Домашнее задание

Задача о писателях и читателях

Бишаев Алан Имранович

БПИ 202

**ОПИСАНИЕ ЗАДАНИЯ (вариант 3):**

Для полного описания задачи приведу условие, которое написано в файле с задачей:

Базу данных разделяют два типа процессов – читатели и писатели. Читатели выполняют транзакции, которые просматривают записи базы данных, транзакции писателей и просматривают и изменяют записи. Предполагается, что в начале БД находится в *непротиворечивом* состоянии (например, если каждый элемент — число, то они все отсортированы). Каждая отдельная транзакция переводит БД из одного *непротиворечивого* состояния в *другое*. Для предотвращения взаимного влияния транзакций процесс-писатель должен иметь исключительный доступ к БД. Если к БД не обращается ни один из процессов-писателей, то выполнять транзакции могут одновременно сколько угодно читателей. Создать многопоточное приложение с потоками- писателями и потоками-читателями. Реализовать решение, используя семафоры.

В сущности, требуется разработать многопоточное консольное приложение на языке программирования С++, организация потоков в котором соответствует модели “производитель – потребитель” *(стиль написания – произвольный)*. Исходя из условия, можно выделить 3 потока, которые должны быть реализованы:

|  |  |
| --- | --- |
| **Название потока** | **Описание** |
| Главный поток | В этом потоке выполняется функция main, из которой и стартует приложение. |
| Поток читателя | просмотр записей базы данных, транзакции писателей и просмотр и изменение записей |
| Поток писателя | Транзакции |

Также нужно: описать подробно используемую модель вычислений; зафиксировать количество заголовочных файлов, программных файлов, общий размер исходных текстов, полученный размер исполняемого кода (если он формируется), время выполнения программы для различных тестовых наборов данных.

**ОПИСАНИЕ ВЗАИМОДЕЙСТВИЯ С ПРОГРАММОЙ**

Ввод данные происходит через строку аргументов при запуске программы. Для ввода информации об имуществе из аргументов:

Прикреплю файлы с входными и выходными файлами

**ОПИСАНИЕ ИСПОЛЬЗОВАННОЙ МОДЕЛИ ВЫЧИСЛЕНИЙ:**

В презентации в лекции (и в учебном пособии “СРЕДСТВА РАЗРАБОТКИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ” от Сибирского Федерального Университета соответственно) дано следующее определение модели “ производитель-потребитель”:

**Производители и потребители**– это парадигма взаимодействующих неравноправных процессов. Одни из процессов «производят» данные, другие их «потребляют». Часто такие процессы организуются в *конвейер*, через который проходит информация. Каждый процесс конвейера потребляет выход своего предшественника и производит входные данные для своего последователя. Другой распространенный способ организации потоков – древовидная структура, на этом основан, в частности, принцип *дихотомии*.

На деле, в данной модели все потоки разделяются на 2 категории – производители и потребители. Производители добавляют данные в буфер, а потребители – как-либо их обрабатывают (либо удаляют из буфера). Буфером же является некоторое хранилище данных (например, массив). Схематично модель можно представить следующим образом:

\

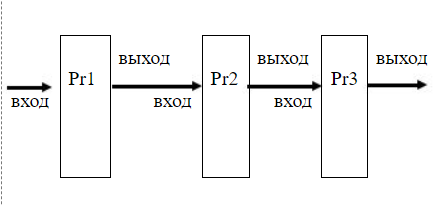
Существуют 2 различные разновидности данной модели – с ограниченным буфер и без. Модели с неограниченным буфером менее подвержены проблемам с многопоточностью, но при этом трудно реализуемы с практической точки зрения (нельзя говорить о бесконечной памяти, работая с физическим устройством, пространство памяти которого в принципе конечно). Модель же с ограниченным буфером сталкивается с рядом проблем

* Производитель пытается положить данные в полностью заполненный буфер. Если такое происходит, то это приводит к утечке данных либо тех, которые уже есть в буфере, либо тех, которые сгенерировал производитель. Следовательно, надо всегда проверять возможность добавить данные в буфер, прежде чем сделать это.
* Потребитель пытается достать данные из пустого буфера. Противоположная предыдущему пункту проблема. Здесь не происходит потеря данных, однако такая ситуация может привести к непредсказуемому поведению программы, когда мы обращаемся к неинициализированным ячейкам памяти, которые в целом могут быть зарезервированы под какой-то служебный процесс (следовательно, доступ к ним закрыт и программа упадет), либо содержать мусор. Для этих целей в потоке потребителя всегда нужно проверять на возможность достать элемент из буфера, прежде чем делать это.
* Наличие общей памяти. Буфер является общей памятью для производителей и потребителей, что приводит к тому, что одна область данных доступна сразу нескольким потокам, очередность выполнения которых предсказать в принципе невозможно. Это приводит к некорректной обработке неатомарных операций (таких, как инкремент, например), а так как неатомарных операций в программировании на самом деле много, то это становится большой проблемой в разработке приложения. Для совершения операций с общей памятью возникает необходимость синхронизации потоков. Обычно в этих целях применяют mutex объекты (блокирование доступа только за одним потоком) или семафор (похож на mutex с отличием, что доступ можно выдавать сразу нескольким потокам).

Если обратиться к вышеупомянутому пособию Сибирского Федерального Университета, то в нем же поясняется понятие организации потоков в конвейер:

“ *конвейер*– последовательность процессов, в которой каждый потребляет данные предшественника и поставляет данные для последующего процесса”

К этому определению приложена иллюстрация, которая дает более наглядное понимание вышесказанного:



Для упрощения здесь опущено изображение промежуточных буферов, которые существуют для связей между потоками. Стоит также отметить, что связи в конвейере однонаправленные, то есть производитель может выдавать данные потребителю, но потребитель как-либо влиять на работу производитель – нет.

Из вышеприведенной схемы и данного определения можно сделать вывод, что модели “производитель-потребитель” соответствует наличие в программе несколько буферов, для которых есть как производители, так и потребители, притом производитель для одного буфера может быть потребителем другого. Именно эти рассуждения я и использовал для реализации программы в рамках текущего задания.

В качестве примера использования данной модели приведу пример текущего задания. По условию задания каждый из прапорщиков выполняет строго свою функцию, разбивая общую задачу (своровать имущество со склада) на компоненты. Логично в данном случае применить конвейерную реализацию, в которой каждый поток-человек готовит данные для последующего. В то же время появилось разделение и для 3-х буферов, которые выступают как источниками данных, так и связками между потоками.

**ОСНОВНЫЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ:**

* Число заголовочных файлов – 2 (buffer.h, print.h)
* Число модулей реализации – 2
* Общий размер исходных текстов – 248 строк кода вместе с комментариями (6.91 кб).
* Размер исполняемого файла – 24.4 кб.