НИТУ «МИСиС»
Кафедра Инженерной кибернетики

Параллельные вычисления

Лабораторная работа №5. «Потокобезопасные объекты. Механизмы Mutex Exclusion, разработка потокобезопасных объектов»

Сенченко Р.В.

8 мая 2020 г.

I. Базовая часть. Необходимо разработать потокобезопасные объекты (программные компоненты, классы), реализующие абстрактный тип данных *вектора*, *очереди* и *стека*, с механизмом "взаимного исключения" — т.н. *мьютексами (mutex exclusion)*. Разработанные объекты должны обладать хорошей объектно-ориентированной архитектурой, а также обеспечивать базовую функциональность соответствующих абстрактных типов данных. Подробную информацию по реализуемым классам стека, очереди и вектора можно найти по ссылкам: *cppref::stack*, *cppref::queue* и *cppref::vector*, соответственно.

Означенные потокобезопасные объекты должны быть реализованы с использованием класса std::mutex для моделирования мьютексов, а также шаблонных классов std::lock_guard или std::unique_lock — по выбору и на усмотрение студента, — для реализации механизмов *блокировки (lock)* и *разблокировки (unlock)* мьютексов.

Разработанные программные компоненты должны быть протестированы; для этого необходимо создать множество потоков std::thread, каждому из которых предстоит обращаться на чтение и запись к экземплярам потокобезопасных объектов. По результату тестирования должна быть продемонстрирована корректная работа компонентов.

Дополнительно необходимо:

1. Фиксировать время запроса тем или иным потоком разделяемого ресурса, время его фактического захвата и на этой основе оценить минимальное, максимальное и среднее время ожидания разделяемого ресурса.

2. Оценить коэффициент эффективного использования разделяемого ресурса, как отношение суммарного времени работы ресурса (в lock-состоянии) к общему времени исполнения многопоточного участка.

Реализованный алгоритм должен быть хорошо документирован и снабжён исчерпывающими комментариями. Программный язык разработки: c++.

II. Дополнительная часть.

1. **Объект логирования, 4⋆.** Построить объект logger, поддерживающий потокобезопасную работу для множества параллельных потоков исполнения. При этом необходимо предусмотреть следующие особенности объекта и/или наличие методов:

   (a) Объект logger ассоциируется с некоторым текстовым файлом .txt, используемый в процессе работы с объектом для сохранения логов. Параметры работы с файлом, его имя, расположение, а также режим дополнения/перезаписи должны задаваться параметрами конструктора.

   (b) Предусмотреть метод void log(std::chrono::time_point T, std::string logEntry), позволяющий зарегистрировать запись с текстом logEntry в объекте логирования, параметр T отражает дату и время формирования записи в лог.
   См. дополнительно ссылку *cppref::chrono* для работы с объектами времени.

   (c) Объект накапливает события логирования (log-entry) во внутренней очереди. События из очереди периодически сохраняются в ассоциированный файл логирования; после сохранения внутренння очередь очищается. Указанную процедуру "отгрузки" событий логирования в файл необходимо запускать всякий раз, когда суммарная длина всех хранимых логов во внутренней очереди достигнет и/или превзойдёт длину в 2000 символов.

   (d) Необходимо обеспечить, чтобы события логирования сохранялись в ассоциированном файле в сортированном порядке по времени логирования.

   Работу объекта необходимо проверить на какой-либо вычислительной задаче с логированием. Например, на задаче переборного поиска всех треугольников с целочисленной длиной сторон a, b и c, не превосходящей 10 000; логировать следует событие нахождения очередного треугольника (при этом, очевидно, необходимо логировать также и сами значения (a, b, c)). Допускается выбрать любую другую, собственную задачу для тестирования объекта логирования logger.

2. **Асинхронный репликатор, 5⋆.** Пусть задан потокобезопасный вектор V, разработанный в рамках Базовой части лабораторной работы. Вектор V будет представлять собой некое *модельное хранилище данных*. Предполагается что над вектором V производят преобразования вставки (insert) и удаления (erase) элементов в параллельном и независимом режиме множеством т.н. "пользователей".

   Необходимо построить объект Replicator, который имеет readonly-доступ к вектору V и **не может** непосредственно отслеживать вызов операции вставки/удаления элементов. Объект должен отслеживать *инкремент* в данных вектора V и транслировать все изменения на реплицированный вектор-копию V_r. Ясно, что изменения, произведённые с вектором V, будут отображаться на векторе V_r с некоторым запаздыванием (*лагом по времени*). Объект Replicator должен содержать в своей внутренней структуре подходящее количество потоков исполнения (с соответствующими

"ролями" и назначениями), которые позволили бы эффективно отслеживать состояние вектора-оригинала V и транслировать все его изменения на вектор-копию V_r.

3. **АВЛ-дерево, 4⋆.** Разработать класс, моделирующий потокобезопасное относительно операции вставки элементов *АВЛ-дерево (сбалансированное по высоте двоичное дерево поиска)*. Разработанный класс должен содержать базовую функциональность АВЛ-дерева и поддерживать операции:

   (a) *вставки элемента по заданному ключу*;

   (b) *поиска элемента по заданному ключу*;

   (c) *вставки элемента в определённое место*;

   (d) *удаления определённого элемента.*

   Необходимо продемонстировать корректность работы разработанного объекта на соответствующие тестовых данных.

   Подробнее со структурой АВЛ-дерева, псевдокодом операций и пр. информацией можно ознакомиться в книге *Кнут Д.Э. Искусство программирования. Том 3. Сортировка и поиск, (раздел 6.2.3 на эл.стр. 490)*.