Микропроект №2 по дисциплине «Архитектура вычислительных систем»

Пояснительная записка

Вариант 4 (26 % 22)

Работу выполнила студент БПИ198
Скрыпина Дарья Кирилловна

1) Текст задания:

«4. *Задача об обедающих философах*. Пять философов сидят возле круглого стола. Они проводят жизнь, чередуя приемы пищи и размышления. В центре стола находится большое блюдо спагетти. Спагетти длинные и запутанные, философам тяжело управляться с ними, поэтому каждый из них, чтобы съесть порцию, должен пользоваться двумя вилками. К несчастью, философам дали только пять вилок. Между каждой парой философов лежит одна вилка, поэтому эти высококультурные и предельно вежливые люди договорились, что каждый будет пользоваться только теми вилками, которые лежат рядом с ним (слева и справа). Написать многопоточную программу, моделирующую поведение философов с помощью семафоров. Программа должна избегать фатальной ситуации, в которой все философы голодны, но ни один из них не может взять обе вилки (например, каждый из философов держит по одной вилке и не хочет отдавать ее). Решение должно быть симметричным, то есть все потоки-философы должны выполнять один и тот же код.»

2) Сценарий взаимодействия объектов программы и алгоритм работы программы

Данная программа написана на языке C++ с использованием стандартной библиотеки потоков языка C++. При разработке программы использовался стандарт C++20 в связи с возникшими трудностями с использованием семафоров из библиотеки pthread.

При запуске программа анализирует параметр командной строки, представляющий собой число, обозначающее количество секунд, выделенное на обед философов. Такое временное ограничение необходимо для того, чтобы программа работала не бесконечно. Если число не подходит под заданный формат (см. пункт 3), то программа выводит сообщение об ошибке и завершает свою работу.

Далее создаётся вектор из 5 потоков, представляющих собой обедающих философов. Основной поток ждёт заданное аргументом командной строки количество секунд (по истечении которых глобальная переменная типа bool **dinner_over** принимает значение **true**), а потоки философов запускаются и начинают выполнять функцию **void philosopher(int id).**

В данной функции представлен классический обеденный цикл философа. Пока обед не закончился, каждый философ входит в следующий цикл:

- Сначала каждый философ размышляет некоторое количество секунд (от 1 до 5 секунд). Время на размышление задаётся с помощью генератора случайных чисел. В консоль выводится соответствующее сообщение. Данная схема реализована в методе **think** данной программы.
- Обозначим вилки, лежащие на столе, номерами от 0 до 4. Каждый философ пробует взять в руки вилки, лежащие рядом с ним. Рядом с каждым философом лежит одна вилка с его же номером и одна вилка с ещё одним номером, равная увеличенному на 1 номеру первой вилки, взятым по остатку 5. Так, у 0-ого философа будут вилки с номерами 0 и 1, а у 4-ого философа – с номерами 4 и 0. **Сначала** каждый философ пробует взять свою вилку с **наименьшим** номером. Для каждого философа, кроме 4-ого, эта вилка будет лежать слева от него. То есть 0-ой философ попробует взять 0-ую вилку, 1-ый философ – 1-ую

вилку, …, а 4-ый философ пробует взять не 4-ую, а 0-ую вилку. Таким образом задаётся чёткий порядок взятия вилок, который предотвращает ситуацию **deadlock** – в противном случае мог возникнуть случай, когда все философы взяли вилку слева от себя (т.е. вилку с таким же номером, как и у себя), но никто не смог бы взять правую вилку, поскольку все вилки уже заняты. **Затем** каждый философ пробует взять свою вилку с **наибольшим** номером, т.е. 0-ой философ – 1-ую вилку, 1-ый философ – 2-ую вилку, …, 4-ый философ – 4-ую вилку. Если у философа не получается взять какую-либо из вилок, он ждёт, пока она освободится.

Данный алгоритм реализован с помощью двоичных семафоров, имеющих тип **std::binary_semaphore,** введённый в стандарте C++20. У каждой вилки есть свой бинарный семафор (хранящийся в массиве бинарных семафоров по соответствующему индексу), сигнализирующий о том, занята ли данная вилка сейчас или нет. Когда философ (поток) пробует взять вилку, он использует библиотечный метод **arr[<номер_вилки>].acquire(),** который вычитает 1 из значения семафора. Изначально значение каждого семафора = 1 (поскольку он бинарный). Если **acquire** применяется на двоичном семафоре, значение которого = 0 (т.е. философ пытается взять вилку, которая уже занята), то данный поток (философ) ждёт, пока значение семафора снова не станет равным 1. Данная схема реализована в методе **pick_up_forks** данной программы. Информация о каждой взятой вилке выводится на экран консоли после её успешного взятия.

- После того, как философ успешно взял обе вилки, он начинает есть спагетти. Делает он это в течение некоторого времени (от 1 до 5 секунд), заданного с помощью генератора случайных чисел. Информация о том, что философ начал есть и будет продолжать это делать в течение N секунд, выводится на экран консоли. Данная схема реализована в методе **eat** данной программы.
- После того, как философ поел, он кладёт обе вилки на стол – сначала ту, у которой номер меньше, затем ту, у которой номер больше. Об освобождении каждой из своих вилок каждый поток сообщает в консоль. Освобождение вилок выполняется с помощью библиотечного метода **arr[<номер вилки>].release(),** который прибавляет 1 к значению соответствующего двоичного семафора, тем самым его «освобождая» и давая другому потоку возможность взять эту вилку. Данная схема реализована в методе **put_down_forks** данной программы.

Такой обеденный цикл философа повторяется, пока время, отведённое на обед философов, не истечёт. После истечения времени на обед думающие в текущий момент философы заканчивают свои размышления и прекращают обеденную трапезу, а обедающие в текущий момент философы заканчивают есть, кладут свои вилки на стол и больше их не берут. На этом выполнение программы заканчивается – потоки синхронизируются, а память, выделенная на динамический массив двоичных семафоров (по-другому хранить семафоры в контейнере не вышло), очищается.

3) Входные данные программы

На вход программе подаётся единственное целое число в интервале [5;30] – количество секунд, выделенное философам на трапезу. Если данное число имеет некорректный для типа Integer формат или выходит за заданные границы, то программа выводит в консоль сообщение об ошибке и завершает свою работу.

Для компиляции программы используется следующая конструкция:

cl /EHsc /std:c++latest MP02_Var26.cpp

Для запуска программы используется следующая конструкция:

MP02_Var26 <количество секунд на обед философов>

4) Тестовые примеры

Примеры работы программы при некорректном значении входных параметров: (*Используемые значения: 2, 60, abcde*)

```
MP02_Var26 2

Number of seconds is out of expected range [5; 30].
Correct format: integer number in range [5; 30].
Please try again.
```

```
MP02_Var26 60

Number of seconds is out of expected range [5; 30].
Correct format: integer number in range [5; 30].
Please try again.
```

```
MP02_Var26 abcde

Couldn't parse the number of seconds as the first command-line argument.
Correct format: integer number in range [5; 30].
Please try again.
```

Пример работы программы при использовании корректного входного параметра: (*Здесь представлен всего 1 пример, поскольку выходные параметры занимают слишком много места. Больше тестовых примеров можно найти в файлах test*.txt на GitHub.)*

```
>MP02_Var26 5
Philosopher 0 is thinking for 5 seconds...
Philosopher 1 is thinking for 1 seconds...
Philosopher 2 is thinking for 4 seconds...
Philosopher 3 is thinking for 3 seconds...
Philosopher 4 is thinking for 1 seconds...
Philosopher 4 picked up fork 0.
Philosopher 4 picked up fork 4.
Philosopher 4 is eating for 2 seconds.
Philosopher 1 picked up fork 1.
Philosopher 1 picked up fork 2.
Philosopher 1 is eating for 4 seconds.
Philosopher 3 picked up fork 3.
Philosopher 4 has finished eating.
Philosopher 4 put down fork 0.
Philosopher 4 put down fork 4.
Philosopher 4 is thinking for 1 seconds...
Philosopher 3 picked up fork 4.
Philosopher 3 is eating for 1 seconds.
Philosopher 4 picked up fork 0.
Philosopher 3 has finished eating.
Philosopher 3 put down fork 3.
Philosopher 3 put down fork 4.
Philosopher 3 is thinking for 2 seconds...
Philosopher 4 picked up fork 4.
Philosopher 4 is eating for 1 seconds.
Dinner's over!
Philosopher 1 has finished eating.
Philosopher 1 put down fork 1.
Philosopher 1 put down fork 2.
Philosopher 2 picked up fork 2.
Philosopher 2 picked up fork 3.
Philosopher 2 is eating for 3 seconds.
Philosopher 4 has finished eating.
Philosopher 4 put down fork 0.
Philosopher 4 put down fork 4.
Philosopher 2 has finished eating.
Philosopher 2 put down fork 2.
Philosopher 2 put down fork 3.
```

5) Используемые источники:

- https://en.wikipedia.org/wiki/Dining_philosophers_problem - описание проблемы обедающих философов и возможные варианты её решения.
- https://en.cppreference.com/w/cpp/thread/counting_semaphore - документация языка C++, описывающая особенности конструкций counting_semaphore и binary_semaphore.
- https://en.cppreference.com/w/cpp/thread/counting_semaphore/acquire, https://en.cppreference.com/w/cpp/thread/counting_semaphore/release - описание методов для работы с семафорами в C++20.