
Ким, Хью и Ирен.



ОГЛАВЛЕНИЕ

Предисловие	13
Благодарности	15
Об этой книге	17
Об иллюстрации на обложке	21
ГЛАВА 1. Здравствуй, параллельный мир!	22
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
ГЛАВА 2. Управление потоками	41
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
Передача задач между потоками.....	127
4.2.3. Использование <code>std::promise</code>	129
4.2.4. Сохранение исключения в будущем результате.....	131
4.2.5. 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<bool></code>	170
5.2.4. Операции над <code>std::atomic<T*></code> : арифметика указателей	173
5.2.5. Операции над стандартными атомарными целочисленными типами	175
5.2.6. Основной шаблон класса <code>std::atomic<></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 <code>std::memory_order_seq_cst</code> для создания прототипа	314
7.3.2. Используйте подходящую схему освобождения памяти..	314
7.3.3. Помните о проблеме АВА.....	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 r -значения.....	437
A.1.1. Семантика перемещения.....	439
A.1.2. Ссылки на r -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

C++ Thread Library	492
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

D.6.5. Псевдоним шаблона `std::ratio_divide`..... 635
D.6.6. Шаблон класса `std::ratio_equal` 636
D.6.7. Шаблон класса `std::ratio_not_equal` 636
D.6.8. Шаблон класса `std::ratio_less`..... 637
D.6.9. Шаблон класса `std::ratio_greater` 637
D.6.10. Шаблон класса `std::ratio_less_equal` 638
D.6.11. Шаблон класса `std::ratio_greater_equal` 638
D.7. Заголовок `<thread>` 638
D.7.1. Класс `std::thread`..... 639
D.7.2. Пространство имен `this_thread` 649

РЕСУРСЫ 652

Печатные ресурсы..... 652
Сетевые ресурсы 653

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ 654



ПРЕДИСЛОВИЕ

С идеей многопоточного программирования я столкнулся на своей первой работе после окончания колледжа. Мы занимались приложением, которое должно было помещать входные записи в базу данных. Данных было много, но все они были независимы и требовали значительной предварительной обработки. Чтобы задействовать всю мощь нашего десятипроцессорного компьютера UltraSPARC, мы организовали несколько потоков, каждый из которых обрабатывал свою порцию входных данных. Код был написан на языке C++, с использованием потоков POSIX. Ошибок мы наделали кучу – многопоточность для всех была внове – но до конца все-таки добрались. Именно во время работы над этим проектом я впервые услышал о комитете по стандартизации C++ и о недавно опубликованном стандарте языка C++.

С тех мой интерес к многопоточному программированию и параллелизму не затухает. Там, где другим видятся трудности и источник разнообразных проблем, я нахожу мощный инструмент, который позволяет программе использовать всё наличное оборудование и в результате работать быстрее. Позднее я научился применять эти идеи и при наличии всего одного процессора или ядра, чтобы улучшить быстроту реакции и повысить производительность, – благодаря тому, что одновременная работа нескольких потоков дает программе возможность не простаивать во время таких длительных операций, как ввод/вывод. Я также узнал, как это устроено на уровне ОС и как в процессорах Intel реализовано контекстное переключение задач.

Тем временем интерес к C++ свел меня с членами Ассоциации пользователей C и C++ (ACCU), а затем с членами комиссии по стандартизации C++ при Институте стандартов Великобритании (BSI) и разработчиками библиотек Boost. Я с интересом наблюдал за началом разработки библиотеки многопоточности Boost, а когда автор забросил проект, я воспользовался шансом перехватить инициативу. С тех пор разработка и сопровождение библиотеки Boost Thread Library лежит в основном на мне.

По мере того как в работе комитета по стандартизации C++ наметился сдвиг от исправления дефектов в существующем стандарте в сторону выработки предложений для нового стандарта (получившего условное название C++0x в надежде, что его удастся завершить до 2009 года, и официально названного C++11, так как он наконец был опубликован в 2011 году), я стал принимать более активное участие в деятельности BSI и даже вносить собственные предложения. Когда стало ясно, что многопоточность стоит на повестке дня, я по-настоящему встрепенулся – многие вошедшие в стандарт предложения по многопоточности и параллелизму написаны как мной самим, так и в соавторстве с коллегами. Я считаю большой удачей, что таким образом удалось совместить две основных сферы моих интересов в области программирования – язык C++ и многопоточность.

В этой книге, опирающейся на весь мой опыт работы с C++ и многопоточностью, я ставил целью научить других программистов, как безопасно и эффективно пользоваться библиотекой C++11 Thread Library. Надеюсь, что мне удастся заразить читателей своим энтузиазмом.



БЛАГОДАРНОСТИ

Прежде всего, хочу сказать огромное спасибо своей супруге, Ким, за любовь и поддержку, которую она выказывала на протяжении работы над книгой, отнимавшей изрядную долю моего свободного времени за последние четыре года. Без ее терпения, ободрения и понимания я бы не справился.

Далее я хочу поблагодарить коллектив издательства Manning, благодаря которому эта книга появилась на свет: Марджана Баджи (Marjan Bace), главного редактора; Майкла Стивенса (Michael Stephens), его заместителя; Синтию Кейн (Cynthia Kane), моего редактора-консультанта; Карен Тегтмейер (Karen Tegtmeier), выпускающего редактора; Линду Ректенвальд (Linda Recktenwald), редактора; Кати Теннант (корректора) и Мэри Пирджис, начальника производства. Без их стараний вы не читали бы сейчас эту книгу. Я хочу также поблагодарить других членов комитета по стандартизации C++, которые подавали на рассмотрение материалы, относящиеся к многопоточности: Андрея Александреску (Andrei Alexandrescu), Пита Беккера (Pete Becker), Боба Блэйнера (Bob Blainer), Ханса Бема (Hans Boehm), Бимана Доуса (Beman Dawes), Лоуренса Кроула (Lawrence Crowl), Петера Димова (Peter Dimov), Джеффа Гарланда (Jeff Garland), Кевлина Хэнни (Kevlin Henney), Ховарда Хиннанта (Howard Hinnant), Бена Хатчингса (Ben Hutchings), Йана Кристоферсона (Jan Kristofferson), Дуга Ли (Doug Lea), Пола Маккинни (Paul McKenney), Ника Макларена (Nick McLaren), Кларка Нельсона (Clark Nelson), Билла Пью (Bill Pugh), Рауля Силвера (Raul Silvera), Герба Саттера (Herb Sutter), Детлефа Вольмана (Detlef Vollmann) и Майкла Вонга (Michael Wong), а также всех тех, кто рецензировал материалы, принимал участие в их обсуждении на заседаниях комитета и иными способами содействовал оформлению поддержки многопоточности и параллелизма в C++11.

Наконец, хочу выразить благодарность людям, чьи предложения позволили заметно улучшить книгу: д-ру Джейми Оллсопу (Jamie Allsop), Петеру Димову, Ховарду Хиннанту, Рикку Моллою (Rick

Molloy), Джонатану Уэйкли (Jonathan Wakely) и д-ру Расселу Уиндеру (Russel Winder). Отдельное спасибо Расселу за подробные рецензии и Джонатану, который в качестве технического редактора, тщательно проверил окончательный текст на предмет наличия вопиющих ошибок. (Все оставшиеся ошибки – целиком моя вина.) И напоследок выражаю признательность группе рецензентов: Райану Стивенсу (Ryan Stephens), Нилу Хорлоку (Neil Horlock), Джону Тэйлору младшему (John Taylor Jr.), Эзре Дживану (Ezra Jivan), Джошуа Хейеру (Joshua Heyer), Киту С. Киму (Keith S. Kim), Мишель Галли (Michele Galli) Майку Тянь Чжань Чжану (Mike Tian-Jian Jiang), Дэвиду Стронгу (David Strong), Роджеру Орпу (Roger Orr), Вагнеру Рикку (Wagner Rick), Майкй Буксасу (Mike Buksas) и Бас Воде (Bas Vodde). Также спасибо всем читателям предварительного издания, которые нашли время указать на ошибки и отметить места, нуждающиеся в уточнении.



ОБ ЭТОЙ КНИГЕ

Эта книга представляет собой углубленное руководство по средствам поддержки многопоточности и параллелизма в новом стандарте C++, от базового использования классов и функций из пространств имен `std::thread`, `std::mutex` и `std::async` до сложных вопросов, связанных с атомарными операциями и моделью памяти.

Структура книги

В первых четырех главах описываются различные библиотечные средства и порядок работы с ними.

Глава 5 посвящена низкоуровневым техническим деталям модели памяти и атомарных операций. В частности, рассматривается вопрос об использовании атомарных операций для задания ограничений на порядок выполнения других частей программы. Вводные главы на этом заканчиваются.

В главах 6 и 7 начинается изучение программирования на более высоком уровне, с примерами использования базовых средств для построения сложных структур данных – с блокировками (глава 6) и без блокировок (глава 7).

В главе 8 эта линия продолжается: даются рекомендации по проектированию многопоточных программ, рассматриваются аспекты, влияющие на производительность, и приводятся примеры реализации различных параллельных алгоритмов.

Глава 9 посвящена средствам управления потоками, рассматриваются пулы потоков, очереди работ и прерывание операций.

Тема главы 10 – тестирование и отладка: типы ошибок, методы их отыскания, способы тестирования и так далее.

В приложениях вы найдете краткое описание некоторых языковых средств, добавленных в новый стандарт и имеющих отношение к многопоточности; детали реализации библиотеки передачи сообщениями, упомянутой в главе 4, и полный справочник по библиотеке C++11 Thread Library.

На кого рассчитана эта книга

Если вы пишете многопоточный код на C++, то эта книга для вас. Если вы пользуетесь средствами многопоточности из стандартной библиотеки C++, то здесь вы найдете руководство по основным вопросам. Если вы работаете с другими библиотеками многопоточности, то описанные рекомендации и приемы все равно могут оказаться полезным подспорьем.

Предполагается владение языком C++ на рабочем уровне, но предварительное знакомство с новыми языковыми средствами необязательно – они описаны в приложении А. Также не требуются знания или опыт работы в области многопоточного программирования, хотя их наличие было бы плюсом.

Как пользоваться этой книгой

Если раньше вы не писали многопоточных программ, то я рекомендую читать книгу последовательно от начала до конца, опустив, быть может, кое-какие детали из главы 5. Глава 7 опирается на материал главы 5, поэтому если вы пропустите главу 5, то отложите также чтение седьмой главы.

Если вам не доводилось использовать новые языковые средства, вошедшие в стандарт C++11, то имеет смысл с самого начала бегло просмотреть приложение А, чтобы понимать приведенные в тексте примеры. Впрочем, в основном тексте упоминания о новых средствах графически выделены, так что, встретив что-то незнакомое, вы всегда можете обратиться к приложению.

Если вы располагаете обширным опытом написания многопоточного кода в других средах, то все-таки стоит просмотреть начальные главы, чтобы понять, как знакомые вам понятия соответствуют средствам из нового стандарта C++. Если вы планируете работать с атомарными переменными на низком уровне, то главу 5 следует изучить обязательно. Полезно также ознакомиться с главой 8, где рассказывается о безопасности исключений в многопоточных программах на C++. Если перед вами стоит конкретная задача, то указатель и оглавление помогут быстро найти соответствующий раздел.

Даже после того как вы освоите библиотеку C++ Thread Library, приложение D все равно останется полезным, потому что в нем легко найти детали использования каждого класса и функции. Время от времени вы, наверное, будет заглядывать и в основные главы, чтобы освежить в памяти порядок работы с той или иной конструкцией или взглянуть на пример кода.

Графические выделения и загрузка исходного кода

Исходный код в листингах и основном тексте набран моноширинным шрифтом. Многие листинги сопровождаются аннотациями, в которых излагаются важные концепции. В некоторых случаях в листингах присутствуют нумерованные маркеры, с которыми соотносятся последующие пояснения.

Исходный код всех примеров можно скачать с сайта издательства по адресу www.manning.com/CPlusPlusConcurrencyinAction.

Требования к программному обеспечению

Чтобы приведенный в этой книге код работал без модификаций, понадобится версия компилятора C++ с поддержкой тех вошедших в стандарт C++11 средств, которые перечислены в приложении А. Кроме того, нужна стандартная библиотека многопоточности C++ (Standard Thread Library).

На момент написания этой книги единственный известный мне компилятор, поставляемый с библиотекой Standard Thread Library, – это g++, хотя в предварительную версию Microsoft Visual Studio 2011 она также входит. Что касается g++, то первая реализация основных возможностей библиотеки многопоточности была включена в версию g++ 4.3, а впоследствии добавлялись улучшения и расширения. Кроме того, в g++ 4.3 впервые появилась поддержка некоторых новых языковых средств C++11, и в каждой новой версии она расширяется. Дополнительные сведения см. на странице текущего состояния реализации C++11 в g++¹.

В составе Microsoft Visual Studio 2010 также имеются некоторые новые средства из стандарта C++11, например лямбда-функции и ссылки на г-значения, но реализация библиотеки Thread Library отсутствует.

Моя компания, Just Software Solutions Ltd, продает полную реализацию стандартной библиотеки C++11 Standard Thread Library для Microsoft Visual Studio 2005, Microsoft Visual Studio 2008, Microsoft Visual Studio 2010 различных версий g++². Именно эта реализация применялась для тестирования примеров из этой книги.

¹ Страница состояния компилятора GNU C++0x/C++11 <http://gcc.gnu.org/projects/sxx0x.html>.

² Реализация just::thread библиотеки C++ Standard Thread Library, <http://www.stdthread.co.uk>.

В библиотеке Boost Thread Library³, протестированной на многих платформах, реализован API, основанный на предложениях, поданных в комитет по стандартизации C++. Большинство приведенных в книге примеров будут работать с Boost Thread Library, если заменить `std::` на `boost::` и включить подходящие директивы `#include`. Но некоторые возможности в библиотеке Boost Thread Library либо не поддерживаются вовсе (например, `std::async`), либо называются по-другому (например, `boost::unique_future`).

Автор в Сети

Приобретение книги «Параллелизм на C++ в действии» открывает бесплатный доступ к закрытому форуму, организованному издательством Manning Publications, где вы можете оставить свои комментарии к книге, задать технические вопросы и получить помощь от автора и других пользователей. Получить доступ к форуму и подписаться на список рассылки можно на странице www.manning.com/CPlusPlusConcurrencyinAction. Там же написано, как зайти на форум после регистрации, на какую помощь можно рассчитывать, и изложены правила поведения в форуме.

Издательство Manning обязуется предоставлять читателям площадку для общения с другими читателями и автором. Однако это не означает, что автор обязан как-то участвовать в обсуждениях; его присутствие на форуме остается чисто добровольным (и не оплачивается). Мы советуем задавать автору хитроумные вопросы, чтобы его интерес к форуму не угасал!

Форум автора в сети и архивы будут доступны на сайте издательства до тех пор, пока книга не перестанет печататься.

³ Библиотеки Boost для C++, <http://www.boost.org>.



ОБ ИЛЛЮСТРАЦИИ НА ОБЛОЖКЕ

Рисунок на обложке книги «Параллельное программирование на C++ в действии» называется «Традиционный костюм японской девушки». Репродукция взята из четырехтомного «Собрания костюмов разных народов», напечатанного в Лондоне между 1757 и 1772 годом. Это издание, включающее изумительные раскрашенные вручную гравюры на меди с изображениями одежды народов мира, оказало большое влияние на дизайн театральных костюмов. Разнообразие рисунков позволяет составить наглядное представление о великолепии костюма на Лондонской сцене свыше 200 лет назад. Костюмы, исторические и того времени, позволяли познакомиться с традиционной одеждой людей, живших в разное время в разных странах, и тем самым сделать их ближе и понятнее лондонской театральной публике.

Манера одеваться за последние 100 лет сильно изменилась, и различия между областями, когда-то столь разительные, сгладились. Теперь трудно отличить друг от друга даже выходцев с разных континентов. Но можно взглянуть на это и с оптимизмом – мы обменяли культурное и визуальное разнообразие на иное устройство личной жизни – основанное на многостороннем и стремительном технологическом и интеллектуальном развитии.

Издательство Manning откликается на новации, инициативы и курьезы в компьютерной отрасли обложками своих книг, на которых представлено широкое разнообразие местных укладов и театральной жизни в позапрошлом веке. Мы возвращаем его в виде иллюстраций из этой коллекции.



ГЛАВА 1.

Здравствуй, параллельный мир!

В этой главе:

- Что понимается под параллелизмом и многопоточностью.
- Зачем использовать параллелизм и многопоточность в своих приложениях.
- Замечания об истории поддержки параллелизма в C++.
- Структура простой многопоточной программы на C++.

Для программистов на языке C++ настали радостные дни. Спустя тринадцать лет после публикации первой версии стандарта C++ в 1998 году комитет по стандартизации C++ решил основательно пересмотреть как сам язык, так и поставляемую вместе с ним библиотеку. Новый стандарт C++ (обозначаемый C++11 или C++0x), опубликованный в 2010 году, несет многочисленные изменения, призванные упростить программирование на C++ и сделать его более продуктивным.

К числу наиболее существенных новшеств в стандарте C++11 следует отнести поддержку многопоточных программ. Впервые комитет официально признал существование многопоточных приложений, написанных на C++, и включил в библиотеку компоненты для их разработки. Это позволит писать на C++ многопоточные программы с гарантированным поведением, не полагаясь на зависящие от платформы расширения. И как раз вовремя, потому что разработчики,

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

e-Univers.ru