

**Уильямс, Энтони.**

**У36** Параллельное программирование на C++ в действии. Практика разработки многопоточных программ / Э. Уильямс ; пер. с англ. А. А. Слинкина. — 2-е изд., эл. — 1 файл pdf : 674 с. — Москва : ДМК Пресс, 2023. — Систем. требования: Adobe Reader XI либо Adobe Digital Editions 4.5 ; экран 10". — Текст : электронный.

ISBN 978-5-89818-319-6

В наши дни компьютеры с несколькими многоядерными процессорами стали нормой. Стандарт C++11 языка C++ предоставляет развитую поддержку многопоточности в приложениях. Поэтому, чтобы сохранять конкурентоспособность, вы должны овладеть принципами и приемами их разработки, а также новыми средствами языка, относящимися к параллелизму.

Книга «Параллельное программирование на C++ в действии» не предполагает предварительных знаний в этой области. Вдумчиво читая ее, вы научитесь писать надежные и элегантные многопоточные программы на C++11. Вы узнаете о том, что такое потоковая модель памяти, и о том, какие средства поддержки многопоточности, в том числе запуска и синхронизации потоков, имеются в стандартной библиотеке. Попутно вы познакомитесь с различными нетривиальными проблемами программирования в условиях параллелизма.

УДК 004.438C++11  
ББК 32.973.26-018.2

**Электронное издание на основе печатного издания:** Параллельное программирование на C++ в действии. Практика разработки многопоточных программ / Э. Уильямс ; пер. с англ. А. А. Слинкина. — Москва : ДМК Пресс, 2014. — 672 с. — ISBN 978-5-94074-537-2. — Текст : непосредственный.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

В соответствии со ст. 1299 и 1301 ГК РФ при устранении ограничений, установленных техническими средствами защиты авторских прав, правообладатель вправе требовать от нарушителя возмещения убытков или выплаты компенсации.

ISBN 978-5-89818-319-6

© by Manning Publications Co.  
© Оформление, перевод на русский язык  
ДМК Пресс, 2014



# ОГЛАВЛЕНИЕ

<b>Предисловие .....</b>	<b>13</b>
<b>Благодарности .....</b>	<b>15</b>
<b>Об этой книге .....</b>	<b>17</b>
<b>Об иллюстрации на обложке .....</b>	<b>21</b>
<b>ГЛАВА 1. Здравствуй, параллельный мир! .....</b>	<b>22</b>
1.1. Что такое параллелизм? .....	23
1.1.1. Параллелизм в вычислительных системах .....	23
1.1.2. Подходы к организации параллелизма .....	26
1.2. Зачем нужен параллелизм? .....	29
1.2.1. Применение параллелизма для разделения обязанностей .....	29
1.2.2. Применение параллелизма для повышения производительности .....	30
1.2.3. Когда параллелизм вреден? .....	32
1.3. Параллелизм и многопоточность в C++ .....	33
1.3.1. История многопоточности в C++ .....	34
1.3.2. Поддержка параллелизма в новом стандарте .....	35
1.3.3. Эффективность библиотеки многопоточности для C++ .....	36
1.3.4. Платформенно-зависимые средства .....	37
1.4. В начале пути .....	38
1.4.1. Здравствуй, параллельный мир .....	38
1.5. Резюме .....	40
<b>ГЛАВА 2. Управление потоками .....</b>	<b>41</b>
2.1. Базовые операции управления потоками .....	41
2.1.1. Запуск потока .....	42
2.1.2. Ожидание завершения потока .....	45
2.1.3. Ожидание в случае исключения .....	46
2.1.4. Запуск потоков в фоновом режиме .....	48
2.2. Передача аргументов функции потока .....	51
2.3. Передача владения потоком .....	54
2.4. Задание количества потоков во время выполнения .....	58
2.5. Идентификация потоков .....	61
2.6. Резюме .....	64

## ГЛАВА 3. Разделение данных между потоками 65

3.1. Проблемы разделения данных между потоками.....	66
3.1.1. Гонки.....	68
3.1.2. Устранение проблематичных состояний гонки.....	69
3.2. Защита разделяемых данных с помощью мьютексов.....	70
3.2.1. Использование мьютексов в C++.....	71
3.2.2. Структурирование кода для защиты разделяемых данных.....	73
3.2.3. Выявление состояний гонки, внутренне присущих интерфейсам.....	74
3.2.4. Взаимоблокировка: проблема и решение.....	83
3.2.5. Дополнительные рекомендации, как избежать взаимоблокировок.....	86
3.2.6. Гибкая блокировка с помощью <code>std::unique_lock</code> .....	94
3.2.7. Передача владения мьютексом между контекстами.....	95
3.2.8. Выбор правильной гранулярности блокировки.....	97
3.3. Другие средства защиты разделяемых данных.....	100
3.3.1. Защита разделяемых данных во время инициализации.....	100
3.3.2. Защита редко обновляемых структур данных.....	105
3.3.3. Рекурсивная блокировка.....	107
3.4. Резюме.....	108

## ГЛАВА 4. Синхронизация параллельных операций ..... 110

4.1. Ожидание события или иного условия.....	111
4.1.1. Ожидание условия с помощью условных переменных.....	112
4.1.2. Потокбезопасная очередь на базе условных переменных.....	115
4.2. Ожидание одноразовых событий с помощью механизма будущих результатов.....	121
4.2.1. Возврат значения из фоновой задачи.....	122
4.2.2. Ассоциирование задачи с будущим результатом.....	125
4.2.3. Передача задач между потоками.....	127
4.2.4. Использование <code>std::promise</code> .....	129
4.2.5. Сохранение исключения в будущем результате.....	131
4.2.6. Ожидание в нескольких потоках.....	133
4.3. Ожидание с ограничением по времени.....	136
4.3.1. Часы.....	137
4.3.2. Временные интервалы.....	138
4.3.3. Моменты времени.....	140
4.3.4. Функции, принимающие таймаут.....	142
4.4. Применение синхронизации операций для упрощения кода.....	144

4.4.1. Функциональное программирование с применением будущих результатов .....	145
4.5. Резюме .....	156

## **ГЛАВА 5. Модель памяти C++ и атомарные операции ..... 158**

5.1. Основы модели памяти .....	159
5.1.1. Объекты и ячейки памяти .....	159
5.1.2. Объекты, ячейки памяти и параллелизм .....	161
5.1.3. Порядок модификации .....	162
5.2. Атомарные операции и типы в C++ .....	163
5.2.1. Стандартные атомарные типы .....	163
5.2.2. Операции над <code>std::atomic_flag</code> .....	167
5.2.3. Операции над <code>std::atomic&lt;bool&gt;</code> .....	170
5.2.4. Операции над <code>std::atomic&lt;T*&gt;</code> : арифметика указателей ....	173
5.2.5. Операции над стандартными атомарными целочисленными типами .....	175
5.2.6. Основной шаблон класса <code>std::atomic&lt;&gt;</code> .....	175
5.2.7. Свободные функции для атомарных операций .....	177
5.3. Синхронизация операций и принудительное упорядочение .....	180
5.3.1. Отношение синхронизируется-с .....	182
5.3.2. Отношение происходит-раньше .....	183
5.3.3. Упорядочение доступа к памяти для атомарных операций...	185
5.3.4. Последовательности освобождений и отношение синхронизируется-с .....	208
5.3.5. Барьеры .....	212
5.3.6. Упорядочение неатомарных операций с помощью атомарных .....	214
5.4. Резюме .....	216

## **ГЛАВА 6. Проектирование параллельных структур данных с блокировками ..... 218**

6.1. Что понимается под проектированием структур данных, рассчитанных на параллельный доступ? .....	219
6.1.1. Рекомендации по проектированию структур данных для параллельного доступа .....	220
6.2. Параллельные структуры данных с блокировками .....	222
6.2.1. Потокбезопасный стек с блокировками .....	222
6.2.2. Потокбезопасная очередь с блокировками и условными переменными .....	226
6.2.3. Потокбезопасная очередь с мелкогранулярными блокировками и условными переменными .....	231

6.3. Проектирование более сложных структур данных с блокировками.....	245
6.3.1. Разработка потокобезопасной справочной таблицы с блокировками .....	246
6.3.2. Потокобезопасный список с блокировками .....	253
6.4. Резюме .....	258

## ГЛАВА 7. Проектирование параллельных структур данных без блокировок ..... 260

7.1. Определения и следствия из них.....	261
7.1.1. Типы неблокирующих структур данных.....	262
7.1.2. Структуры данных, свободные от блокировок .....	262
7.1.3. Структуры данных, свободные от ожидания .....	263
7.1.4. Плюсы и минусы структур данных, свободных от блокировок .....	264
7.2. Примеры структур данных, свободных от блокировок .....	266
7.2.1. Потокобезопасный стек без блокировок .....	266
7.2.2. Устранение утечек: управление памятью в структурах данных без блокировок .....	271
7.2.3. Обнаружение узлов, не подлежащих освобождению, с помощью указателей опасности .....	277
7.2.4. Нахождение используемых узлов с помощью подсчета ссылок .....	287
7.2.5. Применение модели памяти к свободному от блокировок стеку.....	293
7.2.6. Потокобезопасная очередь без блокировок.....	299
7.3. Рекомендации по написанию структур данных без блокировок.....	313
7.3.1. Используйте std::memory_order_seq_cst для создания прототипа .....	314
7.3.2. Используйте подходящую схему освобождения памяти..	314
7.3.3. Помните о проблеме ABA.....	315
7.3.4. Выявляйте циклы активного ожидания и помогайте другим потокам.....	316
7.4. Резюме .....	317

## ГЛАВА 8. Проектирование параллельных программ ..... 318

8.1. Методы распределения работы между потоками .....	319
8.1.1. Распределение данных между потоками до начала обработки .....	320
8.1.2. Рекурсивное распределение данных .....	322
8.1.3. Распределение работы по типам задач.....	327

8.2. Факторы, влияющие на производительность параллельного кода.....	330
8.2.1. Сколько процессоров?.....	331
8.2.2. Конкуренция за данные и перебрасывание кэша .....	333
8.2.3. Ложное разделение .....	335
8.2.4. Насколько близки ваши данные?.....	336
8.2.5. Превышение лимита и чрезмерное контекстное переключение .....	337
8.3. Проектирование структур данных для повышения производительности многопоточной программы .....	338
8.3.1. Распределение элементов массива для сложных операций .....	339
8.3.2. Порядок доступа к другим структурам данных .....	342
8.4. Дополнительные соображения при проектировании параллельных программ.....	344
8.4.1. Безопасность относительно исключений в параллельных алгоритмах .....	344
8.4.2. Масштабируемость и закон Амдала .....	353
8.4.3. Скрытие латентности с помощью нескольких потоков...	355
8.4.4. Повышение скорости реакции за счет распараллеливания .....	356
8.5. Проектирование параллельного кода на практике.....	359
8.5.1. Параллельная реализация <code>std::for_each</code> .....	359
8.5.2. Параллельная реализация <code>std::find</code> .....	362
8.5.3. Параллельная реализация <code>std::partial_sum</code> .....	369
8.6. Резюме .....	380

## ГЛАВА 9. Продвинутое управление потоками ... 382

9.1. Пулы потоков .....	383
9.1.1. Простейший пул потоков .....	383
9.1.2. Ожидание задачи, переданной пулу потоков.....	386
9.1.3. Задачи, ожидающие других задач.....	391
9.1.4. Предотвращение конкуренции за очередь работ .....	394
9.1.5. Занимание работ .....	396
9.2. Прерывание потоков.....	401
9.2.1. Запуск и прерывание другого потока .....	402
9.2.2. Обнаружение факта прерывания потока .....	404
9.2.3. Прерывание ожидания условной переменной.....	405
9.2.4. Прерывание ожидания <code>std::condition_variable_any</code> .....	409
9.2.5. Прерывание других блокирующих вызовов .....	411
9.2.6. Обработка прерываний.....	412
9.2.7. Прерывание фоновых потоков при выходе из приложения.....	413
9.3. Резюме .....	415

## ГЛАВА 10. Тестирование и отладка многопоточных приложений ..... 416

10.1. Типы ошибок, связанных с параллелизмом .....	417
10.1.1. Нежелательное блокирование .....	417
10.1.2. Состояния гонки .....	418
10.2. Методы поиска ошибок, связанных с параллелизмом ...	420
10.2.1. Анализ кода на предмет выявления потенциальных ошибок.....	420
10.2.2. Поиск связанных с параллелизмом ошибок путем тестирования .....	423
10.2.3. Проектирование с учетом тестопригодности .....	425
10.2.4. Приемы тестирования многопоточного кода .....	427
10.2.5. Структурирование многопоточного тестового кода.....	431
10.2.6. Тестирование производительности многопоточного кода .....	435
10.3. Резюме .....	436

## ПРИЛОЖЕНИЕ А. Краткий справочник по некоторым конструкциям языка C++ ..... 437

A.1. Ссылки на <i>r</i> -значения.....	437
A.1.1. Семантика перемещения.....	439
A.1.2. Ссылки на <i>r</i> -значения и шаблоны функций.....	442
A.2. Удаленные функции .....	442
A.3. Умалчиваемые функции .....	445
A.4. constexpr-функции .....	449
A.4.1. constexpr и определенные пользователем типы.....	450
A.4.2. constexpr-объекты .....	454
A.4.3. Требования к constexpr-функциям .....	454
A.4.4. constexpr и шаблоны .....	455
A.5. Лямбда-функции .....	456
A.5.1. Лямбда-функции, ссылающиеся на локальные переменные ...	458
A.6. Шаблоны с переменным числом параметров.....	461
A.6.1. Расширение пакета параметров .....	463
A.7. Автоматическое выведение типа переменной.....	466
A.8. Поточно-локальные переменные .....	467
A.9. Резюме .....	469

## ПРИЛОЖЕНИЕ В. Краткое сравнение библиотек для написания параллельных программ ..... 470

## ПРИЛОЖЕНИЕ С. Каркас передачи сообщений и полный пример программы банкомата ..... 472

## ПРИЛОЖЕНИЕ D. Справочник по библиотеке

<b>C++ Thread Library</b> .....	<b>492</b>
D.1. Заголовок <chrono> .....	492
D.1.1. Шаблон класса std::chrono::duration .....	493
D.1.2. Шаблон класса std::chrono::time_point .....	503
D.1.3. Класс std::chrono::system_clock .....	506
D.1.4. Класс std::chrono::steady_clock .....	508
D.1.5. Псевдоним типа std::chrono::high_resolution_clock .....	510
D.2. Заголовок <condition_variable> .....	511
D.2.1. Класс std::condition_variable .....	511
D.2.2. Класс std::condition_variable_any .....	521
D.3. Заголовок <atomic> .....	530
D.3.1. std::atomic_XXX, псевдонимы типов .....	531
D.3.2. ATOMIC_XXX_LOCK_FREE, макросы .....	532
D.3.3. ATOMIC_VAR_INIT, макрос .....	533
D.3.4. std::memory_order, перечисление .....	533
D.3.5. std::atomic_thread_fence, функция .....	534
D.3.6. std::atomic_signal_fence, функция .....	535
D.3.7. std::atomic_flag, класс .....	535
D.3.8. Шаблон класса std::atomic .....	539
D.3.9. Специализации шаблона std::atomic .....	552
D.3.10. Специализации std::atomic<integral-type> .....	552
D.4. Заголовок <future> .....	571
D.4.1. Шаблон класса std::future .....	572
D.4.2. Шаблон класса std::shared_future .....	578
D.4.3. Шаблон класса std::packaged_task .....	585
D.4.4. Шаблон класса std::promise .....	592
D.4.5. Шаблон функции std::async .....	598
D.5. Заголовок <mutex> .....	600
D.5.1. Класс std::mutex .....	601
D.5.2. Класс std::recursive_mutex .....	603
D.5.3. Класс std::timed_mutex .....	606
D.5.4. Класс std::recursive_timed_mutex .....	611
D.5.5. Шаблон класса std::lock_guard .....	615
D.5.6. Шаблон класса std::unique_lock .....	617
D.5.7. Шаблон функции std::lock .....	628
D.5.8. Шаблон функции std::try_lock .....	629
D.5.9. Класс std::once_flag .....	630
D.5.10. Шаблон функции std::call_once .....	630
D.6. Заголовок <ratio> .....	631
D.6.1. Шаблон класса std::ratio .....	632
D.6.2. Псевдоним шаблона std::ratio_add .....	633
D.6.3. Псевдоним шаблона std::ratio_subtract .....	634
D.6.4. Псевдоним шаблона std::ratio_multiply .....	635



## Оглавление



D.6.5. Псевдоним шаблона <code>std::ratio_divide</code> .....	635
D.6.6. Шаблон класса <code>std::ratio_equal</code> .....	636
D.6.7. Шаблон класса <code>std::ratio_not_equal</code> .....	636
D.6.8. Шаблон класса <code>std::ratio_less</code> .....	637
D.6.9. Шаблон класса <code>std::ratio_greater</code> .....	637
D.6.10. Шаблон класса <code>std::ratio_less_equal</code> .....	638
D.6.11. Шаблон класса <code>std::ratio_greater_equal</code> .....	638
D.7. Заголовок <code>&lt;thread&gt;</code> .....	638
D.7.1. Класс <code>std::thread</code> .....	639
D.7.2. Пространство имен <code>this_thread</code> .....	649
<b>РЕСУРСЫ .....</b>	<b>652</b>
Печатные ресурсы.....	652
Сетевые ресурсы .....	653
<b>ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ .....</b>	<b>654</b>