Факультет программной инженерии и компьютерной техники

Теоретические основы компьютерной графики и вычислительной оптики

Лабораторная работа №1

Преподаватель: Доронин Олег Владимирович

Выполнил: студент: Кульбако Артемий Юрьевич, P4115

Санкт-Петербург
2022

ЗАДАНИЕ

Вам нужно зарегистрироваться на сайте leetcode.com. Это система для решения за- дач, которые спрашивают на собеседованиях в крупные IT компании (Google, Amazon, Facebook, ...). Для выполнения лабораторной работы необходимо решить задачу 1226. The Dining Philosophers. В этой задаче всегда 5 философов. Когда философ хочет кушать, то вызывает метод:

```
void wantsToEat(int philosopher,
     function<void()> pickLeftFork,
     function<void()> pickRightFork,
     function<void()> eat,
     function<void()> putLeftFork,
     function<void()> putRightFork);
```

`philosopher` - число от 0 до 4, «имя» философа.

`pickLeftFork, pickRightFork, eat, putLeftFork, putRightFork` - действия которые он может сделать.

Эту задачу можно решать методом атомарного захвата вилок или любым другим методом, который вам больше нравится. Если одновременно могут есть несколько философов, то они должны это делать.

Рис. 1 - визуализация задачи

КОД

```
/*
https://leetcode.com/problems/the-dining-philosophers/

Пусть вилки будут пронумерованы от 0 до 4. Сначала философ берёт вилку с
наименьшим номеров, из лежащих рядом, потом с наибольшим из лежащих рядом.
Получается, что в ситуации, когда 4 философа взяли по вилке, для последнего
философа, вилка с наименьшим номером будет не доступна,
он не сможет взять ни одну вилку и не устроит всем дедлок.
Небольшая модификация позволит ускорить алгоритм в ситуации, когда все
потоки запустились одновременно и работают известное количество времени. В
таком случае,
можно для каждого нечётного потока поменять приоритет захвата (захватывать
сначала ресурс с большим временем, потом с меньшим). Таким образов, как
только первый
поток завершит свою работу и освободит ресурс, запустится смогут оба его
соседа.
*/
#include <mutex>
#include <algorithm> // std::swap

class DiningPhilosophers {
    private:
        std::mutex mtxs[5];
    public:
        DiningPhilosophers() {}

        void wantsToEat(
            int id,
            function<void()> pickLFork,
            function<void()> pickRFork,
            function<void()> eat,
            function<void()> putLFork,
            function<void()> putRFork
        ) {
            int first, second = -1;
            auto get_second = [id]() { return (id + 1) % 5; };
            if (id % 2 == 0) first = id, second = get_second();
            else second = id, first = get_second();
            mtxs[first].lock();
            mtxs[second].lock();
            pickLFork();
            pickRFork();
            eat();
            putRFork();
            putLFork();
            mtxs[second].unlock();
            mtxs[first].unlock();
        }
};
```

ВЫВОД

В процессе выполнения лабораторной работы я вспомнил назначения примитивов синхронизации и воспользовался одним из них (мьютексом) для решения задачи об обедающих философах - классического примера, используемого в информатике для иллюстрации проблем синхронизации при разработке параллельных алгоритмов и техник решения этих проблем. Я реализовал решение на основе иерархии ресурсов, предложенное самим автором задачи - Э. Дейкстрой. Это решение не является самым эффективным ввиду того, что потоку для захвата ресурса с меньшим приоритетом нужно отпустить все захваченные ресурсы с большим приоритетом, но всё же позволяет избежать ситуации дедлока, чего и следовало добиться в данной лабораторной работе.

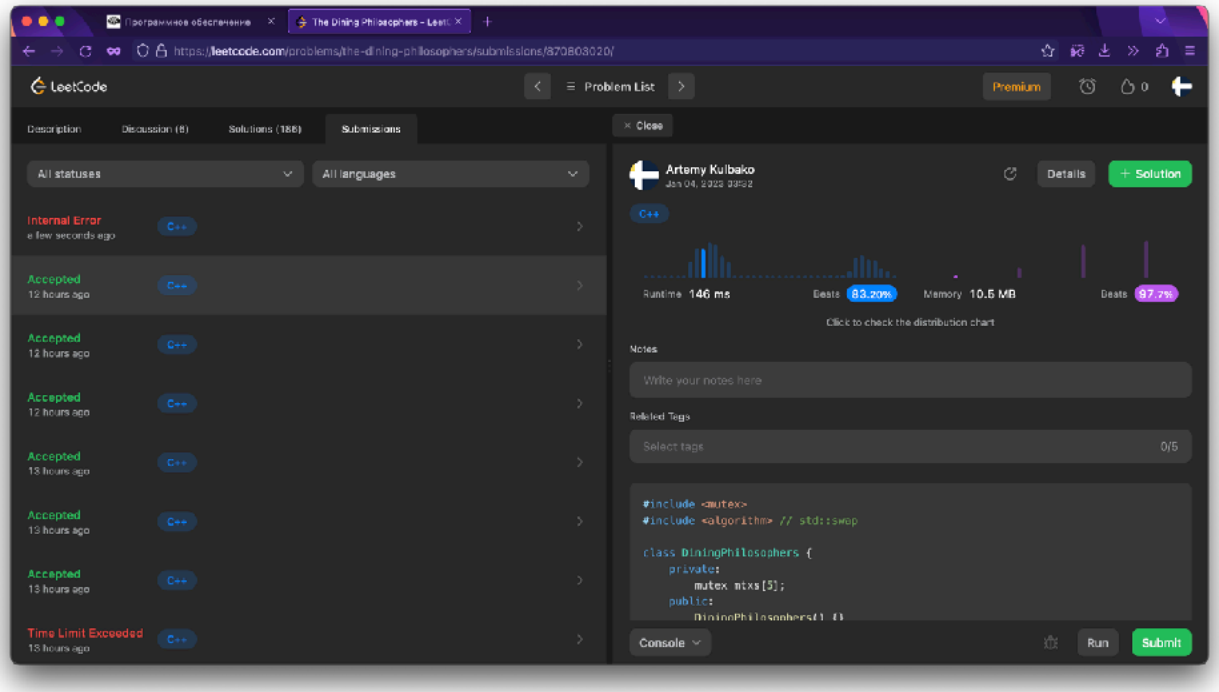

Рис. 2 - решение на LeetCode

Объяснение алгоритма: https://drive.google.com/file/d/1glbb2jOR0VYyMWKLuJ9VgkubnHNVZm4o/view?usp=share_link

Код: https://gitlab.se.ifmo.ru/system-software/itmo_winter_2023/01-philosophers/testpassword