

RUST

В ДЕЙСТВИИ

Тим Макнамара



Оглавление

Предисловие	15
Благодарности	17
О книге	19
Кому следует прочитать эту книгу	19
Организация книги: дорожная карта	19
О программном коде	21
Дискуссионный форум liveBook	21
Другие онлайн-ресурсы	22
Об авторе	22
Об иллюстрации на обложке книги	22
Глава 1. Введение в Rust	23
1.1. Где используется Rust?	24
1.2. С какой целью была написана книга «Rust в действии»	25
1.3. Вкус языка	26
1.3.1. Хитрый путь к «Hello, world!»	27
1.3.2. Ваша первая программа на Rust	29
1.4. Загрузка исходного кода книги	30
1.5. На что похож Rust?	31
1.6. В чем заключается особенность Rust?	34
1.6.1. Цель создания Rust: безопасность	36
1.6.2. Цель создания Rust: производительность	41
1.6.3. Цель создания Rust: управляемость	43
1.7. Особые возможности Rust	45
1.7.1. Достижение высокой производительности	45
1.7.2. Многопоточное выполнение программ	46
1.7.3. Достижение эффективной работы с памятью	46

1.8. Недостатки Rust.....	46
1.8.1. Циклические структуры данных	46
1.8.2. Время, затрачиваемое на компиляцию	46
1.8.3. Строгость	47
1.8.4. Объем языка	47
1.8.5. Излишний ажиотаж	47
1.9. Примеры использования TLS-безопасности	47
1.9.1. Heartbleed.....	48
1.9.2. Goto fail;	48
1.10. Для чего Rust подходит лучше всего?.....	50
1.10.1. В утилитах командной строки	50
1.10.2. В обработке данных.....	51
1.10.3. В расширяемых приложениях	51
1.10.4. В средах с ограниченными ресурсами.....	51
1.10.5. В серверных приложениях	52
1.10.6. В приложениях для ПК.....	52
1.10.7. В автономном режиме	52
1.10.8. В мобильных приложениях.....	53
1.10.9. В веб-режиме.....	53
1.10.10. В системном программировании.....	53
1.11. Скрытая фишка Rust: его сообщество.....	54
1.12. Разговорник по Rust.....	54
Резюме.....	54
Часть I. Особенности языка Rust.....	57
Глава 2. Основы языка	59
2.1. Создание работоспособной программы.....	60
2.1.1. Компиляция одиночных файлов с помощью утилиты <code>rustc</code>	60
2.1.2. Компиляция Rust-проектов с использованием <code>cargo</code>	61
2.2. Взгляд на синтаксис Rust.....	62
2.2.1. Определение переменных и вызов функций	63
2.3. Числа	64
2.3.1. Целые и десятичные (с плавающей точкой) числа	65
2.3.2. Записи целых чисел с основанием 2, 8 и 16	66
2.3.3. Сравнение чисел	68
2.3.4. Рациональные, комплексные числа и другие числовые типы	73

2.4. Управление ходом выполнения программы.....	76
2.4.1. For: основной механизм итераций.....	76
2.4.2. Continue: пропуск оставшейся части текущей итерации	78
2.4.3. While: выполнение цикла, пока не изменится состояние условия	78
2.4.4. Loop: основа для циклических конструкций Rust	79
2.4.5. Break: прерывание цикла.....	80
2.4.6. If, if else и else: условное ветвление	81
2.4.7. Match: соответствие образцу с учетом типов	82
2.5. Определение функций	84
2.6. Использование указателей	85
2.7. Проект: визуализация множества Мандельброта	86
2.8. Расширенные определения функций.....	90
2.8.1. Явные аннотации времени жизни	90
2.8.2. Обобщенные функции.....	92
2.9. Создание grep-lite	95
2.10. Создание списков с использованием массивов, слайсов и векторов	99
2.10.1. Массивы	99
2.10.2. Слайсы	101
2.10.3. Векторы.....	102
2.11. Включение стороннего кода	104
2.11.2. Создание документации по сторонним контейнерам в локальной среде	107
2.11.3. Управление имеющимся в Rust набором инструментальных средств с помощью <code>gustup</code>	107
2.12. Поддержка аргументов командной строки.....	108
2.13. Чтение данных из файлов.....	110
2.14. Чтение из стандартного устройства ввода <code>stdin</code>	113
Резюме.....	114
Глава 3. Составные типы данных	117
3.1. Использование простых функций для экспериментов с API.....	117
3.2. Моделирование файлов с помощью <code>struct</code>	120
3.3. Добавление методов к структуре <code>struct</code> путем использования блока <code>impl</code>	125
3.3.1. Упрощение создания объектов за счет реализации метода <code>new ()</code>	126
3.4. Возвращение сообщений об ошибках.....	129
3.4.1. Изменение значения известной глобальной переменной	129
3.4.2. Использование возвращаемого типа <code>Result</code>	135

3.5. Определение и использование перечисления <code>enum</code>	138
3.5.1. Использование <code>enum</code> для управления внутренним состоянием	141
3.6. Определение общего поведения с помощью типажей	143
3.6.1. Создание типажа <code>Read</code>	143
3.6.2. Реализация <code>std::fmt::Display</code> для ваших собственных типов.....	145
3.7. Выставление своих типов на всеобщее обозрение	148
3.7.1. <code>Protecting private data</code> Защита личных данных	148
3.8. Создание встроенной документации ваших проектов	149
3.8.1. Использование <code>gustdoc</code> для визуализации документов, касающихся одного исходного файла.....	151
3.8.2. Использование <code>cargo</code> для визуализации документов для контейнера и его зависимостей	151
Резюме.....	153
Глава 4. Время жизни, владение и заимствование.....	155
4.1. Реализация имитации наземной станции <code>CubeSat</code>	156
4.1.1. Выявление первой проблемы, связанной со временем жизни.....	158
4.1.2. Особое поведение элементарных типов	161
4.2. Справочник по рисункам, используемым в этой главе	163
4.3. Кто такой владелец? Есть ли у него какие-либо обязанности?	164
4.4. Как происходит переход владения	165
4.5. Решение проблем, связанных с владением.....	167
4.5.1. Если полное владение не требуется, используйте ссылки.....	170
4.5.2. Сократите количество долгоживущих значений	173
4.5.3. Продублируйте значение	180
4.5.4. Заключите данные в специальные типы	184
Резюме.....	186
Часть II. Демистификация системного программирования	189
Глава 5. Углубленное изучение данных	191
5.1. Комбинации битов и типы	191
5.2. Жизнь целых чисел	194
5.2.1. Усвоение порядка следования байтов.....	197
5.3. Представление десятичных чисел	198
5.4. Числа с плавающей точкой	199
5.4.1. Взгляд на <code>f32</code> изнутри.....	200
5.4.2. Выделение знакового бита.....	201

5.4.3. Выделение экспоненты	202
5.4.4. Выделение мантиссы	203
5.4.5. Разбиение числа с плавающей точкой на составные части	205
5.5. Форматы чисел с фиксированной точкой	207
5.6. Генерация случайных вероятностей из случайных байтов	213
5.7. Реализация центрального процессора (CPU), чтобы удостовериться, что функции также являются данными.....	215
5.7.1. CPU RIA/1: сумматор	215
5.7.2. Полный листинг кода для CPU RIA/1: сумматор	220
5.7.3. CPU RIA/2: мультипликатор.....	222
5.7.4. CPU RIA/3: блок вызова.....	226
5.7.5. CPU 4: добавление всего остального	233
Резюме.....	233
Глава 6. Память.....	235
6.1. Указатели	235
6.2. Исследование типов ссылок и указателей, имеющих в Rust.....	238
6.2.1. Обычные указатели, используемые в Rust	243
6.2.2. Экосистема указателей Rust.....	246
6.2.3. Строительные блоки интеллектуальных указателей.....	248
6.3. Предоставление программам памяти для размещения их данных.....	249
6.3.1. Стек	250
6.3.2. Куча	252
6.3.3. Что такое динамическое распределение памяти?	256
6.3.4. Анализ влияния, оказываемого динамическим выделением памяти	264
6.4. Виртуальная память	266
6.4.1. История вопроса.....	266
6.4.2. Шаг 1. Сканирование процессом собственной памяти	267
6.4.3. Преобразование виртуальных адресов в физические.....	271
6.4.4. Шаг 2. Работа с операционной системой для сканирования адресного пространства.....	274
6.4.5. Шаг 3. Чтение и запись в память процесса.....	277
Резюме.....	277
Глава 7. Файлы и хранилища.....	279
7.1. Что такое формат файла?	279
7.2. Создание собственных форматов файлов для хранения данных.....	281
7.2.1. Запись данных на диск с помощью <code>serde</code> и формата <code>bincode</code>	281

7.3. Реализация клона hexdump.....	284
7.4. Файловые операции, проводимые в Rust.....	288
7.4.1. Открытие файла в Rust и управление его режимом доступности.....	288
7.4.2. Безопасное взаимодействие с файловой системой с помощью std::fs::Path	289
7.5. Реализация хранилища «ключ-значение» с архитектурой, структурированной по записям и доступной только для добавления	291
7.5.1. Модель «ключ-значение».....	291
7.5.2. Представление actionkv v1: хранилище ключей и значений в памяти с интерфейсом командной строки.....	292
7.6. Actionkv v1: интерфейсный код.....	293
7.6.1. Настройка продукта условной компиляции	296
7.7. Понимание сути actionkv: контейнер libactionkv	298
7.7.1. Инициализация структуры ActionKV	298
7.7.2. Работа с отдельно взятой записью	302
7.7.3. Запись многобайтных двоичных данных на диск в гарантированном порядке следования байтов	304
7.7.4. Проверка ошибок ввода-вывода с помощью контрольных сумм	306
7.7.5. Вставка в существующую базу данных новой пары «ключ-значение».....	309
7.7.6. Полный код листинга для actionkv.....	310
7.7.7. Работа с ключами и значениями с использованием HashMap и BTreeMap	315
7.7.8. Создание HashMap и ее заполнение значениями.....	318
7.7.9. Извлечение значений из HashMap и BTreeMap	319
7.7.10. Что выбрать: HashMap или BTreeMap?	320
7.7.11. Добавление к actionkv v2.0 индекса базы данных	322
Резюме.....	326
Глава 8. Работа в сети	327
8.1. Все о сетевой работе в семи абзацах	328
8.2. Создание HTTP GET-запроса с использованием reqwest.....	330
8.3. Типажные объекты.....	332
8.3.1. На что способны типажные объекты?	332
8.3.2. Что такое типажные объекты?.....	333
8.3.3. Создание небольшой ролевой игры: rpg-проект	333
8.4. TCP	337
8.4.1. Что такое номер порта?.....	339
8.4.2. Преобразование имени хоста в IP-адрес.....	339

8.5. Способы обработки ошибок, наиболее удобные для помещения в библиотеки.....	347
8.5.1. Проблема: невозможность возвращения нескольких типов ошибок.....	347
8.5.2. Заключение в оболочку нижестоящих ошибок путем определения нашего собственного типа ошибки.....	351
8.5.3. Фокусы с <code>unwrap()</code> и <code>expect()</code>	358
8.6. MAC-адреса.....	358
8.6.1. Создание MAC-адресов.....	360
8.7. Реализация конечных автоматов с помощью перечислений.....	362
8.8. Чистый TCP.....	363
8.9. Создание виртуального сетевого устройства.....	363
8.10. «Чистый» HTTP.....	365
Резюме.....	376
Глава 9. Время и хронометраж.....	377
9.1. Предыстория вопроса.....	378
9.2. Источники времени.....	380
9.3. Определения.....	380
9.4. Кодирование времени.....	382
9.4.1. Представление часовых поясов.....	383
9.5. <code>clock v0.1.0</code> : учим приложение сообщать о времени.....	383
9.6. <code>clock v0.1.1</code> : форматирование меток времени в соответствии с ISO 8601 и стандартами электронной почты.....	384
9.6.1. Реструктуризация кода <code>clock v0.1.0</code> с целью более широкой архитектурной поддержки.....	385
9.6.2. Форматирование времени.....	386
9.6.3. Предоставление полноценного интерфейса командной строки.....	387
9.6.4. <code>clock v0.1.1</code> : полный проект.....	388
9.7. <code>clock v0.1.2</code> : установка времени.....	392
9.7.1. Общее поведение.....	392
9.7.2. Установка времени для операционных систем, использующих <code>libc</code>	392
9.7.3. Установка времени в MS Windows.....	395
9.7.4. <code>clock v0.1.2</code> : листинг полного кода.....	397
9.8. Более совершенные способы обработки ошибок.....	401
9.9. <code>clock v0.1.3</code> : вычисление разницы показания часов с показанием протокола сетевого времени — Network Time Protocol (NTP).....	402
9.9.1. Отправка NTP-запросов и интерпретация ответов.....	402
9.9.2. Корректировка местного времени по ответу сервера.....	405

8.5. Способы обработки ошибок, наиболее удобные для помещения в библиотеки.....	347
8.5.1. Проблема: невозможность возвращения нескольких типов ошибок.....	347
8.5.2. Заключение в оболочку нижестоящих ошибок путем определения нашего собственного типа ошибки.....	351
8.5.3. Фокусы с <code>unwrap()</code> и <code>expect()</code>	358
8.6. MAC-адреса.....	358
8.6.1. Создание MAC-адресов.....	360
8.7. Реализация конечных автоматов с помощью перечислений.....	362
8.8. Чистый TCP.....	363
8.9. Создание виртуального сетевого устройства.....	363
8.10. «Чистый» HTTP.....	365
Резюме.....	376
Глава 9. Время и хронометраж.....	377
9.1. Предыстория вопроса.....	378
9.2. Источники времени.....	380
9.3. Определения.....	380
9.4. Кодирование времени.....	382
9.4.1. Представление часовых поясов.....	383
9.5. <code>clock v0.1.0</code> : учим приложение сообщать о времени.....	383
9.6. <code>clock v0.1.1</code> : форматирование меток времени в соответствии с ISO 8601 и стандартами электронной почты.....	384
9.6.1. Реструктуризация кода <code>clock v0.1.0</code> с целью более широкой архитектурной поддержки.....	385
9.6.2. Форматирование времени.....	386
9.6.3. Предоставление полноценного интерфейса командной строки.....	387
9.6.4. <code>clock v0.1.1</code> : полный проект.....	388
9.7. <code>clock v0.1.2</code> : установка времени.....	392
9.7.1. Общее поведение.....	392
9.7.2. Установка времени для операционных систем, использующих <code>libc</code>	392
9.7.3. Установка времени в MS Windows.....	395
9.7.4. <code>clock v0.1.2</code> : листинг полного кода.....	397
9.8. Более совершенные способы обработки ошибок.....	401
9.9. <code>clock v0.1.3</code> : вычисление разницы показания часов с показанием протокола сетевого времени — Network Time Protocol (NTP).....	402
9.9.1. Отправка NTP-запросов и интерпретация ответов.....	402
9.9.2. Корректировка местного времени по ответу сервера.....	405

9.9.3. Преобразования между представлениями о времени, использующими различные степени точности и эпохи	407
9.9.4. clock v0.1.3: листинг полной версии кода	409
Резюме.....	417
Глава 10. Процессы, потоки и контейнеры.....	419
10.1. Безымянные функции	420
10.2. Порождение потоков	421
10.2.1. Введение в замыкания	421
10.2.2. Порождение потока	422
10.2.3. Эффект от порождения нескольких потоков.....	423
10.2.4. Эффект от порождения множества потоков.....	424
10.2.5. Воспроизведение результатов	426
10.2.6. Совместно используемые переменные	430
10.3. Отличие замыканий от функций.....	433
10.4. Аватары, процедурно генерируемые из многопоточного парсера и генератора кода	434
10.4.1. Как запустить проект render-hex, и как выглядит его предполагаемый вывод	434
10.4.2. Обзор однопоточной версии render-hex.....	435
10.4.3. Порождение потока для каждой логической задачи	446
10.4.4. Использование пула потоков и очереди задач	449
10.5. Конкурентные вычисления и виртуализация задач	457
10.5.1. Потоки.....	460
10.5.2. Что такое контекстное переключение?	460
10.5.3. Процессы	460
10.5.4. WebAssembly	461
10.5.5. Контейнеры	461
10.5.6. А зачем вообще использовать операционную систему?	461
Резюме.....	462
Глава 11. Ядро операционной системы	463
11.1. Оперяющаяся операционная система (FledgeOS).....	463
11.1.1. Настройка среды разработки под создание ядра операционной системы	463
11.1.2. Проверка среды разработки	465
11.2. FledgeOS-0: получение хоть чего-то работоспособного	466
11.2.1. Первая загрузка	466
11.2.2. Инструкции по компиляции.....	468

11.2.3. Листинги исходного кода.....	469
11.2.4. Способы справиться с паникой	474
11.2.5. Вывод информации на экран с использованием VGA-совместимого текстового режима	475
11.2.6 <code>_start ()</code> : функция <code>main ()</code> для FledgeOS.....	477
11.3. <code>fledgeos-1</code> : избавление от цикла занятости	477
11.3.1. Экономия ресурсов за счет прямого взаимодействия с центральным процессором.....	477
11.3.2. Исходный код <code>fledgeos-1</code>	478
11.4. <code>fledgeos-2</code> : самостоятельная обработка исключений.....	479
11.4.1. Почти что правильная обработка исключений	479
11.4.2. Исходный код <code>fledgeos-2</code>	480
11.5. <code>fledgeos-3</code> : текстовый вывод.....	481
11.5.1. Вывод на экран цветного текста.....	482
11.5.2. Управление представлением перечислений в памяти	482
11.5.3. Зачем использовать перечисления?.....	483
11.5.4. Создание шрифта для вывода информации в буфер кадра VGA	483
11.5.5. Вывод на экран.....	484
11.5.6. Исходный код <code>fledgeos-3</code>	485
11.6. <code>fledgeos-4</code> : специализированная обработка паники	487
11.6.1. Реализация обработчика паники, сообщающего пользователю об ошибке.	487
11.6.2. Повторная реализация <code>panic()</code> с использованием <code>core::fmt::Write</code>	487
11.6.3. Реализация <code>core::fmt::Write</code>	488
11.6.4. Исходный код <code>fledge-4</code>	489
Резюме.....	491
Глава 12. Сигналы, прерывания и исключения.....	493
12.1. Глоссарий.....	493
12.1.1. Сравнение сигналов и прерываний	495
12.2. Влияние прерываний на приложения.....	496
12.3. Программные прерывания	498
12.4. Аппаратные прерывания	499
12.5. Обработка сигналов	499
12.5.1. Поведение по умолчанию	499
12.5.2. Приостановка и возобновление работы программы.....	500
12.5.3. Перечень всех сигналов, поддерживаемых операционной системой.....	503

12.6. Обработка сигналов с помощью настраиваемых действий	504
12.6.1. Применение в Rust глобальных переменных	505
12.6.2. Использование глобальной переменной для указания на иницирование завершения выполнения программы	507
12.7. Отправка сигналов, определяемых в приложении.....	510
12.7.1. Общие сведения об указателях на функции и их синтаксисе.....	511
12.8. Игнорирование сигналов.....	512
12.9. Завершение работы из глубокой вложенности в стеке вызовов.....	514
12.9.1. Представление проекта sjlj.....	516
12.9.2. Настройка встроенных функций для их использования в программе	517
12.9.3. Приведение указателя к другому типу	519
12.9.4. Компиляция кода проекта sjlj	520
12.9.5. Исходный код проекта sjlj.....	521
12.10. Заметка о применении этих методов на платформах, не использующих сигналы.	524
12.11. Пересмотр исключений	524
Резюме.....	525