Посвящается моей семье –
брату, родителям, а особенно моей любимой супруге
и моим фантастическим детям,
которые делают каждый день моей жизни
абсолютным счастьем.

– Карл Фредерик Самсон

Оглавление

Предисловие от издательства 12
Об авторе 13
О технических рецензентах 14
Предисловие 15
Для кого написана эта книга 16
Структура книги 16
Как получить максимум пользы от этой книги 18
Загрузка примеров кода 19
Обозначения и графические выделения 19
Оставайтесь на связи 19
Поделитесь своими мыслями 20

ЧАСТЬ I. ОСНОВЫ АСИНХРОННОГО ПРОГРАММИРОВАНИЯ 21

Глава 1. Конкурентность и асинхронное программирование: подробный обзор 22
Технические требования 22
Эволюция многозадачности 23
Невытесняющая многозадачность 23
Вытесняющая многозадачность 24
Гиперпоточность 24
Многоядерные процессоры 25
Действительно ли вы пишете синхронный код? 25
Конкурентность и параллелизм 26
Моя умозрительная модель 27
Проведем параллели с экономикой производства 27
Связь конкурентности с вводом-выводом 29
А как насчет потоков, предоставляемых операционной системой? 30
Выбор правильной системы отсчета 30
Асинхронность и конкурентность 31
Роль операционной системы 31
Конкурентность с точки зрения операционной системы 31
Налаживание контакта с операционной системой 32
Взаимодействие с операционной системой 32
CPU и операционная система 33
Вниз по кроличьей норе 34

Как CPU предотвращает доступ к памяти,
к которой запрещено обращаться? 35
Но не можем ли мы изменить таблицу страниц в CPU? 35
Прерывания, прошивки и ввод-вывод 36
Упрощенный обзор 37
Прерывания 39
Прошивка 40
Резюме 40

Глава 2. Как языки программирования моделируют асинхронный поток программы 41
Определения 42
Потоки 43
Потоки, предоставляемые операционной системой 45
Создание потоков требует времени 45
У каждого потока свой стек 45
Контекстное переключение 46
Планирование 46
Преимущества отделения асинхронных операций от потоков ОС 47
Пример 48
Волокна и зеленые потоки 49
Все стеки имеют фиксированный размер 50
Контекстное переключение 51
Планирование 51
Интерфейс внешних функций (FFI) 52
Подходы на основе обратных вызовов 53
Сопрограммы: обещания и будущие объекты 54
Сопрограммы и async/await 55
Резюме 57

Глава 3. Поддержка ОС для очередей событий, системных вызовов и кросс-платформенных абстракций 58
Технические требования 59
Выполнение примеров для Linux 60
Зачем использовать очередь событий, поддерживаемую ОС? 60
Блокирующий ввод-вывод 61
Неблокирующий ввод-вывод 61
Опрос событий с помощью epoll/kqueue и IOCP 61
Очереди событий на основе готовности 62
Очереди событий на основе завершения 63
epoll, kqueue и IOCP 64
Кросс-платформенные очереди событий 65
Системные вызовы, FFI и кросс-платформенные абстракции 66
Самый нижний уровень абстракции 66
Следующий уровень абстракции 69
Самый верхний уровень абстракции 74
Резюме 74

ЧАСТЬ II. ОЧЕРЕДИ СОБЫТИЙ И ЗЕЛЕНЫЕ ПОТОКИ 77

Глава 4. Создаем собственную очередь событий 78

Технические требования 78
Знакомство с механизмом epoll и его структурой 79
Все ли операции ввода-вывода блокирующие? 84
Модуль ffi 86
Битовые флаги и битовые маски 89
События, срабатывающие по фронту и по уровню 90
Модуль Poll 93
Главная программа 96
Резюме 103

Глава 5. Создание волокон 105

Технические требования 106
Как пользоваться репозиторием, прилагаемым к этой книге 106
Предварительная информация 107
Системы команд, аппаратные архитектуры и ABI 107
System V ABI для x86-64 109
Краткое введение в язык ассемблера 111
Пример, от которого мы будем отталкиваться 112
Подготовка проекта 112
Введение в макрос встроенного ассемблера в Rust 115
Выполнение примера 117
Стек 118
Как выглядит стек? 118
Размер стека 121
Реализация волокон 121
Реализация среды исполнения 124
Функции guard, skip и switch 130
Заключительные мысли 133
Резюме 134

ЧАСТЬ III. БУДУЩИЕ ОБЪЕКТЫ И ASYNC/AWAIT В RUST 137

Глава 6. Будущие объекты в Rust 138

Что такое будущий объект? 138
Листовые будущие объекты 139
Нелистовые будущие объекты 139
Умозрительная модель асинхронной среды выполнения 140
Что берут на себя язык Rust и стандартная библиотека 142
Задачи ввода-вывода и счетные задачи 142
Резюме 143

Глава 7. Сопрограммы и async/await 145

Технические требования ... 145
Введение в бесстековые сопрограммы ... 145
Примеры написанных вручную сопрограмм ... 147
Модуль Futures ... 148
Модуль HTTP ... 149
Обязаны ли будущие объекты быть ленивыми? ... 153
Создание сопрограмм ... 154
async/await ... 160
coroutine/wait ... 161
corofy – препроцессор сопрограмм ... 162
b-async-await – пример преобразования coroutine/wait ... 163
c-async-await – конкурентные будущие объекты ... 166
Заключительные мысли ... 170
Резюме ... 171

Глава 8. Среды исполнения, будильники и паттерн Реактор–Исполнитель ... 172

Технические требования ... 173
Что такое среды исполнения и почему они нужны ... 173
Реакторы и исполнители ... 175
Улучшение базового примера ... 176
Дизайн ... 178
Изменение текущей реализации ... 180
Создание правильной среды исполнения ... 186
Шаг 1: улучшение дизайна среды исполнения путем добавления реактора и будильника ... 189
Создание Waker ... 190
Изменение определения Future ... 192
Шаг 2: реализация исполнителя ... 194
Шаг 3: реализация реактора ... 201
Эксперименты с новой средой исполнения ... 208
Пример использования конкурентности ... 208
Выполнение нескольких будущих объектов конкурентно и параллельно ... 209
Резюме ... 211

Глава 9. Сопрограммы, автореферентные структуры и закрепление ... 212

Технические требования ... 213
Первое улучшение примера – переменные ... 213
Подготовка базовой версии примера ... 213
Улучшение базового примера ... 215
Второе улучшение примера – ссылки ... 220
Третье улучшение примера – это… не… хорошо ... 225
Автореферентные структуры ... 226
Что такое перемещение? ... 228

Закрепление в Rust ... 230
Теория закрепления ... 231
Определения ... 231
Закрепление в куче ... 232
Закрепление в стеке ... 234
Pin-проекции и структурное закрепление ... 236
Четвертое улучшение примера: закрепление спешит на помощь ... 238
future.rs ... 238
http.rs ... 238
main.rs ... 239
executor.rs ... 242
Резюме ... 243

Глава 10. Создаем свою среду исполнения ... 245

Технические требования ... 245
Создание собственной среды исполнения, поддерживающей будущие объекты и async/await ... 246
Подготовка примера ... 247
main.rs ... 247
future.rs ... 248
http.rs ... 248
executor.rs ... 249
reactor.rs ... 252
Экспериментирование с нашей средой исполнения ... 254
Проблемы асинхронного Rust ... 257
Явное и неявное создание экземпляра реактора ... 257
Эргономика против эффективности и гибкости ... 259
Общие характеристики, с которыми все согласны ... 260
Асинхронное уничтожение ... 261
Будущее асинхронного Rust ... 261
Резюме ... 262
Эпилог ... 264

Предметный указатель ... 266

Предисловие от издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте *www.dmkpress.com*, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу *dmkpress@gmail.com*; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу *http://dmkpress.com/authors/publish_book/* или напишите в издательство по адресу *dmkpress@gmail.com*.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу *dmkpress@gmail.com*, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу *dmkpress@gmail.com*.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об авторе

Карл Фредерик Самсон – популярный автор, пишущий о технологиях. Проявляет активность в сообществе Rust с 2018 года. Имеет степень магистра по бизнес-администрированию, специализируется в стратегическом планировании и финансах. Когда не пишет книги, занимается двумя своими детьми и работает исполнительным директором компании, насчитывающей 300 сотрудников. В жизни интересовался разными технологиями, имеет опыт программирования как на старых мейнфреймах IBM, так и в современных облаках – от ассемблера до Visual Basic for Applications. Участвовал в нескольких проектах с открытым исходным кодом, включая написание официальной документации по асинхронному программированию на Rust.

Я хочу поблагодарить сообщество Rust за конструктивность, позитивность и гостеприимство. Этой книги не было бы, если бы не благожелательное и содержательное взаимодействие с сообществом. Отдельное спасибо авторам библиотек, лежащих в основе современной асинхронной экосистемы: mio, Tokio и async-std.

Я также благодарен своему редактору, Киннари, которая проявляла необыкновенное терпение и была готова прийти на помощь на всем протяжении работы над книгой.

О технических рецензентах

Евгений Пирянов – опытный инженер-программист, обладающий глубокими знаниями в области серверных технологий, Web3 и блокчейна. Евгений получил ученую степень по инженерии в Имперском колледже, Лондон, и несколько лет разрабатывал нелинейные решатели на C++. С тех пор он находился на переднем крае создания архитектуры, проектирования и реализации децентрализованных приложений в области Defi (децентрализованных финансовых сервисов) и метавселенной. Страсть Евгения к языку Rust не знает границ, и он искренне верит в его светлое будущее и широкий спектр применения.

Яаге Ху специализируется на системном программировании и архитектуре компиляторов. Он писал код для таких компаний, как Uber, Amazon и Meta[1], а сейчас занимается системными исследованиями в области WebAssembly и Rust. У Яаге и его жены недавно родился первенец, девочка Максин.

[1] Деятельность компании признана экстремистской и запрещена в РФ.

Предисловие

Материал этой книги первоначально был изложен в серии более коротких книг для программистов, желающих изучить асинхронное программирование на Rust с нуля. Те материалы, которые мне удалось найти в то время, были в равной мере разочаровывающими, поучительными и сбивающими с толку, поэтому я захотел с этим что-то сделать.

Вышеупомянутые коротенькие книжки завоевали популярность, поэтому когда я получил шанс написать все во второй раз, улучшить те части, которыми был доволен, и полностью переписать все остальное, собрав в единую связную книгу, оставалось только сесть и сделать. Результат – перед вами.

Люди начинают программировать по разным причинам. Ученые – чтобы строить модели и производить расчеты. Бизнесмены – для решения конкретных задач, стоящих перед их предприятиями. Кто-то поначалу рассматривает программирование как хобби и занимается им в свободное время. Общим для всех таких программистов является то, что они изучают программирование сверху вниз.

По большей части в этом нет ничего плохого, но когда речь заходит об асинхронном программировании вообще и на Rust в частности, гораздо лучше начинать изучение с фундаментальных основ, и настоящая книга призвана стать для вас подспорьем в этом деле.

Асинхронное программирование – это способ разбиения программы на задачи, которые можно приостанавливать и возобновлять в определенных точках. Это, в свою очередь, позволяет языковой библиотеке времени выполнения управлять данными задачами так, чтобы они могли продвигаться к завершению, чередуясь.

Асинхронное программирование по сути своей влияет на весь поток программы и пронизывает ее снизу доверху. Вы должны переписать свою программу, изменив порядок планирования и выполнения ее частей способом, отнюдь не всегда очевидным программисту.

Большинство языков стремятся сделать асинхронное программирование настолько простым, чтобы для продуктивной работы вам не нужно было понимать, как оно работает на самом деле. И на Rust можно вполне продуктивно писать асинхронные программы, не зная, как это работает, однако Rust подходит к делу

более явно и оставляет на поверхности больше сложности, чем большинство других языков. И совладать с этой сложностью будет куда проще, если глубоко разобраться в асинхронном программировании вообще и на Rust в частности.

Еще один большой плюс изучения теоретических основ заключается в том, что приобретенные знания можно будет применить и за пределами Rust, а это, в свою очередь, упростит освоение асинхронного программирования в других языках. Я бы даже рискнул утверждать, что большая часть этих знаний пригодится даже в повседневном программировании. По крайней мере со мной было так.

Я хочу, чтобы, читая эту книгу, вы чувствовали, что вместе со мной отправились в путешествие, где мы будем осваивать одну тему за другой и попутно учиться на примерах и экспериментах. Я не хочу выглядеть как лектор, просто повествующий, как все устроено.

Эта книга написана для людей, по природе любопытных, для тех программистов, которые хотят разобраться в используемых системах, и для тех, кто любит ставить простые и сложные эксперименты, считая их способом исследования и обучения.

Для кого написана эта книга

Эта книга предназначена разработчикам, уже имеющим опыт программирования и желающим освоить асинхронное программирование с нуля, чтобы в совершенстве овладеть асинхронным программированием на Rust и получить возможность участвовать в технических дискуссиях по этой теме. Книга идеально подходит для тех, кто любит писать рабочие примеры – чтобы их можно было разбирать на части, расширять и экспериментировать с ними.

Мне кажется, что эта книга должна особенно понравиться двум категориям людей:

- разработчикам, которые раньше работали с языками более высокого уровня со встроенным сборщиком мусора, интерпретатором или средой исполнения: C#, Java, JavaScript, Python, Ruby, Swift или Go. Для программистов, имеющих богатый опыт асинхронного программирования на любом из таких языков, но желающих изучить его фундаментальные основы, а также для программистов вообще без опыта асинхронного программирования эта книга будет в равной степени полезной;
- разработчикам, знакомым с такими языками, как C или C++, которые имеют ограниченный опыт асинхронного программирования.

Структура книги

В главе 1 «Конкурентность и асинхронное программирование: подробный обзор» приведена краткая история вопроса, которая привела к асинхронному программированию в том виде, в каком мы его знаем. Мы дадим несколько важных определений и представим умозрительную модель, которая объясняет, какого рода задачи в действительности решает асинхронное программирование и чем конкурентность отличается от параллелизма. Мы также рассмотрим важность выбора правильной системы отсчета при обсуждении асинхронного выполнения программы и обсудим несколько важных и фундаментальных

концепций, касающихся процессоров, операционных систем, оборудования, прерываний и ввода-вывода.

В главе 2 «Как языки программирования моделируют асинхронный поток программы» рамки предыдущей главы сужаются, и основное внимание уделяется подходам к асинхронному программированию в различных языках. В начале дается несколько важных определений, а затем объясняются стековые и бесстековые сопрограммы, потоки ОС, зеленые потоки, волокна, обратные вызовы, обещания, будущие объекты и механизм async/await.

В главе 3 «Поддержка ОС для очередей событий, системных вызовов и кросс-платформенных абстракций» объясняется, что такое epoll, kqueue и IOCP и чем они отличаются друг от друга. Здесь же готовится почва для последующих глав: дается введение в системные вызовы, интерфейс внешних функций (FFI) и кросс-платформенные абстракции.

В главе 4 «Создаем собственную очередь событий» вы создадите свою собственную очередь событий по образцу API *mio* (популярной библиотеки Rust, которая лежит в основе большей части современной экосистемы асинхронности). Пример вращается вокруг системного вызова epoll, так что у нас будет возможность немного поговорить о том, как он работает.

В главе 5 «Создание волокон» мы на примере проработаем вопрос о создании стековых сопрограмм, которые называются волокнами. Это вариант зеленых потоков, используемых в Go, и одна из самых широко распространенных и популярных альтернатив той абстракции, которая в Rust представлена будущими объектами и механизмом async/await. Rust использовал эту абстракцию на заре развития, еще до появления версии 1.0, так что она тоже является частью его истории. В этой главе также рассматривается несколько общих концепций программирования – стеки, язык ассемблера, **двоичный интерфейс приложения** (**ABI**) и **архитектура системы команд** (**ISA**), – полезных не только в контексте асинхронного программирования.

Глава 6 «Будущие объекты в Rust» содержит краткое введение и обзор будущих объектов, сред исполнения и асинхронного программирования в Rust.

В главе 7 «Сопрограммы и async/await» вы напишете свои первые сопрограммы, являющиеся упрощенными версиями тех, что создаются с помощью механизма async/await в современном Rust. Несколько таких сопрограмм мы напишем вручную и познакомимся с новым синтаксисом, который позволяет программно переписать то, что кажется обычными функциями, создавая из них сопрограммы.

В главе 8 «Среды исполнения, будильники и паттерн Реактор–Исполнитель» мы познакомимся со средами исполнения и их дизайном. Продолжая пример из главы 7, мы создадим среду исполнения своих сопрограмм, а затем будем ее постепенно улучшать. Мы поэкспериментируем со своей средой исполнения, чтобы лучше понять, как она работает.

В главе 9 «Сопрограммы, автореферентные структуры и закрепление» мы познакомимся с автореферентными структурами и закреплением в Rust. Продолжая совершенствовать свои сопрограммы, мы на собственном опыте узнаем, зачем нужны такие механизмы, как `Pin`, и как он помогает решать встречающиеся на практике проблемы.

В главе 10 «Создаем свою среду исполнения» мы наконец соберем все вместе. Мы еще улучшим сквозной пример, начатый в предыдущих главах, чтобы можно было выполнить будущие объекты Rust. Это позволит нам использовать всю мощь async/await и асинхронного Rust. Мы поставим несколько экспериментов и продемонстрируем, с какими трудностями сталкивается асинхронный Rust и как их лучше всего разрешить.

Как получить максимум пользы от этой книги

У вас должен быть какой-то опыт программирования, и хорошо, если вы уже что-то знаете о Rust. Ознакомление с бесплатной и очень хорошей книгой начального уровня «*The Rust Programming Language*» (*https://doc.rust-lang.org/book/*) должно дать вам более чем достаточно знаний о Rust, чтобы двигаться дальше, поскольку вопросы повышенной сложности будут объясняться шаг за шагом.

В идеале эту книгу следует читать, положив ее рядом с открытым редактором кода. Под рукой также должен быть прилагаемый к книге репозиторий, к которому можно обращаться в случае возникновения вопросов.

Программное и аппаратное обеспечение, рассматриваемые в этой книге	Требования к операционной системе
Rust (версия не ниже 1.51)	Windows, macOS или Linux

Вам понадобится Rust. Если вы еще не установили его, ознакомьтесь с инструкциями по адресу *https://www.rustlang.org/tools/install*.

Для некоторых примеров нужно будет использовать **подсистему Windows для Linux** (**WSL**) в Windows. Если вы прорабатываете книгу на машине с Windows, то я рекомендую активировать WSL (*https://learn.microsoft.com/en-us/windows/wsl/install*) прямо сейчас и установить Rust, следуя инструкциям по адресу *https://www.rust-lang.org/tools/install*.

Если вы пользуетесь цифровой версией книги, то я рекомендую набирать код самостоятельно или брать его из репозитория книги на GitHub (ссылка приведена в следующем разделе). Это поможет избежать потенциальных ошибок из-за копирования и вставки кода.

Репозиторий организован следующим образом:

- код, относящийся к некоторой главе, находится в папке этой главы (например, `ch01`);
- каждому примеру соответствует отдельный крейт;
- буквы в начале имен примеров показывают порядок следования примеров в книге. Например, пример `a-runtime` предшествует примеру `b-reactor-executor`. При таком именовании файлы примеров будут упорядочены хронологически (по крайней мере по умолчанию в большинстве систем);
- для некоторых примеров имеется версия с суффиксом `-bonus`. Такие версии упоминаются в тексте книге и часто содержат вариант примера, который может быть интересно посмотреть, хотя для рассматриваемой темы он не очень важен.

Загрузка примеров кода

Примеры кода для этой книги можно скачать с GitHub по адресу *https://github.com/PacktPublishing/Asynchronous-Programming-in-Rust*. Все обновления будут отражены в репозитории.

В нашем богатом каталоге книг и видео по адресу *https://github.com/PacktPublishing/* есть и другие пакеты кода. Не пропустите!

Обозначения и графические выделения

В этой книге применяется ряд соглашений о наборе текста.

`Код в тексте`: обозначает код в тексте, имена таблиц базы данных, папок и файлов, расширения имен файлов, пути к файлам, URL-адреса, данные, вводимые пользователем, и адреса в Твиттере. Например: «Итак, мы создали свою собственную асинхронную среду исполнения, в которой используются объекты Rust `Future`, `Waker`, `Context` и механизм `async/await`».

Отдельно стоящие фрагменты кода набраны так:

```
pub trait Future {
  type Output;
  fn poll(&mut self) -> PollState<Self::Output>;
}
```

Желая привлечь внимание к определенной части кода, мы выделяем соответствующие строки или элементы полужирным шрифтом:

```
struct Coroutine0 {
  stack: Stack0,
  state: State0,
}
```

Ввод и вывод программы в командной строке обозначаются так:

```
$ cargo run
```

> **Советы или важные замечания**
>
> Выглядят так.

Оставайтесь на связи

Мы всегда рады отзывам читателей.

Отзывы общего характера. Если у вас имеются какие-нибудь вопросы по этой книге, отправьте сообщение на адрес *customercare@packtpub.com*, указав в теме ее название.

Ошибки и опечатки. Мы проверяли содержимое книги со всем тщанием, но какие-то ошибки все же могли проскользнуть. Если вы найдете в нашей книге ошибку, пожалуйста, сообщите нам о ней. Зайдите на страницу *www.packtpub.com/support/errata* и заполните форму.

Нарушение авторских прав. Если вы обнаружите незаконные копии наших изданий в любой форме в интернете, пожалуйста, сообщите нам адрес или название веб-сайта. Просим отправить ссылку на вызывающий подозрение в пиратстве материал по адресу *copyright@packt.com*.

Если вы хотите стать автором. Если вы являетесь специалистом по какой-то теме и хотели бы стать автором или соавтором книги, заходите на страницу *authors.packtpub.com*.

Поделитесь своими мыслями

Если вы прочли эту книгу, мы были бы рады услышать ваше мнение! Перейдите по адресу этой книги на Amazon (*https://packt.link/r/1805128132*) и оставьте отзыв.

Ваш отзыв важен для нас и технического сообщества, потому что поможет нам издавать книги, содержащие высококачественный материал.

Конец ознакомительного фрагмента.
Приобрести книгу можно
в интернет-магазине
«Электронный универс»
e-Univers.ru