SECOND EDITION

Programming Rust
Fast, Safe Systems Development

Jim Blandy, Jason Orendorff, and Leonora F. S. Tindall
# Table of Contents

**Preface.** .................................................................................................................... xv

1. **Systems Programmers Can Have Nice Things.** ............................................... 1  
   Rust Shoulders the Load for You ............................................................... 2  
   Parallel Programming Is Tamed ............................................................ 3  
   And Yet Rust Is Still Fast ...................................................................... 4  
   Rust Makes Collaboration Easier .......................................................... 4  

2. **A Tour of Rust.** ..................................................................................................... 5  
   rustup and Cargo .................................................................................. 6  
   Rust Functions ...................................................................................... 8  
   Writing and Running Unit Tests ......................................................... 10  
   Handling Command-Line Arguments ................................................ 11  
   Serving Pages to the Web ................................................................... 15  
   Concurrency .......................................................................................... 22  
      What the Mandelbrot Set Actually Is ............................................. 23  
      Parsing Pair Command-Line Arguments ................................... 28  
      Mapping from Pixels to Complex Numbers ................................... 30  
      Plotting the Set ............................................................................... 32  
      Writing Image Files ....................................................................... 33  
      A Concurrent Mandelbrot Program ............................................... 35  
      Running the Mandelbrot Plotter ..................................................... 40  
      Safety Is Invisible .......................................................................... 41  
   Filesystems and Command-Line Tools ............................................. 42  
      The Command-Line Interface ....................................................... 43  
      Reading and Writing Files ............................................................ 45  
      Find and Replace ........................................................................... 46
8. Crates and Modules

- Crates
  - Editions
  - Build Profiles
- Modules
  - Nested Modules
  - Modules in Separate Files
  - Paths and Imports
  - The Standard Prelude
  - Making use Declarations pub
  - Making Struct Fields pub
  - Statics and Constants
- Turning a Program into a Library
- The src/bin Directory
- Attributes
- Tests and Documentation
  - Integration Tests
  - Documentation
  - Doc-Tests
- Specifying Dependencies
  - Versions
  - Cargo.lock
- Publishing Crates to crates.io
- Workspaces
- More Nice Things

9. Structs

- Named-Field Structs
- Tuple-Like Structs
- Unit-Like Structs
- Struct Layout
- Defining Methods with impl
  - Passing Self as a Box, Rc, or Arc
  - Type-Associated Functions
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10. Enums and Patterns</td>
<td>231</td>
</tr>
<tr>
<td>Enums</td>
<td>232</td>
</tr>
<tr>
<td>Enums with Data</td>
<td>234</td>
</tr>
<tr>
<td>Enums in Memory</td>
<td>235</td>
</tr>
<tr>
<td>Rich Data Structures Using Enums</td>
<td>236</td>
</tr>
<tr>
<td>Generic Enums</td>
<td>238</td>
</tr>
<tr>
<td>Patterns</td>
<td>241</td>
</tr>
<tr>
<td>Literals, Variables, and Wildcards in Patterns</td>
<td>244</td>
</tr>
<tr>
<td>Tuple and Struct Patterns</td>
<td>245</td>
</tr>
<tr>
<td>Array and Slice Patterns</td>
<td>246</td>
</tr>
<tr>
<td>Reference Patterns</td>
<td>247</td>
</tr>
<tr>
<td>Match Guards</td>
<td>249</td>
</tr>
<tr>
<td>Matching Multiple Possibilities</td>
<td>250</td>
</tr>
<tr>
<td>Binding with @ Patterns</td>
<td>250</td>
</tr>
<tr>
<td>Where Patterns Are Allowed</td>
<td>251</td>
</tr>
<tr>
<td>Populating a Binary Tree</td>
<td>252</td>
</tr>
<tr>
<td>The Big Picture</td>
<td>254</td>
</tr>
<tr>
<td>11. Traits and Generics</td>
<td>255</td>
</tr>
<tr>
<td>Using Traits</td>
<td>257</td>
</tr>
<tr>
<td>Trait Objects</td>
<td>258</td>
</tr>
<tr>
<td>Generic Functions and Type Parameters</td>
<td>260</td>
</tr>
<tr>
<td>Which to Use</td>
<td>264</td>
</tr>
<tr>
<td>Defining and Implementing Traits</td>
<td>266</td>
</tr>
<tr>
<td>Default Methods</td>
<td>267</td>
</tr>
<tr>
<td>Traits and Other People's Types</td>
<td>268</td>
</tr>
<tr>
<td>Self in Traits</td>
<td>270</td>
</tr>
<tr>
<td>Subtraits</td>
<td>271</td>
</tr>
<tr>
<td>Type-Associated Functions</td>
<td>272</td>
</tr>
<tr>
<td>Fully Qualified Method Calls</td>
<td>274</td>
</tr>
<tr>
<td>Traits That Define Relationships Between Types</td>
<td>275</td>
</tr>
<tr>
<td>Associated Types (or How Iterators Work)</td>
<td>276</td>
</tr>
<tr>
<td>Generic Traits (or How Operator Overloading Works)</td>
<td>279</td>
</tr>
<tr>
<td>impl Trait</td>
<td>280</td>
</tr>
<tr>
<td>Associated Consts</td>
<td>282</td>
</tr>
</tbody>
</table>
15. Iterators ................................................................. 347

The Iterator and Intolterator Traits 349
Creating Iterators 350
  iter and iter_mut Methods 350
Intolterator Implementations 351
  from_fn and successors 353
  drain Methods 355
Other Iterator Sources 356
Iterator Adapters 358
  map and filter 358
  filter_map and flat_map 361
  flatten 363
  take and take_while 365
  skip and skip_while 365
  peekable 366
  fuse 367
Reversible Iterators and rev 368
  inspect 369
  chain 370
  enumerate 371
  zip 372
  by_ref 372
  cloned, copied 373
  cycle 374
Consuming Iterators 375
  Simple Accumulation: count, sum, product 375
  max, min 376
  max_by, min_by 376
  max_by_key, min_by_key 377
Comparing Item Sequences 377
  any and all 378
  position, rposition, and ExactSizeIterator 378
  fold and rfold 379
  try_fold and try_rfold 380
Building Collections: collect and FromIterator 384
The Extend Trait 385
partition 386
for_each and try_for_each 387
Implementing Your Own Iterators 388
16. Collections................................................................. 393
  Overview ........................................................................ 394
  Vec<T> ........................................................................... 395
    Accessing Elements .................................................. 396
    Iteration ..................................................................... 398
    Growing and Shrinking Vectors ............................... 398
    Joining ....................................................................... 402
    Splitting ..................................................................... 402
    Swapping ..................................................................... 405
    Filling .......................................................................... 406
    Sorting and Searching .............................................. 406
    Comparing Slices ....................................................... 408
    Random Elements ...................................................... 408
    Rust Rules Out Invalidation Errors ......................... 409
  VecDeque<T> .................................................................. 410
  BinaryHeap<T> .............................................................. 412
  HashMap<K, V> and BTreeMap<K, V> ......................... 413
    Entries ...................................................................... 417
    Map Iteration ............................................................ 420
  HashSet<T> and BTreeSet<T> .......................................... 420
    Set Iteration ............................................................. 422
    When Equal Values Are Different ........................... 422
    Whole-Set Operations ................................................. 423
  Hashing ........................................................................ 424
  Using a Custom Hashing Algorithm ............................ 425
  Beyond the Standard Collections ............................... 427

17. Strings and Text.............................................................. 429
  Some Unicode Background ............................................ 430
    ASCII, Latin-1, and Unicode .................................. 430
    UTF-8 ......................................................................... 430
    Text Directionality .................................................... 432
  Characters (char) ........................................................ 432
    Classifying Characters ............................................. 433
    Handling Digits ........................................................ 434
    Case Conversion for Characters ............................. 435
    Conversions to and from Integers ......................... 436
  String and str .............................................................. 436
    Creating String Values ............................................ 437
    Simple Inspection ..................................................... 438
   Appending and Inserting Text .................................. 439
    Removing and Replacing Text ................................. 440

x | Table of Contents
Conventions for Searching and Iterating 441
Patterns for Searching Text 442
Searching and Replacing 443
Iterating over Text 444
Trimming 446
Case Conversion for Strings 447
Parsing Other Types from Strings 447
Converting Other Types to Strings 448
Borrowing as Other Text-Like Types 449
Accessing Text as UTF-8 449
Producing Text from UTF-8 Data 450
Putting Off Allocation 451
Strings as Generic Collections 453
Formatting Values 454
Formatting Text Values 455
Formatting Numbers 457
Formatting Other Types 458
Formatting Values for Debugging 459
Formatting Pointers for Debugging 460
Referring to Arguments by Index or Name 461
Dynamic Widths and Precisions 461
Formatting Your Own Types 462
Using the Formatting Language in Your Own Code 464
Regular Expressions 466
Basic Regex Use 466
Building Regex Values Lazily 467
Normalization 468
Normalization Forms 469
The unicode-normalization Crate 471

18. Input and Output ................................................................. 473
Readers and Writers 474
Readers 475
Buffered Readers 477
Reading Lines 479
Collecting Lines 481
Writers 482
Files 483
Seeking 484
Other Reader and Writer Types 485
Binary Data, Compression, and Serialization 487
Files and Directories 488
19. Concurrency .............................................................................................................. 499
    Fork-Join Parallelism
    spawn and join
    Error Handling Across Threads
    Sharing Immutable Data Across Threads
    Rayon
    Revisiting the Mandelbrot Set
    Channels
    Sending Values
    Receiving Values
    Running the Pipeline
    Channel Features and Performance
    Thread Safety: Send and Sync
    Piping Almost Any Iterator to a Channel
    Beyond Pipelines
    Shared Mutable State
      What Is a Mutex?
      Mutex<T>
      mut and Mutex
      Why Mutexes Are Not Always a Good Idea
      Deadlock
      Poisoned Mutexes
      Multiconsumer Channels Using Mutexes
      Read/Write Locks (RwLock<T>)
      Condition Variables (Condvar)
      Atomics
      Global Variables
    What Hacking Concurrent Code in Rust Is Like

20. Asynchronous Programming ................................................................................ 541
    From Synchronous to Asynchronous
    Futures
    Async Functions and Await Expressions
    Calling Async Functions from Synchronous Code: block_on
    Spawning Async Tasks
Async Blocks
Building Async Functions from Async Blocks
Spawning Async Tasks on a Thread Pool
But Does Your Future Implement Send?
Long Running Computations: yield_now and spawn_blocking
Comparing Asynchronous Designs
A Real Asynchronous HTTP Client
An Asynchronous Client and Server
Error and Result Types
The Protocol
Taking User Input: Asynchronous Streams
Sending Packets
Receiving Packets: More Asynchronous Streams
The Client's Main Function
The Server's Main Function
Handling Chat Connections: Async Mutexes
The Group Table: Synchronous Mutexes
Chat Groups: tokio's Broadcast Channels
Primitive Futures and Executors: When Is a Future Worth Polling Again?
Invoking Wakers: spawn_blocking
Implementing block_on
Pinning
The Two Life Stages of a Future
Pinned Pointers
The Unpin Trait
When Is Asynchronous Code Helpful?

21. Macros
Macro Basics
   Basics of Macro Expansion
   Unintended Consequences
   Repetition
Built-In Macros
Debugging Macros
Building the json! Macro
   Fragment Types
   Recursion in Macros
   Using Traits with Macros
   Scoping and Hygiene
   Importing and Exporting Macros
Avoiding Syntax Errors During Matching
Beyond macro_rules!
22. Unsafe Code......................................................... 627
   Unsafe from What? 628
   Unsafe Blocks 630
   Example: An Efficient ASCII String Type 631
   Unsafe Functions 633
   Unsafe Block or Unsafe Function? 635
   Undefined Behavior 636
   Unsafe Traits 638
   Raw Pointers 640
      Dereferencing Raw Pointers Safely 643
      Example: RefWithFlag 644
      Nullable Pointers 646
      Type Sizes and Alignments 647
      Pointer Arithmetic 647
      Moving into and out of Memory 649
      Example: GapBuffer 653
      Panic Safety in Unsafe Code 659
   Reinterpreting Memory with Unions 660
   Matching Unions 663
   Borrowing Unions 663

23. Foreign Functions.............................................. 665
   Finding Common Data Representations 666
   Declaring Foreign Functions and Variables 670
   Using Functions from Libraries 671
   A Raw Interface to libgit2 675
   A Safe Interface to libgit2 681
   Conclusion 692

Index................................................................. 693