Параллельные алгоритмы

Максим Ивахненко

March 15, 2016

0.0.1 Блокирующий флаг

Задача. *Рассмотрим реализацию булевого флажка, который по задумке позволял бы потокам с блокировкой дождаться (с помощью метода wait) его простановки (методом set).*

Предполагается, что ждать флага могут несколько потоков, а ставить флаг - только один.

Мьютекс используется для взаимного исключения ждущих потоков, для флага используется атомик, его ставит только один поток.

```
#include <mutex>
#include <condition_variable>

class blocking_flag {
public:
    blocking_flag()
        : ready_(false)
    {}

    void wait() {
        std::unique_lock<std::mutex> lock(mtx_);
        while (!ready_.load()) {
            ready_cond_.wait(lock);
        }
    }

    void set() {
        ready_.set(true);
        ready_cond_.notify_all();
    }

private:
    std::atomic<bool> ready_;
    std::mutex mtx_;
    std::condition_variable ready_cond_;
};
```

Пример использования:

```
#include <thread>
#include <iostream>

int main() {
    blocking_flag f;

    std::thread t(
        [&f]() {
            f.wait();
            std::cout << "ready!" << std::endl;
        }
    );

    f.set();
    t.join();

    return 0;
}
```

Оцените коррекность этого кода.

Решение.
Код работает не корректно.

Пример
Есть два потока A и B.
A ставит флаг, B - ждет.
Пусть в тот момент, когда B выполняет метод $wait()$, после проверки условия в $while$ и перед вызовом $wait()$ поток был остановлен планировщиком.
A выполняет $set()$ и сигналит условной перемнной, но сигнал просто пропадает. Теперь возвращаемся к B, который вызывает $wait()$ и зависает.

Решение проблемы
Использовать семафор вместо условной переменной или захватывать мьютекс в методе $set()$.

P.S
У $atomic$ нет метода $set()$, так что решение рассматривалось в предположении, что вместо $ready.set(true)$ написано $ready.store(true)$.