Примитивы синхронизации в C++11: задача поставщик-потребитель

План

Семафоры

Блокировки и условные перменные

Атомарные операции

План

Семафоры

Блокировки и условные перменные

Атомарные операции

Семафоры

```
std::mutex mtx;
std::vector<float> arr;

void produce(float x) {
    mtx.lock();
    arr.push_back(x);
    mtx.unlock();
}
```

Семафоры

Типы мьютексов

```
std::mutex
std::recursive_mutex
std::timed_mutex
std::recursive_timed_mutex
```

Методы

```
lock()
try_lock()
try_lock_for()
try_lock_until()
```

План

Семафоры

Блокировки и условные перменные

Атомарные операции

Блокировки

```
std::mutex mtx;
std::vector<float> arr;

void produce(float x) {
    std::lock_guard<std::mutex> lock(mtx);
    arr.push_back(x);
}
```

Типы блокировок

`std::lock_guard`
`std::unique_lock`

Условные переменные

```
std::mutex mtx;
std::condition_variable cv;
std::vector<float> arr;

void produce(float x) {
    std::lock_guard<std::mutex> lock(mtx);
    arr.push_back(x);
    cv.notify_one();
}
```

Типы условных переменных

```
std::condition_variable
std::condition_variable_any
```

Условные переменные

```
std::condition_variable cv;

void consume() {
  while (true) {
    std::unique_lock<std::mutex> lock(mtx);
    cv.wait(lock, [&arr](){ return !arr.empty(); });
    float x = arr.back();
    arr.pop_back();
    lock.unlock();
    process(x);
  }
}
```

Условные переменные

```
std::condition_variable_any cv;

void consume() {
  while (true) {
    float x;
    {
      std::lock_guard<std::mutex> lock(mtx);
      cv.wait(lock, [&arr](){ return !arr.empty(); })
      x = arr.back();
      arr.pop_back();
    }
    process(x);
  }
}
```

План

Семафоры

Блокировки и условные перменные

Атомарные операции

Атомарные операции

Поддерживаются

`std::atomic<T>`

`char`	`long`
`signed char`	`unsigned long`
`unsigned char`	`long long`
`short`	`unsigned long long`
`unsigned short`	`char16_t`
`int`	`char32_t`
`unsigned int`	`wchar_t`
`plain-old-data`	`T*`

Не поддерживаются: тип bool, структуры с указателями.

Атомарные операции

Если атомарная операция над каким-либо типом не поддерживается процессором, то она заменяется на операцию с блокировкой.

Отсутствие блокировки гарантируется только для `std::atomic_flag`.

Циклический семафор

```
struct Spin_mutex {

    void lock() {
        while (f.test_and_set());
    }

    void unlock() { f.clear(); }

private:
    std::atomic_flag f = ATOMIC_FLAG_INIT;
};
```

Циклический семафор

```
Spin_mutex mtx;
std::vector<float> arr;

void produce(float x) {
    std::lock_guard<Spin_mutex> lock(mtx);
    arr.push_back(x);
}
```