостиницы. Здесь буфер (общий ресурс) – это целочисленный массив размера 30, в котором хранятся номера клиентов, на данный момент находящихся в комнате с номером <индекс ячейки + 1>, номера клиентов – целые числа от 1 до N (количество клиентов), номера комнат – целые числа от 1 до 30. Потоки-производители – это потоки-клиенты, которые записывают номер клиента в свободную ячейку буфера, увеличивают количество занятых комнат и выводят информацию об этом. Потоки-потребители – это потоки-комнаты, которые освобождают ячейку, соответствующую номеру комнаты (присваивают ей значение 0), увеличивают количество свободных ячеек и выводят сообщение о выселении посетителя.

Для контроля доступа к общему ресурсу были использованы мьютексы и семафоры.

Мьютекс (англ. *mutex*, от ***mut****ual* ***ex****clusion*—«взаимное исключение»)—это базовый механизм синхронизации. Он предназначен для организации взаимоисключающего доступа к общим данным для нескольких потоков с использованием барьеров памяти (для простоты можно считать мьютекс дверью, ведущей к общим данным). То есть, мьютекс служит для того, чтобы в один момент времени в конкретный блок кода мог зайти только один поток.

Семафор – это объект, который используется для контроля доступа нескольких потоков до общего ресурса. В общем случае это какая-то переменная, состояние которой изменяется каждым из потоков. Текущее состояние переменной определяет доступ к ресурсам. В данной программе присутствуют семафоры full – количество занятых ячеек и empty – количество свободных ячеек, поток-производитель увеличивает на 1 значение full и уменьшает на 1 значение empty, поток-потребитель, наоборот, увеличивает empty и уменьшает full. Когда значения данных семафоров достигают предельных допустимых значений, то при full = 30, empty = 0 блокируются все потоки-производители (посетители), при full = 0, empty = 30 блокируются все потоки-потребители (комнаты). Потоки разблокируются тогда, когда вызовется sem_post и значения семафоров перестанут быть равными предельным значениям.

Общий принцип работы приложения

Сначала в цикле создаются и запускаются все потоки-посетители, а затем все потоки-комнаты. У каждого потока вызывается его стартовый метод, в котором содержится блокировка (мьютексы). Сначала посетители в произвольном порядке заполняют все комнаты отеля (сначала кого-то селят в первую комнату, потом во вторую и так далее в каждую комнату). После поселения все посетители спят на протяжении 6 секунд, при этом

семафоры блокируют доступ к гостинице другим посетителям, в какой-то момент времени начинают работать потоки-комнаты, которые начинают выселять посетителей по порядку из каждой комнаты. Как только посетителей начинают выселять, значение семафора изменяется и разблокируются потоки-посетители, на место выселенных посетителей помещаются случайные посетители, которые прямо перед этим не были в гостинице. Этот процесс продолжается до завершения цикла (чтобы не делать бесконечный цикл, я сделала счетчик, который останавливает цикл, когда достигает заранее заданного значения 10000).