Задание 2

Реализовать Класс MyConcurrentQueue

Отчёт

Ши Хуэй

2024

1. Постановка задачи

Реализовать класс/структуру MyConcurrentQueue, которая поддерживает работу с многопоточными операциями. Основные требования к реализации:

Атрибут *MyConcurrentQueue::queue* — очередь, которая поддерживает произвольные POD-данные. Емкость очереди ограничена.

Метод *MyConcurrentQueue::put()* — добавляет элемент в очередь. Если очередь заполнена, поток ждет освобождения места.

Метод *MyConcurrentQueue::get()* — извлекает элемент из очереди. Если очередь пуста, поток ждет появления элемента.

Методы *put()* и *get()* могут вызываться одновременно различным числом потоков (произвольным).

Поток, вызвавший *put()* или *get()*, может завершить свою операцию позже других потоков, несмотря на то, что начал раньше.

Реализовать корректную работу очереди при любом состоянии:

Если очередь пуста, потоки, пытающиеся взять элемент, ожидают.

Если очередь заполнена, потоки, пытающиеся добавить элемент, ожидают.

2. Формат командной строки

g++ -std=c++11 -pthread -o concurrent_queue concurrent_queue.cpp

./concurrent_queue

3. Спецификация системы

Pe- Operation System：Ubuntu 20.04.5 LTS

- Architecture: x86_64

- Model name：Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz

- CPU(s):1

- Memory:8 GB

- g++ (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4

4. зультаты выполнения

4.1 Увеличение требований：

- Производитель добавляет в очередь 15 товаров за один вызов метода put().
- Каждый производитель выполняет случайное количество операций (от 1 до 5), добавляя товары.
- Если количество товаров в очереди падает ниже 2, производитель активируется и начинает производство.
- Каждый потребитель берет из очереди один товар за вызов get(). Если товаров нет, поток ждет.

4.2 Реализация：

4.2.1 Атрибуты:

- *capacity* — максимальная вместимость очереди.
- *queue* — очередь, реализованная с помощью std::queue<T>, хранящая товары.

- *done* — флаг, сигнализирующий об окончании работы всех производителей.
- Мьютекс (mutex) и условные переменные (*cond_full, cond_empty*) для синхронизации потоков.

4.2.2 Методы:

- *put()* — добавляет товары в очередь. Производитель блокируется, если очередь заполнена, и ждет, пока в очереди появится место для новых товаров.
- *get()* — извлекает товары из очереди. Потребитель блокируется, если очередь пуста, и ждет появления товаров.
- *set_done()* — уведомляет, что все производители завершили работу, устанавливая флаг done

4.2.3 Логика работы:

Производители и потребители работают параллельно, синхронизируясь через мьютексы и условные переменные. Производители добавляют товары в очередь, блокируясь, если очередь заполнена, пока не освободится место. Потребители извлекают товары из очереди, блокируясь, если очередь пуста, пока не появятся новые товары. Когда все производители завершают работу, они устанавливают флаг *done*, позволяя потребителям корректно завершить свою работу, когда очередь опустеет.

4.3 Тестирование

4.3.1 Test1

В тесте 1 у нас есть один производитель и три потребителя. Программа выполняется в общей сложности 4,50789 секунды.

4.3.2 Test2

В тесте 2 у нас есть три производителя и один потребитель.

Продолжительность программы составляет в общей сложности 22.928 секунды.

4.3.3 Test3

В тесте 3 у нас есть два производителя и пять потребителей. Программа выполняется в общей сложности 4.50983 секунды.

4.3.4 Test4

В Test4 у нас есть производитель и потребитель. Продолжительность программы составляет в общей сложности 9,01865 секунды.