Symbols

! (exclamation mark), 102
* (asterisk), 236–237
? (question mark operator), 62–65

A
abstract syntax tree (AST), 104–105
acquire memory ordering, 181–182
actors, 174–175
AddressSanitizer, 165
alignment, 20–21, 150
alloc library, 212
allocations, 202–204, 213–215
API Guidelines, 38
APIs, 89
architecture, 79
Arc::make_mut method, 232
Arc type, 39, 146
array, 23
AsRef trait, 40–41
assembly code, 218–219
assert_eq, 39
assertions, 165–166
asterisk (*), 236–237
async/await, 124–126
asynchronous programming
asynchronous interfaces, 120
blocking, 118
futures, 121–133
multithreading, 119–120
spawn, 138–139
standardized polling, 121
synchronous interfaces, 118–119
wakers, 133–134
asynchrony, 175–176
atomic types, 177–178
attributes
fundamental, 30
global_allocator, 214, 216–217

lang_start, 216
macros, 110, 112
must_use, 49
no_main, 216
no_mangle, 196, 205
no_std, 212–214
panic_handler, 216
repr, 21
should_panic, 92
auditing, 80
autotraits, 33, 54–55
Awesome Rust Streaming, 239

B
benchmarking, 98–100
binary library artifacts, 199–200
bindgen library, 207–209
bitflags library, 201
black_box function, 99
blanket implementations, 30, 40
blocking, 118, 137
Booleans, 48–49
Borrow trait, 41
borrow checker, 13–14
bounds checks, 148
Box type, 8, 157
Box<dyn Error> type, 60–61
BufReader and BufWriter types, 231
byte-aligned values, 20
bytes library, 225

C
C++, 209
callbacks, 204
caller-managed memory, 203–204
calling conventions, 198–199
The Cargo Book, 38, 241
cargo clippy, 38
cargo-deny, 224
Index

fetch_* methods on Atomic* types, 187–188
flume library, 226
foreign function interfaces (FFI), 147, 193–209
Formatter::debug_* method, 232–233
fragment types, 106
frame pointers, 230
frames, stack, 5–6
freeing, 6
From trait, 63
function-like macros, 110, 112
functions
  black_box, 99
drop_in_place, 149
iter::once, 230–231
ManuallyDrop::drop, 149
mem::replace, 231
read_unaligned, 150
read_volatile, 150
spawn, 138–139
thread::spawn_unchecked, 150
unreachable_unchecked, 149–150
Waker::from_raw, 149
write_unaligned, 150
write_volatile, 150
fundamental types, 30
#[fundamental] attribute, 30
fused futures, 121
futures, 121–135
fuzzing, 93–94

G
generators, 124–126
generic arguments, 43–44
generic lifetimes, 14–15
generic traits, 28, 104
glob imports, 236–237
#[global_allocator] attribute, 214, 216–217
GlobalAlloc trait, 151–152
grammar, 104–105
Guide to rustc Development, 241

H
Hash trait, 39
HashMap type, 89
hash tables, 148
hdrhistogram library, 226
heap, 6, 131
heapse library, 226
Heisenbugs, 191–192
hidden items, 55
higher-ranked trait bounds, 32
host platform, 221
hygiene, 107–109

I
immutable references, 9–10
implementation-managed memory, 203
indeferences, 234
index pointers, 233–234
indexmap library, 234
inherent methods, 41
Inside Rust blog, 238
Instant::elapsed method, 233
interior mutability, 12
Into trait, 63
intra-workspace dependencies, 72
invariance, 17
iterators, 40
iter::once function, 230–231
itertools library, 226

J
Jung, Ralf, 155

L
#[lang_start] attribute, 216
Law of Least Astonishment, 38
layout, 21–23
leaf futures, 134–135
leaking memory, 6
Learn Rust with Entirely Too Many Linked Lists, 242
Levick, Ryan, 239
libraries
  alloc, 212
  bindgen, 207–209
  bitflags, 201
  bytes, 225
cbindgen, 208
core, 212
criterion, 225
cxx, 225
cxx, 225
flume, 226
libraries (continued)
  hdrhistogram, 226
  heapless, 226
  indexmap, 234
  itertools, 226
  Loom, 97, 190
  metered, 113
  nix, 226
  petgraph, 234
  pin-project, 113, 226
  ring, 226–227
  serde, 40
  slab, 227
  standard, 212–213, 230–233
  static_assertions, 227
  structopt, 227
  thiserror, 227
  tower, 227
  tracing, 113, 227–228
  lifetimes, 12–17
  linear scalability, 169
  linking, 195–198
  link-time optimization (LTO), 75–76
  linting, 92–93
  The Little Book of Rust Books, 242
  The Little Book of Rust Macros, 241
  lock oversharing, 171
  lock-free algorithms, 173
  Loom library, 97, 190

M
  macros, 101–115, 230–231
    attribute, 110, 112
    cfg!, 70
    compile_fail!, 92
    declarative, 102–109
    derive, 110, 111
    function-like, 110, 112
    panic!, 76–77
    procedural, 109–115
    write!, 230
  ManuallyDrop type, 47
  ManuallyDrop::drop function, 149
  marker traits, 33
  marker types, 34
  matchers, 106–107
  MaybeUninit<T> type, 157–158
  MaybeUninit::assume_init method, 148–149
  McNamara, Tim, 239
  McSherry, Frank, 172
  memory
    caller-managed memory, 203–204
    heap, 6
    implementation-managed
      memory, 203
    memory mapping, 217
    operations, 177–178
    out-of-memory handler, 216–217
    stack, 5–6
    static memory, 6–7
    terminology, 2–3
    types in, 19–24
    unsafety, 153
  memory ordering, 178–184
    acquire, 181–182
    relaxed, 180–181
    release, 181–182
    sequentially consistent, 182–184
  mem::replace function, 231
  metadata, 73
  metavariables, 107
  metered library, 113
  methods, 232–233
    Arc::make_mut, 232
    Clone::clone_from, 232
    compare_exchange, 184–187
    compare_exchange_weak, 187
    Error::source, 58
    fetch_* on Atomic* types 187–188
    Formatter::debug_*, 232–233
    inherent, 41
    Instant::elapsed, 233
    MaybeUninit::assume_init, 148–149
    Option::as_deref, 233
    Option::transpose, 233
    Ord::clamp, 233
    Result::transpose, 233
    Vec::swap_remove, 233
  mid-level intermediate representation
    (MIR), 96
  minimum supported Rust version
    (MSRV), 81–83
  Miri, 79, 96–97, 165
  misaligned accesses, 20
  mocks, 88
  monomorphization, 25
multithreading, 119–120  
# [must_use] attribute, 49  
mutable references, 10–11  
mutual exclusion, 170, 185–186

N
naming practices, 38  
naturally aligned values, 20  
niche optimization, 145, 156, 202  
nix library, 226  
#! [no_main] attribute, 216  
#! [no_mangle] attribute, 196, 205  
#! [no_std] attribute, 212–214  
nonblocking interfaces, 120  
Not Yet Awesome list, 241

O
object files, 195  
object-safe traits, 27, 44–45  
Once type, 232  
opaque errors, 59–61  
opaque pointers, 206  
Oppermann, Philipp, 242  
optional dependencies, 69  
Option::as_deref method, 233  
Option::transpose method, 233  
opt-level, 75  
Ord trait, 39  
Ord::clamp method, 233  
Ordering type, 178–184  
orphan rule, 28–29  
out-of-memory handler, 216–217  
ownership, 7–8, 45–46

P
packages vs. crates, 74  
padding, 22  
panic handler, 215–216  
#[panic_handler] attribute, 216  
panic! macro, 76–77  
panics, 158–159, 204  
parallelism, 119–120, 175–176  
PartialEq trait, 39  
PartialOrd trait, 39  
patches, 73–74  
perfect scalability, 169  
performance  
  concurrency and, 169–171  
options, 75  
testing, 97–100  
petgraph library, 234  
Pin type, 127–133  
pin-project library, 113, 226  
place, 2–3  
platforms, 221–222  
pointers  
  frame pointers, 230  
  index pointers, 233–234  
  opaque pointers, 206  
  overview, 2–3  
  pointer casting, 147  
  pointer types, 128, 145  
  raw pointers, 144–147  
  undefined behavior, 144–147  
  void pointers, 206  
  wide pointers, 24  
poll contracts, 134  
polling futures  
  overview, 120  
  poll contracts, 134  
  standardized polling, 121  
preludes, 213, 236–237  
primitive types, 156  
Principle of Least Surprise, 38  
privacy boundary, 163–164  
procedural macros, 109–115  
profiles, 75–79  
program initialization, 216  
projects  
  configuration, 73–77  
  structure, 67–84  
  suggestions for, 240–241  
promises, 121  
property-based testing, 94–95  
ptr module, 150

Q
question mark operator (?), 62–65

R
race conditions, 169  
raw pointers, 144–147  
raw references, 156  
reactors, 135  
read_unaligned function, 150  
read_volatile function, 150
re-exports, 53–54
references, 9–12
reference types, 155–156
relaxed memory ordering, 180–181
release memory ordering, 181–182
#[repr] attribute, 21
representation, 2–3
repr(transparent), 21
Result type, 61–62
Result::transpose method, 233
ring library, 226–227
rlibs, 199
runtime, 215–217
Rust in Action, 239
Rust API Guidelines, 38, 241
Rust blog, 237
Rust Cookbook, 242
Rust Fuzz Book, 242
Rust Language Cheat Sheet, 242
The Rust Performance Book, 241–242
The Rust Programming Language
(Klabnik and Nichols), 14–15
Rust Quiz, 242
Rust Reference, 241
Rust RFC documents
RFC 1105, 38, 80, 237
RFC 2582, 156
RFC 2585, 143
RFC 2945, 198
Rust Unsafe Code Guidelines Reference, 241
rustc, 229–230
Rustonomicon, 165, 241
Rustup, 228
S
safety, 129, 148, 204–207
sanitizers, 165, 190–191
scalability, 169, 172
“Scalability! But at what COST?”, 172
sealed traits, 52–53
selects, 137
self-referential data structures, 127
semantic typing, 48–49
semantic versioning, 80
semver trick, 54
Send trait, 39, 151–153, 205–206
sequentially consistent memory
ordering, 182–184
serde library, 40
Serialize trait, 40
shadowing, 4
shared memory concurrency,
172–173
shared references, 9–10
#[should_panic] attribute, 92
Sized trait, 23–24
slab library, 227
spans, 114–115
spawn function, 138–139
specialization, 104
stack frames, 5–6
stack, pinning to, 131–132
standard library, 212–213, 230–233
Standard Library Developers Guide, 241
static dispatch, 25
‘static lifetime, 6–7
static linking, 195–198
static memory, 6–7
static_assertions library, 227
stores to memory, 177
stress tests, 189
structopt library, 227
subexecutors, 136–137
success ordering, 186
symbols, 194–198
Sync trait, 39, 151, 205–206
synchronous interfaces, 118–119
syntax trees, 105
Systems with JT, 239
T
tagged unions, 202
target platform, 221
target triples, 221
tasks, 136–137
testing
  concurrency, 189–191
doctests, 90–92
performance, 97–100
property-based testing, 94–95
stress tests, 189
test generation techniques,
93–94, 112
test harness, 86–87
test-only APIs, 89
textual scoping, 109
This Week in Rust blog, 238
thiserror library, 227
ThreadSanitizer, 190–191
thread::spawn_unchecked function, 150
tokens, 104
TokenStream, 113–114
token trees, 105
Tokio, 235, 242
Tolnay, David, 54, 241, 242
tower library, 227
tracing library, 113, 227–228
trait bounds, 31–33
trait objects, 27
traits
   AsRef, 40–41
   autotraits, 33, 54–55
   Borrow, 41
   Clone, 27
   coherence, 28–31
   common, 39–40
   compilation, 24–28
   Copy, 7–8, 40
   Debug, 39, 59
   Deref, 40–41, 153
   derived, 32, 52–53
   Deserialize, 40
   dispatch, 24–28
   Display, 59
   Drop, 44–46
   Eq, 39
   Error, 58
   From, 63
   generic, 28, 104
   GlobalAlloc, 151–152
   Hash, 39
   higher-ranked trait bounds, 32
   implementations, 51–53
   Into, 63
   marker, 33
   object-safe, 27, 44–45
   Ord, 39
   orphan rule, 28–31
   PartialEq, 39
   PartialOrd, 39
   sealed, 52–53
   Send, 39, 151–153, 205–206
   Serialize, 40
   Sized, 23–24
   Sync, 39, 151, 205–206
   trait bounds, 31–33
   trait objects, 27
   Unpin, 131, 152–153
   transcribers, 107
   TSan, 190–191
tuple, 23
   Turner, Jonathan, 239
type-erased errors, 59–61
type inference, 34
   type matching, 200–202
types, 231–232
   Arc, 39, 146
   atomic, 177–178
   Box, 8, 157
   Box<dyn Error>, 60–61
   BufReader and BufWriter, 231
   Cell, 12
   common traits for, 39–40
   complex, 23
   Cow, 231–232
dynamically sized types (DSTs), 23–24
   existential, 34–35
   fragment, 106
   fundamental, 30
   HashMap, 89
   inference, 34
   ManuallyDrop, 47
   marker, 34
   MaybeUninit<T>, 157–158
   in memory, 19–24
   Once, 232
   Ordering, 178–184
   Pin, 127–133
   pointer casting, 147
   pointer, 128, 145
   primitive, 156
   reference, 155–156
   Result, 61–62
   Sized, 23–24
   Vec, 157
   VecDeque, 232
   Weak, 146
   zero-sized, 49
type system, 48–50
U
undefined behavior, 143, 154–155
union, 23
Unpin trait, 131, 152–153
unreachable_unchecked function, 149–150
unreleased versions, 83–84
unsafe, 142–153
unsafe code, 129, 153–166, 241
unwinding, 76–77
unwinding panics, 158–159

V
validity, 155–158
values
dropping, 8–9
overview, 2–3
ownership of, 7–8
variables
high-level models, 3–4
low-level models, 4–5
overview, 2–3
variance, 15–16
contravariance, 16
covariance, 16
invariance, 17
Vec type, 157
VecDeque type, 232
Vec::swap_remove method, 233
versioning, 80–84
virtual method tables (vtables), 26
void pointers, 206

W
Waker::from_raw function, 149
wakers, 133–134
“We Need Better Language Specs” (Jung), 155
Weak type, 146
word size, 20
wide pointers, 24
worker pools, 173–174
working groups, 238
workspaces, 70–72
work stealing, 173
wrapper types, 40–41
write! macro, 230
write_unaligned function, 150
write_volatile function, 150
Writing an OS in Rust, 242

Y
yielding, 125

Z
zero-sized types, 49
-Zminimal-versions flag, 82
-Zsanitizer flag, 191
-Zprint-type-sizes flag, 229
-Ztime-passes flag, 229
-Ztimings flag, 229