🦀 Comprehensive Rust

Martin Geisler

[illegible]

[illegible] Comprehensive Rust [illegible] 🦀 **11**

1 [illegible] 13
1.1 [illegible] 14
1.2 [illegible] 17
1.3 [illegible] 17

2 [illegible] cargo 19
2.1 [illegible] Rust 19
2.2 [illegible] 20
2.3 [illegible] Cargo [illegible] 21

I [illegible] 23

3 [illegible] 24

4 [illegible] 26
4.1 [illegible] Rust [illegible] 26
4.2 [illegible] Rust 27
4.3 Playground 28

5 [illegible] 29
5.1 [illegible] 29
5.2 [illegible] 30
5.3 [illegible] 30
5.4 [illegible] 32
5.5 [illegible] 32
5.6 [illegible] 33
5.6.1 [illegible] 33

6 [illegible] 35
6.1 [illegible] `if` 35
6.2 [illegible] 36
6.2.1 `for` 36
6.2.2 `loop` 36
6.3 `break` [illegible] `continue` 37
6.3.1 [illegible] 37
6.4 [illegible] 38

6.4.1 [illegible] [illegible] [illegible] 38
6.5 [illegible] 39
6.6 [illegible] 39
6.7 [illegible]: [illegible] Collatz 40
6.7.1 [illegible] 42

II [illegible] [illegible]: [illegible] [illegible] [illegible] 43

7 [illegible] [illegible] 44

8 [illegible] [illegible] [illegible] [illegible] [illegible] 45
8.1 [illegible] 45
8.2 [illegible] 46
8.3 [illegible] [illegible] 47
8.4 [illegible] [illegible] [illegible] [illegible] 47
8.5 [illegible]: [illegible] [illegible] [illegible] [illegible] 47
8.5.1 [illegible] 49

9 [illegible] 50
9.1 [illegible] [illegible] 50
9.2 [illegible] [illegible] 51
9.3 [illegible] 52
9.4 [illegible] 53
9.5 [illegible]: [illegible] 55
9.5.1 [illegible] 56

10 [illegible] [illegible] [illegible] [illegible] [illegible] 57
10.1 [illegible] [illegible] 57
10.2 [illegible] [illegible] 58
10.3 Enums 59
10.4 `const` 61
10.5 `static` 61
10.6 [illegible] [illegible] [illegible] 62
10.7 [illegible]: [illegible] [illegible] 62
10.7.1 [illegible] 64

III [illegible] [illegible]: [illegible] 66

11 [illegible] [illegible] [illegible] [illegible] [illegible] 67

12 [illegible] 68
12.1 [illegible] [illegible] 68
12.2 [illegible] 69
12.3 Enums 70
12.4 [illegible] [illegible] Let 70
12.5 [illegible]: [illegible] [illegible] 73
12.5.1 [illegible] 75

13 [illegible] [illegible] [illegible] 78
13.1 [illegible] 78

80 . . . . Traits 13.2
80 . . . . Traits □□□□□ □□□□□□ 13.2.1
81 . . . . Supertraits 13.2.2
81 . . . . □□□□□□□ □□□□□□□□□ 13.2.3
82 . . . . Deriving 13.3
82 . . . . Trait Logger :□□□□□□ 13.4
83 . . . . □□□□□□ 13.4.1

85 □□□ :□□□ □□□ IV

86 □□□ □□□ 14

87 Generics 15
87 . . . . Generic □□□□□□ 15.1
88 . . . . Generic □□□□□□□□ □□□□□ 15.2
89 . . . . Generic Traits 15.3
89 . . . . Trait Bounds 15.4
90 . . . . `impl Trait` 15.5
91 . . . . `dyn Trait` 15.6
92 . . . . Generic `min` :□□□□□□ 15.7
93 . . . . □□□□□□ 15.7.1

94 □□□□□□□ □□□□□□□□□□□ □□□□□□□□□ 16
94 . . . . □□□□□□□□□□□ □□□□□□□□□ 16.1
94 . . . . □□□□□□□□ 16.2
95 . . . . Option 16.3
96 . . . . Result 16.4
96 . . . . String 16.5
97 . . . . `Vec` 16.6
98 . . . . `HashMap` 16.7
99 . . . . □□□□□□□□□ :□□□□□□ 16.8
101 . . . . □□□□□□ 16.8.1

102 Traits □□□□□□□□□□ □□□□□□□□□ 17
102 . . . . □□□□□□□ 17.1
103 . . . . □□□□□□□□□□ 17.2
104 . . . . `From` and `Into` 17.3
105 . . . . Casting 17.4
105 . . . . `Read` and `Write` 17.5
106 . . . . The `Default` Trait 17.6
107 . . . . Closures 17.7
108 . . . . ROT13 :□□□□□□ 17.8
109 . . . . □□□□□□ 17.8.1

111 □□□ :□ □□□ V

112 □□□□□□ □□□ □ □□□ □□ 18

113 □□□□□□ □□□□□□□ 19
113 . . . . □□□□□□□ □□□□□□ □□□□□□ 19.1

19.2 [illegible] 114
19.3 [illegible] 115
19.4 [illegible] 116
19.5 Clone 119
19.6 [illegible] 120
19.7 [illegible] Drop 121
19.8 [illegible]: [illegible] 122
19.8.1 [illegible] 123

20 [illegible] 126
20.1 `Box<T>` 126
20.2 `Rc` 128
20.3 Owned Trait Objects 129
20.4 [illegible]: [illegible] 130
20.4.1 [illegible] 132

VI [illegible]: [illegible] 136

21 [illegible] 137

22 [illegible] (Borrowing) 138
22.1 [illegible] 138
22.2 [illegible] 139
22.3 [illegible] 140
22.4 [illegible] 140
22.5 [illegible]: [illegible] 142
22.5.1 [illegible] 143

23 [illegible] 145
23.1 [illegible] 145
23.2 [illegible] 146
23.3 Lifetimes in Data Structures 147
23.4 [illegible]: [illegible] Protobuf 148
23.4.1 [illegible] 151

VII [illegible]: [illegible] 157

24 Welcome to Day 4 158

25 Iterators 159
25.1 `Iterator` 159
25.2 `IntoIterator` 160
25.3 FromIterator 161
25.4 [illegible]: [illegible] Iterator Chaining 162
25.4.1 [illegible] 163

26 [illegible] 164
26.1 [illegible] 164
26.2 [illegible] 165
26.3 [illegible] 166

167 . . . . use, super, self 26.4
167 . . . . □□□□□□□□□ □□□□□□□□ □□□□□ □□□□□□□□□□ □□□□□ □□□□□□□□□□□□□ :□□□□□□ 26.5
170 . . . . □□□□□□ 26.5.1

174 □□□□□□□□□ **27**
174 . . . . (Unit Tests) □□□□□ □□□□□□□□ 27.1
175 . . . . □□□□□□ □□□□□ □□□□□□ 27.2
176 . . . . Clippy □ Lints □□□□□□□□□□ 27.3
176 . . . . Luhn □□□□□□□□□□ :□□□□□□ 27.4
177 . . . . □□□□□□ 27.4.1

180 **□□□ □□ □□□ :□□□□□□ □□□ VIII**

181 □□□ □□□ **28**

182 □□□ □□□□□□□ **29**
182 . . . . □□Panic □□□□ □□ 29.1
183 . . . . `Result` 29.2
184 . . . . Try □□□□□□□ 29.3
185 . . . . □□□□□ □□□□□□□□ □□ (Conversions) □□□□□□□□□ □□□□ 29.4
187 . . . . Dynamic □□□□□□□ □□□□□□ 29.5
187 . . . . `anyhow` □ `thiserror` 29.6
189 . . . . Result □□ □□□□□□□□□□□ :□□□□□□ 29.7
191 . . . . □□□□□□ 29.7.1

194 □□□□□□□ **Rust 30**
194 . . . . □□□□□□□ Rust 30.1
195 . . . . □□□ □□□□□□□□□□□□ □□ □□□□□□ □□□□ 30.2
196 . . . . □□□□□□ □□□□□ □□□□□ □□□□□□□□□□ 30.3
196 . . . . □□□□□□□□□ □□□□□ □□□□ 30.4
197 . . . . □□□□□□ □□□□□□ 30.5
198 . . . . □□□□□□ (Traits) □□□□□ □□□□□ □□□□□□ 30.6
199 . . . . FFI Wrapper □□□□□ □□□□ 30.7
201 . . . . □□□□□□ 30.7.1

205 **□□□□□□□□ IX**

206 □□□□□□□□□□□**Android** □□ **Rust** □□ **31**

207 □□□□□□ **32**

208 □□□□□ □□□□□□□ **33**
209 . . . . Rust Binaries 33.1
209 . . . . Rust □□□□□□□□□□□□ 33.2

211 **AIDL 34**
211 . . . . Birthday □□□□□□ □□□□□□ 34.1
211 . . . . AIDL Interfaces 34.1.1
212 . . . . Generated Service API 34.1.2
212 . . . . □□□□□□□□□ □□□□□□□□□□□□ 34.1.3

213 AIDL Server 34.1.4
214 □□□□□□□ 34.1.5
214 AIDL Client 34.1.6
216 API □□□□□ □□□□□□ 34.1.7
216 □□□□□□□□□ □ □□□□□□□□ □□□□□□□□□□□□□ 34.1.8
217 AIDL □□□□□□ □□ □□□□ 34.2
217 □□□□□□□ □□□□□□□ 34.2.1
217 □□□□□□□□□□□ □□□□□□□□□ 34.2.2
218 Sending Objects 34.2.3
219 □□□□□□□□□□□□□□ 34.2.4
220 □□□□□□□□ □□□□□□□ 34.2.5

222 Android □□ □□□□□□□□□ 35
223 GoogleTest 35.1
224 Mocking 35.2

226 □□□ 36

228 □□□□□□□□ □□□□□□□□ 37
228 C □□ □□□□□□□□ □□□□□□□□ 37.1
228 Bindgen □□ □□□□□□□□□ □□ 37.1.1
230 Rust □□□□□□□□□□□ 37.1.2
232 ++C □□ 37.2
232 □□ □□□□□□□ 37.2.1
232 Rust □□ □□ □□□□□□□ 37.2.2
233 ++Generated C 37.2.3
234 C++ Bridge Declarations 37.2.4
235 □□□□□□□ □□□□□□□ 37.2.5
235 Shared Enums 37.2.6
236 Rust □□□□ □□□□□□□□ 37.2.7
236 C++ □□□□ □□□□□□□□ 37.2.8
236 □□□□□□□ □□□□□□□□□ 37.2.9
237 □□□□□□□□□ □□ □□□□□□37.2.10
237 □□□□□□□□□ □□ □□□□□□37.2.11
238 □□□□□□□□□ □□ □□□□□□37.2.12
238 □□□□□ □□ □□□□□□□□ □□□□□□□□ 37.3

240 □□□□□□□□□ 38

241 X Chromium

242 □□□□□□□□□□□ Chromium □□ Rust □□ 39

243 □□□□□□ 40

245 Cargo □□□□□□□□□□□ □ Chromium □□□□□□□□ 41

248 Chromium Rust □□□□□□□□ 42

249 Build □□□□□□□□ 43
249 `unsafe` Rust □□ □□□□□ 43.1

250 . . . . Chromium C++ □□ Rust Code □□ □□□□ 43.2
250 . . . . Visual Studio Code 43.3
251 . . . . □□□□ □□□□□ □□□□□ 43.4

252 □□□□□□□□ 44
253 . . . . `rust_gtest_interop` Library 44.1
253 . . . . Rust □□□□□□□ □□□□ GN □□□□□ 44.2
253 . . . . `chromium::import!` Macro 44.3
254 . . . . □□□□ □□□□□ 44.4

255 C++ □□ □□□□□□□ □□□□□□□ 45
256 . . . . (□□Binding) □□□□□□□□ □□□□□ 45.1
257 . . . . CXX □□□ □□□□□□□ 45.2
257 . . . . QR □□□□ :CXX □□□ □□□□□□□ 45.2.1
258 . . . . PNG □□□□ :CXX □□□ □□□□□□□ 45.2.2
259 . . . . C++ □□ □□□□□□□ □□□□□□□ :□□□□□ 45.3

261 □□□□ □□□ □□□Crate □□□□□□□□□□□ 46
261 . . . . □□crate □□□□□□□ □□□□ `Cargo.toml` □□□□ □□□□□□□□□□ 46.1
262 . . . . `gnrt_config.toml` □□□□□□□□ 46.2
262 . . . . □□Crate □□□□ □□□□□□□ 46.3
262 . . . . gn Build □□□□□□ □□□□□ 46.4
263 . . . . □□□□□□□ □□ 46.5
263 . . . . □□□□□□□ □□□□□□ □□ □□ □□ □□□□□□□□□□□□□□ □□□□ 46.5.1
□□□□□□ □□□□□□□ □□□□□□□□ □□ □□□□□□Build □□ C++ □□ □□□□□□□□□□□□□□ □□□□ 46.5.2
264 . . . . □□□□□□□
264 . . . . Crate □□ □□ □□□□□□□ 46.6
264 . . . . □□□□ □□□ □□□Crate □□□□□□□□ 46.7
265 . . . . Chromium □□□□ □□ □□ □□Crate □□□□□ 46.8
265 . . . . □□Crate □□□□□ □□□ □□□ □□ 46.9
266 . . . . □□□□□□□□46.10

267 □□□□ □□□□□ --- □□ □□□□□□□□ □□□□ 47

269 □□□□□ □□□□□□□□□□ 48

270 □□□ :Bare Metal □□ XI

271 □□□□□□ □□□ Bare Metal Rust □□ 49

273 `no_std` 50
274 . . . . `no_std`□□ □□□□□□□□ □□□□□□□ □□ 50.1
274 . . . . `alloc` 50.2

276 □□□□□□□□□□□□□□□□□ 51
276 . . . . □□□ MMIO 51.1
278 . . . . □□□□□□ □□□□□□□ □□□Crate 51.2
279 . . . . HAL crates 51.3
279 . . . . Board support crates 51.4
280 . . . . state pattern □□□□ □□ 51.5
281 . . . . `embedded-hal` 51.6

281 . . . . `probe-rs` and `cargo-embed` 51.7
282 . . . . (Debugging) □□□□□ □□□□□□□ 51.7.1
282 . . . . □□□□□ □□□□□□□□□□ 51.8

284 □□□□□□□□ 52
284 . . . . □□□□□□□ 52.1
286 . . . . Bare Metal Rust □□□□□□□□ □□□□□□ 52.2

290 □□□ :Bare Metal □□ XII

291 Application processors 53
291 . . . . Rust □□□□□ □□□□ □□□□□□ 53.1
293 . . . . Inline assembly 53.2
294 . . . . MMIO □□□□□ □□□□□ □□□□□□ □□ □□□□□□□ 53.3
295 . . . . □□□□□□□□□UART □□□□□□□ □□ □□□□□□□ 53.4
296 . . . . □□□□□□ □□□□trait 53.4.1
296 . . . . □□□□□ UART □□□□□□□ □□ 53.5
297 . . . . (Bitflags) □□□□□ □□□□□□□□□ 53.5.1
297 . . . . □□□□□□□□□ □□□□□□□□ 53.5.2
298 . . . . □□□□□□□□ 53.5.3
299 . . . . □□ □□ □□□□□□□□□ □□ 53.5.4
300 . . . . □□□ 53.6
301 . . . . □□ □□ □□□□□□□□□ □□ 53.6.1
302 . . . . □□□□□□□□□□□ 53.7
303 . . . . □□□□□ □□□□□□□□□□□ 53.8

305 □□□□□□□□□ (crates) □□□□□□□□□ 54
305 . . . . `zerocopy` 54.1
306 . . . . `aarch64-paging` 54.2
306 . . . . `buddy_system_allocator` 54.3
307 . . . . `tinyvec` 54.4
307 . . . . `spin` 54.5

309 □□□□□□□□ 55
310 . . . . vmbase 55.1

311 □□□□□□□□ 56
311 . . . . RTC driver 56.1
329 . . . . Bare Metal Rust □□□□□□□□□□□□□□ 56.2

334 □□□ :□□□□□□□□□ XIII

335 □□□□□□□□□□ Rust □□ Concurrency □□□□□ □□ 57

336 □□□□□□ 58
336 . . . . □□□□□ □□□□□□□□ 58.1
337 . . . . □□□□□□ □□□□□□□□ 58.2

339 □□□□□□□□ 59
339 . . . . Receivers □ Senders 59.1

340 . . . . □□□□□□□ □□□□□□□□□ 59.2
340 . . . . □□□□□ □□□□□□□□□ 59.3

342 Sync □ Send 60
342 . . . . □□□□□□ □□□□□□□□□ 60.1
342 . . . . Send 60.2
343 . . . . Sync 60.3
343 . . . . □□□□□□ 60.4

345 □□□□□ □□□□□□□□□ 61
345 . . . . Arc 61.1
346 . . . . Mutex 61.2
346 . . . . □□□□ 61.3

348 □□□□□□□ 62
348 . . . . Dining □□□□□ 62.1
349 . . . . □□□□□ □□□ □□□□□ □□□□□□□□ 62.2
351 . . . . □□□□ □□□ 62.3

357 □□□ :□□□□□□□□ XIV

358 □□□□□ □□□ 63

359 Async □□□□□ 64
359 . . . . async/await 64.1
360 . . . . Futures 64.2
361 . . . . Runtimes 64.3
361 . . . . Tokio 64.3.1
362 . . . . Task 64.4

363 Control Flow □ □□□□□□□ 65
363 . . . . Async □□□□□□□□ 65.1
364 . . . . Join 65.2
365 . . . . Select 65.3

366 □□Pitfall 66
366 . . . . executor □□□□ □□□□□ 66.1
367 . . . . Pin 66.2
369 . . . . Async □□□□ 66.3
371 . . . . □□□ 66.4

374 □□□□□□□ 67
374 . . . . Dining Philosophers --- Async 67.1
375 . . . . □□ □□□□□□ □□□ 67.2
378 . . . . □□□□ □□□ 67.3

383 □□□ □□□□□ XV

384 !□□□□ 68

385 □□□□□□□□ 69

390 **Rust □□□□ □□□□□ □□□□ 70**

392 **□□□□□□□□ 71**

[illegible] Comprehensive Rust [illegible] 🦀

stars 28k contributors 303 build passing

[illegible] Rust [illegible]. [illegible] Rust [illegible], [illegible].

> [illegible] https://google.github.io/comprehensive-rust/ [illegible]. [illegible], [illegible].
>
> [illegible]. [illegible].
>
> [illegible] PDF [illegible].

[illegible] Rust [illegible]. [illegible] Rust [illegible]:

- [illegible] syntax [illegible] Rust [illegible].
- [illegible] Rust [illegible].
- [illegible] Rust [illegible].

[illegible] Rust [illegible].

[illegible]:

- Android: [illegible] Rust [illegible] (AOSP). [illegible] C [illegible] C++ [illegible] Java [illegible].
- Chromium: [illegible] Rust [illegible] Chromium. [illegible] C ++ [illegible] (crates) [illegible].
- Bare-metal: [illegible] Rust [illegible] bare-metal [illegible]). [illegible].
- [illegible]: [illegible] concurrency [illegible] Rust [illegible]. [illegible] concurrency [illegible] (preemptively [illegible] thread [illegible] mutex [illegible]) [illegible] async/await concurrency (multitasking [illegible] futures [illegible].

[illegible]

[illegible] Rust, [illegible]:

- [illegible] Macro: [illegible] Chapter 19.5 in the Rust Book [illegible] Rust by Example [illegible].

[illegible]

[illegible]. Rust [illegible] Rust [illegible] C [illegible]

C++

[illegible] Rust [illegible].

[illegible]-[illegible].

[illegible] *speaker note*. [illegible].. [illegible].

1 [illegible]

[illegible]

[illegible].

[illegible].

[illegible].

[illegible]:

1. [illegible] (!)[illegible] «[illegible]» [illegible].

2. [illegible].

3. [illegible] --- [illegible] live-coding [illegible].

4. [illegible] `mdbook serve` [illegible] ([illegible]) [illegible].

5. [illegible] ([illegible]

[illegible] (standard library) [illegible].

[illegible]!

[illegible]!

1.1 [illegible]

[illegible].

Rust [illegible]

[illegible] Rust [illegible]!

:[illegible]

- [illegible] ([illegible])

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

- [illegible] ([illegible])

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

- [illegible] ([illegible])

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

- [illegible] ([illegible])

[illegible]	[illegible]
Generics	[illegible]
[illegible]	[illegible]
[illegible] Traits	[illegible]

- [illegible] ([illegible])

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

- [illegible]([illegible])

[illegible]	[illegible]
[illegible] (Borrowing)	[illegible]
[illegible]	[illegible]

- [illegible] ([illegible])

[illegible]	[illegible]
[illegible]	[illegible]
Iterators	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

- [illegible] ([illegible])

[illegible]	[illegible]
[illegible]	[illegible]
Rust [illegible]	[illegible]

[illegible]

[illegible] 4 [illegible] Rust Fundamentals [illegible]:

Rust [illegible]

[illegible] Rust [illegible] [illegible] Rust [illegible]. [illegible] C[illegible]

C++

[illegible].

[illegible] ASOP [illegible] ,[illegible] [illegible] [illegible] /src/android [illegible] ASOP [illegible]. [illegible] build [illegible] Android.bp [illegible] `/src/android` [illegible].

[illegible] `adb sync` [illegible]- Android [illegible] `src/android/build_all.sh` [illegible]. [illegible].

Rust [illegible]

[illegible] Rust in Chromium [illegible] Rust [illegible] Chromium. [illegible] Rust [illegible] Chromium gn[illegible]- [illegible] (”crates”) [illegible] C++ [illegible].

[illegible] Chromium [illegible] --- [illegible] [[illegible]] (chromium/setup.md/..) [illegible]. [illegible] Chromium [illegible].

Rust [illegible]

[illegible] Rust [illegible] Rust[illegible] (embedded) [illegible]. [illegible].

[illegible] v2 BBCmicro:bit [illegible]. [illegible] welcome page [illegible].

Rust [illegible]

The Concurrency in Rust deep dive is a full day class on classical as well as `async/await` concurrency.

[illegible] crate [illegible]. [illegible] `src/main.rs` [illegible]/[illegible]:

```
cargo init concurrency
cd concurrency
cargo add tokio --features full
cargo run
```

[illegible]:

- [illegible] ([illegible])

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
Send [illegible] Sync	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

- [illegible] ([illegible])

[illegible]	[illegible]
Async [illegible]	[illegible]
Control Flow [illegible]	[illegible]
[illegible]Pitfall	[illegible]
[illegible]	[illegible]

[illegible]

[illegible] Rust [illegible]!

1.2 [illegible]

[illegible] mdBook [illegible]:

- Arrow-Left
  : [illegible].
- Arrow-Right
  : [illegible].
- Ctrl + Enter
  : Execute the code sample that has focus.
- s
  : [illegible].

1.3 [illegible]

[illegible]:

- [illegible] @joaovicmendes, @hugojacob, rastringer, [illegible] @henrif75.
- Chinese (Simplified) by @suetfei, @wnghl, @anlunx, @kongy, @noahdragon, @superwhd, @SketchK, and @nodmp.
- [illegible] @kuanhungchen, @mingyc, @victorhsieh, hueich, [illegible] @johnathan79717.
- Japanese by @CoinEZ-JPN, @momotaro1105, @HidenoriKobayashi and @kantasv.
- [illegible] @jiyongp, keispace, [illegible] @jooyunghan.
- [illegible] @deavid.
- Ukrainian by @git-user-cpp, @yaremam and @reta.

[illegible].

[illegible]

[illegible]:

- Arabic by @younies

- □□□□□□□ □□□□□ @raselmandol.
- Farsi by @Alix1383, @DannyRavi, @hamidrezakp, @javad-jafari and @moaminsharifi.
- □□□□□□□□□□ □□□□□ @KookaS □ @vcaen.
- □□□□□□□ □□□□□ @Throvn □ @ronaldfw.
- □□□□□□□□□□□ □□□□□ @Throvn □ @ronaldfw.

□□□□□□ □□□□□ □□□□□□□□□ □□ □□□□□□ □□□□□□□□□ □□□□ □□ □□□□□□ □□□□□□□□□□□□□ □□ □□□□□□□□□□□□ □□□□ □□ □□□□□□ □□□□□ □□□□□.

□□□□ □□□□□□□□□□□ □□ □□□□ □□□□ □□□□ □□□□□□□ □□□□□□□ □□ □□□□□□□□□□□□□□□□□ □□□ □□□□ □□□□□□□□ □□□□□□ □□□□ □□□□□□□□. □□□□□□ □□□ □□ issue tracker □□□□□□□□ □ □□□□□□ □□□ □□□□□.

2 [illegible]

cargo [illegible]

[illegible] Cargo [illegible] Rust [illegible]
.[illegible] Rust [illegible] Rust [illegible]
[illegible] Rust [illegible]
.[illegible]

[illegible]

.https://rustup.rs [illegible]

[illegible] (Rust (`rustc` [illegible] (Cargo (`cargo` [illegible]
[illegible] (CLI)[illegible] `rustup` [illegible] .[illegible]
.[illegible]

[illegible] .[illegible] Rust [illegible] IDE [illegible] Rust[illegible]
[illegible] rust-analyzer [illegible]
[illegible] VS Code[illegible] Emacs[illegible] Vim/Neovim [illegible]
.[illegible] RustRover [illegible] IDE [illegible] .[illegible]

- [illegible] `apt` [illegible] Rust formatter [illegible] Rust [illegible] Cargo [illegible]/[illegible] .[illegible] Rust [illegible] command .[illegible]:

  ```
  sudo apt install cargo rust-src rustfmt
  ```

- [illegible] Rust [illegible] Homebrew [illegible] macOS [illegible] .[illegible] Rust [illegible].

2.1 Rust [illegible]

:[illegible] Rust [illegible]

- [illegible] `.rs` [illegible] Rust [illegible] :`rustc` .[illegible]
- [illegible] Cargo .[illegible] build tool [illegible] Rust [illegible] :`cargo` [illegible] https://crates.io [illegible]

[illegible] `rustc` [illegible]. Cargo [illegible] unit test [illegible].

- 'rustup': [illegible] rustchain [illegible]. [illegible] "rustc" [illegible] "cargo" [illegible] Rust [illegible]. [illegible] "rustup" [illegible]. [illegible] Rust [illegible] "rustup" [illegible].

[illegible]:

- Rust [illegible]. [illegible] --- [illegible].
- [illegible]: "beta"، "stable" [illegible] "nightly".
- [illegible] "nightly" [illegible] "beta" [illegible] "stable" [illegible].
- [illegible] registries [illegible] git [illegible].
- Rust [illegible] editions [illegible]: [illegible] Rust 2021 [illegible]. [illegible] Rust 2015 [illegible] Rust 2018 [illegible].
  - [illegible] backwards incompatible [illegible].
  - [illegible] breaking code [illegible]: [illegible]- [illegible] crate [illegible] `Cargo.toml` [illegible].
  - [illegible] Rust [illegible].
  - [illegible] (rustc) [illegible] `cargo` [illegible] ([illegible]).
  - [illegible] Cargo [illegible]. [illegible]:
    - [illegible]/[illegible]
    - workspaces
    - [illegible] Dev [illegible] Runtime Management/Caching
    - build scripting
    - global installation
    - [illegible] command plugin [illegible] (cargo clippy [illegible]) [illegible].
  - [illegible] official Cargo Book [illegible].

2.2 [illegible]

[illegible] Rust [illegible].

[illegible] Cargo [illegible]: [illegible]. [illegible] Cargo [illegible].

[illegible] (interactive) [illegible]:

```rust
fn main() {
    println!("Edit me!");
}
```

You can use

Ctrl + Enter

to execute the code when focus is in the text box.

[illegible]. [illegible]:

- [illegible] embedded playground [illegible] unit tests [illegible]. [illegible] Playground [illegible] unit tests [illegible].
- [illegible] embedded playgrounds [illegible]! [illegible] local Rust installation [illegible] Playground [illegible].

2.3 [illegible] Cargo

[illegible] Rust [illegible]. [illegible] instructions in the Rust Book [illegible]. [illegible] `rustc` [illegible] `cargo` [illegible]. [illegible] Rust [illegible] version number [illegible]:

```
% rustc --version
rustc 1.69.0 (84c898d65 2023-04-16)
% cargo --version
cargo 1.69.0 (6e9a83356 2023-04-12)
```

[illegible] Rust [illegible].

[illegible] Rust:

1. [illegible] "[illegible]" [illegible].
2. [illegible] `cargo new exercise` [illegible] `/excerise` [illegible]:

   ```
   $ cargo new exercise
        Created binary (application) `exercise` package
   ```

3. [illegible] `/exercise` [illegible] `cargo run` [illegible]:

   ```
   $ cd exercise
   $ cargo run
      Compiling exercise v0.1.0 (/home/mgeisler/tmp/exercise)
       Finished dev [unoptimized + debuginfo] target(s) in 0.75s
        Running `target/debug/exercise`
   Hello, world!
   ```

4. [illegible] `src/main.rs` [illegible]. [illegible] `src/main.rs` [illegible].

```
} ()fn main
;("!println!("Edit me
{
```

5. [illegible] cargo run [illegible]:

```
cargo run $
(Compiling exercise v0.1.0 (/home/mgeisler/tmp/exercise
Finished dev [unoptimized + debuginfo] target(s) in 0.24s
`Running `target/debug/exercise
!Edit me
```

6. [illegible] cargo check [illegible] cargo build [illegible] /target/debug [illegible] /target/release [illegible] cargo build --release [illegible].

7. [illegible] Cargo.toml [illegible] cargo [illegible].

[illegible] Cargo [illegible].

I [illegible]

[illegible] :[illegible] [illegible]

[illegible] 3

[illegible]

[illegible] Rust [illegible]. [illegible]:

- [illegible]: [illegible], [illegible], structs, enums, [illegible], [illegible], [illegible].
- Types and type inference.
- [illegible]: [illegible].
- [illegible]: [illegible] enums.
- [illegible]: [illegible] structs, enums [illegible].

[illegible]

[illegible] 10 [illegible] 2 [illegible] 5 [illegible]. [illegible]:

[illegible]	[illegible]
[illegible]	[illegible]
[illegible], [illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

[illegible]:

- [illegible].
- [illegible]!
  - [illegible] Rust [illegible].
- [illegible].
  - [illegible]! [illegible]. [illegible].

[illegible] ”[illegible]” [illegible] Rust [illegible]. [illegible] Rust [illegible].

[illegible]. [illegible]. [illegible]. [illegible]. [illegible]!

4 [illegible]

[illegible], [illegible]

[illegible]. [illegible]:

[illegible]	[illegible]
[illegible] Rust [illegible]	[illegible]
Rust [illegible]	[illegible]
Playground	[illegible]

4.1 [illegible] Rust [illegible]

Rust [illegible] 1.0 [illegible] 2015 [illegible]:

- [illegible] Rust, [illegible]

  C++

  [illegible]

  - `rustc` [illegible] `LLVM` [illegible].
- [illegible]:
  - x86, ARM, WebAssembly, ...
  - Linux, Mac, Windows, ...
- [illegible] Rust [illegible]:
  - [illegible] (firmware) [illegible] (boot loaders)
  - [illegible],
  - [illegible],
  - [illegible],
  - [illegible].

Rust [illegible]

C++

[illegible]:

- [illegible].

- [illegible].
- [illegible].
- [illegible] (runtime) [illegible] (garbage collection) [illegible].
- [illegible].

4.2 [illegible] Rust

[illegible] Rust:

- [illegible] - [illegible]
  - [illegible] (uninitialized) [illegible].
  - [illegible].
  - [illegible].
  - [illegible] NULL [illegible].
  - [illegible].
  - [illegible] (data races) [illegible].
  - [illegible] (iterators) [illegible]..
- [illegible] - [illegible] Rust [illegible]
  - [illegible].
  - [illegible] ([illegible] wrap-around).
- [illegible] - [illegible]
  - Enum [illegible].
  - [illegible].
  - FFI [illegible].
  - [illegible].
  - [illegible].
  - [illegible]-[illegible].
  - [illegible]-[illegible].
  - [illegible] LSP.

[illegible].

[illegible] Rust [illegible]::

- [illegible] C [illegible]

  C++

  : [illegible] Rust [illegible] ([illegible] borrow [illegible] t. [illegible] C [illegible]

  C++

  [illegible].
- [illegible] JavaScript [illegible] Python [illegible] Go [illegible] Java...: [illegible] (memory safety) [illegible] C [illegible]

++C

([illegible] garbage collector) [illegible] ([illegible]) [illegible].

4.3 Playground

Rust Playground [illegible] Rust [illegible]. [illegible] "Hello-world" [illegible]. [illegible]:

- [illegible] "[illegible]" [illegible] "rustfmt" [illegible] "[illegible]" [illegible].
- Rust [illegible] "[illegible]" [illegible]: Debug ([illegible]) [illegible] Release ([illegible]). [illegible] «[illegible]» [illegible].
- [illegible] "ASM" [illegible] "..." [illegible].

[illegible] playground [illegible]. [illegible]. [illegible] Rust [illegible].

[illegible] 5

[illegible] [illegible] [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible]:

[illegible]	[illegible] [illegible]
[illegible], [illegible]	[illegible] [illegible]
[illegible]	[illegible] [illegible]
[illegible]	[illegible] [illegible]
[illegible] [illegible]	[illegible] [illegible]
[illegible] [illegible] [illegible]	[illegible] [illegible]
[illegible]: [illegible] [illegible]	[illegible] [illegible]

5.1 [illegible], [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] Rust [illegible] [illegible] [illegible] [illegible] Hello World [illegible] [illegible]:

```
fn main() {
    println!("[illegible] 🌍!");
}
```

[illegible] [illegible] [illegible]:

- [illegible] [illegible] `fn` [illegible] [illegible].
- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] C [illegible] C++ [illegible] [illegible].
- [illegible] `main` [illegible] [illegible] [illegible] [illegible].
- [illegible] Rust [illegible] [illegible] hygienic [illegible] `println!` [illegible] [illegible] [illegible] [illegible] [illegible].
- [illegible] Rust [illegible] [illegible] UTF-8 [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

[illegible] Rust [illegible]. [illegible]..

[illegible]:

- [illegible] ,Rust [illegible] /C++C/Java [illegible] (imperative) [illegible].
- [illegible] ,Rust [illegible].
- Rust [illegible] ([illegible]).
- [illegible] (hygienic) «[illegible]» [illegible] Rust [illegible] [[illegible]](https://veykril.github.io/tlborm/decl-macros/minutiae/hygiene.html) [illegible].
- [illegible] ,Rust [illegible] `(functional)` [illegible].

5.2 [illegible]

[illegible] ,Rust [illegible] «[illegible]» (immutable) [illegible]:

```rust
fn main() {
    let x: i32 = 10;
    println!("x: {x}");
    // x = 20;
    // println!("x: {x}");
}
```

- [illegible] "x = 20" [illegible] «mut» [illegible].
- «i32» [illegible] ([illegible]) [illegible].

5.3 [illegible]

[illegible].

[illegible]	[illegible]
[illegible]	i8, i16, i32, i64, i128, isize

10-

,

0

,

000_1

,

i64_123

|u8, u16, u32, u64, u128, `usize`|[illegible] [illegible] [illegible]| |

0

,

123

,

u16_10

|`f32`, `f64`|[illegible] [illegible] [illegible] [illegible]| |

3.14

,

10.0e20-

,

f32_2

|`bool`|`true`, `false`|[illegible]| |'∞' ,'`char`|'a', 'α|[illegible] [illegible] [illegible]| |

:[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

- `iN, uN` [illegible] `fN` [illegible] [illegible] N [illegible] [illegible] [illegible].,
- `isize` [illegible] `usize` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible],
- `char` [illegible] [illegible] 32 [illegible] [illegible] [illegible].,
- `bool` [illegible] [illegible] 8 [illegible] [illegible] [illegible].

:[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

- [illegible] [illegible] [illegible] [illegible] `_` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. »

  `000_1`

  [illegible] [illegible] [illegible]

  `1000`

  [illegible])

  `00_10`

  [illegible] [illegible] [illegible] (

  `i64_123`

  [illegible] [illegible] [illegible]

```
123i64
```

.«[illegible]

5.4 [illegible]

```
} fn interproduct(a: i32, b: i32, c: i32) -> i32
      ;return a * b + b * c + c * a
                                   {

                            } ()fn main
;((println!("result: {}", interproduct(120, 100, 248
                                   {
```

[illegible] ”main” [illegible] :[illegible] .[illegible] .[illegible]

.[illegible]

What about integer overflow? In C and C++ overflow of *signed* integers is actually undefined, .and might do unknown things at runtime. In Rust, it's defined

«i32» [illegible] «i16» [illegible] ([illegible]) [illegible] .[illegible] .[illegible] a *) [illegible] .(b).saturating_add(b * c).saturating_add(c *a

.[illegible]

5.5 [illegible]

:[illegible] Rust [illegible]

```
    } (fn takes_u32(x: u32
  ;("{println!("u32: {x
                       {

      } (fn takes_i8(y: i8
   ;("{println!("i8: {y
                       {

              } ()fn main
          ;let x = 10
          ;let y = 20

        ;(takes_u32(x
         ;(takes_i8(y
     ;(takes_u32(y //
                       {
```

[illegible] Rust [illegible] □ [illegible].

[illegible] «any [illegible]» [illegible]. [illegible] □ [illegible]. [illegible].

[illegible] Rust [illegible] «i32» [illegible]. [illegible] «{integer}» [illegible]. [illegible] «f64» [illegible].

```
} ()fn main
;let x = 3.14
;let y = 20
;(assert_eq!(x, y
`{ERROR: no implementation for `{float} == {integer //
{
```

5.6 [illegible]: [illegible]

[illegible] «[0□1]» [illegible]. [illegible] n>1□ [illegible] n [illegible] n-1 □ n-2 [illegible].

[illegible] (fib(n [illegible] n [illegible]. [illegible] panic [illegible]

```
} fn fib(n: u32) -> u32
} if n < 2
[illegible]//
("[illegible]")!todo
} else {
[illegible] //
("[illegible]")!todo
{
{

} ()fn main
;let n = 20
;((println!("fib({n}) = {}", fib(n
{
```

5.6.1 [illegible]

```
} fn fib(n: u32) -> u32
} if n < 2
;return n
} else {
;(return fib(n - 1) + fib(n - 2
{
{
                           } ()fn main
                       ;let n = 20
;((println!("fib({n}) = {}", fib(n
                                     {
```

6 □□□

□□□□□□ □□□□□□ □□□□□ □□□□□□

:□□□□□ □□□ .□□□□□ □□□□ □□□□□□ □□□ □□□□□ □□□□□ □□□□ □□□□

□□□□□□□□	□□□□□ □□□□
if □□□□□□□	□□□□□□ □
□□□□□□□	□□□□□□ □
continue □ break	□□□□□□ □
□□□□□□□□□ □ □□□□□□□	□□□□□□ □
□□□□□□	□□□□□□ □
□□□□□□□□	□□□□□□ □
Collatz □□□□□□□ :□□□□□□	□□□□□□ □□

if □□□□□□□ 6.1

:□□□□□□□ □□□□□□□□ □□□□□□□ □□□□□ □□□□□□ □□ □□ if □□□□□□ □□□□

```
                    } ()fn main
                   ;let x = 10
                   } if x == 0
          ;("!□□□")!println
            } else if x < 100 {
       ;("println!("biggish
                       } else {
          ;("println!("huge
                             {
                                 {
```

□□□□□□□□□ □□ □□□□□□ □□□□□□□ □□□□□□□ □□ □□□□□□ □□ □□□□□□ □□ if □□ □□□□□□□□□□ ,□□□□□□ □□□□ □□□□□ □□□
:□□□ □□□□□□□□□ □□□□ □ □□□□□□ □□□□ if □□□□□ □□ □□□□ □□□□□□ □□□□□□ .□□□□□

```
                                  } ()fn main
                                 ;let x = 10
;{ "□□□□□" } else { "□□□□□" } let size = if x < 20
                 ;(size ,"{} :□□□ □□□□□□□")!println
                                     {
```

[illegible] if [illegible] (if [illegible] else) [illegible] x / 2 [illegible] ; [illegible] .[illegible]

An if expression should be used in the same way as the other expressions. For example, when it is used in a let statement, the statement must be terminated with a ; as well. Remove .the ; before println! to see the compiler error

6.2 [illegible]

[illegible] Rust [illegible]: ”loop” [illegible] ”while” [illegible] ”for”:

[illegible] while

[illegible]while [illegible].

```
                                  } ()fn main
                    ;let mut x = 200
                     } while x >= 10
                      ;x = x / 2
                                    {
          ;("{x: {x [illegible]")!println
                                        {
```

6.2.1 for

[illegible] for [illegible]:

```
                                  } ()fn main
                     } for x in 1..5
             ;("{println!("x: {x
                                    {

         } [for elem in [1, 2, 3, 4, 5
         ;("{println!("elem: {elem
                                    {
                                        {
```

- «for» [illegible] «[illegible]» [illegible] Iterators [illegible].
- [illegible] for [illegible] 4 [illegible] 5=..1 [illegible].

6.2.2 loop

loop [illegible] «break» [illegible].

```
                                  } ()fn main
                       ;let mut i = 0
                                  } loop
                           ;i += 1
```

```
;("{println!("{i
    } if i > 100
      ;break
               {
                   {
                       {
```

continue [illegible] break 6.3

.[illegible] `continue` [illegible]

If you want to exit any kind of loop early, use `break`. With `loop`, this can take an optional .expression that becomes the value of the `loop` expression

```
               } ()fn main
        ;let mut i = 0
                } loop
           ;i += 1
        } if i > 5
        ;break
                 {
   } if i % 2 == 0
     ;continue
                 {
;(println!("{}", i
                     {
                         {
```

Note that `loop` is the only looping construct which can return a non-trivial value. This is because it's guaranteed to only return at a `break` statement (unlike `while` and `for` loops, .(which can also return when the condition fails

[illegible] 6.3.1

(label) [illegible] `break` [illegible] `continue` [illegible] :[illegible]

```
                                          } ()fn main
   ;[[let s = [[5, 6, 7], [8, 9, 10], [21, 15, 32
                   ;let mut elements_searched = 0
                           ;let target_value = 10
                         } outer: for i in 0..=2'
                             } for j in 0..=2
                  ;elements_searched += 1
             } if s[i][j] == target_value
                        ;break 'outer
                                        {
                                            {
                                                {
;("{print!("elements searched: {elements_searched
                                                    {
```

6.4 [illegible]

[illegible]

[illegible] Rust [illegible] «{}» [illegible]. [illegible]:

```rust
fn main() {
    let z = 13;
    let x = {
        let y = 10;
        println!("y: {y}");
        z - y
    };
    println!("x: {x}");
}
```

[illegible] `;` [illegible] `()` [illegible].

- [illegible]. [illegible] `;` [illegible] `return` [illegible].

6.4.1 [illegible]

[illegible] (scope) [illegible].

[illegible]:

```rust
fn main() {
    let a = 10;
    println!("before: {a}");
    {
        let a = "hello";
        println!("inner scope: {a}");

        let a = true;
        println!("shadowed in inner scope: {a}");
    }

    println!("after: {a}");
}
```

- [illegible] «b» [illegible].
- Shadowing is different from mutation, because after shadowing both variables' memory locations exist at the same time. Both are available under the same name, depending where you use it in the code.
- [illegible].
- [illegible] `.unwrap()`

[illegible].

6.5 [illegible]

```
fn gcd(a: u32, b: u32) -> u32 {
    if b > 0 {
        gcd(b, a % b)
    } else {
        a
    }
}

fn main() {
    println!("gcd: {}", gcd(143, 52));
}
```

- [illegible] ([illegible] [illegible]).
- [illegible] ([illegible]) [illegible]. [illegible]

  ;

  [illegible].
- Some functions have no return value, and return the 'unit type', `()`. The compiler will infer this if the return type is omitted.
- [illegible] (overloading) [illegible] -- [illegible].
  - [illegible]. [illegible]. [illegible].
  - [illegible].

6.6 [illegible]

[illegible] Rust [illegible]. [illegible] «!» [illegible]. [illegible] Rust [illegible].

- `println!(format, ..)` [illegible] [`std::fmt`](https://doc.rust-lang.org/std/fmt/index.html) [illegible].
- `format!(format, ..)` [illegible] `println!` [illegible].
- `dbg!(expression)` [illegible].
- `todo!()` [illegible]. panic [illegible].
- `unreachable!()` [illegible]. [illegible].

```
           } fn factorial(n: u32) -> u32
                ;let mut product = 1
                    } for i in 1..=n
             ;(product *= dbg!(i
                                   {
                             product
                                       {

            } fn fizzbuzz(n: u32) -> u32
                             ()!todo
                                       {

                             } ()fn main
                          ;let n = 4
;((println!("{n}! = {}", factorial(n
                                       {
```

[illegible] .[illegible] [illegible] .[illegible]

[illegible] .[illegible]

Collatz [illegible] :[illegible] 6.7

The Collatz Sequence is defined as follows, for an arbitrary n

1

:greater than zero

- n* [illegible]

  i

  n* [illegible] (sequence) [illegible] = *

  i

  .[illegible] *

- n* [illegible]

  i

  n* [illegible] *

  i+[illegible]

  n* = *

  i

  .[illegible]/*

- n* [illegible]

  i

n* □□□□□□□ □□□□□□□ □□□□ *

i+□

n* * □ = *

i

.□ + *

n* □□ □□□□□ □□ □□□□□□□ □□□□□□□ □□

i

:□ = *

- n* □□ □□□□□ □□□□ □

2

;10 = □ + □ * □ = *

- n* □□ □□□□□ □□□□ □□

3

;□ = □ / □□ = *

- n* □□ □□□□□ □□□□ □

4

;16 = □ + □ * □ = *

- n* □□ □□□□□ □□□□ □□

5

;8 = □ / □□ = *

- n* □□ □□□□□ □□□□ □

6

;4 = □ / □ = *

- n* □□ □□□□□ □□□□ □

7

;□ = □ / □ = *

- n* □□ □□□□□ □□□□ □

□

□ ;□ = *

- .□□□□□□ □□□□□□ □□ □□□□□□□

.□□□□ □□□□□□□ □□ □□□□ □□□□□ □□□□□□ n □□ □□□□□ Collatz □□□□□□□ □□□□ □□ □□□□□□□□ □□□□□ □□

```
.`Determine the length of the collatz sequence beginning at `n ///
                        } fn collatz_length(mut n: i32) -> u32
                            ("□□ □□□□□□□□□□ □□ □□□")!todo
                                                             {
} ()fn main
("□□ □□□□□□□□□ □□ □□□")!todo
{
```

□□□□□ 6.7.1

```
.`Determine the length of the collatz sequence beginning at `n ///
} fn collatz_length(mut n: i32) -> u32
;let mut len = 1
} while n > 1
;{ n = if n % 2 == 0 { n / 2 } else { 3 * n + 1
;len += 1
{
len
{

} ()fn test_collatz_length
;(assert_eq!(collatz_length(11), 15
{

} ()fn main
;((println!("Length: {}", collatz_length(11
{
```

II [illegible]

[illegible] [illegible] [illegible] :[illegible] [illegible]

[illegible] 7

[illegible] [illegible]

[illegible] [illegible] 10 [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] 2 [illegible] [illegible] 35 [illegible] [illegible] [illegible]. [illegible] [illegible]:

[illegible]	[illegible] [illegible]
[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]	[illegible] [illegible] [illegible] [illegible]
[illegible] [illegible] [illegible] [illegible] [illegible]	[illegible] [illegible]

8 □□□

□□ □□□□□□ □ □□ □□□□□

:□□□□□ □□□□ .□□□□□ □□□□ □□□□□□ 35 □□□□□ □□□□□ □□□□ □□□□

□□□□□ □□□□	□□□□□□□□
□□□□□□ □	□□□□□□□□□
□□□□□□ □	□□□□□□□□
□□□□□□ □	□□□□□□ □□□□□□
□□□□□□ □	□□□□□□□□□□□ □□□□ □ □□□□□□□□
□□□□□□ □□	□□ □□ □□ □□□□□□□□□□□ :□□□□□□

□□□□□□□□ 8.1

```
} ()fn main
;[let mut a: [i8; 10] = [42; 10
;a[5] = 0
;("{?:println!("a: {a
{
```

□□□□□□ □□□□ □□ □□□□□□ □□ •

[T; N]

□□□□□□

N

□□□□□□ □□□□ □□ □□□□□ (□□□□□□□□□ □□□□□ □□□□□ □□)

T

□□ □□□□□ □□□□ □□ □□□□□ □□ □□□□ □□ □□□□□ □□□□□□ □□□□ □□ □□□□□□ □□□□□□ □□□□□ .□□□□

[u8; 3]

□

[u8; 4]

.□□□□□□□ □□□□□□□ □□□□ □□ □□□□□□□ □□□□ □□

- [illegible] Rust [illegible].

- [illegible].

- [illegible]

  `println!`

  [illegible]

  `?`

  [illegible]:

  `{}`

  [illegible]

  `{:?}`

  [illegible].

- [illegible]

  `#`

  [illegible]

  `{a:#?}`

  [illegible] «[illegible]» [illegible].

8.2 [illegible]

```rust
fn main() {
    let t: (i8, bool) = (7, true);
    println!("t.0: {}", t.0);
    println!("t.1: {}", t.1);
}
```

- [illegible].

- [illegible].

- [illegible]

  `t.0`

  [illegible]

  `t.1`

  [illegible].

- [illegible] `()` [illegible] `unit type` [illegible] `void` [illegible].

8.3 [illegible]

[illegible] for [illegible] ([illegible]) [illegible].

```
                                                  } ()fn main
        ;[let primes = [2, 3, 5, 7, 11, 13, 17, 19
                                   } for prime in primes
                                 } for i in 2..prime
                        ;(assert_ne!(prime % i, 0
                                                    {
                                                        {
                                                            {
```

[illegible] IntoIterator [illegible].

The `assert_ne!` macro is new here. There are also `assert_eq!` and `assert!` macros. These are always checked, while debug-only variants like debug_`assert!` compile to nothing in .release builds

8.4 [illegible]

[illegible]:

```
                  } ((fn print_tuple(tuple: (i32, i32
                              ;let left = tuple.0
                             ;let right = tuple.1
     ;("{println!("left: {left}, right: {right
                                                {
```

[illegible] Rust [illegible]:

```
                  } ((fn print_tuple(tuple: (i32, i32
                         ;let (left, right) = tuple
     ;("{println!("left: {left}, right: {right
                                                {
```

- [illegible] "irrefutable" [illegible] = [illegible].
- [illegible] «let» [illegible].
- Rust [illegible].
- [illegible]-[illegible].

8.5 [illegible]: [illegible]

[illegible]:

`let array = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];`

[illegible]

[illegible] transpose

[illegible]:

```
 ⎛⎡1 2 3⎤⎞      ⎡1 4 7⎤
"transpose"⎜⎢4 5 6⎥⎟  "=="⎢2 5 8⎥
 ⎝⎣7 8 9⎦⎠      ⎣3 6 9⎦
```

[illegible] https://play.rust-lang.org/ [illegible]:

```rust
// TODO: [illegible].

fn transpose(matrix: [[i32; 3]; 3]) -> [[i32; 3]; 3] {
    unimplemented!()
}

fn test_transpose() {
    let matrix = [
        [101, 102, 103], //
        [201, 202, 203],
        [301, 302, 303],
    ];
    let transposed = transpose(matrix);
    assert_eq!(
        transposed,
        [
            [101, 201, 301], //
            [102, 202, 302],
            [103, 203, 303],
        ]
    );
}

fn main() {
    let matrix = [
        [101, 102, 103], // <-- [illegible] rustfmt [illegible].
        [201, 202, 203],
        [301, 302, 303],
    ];

    println!("matrix: {:#?}", matrix);
    let transposed = transpose(matrix);
    println!("[illegible]: {:#?}", transposed);
}
```

□□□□□ 8.5.1

```
                            } [fn transpose(matrix: [[i32; 3]; 3]) -> [[i32; 3]; 3
                                                 ;[let mut result = [[0; 3]; 3
                                                               } for i in 0..3
                                                           } for j in 0..3
                                          ;[result[j][i] = matrix[i][j
                                                                         {
                                                                             {
                                                                        result
                                                                                 {

                                                             } ()fn test_transpose
                                                                ] = let matrix
                                                       // ,[101, 102, 103]
                                                          ,[201, 202, 203]
                                                          ,[301, 302, 303]
                                                                            ;[
                                           ;(let transposed = transpose(matrix
                                                                   )!assert_eq
                                                               ,transposed
                                                                         ]
                                                   // ,[101, 201, 301]
                                                      ,[102, 202, 302]
                                                      ,[103, 203, 303]
                                                                         [
                                                                            ;(
                                                                                 {

                                                                       } ()fn main
                                                                ] = let matrix
.□□□ □□□□□□ □□□□□ □□ □□ rustfmt □□□□□□ □□□□□ □□□□□□ □□□ --> // ,[101, 102, 103]
                                                          ,[201, 202, 203]
                                                          ,[301, 302, 303]
                                                                            ;[

                                            ;(println!("matrix: {:#?}", matrix
                                           ;(let transposed = transpose(matrix
                                 ;(transposed ,"{?#:} :□□□ □□□ □□□□□□")!println
                                                                                 {
```

9 [illegible]

[illegible]

:[illegible] [illegible] .[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

[illegible]	[illegible] [illegible]
[illegible] [illegible]	[illegible] [illegible]
[illegible] [illegible]	[illegible] [illegible]
[illegible]	[illegible] [illegible]
[illegible]	[illegible] [illegible]
[illegible] :[illegible]	[illegible] [illegible]

9.1 [illegible] [illegible]

A reference provides a way to access another value without taking ownership of the value, and is also called ”borrowing”. Shared references are read-only, and the referenced data .cannot change

```
} ()fn main
    ;'let a = 'A
    ;'let b = 'B
    ;let mut r: &char = &a
    ;(println!("r: {}", *r
    ;r = &b
    ;(println!("r: {}", *r
{
```

.[illegible] [illegible] [illegible] & [illegible] [illegible] [illegible] [illegible] [illegible] .[illegible] T& [illegible] [illegible] T [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] .[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] ”[illegible] [illegible]“ [illegible] [illegible] [illegible] * [illegible]

:[illegible] [illegible] [illegible] (dangling) [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

```
} (fn x_axis(x: &i32) -> &(i32, i32
    ;(let point = (*x, 0
    ;return &point
{
```

- [illegible] ([illegible]) ”borrow” [illegible] 3 [illegible].
- [illegible] C [illegible] C++ [illegible] Rust [illegible].
- Rust [illegible] & - [illegible].
- [illegible] Dereference [illegible])

  ()ref_x.count_ones

  [illegible].(
- [illegible] r [illegible] (r = &b). [illegible] r [illegible]. [illegible] C++ [illegible].
- [illegible] ”r = ”X*” [illegible].
- Rust [illegible]. [illegible] Rust [illegible]. x_axis [illegible] point [illegible] «[illegible]» [illegible].
- [illegible] «[illegible]» [illegible].

9.2 [illegible]

[illegible] &T mut [illegible].

```
} ()fn main
    ;(let mut point = (1, 2
    ;let x_coord = &mut point.0
    ;x_coord = 20*
    ;("{?:println!("point: {point
{
```

[illegible]:

- ”[illegible]” [illegible] ([illegible]) [illegible]. [illegible] x_coord [illegible] point.0& [illegible] point.0 [illegible].
- [illegible] «let mut x_coord: &i32» [illegible] «let x_coord: &mut i32» [illegible].
[illegible]

[illegible].

9.3 [illegible]

[illegible] (view) [illegible]:

```
                                   } ()fn main
        ;[let mut a: [i32; 6] = [10, 20, 30, 40, 50, 60
                            ;("{?:println!("a: {a

                      ;[let s: &[i32] = &a[2..4

                            ;("{?:println!("s: {s
                                                   {
```

- [illegible].
- [illegible]: [illegible]

  `[a[3`

  [illegible]

  `s`

  [illegible]
- [illegible]

  `a`

  [illegible] (slice) [illegible].
- [illegible] ([illegible]) [illegible]

  `[()a[0..a.len&`

  [illegible]

  `[()a[..a.len&`

  [illegible].
- [illegible]

  `[()a[2..a.len&`

  [illegible]

  `[..a[2&`

  [illegible].
- [illegible]

  `[..]a&`

  [illegible].

- `s`

[illegible] i32 [illegible].

`s`

`)`

`&[i32]`

([illegible].
[illegible].

- [illegible].

`a`

[illegible] ([illegible]) [illegible].

- [illegible]

`[a[3`

[illegible]
[illegible]

`a`

[illegible]

`a`

`□`

`s`

[illegible] . [illegible]

`!println`

[illegible]. [illegible]-
[illegible] (the borrow checker) [illegible].

9.4 [illegible]

[illegible]:

- `&str` [illegible] UTF-8 [illegible] `&[u8]` [illegible].
- `String` is an owned buffer of UTF-8 encoded bytes, similar to `Vec<T>`.

```
fn main() {
    let s1: &str = "[illegible]";
    println!("s1: {s1}");

    let mut s2: String = String::from("[illegible]");
    println!("s2: {s2}");
    s2.push_str(s1);
    println!("s2: {s2}");

    let s3: &str = &s2[s2.len() - s1.len()..];
```

```
;("{println!("s3: {s3
{
```

- str&

[illegible] UTF-8 [illegible]. [illegible]

String

("Hello") [illegible].

- [illegible]

String

[illegible] wrapper [illegible]. [illegible]

<Vec<T

[illegible] Owned [illegible].

- [illegible]

()String::from

[illegible].

()String::new

[illegible]

()push

[illegible]

()push_str

[illegible].

- [illegible]

()!format

[illegible] Owned [illegible]. [illegible]

()!println

[illegible].

- [illegible] str& [illegible] String [illegible] & [illegible]. [illegible] chars [illegible].

- [illegible]

++C

:

str&

[illegible]

`const char*`

[illegible]

C++

[illegible] [illegible]. [illegible] [illegible]

`String`

[illegible] [illegible]

`std::string`

[illegible]

C++

[illegible] ([illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] UTF-8 [illegible] [illegible] [illegible] [illegible] [illegible] Small-String [illegible] [illegible] [illegible]).

- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

`&[u8]`

[illegible] [illegible]:

```
fn main() {
    println!("{:?}", b"abc");
    println!("{:?}", &[97, 98, 99]);
}
```

- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

&str

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]: `"\\n" == r"\n"`. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] # [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]:

```
fn main() {
    println!(r#"<a href="link.html">link</a>"#);
    println!("<a href=\"link.html\">link</a>");
}
```

9.5 [illegible]: [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `[f64;3]` [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

```
// [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
// [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] `sqrt()` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `v.sqrt()`


fn magnitude(...) -> f64 {
    todo!()
}

// [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
```

```
.[illegible] [illegible] [illegible] [illegible] [illegible] //

} (...)fn normalize
        ()!todo
                {

.[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `main` [illegible] //

} ()fn main
;(([magnitude(&[0.0, 1.0, 0.0 ,"{} :[illegible] [illegible] [illegible] [illegible]")!println

;[let mut v = [1.0, 2.0, 9.0
;((v:?}: {}", magnitude(&v} [illegible]")!println
;(normalize(&mut v
;((magnitude(&v ,"{} :[illegible] [illegible] [illegible] {?:v} [illegible]")!println
{
```

[illegible] 9.5.1

```
.[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] ///
} fn magnitude(vector: &[f64; 3]) -> f64
    ;let mut mag_squared = 0.0
        } for coord in vector
    ;mag_squared += coord * coord
                            {
            ()mag_squared.sqrt
                              {

.[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] 1.0 [illegible] [illegible] [illegible] [illegible] ///
} ([fn normalize(vector: &mut [f64; 3
    ;(let mag = magnitude(vector
        } for item in vector
           ;item /= mag*
                       {
                         {

} ()fn main
;(([magnitude(&[0.0, 1.0, 0.0 ,"{} :[illegible] [illegible] [illegible] [illegible]")!println

;[let mut v = [1.0, 2.0, 9.0
;((v:?}: {}", magnitude(&v} [illegible]")!println
;(normalize(&mut v
;((magnitude(&v ,"{} :[illegible] [illegible] [illegible] {?:v} [illegible]")!println
{
```

10 [illegible]

[illegible]

:[illegible] .[illegible]

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
Enums	[illegible]
Static	[illegible]
[illegible]	[illegible]
:[illegible]	[illegible]

10.1 [illegible]

:[illegible] Rust [illegible] ++C [illegible] C [illegible]

```
} struct Person
,name: String
,age: u8
{

} (fn describe(person: &Person
;(person.name, person.age ,"[illegible] [illegible] {} {}")!println
{

} ()fn main
;{ age: 27 ,("[illegible]")let mut peter = Person { name: String::from
;(describe(&peter

;peter.age = 28
;(describe(&peter

;("[illegible]")let name = String::from
;let age = 39
;{ let avery = Person { name, age
```

```
                                             ;(describe(&avery

  ;{ avery.. ,("□□□")let jackie = Person { name: String::from
                                            ;(describe(&jackie
                                                              {
```

:□□□□□□ □□□□□

- .□□□□□□□ □□□□ ++C □□ C □□□□□□ Rust □□ (Structs) □□□□□□□□□□
  - .□□□□□ `typedef` □□ □□□□□□ □□□□ □□ □□□□□□ □□□□□ □C □□□□□□□ □ ++C □□□□□□
  - .□□□□□□ □□□□□ □□□□□□□□ □□□□□□□□□□ □□□□ Rust □□ □++C □□□□□□□
- □□□□□ □□□□□□□□□□ □□ □□□□□□□□ □□□□□□ □□ □□□□□ □□□□□□□ □□□□□□□□ □□ □□ □□□□ □□□□□□□□ □□□□□ □□□□ .□□□□□
  - □□ □□□□□ □□□□□□□□□ □□□□□□ □□□□ □□□□□ (`;struct Foo` □□□□□□) □□□□□□□□ □□□□□ □□□□□□□□□□□ □□□□□□□□ □□□□□□□□ □□□□ □□□□□□ □□□□□□□□□□□ □□□□ □□ □□□□ □□ □□ (trait) □□□□ □□ □□□□□□□□□□ .□□□□□ □□□□□□ □□□□□□ □□□□ □□ □□□□□□□□ □□
  - □□□□□□ □□ □□□□□ □□□□□□ □□□□□□ □□ (Tuple structs) □□□□□ □□□□□□□□□□□ □□□□□ □□□□□□□□ .□□□□□□□□ □□□□ □□□□□□□□ □□□□ □□ □□□□□□□□ □□□□□□□□□□
- □□ □□□□□□□□□ □□ □□ □□□□□□□□ □□□□□□□□□□ □□□□□□□□ □□□□□□ □□□□□□□ □□ □□□□□□□□□□□ □□□□ □□ □□□□ .□□□□□ □□□□□□ □□□□□□□□ □□□□ □□
- □□□□□ □□□□ □□□□□□□ □□□□□□□□ □□ □□ □□□□□□□□ □□□□□ □□ □□□□□□□ □□□□□□ □□ □□ `avery..` □□□□□□□□ .□□□□□ □□□□□ □□□□□□□ □□□□□□□ □□□□□ □□□□ .□□□□□ □□□□□ □□□□□□□□ □□ □□□□□ □□□□ □□□□□□□ □□□□□

□□□□□□□ □□□□□□□ 10.2

:□□□□□ □□□□□□□□□ tuple □□□□□□□ □□ □□□□□□□□□□ □□□□□□□ □□□□□□□□□ □□□□□□□ □□□□ □□□□

```
                                               ;(struct Point(i32, i32

                                                             } ()fn main
                                           ;(let p = Point(17, 23
                                ;(println!("({}, {})", p.0, p.1
                                                                       {
```

:□□□□□□ □□□□□□□□□ (□□□□□□□ □□□□□□□ newtypes □□) single-field wrapper□□□□ □□□□□ □□□□

```
                                         ;(struct PoundsOfForce(f64
                                              ;(struct Newtons(f64

                      } fn compute_thruster_force() -> PoundsOfForce
           ("□□□□□ □□□□□ □□ □□□□□ □□□□□ □□□□□□□□ □□ □□")!todo
                                                                   {

                         } (fn set_thruster_force(force: Newtons
                                                         ... //
                                                                   {

                                                         } ()fn main
                      ;()let force = compute_thruster_force
                                     ;(set_thruster_force(force
                                                                   {
```

- Newtypes □□ □□□□ □□□□□□ □□□□□ □□□□□□□□□ □□□□□□□□□ □□□□□□ □□□□ □□□□□ □□ □□□ □□□ □□□□□□ (primitive type) □□□□□ □□ □□□□□□ □□□□□:
  - □□□ □□□ □□ □□□□□ □□□□□□□ □□□□□□□ □□□□□ □□□□□□: Newtons □□ □□□□□ □□□□□.
  - □□□□□□ □□□□□□ □□ □□□□□□ □□ □□□□□□□ □□□□□□□□□□□□ □□ □□□□□□□ □□□□□ □□□□□□□□□□ □□□□□ □□□□□ □□ □□ □□□□ □□□□□□□□ □□□□□□□ □□ □□ □□□□□□ □□□□□: PhoneNumber(String) □□ OddNumber(u32).
- □□□□□ □□□□□□□ □□□□□□ f64 □□ □□□□ Newtons □□ □□ □□□□□□□ □□ single field □□ □□□□ □□□□□ □□□□□ □□□□□.
  - Rust □□□□□□□□ □□□□□□□□ □□□ □□□□□ □□ □□□□□ □□□□□□□ □□□□□□ automatic unwrapping □□ □□□□□□□□□□□□□□ □□□□□□□□□ □□ boolean□□ □□ □□□□□□ □□□□□□ □□□□□.
  - □□□□□ Operator overloading □□ □□□□ □□□□ □□□□□ □□□□ □□□□□ □□ □□□□□ (generics).
- □□□□ □□□□□□ □□□□□□ □□□□□□□ □□ □□□□□ □□□□□□□□□□ □□□□□□□□□ □□□□□ □□□□.

10.3 Enums

□□□□□ □□□□□□ enum □□□□□□ □□□□□□ □□□□ □□□□□□□□ □□ □□ □□□□ □□ □□□□□□ □□□□□□□ □□□□□ □□□□□□□ □□□□:

```rust
enum Direction {
    Left,
    Right,
}

enum PlayerMove {
    Pass,                        // Simple variant
    Run(Direction),              // Tuple variant
    Teleport { x: u32, y: u32 }, // Struct variant
}

fn main() {
    let m: PlayerMove = PlayerMove::Run(Direction::Left);
    println!("□□ □□□ □□□: {:?}", m);
}
```

□□□□□ □□□□□□:

- Enum□□ □□ □□□□ □□□□□□ □□ □□□□□ □□□□□□□□□□ □□ □□□□□□□ □□□□□□ □□ □□□□□□□ □□□□□□ □□ □□□□ □□ □□ □□□□ □□□□□ □□□□□.
- Direction □□ type □□ □□□□□□□□ □□□□□□ □□□□. □□ □□□□□□ Direction □□□□□ □□□□□: Direction::Left □ Direction::Right.
- PlayerMove is a type with three variants. In addition to the payloads, Rust will store a discriminant so that it knows at runtime which variant is in a PlayerMove value.
- □□□□□ □□□□□ □□□□□ □□□□□□□ □□□□□□□□□□ □ Enum□□□□□:
  - □□ □□□□ □□ □□□□□□□ □□ □□□□□ □□□□□ □□□□□ □□□□□ (unit struct) □□ □□□□ □□ □□□□□□ □□□□□□ □□□□□ (variant payloads) □□□□□□ □□□□□□.
  - □□□□ □□□□□ □□□□□□□ □□□□□□□ □□ Enum □□ □□ □□□□□□□□□□ □□□□□□□□□ □□□□□□ □□□□□ □□□□□□□ □□□ □□ □□ □□□□□ □□□□□ □□ □□□□□ □□□□□□ □□□□□□ □□□□□□ □□□□□ Enum □□□□□□□□ □□□□.
- Rust □□ □□□□□□ □□□□ □□□□□ □□□□□□□□□□ □□□□□□□□□□□□ (discriminant) □□□□□□□□□ □□□□□□.
  - □□ □□□□□ □□□□□□ □□ □□□□ □□□□□ □□ □□□□□□□□□□ □□□□□□□□ □□□□□ □□□□□ □□ □□□□□□ □□□□□□.
  - □□□□ □□□□□□□ □□□□□□ □□□□□ □□□□ □□□□□□□□ bit □□ □□□□□□□ □□ □□□□□□□□ bit □□□□□□□□ □□□□□□□□□□ □□□□□□□□ □□□□□□ (□□ "niche optimization") □□□□□□□□□ □□□□□□. □□□□□

[illegible] None [illegible] NULL [illegible] <Option<&u8 [illegible]
.[illegible]

[illegible] C) discriminant [illegible]) [illegible] –
:[illegible]

```
                                   } enum Bar
                                     A, // 0
                                  ,B = 10000
                                 C, // 10001
                                           {

                                  } ()fn main
     ;(println!("A: {}", Bar::A as u32
     ;(println!("B: {}", Bar::B as u32
     ;(println!("C: {}", Bar::C as u32
                                           {
```

[illegible] 2 [illegible] 10001 [illegible] discriminant [illegible] repr [illegible]
.[illegible]

[illegible]

[illegible] Enum [illegible] Rust [illegible]
.[illegible]

- [illegible] Rust [illegible] :NULL [illegible]

  ()<size_of::<T

  [illegible]

  ()<<size_of::<Option<T

  .[illegible]

[illegible] ,[illegible] .[illegible] .[illegible]

```
                         ;use std::mem::transmute

                          } macro_rules! dbg_bits
                  } <= (e:expr, $bit_type:ty$)
;((println!("- {}: {:#x}", stringify!($e), transmute::<_, $bit_type>($e
                                              ;{
                                                  {

                                     } ()fn main
                                     } unsafe
                            ;(":println!("bool
                        ;(dbg_bits!(false, u8
                         ;(dbg_bits!(true, u8

                   ;(":<println!("Option<bool
               ;(dbg_bits!(None::<bool>, u8
                ;(dbg_bits!(Some(false), u8
                ;(dbg_bits!(Some(true), u8

        ;(":<<println!("Option<Option<bool
         ;(dbg_bits!(Some(Some(false)), u8
          ;(dbg_bits!(Some(Some(true)), u8
        ;(dbg_bits!(Some(None::<bool>), u8
      ;(dbg_bits!(None::<Option<bool>>, u8

                ;(":<println!("Option<&i32
            ;(dbg_bits!(None::<&i32>, usize
             ;(dbg_bits!(Some(&0i32), usize
                                           {
                                               {
```

const 10.4

:Constants are evaluated at compile time and their values are inlined wherever they are used

```
                                         ;const DIGEST_SIZE: usize = 3
                                    ;(const ZERO: Option<u8> = Some(42

                         } [fn compute_digest(text: &str) -> [u8; DIGEST_SIZE
                       ;[let mut digest = [ZERO.unwrap_or(0); DIGEST_SIZE
                    } ()for (idx, &b) in text.as_bytes().iter().enumerate
;(digest[idx % DIGEST_SIZE] = digest[idx % DIGEST_SIZE].wrapping_add(b
                                                                 {
                                                           digest
                                                                     {

                                                            } ()fn main
                                   ;("let digest = compute_digest("Hello
                                        ;("{?:println!("digest: {digest
                                                                     {
```

.[illegible] Rust RFC [illegible]

[illegible] const [illegible] const [illegible] .[illegible] const [illegible] const [illegible] ([illegible]) [illegible]

- [illegible] constexpr [illegible] const [illegible]

  ++C

  .[illegible]

- [illegible] .[illegible]static [illegible] const [illegible]

static 10.5

:[illegible]

```
;"[illegible] [illegible] RustOS 3.14 [illegible]" = static BANNER: &str

} ()fn main
;("{println!("{BANNER
{
```

As noted in the Rust RFC Book, these are not inlined upon use and have an actual associated memory location. This is useful for unsafe and embedded code, and the variable lives through the entirety of the program execution. When a globally-scoped value does not have a reason .to need object identity, `const` is generally preferred

- .++`static` is similar to mutable global variables in C
- `static` [illegible] [illegible] [illegible] [illegible] [illegible]: [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

  `<Mutex<T`

  .[illegible] [illegible] [illegible]

[illegible] [illegible] [illegible]

Because `static` variables are accessible from any thread, they must be `Sync`. Interior .mutability is possible through a `Mutex`, atomic or similar

.Thread-local data can be created with the macro `std::thread_local`

10.6 [illegible] [illegible] [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] alias [illegible] .[illegible]

```
} enum CarryableConcreteItem
,Left    
,Right    
{

;type Item = CarryableConcreteItem

:Aliases are more useful with long, complex types //
;use std::cell::RefCell
;{use std::sync::{Arc, RwLock
;<<<<type PlayerInventory = RwLock<Vec<Arc<RefCell<Item
```

.[illegible] [illegible] `typedef` [illegible] [illegible] [illegible] [illegible] C [illegible]

10.7 [illegible]: [illegible] [illegible]

.[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] .[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] .[illegible] [illegible] `{?:}` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `[(derive(Debug]#` [illegible]

□□□□ □□□□□□ □□□□ □□ □□□□□□ □ □□ □□□□□ □□□□□□□□□□□ □□□□□ □□□□□ □□□□□ □□ main □□□□□ □□□□ □□□□□ □□□□. □□□□ □□□□□ □□□□ □□□□□ □□□□□□□ □□□□□□□ □□ □□□□ □□□□□□□□□□ □□ □□□□□ □□□□□□.

```rust
/// An event in the elevator system that the controller must react to.
enum Event {
    // TODO: add required variants
}

/// A direction of travel.
enum Direction {
    Up,
    Down,
}

/// The car has arrived on the given floor.
fn car_arrived(floor: i32) -> Event {
    todo!()
}

/// The car doors have opened.
fn car_door_opened() -> Event {
    todo!()
}

/// The car doors have closed.
fn car_door_closed() -> Event {
    todo!()
}

/// A directional button was pressed in an elevator lobby on the given floor.
fn lobby_call_button_pressed(floor: i32, dir: Direction) -> Event {
    todo!()
}

/// A floor button was pressed in the elevator car.
fn car_floor_button_pressed(floor: i32) -> Event {
    todo!()
}

fn main() {
    println!(
        "A ground floor passenger has pressed the up button: {:?}",
        lobby_call_button_pressed(0, Direction::Up)
    );
    println!("□□□□□□ □□ □□□□□ □□□□□ □□□□□□ □□□: {:?}", car_arrived(0));
    println!("□□ □□□□□□ □□□□ □□: {:?}", car_door_opened());
    println!(
        "□□ □□□□□□ □□□□□ □□□□□ 3 □□ □□□□□ □□□□□ □□□□: {:?}",
        car_floor_button_pressed(3)
    );
    println!("□□ □□□□□□ □□□□□ □□: {:?}", car_door_closed());
```

```
   ;((car_arrived(3 ,"{?:} :□□□ □□□□□□ □ □□□□□ □□ □□□□□□")!println
                                                                     {
```

□□□□□□ 10.7.1

```
.An event in the elevator system that the controller must react to ///
                                                          } enum Event
                                         .A button was pressed ///
                                            ,(ButtonPressed(Button

                       .The car has arrived at the given floor ///
                                                ,(CarArrived(Floor

                                  .The car's doors have opened ///
                                                    ,CarDoorOpened

                                  .The car's doors have closed ///
                                                    ,CarDoorClosed
                                                                     {

                         .A floor is represented as an integer ///
                                                     ;type Floor = i32

                                        .A direction of travel ///
                                                      } enum Direction
                                                               ,Up
                                                             ,Down
                                                                     {

                                     .A user-accessible button ///
                                                         } enum Button
            .A button in the elevator lobby on the given floor ///
                                      ,(LobbyCall(Direction, Floor

                                .A floor button within the car ///
                                                  ,(CarFloor(Floor
                                                                     {

                           .The car has arrived on the given floor ///
                                 } fn car_arrived(floor: i32) -> Event
                                          (Event::CarArrived(floor
                                                                     {

                                        .The car doors have opened ///
                                       } fn car_door_opened() -> Event
                                              Event::CarDoorOpened
                                                                     {

                                        .The car doors have closed ///
                                       } fn car_door_closed() -> Event
                                              Event::CarDoorClosed
```

```
                                                                            {

.A directional button was pressed in an elevator lobby on the given floor ///
          } fn lobby_call_button_pressed(floor: i32, dir: Direction) -> Event
                      ((Event::ButtonPressed(Button::LobbyCall(dir, floor
                                                                            {

                          .A floor button was pressed in the elevator car ///
                           } fn car_floor_button_pressed(floor: i32) -> Event
                            ((Event::ButtonPressed(Button::CarFloor(floor
                                                                            {

                                                                  } ()fn main
                                                                )!println
          ,"{?:} :A ground floor passenger has pressed the up button"
                          (lobby_call_button_pressed(0, Direction::Up
                                                                       ;(
          ;((car_arrived(0 ,"{?:} :□□□ □□□□□ □□□□ □□□□ □□ □□□□□")!println
                    ;(()car_door_opened ,"{?:} :□□ □□□ □□□□□ □□")!println
                                                                )!println
                       ,"{?:} :□□□ □□□□ □□□□ □□ 3 □□□□ □□□□ □□□□□ □□"
                                          (car_floor_button_pressed(3
                                                                       ;(
                   ;(()car_door_closed ,"{?:} :□□ □□□□ □□□□□ □□")!println
             ;((car_arrived(3 ,"{?:} :□□□ □□□□□ □ □□□□ □□ □□□□□")!println
                                                                            {
```

III □□□

□□□ :□ □□□

11 [illegible]

[illegible]

[illegible] Rust [illegible] Rust [illegible]:

- [illegible]: [illegible].
- [illegible]: [illegible].
- Traits: [illegible].
- Generics: [illegible].
- [illegible] traits: [illegible] Rust.

[illegible]

[illegible]. [illegible]:

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

12 [illegible]

[illegible]

[illegible]. [illegible]:

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
Enums [illegible]	[illegible]
Let [illegible]	[illegible]
[illegible]: [illegible]	[illegible]

12.1 [illegible]

[illegible] match [illegible]. [illegible].

[illegible] switch [illegible] C [illegible] C++:

```rust
fn main() {
    let input = 'x';
    match input {
        'q' => println!("[illegible]"),
        'a' | 's' | 'w' | 'd' => println!("[illegible]"),
        '0'..='9' => println!("[illegible]"),
        key if key.is_lowercase() => println!("[illegible]: {key}"),
        _ => println!("[illegible]"),
    }
}
```

[illegible] _ [illegible] (Wildcard) [illegible]. [illegible] _ [illegible].

match [illegible]. [illegible] if [illegible] match [illegible]. [illegible]. [illegible] () [illegible].

[illegible] (key [illegible]) [illegible].

[illegible] guard [illegible] `match` [illegible].

[illegible]:

- [illegible]
  - `|` [illegible] or
  - `..` [illegible]
  - 1..=5

    [illegible]
  - `_` [illegible]
- guard [illegible], [illegible].
- [illegible] `if` [illegible]. [illegible] `if` [illegible] (`=>` [illegible]). [illegible] `if` [illegible] `match` [illegible].
- [illegible] guard [illegible] `|` [illegible].

12.2 [illegible]

tuple [illegible]:

```
struct Foo {
    x: (u32, u32),
    y: u32,
}

fn main() {
    let foo = Foo { x: (1, 2), y: 3 };
    match foo {
        Foo { x: (1, b), y } => println!("x.0 = 1, b = {b}, y = {y}"),
        Foo { y: 2, x: i }   => println!("y = 2, x = {i:?}"),
        Foo { y, .. }        => println!("y = {y}, [illegible]"),
    }
}
```

- [illegible] foo [illegible].
- [illegible] Foo [illegible].
- [illegible] 2 [illegible] const [illegible].

12.3 Enums

[illegible] tuple [illegible] enum [illegible]:

[illegible] enum [illegible]:

```
} enum Result
,(Ok(i32
,(Err(String
{

} fn divide_in_two(n: i32) -> Result
} if n % 2 == 0
(Result::Ok(n / 2
} else {
(("[illegible] {n} [illegible]")!Result::Err(format
{
{

} ()fn main
;let n = 100
} (match divide_in_two(n
,("{half} [illegible] {Result::Ok(half) => println!("{n
,("{msg} :[illegible]")!Result::Err(msg) => println
{
{
```

[illegible] Result [illegible]. [illegible] `half` [illegible] Ok [illegible]. [illegible] `msg` [illegible].

- [illegible] `if/else` [illegible] enum [illegible] `match` [illegible].
- [illegible] None [illegible] enum [illegible].
- [illegible] enum [illegible].
- [illegible] ([illegible]) [illegible]. [illegible] Rust [illegible].
- [illegible] `divide_in_two` [illegible] `result` [illegible] `match` [illegible] `msg` [illegible] `&result` [illegible] `result` [illegible] `msg` [illegible] “match ergonomics” [illegible] Rust 2018 [illegible] Rust [illegible] `msg` [illegible] `ref msg` [illegible].

12.4 [illegible] Let

[illegible] Rust [illegible]. [illegible]:

- [illegible] `if let`
- [illegible] `let else`
- [illegible] `while let`

[illegible] `if let`

`[illegible] if let` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]:

```
use std::time::Duration;

fn sleep_for(secs: f32) {
    if let Ok(dur) = Duration::try_from_secs_f32(secs) {
        std::thread::sleep(dur);
        println!("slept for {:?}", dur);
    }
}

fn main() {
    sleep_for(-10.0);
    sleep_for(0.8);
}
```

[illegible] `let else`

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `let else` [illegible] [illegible]. [illegible] [illegible] [illegible] ”else” [illegible] [illegible] [illegible] ([illegible] break [illegible]return[illegible] [illegible] panic - [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]).

```
fn hex_or_die_trying(maybe_string: Option<String>) -> Result<u32, String> {
    if let Some(s) = maybe_string {
        if let Some(first_byte_char) = s.chars().next() {
            if let Some(digit) = first_byte_char.to_digit(16) {
                Ok(digit)
            } else {
                return Err(String::from("[illegible] [illegible] hex digit"));
            }
        } else {
            return Err(String::from("[illegible] string [illegible] [illegible] [illegible]"));
        }
    } else {
        return Err(String::from("[illegible]"));
    }
}

fn main() {
    println!("[illegible]: {:?}", hex_or_die_trying(Some(String::from("foo"))));
}
```

[illegible] `if let`[illegible] [illegible] [illegible] `while let` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] ([illegible]

:□□□□□□ □□□□□□ (□□□□□□□□□

```
                                                    } ()fn main
                ;("🦀 let mut name = String::from("Comprehensive Rust
                                   } ()while let Some(c) = name.pop
                                    ;("{println!("character: {c
                                                                 {
        (!There are more efficient ways to reverse a string) //
                                                                   {
```

□□ □ □□□□□□□□□□□□ □□ (Some(c □□□□□ □□□□□ □□□□□ □□□□□ □□□ □□□□□□ □□□ String::pop □□□□□□ □□□ □□□□ □□□ □□□ □□□□□□ □□□ □□□□□□ □□□□ □□□ □□□ while let □□□ □□□□□□□□□ .□□□□□□□□□□ □□□□ □□□ None □□ □□□ .□□□□□ □□□□□ □□□□□□ □□□□ □□□□□ □□□ □□□□□□

if-let

- □□□□□□□□□□ □□□□□ □□□□□□ □□□□□ □□ □□□□□ □□□□ □□□□□ □□□□ if let □□□□□□ match □□□□□□ □□□□□□□ .□□□□□ □□□□□□□□□ match □□□□□□ □□ □□□□□ □□□□□ □□ □□□□□ □□□□ □□□□□□□□□□ □□□□ .□□□□□□□
- Option □□ □□□□ □□□□□□ Some □□□□□□□ □□ □□□□□□□ □if let □□□□□□ □□ □□□□□ □□□□□□□□□ □□ .□□□
- □□ if let □□□□□□ match □□□□□□ □□□□□□□

  <=

  .□□□□□□□ □□□□□□□□□ □□□□□ □□□□□□ □□□□□

let-else

□□□□ □□□□□ □□□□□ □□□□□□ □□ □□ □□□□□□□□□ □□□□□□ □□□□□□□□ □□ □□ □□ □□□□□ □□ □□□□□□□□□ □□ if-let □□ □□□□□□□ □□□□□□ .□□□□□ □□□□□□□□□□ □□ □□ □□ □□□□□□ □□□□ □□□□□ □□□□ □□ let-else □□□□□□□ .□□□□ .□□□□□ □□□□□□□ □□ □□ □□□□□□ □□□□□□□□ □□ □□□□□ □□□□□□□□□□ □□□□□□□□□□□ □□□□□

:□□□□ □□□□ □□□□□ □□ □□□□ □□□□□□□□□□ □□□□□□

```
} <fn hex_or_die_trying(maybe_string: Option<String>) -> Result<u32, String
                                        } let Some(s) = maybe_string else
                                  ;(("□□□□□□□□□")return Err(String::from
                                                                      ;{

                 } let Some(first_byte_char) = s.chars().next() else
         ;(("□□□□□ □□□□□□□ □□□□□ string □□")return Err(String::from
                                                                      ;{

                } let Some(digit) = first_byte_char.to_digit(16) else
                         ;(("hex digit □□ □□")return Err(String::from
                                                                      ;{

                                                    ;(return Ok(digit
                                                                        {
```

while-let

- [illegible] while let [illegible] ([illegible]) [illegible].
- [illegible] while let [illegible] if [illegible] (unwrap) [illegible] `name.pop()` [illegible] `while let` [illegible] Syntactic sugar [illegible].

12.5 [illegible]

[illegible].

[illegible] Box [illegible] `Box::new` "[illegible]" [illegible] deref (*) [illegible] "[illegible]" [illegible]: `eval(*boxed_expr)`.

[illegible] `Result<Value, String>` [illegible] enum [illegible] `Ok(Value)` [illegible] `Err(String)` [illegible].

[illegible] Rust Playground [illegible] `eval` [illegible] `todo!()` [illegible] `#[ignore]` [illegible]:

```
[test]#
[ignore]#
{ .. } ()fn test_value
```

[illegible] `Result` [illegible] panic [illegible]

```
.An operation to perform on two subexpressions ///
} enum Operation
,Add
,Sub
,Mul
,Div
{

.An expression, in tree form ///
} enum Expression
.An operation on two subexpressions ///
,{ <Op { op: Operation, left: Box<Expression>, right: Box<Expression

A literal value ///
,(Value(i64
{

} <fn eval(e: Expression) -> Result<i64, String
()!todo
{
                                    } ()fn test_value
;((assert_eq!(eval(Expression::Value(19)), Ok(19
                                                    {

                                      } ()fn test_sum
                                      )!assert_eq
                        } eval(Expression::Op
                      ,op: Operation::Add
   ,((left: Box::new(Expression::Value(10
  ,((right: Box::new(Expression::Value(20
                                          ,({
                                       (Ok(30
                                               ;(
                                                    {

                                } ()fn test_recursion
                     } let term1 = Expression::Op
                          ,op: Operation::Mul
       ,((left: Box::new(Expression::Value(10
       ,((right: Box::new(Expression::Value(9
                                               ;{
                     } let term2 = Expression::Op
                          ,op: Operation::Mul
              } left: Box::new(Expression::Op
                      ,op: Operation::Sub
    ,((left: Box::new(Expression::Value(3
   ,((right: Box::new(Expression::Value(4
                                          ,({
       ,((right: Box::new(Expression::Value(5
                                               ;{
                                      )!assert_eq
                        } eval(Expression::Op
                      ,op: Operation::Add
                   ,(left: Box::new(term1
                  ,(right: Box::new(term2
                                          ,({
                                       (Ok(85
                                               ;(
                                                    {

                                    } ()fn test_error
                                      )!assert_eq
                        } eval(Expression::Op
                      ,op: Operation::Div
   ,((left: Box::new(Expression::Value(99
   ,((right: Box::new(Expression::Value(0
                                          ,({
           (("□□□ □□ □□□□□□")Err(String::from
                                               ;(
                                                    {
```

□□□□□ 12.5.1

```
/// An operation to perform on two subexpressions.
enum Operation {
    Add,
    Sub,
    Mul,
    Div,
}

/// An expression, in tree form.
enum Expression {
    /// An operation on two subexpressions.
    Op { op: Operation, left: Box<Expression>, right: Box<Expression> },

    /// A literal value
    Value(i64),
}

fn eval(e: Expression) -> Result<i64, String> {
    match e {
        Expression::Op { op, left, right } => {
            let left = match eval(*left) {
                Ok(v) => v,
                e @ Err(_) => return e,
            };
            let right = match eval(*right) {
                Ok(v) => v,
                e @ Err(_) => return e,
            };
            Ok(match op {
                Operation::Add => left + right,
                Operation::Sub => left - right,
                Operation::Mul => left * right,
                Operation::Div => {
                    if right == 0 {
                        return Err(String::from("□□□ □□ □□□□□□"));
                    } else {
                        left / right
                    }
                }
            })
        }
        Expression::Value(v) => Ok(v),
    }
}

fn test_value() {
    assert_eq!(eval(Expression::Value(19)), Ok(19));
}
```

```
                                    } ()fn test_sum
                                    )!assert_eq
                      } eval(Expression::Op
                    ,op: Operation::Add
 ,((left: Box::new(Expression::Value(10
,((right: Box::new(Expression::Value(20
                                        ,({
                                     (Ok(30
                                             ;(
                                                  {

                              } ()fn test_recursion
                   } let term1 = Expression::Op
                        ,op: Operation::Mul
     ,((left: Box::new(Expression::Value(10
     ,((right: Box::new(Expression::Value(9
                                             ;{
                   } let term2 = Expression::Op
                        ,op: Operation::Mul
            } left: Box::new(Expression::Op
                    ,op: Operation::Sub
  ,((left: Box::new(Expression::Value(3
 ,((right: Box::new(Expression::Value(4
                                        ,({
     ,((right: Box::new(Expression::Value(5
                                             ;{
                                    )!assert_eq
                      } eval(Expression::Op
                    ,op: Operation::Add
                 ,(left: Box::new(term1
                ,(right: Box::new(term2
                                        ,({
                                     (Ok(85
                                             ;(
                                                  {

                                  } ()fn test_error
                                    )!assert_eq
                      } eval(Expression::Op
                    ,op: Operation::Div
 ,((left: Box::new(Expression::Value(99
 ,((right: Box::new(Expression::Value(0
                                        ,({
          (("□□□ □□ □□□□□□")Err(String::from
                                             ;(
                                                  {

                                        } ()fn main
                    } let expr = Expression::Op
                        ,op: Operation::Sub
     ,((left: Box::new(Expression::Value(20
,((right: Box::new(Expression::Value(10
                                         ;{
              ;(println!("expr: {:?}", expr
       ;((eval(expr ,"{?:} :□□□□□□")!println
                                              {
```

13 [illegible]

[illegible] [illegible] [illegible]

[illegible]:

[illegible]	[illegible]
[illegible]	[illegible]
Traits	[illegible]
Deriving	[illegible]
Generic [illegible]: [illegible]	[illegible]

13.1 [illegible]

Rust [illegible]. [illegible] `impl` [illegible]:

```rust
struct Race {
    name: String,
    laps: Vec<i32>,
}

impl Race {
    // No receiver, a static method
    fn new(name: &str) -> Self {
        Self { name: String::from(name), laps: Vec::new() }
    }

    // Exclusive borrowed read-write access to self
    fn add_lap(&mut self, lap: i32) {
        self.laps.push(lap);
    }

    // Shared and read-only borrowed access to self
    fn print_laps(&self) {
        println!("{} [illegible] [illegible] {}:", self.name, self.laps.len());
        for (idx, lap) in self.laps.iter().enumerate() {
```

```
;("println!("Lap {idx}: {lap} sec
{
{

Exclusive ownership of self //
} (fn finish(self
;()let total: i32 = self.laps.iter().sum
;(self.name, total ,"{} :[illegible] {} [illegible]")!println
{
{

} ()fn main
;("[illegible]")let mut race = Race::new
;(race.add_lap(70
;(race.add_lap(68
;()race.print_laps
;(race.add_lap(71
;()race.print_laps
;()race.finish
;(race.add_lap(42 //
{
```

[illegible] self ”[illegible]” [illegible] - [illegible]. [illegible]:

- &self: [illegible]. [illegible].
- &mut self: [illegible]. [illegible].
- self: [illegible]. [illegible]. [illegible].
- mut self: [illegible].
- [illegible]: [illegible]. [illegible] new [illegible].

[illegible]:

- [illegible].
  - [illegible] (struct [illegible] enum) [illegible] self.
  - [illegible]. [illegible].
- [illegible] self[illegible].
  - [illegible] self [illegible] Self: self [illegible] struct [illegible].
  - [illegible] Self [illegible] impl [illegible].
  - Note how self is used like other structs and dot notation can be used to refer to individual fields.
  - [illegible] &self [illegible] self [illegible]

□□□□□ □□□□□ □□□□□□ □□□ `finish` □□ □□□.

– □□□□□□ □□ □□□□□□□□□ □□□□□□ `self`□ □□□□□□□□□ special wrapper types □□□ □□□□□ □□□□□□ □□ □□□□□□□□□ □□□□□□□□□ □□□□□□□ □□□□□ □□□□□□□ □□□□□□ `Box<Self>`.

13.2 Traits

□□□□□ □□ □□□□ □□□□ □□□□□□ □□ □□□□□□ □□ □□ □□□□□□□□□ □□ traits □□ □□□□ □□□□□□□□ □□□□□□□ □□□□□□ □□□□□. □□□□□ □□□□□□ interface □□ □□□□□□:

```
trait Pet {
    /// Return a sentence from this pet.
    fn talk(&self) -> String;

    /// Print a string to the terminal greeting this pet.
    fn greet(&self);
}
```

- □□ trait □□□□□□□□□ □□ □□□□□□ □□ □□□□□□ □□□□□□ □□ □□□□□□□ □□□□□ □□□□□ □□ □□□□□□ □□□□□□ □□ □□□□□□□□□ □□ trait □□ □□□□□□□□□□ □□□□□.
- □□ □□□ ”Generics”□ □□ □□□□□□ □□□□□□□ □□□□ □□ □□□□□□ □□□□□□□□□□ □□□□□□□□□ □□□□□□□ □□ generic □□ □□□□ □□□□□ □□□□□□□□ □□ □□ trait □□ □□□□□□□□□□□ □□□□□□□□ □□□□□.

13.2.1 □□□□□□ □□□□□ Traits

```
trait Pet {
    fn talk(&self) -> String;

    fn greet(&self) {
        println!("□□□ □□ □□□□□ □□□□□ □□□□□! □□□□□ □□□□□ {}", self.talk());
    }
}

struct Dog {
    name: String,
    age: i8,
}

impl Pet for Dog {
    fn talk(&self) -> String {
        format!("Woof □□□ □□ {} □□□!", self.name)
    }
}

fn main() {
    let fido = Dog { name: String::from("Fido"), age: 5 };
    fido.greet();
}
```

- □□□□□ □□□□□□□□□□ `Trait` □□□□ `Type`□ □□ □□□□ `impl Trait for Type { .. }` □□□□□□□□ □□□□□□□.

- [illegible] Go [illegible]: Cat [illegible] talk() [illegible] Pet [illegible] impl Pet [illegible].

- Traits [illegible]. [illegible]- [illegible] trait [illegible]. [illegible] greet [illegible] talk [illegible].

13.2.2 Supertraits

[illegible] trait [illegible] traits [illegible] *supertraits* [illegible]. [illegible] Pet [illegible] Animal [illegible].

```rust
trait Animal {
    fn leg_count(&self) -> u32;
}

trait Pet: Animal {
    fn name(&self) -> String;
}

struct Dog(String);

impl Animal for Dog {
    fn leg_count(&self) -> u32 {
        4
    }
}

impl Pet for Dog {
    fn name(&self) -> String {
        self.0.clone()
    }
}

fn main() {
    let puppy = Dog(String::from("Rex"));
    println!("{} has {} legs", puppy.name(), puppy.leg_count());
}
```

[illegible] ”trait inheritance” [illegible] (OO) [illegible]. [illegible] trait [illegible].

13.2.3 [illegible]

[illegible] trait [illegible].

```rust
struct Meters(i32);
struct MetersSquared(i32);
```

```
trait Multiply {
    type Output;
    fn multiply(&self, other: &Self) -> Self::Output;
}

impl Multiply for Meters {
    type Output = MetersSquared;
    fn multiply(&self, other: &Self) -> Self::Output {
        MetersSquared(self.0 * other.0)
    }
}

fn main() {
    println!("{:?}", Meters(10).multiply(&Meters(20)));
}
```

- [illegible] ”[illegible]” [illegible].
- [illegible] trait [illegible] Iterator [illegible].

13.3 Deriving

Trait [illegible]:

```
struct Player {
    name: String,
    strength: u8,
    hit_points: u8,
}

fn main() {
    let p1 = Player::default(); // Default trait adds `default` constructor.
    let mut p2 = p1.clone(); // Clone trait adds `clone` method.
    p2.name = String::from("dog");
    // Debug trait adds support for printing with `{:?}`.
    println!("{:?} vs. {:?}", p1, p2);
}
```

[illegible] (Derivation) [illegible] crate [illegible] serde [illegible] `#[derive(Serialize)]` [illegible].

13.4 [illegible]: Trait Logger

[illegible] trait [illegible] Logger [illegible] log [illegible] `&impl Logger` [illegible].

[illegible].

[illegible] StderrLogger [illegible]. [illegible] VerbosityFilter [illegible].

[illegible]: [illegible] trait [illegible] trait [illegible]. [illegible]

```rust
use std::fmt::Display;

pub trait Logger {
    /// Log a message at the given verbosity level.
    fn log(&self, verbosity: u8, message: impl Display);
}

struct StderrLogger;

impl Logger for StderrLogger {
    fn log(&self, verbosity: u8, message: impl Display) {
        eprintln!("[illegible] [illegible]={verbosity}: {message}");
    }
}

fn do_things(logger: &impl Logger) {
    logger.log(5, "FYI");
    logger.log(2, "[illegible]");
}

// TODO: Define and implement `VerbosityFilter`.

fn main() {
    let l = VerbosityFilter { max_verbosity: 3, inner: StderrLogger };
    do_things(&l);
}
```

13.4.1 [illegible]

```rust
use std::fmt::Display;

pub trait Logger {
    /// Log a message at the given verbosity level.
    fn log(&self, verbosity: u8, message: impl Display);
}

struct StderrLogger;

impl Logger for StderrLogger {
    fn log(&self, verbosity: u8, message: impl Display) {
        eprintln!("[illegible] [illegible]={verbosity}: {message}");
```

```
                                                                 {
                                                                     {

                                  } (fn do_things(logger: &impl Logger
                                             ;("logger.log(5, "FYI
                                            ;("□□□□" ,logger.log(2
                                                                     {

                .Only log messages up to the given verbosity level ///
                                              } struct VerbosityFilter
                                                ,max_verbosity: u8
                                              ,inner: StderrLogger
                                                                     {

                                     } impl Logger for VerbosityFilter
             } (fn log(&self, verbosity: u8, message: impl Display
                          } if verbosity <= self.max_verbosity
                       ;(self.inner.log(verbosity, message
                                                             {
                                                                 {
                                                                     {

                                                           } ()fn main
;{ let l = VerbosityFilter { max_verbosity: 3, inner: StderrLogger
                                                    ;(do_things(&l
                                                                     {
```

IV □□□

□□□ :□□□ □□□

14 [illegible]

[illegible] [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible]:

[illegible]	[illegible] [illegible]
Generics	[illegible] [illegible]
[illegible] [illegible] [illegible]	[illegible] [illegible]
[illegible] [illegible] Traits	[illegible] [illegible] [illegible] [illegible] [illegible]

15 [illegible]

Generics

:[illegible] [illegible] .[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

[illegible]	[illegible] [illegible]
Generic [illegible]	[illegible] [illegible]
Generic [illegible] [illegible]	[illegible] [illegible]
Trait Bounds	[illegible] [illegible]
impl Trait	[illegible] [illegible]
dyn Trait	[illegible] [illegible]
Generic min :[illegible]	[illegible] [illegible]

15.1 Generic [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] generics [illegible] Rust [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] (([illegible] [illegible] [illegible] [illegible] [illegible]) [illegible].

```
.`Pick `even` or `odd` depending on the value of `n ///
} fn pick<T>(n: i32, even: T, odd: T) -> T
} if n % 2 == 0
even
} else {
odd
{
{

} ()fn main
;((pick(97, 222, 333 ,"{?:} :[illegible] [illegible] [illegible] [illegible] [illegible]")!println
;(((2 ,"[illegible]") ,(1 ,"[illegible]") ,pick(28 ,"{?:} :[illegible] [illegible] [illegible] [illegible]")!println
{
```

- .[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] T [illegible] Rust
- [illegible] [illegible] [illegible] [illegible] generic [illegible] Rust [illegible] [illegible] C++ [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

[illegible]. [illegible] `pick` [illegible] `n == 0` [illegible] `even + odd` [illegible]. [illegible] `pick` [illegible] Rust [illegible]. [illegible] C++ [illegible].

- [illegible] generic [illegible] non-generic [illegible]. [illegible]: [illegible].

15.2 [illegible] Generic

[illegible] generic [illegible]:

```
struct Point<T> {
    x: T,
    y: T,
}

impl<T> Point<T> {
    fn coords(&self) -> (&T, &T) {
        (&self.x, &self.y)
    }

    fn set_x(&mut self, x: T) {
        self.x = x;
    }
}

fn main() {
    let integer = Point { x: 5, y: 10 };
    let float = Point { x: 1.0, y: 4.0 };
    println!("{integer:?} [illegible] {float:?}");
    println!("coords: {:?}", integer.coords());
}
```

- [illegible]: `T` [illegible] `impl<T> Point<T>{}` [illegible]
  - [illegible] generic [illegible] generic [illegible]. [illegible] generic [illegible].
  - [illegible] `T` [illegible].
  - [illegible] `impl Point<u32>{ .. }` [illegible].
    * `Point` [illegible] generic [illegible] `Point<f64>` [illegible] `Point<u32>` [illegible].

- [illegible] `let p = Point { x: 5, y: 10.0 };` [illegible]. [illegible] `T` [illegible] `U`.

Generic Traits 15.3

[illegible] trait [illegible] generic [illegible] Traits
.[illegible]

```
;(struct Foo(String

 } impl From<u32> for Foo
  } fn from(from: u32) -> Foo
(("{integer: {from [illegible]")!Foo(format
                                       {
                                          {

 } impl From<bool> for Foo
  } fn from(from: bool) -> Foo
(("{bool: {from [illegible]")!Foo(format
                                       {
                                          {

                         } ()fn main
;(let from_int = Foo::from(123
;(let from_bool = Foo::from(true
;("{?:println!("{from_int:?}, {from_bool
                                          {
```

- [illegible] std [illegible] From trait
.[illegible]
- [illegible] trait [illegible] .[illegible]
Foo [illegible] From<&str> [illegible] ("Foo::from("hello
.[illegible]
- [illegible] ”[illegible]” [illegible] Generic [illegible]trait
[illegible] trait [illegible] .[illegible] ”[illegible]” [illegible]
.[illegible]
- [illegible] T [illegible] trait [illegible] Rust [illegible]
”[illegible]” [illegible] Rust [illegible] .[illegible]
[illegible] .[illegible]
.[illegible]

Trait Bounds 15.4

[illegible] trait [illegible] generic[illegible]
.[illegible] trait [illegible]

:You can do this with `T: Trait`

```
} (fn duplicate<T: Clone>(a: T) -> (T, T
       (()a.clone(), a.clone)
                              {

;struct NotClonable //
```

```
} ()fn main
;("let foo = String::from("foo
;(let pair = duplicate(foo
;("{?:println!("{pair
{
```

- [illegible] NonClonable [illegible] duplicate [illegible].
- [illegible] + [illegible].
- [illegible] where [illegible].

```
(fn duplicate<T>(a: T) -> (T, T
where
,T: Clone
}
(()a.clone(), a.clone)
{
```

  - [illegible] where [illegible].
  - [illegible].
    - [illegible] : [illegible] Option<T> [illegible].

- [illegible] Rust [illegible]. [illegible] duplicate [illegible] duplicate(a: u32) [illegible].

15.5 impl Trait

[illegible] trait [illegible] syntax impl Trait [illegible]:

```
:Syntactic sugar for //
} fn add_42_millions<T: Into<i32>>(x: T) -> i32   //
} fn add_42_millions(x: impl Into<i32>) -> i32
x.into() + 42_000_000
{

} fn pair_of(x: u32) -> impl std::fmt::Debug
(x + 1, x - 1)
{

} ()fn main
;(let many = add_42_millions(42_i8
;("{println!("{many
;(let many_more = add_42_millions(10_000_000
;("{println!("{many_more
;(let debuggable = pair_of(27
    println!("□□□□□ □□□□□□: {debuggable:?}");
}
```

`impl Trait` □□ □□□□□□ □□□□□□ □□ □□□□□□□□□ □□□□ □□□□□ □□ □□□□□□□□□□ □□□□ □□□□□□. □□□□□ `impl Trait` □□ □□□□□□□□□□ □□□□□□ □□□□ □□□□□□□ □□□□.

- □□□□□ □□ □□□□□□□□□ `impl Trait` □□□□□ □□ □□ □□□□□□□□ generic □□□□□□□□□ □□ □□ □□□□□□□□ trait □□□□.
- □□□□□ □□□□□ □□□□□□□□□□ □□ □□□□ □□□□□□□□ □□□□□ □□□□□□□□□ □□□□□ □□□□□□ □□□□ □□ trait □□ □□□□□□□□□□ □□□□□□□□ □□□□□ □□□□□□ □□□□□ □□ □□□□ □□□□□□. □□□□ □□□□□□□□□ □□□□□□ □□□□□ □□□□□ □□ □□□□□□□□□□□□ □□□□□ □□□□□ □□ □□ □□ API □□□□□□ □□□□□.

Inference □□ □□□□□□□ □□□□□□□□□ □□□□□□ □□□□. □□□□□□ □□ `impl Foo` □□ □□□□□□□□□□□□□ □□□□□ □□□□□□ □□ □□ □□□□□□□□□□□ □□□□□□□□ □□□□□□□□ □□□□□ □□□□□□ □□ □□ □□ □□□□ □□□□□ □□ □□□□□ □□□□□□□□. □□□□□□ □□ □□□□□ generic □□□□□□ `collect<B>() -> B` □□ □□□□□□□□□□□□ □□□□□□□□ □□ `B` □□ □□□□□ □□ □□□□□□□□ □□□□□□ □□□□□□□□□□□ □ □□□□□ □□□□ □□□□□□□□□□□□□□□□ □□□□□ □□ □□□□□□□ □□□□ □□ □□□□□ □□□□□□ □□□□□□ □□□□□□ `let x: Vec<_> = foo.collect()` □□ □□ □□□□□□□□ □□ `foo.collect::<Vec<_>>()` □turbofish.

□□□□ `debuggable` □□□□□□ □□□□ □□□□□ ‘let debuggable: () = .. □□ □□□□□□□ □□□□□ □□ □□□□□□□ □□□□□ □□□□ □□ □□□□□ □□ □□□□□ □□□□□□.

15.6 dyn Trait

□□□□□□ □□ □□□□□□□□ □□ □□□□□□□ □□□□□ □□□□□□□□□ □□□□□□□□ □□ □□□□□ generic□□□ Rust □□□□□□□ □□ □□□□□□□□ □□ □□□□□ □□□□□□□□□ □□□□□□□□□ □□ □□□□□□□□ □□ □□□□□ □□□□□□ trait □□□□□□□□□ □□□□□□:

```
struct Dog {
    name: String,
    age: i8,
}
struct Cat {
    lives: i8,
}

trait Pet {
    fn talk(&self) -> String;
}

impl Pet for Dog {
    fn talk(&self) -> String {
        format!("Woof □□□ □□ {} □□□!", self.name)
    }
}

impl Pet for Cat {
    fn talk(&self) -> String {
        String::from("Miau!")
    }
}
```

```rust
// Uses generics and static dispatch.
fn generic(pet: &impl Pet) {
    println!("□□□□□□ □□□□ □□ □□□□□□□ {}", pet.talk());
}

// Uses type-erasure and dynamic dispatch.
fn dynamic(pet: &dyn Pet) {
    println!("□□□□□□ □□□□ □□ □□□□□□□ {}", pet.talk());
}

fn main() {
    let cat = Cat { lives: 9 };
    let dog = Dog { name: String::from("Fido"), age: 5 };

    generic(&cat);
    generic(&dog);

    dynamic(&cat);
    dynamic(&dog);
}
```

- □□□Generic □□ □□□□□ `impl Trait`□ □□ monomorphization □□□□□ □□□□□□ □□ □□□□□□ □□□□□□ □□ □□□□□ □□□□□ □□ □□□□□□□ □□ □□ □□ □□□□□□□□□□ □□□□ □□□□□□□□ □□□□□□□. □□□□ □□□□□ □□□□□□□ □□ □□□□□□□□□ □□ □□□ trait □□ □□□□□ □□ □□□□ generic □□□□□□□ □□ □□□□□□□□□ □□□□□□□□ □□□□□□□□ □□□□□□□ □□□□□ □□□□□□□□□ □□□□□□□□□ □□□□□ □□□□□ □□ □□□□□ □ □□□□□□□□□ □□□□□□□□□□ trait □□□□□□ □□ □□□□□ □□ □□□□□ □□□□.
- □□□□□□ □□ □□ `dyn Trait` □□□□□□□□ □□□□□□□ □□□□□□ □□ □□ □□□□□□□□□ □□□□□□□□□□ □□ □□□□□ □□ vtable (virtual method table) □□□□□□□□ □□□□□□. □□□□ □□□□□□ □□□□□□□ □□ □□ □□□□ □□□□□ □□ `fn dynamic` □□□□□ □□□□□ □□ □□□□□ □□□□□ □□ □□□□□ `Pet` □□ □□□□□ □□□□□□□ □□□□□□□□ □□□□□□.
- □□□□□□ □□ □□ `dyn Trait` □□□□□□□□ □□□□□□□ □□ trait □□□□□ □□□□ □□ □□□□□ □□□□□ □□□□□ □□□□□□. □□ □□ □□□□□□ □□□□ □□□□□ □□□□ □□ □□□□□□ □□□□□ □□□□□□ □□□□□□□ □□□□□□□□□□□ □□□□□□□ □□□□□□ `Box` □□□□ □□□□□□□□□□ □□□□□□□□□ □□□□□ (□□□□ □□□□□□ □□ □□□□ □□□□ □□□□□ □□□□□ □□□□□□ □□).
- □□ □□□□□ □□□□□□ □□ `&dyn Pet` □□□□□□□□ ”□□□□□□□□□ □□□□” (fat pointer) □□□□□□□ □□□□□□□ □□□□□ □□ □□□ □□ □□ □□□□□□□□: □□□□ □□ □□□□□□□□□□ □□ □□□ □□□□□□ □□ `Pet` □□ □□□□□□□□□□ □□□□□□ □□□□□□ □□□□□ □ □□□□□□ □□ vtable □□□□□ □□□□□□□□□□ □□□□□ □□ □□□□ □□□□□□ □□□□□□. □□□□□□ □□□□□□□□□ `talk` □□□□ □□ `&dyn Pet`□ □□□□□□□□□□ □□□□□ □□□□□ `talk` □□ □□ vtable □□□□□□ □□□□□ □ □□□□ □□□□□ □□ □□□□□□□□□ □ □□□□□□□□ □□ `Dog` □□ `Cat` □□ □□ □□ □□□□□ □□□□. □□□□□□□□□□ □□□□□□ □□ □□□□□□□ □□□□□ □□□□□ `Pet` □□□□□□ □□□□ □□□□ □□□□□□.
- □□ `dyn Trait` □□□□□□□□ ”□□□□□ □□□□ □□□□” (type-erased) □□ □□□□ □□□□□□ □□□□□□□ □□□□□ □□□□□ □□ □□□□□ □□□□□□□□ □□□□□□□□□□ □□□□□□□ □□□□□ □□□□□ □□□□□□□□.

15.7 □□□□□□: Generic `min`

□□ □□□□ □□□□□□□ □□□□□□ □□□□ □□ □□□□□ generic `min`□□ □□□□□□□□□□□ □□□□□□□ □□□□ □□ □□□□□□ □□ □□ □□ □□□□□□ □□□□□□□ □□□□□□□ □□ □□□□□□□□□ □□ `Ord` trait.

```rust
use std::cmp::Ordering;

// TODO: implement the `min` function used in `main`.
```

```
} ()fn main
;(assert_eq!(min(0, 10), 0
;(assert_eq!(min(500, 123), 123

;('assert_eq!(min('a', 'z'), 'a
;('assert_eq!(min('7', '1'), '1

;("assert_eq!(min("hello", "goodbye"), "goodbye
;("assert_eq!(min("bat", "armadillo"), "armadillo
{
```

- `Ord` trait [illegible] `Ordering` enum [illegible] [illegible] [illegible] [illegible] [illegible].

15.7.1 [illegible]

```
;use std::cmp::Ordering

} fn min<T: Ord>(l: T, r: T) -> T
} (match l.cmp(&r
,Ordering::Less | Ordering::Equal => l
,Ordering::Greater => r
{
{

} ()fn main
;(assert_eq!(min(0, 10), 0
;(assert_eq!(min(500, 123), 123

;('assert_eq!(min('a', 'z'), 'a
;('assert_eq!(min('7', '1'), '1

;("assert_eq!(min("hello", "goodbye"), "goodbye
;("assert_eq!(min("bat", "armadillo"), "armadillo
{
```

16 [illegible]

[illegible]

[illegible]:

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
Option	[illegible]
Result	[illegible]
String	[illegible]
Vec	[illegible]
HashMap	[illegible]
[illegible]: [illegible]	[illegible]

[illegible].

16.1 [illegible]

Rust [illegible] Rust [illegible] `String` [illegible].

[illegible] Rust [illegible]: `alloc`, `core` [illegible] `std`.

- `core` [illegible] `libc` [illegible].
- `alloc` [illegible] `Box`, `Vec` [illegible] `Arc`.
- [illegible] Rust [illegible] `core` [illegible] `alloc` [illegible].

16.2 [illegible]

Rust [illegible]:

- .□□□□□□□ □□ □□□□□□ □□□□□□□ □□□□□
- .u8 □□□□□□ □□□□□□□□ □□□□□□□□□
- .BinaryHeap □□ Option □□□□□□ □□□□□□□□□□□ □□□□□□□□□ □□□□□□□□

:□□□□□ □□□□□□ □□ □□□□ □□ □□□□□□□□□□ □□□□ □□□□□□ □□

```
.Determine whether the first argument is divisible by the second argument ///
///
.If the second argument is zero, the result is false ///
} fn is_divisible_by(lhs: u32, rhs: u32) -> bool
    } if rhs == 0
        ;return false
    {
    lhs % rhs == 0
{
```

-□□ Rust □□□□□□□□□□ □□□□□□□□□□□□ □□□crate □□□□□ .□□□□□□□ □□□□□□□ Markdown □□□□□□□□□ □□□□□□□□□ □□□□□ □□□□□ □□□□□□□ .□□□□□□□ □□□□□□ rustdoc □□□□□□ □□ □□□□□□□□□ □□ docs.rs □□ □□□□□□□ □□□□ .□□□□ □□□□□□ □□□□□ □□□□□□□ □□□□□ □□□□ □□ □□□□□□□□□ □□ API □□ □□ □□□□□□ □□□□□□□□□

`/*! .. */` □□ `//!` □□ □(□□□□□□ □□ □□□□□ □□□□□□) □□□□□ □□□□ □□□□□ □□ □□□□□ □□ □□□□□ □□□□□□ □□□□□ :□□□□□□□□□ ”□□□□□□ □□□□□□□□ □□□□□□□□□□□” □□ □□ □□ □□□□□ □□□□□□□□□

```
.This module contains functionality relating to divisibility of integers !//
```

- □□□□□ □□□□□□□□□□□□ □□ https://docs.rs/rand □□ □□ rand crate □□□□□ □□□□□□□□□□ □□□□□□□□□ .□□□□□

Option 16.3

□□ □□□□□□□□ □□ □□□□□ □□□□ .□□□□□□□□□ □□□□□□□ □□ Option<T> □□ □□□□□□□□□□□ □□□□□ □□□□□□ □□ □□ String::find □□□□□□ □□□□□□ □□ .□□□□□□□□ □□□□□□□ □□ □□□□□ □□□□ □□ □□□□□□ □□□□□□ □□ T□□□□□ .□□□□□□□□□□□ □□ Option<usize>

```
} ()fn main
    ;"let name = "Löwe 老虎 Léopard Gepardi
    ;('let mut position: Option<usize> = name.find('é
    ;("{?:position}□□□□□□□□ □□□ □□□□ □□□□")!println
    ;(assert_eq!(position.unwrap(), 14
    ;('position = name.find('Z
    ;("{?:position}□□□□□□□□ □□□ □□□□ □□□□")!println
    ;(assert_eq!(position.expect("Character not found"), 0
{
```

- -□□□□ □□□□□□ □□□□□□□□□□□ □□□□□□□□□ □□ □□□□□ □ □□□□□□ □□□□□□□□ □□□□□□□□□ □□□□□□ Option .□□□
- □□□□□□ expect .□□□□□□ panic □□□□ □□ □□□□□□□□□□□ □□ Option □□ □□ □□□□□□ □□□□□□ unwrap .□□□□□□□□ □□□□ □□□□□ □□□□□□ □□□□ □□□□
  - □□□□□□□ ”□□□□□□□” □□□□□□ □□□□□□□□□□ □□□□ □□□□□□ panic None □□ □□□□□□□ □□ □□□□□□□□□ .□□□□□ □□□□□□ None □□ □□□□□
  - □□ □□□□ □□□□□ □□□□□ □□□□□ □□□□□ □□□□□ □□□□□□ □□□□□□ □□ unwrap/expect □□ □□□□□□□□ .□□□□□□ □□□□□□□ □□□□□□□□ □□□□□□□□□ □□ None □□□□□□□□ □□□□□□□
- □□□□□□ □□ T □□ □□□□□□ □□□□□□□□□ □□□□□ Option<T> □□ □□□□□□□ □□□□ □□ niche □□□□□□□□□□ .□□□□□

16.4 Result

[illegible] Option [illegible] Result
Ok [illegible] T [illegible] Result<T, E> :[illegible] .[illegible] enum [illegible]
.[illegible] Err [illegible] E [illegible]

```
;use std::fs::File
;use std::io::Read

} ()fn main
;("let file: Result<File, std::io::Error> = File::open("diary.txt
} match file
} <= (Ok(mut file
;()let mut contents = String::new
} (if let Ok(bytes) = file.read_to_string(&mut contents
;("(contents} ({bytes} bytes} :[illegible]")!println
} else {
;("[illegible]")!println
{
{
} <= (Err(err
;("{err} :[illegible]")!println
{
{
{
```

- [illegible] Result [illegible] □Option [illegible] [illegible] .[illegible] .[illegible] [illegible] expect() [illegible] unwrap() [illegible] .[illegible]
- .[illegible] .[illegible] Result [illegible] .[illegible]
- [illegible] Result .[illegible]

16.5 String

:[illegible] UTF-8 [illegible] String

```
} ()fn main
;()let mut s1 = String::new
;("[illegible]")s1.push_str
;(()println!("s1: len = {}, capacity = {}", s1.len(), s1.capacity

;(let mut s2 = String::with_capacity(s1.len() + 1
;(s2.push_str(&s1
;('!')s2.push
;(()println!("s2: len = {}, capacity = {}", s2.len(), s2.capacity

;("let s3 = String::from("🇨🇭
    println!("s3: len = {}, number of chars = {}", s3.len(), s3.chars().count());
}
```

`String` [illegible] `Deref<Target = str>` [illegible] `str` [illegible] `String` [illegible].

- `String::new` [illegible]. `String::with_capacity` [illegible].
- `String::len` [illegible] `String` [illegible] ([illegible]).
- `String::chars` [illegible] (iterator) [illegible]. [illegible] `char` [illegible] ”[illegible]“ [illegible] grapheme clusters.
- [illegible] `&str` [illegible] `String`.
- [illegible] `Deref<Target = T>` [illegible] `T` [illegible].
  - [illegible] `Deref` trait [illegible].
  - `String` [illegible] `Deref<Target = str>` [illegible] `str` [illegible].
  - `let s3 = s1.deref();` [illegible] `let s3 = &*s1;` [illegible].
- [illegible] `let s3 = s1.deref();` [illegible] `let s3 = &*s1;`.
- [illegible] `String` [illegible]:
  - [illegible] `s3.chars().nth(i).unwrap()` [illegible] `i` [illegible].
  - [illegible] `s3[0..4]` [illegible].
- [illegible] `to_string` [illegible]. [illegible] `Display` [illegible].

16.6 Vec

`Vec` [illegible] heap-allocated [illegible]:

```
fn main() {
    let mut v1 = Vec::new();
    v1.push(42);
    println!("v1: len = {}, capacity = {}", v1.len(), v1.capacity());

    let mut v2 = Vec::with_capacity(v1.len() + 1);
    v2.extend(v1.iter());
    v2.push(9999);
    println!("v2: len = {}, capacity = {}", v2.len(), v2.capacity());

    // Canonical macro to initialize a vector with elements.
    let mut v3 = vec![0, 0, 1, 2, 3, 4];

    // Retain only the even elements.
    v3.retain(|x| x % 2 == 0);
```

```
;("{?:println!("{v3

.Remove consecutive duplicates //
;()v3.dedup
;("{?:println!("{v3
{
```

[illegible] `<[Deref<Target = [T` [illegible] Vec
.[illegible] Vec [illegible]

- [illegible] heap [illegible] .HashMap [illegible] String [illegible] Vec [illegible] .[illegible]
- T [illegible] generic [illegible] Vec<T> [illegible] .[illegible] Rust [illegible] T [illegible] .[illegible] push
- [illegible] Vec::new() [illegible] vec![...] .[illegible] vector [illegible]
- [illegible] [ ] [illegible] vector [illegible] pop [illegible] .[illegible] Option [illegible] get [illegible] .[illegible] panic .[illegible]
- [illegible] .[illegible] Vec [illegible].

HashMap 16.7

:HashDoS [illegible] hash [illegible]

```
;use std::collections::HashMap

} ()fn main
    ;()let mut page_counts = HashMap::new
    ;(207 ,"[illegible]")page_counts.insert
    ;(751 ,"[illegible]")page_counts.insert
    ;(303 ,"[illegible]")page_counts.insert

    } ("if !page_counts.contains_key("Les Misérables
        )!println
            ,".[illegible] Les Misérables [illegible] {} [illegible]"
            ()page_counts.len
        ;(
    {

    } ["[illegible]" ,"[illegible]"] for book in
        } (match page_counts.get(book
            ,("[illegible] {Some(count) => println!("{book}: {count
            ,(".[illegible] {None => println!("{book
        {
    {

    .Use the .entry() method to insert a value if nothing is found //
```

```
      } ["□□□□□□ □□□□□□□ □□ □□□□□ □□□□□□□□□□□" ,"□□□□□ □ □□□□□"] for book in
;(let page_count: &mut i32 = page_counts.entry(book).or_insert(0
                                                   ;page_count += 1*
                                                                  {

                                           ;("{?#:println!("{page_counts
                                                                      {
```

- .□□□□ □□□□□ scope □□ □□□□□ □ □□□□□ □□□□□□ prelude □□ HashMap
- HashMap □□ □□□□□ □□ □□□□ □□ □□□□□□□ □□□□□□ □□□□ □□□□ .□□□□□ □□□□□□□□ □□ □□□□ □□ □□□□□□□ □□□□□□ □□□□ □□□□ .□□□□□□□□□□□□ □□□□□□□□□ □□□□□□ □□ □□□□□□□□ □□□□□ □□□□ □ □□□□ □□ □□□□□ □□□□□ .□□□□ □□□□□ □□□□□ □□□□ □□□□□□ □□□□□ HashMap □□ □□ □□□□□□□□□

```
                                               let pc1 = page_counts
                                   ("□□□□□ □□□□ □ □□□□□ □□□□")get.
                                                   ;(unwrap_or(&336.
                                               let pc2 = page_counts
                                    ("entry("The Hunger Games.
                                                   ;(or_insert(374.
```

- .□□□□□□ □□□□□ hashmap! □□□□□□□□□□ □□□□□□□ □□□□□□□□□□ □vec! □□□□□□□

  □□□□ `<[From<[(K, V); N` □□□□□□□□□□□□□□□□□□□□ HashMap □□□□□ □□□ Rust 1.56 □□□□□ □□□ –
  :□□□□□ □□□□□□ □□□□□□□□□□ □□□□□□ □□ □□□ □□□ HashMap □□ □□□□□□□□□ □□□□□□ □□□□□□ □□ □□□ □□□

```
                              ])let page_counts = HashMap::from
                 ,(to_string(), 336."□□□□□ □□□□ □ □□□□□ □□□□")
                    ,(The Hunger Games".to_string(), 374")
                                                        ;([
```

- □□□□□□ □□□ key-value □□□□□□□ □□□ Iterator □□ □□□ □□□□□□□□ HashMap □□□□□□□□□□ □□□□□□ .□□□□ □□□□□□ □□□□□□□
- □□□□□ □□□□□□□□□ &str □□ □□□□□□□□□ □□□ □ □□□□□□□□ □□□□□□ □□□ HashMap<String, i32> □□□ □□□□□□ □□□□□□□□□□ □□□ □□□□□□□□□ □□□ □□□□□□□□□ .□□□□□ □□□□□□□□ □□□□□□□□ □□□ □□□□□□□□ □□□□□□□ .□□□□ □□□□□ borrow checker □□□ □□□□□□□□□ □□□ □□□□□□□□□ □□□□ □□□□□ □□□□□

  □□□□ `to_string()` □□□ □□□□□ □□□□□ □□□ □□□□□□□□ □□□□□ □ □□□□□□□□ □□□□ □□□□□ □□□□□□□□□ –
  □□□□□□ □□□□□□ □□□□□□□□□ □□□ □□□ □□□□ □□□□□ □□□□□□□ □□□□. □□□□ □□□ □□□□□□□

- .std::collections::hash_map::Keys □□□□□□ □□□□□□ ”□□□□ □□□□ □□□□□□□□□ □□□□□“ □□□□□ □□□□□□ □□□□ □□□ □□□□□ □□□□ □□□□□□□□□ .□□□□□□□ □□□□□ Rust □□□□□□□□□ □□□□□□□□□□ □□□ □□□□□□□□□ □□□□□□□ □□□□ .□□□□□ □□□□□□ □□□□ □□□ keys □□□□ □□□ □□□□□□□□□ □□□□□ □□□□□□ □ □□□□□ □□□□□ □□□□□□□□□□□□□ □□□

16.8 □□□□□□□□□ :□□□□□□

□□□□□□□ □□□□ .□□□□ □□□□□□□ generic □□□□□□□ □□□ □□□□□ □□□□□□ □□□□□ □□□□□□□ □□□ □□□□ □□□□□□□ □□□□ □□□ □□□□□□□ □ □□□□□□□ □□□□□□□ □□□□□□□□ □□□ □□□□□□ □□□□□□□ □□□□□ `std::collections::HashMap` □□□ .□□□□□□ □□□□□□□□□ □□□□□□□□□ □□□□□ □□□□ □□□□

□□□□□□□ □ □□□□□□□ .□□□□ □□□□ □□□□□□□□□ u32 □□□□□□□ □□□□□ □□□□□□□□□□□ □□□□□□ Counter □□□□□□ □□□□□ □□□ □□□□□□□ □□□□□□ □□□□□□ □□□□□ □□□□□□□ □□□□ □□□ □□□ □□□□□□□ □□□□□ □□□□□ □□□ generic □□□□□□□ □□□ □□□ .□□□□ □□□□□□□ □□□ □□□□□□□ □□□□□ □□□ □□□□□□□ Counter

□□□□□ □□ □□□□□□□□□ □□□□□□ □□ □□□□□ □□□□□□□□ entry □□□□ □□ □□□□□ □□□□ □□□□□□□ □□□□□ □□□□ □□□□
.□□□□□ □□□□□ □□□□ □□ □□ count □□□□ □□□□□□□□□□ □□□□□ □□□□□

```
;use std::collections::HashMap

.Counter counts the number of times each value of type T has been seen ///
} struct Counter
,<values: HashMap<u32, u64
{

} impl Counter
.Create a new Counter ///
} fn new() -> Self
} Counter
,()values: HashMap::new
{
{

.Count an occurrence of the given value ///
} (fn count(&mut self, value: u32
} (if self.values.contains_key(&value
;self.values.get_mut(&value).unwrap() += 1*
} else {
;(self.values.insert(value, 1
{
{

.Return the number of times the given value has been seen ///
} fn times_seen(&self, value: u32) -> u64
()self.values.get(&value).copied().unwrap_or_default
{
{

} ()fn main
;()let mut ctr = Counter::new
;(ctr.count(13
;(ctr.count(14
;(ctr.count(16
;(ctr.count(14
;(ctr.count(14
;(ctr.count(11

} for i in 10..20
;(ctr.times_seen(i), i ,"□□□□□□□ □□ {} □□ □□□□□ {} □□□□□□")!println
{

;()let mut strctr = Counter::new
;("□□□")strctr.count
;("strctr.count("orange
;("□□□")strctr.count
;(("□□□")strctr.times_seen ,"□□□□□ {} □□□□□")!println
```

```
                                                                          {
```

□□□□□ 16.8.1

```
                                             ;use std::collections::HashMap
                                                       ;use std::hash::Hash

 .Counter counts the number of times each value of type T has been seen ///
                                                        } <struct Counter<T
                                               ,<values: HashMap<T, u64
                                                                          {

                                            } <impl<T: Eq + Hash> Counter<T
                                              .Create a new Counter ///
                                                     } fn new() -> Self
                                 { ()Counter { values: HashMap::new
                                                                      {

                            .Count an occurrence of the given value ///
                                        } (fn count(&mut self, value: T
                       ;self.values.entry(value).or_default() += 1*
                                                                      {

          .Return the number of times the given value has been seen ///
                                } fn times_seen(&self, value: T) -> u64
               ()self.values.get(&value).copied().unwrap_or_default
                                                                      {
                                                                          {

                                                                } ()fn main
                                          ;()let mut ctr = Counter::new
                                                         ;(ctr.count(13
                                                         ;(ctr.count(14
                                                         ;(ctr.count(16
                                                         ;(ctr.count(14
                                                         ;(ctr.count(14
                                                         ;(ctr.count(11

                                                      } for i in 10..20
;(ctr.times_seen(i), i ,"□□□□□□□ □□ {} □□ □□□□□ {} □□□□□□")!println
                                                                      {

                                       ;()let mut strctr = Counter::new
                                                   ;("□□□")strctr.count
                                                ;("strctr.count("orange
                                                   ;("□□□")strctr.count
                  ;(("□□□")strctr.times_seen ,"□□□□□ {} □□□□□")!println
                                                                          {
```

17 [illegible]

Traits [illegible] [illegible]

[illegible]:

[illegible]	[illegible] [illegible]
[illegible]	[illegible]
[illegible]	[illegible]
From and Into	[illegible]
Casting	[illegible]
Read and Write	[illegible]
Default, struct update syntax	[illegible]
Closures	[illegible]
ROT13 :[illegible]	[illegible]

[illegible] trait [illegible].

[illegible].

17.1 [illegible]

[illegible] trait [illegible] trait [illegible] trait [illegible].

PartialEq and Eq

`PartialEq` [illegible] `eq` [illegible] `ne` [illegible]. [illegible] `==` [illegible] `!=` [illegible].

```rust
struct Key {
    id: u32,
    metadata: Option<String>,
}
impl PartialEq for Key {
    fn eq(&self, other: &Self) -> bool {
        self.id == other.id
```

```
{
{
```

.[illegible] PartialEq [illegible] (transitive [illegible]) [illegible] Eq
.[illegible] trait bound [illegible] Eq [illegible]

PartialOrd and Ord

[illegible] .[illegible] partial_cmp [illegible] PartialOrd
.[illegible] < [illegible] =< [illegible]=> [illegible]> [illegible]

```
;use std::cmp::Ordering
} struct Citation
,author: String
,year: u32
{
} impl PartialOrd for Citation
} <fn partial_cmp(&self, other: &Self) -> Option<Ordering
} (match self.author.partial_cmp(&other.author
,(Some(Ordering::Equal) => self.year.partial_cmp(&other.year
,author_ord => author_ord
{
{
{
```

.[illegible] Ordering [illegible] cmp [illegible] Ord

[illegible] Eq [illegible] PartialEq
:[illegible]

```
} struct Key
,id: u32
,<metadata: Option<String
{
} impl PartialEq<u32> for Key
} fn eq(&self, other: &u32) -> bool
self.id == *other
{
{
```

[illegible] trait [illegible]
.[illegible]

[illegible] 17.2

:[illegible] `std::ops` [illegible] traits [illegible]

```
} struct Point
,x: i32
,y: i32
{

} impl std::ops::Add for Point
```

```
                                   ;type Output = Self

                        } fn add(self, other: Self) -> Self
         { Self { x: self.x + other.x, y: self.y + other.y
                                                           {
                                                              {

                                                  } ()fn main
                               ;{ let p1 = Point { x: 10, y: 20
                             ;{ let p2 = Point { x: 100, y: 200
                ;(println!("{:?} + {:?} = {:?}", p1, p2, p1 + p2
                                                              {
```

[illegible]:

- Add [illegible] &Point [illegible].
  – [illegible]: Add::add [illegible] self [illegible]. [illegible] T [illegible] Copy [illegible] &T [illegible].
- Output [illegible]
  – [illegible]: [illegible] (Output [illegible]) [illegible] trait [illegible].
- Add [illegible] impl Add<(i32, i32)> for Point [illegible] tuple [illegible] Point [illegible].

The Not trait (`!` operator) is notable because it does not ”boolify” like the same operator in C-family languages; instead, for integer types it negates each bit of the number, which arithmetically is equivalent to subtracting it from -1: `!5 == -6`.

17.3 From and Into

Types implement `From` and `Into` to facilitate type conversions. Unlike `as`, these traits correspond to lossless, infallible conversions.

```
                                                  } ()fn main
                              ;("let s = String::from("hello
                 ;([let addr = std::net::Ipv4Addr::from([127, 0, 0, 1
                                  ;(let one = i16::from(true
                              ;(let bigger = i32::from(123_i16
                        ;("{println!("{s}, {addr}, {one}, {bigger
                                                              {
```

`Into` [illegible] `From` [illegible]:

```
                                                  } ()fn main
                              ;()let s: String = "hello".into
               ;()let addr: std::net::Ipv4Addr = [127, 0, 0, 1].into
                                  ;()let one: i16 = true.into
                            ;()let bigger: i32 = 123_i16.into
                        ;("{println!("{s}, {addr}, {one}, {bigger
                                                              {
```

- [illegible] From [illegible] Into [illegible].
- [illegible] ”[illegible] String [illegible]” [illegible] Into [illegible]. [illegible] From [illegible] Into [illegible].

17.4 Casting

Rust [illegible] *implicit* [illegible] as [illegible]. [illegible] C [illegible].

```rust
fn main() {
    let value: i64 = 1000;
    println!("as u16: {}", value as u16);
    println!("as i16: {}", value as i16);
    println!("as u8: {}", value as u8);
}
```

[illegible] as [illegible] Rust [illegible]. [illegible] -- [illegible].

[illegible] as [illegible]. [illegible] ([illegible] 32 [illegible] u32 [illegible] u64 as [illegible]).

[illegible] ([illegible] u32 [illegible] u64) [illegible] From [illegible] Into [illegible] as [illegible]. [illegible] TryFrom [illegible] TryInto [illegible].

[illegible].

as [illegible] static_cast [illegible] C++ [illegible]. [illegible] as [illegible].

[illegible] usize [illegible].

17.5 Read and Write

[illegible] Read [illegible] BufRead [illegible] u8 [illegible]:

```rust
use std::io::{BufRead, BufReader, Read, Result};

fn count_lines<R: Read>(reader: R) -> usize {
    let buf_reader = BufReader::new(reader);
    buf_reader.lines().count()
}

fn main() -> Result<()> {
    let slice: &[u8] = b"foo\nbar\nbaz\n";
```

```
       ;((println!("lines in slice: {}", count_lines(slice

;?(?()let file = std::fs::File::open(std::env::current_exe
         ;((println!("lines in file: {}", count_lines(file
                                                    (())Ok
                                                             {
```

:□□□□□ □□□□□□□ u8 □□□□□□ □□□□ □□ □□ □□□□□□ □□□□□□ □□□□ □□ Write □□□□□□□ □□□□ □□

```
                              ;{use std::io::{Result, Write

} <()>fn log<W: Write>(writer: &mut W, msg: &str) -> Result
                     ;?(()writer.write_all(msg.as_bytes
                      (()writer.write_all("\n".as_bytes
                                                          {
```

```
            } <()>fn main() -> Result
     ;()let mut buffer = Vec::new
       ;?("□□□□□" ,log(&mut buffer
       ;?("□□□□□" ,log(&mut buffer
;(println!("Logged: {:?}", buffer
                           (())Ok
                                    {
```

The Default Trait 17.6

.□□□□□□ □□□□□□ □□□□□ □□ □□□□□ □□□□□□□ □□□□□□ □□ Default □□□□□□

```
                                                } struct Derived
                                                     ,x: u32
                                                  ,y: String
                                             ,z: Implemented
                                                               {

                                     ;(struct Implemented(String

                                  } impl Default for Implemented
                                      } fn default() -> Self
                               (()Self("John Smith".into
                                                           {
                                                               {

                                                     } ()fn main
                    ;()let default_struct = Derived::default
                            ;("{?#:println!("{default_struct

                                 = let almost_default_struct
;{ ()Derived { y: "Y is set!".into(), ..Derived::default
                     ;("{?#:println!("{almost_default_struct

                        ;let nothing: Option<Derived> = None
```

```
    println!("{:#?}", nothing.unwrap_or_default());
}
```

- [illegible] #[derive(Default)] [illegible].
- [illegible].
  - [illegible] Default [illegible].
- [illegible] Rust [illegible] Default [illegible] (0 [illegible] "" [illegible]).
- [illegible] Default [illegible].
- [illegible] Rust [illegible] Default [illegible].
- [illegible] .. [illegible].

17.7 Closures

[illegible]. [illegible] FnMut [illegible] Fn traits [illegible] FnOnce [illegible]:

```
fn apply_and_log(func: impl FnOnce(i32) -> i32, func_name: &str, input: i32) {
    println!("[illegible] {func_name}({input}): {}", func(input));
}

fn main() {
    let n = 3;
    let add_3 = |x| x + n;
    apply_and_log(&add_3, "add_3", 10);
    apply_and_log(&add_3, "add_3", 20);

    let mut v = Vec::new();
    let mut accumulate = |x: i32| {
        v.push(x);
        v.iter().sum::<i32>()
    };
    apply_and_log(&mut accumulate, "[illegible]: {}", 4);
    apply_and_log(&mut accumulate, "[illegible]: {}", 5);

    let multiply_sum = |x| x * v.into_iter().sum::<i32>();
    apply_and_log(multiply_sum, "multiply_sum", 3);
}
```

An Fn (e.g. add_3) neither consumes nor mutates captured values. It can be called needing only a shared reference to the closure, which means the closure can be executed repeatedly and even concurrently.

An `FnMut` (e.g. `accumulate`) might mutate captured values. The closure object is accessed via exclusive reference, so it can be called repeatedly but not concurrently.

If you have an `FnOnce` (e.g. `multiply_sum`), you may only call it once. Doing so consumes the closure and any values captured by move.

□□□□□□ □□ .□□□□ FnOnce □ FnMut □□ □□□□□□□□□ □□ □□□□ Fn .□□□□ FnOnce □□ □□□□□□□□□ □□ FnMut □□ □□□□□ □□ Fn □□ □ □□□□□ □□□□□□□□□ □□□□ □□□□□ FnOnce □□ □□□□□ □□ FnMut □□ □□□□□□□□□□ □□□□□□ .□□□□□ □□□□□□□□□ □□□□ □□□□□ FnOnce □□ FnMut

□□□□□ □□□□□□□□□ FnOnce □□ □□□□□ □□□□□□□□□ □□ closure □□ □□ □□□□□□□□ □□□□□□□ □□□□□□□ □□ □□□□□□□ □□ FnMut □□ □□ □(□□□□□□□ □□□□□□□□□□ □□□□ □□ □□□□□) □□□□□□□□ □□□□□□□□□□□ □□ □□ □□□□ □□ □□□□ □□□□ □□□□□□□□□□□□□□□ □□□□□ □□ □□□□□□□□□□□□□ □□□□□□□□ □□□□ □□□□ .Fn □□ □□□□□□ □□ □ □□□□□□□ □□□□ □□□□ .□□□□□□ □□□□□□

□□□□□□ □□□□□□ □□□□□□□□□□ □□ □□□□□□□□□□□□□ □□□□□□□□□ □□□□□□□ closure □□ □□ □□□□□□ □□□□□□□ □□ .FnOnce □□□□□□ □□ □ FnMut □□□□ □(□□□□ □□□□□□□□□ □□□□□ □□ □□ □□□□□□□□□ □□) □□□□ Fn

The compiler also infers `Copy` (e.g. for `add_3`) and `Clone` (e.g. `multiply_sum`), depending on what the closure captures. Function pointers (references to `fn` items) implement `Copy` and `.Fn`

□□□□□□□ □□ □□ □□□□□□□ □□□□□□ □□ □□ □□□□□□ □□ (closures) □□□□□□□□□□□ □□□□□□□□□ □□□□□ □□ (□□□□□□□ □□ □□□□ □□□□□□□□□□ □□□□□□ □□□□ □□□□□□ □□□□□ □□□□ □□□□□□ □□□□□□ □□) □□□□□ □□□□□□□□ □□□□ .□□□□□□ □□□□□□□ value □□□□□ □□ □□ capture □□□□□□□□ □□ move □□□□□□□□□ .□□□□□□□ capture

```
                  } (fn make_greeter(prefix: String) -> impl Fn(&str
              ;(return move |name| println!("{} {}", prefix, name
                                                                    {

                                                          } ()fn main
                   ;(()let hi = make_greeter("Hi".to_string
                                                  ;("hi("Greg
                                                                    {
```

ROT13 :□□□□□□   17.8

□□ □□□□ .□□□□ □□□□□□□□ □□□□□□□□□□□ □□ ”ROT13” □□□□□□□□□ □□□□□□□□ □□□□□□□□□□□ □□□□ □□□□□□ □□□□ □□ □□□□□□□ □□□□□ □□□□□ .□□□□□ □□□□□□□□□□□□ □□ □□ □□□□□ □□□□□□□□ □ □□□□□ □□□□ Playground □□□□□ □□ □□ .□□□□□□ □□□□□ □□□□□□ UTF-8 □□□□□□□□ □□□□□□ □□ □□□□□□□□□□ □□ ASCII

```
                                                 ;use std::io::Read

                                        } <struct RotDecoder<R: Read
                                                           ,input: R
                                                             ,rot: u8
                                                                    {

              .`Implement the `Read` trait for `RotDecoder //

                                                          } ()fn main
                                                       = let mut rot
;{ RotDecoder { input: "Gb trg gb gur bgure fvqr!".as_bytes(), rot: 13
                                       ;()let mut result = String::new
                        ;()rot.read_to_string(&mut result).unwrap
                                                ;(println!("{}", result
                                                                    {

                                                           } mod test
```

```
;*::use super

} ()fn joke
= let mut rot
;{ RotDecoder { input: "Gb trg gb gur bgure fvqr!".as_bytes(), rot: 13
;()let mut result = String::new
;()rot.read_to_string(&mut result).unwrap
;("!assert_eq!(&result, "To get to the other side
{

} ()fn binary
;()let input: Vec<u8> = (0..=255u8).collect
;{ let mut rot = RotDecoder::<&[u8]> { input: input.as_ref(), rot: 13
;[let mut buf = [0u8; 256
;(assert_eq!(rot.read(&mut buf).unwrap(), 256
} for i in 0..=255
} [if input[i] != buf[i
;(()assert!(input[i].is_ascii_alphabetic
;(()assert!(buf[i].is_ascii_alphabetic
{
{
{
{
```

[illegible] RotDecoder [illegible]

[illegible] 17.8.1

```
;use std::io::Read

} <struct RotDecoder<R: Read
,input: R
,rot: u8
{

} <impl<R: Read> Read for RotDecoder<R
} <fn read(&mut self, buf: &mut [u8]) -> std::io::Result<usize
;?(let size = self.input.read(buf
} [for b in &mut buf[..size
} ()if b.is_ascii_alphabetic
;let base = if b.is_ascii_uppercase() { 'A' } else { 'a' } as u8
;b = (*b - base + self.rot) % 26 + base*
{
{
(Ok(size
{
{

} ()fn main
= let mut rot
```

```
    ;{ RotDecoder { input: "Gb trg gb gur bgure fvqr!".as_bytes(), rot: 13
                                               ;()let mut result = String::new
                                     ;()rot.read_to_string(&mut result).unwrap
                                                       ;(println!("{}", result
                                                                                 {

                                                                        } mod test
                                                                 ;*::use super

                                                                   } ()fn joke
                                                             = let mut rot
;{ RotDecoder { input: "Gb trg gb gur bgure fvqr!".as_bytes(), rot: 13
                                           ;()let mut result = String::new
                                 ;()rot.read_to_string(&mut result).unwrap
                         ;("!assert_eq!(&result, "To get to the other side
                                                                             {

                                                                 } ()fn binary
                               ;()let input: Vec<u8> = (0..=255u8).collect
     ;{ let mut rot = RotDecoder::<&[u8]> { input: input.as_ref(), rot: 13
                                                 ;[let mut buf = [0u8; 256
                             ;(assert_eq!(rot.read(&mut buf).unwrap(), 256
                                                        } for i in 0..=255
                                               } [if input[i] != buf[i
                          ;(()assert!(input[i].is_ascii_alphabetic
                            ;(()assert!(buf[i].is_ascii_alphabetic
                                                                     {
                                                                         {
                                                                             {
                                                                                 {
```

V [illegible]

[illegible] : [illegible]

18 [illegible]

[illegible]

:[illegible]

- [illegible] (borrow checker): [illegible] Rust [illegible].
- [illegible]: [illegible].

[illegible]

[illegible]. [illegible]:

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

[illegible] 19

[illegible]

[illegible]:

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
Clone	[illegible]
[illegible]	[illegible]
Drop	[illegible]
[illegible]: [illegible]	[illegible]

19.1 [illegible]

[illegible]:

- Stack: [illegible] ([illegible]) [illegible].
  - [illegible].
  - [illegible]: [illegible] stack [illegible].
  - [illegible]: [illegible].
  - [illegible].
- Heap: [illegible].
  - [illegible].
  - [illegible] stack: [illegible].
  - [illegible].

[illegible]

[illegible] String metadata [illegible] stack [illegible] heap [illegible]:

```rust
fn main() {
    let s1 = String::from("[illegible]");
}
```

```
                                                Stack
                         Heap      .- - - - - - - - - - - - - - -.
.- - - - - - - - - - - - - - - -.  :                             :
:                               :  :                       s1    :
:                               :  :   +-------+-----------+     :
:  +----+----+----+----+----+   :  :   | capacity  |     5 |     :
:  |  ptr       |      o-+---+-----+-->| H  | e  | l  | l  | o |  :
:  +----+----+----+----+----+   :  :   | len       |     5 |     :
:                               :  :   +-------+-----------+     :
:                               :  :                             :
'- - - - - - - - - - - - - - - -`  '- - - - - - - - - - - - - - -`
```

- [illegible] String [illegible] Vec [illegible] heap [illegible].
- [illegible] System Allocator [illegible] heap [illegible] Allocator API [illegible].

[illegible]

[illegible] Rust [illegible] (unsafe) [illegible]. [illegible]!

```rust
fn main() {
    let mut s1 = String::from("[illegible]");
    s1.push(' ');
    s1.push_str("[illegible]");
    // DON'T DO THIS AT HOME! For educational purposes only.
    // String provides no guarantees about its layout, so this could lead to
    // undefined behavior.
    unsafe {
        let (capacity, ptr, len): (usize, usize, usize) = std::mem::transmute(s1);
        println!("capacity = {capacity}, ptr = {ptr:#x}, len = {len}");
    }
}
```

19.2 [illegible]

[illegible]:

- [illegible]: C [illegible] C++ [illegible] …
  - [illegible] heap [illegible].
  - [illegible].
  - [illegible].
- [illegible]: [illegible] …

- [illegible].
- Typically implemented with reference counting or garbage collection.

Rust [illegible]:

[illegible].

[illegible].

[illegible] Rust [illegible].

- C [illegible] heap [illegible] `malloc` [illegible] `free` [illegible]. [illegible] `free` [illegible] dereference [illegible].
- C++ [illegible] (`unique_ptr`, `shared_ptr`) [illegible] (destructor) [illegible]. [illegible] C [illegible].
- [illegible] (garbage collector) [illegible]-[illegible]. [illegible] dereference [illegible] (use-after-free) [illegible]. [illegible] GC [illegible].

[illegible] (ownership and borrowing [illegible] (Rust [illegible] C [illegible] -- [illegible] -- [illegible]. [illegible] C++ [illegible]. [illegible] crates [illegible] ([illegible]).

19.3 [illegible]

[illegible]:

```rust
struct Point(i32, i32);

fn main() {
    let p = Point(3, 4);
    println!("x: {}", p.0);
    println!("y: {}", p.1);
}
```

[illegible]. [illegible] Rust [illegible].

[illegible]. [illegible] (destructor) [illegible]-[illegible].

[illegible]-

[illegible] ”[illegible]” [illegible]

[illegible]. ”[illegible]” Rust [illegible].

19.4 [illegible]

[illegible], [illegible]:

```
                                                        } ()fn main
                        ;("![illegible]")let s1: String = String::from
                                                   ;let s2: String = s1
                                                ;("{println!("s2: {s2
                                           ;("{println!("s1: {s1 //
                                                                      {
```

- [illegible] s1[illegible] s2 [illegible].
- [illegible] s1 [illegible], [illegible]: [illegible] s1 [illegible].
- [illegible] s2 [illegible], [illegible].

[illegible] s2 :

```
                        Stack                               Heap
.- - - - - - - - - - - - - - - - - - - -.     .- - - - - - - - - - - - - - -.
:                                       :     :                              :
:                                       :     :                       s1     :
:   +----+----+----+----+----+-----+    :     :   +-------+-----------+      :
:   |  ! |  ptr       |   o---+---+-----+-->| H  | e  | l  | l  | o |      :
:   +----+----+----+----+----+-----+    :     :   | len       |     6 |      :
:                                       :     :   | capacity  |     6 |      :
:                                       :     :   +-------+-----------+      :
'- - - - - - - - - - - - - - - - - - - -`     :                              :
                                              :                              :
                                              '- - - - - - - - - - - - - - -`
```

[illegible] s2 :

```
                        Stack                               Heap
.- - - - - - - - - - - - - - - - - - - -.     .- - - - - - - - - - - - - - -.
:                                       :     :                              :
:                                       :     :    "(s1 "(inaccessible       :
:   +----+----+----+----+----+-----+    :     :   +-------+-----------+      :
:   |  ! |  ptr       |   o---+---+--+--+-->| H  | e  | l  | l  | o |      :
:   +----+----+----+----+----+-----+    :  |  :   | len       |     6 |      :
:                                       :  |  :   | capacity  |     6 |      :
:                                       :  |  :   +-------+-----------+      :
'- - - - - - - - - - - - - - - - - - - -`  |  :                              :
                                           |  :                       s2     :
                                           |  :   +-------+-----------+      :
                                           '--+---+---ptr       |   o |      :
                                              :   | len       |     6 |      :
                                              :   | capacity  |     6 |      :
                                              :   +-------+-----------+      :
```

```
:                                                    :
'- - - - - - - - - - - - - - - -`
```

[illegible]
:[illegible] .[illegible]

```
} (fn say_hello(name: String
    ("{name} [illegible]")!println
                                        {

                    } ()fn main
  ;("[illegible]")let name = String::from
                        ;(say_hello(name
                     ;(say_hello(name //
                                        {
```

- [illegible]

  ++C

  [illegible]

  std::move

  (![illegible]) [illegible]

- [illegible] .[illegible] [illegible] (aggressively) [illegible].
- .([illegible]) Copy [illegible] ([illegible]) [illegible]
- .(clone [illegible]) [illegible] ،Rust [illegible]

:say_hello [illegible]

- [illegible] .[illegible] name [illegible] main [illegible] ،say_hello [illegible] .[illegible] main [illegible] name
- [illegible] say_hello [illegible] name [illegible] .[illegible]
-) [illegible] name [illegible] main [illegible]

  name&

  .[illegible] say_hello [illegible] (

- [illegible] name [illegible] main [illegible]) [illegible]

  ()name.clone

  .[illegible] (

- [illegible] Rust [illegible]

  ++C

[illegible] «[illegible]» [illegible] ,[illegible] .[illegible]

[illegible]

[illegible] C++ [illegible]

++C

[illegible]:

```
;"std::string s1 = "Cpp
.std::string s2 = s1;  // Duplicate the data in s1
```

- [illegible] s1 [illegible] s2 [illegible].
- [illegible] s1 [illegible] s2 [illegible].

[illegible]:

```
                    Stack                             Heap
.- - - - - - - - - - - -.     .- - - - - - - - - - - - - -.
:                       :     :                           :
:                       :     :                     s1    :
:    +----+----+----+   :     :   +-------+-----------+   :
:    |  ptr       |   o---+---+--+--+-->| C  | p  | p |   :
:    +----+----+----+   :     :   | len       |     3 |   :
:                       :     :   | capacity  |     3 |   :
:                       :     :   +-------+-----------+   :
'- - - - - - - - - - - -`     :                           :
                              '- - - - - - - - - - - - - -`
```

[illegible]:

```
                    Stack                             Heap
.- - - - - - - - - - - -.     .- - - - - - - - - - - - - -.
:                       :     :                           :
:                       :     :                     s1    :
:    +----+----+----+   :     :   +-------+-----------+   :
:    |  ptr       |   o---+---+--+--+-->| C  | p  | p |   :
:    +----+----+----+   :     :   | len       |     3 |   :
:                       :     :   | capacity  |     3 |   :
:                       :     :   +-------+-----------+   :
:                       :     :                           :
:                       :     :                     s2    :
:    +----+----+----+   :     :   +-------+-----------+   :
:    |  ptr       |   o---+---+-----+-->| C  | p  | p |   :
:    +----+----+----+   :     :   | len       |     3 |   :
:                       :     :   | capacity  |     3 |   :
:                       :     :   +-------+-----------+   :
'- - - - - - - - - - - -`     :                           :
                              '- - - - - - - - - - - - - -`
```

[illegible]:

- [illegible]

++C

[illegible] Rust [illegible]. [illegible] = [illegible] [illegible]. [illegible] [illegible].

- [illegible]

++C

] [illegible]

`std::move`

[illegible] (https://en.cppreference.com/w/cpp/utility/move)[[illegible]. [illegible]

`(s2 = std::move(s1`

[illegible] s1 [illegible] [illegible] Rust, [illegible]

++C

[illegible] s1 [illegible].

- [illegible] Rust[illegible] = [illegible]

++C

[illegible].

Clone 19.5

[illegible]. Clone [illegible].

```
} (fn say_hello(name: String
 ("{name} □□□□□")!println
                        {

            } ()fn main
;("□□□□□")let name = String::from
      ;(()say_hello(name.clone
              ;(say_hello(name
                        {
```

- [illegible] Clone [illegible] heap [illegible]. [illegible] `.clone()` [illegible] `vec!` [illegible] `Box::new` [illegible].
- [illegible] (borrow checker) [illegible].
- clone [illegible].
- [illegible] clone [illegible].

19.6 [illegible]

[illegible]:

```
fn main() {
    let x = 42;
    let y = x;
    println!("x: {x}"); // would not be accessible if not Copy
    println!("y: {y}");
}
```

[illegible] Copy [illegible].

[illegible]:

```
struct Point(i32, i32);

fn main() {
    let p1 = Point(3, 4);
    let p2 = p1;
    println!("p1: {p1:?}");
    println!("p2: {p2:?}");
}
```

- [illegible] p1 [illegible] p2 [illegible].
- [illegible]

  `p1.clone()`

  [illegible].

[illegible]:

- [illegible].
- [illegible] (constructors [illegible] C++).
- [illegible] Clone [illegible].
- [illegible] Drop [illegible].

[illegible]:

- [illegible] String [illegible] `struct Point` [illegible] String [illegible] Copy [illegible].
- [illegible] Copy [illegible] derive [illegible] `println!` [illegible] p1 [illegible].
- [illegible] p1 [illegible].

[illegible]

- [illegible] (shared references) [illegible] Copy/Clone [illegible] (mutable references) [illegible]. [illegible] Rust [illegible] Rust [illegible].

19.7 Drop [illegible]

[illegible] Drop [illegible]:

```rust
struct Droppable {
    name: &'static str,
}

impl Drop for Droppable {
    fn drop(&mut self) {
        println!("Dropping {}", self.name);
    }
}

fn main() {
    let a = Droppable { name: "a" };
    {
        let b = Droppable { name: "a" };
        {
            let c = Droppable { name: "c" };
            let d = Droppable { name: "d" };
            println!("Exiting block B");
        }
        println!("Exiting block A");
    }
    drop(a);
    println!("Exiting main");
}
```

- [illegible] std::mem::drop [illegible] std::ops::Drop::drop [illegible].
- [illegible].
- [illegible] std::ops::Drop [illegible] Drop::drop [illegible].
- [illegible] Drop [illegible].
- std::mem::drop [illegible].
  - [illegible] drop [illegible]: [illegible].

[illegible] [illegible]:

- [illegible] `self` `Drop::drop` [illegible] [illegible]
  - [illegible] [illegible]: [illegible] [illegible] [illegible] `std::mem::drop` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `Drop::drop` [illegible] [illegible] [illegible] [illegible] (stack overflow) stack [illegible]!
- [illegible] [illegible] `drop(a)` [illegible] [illegible] `a.drop()` [illegible] [illegible].

19.8 [illegible]: [illegible] [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] ”[illegible] [illegible]” [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

[illegible] [illegible] [illegible] [illegible] [illegible].

```rust
enum Language {
    Rust,
    Java,
    Perl,
}

struct Dependency {
    name: String,
    version_expression: String,
}

/// A representation of a software package.
struct Package {
    name: String,
    version: String,
    authors: Vec<String>,
    dependencies: Vec<Dependency>,
    language: Option<Language>,
}

impl Package {
    /// Return a representation of this package as a dependency, for use in
    /// building other packages.
    fn as_dependency(&self) -> Dependency {
        todo!("1")
    }
}

/// A builder for a Package. Use `build()` to create the `Package` itself.
struct PackageBuilder(Package);

impl PackageBuilder {
    fn new(name: impl Into<String>) -> Self {
        todo!("2")
    }
```

```
                                                   .Set the package version ///
                     } fn version(mut self, version: impl Into<String>) -> Self
                                           ;()self.0.version = version.into
                                                                       self
                                                                              {

                                                   .Set the package authors ///
                           } fn authors(mut self, authors: Vec<String>) -> Self
                                                                 ("todo!("3
                                                                              {

                                              .Add an additional dependency ///
                      } fn dependency(mut self, dependency: Dependency) -> Self
                                                                 ("todo!("4
                                                                              {

                   .Set the language. If not set, language defaults to None ///
                            } fn language(mut self, language: Language) -> Self
                                                                 ("todo!("5
                                                                              {

                                                    } fn build(self) -> Package
                                                                     self.0
                                                                              {
                                                                                  {

                                                                        } ()fn main
;()let base64 = PackageBuilder::new("base64: {base64:?}").version("0.13").build
                                                ;("{?:println!("base64: {base64
                                                                      = let log
;()PackageBuilder::new("log").version("0.4").language(Language::Rust).build
                                                      ;("{?:println!("log: {log
                                       ("let serde = PackageBuilder::new("serde
                                          ([()authors(vec!["djmitche".into.
                                              (("version(String::from("4.0.
                                        (()dependency(base64.as_dependency.
                                           (()dependency(log.as_dependency.
                                                                  ;()build.
                                                  ;("{?:println!("serde: {serde
                                                                                  {
```

□□□□□ 19.8.1

```
                                                                    } enum Language
                                                                          ,Rust
                                                                          ,Java
                                                                          ,Perl
                                                                                  {

                                                                } struct Dependency
```

```
                                                               ,name: String
                                                 ,version_expression: String
                                                                               {

                                     .A representation of a software package ///
                                                                } struct Package
                                                               ,name: String
                                                            ,version: String
                                                       ,<authors: Vec<String
                                              ,<dependencies: Vec<Dependency
                                                 ,<language: Option<Language
                                                                               {

                                                                  } impl Package
     Return a representation of this package as a dependency, for use in ///
                                                ".building other packages ///
                                      } fn as_dependency(&self) -> Dependency
                                                            } Dependency
                                                ,()name: self.name.clone
                           ,()version_expression: self.version.clone
                                                                       {
                                                                           {
                                                                               {

      .A builder for a Package. Use `build()` to create the `Package` itself ///
                                                 ;(struct PackageBuilder(Package

                                                           } impl PackageBuilder
                                   } fn new(name: impl Into<String>) -> Self
                                                          } Self(Package
                                                  ,()name: name.into
                                             ,()version: "0.1".into
                                                    ,[]!authors: vec
                                               ,[]!dependencies: vec
                                                     ,language: None
                                                                      ({
                                                                           {

                                                ".Set the package version ///
                  } fn version(mut self, version: impl Into<String>) -> Self
                                        ;()self.0.version = version.into
                                                                    self
                                                                           {

                                                ".Set the package authors ///
                        } fn authors(mut self, authors: Vec<String>) -> Self
                                               ;self.0.authors = authors
                                                                    self
                                                                           {

                                           .Add an additional dependency ///
```

```
                      } fn dependency(mut self, dependency: Dependency) -> Self
                                      ;(self.0.dependencies.push(dependency
                                                                       self
                                                                              {

                   .Set the language. If not set, language defaults to None ///
                            } fn language(mut self, language: Language) -> Self
                                          ;(self.0.language = Some(language
                                                                       self
                                                                              {

                                                    } fn build(self) -> Package
                                                                     self.0
                                                                              {
                                                                                  {

                                                                        } ()fn main
;()let base64 = PackageBuilder::new("base64: {base64:?}").version("0.13").build
                                                ;("{?:println!("base64: {base64
                                                                      = let log
;()PackageBuilder::new("log").version("0.4").language(Language::Rust).build
                                                      ;("{?:println!("log: {log
                                       ("let serde = PackageBuilder::new("serde
                                          ([()authors(vec!["djmitche".into.
                                              (("version(String::from("4.0.
                                        (()dependency(base64.as_dependency.
                                           (()dependency(log.as_dependency.
                                                                  ;()build.
                                                  ;("{?:println!("serde: {serde
                                                                                  {
```

20 [illegible]

[illegible]

[illegible]. [illegible]:

[illegible]	[illegible]
Box<T>	[illegible]
Rc	[illegible]
Owned Trait Objects	[illegible]
[illegible]: [illegible]	[illegible]

20.1 Box<T>

Box [illegible] heap [illegible]:

```rust
fn main() {
    let five = Box::new(5);
    println!("five: {}", *five);
}
```

```
 Stack                           Heap
.- - - - - - - -.      .- - - - - - - -.
:               :      :               :
:               :      :     five      :
:   +------+    :      :   +------+    :
:   |  o---|---+------+-->|  5  |     :
:   +------+    :      :   +------+    :
:               :      :               :
:               :      :               :
'- - - - - - - -`      '- - - - - - - -`
```

Box<T> [illegible] Deref<Target = T> [illegible] T [illegible] Box<T> [illegible].

[illegible] inline [illegible] pointer indirection [illegible] Box [illegible]:

```
                                                                 } <enum List<T
              .A non-empty list: first element and the rest of the list ///
                                                     ,(<<Element(T, Box<List<T
                                                            .An empty list ///
                                                                         ,Nil
                                                                              {

                                                                    } ()fn main
                                                        = <let list: List<i32
    ;((((List::Element(1, Box::new(List::Element(2, Box::new(List::Nil
                                                        ;("{?:println!("{list
                                                                              {
```

```
                                                  Stack                          Heap
.- - - - - - - - - - - - - - - - - - - - - - - - - -.     .- - - - - - - - - - - - - - -.
:                                                    :     :                              :
:                                                    :     :                   list       :
:  +----+----+------+    +----+----+---------+       :     :    +----+----+---------+     :
:  | // | // |  Element | 1  | o--+----+-----+--->| Element | 2  | o--+--->| Nil |       :
:  +----+----+------+    +----+----+---------+       :     :    +----+----+---------+     :
:                                                    :     :                              :
:                                                    :     :                              :
'- - - - - - - - - - - - - - - - - - - - - - - - - -'     '- - - - - - - - - - - - - - - -'
```

- Box [illegible] std::unique_ptr [illegible] C++ [illegible] (null) [illegible].
- Box [illegible]:
  - [illegible] Rust [illegible].
  - [illegible] Box [illegible] heap [illegible].
- [illegible] Box [illegible] List [illegible] List [illegible] List [illegible].
- Box [illegible] List [illegible] heap [illegible].
- Box [illegible] List [illegible] "recursive without indirection" [illegible] Box [illegible].

[illegible]

Niche [illegible]

[illegible] Box [illegible] std::unique_ptr [illegible] C++ [illegible] null/[illegible]. [illegible] Box [illegible] enum[illegible].

[illegible] Option<Box<T>> [illegible] Box<T> [illegible] NULL [illegible] variant [illegible] ("[illegible]"):

```
;use std::mem::size_of_val

;(struct Item(String

} ()fn main
    ;((()let just_box: Box<Item> = Box::new(Item("Just box".into
    = <<let optional_box: Option<Box<Item
        ;(((()Some(Box::new(Item("Optional box".into
    ;let none: Option<Box<Item>> = None

    ;((assert_eq!(size_of_val(&just_box), size_of_val(&optional_box
    ;((assert_eq!(size_of_val(&just_box), size_of_val(&none

    ;((println!("Size of just_box: {}", size_of_val(&just_box
    ;((println!("Size of optional_box: {}", size_of_val(&optional_box
    ;((println!("Size of none: {}", size_of_val(&none
{
```

20.2 Rc

Rc [illegible]. [illegible]:

```
;use std::rc::Rc

} ()fn main
    ;(let a = Rc::new(10
    ;(let b = Rc::clone(&a

    ;("{println!("a: {a
    ;("{println!("b: {b
{
```

- [illegible] (multi-threaded) [illegible] Arc [illegible] Mutex [illegible].
- [illegible] Weak [illegible].
- Rc [illegible].
- Rc [illegible] Rust [illegible] std::shared_ptr [illegible] C++ [illegible].
- Rc::clone [illegible]: [illegible] (allocation) [illegible] (deep clone) [illegible].
- make_mut [illegible] ("clone-on-write") [illegible] (mutable reference) [illegible].
- Rc::strong_count [illegible].
- Rc::downgrade [illegible] (RefCell [illegible]).

20.3 Owned Trait Objects

[illegible] (trait objects) [illegible] &dyn Pet. [illegible] Box [illegible] (owned trait object) [illegible]: Box<dyn Pet>.

```rust
struct Dog {
    name: String,
    age: i8,
}
struct Cat {
    lives: i8,
}

trait Pet {
    fn talk(&self) -> String;
}

impl Pet for Dog {
    fn talk(&self) -> String {
        format!("Woof [illegible] {}!", self.name)
    }
}

impl Pet for Cat {
    fn talk(&self) -> String {
        String::from("Miau!")
    }
}

fn main() {
    let pets: Vec<Box<dyn Pet>> = vec![
        Box::new(Cat { lives: 9 }),
        Box::new(Dog { name: String::from("Fido"), age: 5 }),
    ];
    for pet in pets {
        println!("[illegible] {}", pet.talk());
    }
}
```

[illegible] pets:

```
                                          Stack                            Heap
.- - - - - - - - - - - - - - - - - - - - - - -.     .- - - - - - - - - - - - - -.
:                                             :     :                           :
: +----+----+----+----+         "pets: Vec<dyn Pet>"   :     :   "data: Cat"    :
: |  F  | i  | d  | o |    +-------+-------+  :     :   +-------+-----------+   :
: +----+----+----+----+    | ptr       |   o---+---+--.  :  | lives |     9 |   :
:                   ^      +-------+-------+  :  |  :   | len       |     2 |   :
:                   |                 ^       :  |  :   | capacity  |     2 |   :
:           .-------'                 |       :  |  :   +-------+-----------+   :
  :           |"data:"Dog               |       :  |  :                           :
  :   +-------|--+-------+              |       :  |  :                           :
  :   | name  |  o, 4, 4 |      +-----+-|---+   :  |  '- - - - - - - - - - - - - -`
  :   | o o | o o-|----->| age   |        5 |<--+--`
  :   +----------+-------+      +---|-+---|-+   :
  :                                 |     |     :
  '- - - - - - - - - - - - - - - - -| - - |- - -`
                                    |     |
"Program text"                      |     |
  .- - - - - - - - - - - - - - - - -| - - |- - -.
  :                    vtable       |     |     :
  :   +----------------------+      |     |     :
  :   | "Dog as Pet>::talk>" |<-----`     |     :
  :   +----------------------+            |     :
  :                    vtable             |     :
  :   +----------------------+            |     :
  :   | "Cat as Pet>::talk>" |<-----------'     :
  :   +----------------------+                  :
  :                                             :
  '- - - - - - - - - - - - - - - - - - - - - - -'
```

- [illegible] Vec<dyn Pet> [illegible].
- dyn Pet [illegible] Pet [illegible].
- [illegible] pets [illegible] stack [illegible] vector [illegible] heap [illegible]. [illegible] vector [illegible] (fat pointers) [illegible]:
  - [illegible] (fat pointer) [illegible]. [illegible]: [illegible] (vtable) [illegible] Pet [illegible].
  - [illegible] Dog [illegible] Fido [illegible] name [illegible] age [illegible]. Cat [illegible] lives [illegible].
- [illegible]:

```
   ;(()<println!("{} {}", std::mem::size_of::<Dog>(), std::mem::size_of::<Cat
 ;(()<println!("{} {}", std::mem::size_of::<&Dog>(), std::mem::size_of::<&Cat
                              ;(()<println!("{}", std::mem::size_of::<&dyn Pet
                          ;(()<<println!("{}", std::mem::size_of::<Box<dyn Pet
```

20.4 [illegible]

[illegible] (binary tree) [illegible]. [illegible] N [illegible] N [illegible] N [illegible].

[illegible].

[illegible]: [illegible] (iterator) [illegible] (in-order) [illegible].

```
.A node in the binary tree ///
```

```
                                                 } <struct Node<T: Ord
                                                         ,value: T
                                                 ,<left: Subtree<T
                                                ,<right: Subtree<T
                                                                     {

                                         .A possibly-empty subtree ///
                         ;(<<<struct Subtree<T: Ord>(Option<Box<Node<T

         .A container storing a set of values, using a binary tree ///
                                                                   ///
.If the same value is added multiple times, it is only stored once ///
                                       } <pub struct BinaryTree<T: Ord
                                                 ,<root: Subtree<T
                                                                     {

                                          } <impl<T: Ord> BinaryTree<T
                                                } fn new() -> Self
                                 { ()Self { root: Subtree::new
                                                                 {

                                  } (fn insert(&mut self, value: T
                                      ;(self.root.insert(value
                                                                 {

                                } fn has(&self, value: &T) -> bool
                                          (self.root.has(value
                                                                 {

                                          } fn len(&self) -> usize
                                               ()self.root.len
                                                                 {
                                                                     {

         .`Implement `new`, `insert`, `len`, and `has` for `Subtree //

                                                           } mod tests
                                                     ;*::use super

                                                        } ()fn len
                             ;()let mut tree = BinaryTree::new
                                    ;(assert_eq!(tree.len(), 0
                                               ;(tree.insert(2
                                    ;(assert_eq!(tree.len(), 1
                                               ;(tree.insert(1
                                    ;(assert_eq!(tree.len(), 2
                          tree.insert(2); // not a unique item
                                    ;(assert_eq!(tree.len(), 2
                                                                 {

                                                        } ()fn has
```

```
                               ;()let mut tree = BinaryTree::new
            } ([fn check_has(tree: &BinaryTree<i32>, exp: &[bool
                                        = <let got: Vec<bool
;()exp.len()).map(|i| tree.has(&(i as i32))).collect..0)
                                      ;(assert_eq!(&got, exp
                                                               {

         ;([check_has(&tree, &[false, false, false, false, false
                                                 ;(tree.insert(0
          ;([check_has(&tree, &[true, false, false, false, false
                                                 ;(tree.insert(4
           ;([check_has(&tree, &[true, false, false, false, true
                                                 ;(tree.insert(4
           ;([check_has(&tree, &[true, false, false, false, true
                                                 ;(tree.insert(3
            ;([check_has(&tree, &[true, false, false, true, true
                                                                   {

                                                   } ()fn unbalanced
                               ;()let mut tree = BinaryTree::new
                                               } for i in 0..100
                                             ;(tree.insert(i
                                                               {
                                    ;(assert_eq!(tree.len(), 100
                                         ;((assert!(tree.has(&50
                                                                   {
                                                                       {
```

□□□□□ 20.4.1

```
                                                ;use std::cmp::Ordering

                                          .A node in the binary tree ///
                                                   } <struct Node<T: Ord
                                                           ,value: T
                                                   ,<left: Subtree<T
                                                  ,<right: Subtree<T
                                                                       {

                                           .A possibly-empty subtree ///
                           ;(<<<struct Subtree<T: Ord>(Option<Box<Node<T

           .A container storing a set of values, using a binary tree ///
                                                                     ///
  .If the same value is added multiple times, it is only stored once ///
                                         } <pub struct BinaryTree<T: Ord
                                                   ,<root: Subtree<T
                                                                       {

                                            } <impl<T: Ord> BinaryTree<T
                                                  } fn new() -> Self
```

```
                         { ()Self { root: Subtree::new
                                                         {

                          } (fn insert(&mut self, value: T
                              ;(self.root.insert(value
                                                         {

                        } fn has(&self, value: &T) -> bool
                                  (self.root.has(value
                                                         {

                                  } fn len(&self) -> usize
                                       ()self.root.len
                                                         {
                                                             {

                                     } <impl<T: Ord> Subtree<T
                                        } fn new() -> Self
                                            (Self(None
                                                         {

                          } (fn insert(&mut self, value: T
                                   } match &mut self.0
,(((None => self.0 = Some(Box::new(Node::new(value
            } (Some(n) => match value.cmp(&n.value
       ,(Ordering::Less => n.left.insert(value
                         {} <= Ordering::Equal
   ,(Ordering::Greater => n.right.insert(value
                                                ,{
                                                     {
                                                         {

                        } fn has(&self, value: &T) -> bool
                                       } match &self.0
                                    ,None => false
            } (Some(n) => match value.cmp(&n.value
          ,(Ordering::Less => n.left.has(value
                      ,Ordering::Equal => true
      ,(Ordering::Greater => n.right.has(value
                                                ,{
                                                     {
                                                         {

                                  } fn len(&self) -> usize
                                       } match &self.0
                                        ,None => 0
      ,()Some(n) => 1 + n.left.len() + n.right.len
                                                     {
                                                         {
                                                             {
                                                  } <impl<T: Ord> Node<T
                                          } fn new(value: T) -> Self
     { ()Self { value, left: Subtree::new(), right: Subtree::new
                                                                   {
                                                                       {

                                                             } ()fn main
                                   ;()let mut tree = BinaryTree::new
                                                 ;("tree.insert("foo
                                          ;(assert_eq!(tree.len(), 1
                                                 ;("tree.insert("bar
                                          ;(("assert!(tree.has(&"foo
                                                                       {

                                                             } mod tests
                                                       ;*::use super

                                                          } ()fn len
                               ;()let mut tree = BinaryTree::new
                                      ;(assert_eq!(tree.len(), 0
                                                 ;(tree.insert(2
                                      ;(assert_eq!(tree.len(), 1
                                                 ;(tree.insert(1
                                      ;(assert_eq!(tree.len(), 2
                            tree.insert(2); // not a unique item
                                      ;(assert_eq!(tree.len(), 2
                                                                   {

                                                          } ()fn has
                               ;()let mut tree = BinaryTree::new
            } ([fn check_has(tree: &BinaryTree<i32>, exp: &[bool
                                        = <let got: Vec<bool
;()exp.len()).map(|i| tree.has(&(i as i32))).collect..0)
                                      ;(assert_eq!(&got, exp
                                                               {

         ;([check_has(&tree, &[false, false, false, false, false
                                                 ;(tree.insert(0
          ;([check_has(&tree, &[true, false, false, false, false
                                                 ;(tree.insert(4
           ;([check_has(&tree, &[true, false, false, false, true
                                                 ;(tree.insert(4
           ;([check_has(&tree, &[true, false, false, false, true
                                                 ;(tree.insert(3
            ;([check_has(&tree, &[true, false, false, true, true
                                                                   {

                                                   } ()fn unbalanced
                               ;()let mut tree = BinaryTree::new
                                               } for i in 0..100
                                             ;(tree.insert(i
                            {
;(assert_eq!(tree.len(), 100
     ;((assert!(tree.has(&50
                                {
                                    {
```

VI □□□

□□□ □□ □□□ :□ □□□

21 [illegible]

[illegible] [illegible]

[illegible] .[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
:[illegible]

[illegible]	[illegible] [illegible]
(Borrowing) [illegible]	[illegible] [illegible]
[illegible]	[illegible] [illegible]

22 [illegible]

(Borrowing) [illegible]

:[illegible] [illegible] .[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
:[illegible]	[illegible]

22.1 [illegible]

:[illegible]

```
;(struct Point(i32, i32

} fn add(p1: &Point, p2: &Point) -> Point
      (Point(p1.0 + p2.0, p1.1 + p2.1
                                        {

                                 } ()fn main
            ;(let p1 = Point(3, 4
          ;(let p2 = Point(10, 20
          ;(let p3 = add(&p1, &p2
;("{?:println!("{p1:?} + {p2:?} = {p3
                                        {
```

- .[illegible] add [illegible]
- .[illegible]

.[illegible]-[illegible]

[illegible] [illegible] [illegible]

[illegible] [illegible] [illegible] [illegible] stack [illegible] [illegible] (inlining):

- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] add [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] (inlining) [illegible] `add` [illegible] [illegible] `main`, [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] Playground [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] Godbolt [illegible] [illegible]. [illegible] [illegible] [illegible] "DEBUG" [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] "RELEASE" [illegible] [illegible]:

```rust
struct Point(i32, i32);

fn add(p1: &Point, p2: &Point) -> Point {
    let p = Point(p1.0 + p2.0, p1.1 + p2.1);
    println!("&p.0: {:p}", &p.0);
    p
}

pub fn main() {
    let p1 = Point(3, 4);
    let p2 = Point(10, 20);
    let p3 = add(&p1, &p2);
    println!("&p3.0: {:p}", &p3.0);
    println!("{p1:?} + {p2:?} = {p3:?}");
}
```

- [illegible] Rust [illegible] [illegible] [illegible] (inlining) [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `#[inline(never)]` [illegible] [illegible].
- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] Godbolt [illegible] Playground, [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] ABI [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] amd64, [illegible] `i32` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] (`eax` [illegible] `edx`) [illegible] [illegible] [illegible].

22.2 [illegible] [illegible] [illegible]

[illegible] [illegible] (borrow checker) [illegible] Rust [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]:

- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

```rust
fn main() {
    let mut a: i32 = 10;
    let b: &i32 = &a;

    {
        let c: &mut i32 = &mut a;
        *c = 20;
    }

    println!("a: {a}");
```

```
;("{println!("b: {b
{
```

- [illegible] dereferenced [illegible].
- [illegible] a [illegible] (c [illegible]) [illegible] (b [illegible]) [illegible].
- [illegible] `println!` [illegible] b [illegible] c [illegible].
- [illegible] b [illegible] a [illegible] c [illegible]. [illegible] non-lexical lifetimes [illegible].
- [illegible]. Rust [illegible]-[illegible] (data races) [illegible]. Rust [illegible].
- [illegible]. [illegible] "borrow checker [illegible]" [illegible].

22.3 [illegible]

[illegible]:

```
} ()fn main
;[let mut vec = vec![1, 2, 3, 4, 5
;[let elem = &vec[2
;(vec.push(6
;("{println!("{elem
{
```

[illegible] (iterator) [illegible]:

```
} ()fn main
;[let mut vec = vec![1, 2, 3, 4, 5
} for elem in &vec
;(vec.push(elem * 2
{
{
```

- [illegible].

22.4 [illegible]

[illegible] ([illegible]) [illegible].

[illegible]) [illegible] (interior mutability) ”[illegible]” [illegible] [illegible] .[illegible] ([illegible] [illegible] .[illegible]

Cell

`Cell` wraps a value and allows getting or setting the value using only a shared reference to the `Cell`. However, it does not allow any references to the inner value. Since there are no .references, borrowing rules cannot be broken

```
;use std::cell::Cell

} ()fn main
    .Note that `cell` is NOT declared as mutable //
    ;(let cell = Cell::new(5

    ;(cell.set(123
    ;(()println!("{}", cell.get
{
```

RefCell

`RefCell` allows accessing and mutating a wrapped value by providing alternative types `Ref` .and `RefMut` that emulate `&T`/`&mut T` without actually being Rust references

[illegible] dynamic [illegible] `RefCell` [illegible] type [illegible] .[illegible] `Ref`/`RefMut` [illegible] `RefMut` [illegible]

[illegible] (`RefMut` [illegible] `DerefMut` [illegible]) `Deref` [illegible] .[illegible]

```
;use std::cell::RefCell

} ()fn main
    .Note that `cell` is NOT declared as mutable //
    ;(let cell = RefCell::new(5

    }
        ;()let mut cell_ref = cell.borrow_mut
        ;cell_ref = 123*

        .This triggers an error at runtime //
        ;()let other = cell.borrow //
        ;(println!("{}", *other //
    {

    ;("{?:println!("{cell
{
```

[illegible] Rust [illegible] .[illegible] `Cell` [illegible] `RefCell` [illegible] .[illegible]

- `RefCell` [illegible] [illegible] [illegible] Rust ([illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]) [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

  – [illegible] [illegible] [illegible] `RefCell` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `borrow_mut` [illegible] [illegible] [illegible] [illegible] [illegible] `cell`. [illegible] [illegible] [illegible] [illegible] `RefCell` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `"{borrowed}"` [illegible] [illegible].

- `Cell` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]: [illegible] [illegible] [illegible] [illegible] [illegible] `set` [illegible] [illegible] `&self` [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

- [illegible] `RefCell` [illegible] `Cell` [illegible] `!Sync` [illegible] [illegible] [illegible] [illegible] [illegible] `&RefCell` [illegible] `&Cell` [illegible]-[illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

22.5 [illegible]: [illegible] [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

[illegible] [illegible] [illegible] [illegible] [illegible] `impl` [illegible] [illegible] [illegible] `struct` [illegible] [illegible] `User` [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] `struct User` [illegible] [illegible] [illegible] [illegible] `impl` [illegible] [illegible] [illegible].

[illegible] [illegible] [illegible] https://play.rust-lang.org/ [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]:

```rust
// TODO: [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

pub struct User {
    name: String,
    age: u32,
    height: f32,
    visit_count: usize,
    last_blood_pressure: Option<(u32, u32)>,
}

pub struct Measurements {
    height: f32,
    blood_pressure: (u32, u32),
}

pub struct HealthReport<'a> {
    patient_name: &'a str,
    visit_count: u32,
    height_change: f32,
    blood_pressure_change: Option<(i32, i32)>,
}

impl User {
    pub fn new(name: String, age: u32, height: f32) -> Self {
        Self { name, age, height, visit_count: 0, last_blood_pressure: None }
    }
    } pub fn visit_doctor(&mut self, measurements: Measurements) -> HealthReport
   ("□□□□ □□ □□□□□□ □□□□□ □□□□□□□□□□ □□□□ □□ □□□□□ □□ □□□□ □□□□□□□□□□")!todo
                                                                               {
                                                                                   {

                                                                         } ()fn main
                            ;(let bob = User::new(String::from("Bob"), 32, 155.2
                       ;(bob.name, bob.age ,"□□□ {} □□ □□ □ □□□□ {} □□")!println
                                                                                   {

                                                                   } ()fn test_visit
                        ;(let mut bob = User::new(String::from("Bob"), 32, 155.2
                                                 ;(assert_eq!(bob.visit_count, 0
                                                                    = let report
;({ (bob.visit_doctor(Measurements { height: 156.1, blood_pressure: (120, 80
                                         ;("assert_eq!(report.patient_name, "Bob
                                              ;(assert_eq!(report.visit_count, 1
                                 ;(assert_eq!(report.blood_pressure_change, None
                          ;(assert!((report.height_change - 0.9).abs() < 0.00001

                                                                    = let report
;({ (bob.visit_doctor(Measurements { height: 156.1, blood_pressure: (115, 76

                                              ;(assert_eq!(report.visit_count, 2
                       ;(((assert_eq!(report.blood_pressure_change, Some((-5, -4
                                          ;(assert_eq!(report.height_change, 0.0
                                                                                   {
```

□□□□□ 22.5.1

```
                                                                   } pub struct User
                                                                   ,name: String
                                                                       ,age: u32
                                                                    ,height: f32
                                                             ,visit_count: usize
                                        ,<(last_blood_pressure: Option<(u32, u32
                                                                                   {

                                                           } pub struct Measurements
                                                                    ,height: f32
                                                     ,(blood_pressure: (u32, u32
                                                                                   {

                                                       } <pub struct HealthReport<'a
                                                          ,patient_name: &'a str
                                                               ,visit_count: u32
                                                             ,height_change: f32
                                      ,<(blood_pressure_change: Option<(i32, i32
                                                                                   {
                                                                         } impl User
                       } pub fn new(name: String, age: u32, height: f32) -> Self
       { Self { name, age, height, visit_count: 0, last_blood_pressure: None
                                                                               {

    } pub fn visit_doctor(&mut self, measurements: Measurements) -> HealthReport
                                                      ;self.visit_count += 1
                                       ;let bp = measurements.blood_pressure
                                                 } let report = HealthReport
                                               ,patient_name: &self.name
                                   ,visit_count: self.visit_count as u32
                       ,height_change: measurements.height - self.height
                 } blood_pressure_change: match self.last_blood_pressure
                                                      } <= (Some(lbp
  ((Some((bp.0 as i32 - lbp.0 as i32, bp.1 as i32 - lbp.1 as i32
                                                                   {
                                                       ,None => None
                                                                      ,{
                                                                          ;{
                                          ;self.height = measurements.height
                                        ;(self.last_blood_pressure = Some(bp
                                                                      report
                                                                               {
                                                                                   {

                                                                         } ()fn main
                            ;(let bob = User::new(String::from("Bob"), 32, 155.2
                       ;(bob.name, bob.age ,"□□□ {} □□ □□ □ □□□□ {} □□")!println
                                                                                   {

                                                                   } ()fn test_visit
                        ;(let mut bob = User::new(String::from("Bob"), 32, 155.2
                                                 ;(assert_eq!(bob.visit_count, 0
                                                                    = let report
;({ (bob.visit_doctor(Measurements { height: 156.1, blood_pressure: (120, 80
                                         ;("assert_eq!(report.patient_name, "Bob
                                              ;(assert_eq!(report.visit_count, 1
                                 ;(assert_eq!(report.blood_pressure_change, None
                          ;(assert!((report.height_change - 0.9).abs() < 0.00001

                                                                    = let report
;({ (bob.visit_doctor(Measurements { height: 156.1, blood_pressure: (115, 76

                                              ;(assert_eq!(report.visit_count, 2
                       ;(((assert_eq!(report.blood_pressure_change, Some((-5, -4
                                          ;(assert_eq!(report.height_change, 0.0
                                                                                   {
```

23 [illegible]

[illegible]

:[illegible] .[illegible]

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
Protobuf [illegible] :[illegible]	[illegible]

23.1 [illegible]

.[illegible] .[illegible]

.a Point & 'document str'& :[illegible] - [illegible] .[illegible] a' ' [illegible] .[illegible] Point [illegible] ”[illegible] a Point'& .[illegible] .[illegible] ”[illegible] a [illegible]

.[illegible] :[illegible] .[illegible]

.[illegible] -[illegible]

```
;(struct Point(i32, i32

} fn left_most(p1: &Point, p2: &Point) -> &Point
                            } if p1.0 < p2.0
                                       p1
                                   } else {
                                       p2
                                           {
                                               {
} ()fn main
;(let p1: Point = Point(10, 10
;(let p2: Point = Point(20, 20
?let p3 = left_most(&p1, &p2); // What is the lifetime of p3
;("{?:println!("p3: {p3
{
```

[illegible] .[illegible] p3 [illegible]
p2 [illegible] p1 [illegible] p3 [illegible]
[illegible] .[illegible]
.[illegible]

:[illegible] a' [illegible] `left_most` [illegible]

```
} fn left_most<'a>(p1: &'a Point, p2: &'a Point) -> &'a Point
```

[illegible] a' [illegible] p2 [illegible] p1 [illegible]” [illegible]
.[illegible] a' [illegible]

[illegible]
.[illegible]

23.2 [illegible]

[illegible]
[illegible] .[illegible] Rust
.[illegible] -- [illegible]

- .[illegible] lifetime annotation [illegible]
- -[illegible]
  .[illegible]
- [illegible] self [illegible]
  .[illegible]

```
;(struct Point(i32, i32

} fn cab_distance(p1: &Point, p2: &Point) -> i32
()p1.0 - p2.0).abs() + (p1.1 - p2.1).abs)
{

} <fn nearest<'a>(points: &'a [Point], query: &Point) -> Option<&'a Point
;let mut nearest = None
} for p in points
} if let Some((_, nearest_dist)) = nearest
;(let dist = cab_distance(p, query
} if dist < nearest_dist
;((nearest = Some((p, dist
{
} else {
;(((nearest = Some((p, cab_distance(p, query
;{
{
(nearest.map(|(p, _)| p
{

} ()fn main
)!println
,"{?:}"
)nearest
,[,(Point(1, 0), Point(1, 0), Point(-1, 0), Point(0, -1]&
(Point(0, 2&
(
;(
{
```

.[illegible] cab_distance [illegible]

[illegible] nearest [illegible] .[illegible]

:[illegible] ”[illegible]” [illegible]

```
} <fn nearest<'a, 'q>(points: &'a [Point], query: &'q Point) -> Option<&'q Point
```

[illegible] ([illegible]) [illegible] Rust [illegible].

[illegible] ,Rust [illegible] .[illegible]

Lifetimes in Data Structures 23.3

:[illegible]

```
;(struct Highlight<'doc>(&'doc str

} (fn erase(text: String
;("!{println!("Bye {text
{

} ()fn main
;(".let text = String::from("The quick brown fox jumps over the lazy dog
;([let fox = Highlight(&text[4..19
;([let dog = Highlight(&text[35..43
;(erase(text //
;("{?:println!("{fox
;("{?:println!("{dog
{
```

- &str [illegible] Highlight [illegible] Highlight [illegible].

- [illegible] `text` [illegible] `fox` (dog [illegible]) [illegible] (borrow checker) [illegible].
- [illegible] (borrowed data) [illegible].
- [illegible].
- [illegible].

23.4 [illegible]: Protobuf [illegible]

[illegible].

[illegible] `proto` [illegible] `match` [illegible].

[illegible]:

```
message PhoneNumber {
  optional string number = 1;
  optional string type = 2;
}

message Person {
  optional string name = 1;
  optional int32 id = 2;
  repeated PhoneNumber phones = 3;
}
```

[illegible] "[illegible]" [illegible] ([illegible] 2 [illegible] `id` [illegible] `Person`) [illegible] wire type [illegible].

[illegible] VARINT [illegible]. [illegible] `parse_varint` [illegible]. [illegible] `Person` [illegible] `PhoneNumber` [illegible].

[illegible] `parse_field` [illegible] `ProtoMessage` [illegible] `Person` [illegible] `PhoneNumber` [illegible].

```
/// A wire type as seen on the wire.
enum WireType {
    /// The Varint WireType indicates the value is a single VARINT.
    Varint,
    /// The I64 WireType indicates that the value is precisely 8 bytes in
```

```
.little-endian order containing a 64-bit signed integer or double type ///
                                    I64,  -- not needed for this exercise//
The Len WireType indicates that the value is a length represented as a ///
                      .VARINT followed by exactly that number of bytes ///
                                                                       ,Len
      The I32 WireType indicates that the value is precisely 4 bytes in //
  .little-endian order containing a 32-bit signed integer or float type //
                                    I32,  -- not needed for this exercise//
                                                                           {

                            .A field's value, typed based on the wire type ///
                                                          } <enum FieldValue<'a
                                                                ,(Varint(u64
                              I64(i64),  -- not needed for this exercise//
                                                              ,([Len(&'a [u8
                              I32(i32),  -- not needed for this exercise//
                                                                           {

                       .A field, containing the field number and its value ///
                                                             } <struct Field<'a
                                                                ,field_num: u64
                                                         ,<value: FieldValue<'a
                                                                           {

                                              } trait ProtoMessage<'a>: Default
                                         ;(<fn add_field(&mut self, field: Field<'a
                                                                           {

                                                  } impl From<u64> for WireType
                                                 } fn from(value: u64) -> Self
                                                             } match value
                                                  ,WireType::Varint <= 0
            WireType::I64,  -- not needed for this exercise <= 1//
                                                     ,WireType::Len <= 2
            WireType::I32,  -- not needed for this exercise <= 5//
                         ,("{value} :□□□□□□□ □□□ □□□")!panic <= _
                                                                   {
                                                                      {
                                                                           {

                                                       } <impl<'a> FieldValue<'a
                                                 } fn as_str(&self) -> &'a str
                                 } let FieldValue::Len(data) = self else
                 ;("□□□□ Len □□□□ □□ □□□□ □□ □□□□□ □□□□□□")!panic
                                                                   ;{
                   ("string □□□□□□□")std::str::from_utf8(data).expect
                                                                      {

                                              } [fn as_bytes(&self) -> &'a [u8
                                 } let FieldValue::Len(data) = self else
               ;("□□□□□ `Len` □□□□ □□ □□□□□□ □□□□ □□□□□□□")!panic
```

```
                                                                              ;{
                                                                            data
                                                                                   {

                                                           } fn as_u64(&self) -> u64
                                     } let FieldValue::Varint(value) = self else
                        ;("□□□□□ `Varint` □□□□ □□ `u64` □□□□□□ □□□□□□□")!panic
                                                                              ;{
                                                                          value*
                                                                                   {
                                                                                       {

                 .Parse a VARINT, returning the parsed value and the remaining bytes ///
                                          } ([fn parse_varint(data: &[u8]) -> (u64, &[u8
                                                                     } for i in 0..7
                                                } let Some(b) = data.get(i) else
                                       ;("□□□□□ varint □□□□□ □□□□□ □□□□□")!panic
                                                                              ;{
                                                              } if b & 0x80 == 0
                    This is the last byte of the VARINT, so convert it to //
                                                      .a u64 and return it //
                                                       ;let mut value = 0u64
                                          } ()for b in data[..=i].iter().rev
                               ;value = (value << 7) | (b & 0x7f) as u64
                                                                           {
                                             ;([..return (value, &data[i + 1
                                                                               {
                                                                                   {

                                                    .More than 7 bytes is invalid //
                                          ;("varint □□□□ □□□□□ □□□□□□□ □□□□□")!panic
                                                                                       {

                                   .Convert a tag into a field number and a WireType ///
                                            } (fn unpack_tag(tag: u64) -> (u64, WireType
                                                           ;let field_num = tag >> 3
                                          ;(let wire_type = WireType::from(tag & 0x7
                                                              (field_num, wire_type)
                                                                                       {


                                        Parse a field, returning the remaining bytes ///
                                         } ([fn parse_field(data: &[u8]) -> (Field, &[u8
                                          ;(let (tag, remainder) = parse_varint(data
                                       ;(let (field_num, wire_type) = unpack_tag(tag
                                     } let (fieldvalue, remainder) = match wire_type
'.□□□ □□□□□ □□ □□□□□ □□□□□□ □□ □□□□□ □□ □□□□□□□□□ □□□□□ □□ □□□□□ □□□□ □□□□□ □□")!todo <= _
                                                                                  ;{
                                     (".□□□□□□□□□□ □□ □□□□□ □□□□□ □□□□□ □□ □ □□□□□")!todo
                                                                                       {
```

```
Parse a message in the given data, calling `T::add_field` for each field in ///
                                                          .the message ///
                                                                       ///
                                              .The entire input is consumed ///
          } fn parse_message<'a, T: ProtoMessage<'a>>(mut data: &'a [u8]) -> T
                                                    ;()let mut result = T::default
                                                            } ()while !data.is_empty
                                          ;(let parsed = parse_field(data
                                             ;(result.add_field(parsed.0
                                                      ;data = parsed.1
                                                                     {
                                                                 result
                                                                        {



                                                        } <struct Person<'a
                                                         ,name: &'a str
                                                              ,id: u64
                                              ,<<phone: Vec<PhoneNumber<'a
                                                                        {

                    .TODO: Implement ProtoMessage for Person and PhoneNumber //

                                                                 } ()fn main
                                         ]&)let person: Person = parse_message
,0x0a, 0x07, 0x6d, 0x61, 0x78, 0x77, 0x65, 0x6c, 0x6c, 0x10, 0x2a, 0x1a
,0x16, 0x0a, 0x0e, 0x2b, 0x31, 0x32, 0x30, 0x32, 0x2d, 0x35, 0x35, 0x35
,0x2d, 0x31, 0x32, 0x31, 0x32, 0x12, 0x04, 0x68, 0x6f, 0x6d, 0x65, 0x1a
,0x18, 0x0a, 0x0e, 0x2b, 0x31, 0x38, 0x30, 0x30, 0x2d, 0x38, 0x36, 0x37
,0x2d, 0x35, 0x33, 0x30, 0x38, 0x12, 0x06, 0x6d, 0x6f, 0x62, 0x69, 0x6c
                                                                  ,0x65
                                                                     ;([
                                             ;(println!("{:#?}", person
                                                                        {
```

- [illegible] protobuf [illegible] i32 [illegible] Rust [illegible] Result [illegible] Result [illegible] Rust [illegible]

23.4.1 [illegible]

```
                                         .A wire type as seen on the wire ///
                                                              } enum WireType
                .The Varint WireType indicates the value is a single VARINT ///
                                                                      ,Varint
           The I64 WireType indicates that the value is precisely 8 bytes in ///
      .little-endian order containing a 64-bit signed integer or double type ///
```

```
                                   I64,  -- not needed for this exercise//
The Len WireType indicates that the value is a length represented as a ///
                      .VARINT followed by exactly that number of bytes ///
                                                                      ,Len
      The I32 WireType indicates that the value is precisely 4 bytes in //
  .little-endian order containing a 32-bit signed integer or float type //
                                   I32,  -- not needed for this exercise//
                                                                             {

                            .A field's value, typed based on the wire type ///
                                                         } <enum FieldValue<'a
                                                              ,(Varint(u64
                              I64(i64),  -- not needed for this exercise//
                                                            ,([Len(&'a [u8
                              I32(i32),  -- not needed for this exercise//
                                                                             {

                       .A field, containing the field number and its value ///
                                                            } <struct Field<'a
                                                           ,field_num: u64
                                                    ,<value: FieldValue<'a
                                                                             {

                                             } trait ProtoMessage<'a>: Default
                                ;(<fn add_field(&mut self, field: Field<'a
                                                                             {

                                                 } impl From<u64> for WireType
                                             } fn from(value: u64) -> Self
                                                         } match value
                                            ,WireType::Varint <= 0
            WireType::I64,  -- not needed for this exercise <= 1//
                                               ,WireType::Len <= 2
            WireType::I32,  -- not needed for this exercise <= 5//
                          ,("{value} :□□□□□□□□□ □□□□ □□□□")!panic <= _
                                                                     {
                                                                         {
                                                                             {

                                                     } <impl<'a> FieldValue<'a
                                             } fn as_str(&self) -> &'a str
                               } let FieldValue::Len(data) = self else
                  ;("□□□□□ Len □□□□□ □□ □□□□□ □□ □□□□□□ □□□□□□□□")!panic
                                                                    ;{
                    ("string □□□□□□□□□")std::str::from_utf8(data).expect
                                                                         {

                                          } [fn as_bytes(&self) -> &'a [u8
                               } let FieldValue::Len(data) = self else
                ;("□□□□□□□ `Len` □□□□□ □□ □□□□□□□□ □□□□□ □□□□□□□□□□")!panic
                                                                    ;{
                                                                            data
                                                                                   {

                                                           } fn as_u64(&self) -> u64
                                     } let FieldValue::Varint(value) = self else
                        ;("□□□□ `Varint` □□□□ □□ `u64` □□□□□ □□□□□□")!panic
                                                                              ;{
                                                                          value*
                                                                                   {
                                                                                       {

                 .Parse a VARINT, returning the parsed value and the remaining bytes ///
                                          } ([fn parse_varint(data: &[u8]) -> (u64, &[u8
                                                                     } for i in 0..7
                                                } let Some(b) = data.get(i) else
                                       ;("□□□□ varint □□□□ □□□□ □□□□")!panic
                                                                              ;{
                                                              } if b & 0x80 == 0
                   This is the last byte of the VARINT, so convert it to //
                                                     .a u64 and return it //
                                                       ;let mut value = 0u64
                                          } ()for b in data[..=i].iter().rev
                               ;value = (value << 7) | (b & 0x7f) as u64
                                                                           {
                                             ;([..return (value, &data[i + 1
                                                                               {
                                                                                   {

                                                    .More than 7 bytes is invalid //
                                          ;("varint □□□□ □□□□□ □□□□□□□ □□□□□")!panic
                                                                                       {

                                   .Convert a tag into a field number and a WireType ///
                                            } (fn unpack_tag(tag: u64) -> (u64, WireType
                                                           ;let field_num = tag >> 3
                                          ;(let wire_type = WireType::from(tag & 0x7
                                                              (field_num, wire_type)
                                                                                       {

                                        Parse a field, returning the remaining bytes ///
                                         } ([fn parse_field(data: &[u8]) -> (Field, &[u8
                                          ;(let (tag, remainder) = parse_varint(data
                                       ;(let (field_num, wire_type) = unpack_tag(tag
                                     } let (fieldvalue, remainder) = match wire_type
                                                           } <= WireType::Varint
                           ;(let (value, remainder) = parse_varint(remainder
                                      (FieldValue::Varint(value), remainder)
                                                                               {
                                                              } <= WireType::Len
                             ;(let (len, remainder) = parse_varint(remainder
□□□□ □□□□□ `usize` □□ len □□ □□□□□□□")let len: usize = len.try_into().expect
```

```
                                         } if remainder.len() < len
                                      ;("□□□□□□□ □□□ panic!("EOF
                                                                  {
                  ;(let (value, remainder) = remainder.split_at(len
                                (FieldValue::Len(value), remainder)
                                                                      {
                                                                         ;{
                        (Field { field_num, value: fieldvalue }, remainder)
                                                                              {

Parse a message in the given data, calling `T::add_field` for each field in ///
                                                               .the message ///
                                                                            ///
                                              .The entire input is consumed ///
           } fn parse_message<'a, T: ProtoMessage<'a>>(mut data: &'a [u8]) -> T
                                             ;()let mut result = T::default
                                                   } ()while !data.is_empty
                                        ;(let parsed = parse_field(data
                                            ;(result.add_field(parsed.0
                                                       ;data = parsed.1
                                                                          {
                                                                     result
                                                                              {

                                                       } <struct PhoneNumber<'a
                                                           ,number: &'a str
                                                            ,type_: &'a str
                                                                              {

                                                            } <struct Person<'a
                                                             ,name: &'a str
                                                                   ,id: u64
                                               ,<<phone: Vec<PhoneNumber<'a
                                                                              {

                                     } <impl<'a> ProtoMessage<'a> for Person<'a
                                } (<fn add_field(&mut self, field: Field<'a
                                                } match field.field_num
                             ,()self.name = field.value.as_str <= 1
                               ,()self.id = field.value.as_u64 <= 2
       ,((()self.phone.push(parse_message(field.value.as_bytes <= 3
                                    skip everything else // {} <= _
                                                                      {
                                                                          {
                                                                              {

                                } <impl<'a> ProtoMessage<'a> for PhoneNumber<'a
                                } (<fn add_field(&mut self, field: Field<'a
                                                } match field.field_num
                           ,()self.number = field.value.as_str <= 1
                            ,()self.type_ = field.value.as_str <= 2
```

```
                                        skip everything else // {} <= _
                                                                          {
                                                                              {
                                                                                  {

                                                                        } ()fn main
                                          ]&)let person: Person = parse_message
    ,0x0a, 0x07, 0x6d, 0x61, 0x78, 0x77, 0x65, 0x6c, 0x6c, 0x10, 0x2a, 0x1a
    ,0x16, 0x0a, 0x0e, 0x2b, 0x31, 0x32, 0x30, 0x32, 0x2d, 0x35, 0x35, 0x35
    ,0x2d, 0x31, 0x32, 0x31, 0x32, 0x12, 0x04, 0x68, 0x6f, 0x6d, 0x65, 0x1a
    ,0x18, 0x0a, 0x0e, 0x2b, 0x31, 0x38, 0x30, 0x30, 0x2d, 0x38, 0x36, 0x37
    ,0x2d, 0x35, 0x33, 0x30, 0x38, 0x12, 0x06, 0x6d, 0x6f, 0x62, 0x69, 0x6c
                                                                      ,0x65
                                                                            ;([
                                                     ;(println!("{:#?}", person
                                                                                  {

                                                                        } mod tests
                                                                  ;*::use super

                                                                 } ()fn test_id
                      ;([let person_id: Person = parse_message(&[0x10, 0x2a
        ;({ []!assert_eq!(person_id, Person { name: ".", id: 42, phone: vec
                                                                              {

                                                               } ()fn test_name
                                 ]&)let person_name: Person = parse_message
,0x0a, 0x0e, 0x62, 0x65, 0x61, 0x75, 0x74, 0x69, 0x66, 0x75, 0x6c, 0x20
                                                ,0x6e, 0x61, 0x6d, 0x65
                                                                        ;([
                                                                )!assert_eq
                                                           ,person_name
                { []!Person { name: "beautiful name", id: 0, phone: vec
                                                                         ;(
                                                                              {

                                                        } ()fn test_just_person
                                               = let person_name_id: Person
      ;([parse_message(&[0x0a, 0x04, 0x45, 0x76, 0x61, 0x6e, 0x10, 0x16
;({ []!assert_eq!(person_name_id, Person { name: "Evan", id: 22, phone: vec
                                                                              {

                                                              } ()fn test_phone
                                       ]&)let phone: Person = parse_message
,0x0a, 0x00, 0x10, 0x00, 0x1a, 0x16, 0x0a, 0x0e, 0x2b, 0x31, 0x32, 0x33
,0x34, 0x2d, 0x37, 0x37, 0x37, 0x2d, 0x39, 0x30, 0x39, 0x30, 0x12, 0x04
                                                ,0x68, 0x6f, 0x6d, 0x65
                                                                        ;([
                                                                )!assert_eq
                                                                 ,phone
                                                               } Person
```

```
                                                            ,"." :name
                                                                ,id: 0
,[,{ "□□□□□" :_phone: vec![PhoneNumber { number: "+1234-777-9090", type
                                                                         {
                                                                            ;(
                                                                                 {
                                                                                     {
```

VII [illegible]

[illegible] :[illegible] [illegible]

24 [illegible]

Welcome to Day 4

:[illegible] Rust [illegible]

- .`Iterator` [illegible] :Iterators
- .[illegible]
- .[illegible]
- .? [illegible] `Result` [illegible] panics :[illegible]
- [illegible] safe Rust [illegible] :Unsafe Rust [illegible] .[illegible]

[illegible]

:[illegible] .[illegible]

[illegible]	[illegible]
[illegible]	[illegible]
Iterators	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

25 [illegible]

Iterators

[illegible]: [illegible]. [illegible]

[illegible]	[illegible]
Iterator	[illegible]
IntoIterator	[illegible]
FromIterator	[illegible]
[illegible]: [illegible] Iterator Chaining	[illegible]

25.1 Iterator

[illegible] 'Iterator' [illegible]. [illegible] next [illegible]. [illegible] Iterator [illegible]:

```
struct Fibonacci {
    curr: u32,
    next: u32,
}

impl Iterator for Fibonacci {
    type Item = u32;

    fn next(&mut self) -> Option<Self::Item> {
        let new_next = self.curr + self.next;
        self.curr = self.next;
        self.next = new_next;
        Some(self.curr)
    }
}

fn main() {
    let fib = Fibonacci { curr: 0, next: 1 };
    for (i, n) in fib.enumerate().take(5) {
        println!("fib({i}): {n}");
    }
}
```

- ▯▯▯▯▯▯ Iterator ▯▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯ ▯▯▯ collection▯▯ ▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯ (▯▯▯▯▯▯ map, filter, reduce,▯ ▯▯▯▯▯). ▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯ ▯▯ ▯▯ ▯▯ ▯▯-▯▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯ ▯▯ ▯▯▯▯▯. ▯▯ Rust▯ ▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯ ▯▯ ▯▯ ▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯.
- IntoIterator ▯▯▯▯▯▯ ▯▯▯▯ ▯▯ ▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯▯. ▯▯▯▯ ▯▯▯▯▯▯▯ (collection) ▯▯▯▯▯ ▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯▯▯▯▯ Vec<T> ▯ ▯▯▯▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ &Vec<T> ▯ &[T] ▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯. Ranges ▯▯▯ ▯▯ ▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯. ▯▯ ▯▯▯▯▯ ▯▯▯▯ ▯▯▯ ▯▯-▯▯▯▯▯▯▯ ▯▯▯▯ ▯▯ ▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯ i ▯▯ some_vec { .. } ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯▯▯▯▯ some_vec.next() ▯▯▯▯▯ ▯▯▯▯▯▯.

25.2 IntoIterator

▯▯▯▯▯▯ Iterator ▯▯▯ ▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯▯ ▯▯ ▯▯ ▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯▯ *iterate* ▯▯▯▯▯. ▯▯▯▯▯▯ IntoIterator ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯ ▯▯ ▯▯▯▯ ▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯. ▯▯ ▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯ for ▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯.

```
struct Grid {
    x_coords: Vec<u32>,
    y_coords: Vec<u32>,
}

impl IntoIterator for Grid {
    type Item = (u32, u32);
    type IntoIter = GridIter;
    fn into_iter(self) -> GridIter {
        GridIter { grid: self, i: 0, j: 0 }
    }
}

struct GridIter {
    grid: Grid,
    i: usize,
    j: usize,
}

impl Iterator for GridIter {
    type Item = (u32, u32);

    fn next(&mut self) -> Option<(u32, u32)> {
        if self.i >= self.grid.x_coords.len() {
            self.i = 0;
            self.j += 1;
            if self.j >= self.grid.y_coords.len() {
                return None;
            }
        }
        let res = Some((self.grid.x_coords[self.i], self.grid.y_coords[self.j]));
        self.i += 1;
        res
    }
}

fn main() {
    let grid = Grid { x_coords: vec![3, 5, 7, 9], y_coords: vec![10, 20, 30, 40] };
    for (x, y) in grid {
        println!("point = {x}, {y}");
    }
}
```

[illegible] IntoIterator[illegible] [illegible]. [illegible] IntoIterator [illegible]:

- Item: [illegible] i8[illegible]
- IntoIter: [illegible] «Iterator» [illegible] into_iter [illegible].

[illegible] IntoIter [illegible] Item[illegible] link [illegible]: [illegible] (iterator) [illegible] type Item [illegible] Option<Item> [illegible].

[illegible] x [illegible] y [illegible].

[illegible] main [illegible]. [illegible] IntoIterator::into_iter [illegible] self [illegible].

[illegible] IntoIterator [illegible] &Grid [illegible] reference [illegible] Grid [illegible] GridIter [illegible].

[illegible]: [illegible] e [illegible] some_vector [illegible] some_vector [illegible]. [illegible] e [illegible] &some_vector [illegible] some_vector [illegible].

25.3 FromIterator

[illegible] FromIterator [illegible] [Iterator](https://doc.rust-lang.org/std/iter/trait.Iterator.html) [illegible] collection [illegible].

```
fn main() {
    let primes = vec![2, 3, 5, 7];
    let prime_squares = primes.into_iter().map(|p| p * p).collect::<Vec<_>>();
    println!("prime_squares: {prime_squares:?}");
}
```

[illegible] Iterator

```
fn collect<B>(self) -> B
where
    B: FromIterator<Self::Item>,
    Self: Sized
```

[illegible] B[illegible]:

- [illegible] “turbofish”: `some_iterator.collect::<COLLECTION_TYPE>()` [illegible] _ [illegible] Rust [illegible] “Vec” [illegible].
- [illegible] inference: `let prime_squares: Vec<_> = some_iterator.collect()`. [illegible].

[illegible] `FromIterator` [illegible] `HashMap`[illegible] `Vec`, [illegible]. [illegible] `Iterator<Item = Result<V, E>>` [illegible] `Result<Vec<V>, E>` [illegible].

25.4 [illegible]: Iterator Chaining

[illegible] `Iterator` [illegible].

[illegible] https://play.rust-lang.org/ [illegible] (iterator) [illegible] (`collect`) [illegible].

```rust
/// Calculate the differences between elements of `values` offset by `offset`,
/// wrapping around from the end of `values` to the beginning.
///
/// Element `n` of the result is `values[(n+offset)%len] - values[n]`.
fn offset_differences<N>(offset: usize, values: Vec<N>) -> Vec<N>
where
    N: Copy + std::ops::Sub<Output = N>,
{
    unimplemented!()
}

fn test_offset_one() {
    assert_eq!(offset_differences(1, vec![1, 3, 5, 7]), vec![2, 2, 2, -6]);
    assert_eq!(offset_differences(1, vec![1, 3, 5]), vec![2, 2, -4]);
    assert_eq!(offset_differences(1, vec![1, 3]), vec![2, -2]);
}

fn test_larger_offsets() {
    assert_eq!(offset_differences(2, vec![1, 3, 5, 7]), vec![4, 4, -4, -4]);
    assert_eq!(offset_differences(3, vec![1, 3, 5, 7]), vec![6, -2, -2, -2]);
    assert_eq!(offset_differences(4, vec![1, 3, 5, 7]), vec![0, 0, 0, 0]);
    assert_eq!(offset_differences(5, vec![1, 3, 5, 7]), vec![2, 2, 2, -6]);
}

fn test_custom_type() {
    assert_eq!(
        offset_differences(1, vec![1.0, 11.0, 5.0, 0.0]),
        vec![10.0, -6.0, -5.0, 1.0]
    );
}

fn test_degenerate_cases() {
```

```
                      ;([assert_eq!(offset_differences(1, vec![0]), vec![0
                      ;([assert_eq!(offset_differences(1, vec![1]), vec![0
                                             ;[]!let empty: Vec<i32> = vec
                         ;([]!assert_eq!(offset_differences(1, empty), vec
                                                                             {
```

□□□□□ 25.4.1

```
,`Calculate the differences between elements of `values` offset by `offset ///
                .wrapping around from the end of `values` to the beginning ///
                                                                           ///
        .`[Element `n` of the result is `values[(n+offset)%len] - values[n ///
             <fn offset_differences<N>(offset: usize, values: Vec<N>) -> Vec<N
                                                                         where
                                      ,<N: Copy + std::ops::Sub<Output = N
                                                                             }
                                            ;()let a = (&values).into_iter
                       ;(let b = (&values).into_iter().cycle().skip(offset
                                  ()a.zip(b).map(|(a, b)| *b - *a).collect
                                                                             {

                                                        } ()fn test_offset_one
   ;([assert_eq!(offset_differences(1, vec![1, 3, 5, 7]), vec![2, 2, 2, -6
         ;([assert_eq!(offset_differences(1, vec![1, 3, 5]), vec![2, 2, -4
               ;([assert_eq!(offset_differences(1, vec![1, 3]), vec![2, -2
                                                                             {

                                                    } ()fn test_larger_offsets
  ;([assert_eq!(offset_differences(2, vec![1, 3, 5, 7]), vec![4, 4, -4, -4
 ;([assert_eq!(offset_differences(3, vec![1, 3, 5, 7]), vec![6, -2, -2, -2
    ;([assert_eq!(offset_differences(4, vec![1, 3, 5, 7]), vec![0, 0, 0, 0
   ;([assert_eq!(offset_differences(5, vec![1, 3, 5, 7]), vec![2, 2, 2, -6
                                                                             {

                                                       } ()fn test_custom_type
                                                               )!assert_eq
                     ,([offset_differences(1, vec![1.0, 11.0, 5.0, 0.0
                                           [vec![10.0, -6.0, -5.0, 1.0
                                                                        ;(
                                                                             {

                                                  } ()fn test_degenerate_cases
                      ;([assert_eq!(offset_differences(1, vec![0]), vec![0
                      ;([assert_eq!(offset_differences(1, vec![1]), vec![0
                                             ;[]!let empty: Vec<i32> = vec
                         ;([]!assert_eq!(offset_differences(1, empty), vec
                                                                             {

                                                                  {} ()fn main
```

26 [illegible]

[illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible]:

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
use, super, self	[illegible]
[illegible]: [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]	[illegible]

26.1 [illegible]

[illegible] [illegible] [illegible] impl [illegible] [illegible] [illegible] namespace functions [illegible] [illegible] [illegible] type [illegible] [illegible].

[illegible] [illegible] [illegible] mod [illegible] [illegible] [illegible] [illegible] namespace type [illegible] [illegible] [illegible] [illegible]:

```rust
mod foo {
    pub fn do_something() {
        println!("foo [illegible] [illegible]");
    }
}

mod bar {
    pub fn do_something() {
        println!("[illegible] [illegible] [illegible]");
    }
}

fn main() {
    foo::do_something();
    bar::do_something();
}
```

- [illegible] Package [illegible] Cargo.toml [illegible] crate [illegible] +1 [illegible].
- [illegible] Crate [illegible] crate [illegible] crate [illegible].
- [illegible] scope [illegible] organization [illegible].

26.2 [illegible]

[illegible] Rust [illegible]:

```
;mod garden
```

[illegible] rust [illegible] garden [illegible] `src/garden.rs` [illegible]. [illegible] `garden::vegetables` [illegible] `src/garden/vegetables.rs` [illegible].

[illegible] `crate` [illegible]:

- (`src/lib.rs` (for a library crate
- (`src/main.rs` (for a binary crate

[illegible] «[illegible]» [illegible]. [illegible] - [illegible].

```
This module implements the garden, including a highly performant germination !//
                                                            .implementation !//

                                                .Re-export types from this module //
                                                                ;pub use garden::Garden
                                                            ;pub use seeds::SeedPacket

                                                          .Sow the given seed packets ///
                                                    } (<pub fn sow(seeds: Vec<SeedPacket
                                                                               ()!todo
                                                                                     {

                                 .Harvest the produce in the garden that is ready ///
                                                   } (pub fn harvest(garden: &mut Garden
                                                                               ()!todo
                                                                                     {
```

- [illegible] Rust 2018 [illegible] `module.rs` [illegible] `module/mod.rs` [illegible] 2018 [illegible].
- [illegible] `filename.rs` [illegible] `filename/mod.rs` [illegible] `mod.rs` [illegible] IDE [illegible].
- [illegible] (nesting) [illegible] folder [illegible]:

```
          /src
    main.rs ──├
top_module.rs ──├
  /top_module ──└
sub_module.rs ──└
```

- [illegible] rust [illegible]:

```
;mod some_module
```

[illegible] some_module_test.rs [illegible] (convention) [illegible] Go.

26.3 [illegible]

[illegible]:

- [illegible] private [illegible] ([illegible]).
- [illegible] sibling [illegible] sibling [illegible].
- [illegible] foo [illegible] foo[illegible].

```
} mod outer
    } ()fn private
        ;("println!("outer::private
    {

    } ()pub fn public
        ;("println!("outer::public
    {

    } mod inner
        } ()fn private
            ;("println!("outer::inner::private
        {

        } ()pub fn public
            ;("println!("outer::inner::public
            ;()super::private
        {
    {
{

} ()fn main
    ;()outer::public
{
```

- [illegible] pub [illegible] public [illegible].

[illegible] pub(. . .) [illegible].

- [illegible] Rust Reference.
- [illegible] (pub(crate [illegible].
- [illegible].
- [illegible] ([illegible]) [illegible].

26.4 use, super, self

[illegible] use [illegible]. [illegible]:

```rust
use std::collections::HashSet;
use std::process::abort;
```

[illegible]

[illegible] (Paths) [illegible]:

1. [illegible] path [illegible]:
   - [illegible] foo [illegible] self::foo [illegible] foo [illegible]
   - super::foo refers to foo in the parent module.
2. [illegible] path [illegible]:
   - [illegible] crate::foo [illegible] foo [illegible],
   - [illegible] bar::foo [illegible] foo [illegible] bar crate [illegible].

- [illegible] "re-export" [illegible]. [illegible] lib.rs [illegible] crate [illegible]

  ```rust
  mod storage;

  pub use storage::disk::DiskStorage;
  pub use storage::network::NetworkStorage;
  ```

  [illegible] DiskStorage [illegible] NetworkStorage [illegible] crate [illegible].

- [illegible] use [illegible]. [illegible] (trait) [illegible] method [illegible]. [illegible] read_to_string [illegible] Read [illegible] use std::io::Read [illegible].

- [illegible] use [illegible]: use std::io::*. [illegible] import [illegible].

26.5 [illegible]: [illegible]

[illegible] GUI [illegible]. [illegible] Widget [illegible] main [illegible].

[illegible].

Cargo Setup

[illegible] Rust playground [illegible] Cargo [illegible]:

```
cargo init gui-modules
cd gui-modules
cargo run
```

□□□ src/main.rs □□□□□ □□ □□□□□□□ □□□□□ □□ □□□□□□□ mod □□ □□□□□□ □□□□□ □ □□□□□□□□□ □□□□□□ □□
□□ □□□□□□□□□□ src □□□□□□ □□□□□.

□□□□□

□□ □□□□□□ □□□□□□ □□ □□□□□□ □□□□□□□□□ GUI □□□□□ □□□□:

```
pub trait Widget {
    /// Natural width of `self`.
    fn width(&self) -> usize;

    /// Draw the widget into a buffer.
    fn draw_into(&self, buffer: &mut dyn std::fmt::Write);

    /// Draw the widget on standard output.
    fn draw(&self) {
        let mut buffer = String::new();
        self.draw_into(&mut buffer);
        println!("{buffer}");
    }
}

pub struct Label {
    label: String,
}

impl Label {
    fn new(label: &str) -> Label {
        Label { label: label.to_owned() }
    }
}

pub struct Button {
    label: Label,
}

impl Button {
    fn new(label: &str) -> Button {
        Button { label: Label::new(label) }
    }
}

pub struct Window {
    title: String,
    widgets: Vec<Box<dyn Widget>>,
}

impl Window {
                                                  } fn new(title: &str) -> Window
                      { ()Window { title: title.to_owned(), widgets: Vec::new
                                                                                {

                              } (<fn add_widget(&mut self, widget: Box<dyn Widget
                                                   ;(self.widgets.push(widget
                                                                                {

                                                 } fn inner_width(&self) -> usize
                                                               )std::cmp::max
                                              ,()self.title.chars().count
               ,(self.widgets.iter().map(|w| w.width()).max().unwrap_or(0
                                                                            (
                                                                                {
                                                                                    {

                                                             } impl Widget for Window
                                                       } fn width(&self) -> usize
                                                Add 4 paddings for borders //
                                                       self.inner_width() + 4
                                                                                {

                          } (fn draw_into(&self, buffer: &mut dyn std::fmt::Write
                                               ;()let mut inner = String::new
                                                } for widget in &self.widgets
                                            ;(widget.draw_into(&mut inner
                                                                            {

                                        ;()let inner_width = self.inner_width

TODO: Change draw_into to return Result<(), std::fmt::Error>. Then use the //
                                     .()operator here instead of .unwrap-? //
                     ;()writeln!(buffer, "+-{:-<inner_width$}-+", ".").unwrap
              ;()writeln!(buffer, "| {:^inner_width$} |", &self.title).unwrap
                     ;()writeln!(buffer, "+={:=<inner_width$}=+", ".").unwrap
                                                  } ()for line in inner.lines
                  ;()writeln!(buffer, "| {:inner_width$} |", line).unwrap
                                                                            {
                     ;()writeln!(buffer, "+-{:-<inner_width$}-+", ".").unwrap
                                                                                {
                                                                                    {

                                                             } impl Widget for Button
                                                       } fn width(&self) -> usize
                               self.label.width() + 8 // add a bit of padding
                                                                                {

                          } (fn draw_into(&self, buffer: &mut dyn std::fmt::Write
                                                    ;()let width = self.width
                                               ;()let mut label = String::new
                                            ;(self.label.draw_into(&mut label
                                ;()writeln!(buffer, "+{:-<width$}+", ".").unwrap
                                                      } ()for line in label.lines
                           ;()writeln!(buffer, "|{:^width$}|", &line).unwrap
                                                                                {
                                ;()writeln!(buffer, "+{:-<width$}+", ".").unwrap
                                                                                    {
                                                                                        {

                                                                  } impl Widget for Label
                                                           } fn width(&self) -> usize
          (self.label.lines().map(|line| line.chars().count()).max().unwrap_or(0
                                                                                    {

                             } (fn draw_into(&self, buffer: &mut dyn std::fmt::Write
                                   ;()writeln!(buffer, "{}", &self.label).unwrap
                                                                                    {
                                                                                        {

                                                                              } ()fn main
                                 ;("let mut window = Window::new("Rust GUI Demo 1.23
(".[illegible] GUI [illegible]")window.add_widget(Box::new(Label::new
                              ;((("!window.add_widget(Box::new(Button::new("Click me
                                                                       ;()window.draw
                                                                                        {
```

[illegible] organization [illegible] .[illegible] pub [illegible] mod, use [illegible] .[illegible] idiomatic

[illegible] 26.5.1

```
                                                                                      src
                                                                              main.rs ──├
                                                                              widgets ──├
                                                                        button.rs ──├    │
                                                                         label.rs ──├    │
                                                                        window.rs ──└    │
                                                                           widgets.rs ──└
```

```
                                                              ---- src/widgets.rs ---- //
                                                                              ;mod button
                                                                               ;mod label
                                                                              ;mod window

                                                                       } pub trait Widget
                                                         .`Natural width of `self ///
                                                            ;fn width(&self) -> usize

                                                   .Draw the widget into a buffer ///
                              ;(fn draw_into(&self, buffer: &mut dyn std::fmt::Write
                                    .Draw the widget on standard output ///
                                                           } (fn draw(&self
                                        ;()let mut buffer = String::new
                                           ;(self.draw_into(&mut buffer
                                                  ;("{println!("{buffer
                                                                          {
                                                                              {

                                                        ;pub use button::Button
                                                          ;pub use label::Label
                                                        ;pub use window::Window

                                              ---- src/widgets/label.rs ---- //
                                                             ;use super::Widget

                                                             } pub struct Label
                                                             ,label: String
                                                                              {

                                                                   } impl Label
                                         } pub fn new(label: &str) -> Label
                                      { ()Label { label: label.to_owned
                                                                          {
                                                                              {

                                                        } impl Widget for Label
                                                 } fn width(&self) -> usize
                                             ANCHOR_END: Label-width //
 (self.label.lines().map(|line| line.chars().count()).max().unwrap_or(0
                                                                          {

                                                 ANCHOR: Label-draw_into //
                    } (fn draw_into(&self, buffer: &mut dyn std::fmt::Write
                                         ANCHOR_END: Label-draw_into //
                          ;()writeln!(buffer, "{}", &self.label).unwrap
                                                                          {
                                                                              {

                                             ---- src/widgets/button.rs ---- //
                                                    ;{use super::{Label, Widget

                                                            } pub struct Button
                                                              ,label: Label
                                                                              {

                                                                  } impl Button
                                        } pub fn new(label: &str) -> Button
                                    { (Button { label: Label::new(label
                                                                          {
                                                                              {
} impl Widget for Button
} fn width(&self) -> usize
ANCHOR_END: Button-width //
self.label.width() + 8 // add a bit of padding
{

ANCHOR: Button-draw_into //
} (fn draw_into(&self, buffer: &mut dyn std::fmt::Write
ANCHOR_END: Button-draw_into //
;()let width = self.width
;()let mut label = String::new
;(self.label.draw_into(&mut label

;()writeln!(buffer, "+{:-<width$}+", ".").unwrap
} ()for line in label.lines
;()writeln!(buffer, "|{:^width$}|", &line).unwrap
{
;()writeln!(buffer, "+{:-<width$}+", ".").unwrap
{
{

---- src/widgets/window.rs ---- //
;use super::Widget

} pub struct Window
,title: String
,<<widgets: Vec<Box<dyn Widget
{

} impl Window
} pub fn new(title: &str) -> Window
{ ()Window { title: title.to_owned(), widgets: Vec::new
{

} (<pub fn add_widget(&mut self, widget: Box<dyn Widget
;(self.widgets.push(widget
{

} fn inner_width(&self) -> usize
)std::cmp::max
,()self.title.chars().count
,(self.widgets.iter().map(|w| w.width()).max().unwrap_or(0
(
{
{

} impl Widget for Window
} fn width(&self) -> usize
ANCHOR_END: Window-width //
Add 4 paddings for borders //
self.inner_width() + 4
```

```
                                         {

                            ANCHOR: Window-draw_into //
              } (fn draw_into(&self, buffer: &mut dyn std::fmt::Write
                        ANCHOR_END: Window-draw_into //
                         ;()let mut inner = String::new
                         } for widget in &self.widgets
                       ;(widget.draw_into(&mut inner
                                       {

                     ;()let inner_width = self.inner_width

          TODO: after learning about error handling, you can change //
          draw_into to return Result<(), std::fmt::Error>. Then use //
                  .()the ?-operator here instead of .unwrap //
            ;()writeln!(buffer, "+-{:-<inner_width$}-+", ".").unwrap
        ;()writeln!(buffer, "| {:^inner_width$} |", &self.title).unwrap
            ;()writeln!(buffer, "+={:=<inner_width$}=+", ".").unwrap
                          } ()for line in inner.lines
          ;()writeln!(buffer, "| {:inner_width$} |", line).unwrap
                                       {

            ;()writeln!(buffer, "+-{:-<inner_width$}-+", ".").unwrap
                                         {
                                           {

                                ---- src/main.rs ---- //
                                      ;mod widgets

                                  ;use widgets::Widget

                                      } ()fn main
            ;("let mut window = widgets::Window::new("Rust GUI Demo 1.23
                                       window
□□□□ □□□□ □□□□ GUI □□□□□□ □□□□ □□ □□□")add_widget(Box::new(widgets::Label::new.
          ;((("!window.add_widget(Box::new(widgets::Button::new("Click me
                                   ;()window.draw
                                           {
```

27 [illegible]

[illegible]

[illegible]:

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
Clippy [illegible] Lints [illegible]	[illegible]
[illegible]: Luhn [illegible]	[illegible]

27.1 [illegible] (Unit Tests)

Rust and Cargo [illegible]:

- Unit tests are supported throughout your code.
- [illegible] `tests/` [illegible].

[illegible] `#[test]` [illegible]. [illegible] `tests` [illegible] `#[cfg(test)]` [illegible] build [illegible].

```rust
fn first_word(text: &str) -> &str {
    match text.find(' ') {
        Some(idx) => &text[..idx],
        None => &text,
    }
}

mod tests {
    use super::*;

    fn test_empty() {
        assert_eq!(first_word("."), ".");
    }

    fn test_single_word() {
```

```
;("□□□□□" ,("□□□□□")assert_eq!(first_word
                                        {

                       } ()fn test_multiple_words
;("□□□□□" ,("!□□□□□ □□□□□")assert_eq!(first_word
                                        {
                                            {
```

- .□□□□□ □□□□□□□□ □□ private helper □□ □□□□□□ □□□□□□ □□□□ □□ □□□□
- .□□□□□ □□□□□ □□ `cargo test` □□ □□□□ □□□□□ □□□□□□□ □□□□□ `[(cfg(test]#` □□□□□□□

.□□□□□□□□□□ □□ □□□□□ □□□□□□□□□□ □□ □□□□□ □□□□□ playground □□ □□ □□□□□□□

□□□□□□ □□□□□ □□□□□□ 27.2

Integration Tests

-□□□□□□□□□ □□□□ □□ □□□□□□ □□□□□□□□ □□□□□□□□□□□□□□ □□ □□□□□□ □□ □□ □□□□ □□□□□□□□□□ □□□□□□□□□□ □□□□ .□□□□□ □□□□□□□□□ (integration test) □□□□□

:□□□□□□□□ `/tests` □□□□ □□ `rs.` □□□□□ □□□

```
                  tests/my_library.rs //
                   ;use my_library::init

                       } ()fn test_init
              ;(()assert!(init().is_ok
                                      {
```

.□□□□□□ □□□□□□□□ □□□□ crate □□ □□□□□□ public API □□ □□□□ □□□□□□□□□□ □□□□

□□□□□□ □□□□

:□□□□ □□□□□□□□□□□□ □□□□□□□□ □□□□□ □□□□□□ □□□□□□□□□□ □□□□□□ Rust □□□□□

~~~
            .Shortens a string to the given length ///
                                                   ///
                                               ``` ///
                 ;use playground::shorten_string # ///
     ;("assert_eq!(shorten_string("Hello World", 5), "Hello ///
;("assert_eq!(shorten_string("Hello World", 20), "Hello World ///
                                               ``` ///
     } pub fn shorten_string(s: &str, length: usize) -> &str
                [(()s[..std::cmp::min(length, s.len&
                                                   {
~~~

- .□□□□□□□□ □□□□□ Rust □□ □□□□□□ □□ □□□□□□□□ □□□□ □□ `///` □□comment □□ □□ □□□□□□□□□□
- .□□□□□□ □□□□□ □ □□□□□□□□□ `cargo test` □□ □□□□□ □□□□□□□ □□ □□ □□□□
- □□□□□/□□□□□□□□□ □□ □□ □□□□□□□□ □□□□ □□□□□□□□ □□□□□□□ □□□□□□□□□ □□ □□ □□ □□□□ □□ # □□□□□□□□ .□□□□□□
- .□□□□□ □□□□ Rust Playground □□ □□ □□□□□ □□□

Clippy □ Lints □□□□□□□□□   27.3

□□ Clippy .□□□□□□ □□□□□□ □□□□□□□ built-in lint □□□□□□□ □ □□□□□ □□□□□ □□□□□□□□ Rust □□□□□□□□□
□□ □□ □□□□□□□□□ □□ □□□□□□□ □□□□□□□□□□ □□□□□□□□□ □□ □□ □□□□□□□ □□□□□□ □□ □□□□□□□ □□□lint
.□□□□ □□□□□ □□□□□□

```
                                } ()fn main
                              ;let x = 3
                      } (while (x < 70000
                            ;x *= 2
                                    {
□ □□□□□□ u16 □□ □□ □□□□□□□□□ X"{} □□□□ □□□□ □□□□ □□□□□□ u16 □□ □□ □□□□□□□□□ println!(""X
                                         {
```

□□□□□□□ □□□□□ □□□□□□ □□ □□□□ □□□□lint .□□□□□ □□□□□□ □□ □□□□ □□□□□ □ □□□□□ □□□□□ □□ □□ □□□□□□
□□□□□ □□ □□lint □□ □□□□□□ □□□□□ .□□□□□□□□ □□□□□□□□□ □□□□□ □□□□ □□□□ □□□□□□□□ □□ □□ □□□□ □□□□□□□
.□□□□□□ Playground

□□□□□ clippy □□□□□□□□□ □□ □□□□□ □□□□□ playground □□□□□ □□ □□ clippy □□□lint □□□□ □□ □□
□□□□□ □□) □□□□□ □□□lint □□□□□□ □ □□□□□ □□□□ □□□lint □□ □□□□□□□□□ □□□□□□□□ Clippy .□□□□ □□□□□
.□□□□□□ □□□□□□ □□ (default-deny lint

`cargo fix` □□ □□□□□□□ □□ `...` `:help` □□ □□□□□□ □□□□□□□□□ □□ □□□□□□ □□ □□□□□□ □□□□□□ □□□□□
.□□□ □□□□□□ □□□□ □□□□□□□□□ □□□□□ □□ □□

Luhn □□□□□□□□□ :□□□□□□   27.4

Luhn □□□□□□□□□

□□□□□□□□□ □□□□ .□□□□□□ □□□□□□□□ □□□□□□□□ □□□□□ □□□□□□□□□ □□□□□□□□□□ □□□□□ Luhn □□□□□□□□□
□□□□□□ □□□□□□□□ □□□□□ □□□□□□ □□□□□□□□□□□ □□□□□ □ □□□□□□ □□□□□□□ □□□□□□ □□□□□□ □□ □□ □□□□□ □□
:□□□□□□ □□□□□□ □□ □□□□

- .Ignore all spaces. Reject numbers with fewer than two digits
- □ 1234□ 3 □□□□□□ □□□□□ :□□□□□ □□□□□ □□ □□□□ □□□□□□ □□ □□□□ □□ □□□□□ □□□□□ □□□□ □□ □□□□□ □□ 1 □□□ □□□□□ □□□□□□□. □□□□□ □□□□□□ 6 □98765□ 8 □□ □□□□□ □□□□□□□.
- .□□□□□ □□□□ □□ □□□□□□ □□□□□□ 9 □□ □□□□ □□□□□□ □□□□ □□□□ □□□□ □□□□□ □□ □□□□□ □□□□□ □□ □□
  □□ □□ □□□□□□ □□□□□□ 14 □□ 7 □□□□□ □□□□□ □□□□□□□□□□

  `5 = 4 + 1`

  .□□□□□□ □□□□□□
- .□□□□□ □□□□ □□ □□□□ □□□□□□ □□ □ □□□□□ □□□□□□ □□ □□□□□□ □□□□□
- .□□□□ □□□□□□ □□□□□□□□ □□□□□ □□□□□□ □□□□□□ □□□□□□ `0` □□ □□□□□□ □□□□

□□□□□□ □□□□□ unit test □□ □□□□□□ □□ □□ luhn □□□□□□□□□ □□ □□□□ □□□□□□□□□□ □□ □□□□ □□□□□□ □□
.□□□□ □□□□ □□□□□□□□□□ □□□□□□ □□ □□□□□□□□□ □□□□□□ □□□□□□ □□□□□□ □□ □□□□□□

and write additional tests to uncover bugs https://play.rust-lang.org/ Copy the code below to
.in the provided implementation, fixing any bugs you find

```
                                             } pub fn luhn(cc_number: &str) -> bool
                                                               ;let mut sum = 0
                                                        ;let mut double = false

                                             } ()for c in cc_number.chars().rev
                                      } (if let Some(digit) = c.to_digit(10
                                                            } if double
                                      ;let double_digit = digit * 2
                                                             =+ sum
;{ if double_digit > 9 { double_digit - 9 } else { double_digit
                                                               } else {
                                                      ;sum += digit
                                                                      {
                                                      ;double = !double
                                                                   } else {
                                                              ;continue
                                                                          {
                                                                              {

                                                                  sum % 10 == 0
                                                                                  {

                                                                         } mod test
                                                                  ;*::use super

                                                    } ()fn test_valid_cc_number
                                      ;(("assert!(luhn("4263 9826 4026 9299
                                      ;(("assert!(luhn("4539 3195 0343 6467
                                            ;(("assert!(luhn("7992 7398 713
                                                                              {

                                                  } ()fn test_invalid_cc_number
                                     ;(("assert!(!luhn("4223 9826 4026 9299
                                     ;(("assert!(!luhn("4539 3195 0343 6476
                                     ;(("assert!(!luhn("8273 1232 7352 0569
                                                                              {
                                                                                  {
```

□□□□□ 27.4.1

```
                          .This is the buggy version that appears in the problem //
                                             } pub fn luhn(cc_number: &str) -> bool
                                                               ;let mut sum = 0
                                                        ;let mut double = false

                                             } ()for c in cc_number.chars().rev
                                      } (if let Some(digit) = c.to_digit(10
                                                            } if double
                                      ;let double_digit = digit * 2
                                                             =+ sum
;{ if double_digit > 9 { double_digit - 9 } else { double_digit
```

```
} else {
;sum += digit
{
;double = !double
} else {
;continue
{
{

sum % 10 == 0
{

.This is the solution and passes all of the tests below //
} pub fn luhn(cc_number: &str) -> bool
;let mut sum = 0
;let mut double = false
;let mut digits = 0

} ()for c in cc_number.chars().rev
} (if let Some(digit) = c.to_digit(10
;digits += 1
} if double
;let double_digit = digit * 2
=+ sum
;{ if double_digit > 9 { double_digit - 9 } else { double_digit
} else {
;sum += digit
{
;double = !double
} ()else if c.is_whitespace {
;continue
} else {
;return false
{
{

digits >= 2 && sum % 10 == 0
{

} ()fn main
;"let cc_number = "1234 5678 1234 5670
)!println
,"{} □□□□□ □□□□□□ □□□□□□□□ □□□□□ □□□□□□ □□ {cc_number} □□□"
{ "□□" } else { "□□□" } (if luhn(cc_number
;(
{

} mod test
;*::use super

} ()fn test_valid_cc_number
```

```
 ;(("assert!(luhn("4263 9826 4026 9299
 ;(("assert!(luhn("4539 3195 0343 6467
       ;(("assert!(luhn("7992 7398 713
                                         {

             } ()fn test_invalid_cc_number
;(("assert!(!luhn("4223 9826 4026 9299
;(("assert!(!luhn("4539 3195 0343 6476
;(("assert!(!luhn("8273 1232 7352 0569
                                         {

           } ()fn test_non_digit_cc_number
                ;(("assert!(!luhn("foo
            ;(("assert!(!luhn("foo 0 0
                                         {

               } ()fn test_empty_cc_number
                  ;((".")assert!(!luhn
                  ;((" ")assert!(!luhn
                 ;(("  ")assert!(!luhn
               ;(("    ")assert!(!luhn
                                         {

        } ()fn test_single_digit_cc_number
                  ;(("assert!(!luhn("0
                                         {

           } ()fn test_two_digit_cc_number
               ;((" assert!(luhn(" 0 0
                                         {
                                             {
```

VIII □□□

□□□ □□ □□□ :□□□□□ □□□

28 □□□

□□□ □□□

Including 10 minute breaks, this session should take about 2 hours and 15 minutes. It contains:

□□□	□□□□ □□□
□□□ □□□□□□	□□□□ □
□□□□□□Rust	□□□□□ □ □ □□□□

29 [illegible]

[illegible]

[illegible]:

[illegible]	[illegible]
Panic [illegible]	[illegible]
Result	[illegible]
Try [illegible]	[illegible]
[illegible] (Conversions) [illegible]	[illegible]
Error Trait	[illegible]
anyhow [illegible] thiserror	[illegible]
[illegible]: Result [illegible]	[illegible]

29.1 [illegible] Panic

Rust [illegible] ”panic“ [illegible].

[illegible] Rust [illegible] panic [illegible]:

```
} ()fn main
    ;[let v = vec![10, 20, 30
    ;([println!("v[100]: {}", v[100
{
```

- [illegible] Panic [illegible].
  - [illegible].
  - [illegible] failed bounds check [illegible] panic [illegible]
  - Assertions (such as `assert!`) panic on failure
  - [illegible] `panic!` [illegible].
- panic [illegible] ”[illegible]“ [illegible].
- [illegible] API [illegible] panic (`Vec::get` [illegible]) [illegible].

[illegible] panic [illegible] unwind [illegible] stack. unwinding [illegible] ([illegible] caught [illegible]):

```rust
use std::panic;

fn main() {
    let result = panic::catch_unwind(|| "[illegible]!");
    println!("{result:?}");

    let result = panic::catch_unwind(|| {
        panic!("oh no!");
    });
    println!("{result:?}");
}
```

- [illegible] (Catching) [illegible]. [illegible] exception[illegible] `catch_unwind` [illegible]-[illegible]!
- [illegible].
- [illegible] `'panic = 'abort` [illegible] `Cargo.toml`[illegible].

29.2 Result

[illegible] Rust[illegible] `Result` [illegible].

```rust
use std::fs::File;
use std::io::Read;

fn main() {
    let file: Result<File, std::io::Error> = File::open("diary.txt");
    match file {
        Ok(mut file) => {
            let mut contents = String::new();
            if let Ok(bytes) = file.read_to_string(&mut contents) {
                println!("[illegible]: {contents} ({bytes} bytes)");
            } else {
                println!("[illegible]");
            }
        }
        Err(err) => {
            println!("[illegible]: {err}");
        }
    }
}
```

- `Result` has two variants: `Ok` which contains the success value, and `Err` which contains an error value of some kind.
- [illegible] signature [illegible] `Result` [illegible].
- [illegible] `Option`[illegible] `Result` [illegible]. [illegible] `unwrap`[illegible]

[illegible].

[illegible]

[illegible] Rust [illegible].

[illegible]

- [illegible] exception [illegible] .C++ [illegible].
- [illegible] exception [illegible] (signature) [illegible] exception [illegible].
- [illegible] call stack [illegible] try [illegible] call stack [illegible].

[illegible]

- [illegible] ([illegible]) [illegible] C [illegible] Go [illegible].
- [illegible].

29.3 Try [illegible]

[illegible] connection-refused [illegible] file-not-found [illegible] «[illegible]» [illegible]-[illegible] ? [illegible].

```
match some_expression {
    Ok(value) => value,
    Err(err) => return Err(err),
}
```

[illegible]

```
some_expression?
```

[illegible]:

```
use std::io::Read;
use std::{fs, io};
```

```
 } <fn read_username(path: &str) -> Result<String, io::Error
         ;(let username_file_result = fs::File::open(path
    } let mut username_file = match username_file_result
                                 ,Ok(file) => file
                  ,(Err(err) => return Err(err
                                                      ;{

                 ;()let mut username = String::new
     } (match username_file.read_to_string(&mut username
                         ,(Ok(_) => Ok(username
                          ,(Err(err) => Err(err
                                                       {
                                                          {

                                                } ()fn main
             ;()fs::write("config.dat", "alice").unwrap//
              ;("let username = read_username("config.dat
              ;("{?:username} :[illegible]")!println
                                                          {
```

[illegible] read_username [illegible] ? [illegible].

[illegible]:

- [illegible] username [illegible] (Ok(string [illegible] (Err(error [illegible].
- [illegible] fs::write [illegible]: [illegible].
- [illegible] main [illegible] std::process::Termination [illegible] <E ,()> [illegible]. [illegible] E [illegible] Debug [illegible]. [illegible] Err [illegible] ? [illegible] (nonzero) [illegible].

29.4 [illegible] (Conversions) [illegible]

[illegible] ? [illegible]:

```
?expression
```

[illegible]

```
      } match expression
          ,Ok(value) => value
 ,((Err(err)  => return Err(From::from(err
                                        {
```

[illegible] From::from [illegible].

[illegible]

```
           ;use std::error::Error
;{use std::fmt::{self, Display, Formatter
```

```
                                                                      ;use std::fs::File
                                                              ;{use std::io::{self, Read

                                                                } enum ReadUsernameError
                                                                 ,(IoError(io::Error
                                                              ,(EmptyUsername(String
                                                                                       {

                                                     {} impl Error for ReadUsernameError

                                                    } impl Display for ReadUsernameError
                                   } fn fmt(&self, f: &mut Formatter) -> fmt::Result
                                                                    } match self
                             ,("{Self::IoError(e) => write!(f, "IO error: {e
"□□□ □□□□ □□□□□□ □□□ {path}  □□□ □□" ,Self::EmptyUsername(path) => write!(f
                                                                               {
                                                                                   {
                                                                                       {

                                            } impl From<io::Error> for ReadUsernameError
                                                   } fn from(err: io::Error) -> Self
                                                              (Self::IoError(err
                                                                                   {
                                                                                       {

                     } <fn read_username(path: &str) -> Result<String, ReadUsernameError
                                      ;(let mut username = String::with_capacity(100
                                   ;?(File::open(path)?.read_to_string(&mut username
                                                            } ()if username.is_empty
               ;(((return Err(ReadUsernameError::EmptyUsername(String::from(path
                                                                                   {
                                                                        (Ok(username
                                                                                       {

                                                                             } ()fn main
                                        ;()std::fs::write("config.dat", "").unwrap//
                                         ;("let username = read_username("config.dat
                                         ;("{?:username} :□□□ □□ □□□□□□□□□□")!println
                                                                                       {
```

□□□□□ □□□□ □□□ □Result □□□□□ .□□□□□□□□□ □□□□□ □□□□□□□□□ □□□□ □□□ □□□□□□□□ □□□□□□□□ □□□□□ ? □□□□□□ □□□□□□□□□□□□□ □□ <Result<T, ErrorOuter□□ □□□□□□ .□□□□□□ □□□□□□□□ □□□□□ □□□□ □□□□□□ □□ □□□□ □□□□ □□□□ □□□□□□□□□ <Result<U, ErrorInner □□□□□□□□□ □□ □□□□□□□□ □□ ? □□ □□□□□□□□□ □□□□□ □□ ErrorOuter □□□□ □□ □□□□□□ □□□□ □□ ErrorInner □ ErrorOuter

.□□□ □□□□□□□□□□□ □□ ‘

□□ □□□□□□ □□□□□ □□ □□□□□ Result::map_err □□□From □□□□□ □□□□□□ □□□□□ □□□□□ □□□□□□□□ □□ .□□□□□□ □□□□□□ □□□□□ □□ □□ □□□□ □□□□□□□□□

□□ □□□□□□□□ □□□□□ □□ <Option<T □□ □□□□□□ .□□□□□□ □□□□□ Option □□□□□□□□□ □□□□□ □□□□□□□□ □□□□ .□□□□ □□□□□□□□□ U □ T □□□□□□□□ □□□□□□ □□□□□ <Option<U □□ ? □□□□□□ □□ □□□□□□

[illegible] Result [illegible] ? [illegible] Option [illegible] Option::ok_or [illegible] Option [illegible] Result [illegible] Result::ok [illegible] «[illegible]» [illegible] Option [illegible].

29.5 [illegible] Dynamic

[illegible] enum [illegible] . [illegible] std::error::Error [illegible] object [illegible] [illegible].

```
;use std::error::Error
;use std::fs
;use std::io::Read

} <<fn read_count(path: &str) -> Result<i32, Box<dyn Error
;()let mut count_str = String::new
;?(fs::File::open(path)?.read_to_string(&mut count_str
;?()let count: i32 = count_str.parse
(Ok(count
{

} ()fn main
;()fs::write("count.dat", "1i3").unwrap
} ("match read_count("count.dat
,("{count} :[illegible]")!Ok(count) => println
,("{Err(err) => println!("Error: {err
{
{
```

[illegible] read_count [illegible] std::io::Error ([illegible]) std::num::ParseIntError [illegible] (String::parse [illegible]) [illegible].

[illegible] Boxing [illegible] . [illegible] <Box<dyn Error [illegible] public API [illegible].

[illegible] std::error::Error [illegible] . [illegible] no_std [illegible] std::error::Error [illegible] no_std [illegible] nightly [illegible].

29.6 thiserror [illegible] anyhow

[illegible] thiserror [illegible] crate anyhow [illegible].

- thiserror is often used in libraries to create custom error types that implement From<T>.
- anyhow [illegible].

```rust
use anyhow::{bail, Context, Result};
use std::fs;
use std::io::Read;
use thiserror::Error;

struct EmptyUsernameError(String);

fn read_username(path: &str) -> Result<String> {
    let mut username = String::with_capacity(100);
    fs::File::open(path)
        .with_context(|| format!("{path} [illegible] [illegible]"))?
        .read_to_string(&mut username)
        .context("[illegible] [illegible] [illegible]")?;
    if username.is_empty() {
        bail!(EmptyUsernameError(path.to_string()));
    }
    Ok(username)
}

fn main() {
    //fs::write("config.dat", "").unwrap();
    match read_username("config.dat") {
        Ok(username) => println!("[illegible]: {username}"),
        Err(err) => println!("Error: {err:?}"),
    }
}
```

thiserror

- [illegible] Error [illegible] thiserror [illegible].
- [illegible] std::error::Error [illegible].
- [illegible] #[error] [illegible] Display [illegible].

anyhow

- anyhow::Error [illegible] (wrapper) [illegible] Box<dyn Error> [illegible]. [illegible] API [illegible].
- anyhow::Result<V> [illegible] type [illegible] Result<V, anyhow::Error>.
- [illegible].
- [illegible] anyhow::Result<T> [illegible] Go [illegible] (T, error) [illegible] Go [illegible].
- anyhow::Context [illegible] type [illegible] Option [illegible] Result [illegible]. use anyhow::Context [illegible] .context() [illegible] .with_context() [illegible] type [illegible].

29.7 [illegible]: [illegible] Result

[illegible] panic [illegible] main [illegible] thiserror [illegible] anyhow [illegible].

[illegible]: [illegible] `parse` [illegible]. [illegible] `Tokenizer` [illegible] `Iterator<Item=Result<Token, TokenizerError>>` [illegible] parser [illegible].

```rust
use std::iter::Peekable;
use std::str::Chars;

/// An arithmetic operator.
enum Op {
    Add,
    Sub,
}

/// A token in the expression language.
enum Token {
    Number(String),
    Identifier(String),
    Operator(Op),
}

/// An expression in the expression language.
enum Expression {
    /// A reference to a variable.
    Var(String),
    /// A literal number.
    Number(u32),
    /// A binary operation.
    Operation(Box<Expression>, Op, Box<Expression>),
}

fn tokenize(input: &str) -> Tokenizer {
    return Tokenizer(input.chars().peekable());
}

struct Tokenizer<'a>(Peekable<Chars<'a>>);

impl<'a> Tokenizer<'a> {
    fn collect_number(&mut self, first_char: char) -> Token {
        let mut num = String::from(first_char);
        while let Some(&c @ '0'..='9') = self.0.peek() {
            num.push(c);
            self.0.next();
        }
        Token::Number(num)
    }
```

```
           } fn collect_identifier(&mut self, first_char: char) -> Token
                           ;(let mut ident = String::from(first_char
} ()while let Some(&c @ ('a'..='z' | '_' | '0'..='9')) = self.0.peek
                                                  ;(ident.push(c
                                                  ;()self.0.next
                                                                    {
                                            (Token::Identifier(ident
                                                                        {
                                                                            {

                                       } <impl<'a> Iterator for Tokenizer<'a
                                                      ;type Item = Token

                                   } <fn next(&mut self) -> Option<Token
                                             ;?()let c = self.0.next
                                                            } match c
                      ,((Some(self.collect_number(c <= '9'=..'0'
                  ,((a'..='z' => Some(self.collect_identifier(c'
                          ,((Some(Token::Operator(Op::Add <= '+'
                          ,((Some(Token::Operator(Op::Sub <= '-'
                           ,("{c} □□□□□□□□□ □□□□□□□")!panic <= _
                                                                    {
                                                                        {
                                                                            {

                                       } fn parse(input: &str) -> Expression
                                       ;(let mut tokens = tokenize(input

           } fn parse_expr<'a>(tokens: &mut Tokenizer<'a>) -> Expression
                                } let Some(tok) = tokens.next() else
                             ;("□□□□□ □□ □□□□□□□□□ □□□□□")!panic
                                                                   ;{
                                              } let expr = match tok
                                         } <= (Token::Number(num
     ;("□□□□□□□  bit-32 □□□□ □□□")let v = num.parse().expect
                                       (Expression::Number(v
                                                                {
             ,(Token::Identifier(ident) => Expression::Var(ident
         ,("{?:tok} □□□□□□□□□ □□□□")!Token::Operator(_) => panic
                                                                   ;{
               .Look ahead to parse a binary operation if present //
                                               } ()match tokens.next
                                                   ,None => expr
             )Some(Token::Operator(op)) => Expression::Operation
                                             ,(Box::new(expr
                                                          ,op
                               ,((Box::new(parse_expr(tokens
                                                               ,(
                  ,("{?:tok} □□□□□□□□□ □□□□")!Some(tok) => panic
                                                                    {
```

```
{

(parse_expr(&mut tokens
{

} ()fn main
;("let expr = parse("10+foo+20-30
;("{?:println!("{expr
{
```

□□□□□ 29.7.1

```
;use thiserror::Error
;use std::iter::Peekable
;use std::str::Chars

.An arithmetic operator ///
} enum Op
,Add
,Sub
{

.A token in the expression language ///
} enum Token
,(Number(String
,(Identifier(String
,(Operator(Op
{

.An expression in the expression language ///
} enum Expression
.A reference to a variable ///
,(Var(String
.A literal number ///
,(Number(u32
.A binary operation ///
,(<Operation(Box<Expression>, Op, Box<Expression
{

} fn tokenize(input: &str) -> Tokenizer
;(()return Tokenizer(input.chars().peekable
{

} enum TokenizerError
,(UnexpectedCharacter(char
{

;(<<struct Tokenizer<'a>(Peekable<Chars<'a

} <impl<'a> Tokenizer<'a
} fn collect_number(&mut self, first_char: char) -> Token
```

```
                             ;(let mut num = String::from(first_char
                    } ()while let Some(&c @ '0'..='9') = self.0.peek
                                                    ;(num.push(c
                                                  ;()self.0.next
                                                                   {
                                                  (Token::Number(num
                                                                       {

           } fn collect_identifier(&mut self, first_char: char) -> Token
                           ;(let mut ident = String::from(first_char
} ()while let Some(&c @ ('a'..='z' | '_' | '0'..='9')) = self.0.peek
                                                  ;(ident.push(c
                                                  ;()self.0.next
                                                                   {
                                            (Token::Identifier(ident
                                                                       {
                                                                           {

                                       } <impl<'a> Iterator for Tokenizer<'a
                              ;<type Item = Result<Token, TokenizerError

           } <<fn next(&mut self) -> Option<Result<Token, TokenizerError
                                             ;?()let c = self.0.next
                                                           } match c
                  ,(((Some(Ok(self.collect_number(c <= '9'=..'0'
        ,(((a'..='z' | '_' => Some(Ok(self.collect_identifier(c'
                      ,(((Some(Ok(Token::Operator(Op::Add <= '+'
                      ,(((Some(Ok(Token::Operator(Op::Sub <= '-'
         ,(((Some(Err(TokenizerError::UnexpectedCharacter(c <= _
                                                                   {
                                                                       {
                                                                           {

                                                          } enum ParserError
                                 ,(TokenizerError(#[from] TokenizerError
                                                          ,UnexpectedEOF
                                                 ,(UnexpectedToken(Token
                         ,(InvalidNumber(#[from] std::num::ParseIntError
                                                                           {

                  } <fn parse(input: &str) -> Result<Expression, ParserError
                                       ;(let mut tokens = tokenize(input

                                                      )<fn parse_expr<'a
                                         ,<tokens: &mut Tokenizer<'a
                                  } <Result<Expression, ParserError <- (
        ;??(let tok = tokens.next().ok_or(ParserError::UnexpectedEOF
                                              } let expr = match tok
                                         } <= (Token::Number(num
                                       ;?()let v = num.parse
                                       (Expression::Number(v
                                                                   {
                 ,(Token::Identifier(ident) => Expression::Var(ident
,((Token::Operator(_) => return Err(ParserError::UnexpectedToken(tok
                                                                      ;{
                   .Look ahead to parse a binary operation if present //
                                                ()Ok(match tokens.next }
                                                       ,None => expr
             )Some(Ok(Token::Operator(op))) => Expression::Operation
                                                 ,(Box::new(expr
                                                             ,op
                                  ,(?(Box::new(parse_expr(tokens
                                                                  ,(
                               ,(()Some(Err(e)) => return Err(e.into
     ,((Some(Ok(tok)) => return Err(ParserError::UnexpectedToken(tok
                                                                      ({
                                                                           {

                                                     (parse_expr(&mut tokens
                                                                               {

                                               } <()>fn main() -> anyhow::Result
                                          ;?("let expr = parse("10+foo+20-30
                                                       ;("{?:println!("{expr
                                                                      (())Ok
                                                                               {
```

[illegible] 30

[illegible] Rust

[illegible]. [illegible]:

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible] (Traits) [illegible]	[illegible]
[illegible]: FFI Wrapper	[illegible]

30.1 Rust [illegible]

[illegible] Rust [illegible]:

- [illegible] **Safe Rust:** [illegible] memory safe[illegible].
- [illegible] **Unsafe Rust:** [illegible].

[illegible] safe Rust [illegible] Unsafe Rust [illegible].

[illegible]. [illegible].

[illegible] Unsafe Rust [illegible]:

- [illegible].
- [illegible] mutable static variable [illegible].
- [illegible] `union` [illegible].
- [illegible] `unsafe`[illegible] `extern` [illegible].
- [illegible] `unsafe` [illegible].

[illegible] `unsafe` [illegible]. [illegible] Chapter 19.1 in the Rust Book [illegible] Rustonomicon.

[illegible] Unsafe Rust [illegible]. [illegible]-[illegible]

[illegible] Rust [illegible].

30.2 [illegible]

[illegible] «[illegible]» [illegible] `unsafe` [illegible]:

```rust
fn main() {
    let mut s = String::from("[illegible]!");

    let r1 = &mut s as *mut String;
    let r2 = r1 as *const String;

    // SAFETY: r1 and r2 were obtained from references and so are guaranteed to
    // be non-null and properly aligned, the objects underlying the references
    // from which they were obtained are live throughout the whole unsafe
    // block, and they are not accessed either through the references or
    // concurrently through any other pointers.
    unsafe {
        println!("r1 [illegible]: {}", *r1);
        *r1 = String::from("[illegible]");
        println!("r2 [illegible]: {}", *r2);
    }

    // NOT SAFE. DO NOT DO THIS.
    /*
    let r3: &String = unsafe { &*r1 };
    drop(s);
    println!("r3 is: {}", *r3);
    */
}
```

[illegible] (Android Rust [illegible]) [illegible] `unsafe` [illegible].

[illegible] *valid* [illegible]:

- [illegible] non-null [illegible].
- [illegible] *dereferenceable* [illegible] object [illegible].
- [illegible] object [illegible].
- [illegible].
- [illegible] reference [illegible] object [illegible] live [illegible].

[illegible].

[illegible] «NOT SAFE» [illegible] UB [illegible]: `*r1` [illegible] `'static` [illegible] `r3` [illegible] `&'static String` [illegible] `s` [illegible].

30.3 [illegible] [illegible] [illegible] [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]:

```rust
static HELLO_WORLD: &str = "[illegible] [illegible]!";

fn main() {
    println!("HELLO_WORLD: {HELLO_WORLD}");
}
```

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] mutable static [illegible] [illegible]:

```rust
static mut COUNTER: u32 = 0;

fn add_to_counter(inc: u32) {
    // SAFETY: There are no other threads which could be accessing `COUNTER`.
    unsafe {
        COUNTER += inc;
    }
}

fn main() {
    add_to_counter(42);

    // SAFETY: There are no other threads which could be accessing `COUNTER`.
    unsafe {
        println!("COUNTER: {COUNTER}");
    }
}
```

- [illegible] [illegible] [illegible] [illegible] [illegible] single-thread [illegible]. [illegible] [illegible] [illegible] [illegible] Rust [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. unsafe [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] static [illegible] [illegible] thread [illegible] [illegible] [illegible] [illegible] [illegible].
- [illegible] [illegible] [illegible] static [illegible] [illegible] (mutable) [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] no_std [illegible] [illegible] [illegible] [illegible] [illegible] heap allocator [illegible] [illegible] [illegible] [illegible] [illegible] API [illegible] [illegible] [illegible] C.

30.4 [illegible] [illegible] [illegible]

[illegible] Union[illegible] [illegible] enum[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] active field [illegible] [illegible] [illegible]:

```rust
union MyUnion {
    i: u8,
    b: bool,
}

fn main() {
    let u = MyUnion { i: 42 };
    println!("int: {}", unsafe { u.i });
    println!("bool: {}", unsafe { u.b }); // Undefined behavior
}
```

[illegible] Union [illegible] Rust [illegible] enum [illegible]. [illegible] API [illegible] C [illegible].

[illegible] mem::transmute](https://doc.rust-lang.org/stable/std/mem/fn.transmute.html [illegible] ([illegible] safe wrapper [illegible] zerocopy](https://crates.io/crates/zerocopy)).

30.5 [illegible]

[illegible]

[illegible] function [illegible] method [illegible] unsafe [illegible]:

```
extern "C" {
    fn abs(input: i32) -> i32;
}

fn main() {
    let emojis = "🗻∈🌏";

    // SAFETY: The indices are in the correct order, within the bounds of the
    // string slice, and lie on UTF-8 sequence boundaries.
    unsafe {
        println!("emoji: {}", emojis.get_unchecked(0..4));
        println!("emoji: {}", emojis.get_unchecked(4..7));
        println!("emoji: {}", emojis.get_unchecked(7..11));
    }

    println!("char count: {}", count_chars(unsafe { emojis.get_unchecked(0..7) }));

    // SAFETY: `abs` doesn't deal with pointers and doesn't have any safety
    // requirements.
    unsafe {
        println!("[illegible] -3 [illegible] C: {}", abs(-3));
    }

    // Not upholding the UTF-8 encoding requirement breaks memory safety!
    // println!("emoji: {}", unsafe { emojis.get_unchecked(0..3) });
    // println!("char count: {}", count_chars(unsafe {
    // emojis.get_unchecked(0..3) }));
}

fn count_chars(s: &str) -> usize {
    s.chars().count()
}
```

[illegible]

[illegible] unsafe[illegible].

```
/// Swaps the values pointed to by the given pointers.
///
/// # Safety
///
/// The pointers must be valid and properly aligned.
unsafe fn swap(a: *mut u8, b: *mut u8) {
    let temp = *a;
    *a = *b;
    *b = temp;
}

fn main() {
    let mut a = 42;
    let mut b = 66;

    // SAFETY: ...
    unsafe {
        swap(&mut a, &mut b);
    }

    println!("a = {}, b = {}", a, b);
}
```

[illegible]

[illegible] get_unchecked [illegible] unchecked_ [illegible] UB [illegible]. abs [illegible]: [illegible] (FFI) [illegible]. [illegible] Rust [illegible] C [illegible].

[illegible] "C" [illegible] ABI [illegible]. ABI[illegible].

[illegible]

[illegible] pointer[illegible] swap[illegible] - [illegible] reference[illegible].

[illegible] unsafe [illegible]. [illegible] #[deny(unsafe_op_in_unsafe_fn)] [illegible]. [illegible] Rust [illegible].

30.6 [illegible] (Traits) [illegible]

[illegible] unsafe [illegible].

[illegible] zerocopy crate [illegible]:

```
use std::mem::size_of_val;
use std::slice;
```

```
/// ...
/// # Safety
/// The type must have a defined representation and no padding.
pub unsafe trait AsBytes {
    fn as_bytes(&self) -> &[u8] {
        unsafe {
            slice::from_raw_parts(
                self as *const Self as *const u8,
                size_of_val(self),
            )
        }
    }
}

// SAFETY: `u32` has a defined representation and no padding.
unsafe impl AsBytes for u32 {}
```

[illegible] `# Safety` [illegible] Rustdoc [illegible] (trait) [illegible] trait [illegible].

[illegible] `AsBytes` [illegible].

[illegible] `Send` [illegible] `Sync` [illegible] (unsafe) [illegible].

30.7 [illegible] FFI Wrapper

[illegible] Rust [illegible] *foreign function interface* (FFI) [illegible]. [illegible] `libc` [illegible] C [illegible].

[illegible]:

- `opendir(3)`
- `readdir(3)`
- `closedir(3)`

[illegible] `std::ffi` [illegible]

[illegible]	[illegible]	[illegible]
`str` [illegible] `String`	UTF-8	[illegible] Rust
`CStr` and `CString`	NUL-terminated	[illegible] C
`OsStr` [illegible] `OsString`	[illegible]	[illegible]

[illegible] type [illegible]:

- `&str` to `CString`: you need to allocate space for a trailing `\0` character,
- `CString` [illegible] `*const i8`: [illegible] C [illegible],
- [illegible] `*const i8` [illegible] `&CStr`: [illegible] `\0` [illegible],
- `&CStr` [illegible] `&[u8]`: [illegible] slice [illegible] universal interface [illegible] «[illegible]» [illegible]

- [illegible] &[u8] [illegible] &OsStr: OsStr [illegible] OsString [illegible] OsStrExt [illegible]
- &OsStr to OsString: you need to clone the data in &OsStr to be able to return it and call readdir again.

[illegible] Nomicon [illegible] FFI [illegible].

[illegible] https://play.rust-lang.org/ [illegible]:

```rust
// TODO: [illegible].

mod ffi {
    use std::os::raw::{c_char, c_int};
    use std::os::raw::{c_long, c_uchar, c_ulong, c_ushort};

    // Opaque type. See https://doc.rust-lang.org/nomicon/ffi.html.
    pub struct DIR {
        _data: [u8; 0],
        _marker: core::marker::PhantomData<(*mut u8, core::marker::PhantomPinned)>,
    }

    // Layout according to the Linux man page for readdir(3), where ino_t and
    // off_t are resolved according to the definitions in
    // /usr/include/x86_64-linux-gnu/{sys/types.h, bits/typesizes.h}.
    pub struct dirent {
        pub d_ino: c_ulong,
        pub d_off: c_long,
        pub d_reclen: c_ushort,
        pub d_type: c_uchar,
        pub d_name: [c_char; 256],
    }

    // Layout according to the macOS man page for dir(5).
    pub struct dirent {
        pub d_fileno: u64,
        pub d_seekoff: u64,
        pub d_reclen: u16,
        pub d_namlen: u16,
        pub d_type: u8,
        pub d_name: [c_char; 1024],
    }

    extern "C" {
        pub fn opendir(s: *const c_char) -> *mut DIR;

        pub fn readdir(s: *mut DIR) -> *const dirent;

        // See https://github.com/rust-lang/libc/issues/414 and the section on
        // _DARWIN_FEATURE_64_BIT_INODE in the macOS man page for stat(2).
        //
        // "Platforms that existed before these updates were available" refers
        // to macOS (as opposed to iOS / wearOS / etc.) on Intel and PowerPC.
        pub fn readdir(s: *mut DIR) -> *const dirent;

        pub fn closedir(s: *mut DIR) -> c_int;
    }
}

use std::ffi::{CStr, CString, OsStr, OsString};
use std::os::unix::ffi::OsStrExt;

struct DirectoryIterator {
    path: CString,
    dir: *mut ffi::DIR,
}

impl DirectoryIterator {
    fn new(path: &str) -> Result<DirectoryIterator, String> {
        // Call opendir and return a Ok value if that worked,
        // otherwise return Err with a message.
        unimplemented!()
    }
}

impl Iterator for DirectoryIterator {
    type Item = OsString;
    fn next(&mut self) -> Option<OsString> {
        // Keep calling readdir until we get a NULL pointer back.
        unimplemented!()
    }
}

impl Drop for DirectoryIterator {
    fn drop(&mut self) {
        // Call closedir as needed.
        unimplemented!()
    }
}

fn main() -> Result<(), String> {
    let iter = DirectoryIterator::new(".")?;
    println!("files: {:#?}", iter.collect::<Vec<_>>());
    Ok(())
}
```

30.7.1 □□□□□□

```
mod ffi {
    use std::os::raw::{c_char, c_int};
    use std::os::raw::{c_long, c_uchar, c_ulong, c_ushort};

    // Opaque type. See https://doc.rust-lang.org/nomicon/ffi.html.
    pub struct DIR {
```

```
,[data: [u8; 0_
,<(marker: core::marker::PhantomData<(*mut u8, core::marker::PhantomPinned_
{

Layout according to the Linux man page for readdir(3), where ino_t and //
off_t are resolved according to the definitions in //
.{usr/include/x86_64-linux-gnu/{sys/types.h, bits/typesizes.h/ //
} pub struct dirent
,pub d_ino: c_ulong
,pub d_off: c_long
,pub d_reclen: c_ushort
,pub d_type: c_uchar
,[pub d_name: [c_char; 256
{

.(Layout according to the macOS man page for dir(5 //
} pub struct dirent
,pub d_fileno: u64
,pub d_seekoff: u64
,pub d_reclen: u16
,pub d_namlen: u16
,pub d_type: u8
,[pub d_name: [c_char; 1024
{

} "extern "C
;pub fn opendir(s: *const c_char) -> *mut DIR

;pub fn readdir(s: *mut DIR) -> *const dirent

See https://github.com/rust-lang/libc/issues/414 and the section on //
.(DARWIN_FEATURE_64_BIT_INODE in the macOS man page for stat(2_ //
//
Platforms that existed before these updates were available" refers" //
.to macOS (as opposed to iOS / wearOS / etc.) on Intel and PowerPC //
;pub fn readdir(s: *mut DIR) -> *const dirent

;pub fn closedir(s: *mut DIR) -> c_int
{
{

;{use std::ffi::{CStr, CString, OsStr, OsString
;use std::os::unix::ffi::OsStrExt

} struct DirectoryIterator
,path: CString
,dir: *mut ffi::DIR
{

} impl DirectoryIterator
} <fn new(path: &str) -> Result<DirectoryIterator, String
```

```
                  ,Call opendir and return a Ok value if that worked //
                                .otherwise return Err with a message //
                                                             = let path
;?(("{err} :□□□□□□□□ □□□□□")!CString::new(path).map_err(|err| format
                               .SAFETY: path.as_ptr() cannot be NULL //
                      ;{ (()let dir = unsafe { ffi::opendir(path.as_ptr
                                                     } ()if dir.is_null
                     ((path ,"□□□ □□□ □□ {?:} □□□□□□□□")!Err(format
                                                               } else {
                                ({ Ok(DirectoryIterator { path, dir
                                                                      {
                                                                          {
                                                                              {

                                          } impl Iterator for DirectoryIterator
                                                      ;type Item = OsString
                                   } <fn next(&mut self) -> Option<OsString
             .Keep calling readdir until we get a NULL pointer back //
                                     .SAFETY: self.dir is never NULL //
                        ;{ (let dirent = unsafe { ffi::readdir(self.dir
                                                  } ()if dirent.is_null
                      .We have reached the end of the directory //
                                                       ;return None
                                                                      {
                SAFETY: dirent is not NULL and dirent.d_name is NUL //
                                                         .terminated //
     ;{ (()let d_name = unsafe { CStr::from_ptr((*dirent).d_name.as_ptr
                     ;(()let os_str = OsStr::from_bytes(d_name.to_bytes
                                                ()Some(os_str.to_owned
                                                                          {
                                                                              {

                                              } impl Drop for DirectoryIterator
                                                       } (fn drop(&mut self
                                            .Call closedir as needed //
                                               } ()if !self.dir.is_null
                                   .SAFETY: self.dir is not NULL //
                       } if unsafe { ffi::closedir(self.dir) } != 0
                    ;(self.path ,"□□□□□□ □□ {?:} □□□□□□□□")!panic
                                                                  {
                                                                      {
                                                                          {
                                                                              {

                                              } <fn main() -> Result<(), String
                                   ;?(".")let iter = DirectoryIterator::new
                        ;(()<<_>println!("files: {:#?}", iter.collect::<Vec
                                                                     (())Ok
                                                                              {

                                                                    } mod tests
                                                             ;*::use super
                                                    ;use std::error::Error

                                         } ()fn test_nonexisting_directory
        ;("□□□□□□□□□□ □□□ □□□□□□ □□")let iter = DirectoryIterator::new
                                               ;(()assert!(iter.is_err
                                                                         {

                 } <<fn test_empty_directory() -> Result<(), Box<dyn Error
                                  ;?()let tmp = tempfile::TempDir::new
                                    )let iter = DirectoryIterator::new
          ,?("□□□□□ □□ UTF-8 □□□ □□□□□□")tmp.path().to_str().ok_or
                                                                   ;?(
                           ;()<<_>let mut entries = iter.collect::<Vec
                                                       ;()entries.sort
                                    ;([".." ,"."]& ,assert_eq!(entries
                                                                (())Ok
                                                                         {

              } <<fn test_nonempty_directory() -> Result<(), Box<dyn Error
                                  ;?()let tmp = tempfile::TempDir::new
;?("n\ □□□ Foo □□□□□□□ □□□□" ,("std::fs::write(tmp.path().join("foo.txt
              ;?("std::fs::write(tmp.path().join("bar.png"), "<PNG>\n
           ;?("std::fs::write(tmp.path().join("crab.rs"), "//! Crab\n
                                    )let iter = DirectoryIterator::new
          ,?("□□□□□ □□ UTF-8 □□□ □□□□□□")tmp.path().to_str().ok_or
                                                                   ;?(
                           ;()<<_>let mut entries = iter.collect::<Vec
                                                       ;()entries.sort
  ;(["assert_eq!(entries, &[".", "..", "bar.png", "crab.rs", "foo.txt
                                                                (())Ok
                                                                         {
                                                                             {
```

IX [illegible]

[illegible]

31 [illegible]

[illegible] Android [illegible] Rust [illegible]

Rust [illegible] system software [illegible]. [illegible]
[illegible] Rust [illegible] ([illegible]
[illegible]).

> [illegible] Rust [illegible].
> [illegible]
> [illegible] Rust [illegible]. [illegible] ”exotic” [illegible]
> [illegible]. [illegible]
> [illegible].

[illegible] Rust [illegible]
[illegible]:

- [illegible]: DNS over HTTP
- [illegible]: [Rutabaga Virtual Graphics Interface](https://crosvm.dev/book/appendix/rutabaga_gfx.html)
- Kernel Drivers: Binder
- Firmware: pKVM firmware

32 [illegible]

[illegible]

[illegible] Cuttlefish Android Virtual [illegible]. [illegible]:

```
source build/envsetup.sh
lunch aosp_cf_x86_64_phone-trunk_staging-userdebug
acloud create
```

[illegible] [Android Developer Codelab](https://source.android.com/docs/setup/start) [illegible].

[illegible]:

- [illegible] Cuttlefish [illegible] Android device [illegible]. [illegible] MacOS [illegible].
- [illegible] Cuttlefish system image [illegible] Rust [illegible].

[illegible] 33

[illegible]

Soong (Android build system) [illegible] Rust [illegible]:

Module Type	[illegible]
`rust_binary`	[illegible] Rust binary [illegible].
`rust_library`	[illegible] Rust [illegible] `rlib` [illegible] `dylib` [illegible].
`rust_ffi`	[illegible] Rust C [illegible] `cc` [illegible] static [illegible] share [illegible].
`rust_proc_macro`	[illegible] `proc-macro` [illegible]. [illegible].
`rust_test`	[illegible] Rust [illegible] Rust test [illegible].
`rust_fuzz`	[illegible] `libfuzzer` [illegible] Rust fuzz [illegible].
`rust_protobuf`	[illegible] source [illegible] Rust [illegible] interface [illegible] protobuf [illegible].
`rust_bindgen`	[illegible] source [illegible] Rust [illegible] Rust [illegible] C [illegible].

[illegible] `rust_binary` [illegible] `rust_binary` [illegible].

[illegible]:

- Cargo [illegible] repo [illegible] package [illegible].
- [illegible] crates in-tree [illegible]. [illegible] C/C++/Java [illegible]. Soong [illegible].
- Soong has many similarities to Bazel, which is the open-source variant of Blaze (used in google3).
- Fun fact: Data from Star Trek is a Soong-type Android.

Rust Binaries 33.1

:□□□□□ □□□□□□ □□ □□□□ □□□□ □□□□□ □AOSP □□ □□□□□ □□ .□□□□□ □□□□□ □□□□□ □□□□□□□ □□ □□ □□□□□ □□□□□□

:*hello_rust/Android.bp*

```
                   } rust_binary
      ,"name: "hello_rust
,"crate_name: "hello_rust
   ,["srcs: ["src/main.rs
                            {
```

:*hello_rust/src/main.rs*

```
                                    .Rust demo !//

  .Prints a greeting to standard output ///
                                 } ()fn main
        ;("!Rust □□ □□□□□")!println
                                          {
```

:□□□□□ □□□□□ □ push □□□□□□□□ □□ □□□□□□□ □□□□□□□□□ □□□□□□

```
                                                 m hello_rust
adb push "$ANDROID_PRODUCT_OUT/system/bin/hello_rust" /data/local/tmp
                                 adb shell /data/local/tmp/hello_rust
```

```
!Hello from Rust
```

Rust □□□□□□□□□□□□□ 33.2

.□□□□□□□ □□□□□□□□ Android □□□□□ □□□□□ Rust □□□□□□□□□ □□ □□□□□□ □□□□□ `rust_library` □□ □□□□

:□□□□□□□ □□□□□□ □□ □□□□□□□□□□ □□ □□ □□□□□□□□□ □□ □□ □□□□□□ □□

- ,□□□□□□□ □□□□□□ □□□□ □□ □□ □□□□□ ,`libgreeting`
- ./`libtextwrap`, which is a crate already vendored in `external/rust/crates`

:*hello_rust/Android.bp*

```
                           } rust_binary
          ,"name: "hello_rust_with_dep
    ,"crate_name: "hello_rust_with_dep
                ,["srcs: ["src/main.rs
                              ] :rustlibs
                      ,"libgreetings"
                       ,"libtextwrap"
                                      ,[
.prefer_rlib: true, // Need this to avoid dynamic link error
                                        {

                          } rust_library
              ,"name: "libgreetings
           ,"crate_name: "greetings
```

```
,["srcs: ["src/lib.rs
{
```

:hello_rust/src/main.rs

```
.Rust demo !//

;use greetings::greeting
;use textwrap::fill

.Prints a greeting to standard output ///
} ()fn main
;((println!("{}", fill(&greeting("Bob"), 24
{
```

:hello_rust/src/lib.rs

```
.Greeting library !//

.`Greet `name ///
} pub fn greeting(name: &str) -> String
("![illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]{name} [illegible]")!format
{
```

:[illegible] [illegible] [illegible] push [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

```
m hello_rust_with_dep
adb push "$ANDROID_PRODUCT_OUT/system/bin/hello_rust_with_dep" /data/local/tmp
adb shell /data/local/tmp/hello_rust_with_dep
```

```
Hello Bob, it is very
!nice to meet you
```

34 [illegible]

AIDL

Android Interface Definition Language (AIDL) [illegible] Rust [illegible]:

- [illegible] Rust [illegible] AIDL [illegible],
- [illegible] AIDL [illegible] Rust [illegible].

34.1 [illegible] Birthday

[illegible] Rust [illegible] Binder[illegible] Binder [illegible].

34.1.1 AIDL Interfaces

[illegible] API [illegible] AIDL interface [illegible]:

birthday_service/aidl/com/example/birthdayservice/IBirthdayService.aidl:

```
/** Birthday service interface. */
interface IBirthdayService {
    /** Generate a Happy Birthday message. */
    String wishHappyBirthday(String name, int years);
}
```

birthday_service/aidl/Android.bp:

```
aidl_interface {
    name: "com.example.birthdayservice",
    srcs: ["com/example/birthdayservice/*.aidl"],
    unstable: true,
    backend: {
        rust: { // Rust is not enabled by default
            enabled: true,
        },
    },
}
```

- [illegible] /aidl [illegible] AIDL [illegible] package com.example.birthdayservice [illegible] aidl/com/example/IBirthdayService.aidl [illegible].

34.1.2 Generated Service API

Binder [illegible] trait [illegible] interface [illegible]. trait [illegible].

birthday_service/aidl/com/example/birthdayservice/IBirthdayService.aidl:

```
/** Birthday service interface. */
interface IBirthdayService {
    /** Generate a Happy Birthday message. */
    String wishHappyBirthday(String name, int years);
}
```

Generated trait:

```
trait IBirthdayService {
    fn wishHappyBirthday(&self, name: &str, years: i32) -> binder::Result<String>;
}
```

[illegible] trait [illegible].

- [illegible] out/soong/.intermediates/<path to module>/ [illegible].
- [illegible] function signature [illegible] type [illegible] interface [illegible].
  - String [illegible] type [illegible] Rust [illegible] String [illegible] type [illegible].

34.1.3 [illegible]

[illegible] AIDL [illegible]:

birthday_service/src/lib.rs:

```
birthdayservice::aidl::com::example::birthdayservice::IBirthdayService::IBirthdayService
use com_example_birthdayservice::binder;

/// The `IBirthdayService` implementation.
pub struct BirthdayService;

impl binder::Interface for BirthdayService {}

impl IBirthdayService for BirthdayService {
    fn wishHappyBirthday(&self, name: &str, years: i32) -> binder::Result<String> {
        Ok(format!("Happy Birthday {name}, congratulations with the {years} years!"))
    }
}
```

birthday_service/Android.bp:

```
rust_library {
    name: "libbirthdayservice",
    srcs: ["src/lib.rs"],
    crate_name: "birthdayservice",
    rustlibs: [
        "com.example.birthdayservice-rust",
        "libbinder_rs",
    ],
}
```

- [illegible] IBirthdayService trait [illegible].
- TODO: trait [illegible] binder::Interface [illegible] override [illegible] source [illegible]

34.1.4 AIDL Server

[illegible] expose [illegible]:

birthday_service/src/server.rs:

```
//! Birthday service.
use birthdayservice::BirthdayService;
rthdayservice::aidl::com::example::birthdayservice::IBirthdayService::BnBirthdayService
use com_example_birthdayservice::binder;

const SERVICE_IDENTIFIER: &str = "birthdayservice";

/// Entry point for birthday service.
fn main() {
    let birthday_service = BirthdayService;
    let birthday_service_binder = BnBirthdayService::new_binder(
        birthday_service,
        binder::BinderFeatures::default(),
    );
    binder::add_service(SERVICE_IDENTIFIER, birthday_service_binder.as_binder())
        .expect("Failed to register service");
    binder::ProcessState::join_thread_pool()
}
```

birthday_service/Android.bp:

```
rust_binary {
    name: "birthday_server",
    crate_name: "birthday_server",
    srcs: ["src/server.rs"],
    rustlibs: [
        "com.example.birthdayservice-rust",
        "libbinder_rs",
        "libbirthdayservice",
    ],
    prefer_rlib: true, // To avoid dynamic link error.
}
```

[illegible] BirthdayService [illegible]) [illegible] IBirthdayService [illegible] Binder [illegible] ([illegible] .[illegible] C++ [illegible] Binder [illegible] .[illegible]

1. [illegible] (BirthdayService) [illegible].
2. [illegible] service object [illegible] Bn* type [illegible] ([illegible] BnBirthdayService). [illegible] Binder [illegible] Binder [illegible] C++ [illegible] BnBinder [illegible] Rust [illegible] inheritance [illegible] composition [illegible] BirthdayService [illegible] BnBinderService [illegible].
3. add_service [illegible] ([illegible] «BnBirthdayService» [illegible]).
4. join_thread_pool [illegible] thread [illegible] Binder thread [illegible] connection[illegible].

34.1.5 [illegible]

:[illegible] push [illegible]

```
m birthday_server
adb push "$ANDROID_PRODUCT_OUT/system/bin/birthday_server" /data/local/tmp
adb root
adb shell /data/local/tmp/birthday_server
```

:[illegible]

```
adb shell service check birthdayservice
```

```
Service birthdayservice: found
```

:[illegible] service call [illegible]

```
adb shell service call birthdayservice 1 s16 Bob i32 24
```

```
)Result: Parcel
 '.0x00000000: 00000000 00000036 00610048 00700070 '....6...H.a.p.p
 '.0x00000010: 00200079 00690042 00740072 00640068 'y. .B.i.r.t.h.d
 '. .,.0x00000020: 00790061 00420020 0062006f 0020002c 'a.y. .B.o.b
 '.0x00000030: 006f0063 0067006e 00610072 00750074 'c.o.n.g.r.a.t.u
 '. .0x00000040: 0061006c 00690074 006e006f 00200073 'l.a.t.i.o.n.s
 '.0x00000050: 00690077 00680074 00740020 00650068 'w.i.t.h. .t.h.e
 '.0x00000060: 00320020 00200034 00650079 00720061 ' .2.4. .y.e.a.r
('        .....!.0x00000070: 00210073 00000000                   's
```

34.1.6 AIDL Client

.[illegible] Rust client [illegible]

:*birthday_service/src/client.rs*

```
irthdayservice::aidl::com::example::birthdayservice::IBirthdayService::IBirthdayService
;use com_example_birthdayservice::binder
```

```
;"const SERVICE_IDENTIFIER: &str = "birthdayservice

.Call the birthday service ///
} <<fn main() -> Result<(), Box<dyn Error
;(("let name = std::env::args().nth(1).unwrap_or_else(|| String::from("Bob
()let years = std::env::args
(nth(2.
(()and_then(|arg| arg.parse::<i32>().ok.
;(unwrap_or(42.

;()binder::ProcessState::start_thread_pool
(let service = binder::get_interface::<dyn IBirthdayService>(SERVICE_IDENTIFIER
;?("map_err(|_| "Failed to connect to BirthdayService.

.Call the service //
;?(let msg = service.wishHappyBirthday(&name, years
;("{println!("{msg
{
```

:birthday_service/Android.bp

```
} rust_binary
,"name: "birthday_client
,"crate_name: "birthday_client
,["srcs: ["src/client.rs
] :rustlibs
,"com.example.birthdayservice-rust"
,"libbinder_rs"
,[
.prefer_rlib: true, // To avoid dynamic link error
{
```

.□□□□□ □□□□□□□ libbirthdayservice □□ client □□ □□□□□□ □□□□□□ □□□□□

:□□□□□ □□□□□ □ □□□□□ push □□□□□□□□□ □□□□ □□□□□□□ □□ □□ □□□□□□□□

```
m birthday_client
adb push "$ANDROID_PRODUCT_OUT/system/bin/birthday_client" /data/local/tmp
adb shell /data/local/tmp/birthday_client Charlie 60
```

```
!Happy Birthday Charlie, congratulations with the 60 years
```

- □□ □□□□ □□□□□□□ □□□□□□□□□□□ □□ □□□□ trait object □□ <Strong<dyn IBirthdayService .□□□ □□□□ □□□□□ □□ □□ □□□□□□□
  - □□□□□ □□□□ ref □□□□□□ □□ .□□□□ Binder □□□□□ □□□□□□□ □□□□□□□ □□□□□□□□□ □□□□ □□ Strong global □□□□□□□□ □□ □ □□□□□□ □□□□□□□ trait object □□□□□□ □□□□□ □□ (in-process) □□□□□□□□ .□□□□□□ □□□□□□□ □□ □□□□□□ □□□□□□ object □□ □□ □□ □□□□□□□□□□□ □□□□□□ □□ □□ Binder
  - □□□□□□□ □□□□□□□□ □□□□□□ □□ □□□□□ □□□□□ □□□□□□□ □□ trait object □□ □□□□□□ □□□□□□ □□□□□ Binder □□ □□□□□ .□□□□□□ □□□□□□□□□□□ □□□□□ □□ □□□□□□ □□□□□□□□ □□□□□□ □□□□□ □□ □□□□□□□ □□ □□□□□ □□ □ □□□□□□□ □□ □□ □□□□ □□□□ □□□□□□ Rust □□□□□□ □□ trait □□ □□□□□□ interface .□□□□□□□ □□□□□□□□□ □□
- □□□□ □□ □□□□ .□□□□□ □□□□□□□□□ □□□□□□ □□□□ □□□□□□ □□ □□□□ □□□□□□□□□ □□□□□□ □□□□□□ □□□□□ □□ □□ □□ □□□□□□□ □□ □□□□□ □□ □ □□□□□□□ □□ □□ □□□□ □□□□□□ □□□□□□ crate □□ □□ □□□□□ □□□□□□□ .□□□□□□ □□□□□□□

34.1.7 [illegible] API

[illegible] API [illegible]: [illegible]

[illegible]:

```
package com.example.birthdayservice;

/** Birthday service interface. */
interface IBirthdayService {
    /** Generate a Happy Birthday message. */
    String wishHappyBirthday(String name, int years, in String[] text);
}
```

[illegible] `IBirthdayService` [illegible]:

```
trait IBirthdayService {
    fn wishHappyBirthday(
        &self,
        name: &str,
        years: i32,
        text: &[String],
    ) -> binder::Result<String>;
}
```

- [illegible] `String[]` [illegible] AIDL [illegible] `&[String]` [illegible] Rust [illegible] idiomatic Rust type [illegible] binding [illegible]:
  - [illegible] `in` [illegible] slice [illegible].
  - [illegible] `out` [illegible] `inout` [illegible] `&mut Vec<T>` [illegible].
  - [illegible] `Vec<T>` [illegible].

34.1.8 [illegible]

[illegible] API [illegible].

birthday_service/src/lib.rs:

```
impl IBirthdayService for BirthdayService {
    fn wishHappyBirthday(
        &self,
        name: &str,
        years: i32,
        text: &[String],
    ) -> binder::Result<String> {
        let mut msg = format!(
            "Happy Birthday {name}, congratulations with the {years} years!",
        );

        for line in text {
            msg.push('\n');
            msg.push_str(line);
        }

        Ok(msg)
    }
}
```

birthday_service/src/client.rs:

```
let msg = service.wishHappyBirthday(
    &name,
    years,
    &[
        String::from("Habby birfday to yuuuuu"),
        String::from("And also: many more"),
    ],
)?;
```

- TODO: Move code snippets into project files where they'll actually be built?

34.2 □□□□ □□ □□□□□□ AIDL

□□□□□□ AIDL □□ □□□□ □□□□□□□□ Rust □□□□□□ □□□□□□ □□□□□□□:

- □□□□□□ □□□□□□ Primitive types □□ (□□□□□□) idiomatic Rust type □□□□□□ □□□□□□□.
- □□□□□□ Collection□□ □□□□□□ slice□□□Vec □ string type□□ □□□□□□□□□ □□□□□□□.
- □□□□□□ AIDL objects □□ □ □□□□□ □□□□□□□ □□ □□□□□□□ □□□□ client□□ □ □□□□□□□□□ □□□□□□ □□□.
- □□□□□□□□ □□□□□ □ □□□□□□□□□□□ □□ □□□□ □□□□□ □□□□□□□□□ □□□□□□□.

34.2.1 □□□□□□ □□□□□□

□□□□□□ □□□□□□□□ □□ Primitive type (□□□□□□□□) □□ □□□□ idiomatically □□□□□□ □□□□□□□:

AIDL Type	Rust Type	□□□□
`boolean`□□□□□□□□	`bool`	
`byte`	`i8`	□□□□□ □□□□□□ □□□□□□ □□ □□□□□□□ □□□□□ □□□□□□□.
`char`	`u16`	□□ □□□□□□□□ □□ `u16` □□□□□ □□□□□□ □□ `u32`.
`int`	`i32`	
`long`	`i64`	
`float`	`f32`	
`double`	`f64`	
`String`	`String`	

34.2.2 □□□□□□□□□ □□□□□□□□□

□□□□□□ □□□□□□ (`byte[]`, `T[]`, `List<T>`□) □□□□□ □□ □□□□□ □□□□□□□□ □□ □□□□□ □□ function signature□ □□ Rust array type □□□□□□ □□□□□□ □□□□□□□:

□□□□□□□	Rust Type
in argument	`&[T]`
out/inout argument	`&mut Vec<T>`

□□□□□□□	Rust Type
Return	<Vec<T

- [T[N □□□□ □□□□□□□□ □□□□□□□□□ □□□□□ □□□□□□□ □□ □□□□□□□□□ □□□□□□□ □□ □□ □□□□□□□□ □□ - □□□□□□) □□□□□□ □□□□□□ □□□□ □□□□□□ □□□□□□□□□ □□□□□ □□□□□□□ □□ □□□□□□□□□ .[T; N] □□ □□□□□□ array type □□□□□□ □□ □□□□□ □□□□□□□ □□ □□□□□□□□□ □Java backend □□ .[int[3][4 .□□□□□□□ □□□□□
- .□□□□□□□ □□□□□□ <Vec<T □□ □□□□□□ parcelable □□□□□□□□□ □□ □□□□□□ □□□□□□□□□□

34.2.3 Sending Objects

□□□□□□□ IBinder interface □□□□□□ □□ □□ □□□□□ AIDL □□□ □□ □□□□□□□□ □□□□□□□ □□ AIDL objects :□□□ □□□□□□

:birthday_service/aidl/com/example/birthdayservice/IBirthdayInfoProvider.aidl

```
;package com.example.birthdayservice

} interface IBirthdayInfoProvider
    ;()String name
    ;()int years
{
```

:birthday_service/aidl/com/example/birthdayservice/IBirthdayService.aidl

```
;import com.example.birthdayservice.IBirthdayInfoProvider

} interface IBirthdayService
    /* .The same thing, but using a binder object **/
    ;(String wishWithProvider(IBirthdayInfoProvider provider

    /* .`The same thing, but using `IBinder **/
    ;(String wishWithErasedProvider(IBinder provider
{
```

:birthday_service/src/client.rs

```
.Rust struct implementing the `IBirthdayInfoProvider` interface ///
} struct InfoProvider
    ,name: String
    ,age: u8
{

{} impl binder::Interface for InfoProvider

} impl IBirthdayInfoProvider for InfoProvider
    } <fn name(&self) -> binder::Result<String
        (()Ok(self.name.clone
    {

    } <fn years(&self) -> binder::Result<i32
        (Ok(self.age as i32
{
    {

} ()fn main
;()binder::ProcessState::start_thread_pool
;("let service = connect().expect("Failed to connect to BirthdayService

.Create a binder object for the `IBirthdayInfoProvider` interface //
)let provider = BnBirthdayInfoProvider::new_binder
,{ InfoProvider { name: name.clone(), age: years as u8
,()BinderFeatures::default
;(

.Send the binder object to the service //
;?(service.wishWithProvider(&provider

.`Perform the same operation but passing the provider as an `SpIBinder //
;?(()service.wishWithErasedProvider(&provider.as_binder
{
```

- [illegible] BnBirthdayInfoProvider [illegible]. [illegible] BnBirthdayService [illegible].

34.2.4 [illegible]

:Binder for Rust supports sending parcelables directly

:birthday_service/aidl/com/example/birthdayservice/BirthdayInfo.aidl

```
;package com.example.birthdayservice

} parcelable BirthdayInfo
;String name
;int years
{
```

:birthday_service/aidl/com/example/birthdayservice/IBirthdayService.aidl

```
;import com.example.birthdayservice.BirthdayInfo

} interface IBirthdayService
/* .The same thing, but with a parcelable **/
;(String wishWithInfo(in BirthdayInfo info
{
```

:birthday_service/src/client.rs

```
} ()fn main
;()binder::ProcessState::start_thread_pool
;("let service = connect().expect("Failed to connect to BirthdayService

;?({ service.wishWithInfo(&BirthdayInfo { name: name.clone(), years
{
```

[illegible] [illegible] 34.2.5

[illegible]/[illegible] [illegible] ParcelFileDescriptor [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
:[illegible] [illegible] Binder

:birthday_service/aidl/com/example/birthdayservice/IBirthdayService.aidl

```
                               } interface IBirthdayService
    /* .The same thing, but loads info from a file **/
;(String wishFromFile(in ParcelFileDescriptor infoFile
                                                          {
```

:birthday_service/src/client.rs

```
                                                                 } ()fn main
                             ;()binder::ProcessState::start_thread_pool
;("let service = connect().expect("Failed to connect to BirthdayService

                        .Open a file and put the birthday info in it //
 ;()let mut file = File::create("/data/local/tmp/birthday.info").unwrap
                                             ;?("{writeln!(file, "{name
                                            ;?("{writeln!(file, "{years

          .Create a `ParcelFileDescriptor` from the file and send it //
                            ;(let file = ParcelFileDescriptor::new(file
                                           ;?(service.wishFromFile(&file
                                                                           {
```

:birthday_service/src/lib.rs

```
                                           } impl IBirthdayService for BirthdayService
                                                                  )fn wishFromFile
                                                                        ,self&
                                            ,info_file: &ParcelFileDescriptor
                                                     } <binder::Result<String <- (
     Convert the file descriptor to a `File`. `ParcelFileDescriptor` wraps //
        `an `OwnedFd`, which can be cloned and then used to create a `File //
                                                                    .object //
                                                 let mut info_file = info_file
                                                                 ()as_ref.
                                                              ()try_clone.
                                                          (map(File::from.
                                            ;("[illegible] [illegible] [illegible]")expect.

                                             ;()let mut contents = String::new
                            ;()info_file.read_to_string(&mut contents).unwrap

                                             ;()let mut lines = contents.lines
                                             ;()let name = lines.next().unwrap
                     ;()let years: i32 = lines.next().unwrap().parse().unwrap

(("!Ok(format!("Happy Birthday {name}, congratulations with the {years} years
                                                                                 {
                                                                                     {
```

- `ParcelFileDescriptor` [illegible] `OwnedFd` [illegible] `File` ([illegible] `OwnedFd` [illegible]) [illegible] `File` [illegible].
- [illegible]. [illegible] UDP [illegible] TCP [illegible] UNIX.

35 [illegible]

Android [illegible] [illegible]

[illegible] [illegible] Testing[illegible] [illegible] [illegible] [illegible] [illegible] unit test[illegible] [illegible] AOSP [illegible] [illegible]. [illegible] [illegible] rust_test [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]:

testing/Android.bp:

```
rust_library {
    name: "libleftpad",
    crate_name: "leftpad",
    srcs: ["src/lib.rs"],
}

rust_test {
    name: "libleftpad_test",
    crate_name: "leftpad_test",
    srcs: ["src/lib.rs"],
    host_supported: true,
    test_suites: ["general-tests"],
}
```

testing/src/lib.rs:

```
//! Left-padding library.

/// Left-pad `s` to `width`.
pub fn leftpad(s: &str, width: usize) -> String {
    format!("{s:>width$}")
}

mod tests {
    use super::*;

    fn short_string() {
        assert_eq!(leftpad("foo", 5), "  foo");
    }

    fn long_string() {
        assert_eq!(leftpad("foobar", 6), "foobar");
    {
{
```

□□ □□ □□□□ □□□□□□□ □□ □□□□□□

```
atest --host libleftpad_test
```

:□□□□ □□□□ □□□□ □□ □□□□□□

```
                    INFO: Elapsed time: 2.666s, Critical Path: 2.40s
                     .INFO: 3 processes: 2 internal, 1 linux-sandbox
                   INFO: Build completed successfully, 3 total actions
comprehensive-rust-android/testing:libleftpad_test_host           PASSED in 2.3s//
                  (PASSED  libleftpad_test.tests::long_string (0.0s
                 (PASSED  libleftpad_test.tests::short_string (0.0s
         Test cases: finished with 2 passing and 0 failing out of 2 test cases
```

□□ □□□□□□□□□ □□□□□ □□ □□□□□□ .□□□□□□□ □□□□ □□ □□□□□□□□□□ crate □□□□□ □□□□ □□□□□□ □□ □□□□□ □□□□□ .□□□□□□□ □□□□□ □□□□□□□ □□□□□□□□□□

GoogleTest 35.1

□□□□□□□□□□□□ □□□□□□□□□ □□□assert □□ □□□□□□ □□□□□□ *matchers* □□ □□□□□□□□□ □□ GoogleTest □□□□□ :□□□□□ □□□□□□ □□□

```
                              ;*::use googletest::prelude

                                 } ()fn test_elements_are
                    ;["let value = vec!["foo", "bar", "baz  
  ;((("expect_that!(value, elements_are!(eq(&"foo"), lt(&"xyz"), starts_with("a  
                                                         {
```

□□ □□□□ □□□ □□□□□□ □□□□□□□ □□□□□ □□□□□□ □□ □□ □□□□□□□ □□□□□□ □□□□□□"!"□□ □□ □□□□□ □□□□□□ □□□□ :□□□□□ □□ □□□□□ □□□□□□□ pin-pointing

```
                        ---- test_elements_are stdout ----
                                           Value of: value
                                      :Expected: has elements
                                       "is equal to "foo .0
                                      "is less than "xyz .1
                                    "!" starts with prefix .2
                                  ,["Actual: ["foo", "bar", "baz
              "!" where element #2 is "baz", which does not start with
                                at src/testing/googletest.rs:6:5
                                  Error: See failure output above
```

- □□ □□ □□ □□□□□ □□□□ □□□□□ □□□□□□□□□□ □□□□□□ Rust Playground □□ □□□□□ GoogleTest `cargo add` □□ □□□□□□□ Cargo □□□□□□ □□ □□ □□□□□ □□□□□□□ □□□□□ .□□□□□ □□□□□ local □□□□□ .□□□□□ □□□□□□□□□ `googletest`
- □□□□□ □□ □□□□□□□□□□ □□□type □ □□□□□□□□□ □□ □□□□□□□ `;*::use googletest::prelude` □□ .□□□□□□
- This just scratches the surface, there are many builtin matchers. Consider going through the first chapter of ”Advanced testing for Rust applications”, a self-guided Rust course: it

provides a guided introduction to the library, with exercises to help you get comfortable .with googletest macros, its matchers and its overall philosophy

- □□□□□ □□□□□□ □□ □□□□□ □□ □□□□ □□□□ □□□string □□ □□□□□□ □□□□ □□ □□□□ □□□□ □□□□ □□□□ □□□□□□ □□ :□□□□ □□ □□□□□

```
                       } ()fn test_multiline_string_diff
                \let haiku = "Memory safety found,\n
\Rust's strong typing guides the way,\n
            ;".Secure code you'll write
                                       )!assert_that
                                          ,haiku
                     \eq("Memory safety found,\n
       \Rust's silly humor guides the way,\n
                 (".Secure code you'll write
                                                  ;(
                                                       {
```

:(□□□□□□□□ □□□□□□□□□ □□□□□□ □□ □□ □□□□) □□□□□□ □□□□□ □□ □□□□□ □□□□□□

```
                                                                     Value of: haiku
 to "Memory safety found,\nRust's silly humor guides the way,\nSecure code you'll write
: "Memory safety found,\nRust's strong typing guides the way,\nSecure code you'll write
o "Memory safety found,\nRust's silly humor guides the way,\nSecure code you'll write
                                                        :(Difference(-actual / +expected
                                                                   ,Memory safety found
                                                   ,Rust's strong typing guides the way-
                                                     ,Rust's silly humor guides the way+
                                                              .Secure code you'll write
                                                     at src/testing/googletest.rs:17:5
```

- .□□□ Rust □□ (/GoogleTest for C++](https://google.github.io/googletest] □□□□□ □□ crate □□□

Mocking 35.2

□□ □□□□□□□□□ □□□□□ .□□□□ □□□□ □□□□□□□□□ □□□□□ □□□□□□ □□□□□□□□□□ □□ □□ Mockall □□ mocking □□□□□ :□□□□□ mock □□ □□□□□ □□□□□ □□ □□□□□□□□□□ □□□□ □□□□□□ □□□□□□ □□□□□□□ □□ □□□□ □□ □□□□□ □□□trait

```
                                          ;use std::time::Duration

                                                   } pub trait Pet
       ;fn is_hungry(&self, since_last_meal: Duration) -> bool
                                                                 {

                                             } ()fn test_robot_dog
                            ;()let mut mock_dog = MockPet::new
               ;(mock_dog.expect_is_hungry().return_const(true
;(assert_eq!(mock_dog.is_hungry(Duration::from_secs(10)), true
                                                                 {
```

- □□□□□□□□□□□□□ .□□□ (Android (AOSP □□ □□□□ □□□□□□ mocking □□□□□□□□□ Mockall mocking □□□□□□□□□ □□□□□□ □□ □□□□□□□ □□□□□□□ □□□□□□ □□ crates.io □□ □□□□□□ mocking

HTTP. [illegible] mocking [illegible] Mockall [illegible] mock [illegible].

- [illegible] mocking [illegible]_[illegible]: mock[illegible] [illegible]. [illegible] mock[illegible] [illegible] [illegible].

If at all possible, it is recommended that you use the real dependencies. As an example, many databases allow you to configure an in-memory backend. This means that you get the correct behavior in your tests, plus they are fast and will automatically clean up after themselves.

[illegible] framework[illegible] process [illegible] `localhost` [illegible]. [illegible] mock [illegible] framework [illegible].

- Mockall [illegible] Rust Playground [illegible] local [illegible]. [illegible] `cargo add mockall` [illegible] Mockall [illegible] Cargo [illegible].

- Mockall [illegible]. [illegible] mock [illegible] cat [illegible] 3 [illegible]:

```rust
fn test_robot_cat() {
    let mut mock_cat = MockPet::new();
    mock_cat
        .expect_is_hungry()
        .with(mockall::predicate::gt(Duration::from_secs(3 * 3600)))
        .return_const(true);
    mock_cat.expect_is_hungry().return_const(false);
    assert_eq!(mock_cat.is_hungry(Duration::from_secs(1 * 3600)), false);
    assert_eq!(mock_cat.is_hungry(Duration::from_secs(5 * 3600)), true);
}
```

- [illegible] `.times(n)` [illegible] mock method [illegible] n[illegible] --- [illegible] panic [illegible].

36 □□□□

□□□

□□□□□□□□□ (host □□□□) `stdout` □□ (□□□□□□□□ □□□□) `logcat`□□ □□□□□□□□ □□□□□ □□□□□ `log` crate □□ □□□□□ :□□□□□

:*hello_rust_logs/Android.bp*

```
} rust_binary
,"name: "hello_rust_logs
,"crate_name: "hello_rust_logs
,["srcs: ["src/main.rs
] :rustlibs
,"liblog_rust"
,"liblogger"
,[
,host_supported: true
{
```

:*hello_rust_logs/src/main.rs*

```
.Rust logging demo !//

;{use log::{debug, error, info

.Logs a greeting ///
} ()fn main
)logger::init
()logger::Config::default
("with_tag_on_device("rust.
,(with_min_level(log::Level::Trace.
;(
;(".□□□□□□□ □□□□□")!debug
;(".□□□□□□ □□□ □□□ □□□□□□")!info
;("!error!("Something went wrong
{
```

:□□□□□□ □□ □□□□ □□□□□□□□□□ □□□□□□ □ push □□□□□□□

```
m hello_rust_logs
adb push "$ANDROID_PRODUCT_OUT/system/bin/hello_rust_logs" /data/local/tmp
adb shell /data/local/tmp/hello_rust_logs
```

:□□□□□□ □□□□□□□□ adb logcat □□ □□□□□□

```
adb logcat -s rust
```

```
    .D rust: hello_rust_logs: Starting program 08:38:32.454  2420  2420 08-09
.I rust: hello_rust_logs: Things are going fine 08:38:32.454  2420  2420 08-09
 !E rust: hello_rust_logs: Something went wrong 08:38:32.454  2420  2420 08-09
```

37 [illegible]

[illegible]

Rust [illegible]. [illegible]:

- [illegible] Rust [illegible].
- [illegible] Rust.

[illegible] (*foreign function interface*) [illegible] FFI [illegible].

37.1 [illegible] C

Rust [illegible] link [illegible] object file [illegible] C [illegible]. [illegible] Rust [illegible] export [illegible] C [illegible].

[illegible]:

```
} "extern "C
;fn abs(x: i32) -> i32
{

} ()fn main
;let x = -42
.SAFETY: `abs` doesn't have any safety requirements //
;{ (let abs_x = unsafe { abs(x
;("{println!("{x}, {abs_x
{
```

[illegible] Safe FFI Wrapper [illegible].

[illegible] production [illegible].

[illegible].

37.1.1 [illegible] Bindgen

bindgen [illegible] C [illegible].

:□□□□□ □□□□□□ C □□□□□ □□□□□□□□□□ □□ □□□□□□

:*interoperability/bindgen/libbirthday.h*

```
             } typedef struct card
               ;const char* name
                      ;int years
                           ;card {

;(void print_card(const card* card
```

:*interoperability/bindgen/libbirthday.c*

```
                                              <include <stdio.h#
                                        "include "libbirthday.h#

                             } (void print_card(const card* card
                                  ;("printf("+--------------\n
                 ;(printf("| Happy Birthday %s!\n", card->name
;(printf("| Congratulations with the %i years!\n", card->years
                                  ;("printf("+--------------\n
                                                               {
```

:□□□□□ □□□□□□ □□□□ Android.bp □□□□□ □□ □□ □□□□

:*interoperability/bindgen/Android.bp*

```
                } cc_library
    ,"name: "libbirthday
,["srcs: ["libbirthday.c
                           {
```

:(□□□□□ □□□□□ □□□□□ □□□□ □□ □□□□□ □□□□ □□) □□□□□ □□□□□□ □□□□□□□□□□ □□□□□ wrapper □□□□ □□□□□ □□

:*interoperability/bindgen/libbirthday_wrapper.h*

```
"include "libbirthday.h#
```

:□□□□□ □□□□□□ □□□□□□□□ □□□□ □□ □□ (bindings) □□□□□□□□□ □□□□□□□ □□ □□□□□□

:*interoperability/bindgen/Android.bp*

```
                           } rust_bindgen
         ,"name: "libbirthday_bindgen
      ,"crate_name: "birthday_bindgen
,"wrapper_src: "libbirthday_wrapper.h
             ,"source_stem: "bindings
        ,["static_libs: ["libbirthday
                                        {
```

:□□□□□ □□□□□□□□□ □□□□ Rust □□□□□□□ □□ □□binding □□ □□□□□□□□□□ □□□□□□□ □□

:*interoperability/bindgen/Android.bp*

```
                         } rust_binary
         ,"□□□□□_□□□□□_□□□□" :name
                ,["srcs: ["main.rs
,["rustlibs: ["libbirthday_bindgen
                                     {
```

:interoperability/bindgen/main.rs

```
.Bindgen demo !//

;{use birthday_bindgen::{card, print_card

} ()fn main
;()unwrap.("[illegible]")let name = std::ffi::CString::new
;{ let card = card { name: name.as_ptr(), years: 42
SAFETY: The pointer we pass is valid because it came from a Rust //
reference, and the `name` it contains refers to `name` above which also //
remains valid. `print_card` doesn't store either pointer to use later //
.after it returns //
} unsafe
;(print_card(&card as *const card
{
{
```

:[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] push [illegible]

```
m print_birthday_card
adb push "$ANDROID_PRODUCT_OUT/system/bin/print_birthday_card" /data/local/tmp
adb shell /data/local/tmp/print_birthday_card
```

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] (bindings) [illegible] [illegible]:

:interoperability/bindgen/Android.bp

```
} rust_test
,"name: "libbirthday_bindgen_test
,["srcs: [":libbirthday_bindgen
,"crate_name: "libbirthday_bindgen_test
,["test_suites: ["general-tests
,auto_gen_config: true
clippy_lints: "none", // Generated file, skip linting
,"lints: "none
{
```

```
atest libbirthday_bindgen_test
```

37.1.2 [illegible] Rust

:[illegible] [illegible] C [illegible] Rust [illegible] [illegible] [illegible] [illegible] Exporting [illegible] [illegible]

interoperability/rust/libanalyze/analyze.rs

```
.Rust FFI demo !//

;use std::os::raw::c_int

.Analyze the numbers ///
} (pub extern "C" fn analyze_numbers(x: c_int, y: c_int
} if x < y
;("!println!("x ({x}) is smallest
                                                        } else {
        ;("[illegible] ({x ({x [illegible] [illegible] [illegible] ({y  ({y[illegible] ")!println
                                                             {
                                                                {
```

interoperability/rust/libanalyze/analyze.h

```
ifndef ANALYSE_H#
define ANALYSE_H#

     } "extern "C
;(void analyze_numbers(int x, int y
                                  {

                        endif#
```

interoperability/rust/libanalyze/Android.bp

```
                       } rust_ffi
       ,"name: "libanalyze_ffi
   ,"crate_name: "analyze_ffi
         ,["srcs: ["analyze.rs
          ,["."] :include_dirs
                               {
```

:[illegible] [illegible] C [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

interoperability/rust/analyze/Android.bp

```
"include "analyze.h#

            } ()int main
  ;(analyze_numbers(10, 20
;(analyze_numbers(123, 123
                 ;return 0
                          {
```

interoperability/rust/analyze/Android.bp

```
                 } cc_binary
   ,"name: "analyze_numbers
           ,["srcs: ["main.c
,["static_libs: ["libanalyze_ffi
                               {
```

:[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] push [illegible]

```
                                         m analyze_numbers
adb push "$ANDROID_PRODUCT_OUT/system/bin/analyze_numbers" /data/local/tmp
                        adb shell /data/local/tmp/analyze_numbers
```

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] Rust [illegible] [illegible] `[no_mangle]#` [illegible] [illegible] [illegible] `["export_name = "some_name]#` [illegible] [illegible] [illegible] .[illegible] [illegible] [illegible] .[illegible] [illegible] [illegible]

37.2 □□ C++

□□□□ CXX crate □□□□□□□ □□□□□□□□ □□□□ □□□ Rust □ C++ □□□ □□□□□□□ □□□□□□.

□□□□□□□□ □□□□ □□ □□□□ □□□□□ □□□□:

37.2.1 □□□□□□ □□

CXX □□□□□ □□ □□□□□□□□ □□ signature□□□ □□□□□ □□□ □□ □□ □□ □□□□□ □□ □□□□□ □□□□□ □□ □□□□□ □□□ □□□□□ □□□□□□□. □□□□ □□□□ □□□□□□□□□ □□ □□ □□□□□□□□□ □□ □□□□□□□□□ □□□□□□ □□ □□ □□□□□□ Rust □□□□□□ □□□□□□□□ □□ □□ attribute □□□□□□□□ #[cxx::bridge] □□□□ □□□□□ □□□ □□□.

```rust
mod ffi {
    // Shared structs with fields visible to both languages.
    struct BlobMetadata {
        size: usize,
        tags: Vec<String>,
    }

    // Rust types and signatures exposed to C++.
    extern "Rust" {
        type MultiBuf;

        fn next_chunk(buf: &mut MultiBuf) -> &[u8];
    }

    // C++ types and signatures exposed to Rust.
    unsafe extern "C++" {
        include!("include/blobstore.h");

        type BlobstoreClient;

        fn new_blobstore_client() -> UniquePtr<BlobstoreClient>;
        fn put(self: Pin<&mut BlobstoreClient>, parts: &mut MultiBuf) -> u64;
        fn tag(self: Pin<&mut BlobstoreClient>, blobid: u64, tag: &str);
        fn metadata(&self, blobid: u64) -> BlobMetadata;
    }
}
```

- □□ □□ □□□□ □□□□ □□ □□ □□□□□□ ffi□□ crate □□□□ □□□□□□ □□□□□□.
- □□ □□□□□□□□□ (declarations) □□□□□□ □□□□ □□ □□□□□□ □□□□ CXX □□□□□□□ □□□□□□ □□ type/function □□ Rust □ C++ □□ □□□□□□ □□□□□□ □□ □□ □□□□□□ □□ □□ □□□□□ □□ □□ □□□□□ □□□□□ □□□.
- □□□□□ □□□□□□□ □□ Rust □□□□□□ □□□□ □□ cargo-expand □□ □□□□□ □□□□□□□ □□□□□□ proc □□□□□□ □□□□□□ □□□□□□□□ □□□□□. □□□□□ □□□□□□ □□□□□□□□ □□ cargo expand ::ffi □□□□ □□□□□ □□□□□□ □□□□□□ ffi □□□□□□□□□ □□□□□ (□□□□□□ □□□□ □□□□□ □□□□□□□□□ Android □□□□□□□ □□□□□□).
- □□□□□ □□□□□□□ □□ C++ □□□□□□ □□□□ □□ target/cxxbridge □□□□□ □□□□□.

37.2.2 □□□□□□ □□□ □□ Rust

```rust
mod ffi {
    extern "Rust" {
```

```
type MyType; // Opaque type
`fn foo(&self); // Method on `MyType
fn bar() -> Box<MyType>; // Free function
{
{

;(struct MyType(i32

} impl MyType
} (fn foo(&self
;(println!("{}", self.0
{
{

} <fn bar() -> Box<MyType
((Box::new(MyType(123
{
```

- [illegible] extern "Rust" reference [illegible] .[illegible]
- [illegible] C++ [illegible] "extern "Rust [illegible] CXX [illegible] C++ [illegible] header. [illegible] Rust [illegible] .rs.h [illegible]

37.2.3 Generated C++

```
} mod ffi
.++Rust types and signatures exposed to C //
} "extern "Rust
;type MultiBuf

;[fn next_chunk(buf: &mut MultiBuf) -> &[u8
{
{
```

:[illegible] C++ ([illegible]) [illegible]

```
} struct MultiBuf final : public ::rust::Opaque
;MultiBuf() = delete~

:private
;friend ::rust::layout
} struct layout
;static ::std::size_t size() noexcept
;static ::std::size_t align() noexcept
;{
;{

rust::Slice<::std::uint8_t const> next_chunk(::org::blobstore::MultiBuf &buf) noexcept::
```

C++ Bridge Declarations 37.2.4

```
                                         } mod ffi
      .C++ types and signatures exposed to Rust //
                          } "++unsafe extern "C
            ;("include!("include/blobstore.h

                         ;type BlobstoreClient

   ;<fn new_blobstore_client() -> UniquePtr<BlobstoreClient
;fn put(self: Pin<&mut BlobstoreClient>, parts: &mut MultiBuf) -> u64
  ;(fn tag(self: Pin<&mut BlobstoreClient>, blobid: u64, tag: &str
             ;fn metadata(&self, blobid: u64) -> BlobMetadata
                                                          {
                                                             {
```

:[illegible] Rust ([illegible]) [illegible]

```
                              } pub struct BlobstoreClient
                    ,private: ::cxx::private::Opaque_
                                                     {

    } <pub fn new_blobstore_client() -> ::cxx::UniquePtr<BlobstoreClient
                                               } "extern "C
           ;fn __new_blobstore_client() -> *mut BlobstoreClient
                                                          {
    { (()unsafe { ::cxx::UniquePtr::from_raw(__new_blobstore_client
                                                             {

                                         } impl BlobstoreClient
               } pub fn put(&self, parts: &mut MultiBuf) -> u64
                                          } "extern "C
                                         )fn __put
                              ,BlobstoreClient& :_
              ,parts: *mut ::cxx::core::ffi::c_void
                                         ;u64 <- (
                                                 {
                                             } unsafe
(put(self, parts as *mut MultiBuf as *mut ::cxx::core::ffi::c_void__
                                                 {
                                                    {
                                                       {

                                                    ... //
```

- [illegible] signature [illegible] C++ [illegible] signature [illegible] CXX .[illegible] .[illegible]
- [illegible] C++ [illegible] unsafe extern [illegible] .[illegible] Rust [illegible]

37.2.5 [illegible]

```
mod ffi {
    struct PlayingCard {
        suit: Suit,
        value: u8,  // A=1, J=11, Q=12, K=13
    }

    enum Suit {
        Clubs,
        Diamonds,
        Hearts,
        Spades,
    }
}
```

- [illegible] C-like (unit) enums [illegible].
- [illegible] `#[derive()]` [illegible]. [illegible] C++ [illegible]. Hash [illegible] `std::hash` [illegible] C++ [illegible].

37.2.6 Shared Enums

```
mod ffi {
    enum Suit {
        Clubs,
        Diamonds,
        Hearts,
        Spades,
    }
}
```

Generated Rust:

```
pub struct Suit {
    pub repr: u8,
}

impl Suit {
    pub const Clubs: Self = Suit { repr: 0 };
    pub const Diamonds: Self = Suit { repr: 1 };
    pub const Hearts: Self = Suit { repr: 2 };
    pub const Spades: Self = Suit { repr: 3 };
}
```

Generated C++:

```
enum class Suit : uint8_t {
    Clubs = 0,
    Diamonds = 1,
    Hearts = 2,
    Spades = 3,
};
```

- [illegible] Rust[illegible] enums [illegible]. [illegible] UB [illegible] C++ [illegible] enum [illegible] Rust [illegible].

37.2.7 [illegible] Rust

```
mod ffi {
    extern "Rust" {
        fn fallible(depth: usize) -> Result<String>;
    }
}

fn fallible(depth: usize) -> anyhow::Result<String> {
    if depth == 0 {
        return Err(anyhow::Error::msg("fallible1 [illegible] > 0 [illegible]"));
    }

    Ok("Success!".into())
}
```

- Rust [illegible] «[illegible]» [illegible] exception[illegible] C++ [illegible].
- exception [illegible] rust::Error [illegible] string [illegible]. [illegible] Display[illegible].
- panic [illegible] Rust [illegible] C++ [illegible].

37.2.8 [illegible] C++

```
mod ffi {
    unsafe extern "C++" {
        include!("example/include/example.h");
        fn fallible(depth: usize) -> Result<String>;
    }
}

fn main() {
    if let Err(err) = ffi::fallible(99) {
        eprintln!("Error: {}", err);
        process::exit(1);
    }
}
```

- C++ [illegible] (declared) [illegible] Result[illegible] exception [illegible] C++ [illegible] Err [illegible] Rust [illegible]-[illegible].
- exception [illegible] extern "C++" function [illegible] CXX [illegible] "[illegible]" [illegible] std::terminate [illegible] C++' [illegible]. [illegible] exception [illegible] noexcept C++ function [illegible].

37.2.9 [illegible]

Rust Type	C++ Type
`String`	`rust::String`
`str&`	`rust::Str`
`CxxString`	`std::string`
`[T]/&mut [T]&`	`rust::Slice`
`Box<T>`	`<rust::Box<T`
`<UniquePtr<T`	`<std::unique_ptr<T`
`<Vec<T`	`<rust::Vec<T`
`<CxxVector<T`	`<std::vector<T`

- □□extern function □ □□□□□□□□□□ □ □□□□□ □□□□□□□□□□ □□□□□□□□ □□ □□□□□□□ □□ □□type □□□
.□□□ □□□□□□□□
- □□□□ .□□□□□□□ □□□□□□ `std::string` □□ □□□□□□□□□ Rust □□ `String` □□ □□□□□□ □□□□□□ □□□□□
:□□□□□ □□□□□ □□□□ □□□□□ □□□□□
  - .□□□□□□□ □□□□□□□□□ □□ □□□□□ □□□□□ □□ □□ `String` □□ □□ UTF-8 □□□□□ `std::string`
  - □□ □□□□□ □□□□□□□□ □□□□□□□□□ □ □□□□□□ □□□□□□ □□ □□□□□□□□ □□□□□□□□□□□ □□□□ □□ □□□□
.□□□ □□□□□□ □□□□□□□ □□□□ □□□□□□□□□
  - `std::string` requires move constructors that don't match Rust's move semantics,
.so a `std::string` can't be passed by value to Rust

□□□□□□□□□ □□□ □□□□□ 37.2.10

□□□□ □□□□□□ □□□□□ □□□□□ □ □□□□ □□□□□ □□□ C++ □□□□□□□□□□ □□□□□ □□□□□ `cc_library_static` □□□
.□□□□□ □□□□□□ CXX

```
                  } cc_library_static
           ,"name: "libcxx_test_cpp
           ,["srcs: ["cxx_test.cpp
                ] :generated_headers
            ,"cxx-bridge-header"
      "libcxx_test_bridge_header"
                                ,[
,["generated_sources: ["libcxx_test_bridge_code
                                    {
```

- `libcxx_test_bridge_code` □ `libcxx_test_bridge_header` □□ □□□□□ □□□□□□ □□□□□ □□□□ □□
□□ □□□□□□ □□□□□□ □□□□□ .□□□□□□ CXX □□□□□ □□□□ □□□□□□ C++ □□□□□□□□□□ □□□□□ □□□□□□□□□□□□□
.□□□ □□□□□□□ □□□□□ □□□□□ □□□□□□□ □□□
- `cxx-bridge-` □□□□□□□□□ □□□ □□□□□ CXX □□□□□ □□□□□□□ □□□□□□ □□□□□ □□□ □□□□□□ □□□□□□ □□□□□
.□□□□□□ □□□□□□□ `header`
- □□□□□ □□□□□ □□□□□ □□□□ □□□ □□□□□□□□□□ □□□ Android □□□ CXX □□□ □□□□□□□□□ □□□□□ □□□□□ □□□□□□□□□
□□□ □□□□□□□□□ □□□□□□□ □□□ □□□□□ □□□ □□□ □□□□□□ □□□ □□□□□□□□□ □□□□ □□□□□ .the Android docs
.□□□□□ □□□□□ □□□□□□□ □□□ □□□□□□□□□□□□□□ □□□□ □□□□□□□□□□ □□□□□□ □□□ □□□ □□□□□□□ □□□□□□□□□□□

□□□□□□□□□ □□□ □□□□□ 37.2.11

□□□□ .CXX □□□□□ □□□□□ □□□□□□□ □□□□□ □□□□□□ □ CXX □□□□ □□□□□□□ □□□□□ □□□□ :□□□□□ □□□□□□ □□□□□ □□□□ □□□
.□□□□□□ □□□□□□□□□ `cc_library_static` □□□□□□ □□□□□□ □□□ □□□□□□ □□□

```
Generate a C++ header containing the C++ bindings //
          .to the Rust exported functions in lib.rs //
```

```
genrule {
    name: "libcxx_test_bridge_header",
    tools: ["cxxbridge"],
    cmd: "$(location cxxbridge) $(in) --header > $(out)",
    srcs: ["lib.rs"],
    out: ["lib.rs.h"],
}

// Generate the C++ code that Rust calls into.
genrule {
    name: "libcxx_test_bridge_code",
    tools: ["cxxbridge"],
    cmd: "$(location cxxbridge) $(in) > $(out)",
    srcs: ["lib.rs"],
    out: ["lib.rs.cc"],
}
```

- □□ .□□□□□□ □□□□□□ □□ □□ □□□□□□ C++ □□□ □□ □□□□ □□□□□□ □□□□□□ □□ cxxbridge□□□□□□ .□□□□ □□□□□□ □□ Soong □□□□□□ □□□□□□ □□ □ □□□□ □□□□□□□□ Android
- lib.rs.h □□□□ header □□□□□ □□□□□□ lib.rs □□□□ Rust □□□□□ □□□□□ □□□□ □□□□□□□□□□ □□□□ □□□□□ □□□□□□□□□□ □□□□□□□□□ □□□□ □□□□□□ .□□□□□ □□□□□□ □□□□ lib.rs.cc □□□□ □□□□□ □□□□□ □ .□□□□□□□

□□□□□□□□ □□ □□□□□ 37.2.12

.□□□□□ □□□□□□ □□□□ cc_library_static □ libcxx □□ □□ □□□□□ □□□□□□ rust_binary□□

```
rust_binary {
    name: "cxx_test",
    srcs: ["lib.rs"],
    rustlibs: ["libcxx"],
    static_libs: ["libcxx_test_cpp"],
}
```

□□□□□ □□ □□□□□□□ □□□□□□□ 37.3

(Java Native Interface (JNI □□□□□ □□□□□ □□□□□] □□□□□ □□ □□ □□□□□□ □□□□object □□□□□□□□ □□□□□ .□□□□□ □□□□□□ □□□□□□□ □□□□□□□□□□ □□ □□□□ □□ □□□□□□ □□□□ □□ jni crate .□□□□ □□□□□□□□□

:□□□□□□□ □□□□□□ Java □□ export □□□□□ Rust □□□□□ □□ □□□□□□

:*interoperability/java/src/lib.rs*

```
//! Rust <-> Java FFI demo.

use jni::objects::{JClass, JString};
use jni::sys::jstring;
use jni::JNIEnv;

/// HelloWorld::hello method implementation.
pub extern "system" fn Java_HelloWorld_hello(
    env: JNIEnv,
                                         ,class: JClass_
                                          ,name: JString
                                               } jstring <- (
;()let input: String = env.get_string(name).unwrap().into
                ;("!{input} ,□□□□")!let greeting = format
          ;()let output = env.new_string(greeting).unwrap
                                      ()output.into_inner
                                                            {
```

:interoperability/java/Android.bp

```
                                            } rust_ffi_shared
                                    ,"name: "libhello_jni
                                 ,"crate_name: "hello_jni
                                    ,["srcs: ["src/lib.rs
                                    ,["rustlibs: ["libjni
                                                            {
```

:□□□□ □□ □□□□□□□□ □□□□ □□ □□ □□□□ □□□ □□□

:interoperability/java/HelloWorld.java

```
                                           } class HelloWorld
         ;(private static native String hello(String name

                                                 } static
                     ;("System.loadLibrary("hello_jni
                                                        {

                 } (public static void main(String[] args
            ;("□□□□")String output = HelloWorld.hello
                          ;(System.out.println(output
                                                        {
                                                            {
```

:interoperability/java/Android.bp

```
                                                } java_binary
                                  ,"name: "helloworld_jni
                      ,["HelloWorld.java"□□□□ □□□□] :srcs
                                ,"main_class: "HelloWorld
                              ,["required: ["libhello_jni
                                                            {
```

:□□□□ □□□□ □ □□□□ □□□□□□□□□ □□□□□□□ □□ □□□□□□ □□□□□□□□ □□□□□□ □□

```
                                             m helloworld_jni
                 adb sync  # requires adb root && adb remount
                         adb shell /system/bin/helloworld_jni
```

38 [illegible]

[illegible]

[illegible]: [illegible]: [illegible] Rust [illegible]. [illegible]:

- [illegible] AIDL [illegible] Rust [illegible].
- [illegible] Rust [illegible].

[illegible]: [illegible] Rust on fly [illegible].

X □□□

Chromium

39 [illegible]

[illegible] Chromium [illegible] Rust [illegible]

[illegible] glue code [illegible] Chromium [illegible] Rust [illegible] .Chromium C++ [illegible] Rust [illegible]

> [illegible] .[illegible] string [illegible] Rust [illegible] UTF8 [illegible] .[illegible]

[illegible] 40

[illegible]

[illegible] Chromium [illegible] build [illegible]. [illegible] build flag [illegible] ([illegible] commit 1223636 [illegible] 2023):

```
gn gen out/Debug
autoninja -C out/Debug chrome
out/Debug/chrome # or on Mac, out/Debug/Chromium.app/Contents/MacOS/Chromium
```

([illegible] component [illegible] debug build [illegible]. [illegible]!)

[illegible] [illegible] Chromium [illegible] [illegible]. [illegible]: [illegible]-[illegible] build Chromium [illegible].

[illegible] Visual Studio code [illegible].

[illegible]

[illegible]

[illegible] 41

[illegible] Chromium [illegible] [illegible] Cargo

[illegible] Rust [illegible] [illegible] `cargo` [illegible] [illegible] crates.io [illegible] [illegible]. Chromium [illegible] [illegible] [illegible] gn [illegible] `ninja` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

[illegible] [illegible] [illegible] [illegible] Rust [illegible] [illegible] [illegible] [illegible]:

- [illegible] gn [illegible] `ninja` [illegible] [illegible] [illegible] `//build/rust/*.gni` [illegible] [illegible] ([illegible] `rust_static_library` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]). [illegible] [illegible] toolchain [illegible] crate[illegible] [illegible] Chromium [illegible] [illegible].
- [illegible] `cargo` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] toolchain [illegible] crate[illegible] [illegible] Chromium [illegible] [illegible]
- [illegible] `cargo` [illegible] [illegible] [illegible] [illegible] toolchain [illegible]/[illegible] [crate[illegible] [illegible] [illegible] [illegible] [illegible]]-(https://crates.io/) [illegible] [illegible].

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] gn [illegible] `ninja` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] Rust [illegible] [illegible] [illegible] Chromium [illegible] [illegible]. [illegible] [illegible] [illegible] Cargo [illegible] [illegible] [illegible] [illegible] Rust [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

[illegible] [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]:

- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `cargo` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
- [illegible] [illegible] [illegible] gn [illegible] `ninja` [illegible] `cargo` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]-[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible]-[illegible] [illegible] [illegible] [illegible].

[illegible]/[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] (“[illegible] [illegible] Cargo [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]”):

- [illegible] Chromium [illegible] crates.io [illegible] crate [illegible] clap) [illegible] itertools [illegible] serde [illegible] (iterators) [illegible].
  - [illegible] Cargo.toml [illegible] cargo [illegible]
  - [illegible] perl [illegible] CPAN [illegible] python + pip [illegible].
- Development experience is made really nice not only by core Rust tools (e.g. using `rustup` to switch to a different `rustc` version when testing a crate that needs to work on nightly, current stable, and older stable) but also by an ecosystem of third-party tools (e.g. Mozilla provides `cargo vet` for streamlining and sharing security audits; `criterion` crate gives a streamlined way to run benchmarks).
  - `cargo install --locked cargo-vet` [illegible] `cargo` [illegible].
  - [illegible] VScode [illegible].
- [illegible] `cargo` [illegible]:
  - [illegible] Rust [illegible].
  - [illegible] Rust [illegible] Rust [illegible] Cargo [illegible] Chromium [illegible] Bazel [illegible] (Android/Soong [illegible] Cargo [illegible].
- [illegible] Chromium [illegible] `cargo` [illegible]:
  - `serde_json_lenient` [illegible] Google [illegible] PR [illegible]
  - `font-types` [illegible]
  - `gnrt` [illegible] `clap` [illegible] `toml` [illegible].
    - Disclaimer: a unique reason for using `cargo` was unavailability of `gn` when building and bootstrapping Rust standard library when building Rust toolchain.
    - `run_gnrt.py` [illegible] Chromium [illegible] `cargo` [illegible] `rustc` [illegible] gnrt. [illegible] `run_gnrt.py` [illegible] `cargo` [illegible] `--locked` [illegible] `Cargo.lock` [illegible]

[illegible]:

- `rustc` (Rust [illegible]) [illegible] LLVM [illegible] Clang [illegible] `rustc` [illegible] GitHub [illegible] Rust [illegible] Rust [illegible] (bootstrapping) [illegible]
- `rustup` [illegible] `rustup` [illegible] https://github.com/rust-lang/ [illegible] `rustc` [illegible]
- `cargo`, `rustfmt` [illegible].

- [illegible] ([illegible] rustc [illegible] toolchain [illegible] Chromium [illegible])
- [illegible] Cargo [illegible] `cargo vet` [illegible] `cargo audit` [illegible].
- [illegible] Rust [illegible] `//third_party/rust` [illegible] ([illegible] security@chromium.org)
- [illegible] Rust ([illegible])

[illegible] 42

[illegible] Chromium Rust

Chromium [illegible] Rust [illegible] Area Tech Leads [illegible].

[illegible] Chromium [illegible]. Rust [illegible].

[illegible] Rust [illegible] C/C++ API [illegible] (expose) [illegible] glue code [illegible].

```
                                           C++"                           Rust"
.- - - - - - - - - - - - - - - - - - - - - - -.           .- - - - - - - - - -.
:                                             :           :                   :
:   Existing Chromium :           :  Chromium Rust              Existing Rust :
:           C++"             :           :  "wrapper"                  crate" :
: +-------------+          +----------------+ :           : +---------------+ :
: |             |          |                | :           : |               | :
: |          <--+----------+-o-----+-+-----------+-+->            o         | :
: |             |    Language  : |                | Crate : |               | :
: +-------------+      boundary  : +----------------+ API : +---------------+ :
:                                             :           :                   :
'- - - - - - - - - - - - - - - - - - - - - - -`           '- - - - - - - - - -`
```

[illegible] Rust glue [illegible] crate [illegible] `third_party/rust/<crate>/<version>/wrapper` [illegible].

[illegible]:

- [illegible] Rust [illegible] (”crates”)
- [illegible] glue code [illegible] crate[illegible] Chromium C++ [illegible].

[illegible].

[illegible] 43

[illegible] Build

[illegible] Rust [illegible] `cargo` [illegible]. Chromium [illegible] gn [illegible] `ninja` [illegible] --- [illegible]. Rust [illegible].

[illegible] Rust [illegible] Chromium

[illegible] `BUILD.gn` [illegible] `rust_static_library` [illegible]:

```
import("//build/rust/rust_static_library.gni")

rust_static_library("my_rust_lib") {
  crate_root = "lib.rs"
  sources = [ "lib.rs" ]
}
```

[illegible] deps [illegible] Rust [illegible]. [illegible].

[illegible] crate root [illegible]. `crate_root` [illegible] Rust [illegible] --- [illegible] `lib.rs` [illegible]. [illegible] `sources` [illegible] `ninja` [illegible].

([illegible] Rust source_set [illegible] Rust [illegible] crate [illegible]. `static_library` [illegible].)

[illegible] gn [illegible] CXX [illegible]. [illegible] gn [illegible] Rust [illegible] interop[illegible] Rust [illegible].

43.1 [illegible] unsafe Rust

[illegible] Rust [illegible] `rust_static_library` [illegible] --- [illegible]. [illegible] unsafe Rust [illegible] `allow_unsafe = true` [illegible] gn [illegible]. ([illegible]

(.□□□ □□□□ □□ □□ □□ □□□ □□□□□□ □□ □□□□□□ □□ □□□□ □□

```
("import("//build/rust/rust_static_library.gni

          } ("rust_static_library("my_rust_lib
                       "crate_root = "lib.rs
                                 ] = sources
                                 ,"lib.rs"
                         "hippopotamus.rs"
                                           [
                         allow_unsafe = true
                                             {
```

Chromium C++ □□ Rust Code □□ □□□□ 43.2

.□□□□□ □□□□□□□ ++Chromium C □□□□□□□ □□ □□□□□ deps □□ □□ □□□□□ □□□□ □□□□□□ □□

```
("import("//build/rust/rust_static_library.gni

          } ("rust_static_library("my_rust_lib
                       "crate_root = "lib.rs
                      [ "sources = [ "lib.rs
                                             {

          .or source_set, static_library etc #
                } ("component("preexisting_cpp
                   [ "deps = [ ":my_rust_lib
                                             {
```

Visual Studio Code 43.3

□□ .□□□□□ C++□□ □□□□□□□□ □□□□ □□□□ IDE □□ □□□□□□□ □□□□□ □□ □□□□ □□□□ □□□□ Rust □□ □□ □□□□□□□ □□□□ □□ □□□□□□□□□ □□□□□ □ □□□□ □□ □□□□ □□□□□ □□ Chromium □□ Rust □□□□□ □□□□□□□□ □□□□□□□

- □□□□□□□□ □□ □□□□□ extension `rust-analyzer` □□□□□□ □□□□ VSCode □□ □□□□□ □□□□□ □□□□□□□□□ Rust □□ □□□□□□□□□□ □□□□□
- □□□□□□ □□□□□□□□□□□ □□□□□ □□ □□□□□□ □□) `gn gen out/Debug --export-rust-project` (□□□
- `ln -s out/Debug/rust-project.json rust-project.json`

code □□□□□□□□□□□ □□ □□□□□ □□□□□□ □□□□□□ □□□□□□□ □□□□□□ □□IDE □□ □□□□□ □□□□□□ □□□□ □□ □□□□□□ □□□□ .□□□□□ □□□□□ □□□□□□□□□ rust-analyzer □□ □□□□□ □ annotation

□□ □□□□□□ Rust □□□□□ □□ □□ □□□□ □□ □□□□) □□□□ □□□□ □□□□□□□□ □□□□□ □□ □□□□ □□□□□ □□□□ □□□□□□ :(□□□□□ □□□□□□□□□ □□□□□□□ □□□□□ □□ □□ □□□□□□ □□ Chromium

- □□□□□ □□□□ □□ `components/qr_code_generator/qr_code_generator_ffi_glue.rs`
- □□□□□ qr_code_generator_ffi_glue.rs‘ □□ (26 □□ □□□□□) `QrCode::new` □□□□□□□□□□□ □□□□ □□ □□□□ □□□□□ □□□□□
- .(typical bindings: vscode = ctrl k i; vim/CoC = K) ** □□□□□□□□□ □□□□□□** □□□□□□□ □□□□□□
- .(**go to definition** (typical bindings: vscode = F12; vim/CoC = g d □□□□□□□□ □□ Demo □□□□□□ (.□□□□□□□□ `third_party/rust/.../qr_code-.../src/lib.rs//` □□ □□ □□□□ □□□□)

- [illegible] **outline** [illegible] `QrCode::with_bits` [illegible] 164[illegible] [illegible] file explorer [illegible] vscode [illegible] (typical vim/CoC bindings = space o
- [illegible] **type annotations** [illegible] `QrCode::with_bits` [illegible]

[illegible] `gn gen ... --export-rust-project` [illegible] `BUILD.gn` [illegible].

43.4 [illegible]

[illegible] Chromium [illegible] Rust target [illegible] `//ui/base/BUILD.gn` [illegible]:

```
pub extern "C" fn hello_from_rust() {
  println!("[illegible] Rust!")
}
```

[illegible]: [illegible] `no_mangle` [illegible] Rust [illegible] (type of unsafety) [illegible] unsafe [illegible] gn target [illegible].

[illegible] Rust [illegible] `//ui/base:base` [illegible]. [illegible] `ui/base/resource/resource_bundle.cc` [illegible] bindings [illegible]:

```
extern "C" void hello_from_rust();
```

[illegible] `ui/base/resource/resource_bundle.cc` [illegible] - [illegible] `ResourceBundle::MaybeMangleLocalizedString` [illegible]. Chromium [illegible] Build [illegible] "Hello from Rust!" [illegible].

[illegible] VSCode [illegible] Rust [illegible] VSCode [illegible]. [illegible] "Go to definition" [illegible] `println!` [illegible].

[illegible] help [illegible]

- [illegible] `rust_static_library` gn template
- [illegible] `#[no_mangle]`
- [illegible] `extern "C"`
- [illegible] gn [illegible] `--export-rust-project` switch
- How to install rust-analyzer in VSCode

[illegible] C [illegible]. [illegible] C++ [illegible] Rust [illegible].

`allow_unsafe = true` [illegible] `#[no_mangle]` [illegible] Rust [illegible] Rust [illegible].

[illegible] Rust [illegible] gn [illegible] `rust_executable` [illegible].

44 [illegible]

[illegible]

[illegible] Rust [illegible] unit test [illegible]. [illegible]:

```
mod tests {
    fn my_test() {
        todo!()
    }
}
```

[illegible] Chromium [illegible] unit test [illegible] Rust [illegible] --- [illegible] `.rs` [illegible] (`test` [illegible]) [illegible].

[illegible] Rust [illegible] Chromium [illegible]:

- [illegible] Native Rust [illegible] (`#[test]`). [illegible] `//third_party/rust` [illegible].
- [illegible] `gtest` [illegible] C++ [illegible] Rust [illegible] FFI [illegible]. [illegible] Rust [illegible] FFI [illegible] unit test [illegible].
- [illegible] `gtest` [illegible] Rust [illegible] crate [illegible] API [illegible] (`pub mod for_testing { ... }` [illegible]) « [illegible]. [illegible].

[illegible] native Rust [illegible] crate [illegible] Chromium [illegible]. ([illegible] --- [illegible] crate [illegible].)

[illegible] C++ `gtest` [illegible] Rust `gtest` [illegible]:

- QR [illegible] Rust [illegible] (FFI [illegible]) [illegible] unit test [illegible] C++ [illegible] C++ [illegible] Rust [illegible] (Rust [illegible] `ScopedFeatureList` [illegible]).
- Hypothetical/WIP PNG [illegible] `libpng` [illegible] png crate [illegible] - [illegible]. RGBA => BGRA [illegible] (gamma correction). [illegible] Rust [illegible].

44.1 rust_gtest_interop Library

[illegible] `rust_gtest_interop` [illegible]:

- [illegible] Rust [illegible] `gtest` [illegible] (attribute [illegible] `#[gtest(...)]`)
- [illegible] `expect_eq!` [illegible] (`assert_eq!` [illegible] assertion [illegible] panic [illegible]).

[illegible]:

```
use rust_gtest_interop::prelude::*;

fn test_addition() {
    expect_eq!(2 + 2, 4);
}
```

44.2 [illegible] GN [illegible] Rust

[illegible] Rust gtest build [illegible] C++ [illegible]. [illegible]:

```
test("ui_base_unittests") {
  ...
  sources += [ "my_rust_lib_unittest.rs" ]
  deps += [ ":my_rust_lib" ]
}
```

[illegible] Rust [illegible] `static_library` [illegible]:

```
rust_static_library("my_rust_lib_unittests") {
  testonly = true
  is_gtest_unittests = true
  crate_root = "my_rust_lib_unittest.rs"
  sources = [ "my_rust_lib_unittest.rs" ]
  deps = [
    ":my_rust_lib",
    "//testing/rust_gtest_interop",
  ]
}

test("ui_base_unittests") {
  ...
  deps += [ ":my_rust_lib_unittests" ]
}
```

44.3 chromium::import! Macro

[illegible] `:my_rust_lib` [illegible] deps GN [illegible] `my_rust_lib` [illegible] `my_rust_lib_unittest.rs` [illegible]. [illegible] `crate_name` [illegible] `my_rust_lib` [illegible] crate [illegible]

[illegible] `chromium::import!` [illegible] `chromium` [illegible] crate [illegible]:

```rust
chromium::import! {
    "//ui/base:my_rust_lib";
}

use my_rust_lib::my_function_under_test;
```

[illegible]:

```rust
extern crate ui_sbase_cmy_urust_ulib as my_rust_lib;

use my_rust_lib::my_function_under_test;
```

[illegible] doc comment [illegible] `chromium::import`.

[illegible] `rust_static_library` [illegible] `crate_name` [illegible] crate [illegible] crates.io [illegible] crate [illegible] GN [illegible] `cargo_crate` [illegible] `gnrt` [illegible] crate [illegible].

44.4 [illegible]

[illegible]!

[illegible] Chromium build [illegible]:

- [illegible] `hello_from_rust` [illegible] n [illegible].
- [illegible] `..._unittest.rs` [illegible].
- [illegible] `BUILD.gn` [illegible].
- [illegible].

45 [illegible]

C++ [illegible]

[illegible] C++/Rust interop [illegible] Rust [illegible] CXX [illegible] Chromium [illegible].

[illegible] interface [illegible] (language boundary) [illegible] CXX [illegible] Rust [illegible] C++ [illegible] Rust [illegible].

[illegible] CXX [illegible].

[illegible]:

- [illegible] Rust [illegible] C++ [illegible] `#[cxx::bridge]` [illegible] C++ [illegible] Rust [illegible] out-of-sync) [illegible] (manual bindings [illegible])
- [illegible] Thunk [illegible] FFI [illegible] C-ABI [illegible] (Rust [illegible] FFI [illegible]) C [illegible].
- [illegible] (core types) [illegible]:
  - `&[T]` [illegible] FFI [illegible] ABI [illegible] `&[T` / `std::span<T>`] [illegible] (bindings) [illegible] pointer [illegible] length [illegible] slice [illegible])
  - `std::shared_ptr<T` ,`std::unique_ptr<T>` [illegible] Box [illegible] native [illegible] (manual bindings) [illegible] C-ABI-compatible raw pointers [illegible].
  - `rust::String` [illegible] `CxxString` [illegible] string [illegible] `rust::String::lossy` [illegible] Rust string [illegible] UTF8 [illegible] `rust::String::c_str` [illegible] string [illegible] NUL [illegible].)

([illegible]Binding) [illegible] 45.1

[illegible] `rs.` [illegible] `cxx::bridge` [illegible] C++/Rust [illegible] CXX
.[illegible]

```
                                   } mod ffi
                                } "extern "Rust
                               ;type MultiBuf

             ;[fn next_chunk(buf: &mut MultiBuf) -> &[u8
                                          {

                          } "++unsafe extern "C
              ;("include!("example/include/blobstore.h

                          ;type BlobstoreClient

       ;<fn new_blobstore_client() -> UniquePtr<BlobstoreClient
;<fn put(self: &BlobstoreClient, buf: &mut MultiBuf) -> Result<u64
                                          {
                                             {

            Definitions of Rust types and functions go here //
```

:[illegible]

- `[cxx::bridge]#` [illegible] Rust [illegible] mod [illegible] [illegible] - [illegible] .[illegible] `ffi` [illegible] mod [illegible]
- Rust [illegible]`std::unique_ptr`[illegible] C++ [illegible] Native [illegible]
- C++ [illegible] Rust Slices [illegible] Native [illegible]
- ([illegible]) Rust [illegible] Rust [illegible] C++ [illegible]
- ([illegible]) C++ [illegible] C++ [illegible] Rust [illegible]

[illegible] Rust [illegible] C++ [illegible] :**[illegible] C++ [illegible] `include#` [illegible] Rust [illegible] .[illegible] .[illegible] C++ [illegible]

CXX [illegible]

.[illegible] type reference [illegible] CXX [illegible]

:[illegible] CXX

- [illegible] Rust-C++ [illegible] .[illegible] declare
- [illegible] CXX [illegible] .[illegible] `[std::unique_ptr, std::string, &[u8`[illegible]

.Option' Rust' [illegible] --- [illegible]

[illegible] ”[illegible]” [illegible] Chromium [illegible] Rust [illegible] .Rust-C++ [illegible] Chromium [illegible] Rust [illegible]

CXX [illegible] (language boundary) [illegible].

[illegible] CXX [illegible]:

- [illegible] C++ exception [illegible] ([illegible])
- [illegible] Function pointer [illegible].

45.2 [illegible] CXX

CXX [illegible] Result<T,E> [illegible] C++ exception [illegible] Chromium [illegible]. [illegible]:

- [illegible] T [illegible] <T, E> [illegible]:
  - [illegible] (&mut T [illegible]) [illegible]. [illegible] T [illegible] FFI [illegible] - [illegible] T [illegible]:
    - * [illegible] type [illegible] ([illegible] u32 [illegible] usize)
    - * [illegible] native [illegible] cxx [illegible] ([illegible] UniquePtr<T>) [illegible] (*unlike* Box<T>).
  - [illegible] Rust [illegible]. [illegible] T [illegible] Rust [illegible] FFI [illegible] UniquePtr<T> [illegible].
- [illegible] E [illegible] Result<T, E> [illegible]:
  - [illegible] boolean [illegible] ([illegible] true [illegible] false [illegible])
  - [illegible].

45.2.1 [illegible] CXX: [illegible] QR

[illegible] QR [illegible] FFI [illegible]:

```
mod ffi {
    extern "Rust" {
        fn generate_qr_code_using_rust(
            data: &[u8],
            min_version: i16,
            out_pixels: Pin<&mut CxxVector<u8>>,
            out_qr_size: &mut usize,
        ) -> bool;
    }
}
```

[illegible] out_qr_size [illegible]. [illegible] QR [illegible] ([illegible] - [illegible]).

[illegible] Rust [illegible] out_qr_size [illegible]. [illegible] Rust [illegible]

.(□□□□□□ UB □□ □□□□□ □□□□□□□□□ □□□□□ □□□□□□ □□□□ □□□□ □□□□□ □□ □□□□□□ □C++ □□□□□□□)

□□□□□ □□□□□□□□□□ □□□□□ CXX □□□□ □□ □□□□□ □□□□□□ □□□□□□□□□ □□□□□ Pin □□□□□□□ □□□□□□□□□□□□ □□□□ □□□□□□ □□□□□□□□ □□ C++□□□□□□□□ □□ □□□□ □□□□ □□□□□ :□□□□□ □□□□□ □□ □□ C++ □□□□□□□□ □□ □□□□□□ self-referential) □□□□□□□□□□□ □□□□□□□□□□□ □□□□□ □□□□ □□□□□ □□□□□ □□□□□ □□□□□□□ Rust □□□□□□□□□ .□□□□□ (pointers

PNG □□□□□ :CXX □□□□ □□□□□□□□ 45.2.2

□□□□ □□ □□□□□□□□□□ □□□□□ □□□□□□□□ □□□□□□□ □□□□□ □□ □□□□□□□ □□□□□ PNG decoder □□□□□□ □□□□□□□ :□□□□ □□□□□□□ □□□□□ □□□ □□□□□ □□□ □ □□□□ □□□□□ FFI

```
} mod ffi
} "extern "Rust
,<This returns an FFI-friendly equivalent of `Result<PngReader<'a ///
.`<() ///
;<<fn new_png_reader<'a>(input: &'a [u8]) -> Box<ResultOfPngReader<'a

.C++ bindings for the `crate::png::ResultOfPngReader` type ///
;<type ResultOfPngReader<'a
;fn is_err(self: &ResultOfPngReader) -> bool
)<fn unwrap_as_mut<'a, 'b
,<self: &'b mut ResultOfPngReader<'a
;<b mut PngReader<'a'& <- (

.C++ bindings for the `crate::png::PngReader` type ///
;<type PngReader<'a
;fn height(self: &PngReader) -> u32
;fn width(self: &PngReader) -> u32
;fn read_rgba8(self: &mut PngReader, output: &mut [u8]) -> bool
{
{
```

□□□□□□□□□□□ □□□□ □□□□ □□ □□□□object --- □□□□□□□ Rust □□□□□□□□□ "ResultOfPngReader" □ "PngReader" `out_parameter:` □□□□□□□□□□□ □□ .□□□□□ □□□□□ FFI □□□□ □□ `<Box<T` □□□□□□□□□□□□ □□□□□□□□□ □□□□□ □□□□□□□ □□ □□ Rust object □□□□□□□□ □□□□□□□ C++ □□ CXX □□□□□ □□□□□□□□ □□□□□□□ `&mut PngReader` .□□□□ □□□□□□□ □□□□□□□

□□□□□□□□□□□ □□□□□□□□ □□□□template □ □□generic □□ CXX □□□□ □□□□ □□ □□□□□□□ □□□□□ □□□□□ □□□□ □□□□□ □□□□□□□□ □□ □□□□□ □ □□□□□ □□□□□ FFI □□□□ □□ □□ □□□□□ □□□□□□□□□□□ □□□□□□□□ □□ □□□□□□□□□□ □□□□□ □□ .□□□□□ □□□□□□□ □□□□□□□□□□□ □□□□ □□ □□ (specializing / monomorphizing) □□□□□□□□/□□□□□□□ □□□□□□ □□) `<Result<T, E` □□□□□□ □□□□□□□□ □□ □□ □□□□ non-generic □□□□ □□ `ResultOfPngReader` .□□□□□□ □□□□□□□ «as_mut» □□□/□ «is_err»□ «unwrap» □□□ □□□□□

Chromium □□ cxx □□ □□□□□□□□□

□□ □□□□□□□□□□ □□ □□□□□□ □□□□ □□ □□□□□ □□□□□□ `cxx::bridge] mod]#` □□ □Chromium □□ □□□□ `rust_static_library` □□ □□□□□ □□□□□□□□□ □□□□ .□□□□□□□□ □□□□□□□ □□ □□□□□ □□□□□□□□□ Rust .□□□□□ □□□□□□□ □□□□ □□ .□□□□□□□

```
[ "cxx_bindings = [ "my_rust_file.rs
list of files containing #[cxx::bridge], not all source files #
allow_unsafe = true
```

▯▯ ▯▯▯ `rust_static_library` ▯▯▯▯▯▯ ▯▯ ▯▯▯▯ `crate_root` ▯ `sources`.

header▯▯▯ C++ ▯▯ ▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯

```
#include "ui/base/my_rust_file.rs.h"
```

▯▯▯▯▯ ▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯ ▯▯ ▯▯ `//base` ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯/▯▯ ▯▯▯▯▯▯▯▯ Chromium C++ ▯▯ ▯▯▯▯▯▯ Rust CXX ▯▯▯▯▯ ▯▯▯▯▯▯▯▯ ▯▯▯ --- ▯▯▯▯▯ ▯▯▯▯▯ anToRustSlice](https://source.chromium.org/chromium/chromium/src/+/main:base/containers/span_rust.h;l=21).

▯▯▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯▯▯▯ --- ▯▯▯▯ ▯▯▯▯▯ ▯▯ `allow_unsafe = true` ▯▯▯▯▯ ▯▯▯▯▯▯▯.

▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯ ▯▯▯▯ ▯▯ ▯▯▯▯ ▯▯ C/C++ ▯▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯ Rust "▯▯▯▯▯" ▯▯▯▯▯. ▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯ ▯ ▯▯▯▯▯▯▯▯ ▯▯ C/C++ ▯▯ Rust ▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯ ▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯ ▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯ Rust ▯▯▯ ▯▯ ▯▯▯ ▯▯▯▯▯▯▯▯. ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯ «▯▯▯▯▯▯» ▯▯ C/C++ ▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯▯ ▯ ▯▯▯▯ [▯▯▯▯▯▯▯▯▯▯▯▯▯](https://steveklabnik.com/writing/the-cxx-debate) ▯▯▯▯▯ ▯▯▯ ▯▯ ▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯ ▯▯ ▯▯▯▯▯▯ ▯▯ ▯▯ ▯▯▯▯▯▯▯ Rust ▯▯▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯▯▯ Rust ▯▯▯▯.

▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯ --- ▯▯ ▯▯▯▯ ▯▯▯▯▯▯ CXX ▯▯▯▯▯▯ Rust «▯▯▯▯▯▯» ▯ `extern "C"` ▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯ ▯▯▯▯ ▯▯▯▯ ▯▯ ▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯.

45.3 ▯▯▯▯▯▯: ▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯ C++

▯▯▯▯ ▯▯▯▯

- ▯▯ ▯▯▯▯ Rust ▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯ ▯▯ `#[cxx::bridge]` ▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯ ▯▯ ▯▯▯▯▯ ▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯ C++▯▯ ▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯ ▯▯ ▯▯▯▯ `hello_from_rust` ▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯ ▯ ▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯.
- ▯▯▯▯▯ `hello_from_rust` ▯▯▯▯▯ ▯▯▯▯ ▯▯ ▯▯▯▯▯ ▯▯▯▯ `extern "C"` ▯ `#[no_mangle]` ▯▯▯▯▯▯. ▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯ ▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯ Rust ▯▯▯▯.
- ▯▯▯▯ gn ▯▯▯▯ ▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯▯▯▯▯▯ (bindings) ▯▯▯▯▯▯ ▯▯▯▯▯.
- ▯▯ C++ ▯▯▯▯▯ forward-declaration ▯▯▯▯▯ `hello_from_rust` ▯▯ ▯▯▯▯ ▯▯▯▯▯. ▯▯ ▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯ ▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯.
- Build ▯ run!

▯▯▯▯ ▯▯▯▯

▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯ ▯▯ ▯▯▯▯ CXX ▯▯ ▯▯▯▯▯ ▯▯▯▯▯. ▯▯▯ ▯▯ ▯▯▯▯ ▯▯▯▯ ▯▯ ▯▯▯▯ ▯▯ ▯▯ ▯▯▯▯ ▯▯▯▯ ▯▯▯▯▯ ▯▯ Rust ▯▯ Chromium ▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯.

▯▯▯▯▯ ▯▯ ▯▯▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯▯▯▯:

- ▯▯ Rust ▯▯▯▯▯▯▯ ▯▯ C++ ▯▯▯▯▯▯▯▯▯. ▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯▯▯▯:
  - ▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯▯▯▯▯ ▯▯ `cxx::bridge` ▯▯▯▯ `include!` ▯▯ ▯▯▯▯▯ ▯▯▯▯▯. ▯▯▯▯ ▯▯▯▯▯ C++ ▯▯▯▯ ▯▯ ▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯.
  - ▯▯ ▯▯▯▯▯ `unsafe` ▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯ ▯▯ ▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯ `unsafe` ▯▯ ▯▯ `#[cxx::bridge]` ▯▯▯▯ ▯▯▯▯▯▯▯▯▯ ▯▯ ▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯ ▯▯▯.
  - ▯▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯ ▯▯▯▯ ▯▯▯▯▯ `#include "third_party/rust/cxx/v1/crate/include/cxx.h"` ▯▯ ▯▯▯▯▯ ▯▯▯▯▯.
- ▯▯ ▯▯▯▯C++ ▯▯ C++ ▯▯ Rust ▯▯▯▯▯ ▯▯▯▯▯.
- ▯▯▯▯▯▯ ▯▯ reference ▯▯ object C++ ▯▯ Rust.
- ▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯ ▯▯▯▯ Rust ▯▯ ▯▯ ▯▯ `#[cxx::bridge]` ▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯▯ ▯▯▯▯▯▯▯ ▯▯▯▯▯ ▯ ▯▯ ▯▯▯▯▯▯▯▯ ▯▯ ▯▯ ▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯.

- [illegible] C++ [illegible] `#[cxx::bridge]` [illegible]
  [illegible].
- [illegible] `std::unique_ptr` [illegible] C++ [illegible] Rust [illegible] Rust [illegible]
  [illegible] C++ object [illegible].
- [illegible] Rust object [illegible] C++ [illegible] C++ [illegible]. ([illegible]:
  [illegible] Box [illegible]).
- [illegible] C++ [illegible]. [illegible] Rust [illegible].
- [illegible] Rust type [illegible]. [illegible] C++ [illegible].

[illegible]

[illegible] CXX interop [illegible] Rust [illegible] Chromium [illegible]. [illegible].

[illegible] help [illegible]

- [illegible] cxx binding reference
- [illegible] `rust_static_library` gn template

[illegible]: [illegible]:

- [illegible] X [illegible] Y [illegible] X [illegible] Y [illegible]. [illegible] C ++ [illegible] `cxx::bridge` [illegible].
- [illegible] C++ [illegible] Rust [illegible]. [illegible] UB [illegible] CXX *opaque* [illegible]. [illegible] CXX [illegible] UB [illegible] CXX [illegible].

[illegible] 46

[illegible] Crate[illegible] [illegible] [illegible]

[illegible] "Rust Crates" [illegible] [illegible] [illegible] crates.io [illegible] [illegible]. [illegible] crates[illegible] Rust [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]!

[illegible]	C++ library	Rust crate
Build system	[illegible]	[illegible]: `Cargo.toml`
[illegible]	Large-ish	[illegible]
[illegible]	Few	[illegible]

[illegible] [illegible] Chromium[illegible] [illegible] [illegible] [illegible]:

- [illegible] crate[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] Chromium [illegible] [illegible]...
- ... [illegible] crate[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

[illegible] [illegible]:

- [illegible] [illegible] crate [illegible] [illegible] Chromium
- [illegible] [illegible] gn [illegible] [illegible] [illegible]
- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

46.1 [illegible] `Cargo.toml` [illegible] [illegible] crate[illegible]

Chromium [illegible] [illegible] [illegible] [illegible] crate [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] `Cargo.toml` [illegible] [illegible]:

```toml
[dependencies]
bitflags = "1"
cfg-if = "1"
cxx = "1"
# lots more...
```

[illegible] `Cargo.toml` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] --- [illegible] [illegible] «[illegible]» [illegible] [illegible] crate [illegible] [illegible] [illegible] [illegible].

[illegible] crate [illegible] Chromium [illegible] gnrt_config.toml [illegible].

46.2 [illegible] gnrt_config.toml

[illegible] Cargo.toml [illegible] gnrt_config.toml [illegible] extension [illegible] Chromium [illegible] crate [illegible].

[illegible] crate [illegible] group [illegible]:

```
# 'safe': The library satisfies the rule-of-2 and can be used in any process.
# 'sandbox': The library does not satisfy the rule-of-2 and must be used in
#            a sandboxed process such as the renderer or a utility process.
# 'test': The library is only used in tests.
```

[illegible]

```
[crate.my-new-crate]
group = 'test' # only used in test code
```

[illegible] crate [illegible] ([illegible]) [illegible].

[illegible].

46.3 [illegible] Crate [illegible]

[illegible] gnrt [illegible] BUILD.gn [illegible].

[illegible] crate [illegible]:

```
cd chromium/src
vpython3 tools/crates/run_gnrt.py -- vendor
```

> [illegible] gnrt [illegible] Chromium [illegible] crates.io [illegible] [illegible] [illegible].

[illegible] vendor command [illegible]:

- [illegible] (crates) [illegible]
- [illegible]
- [illegible] crate [illegible] cargo [illegible] Chromium [illegible].

Chromium [illegible] crate [illegible] rty/rust/chromium_crates_io/patches// [illegible].

46.4 [illegible] gn Build

[illegible] crate [illegible] BUILD.gn [illegible]:

```
vpython3 tools/crates/run_gnrt.py -- gen
```

[illegible] `git status` [illegible]. [illegible]:

- [illegible] `third_party/rust/chromium_crates_io/vendor`
- [illegible] `BUILD.gn` [illegible] `third_party/rust/<crate name>/v<major semver version>`
- [illegible] `README.chromium` [illegible]

[illegible] Rust [illegible].

[illegible] `third_party/rust` [illegible].

[illegible] semver --- [illegible] Chromium [illegible] crate[illegible]. [illegible] Cargo [illegible].

46.5 [illegible]

[illegible] build [illegible] `build.rs` [illegible]: [illegible] build [illegible]. [illegible] `gn` [illegible] `ninja` [illegible] build [illegible] build[illegible].

[illegible] `build.rs` [illegible]. [illegible]:

[illegible]	[illegible] gn [illegible]	[illegible]
[illegible] rustc [illegible]	[illegible]	None
[illegible] CPU [illegible]	[illegible]	None
[illegible]	[illegible]	[illegible] - [illegible] `gnrt_config.toml` [illegible]
Building C/C++	[illegible]	[illegible] Patch
[illegible]	[illegible]	[illegible] Patch

[illegible] crate[illegible] build script [illegible] build script[illegible].

46.5.1 [illegible]

[illegible] `ninja` [illegible] `build.rs` [illegible].

□□ build-script-outputs □□ □□□□ □□□□□ □□ gnrt_config.toml □□□□□ □□□ □□
□□□□□ Chromium □□ □□ □□□□□□□□ □□□□ □□□□ □□□□ □□□□□□□□ □□ □□□ □□□ .□□□ □□□□□ crate
.□□□□ □□□□□ □□□ □□ allow-first-party-usage=false □□ □□□□□ □□□□□□ □□ □□ □□□□□□□□□
:□□□□ □□□□ □□□□ □□ □□ □□□ □□ □□□□□ □□□□□

```
[crate.unicode-linebreak]
allow-first-party-usage = false
["build-script-outputs = ["tables.rs
```

□□□□ □□□□□ □□□□□□□ □□ BUILD.gn □□□□□□□□ □□ □□□□ □□□□ □□□□□□ □□ gnrt.py -- gen □□□□□
.□□□ □□□□ build □□□□□ □□□□□ □□□ □□□□□ □□□□ □□□ □□ □□□□ □□□□□ ninja □□ □□

□□□□□ □□□□□□ □□□□□□□ □□ □□□□□ Build □□ C++ □□ □□□□□□□□□□□ □□□□ 46.5.2 □□□□□□

.□□□□□□□ □□□□□□□ ++C/C □□□□□□□□□□□ link □ build □□□□ cc □□ □□□□□ crate □□ □□crate □□ □□□□
.□□□□□□ □□□□□ □□□ build □□□□□□□□□□ □□ bindgen □□ □□□□□□□ □□ □□ ++C/C □□□□ □□□crate
□ gn□ ninja □□□□ □□□□□ --- □□□ □□□□□□□□ Chromium □□□□□ □□ □□□□□□□ □□ □□□□□□□□ □□□
.□□□ □□□ □□□□□ □□build actions □□□ □□□□□ □□□□ □□ □□ LLVM

:□□ □□□□□□□ □□□ □□□□□□□□□ □□□□□□□□□□

- □□□□ □□□□□□ □□crate □□□ □□
- .□□□□□ crate □□□ (patch) □□□□ □□

<third_party/rust/chromium_crates_io/patches/<crate□□ □□□□ (Patches) □□□□□□
in:third_party/rust/chromium_crates_io/patches/cxx) [□□□□□ □□ □□Patch] □□□□ □□□□-□□□□ □□□□□□□
.□□□□□ □□□□□ ‘gnrt□□□□ □□□□□□ □□□□□ □□□□□ upgrade □□ crate □□ □□□ □□ □ -

Crate □□ □□ □□□□□□□ 46.6

□□□ □□ □□□□ □□ □□ □□□□□ □□□□□ □□ build □□□□□ □ □□□□□ □□□□□ □□□□ □□□ crate □□ □□ □□□□□□
dep □□ □ □□□□ □□□□ □□ □□□ rust_static_library □□□ □□□□□□□ .□□□□ □□□□ □□□□□□□ crate
.□□□□ □□□□□ □□□ crate □□ lib: □□□ □□□

□□□□□□□□□□

```
        +----------------------+       +------------+
"third_party/rust" | crate name | "/v" | major semver version | ":lib//"
        +----------------------+       +------------+
```

□□□□□ □□□□□ □□

```
} ("rust_static_library("my_rust_lib
              "crate_root = "lib.rs
            [ "sources = [ "lib.rs
[ "deps = [ "//third_party/rust/example_rust_crate/v1:lib
                                                        {
```

□□□□ □□□ □□□Crate □□□□□□□ 46.7

□□□□□ □□□□□ □□□□□ □□□ □□□□ Chromium □□ □□□□□ □□□□□ □□□□ □□□□ □□□□□□□□□□ □□□□□□
□□□□ □□□□□□□□□□ □□□□ □crate □□ □□□□ □□ □□□ □□□□ □□ □□□□□□ □□ .□□□□ □□□□ □□□ □□□□□□

safe □□□□□□ □□□□ □□ .□□□□□ □□□□□□ □□□□□ □□□□□□□ □□□□□ □□□□□□ □□□□□□□ □□□□ □□□□□ □□□□□□ □□□□□ □□□□ □□ □□□□□□ □□□□□□ □□ □□ □□□□□ □□□□□□ □□ .□□□□□ □□□□□□ □□□□□□□□ □□□□□□□ □□□□□□ □□□□□□□□□ Rust code

.□□□ □□□□□ cargo vet □□□□□ □□ □□□□□□□□□ □□ □□□□□ □□□□ Chromium □□□□□□ □□□□□ □□

:□□□□□ □□ □□□□□□ □□ □□□□ □□□□□□ □□□□□ □□□□□□ □□□□□ crate □□ □□□□□ □□□□□ □□□□□ □□

- □□□□□ □□□□□□ □□□□ □□□□□□ □□crate □□□□ □□□□□□ .□□□□□□ □□□□□□□□□ crate □□ □□□□ □□ □□□□□□□ □□□□□ □□□□□ □□□□□ (procedural macros) □□□□□□□□ □□□□□□□□□□ □□ `build.rs` □□□□□ □□□□□ □□ □ □□□□□□ □□□□□□ □□□□ □□ Chromium □□ □□□□□ □□ □□□□□ □□□□ .□□□□□□ □□□□□ □□ □□□□□ □□□□□ □□ □□□□□□□ □□□□□□□ □□□□□□ built
- Check each crate seems to be reasonably well maintained
- .□□□□□ □□□□□□□□□ `cd third-party/rust/chromium_crates_io; cargo audit` □□ `cargo install` □□□□□ □□□□□□) □□□□□□□□□□□ □□□□□□□□□□□□□□ □□□□□□ □□□□□ cargo □□□□□□ `cargo-audit` □□ □□ □□□□ □□□□□ □□□□□□□ □□□□□□□□□□□□ □□□□□□□ □□ □□□□□□□□□ □□□□□□□(2
- □□□□ □□□□ □□ □□□□□□ □□□□□ □□□□□ □□□□□□□□ □□ `unsafe` □□ □□ □□□□□ □□□□□□□
- □□□□□ □□□□□□ □□ `net` □□ `fs` □□□API □□ □□□□□□□□□ □□□□□□□□
- □□□□□ □□□□ □□ □□□□ □□□□□ □□ □□□□□ □□ □□□□□□ □□ □□ □□□□□□□□□ □□□□□ □□□□ □□ □□ □□□□□ □□□□□ □□□□□□ □□□□□□ □□ □□□□□□□ □□□□□ □□□□ □□ □□□□□□□ □□□□ □□□□□□ □□) .□□□□□□□□ □□ □□□□□ □□□□ □□□□□ □□□□ □□□□□□ □□□□□□□ :□□□□□□ □□□□□□ □□□□□□ □□□□□ □□□□□.)

□□□ `security@chromium.org` □□ □□□□□□□□□□□□ □□ --- □□□□□□ □□□□□□□□□□□□□□□□□ □□□□ □□□□□□□ .□□□□□ □□□□□ crate □□ □□□□□□□□□ □□□□□ □□□□□□ □□□□ □□ □□□□□

Chromium □□□□□ □□ □□ □□Crate □□□□□□ 46.8

:□□□ □□□□□ □□□□□ `git status`

- `third_party/rust/chromium_crates_io//` □□ Crate □□
- `<README.chromium)` in `//third_party/rust/<crate>/<version` □ `BUILD.gn)` □□□□□□□□□

.□□□□□ □□□□□□ □□□□ □□□□□ □□□□□ □□ OWNERS□□□□ □□ □□□□□□□

□□□□□ □□ □□□□ `gnrt_config.toml` □ `Cargo.toml` □□□□□□□□□ □□□□□□ □□ □□ □□□□□□ □□□□ □□□□□ .□□□□□ □□□□□ Chromium

□□□□□ `gitignore.` □□□□□□□□□ □□□□□ □□□□ □□□□ □□ □□□□□ □□□□□ □□□□□□□□□ `git add -f` □□ □□□□□ :□□□□ .□□□ □□□□□□□□ □□ □□□□□ □□□□ □□ □□□□□ □□□□

As you do so, you might find presubmit checks fail because of non-inclusive language. This is because Rust crate data tends to include names of git branches, and many projects still use :non-inclusive terminology there. So you may need to run

```
e_language_presubmit_exempt_dirs.sh > infra/inclusive_language_presubmit_exempt_dirs.txt
d -p infra/inclusive_language_presubmit_exempt_dirs.txt # add whatever changes are yours
```

□□Crate □□□□□□ □□□□ □□□□ □□ 46.9

□□□□□□□□ □□ □□ □□ □□□□□□ □□□□□□□ □Chromium □□□□□ □□□□ □□□□□□□□□ □□ □□□□□ □□□□□□□□□ □□□□ □□□□□□□ Rust □□□crate □□□□□ □□ □□□□ □□□□□ □□ □□ □□ □□□□ □□□□□ .□□□□□□ □□□□ □□□□□□ □□□□□□□ □□□□□□□□□ □□□□ □□ □□□□□□□□□ □□ □□□□□ □□ □□□□□□□□□ □□□□□□□ □□□□□□□□□ □□□□□□□ □□□□□□ □□□□ □□ □□□□ □□□□□□□ .□□□□□□ □□ □□□□□□□□□ □□□□ □□□□□ □□□□□

46.10 [illegible]

Add uwuify to Chromium, turning off the crate's default features. Assume that the crate will be used in shipping Chromium, but won't be used to handle untrustworthy input.

([illegible] uwuify [illegible] Chromium [illegible]-[illegible]. [illegible] ['rust_executable'](https://source.chromium.org/chromium/chromium/src/+/main:build/rust/rust_executable.gni) [illegible] `uwuify` [illegible]).

[illegible].

[illegible] crate[illegible]:

- `instant`,
- `lock_api`,
- `parking_lot`,
- `parking_lot_core`,
- `redox_syscall`,
- `scopeguard`,
- `smallvec` [illegible],
- `uwuify`.

[illegible].

[illegible] Daniel Liu [illegible] crate!

[illegible] 47

[illegible] --- [illegible]

[illegible] Chromium [illegible].

[illegible]

[illegible] pixy [illegible]. [illegible] Chromium [illegible] pixy [illegible].

[illegible] string [illegible] Chromium [illegible] Pixie [illegible].

[illegible] pixie [illegible] Rust crate [illegible].

[illegible] crate [illegible].

([illegible] Chrome [illegible]. [illegible]!)

[illegible]

`ResourceBundle::MaybeMangleLocalizedString` [illegible] string [illegible]. [illegible] build [illegible] Chromium [illegible] `_mangle_localized_strings` [illegible].

[illegible] Chrome [illegible] pixies [illegible]!

- UTF16 [illegible] UTF8. [illegible] string [illegible] Rust [illegible] UTF8 [illegible] C++ [illegible] `base::UTF16ToUTF8` [illegible].
- [illegible] Rust [illegible] `String::from_utf16`](https://doc.rust-lang.org/std/string/struct.String.html#method] [illegible]. [illegible] CXX [illegible] u16 [illegible].

- [illegible] C++/Rust [illegible]. [illegible] string [illegible] string. [illegible] CXX [illegible] `Pin` [illegible]. [illegible] `Pin` [illegible] CXX [illegible] C++ [illegible]: [illegible] C++ [illegible] Rust [illegible] (self-referential pointers) [illegible].
- [illegible] C++ [illegible] `ResourceBundle::MaybeMangleLocalizedString` [illegible] `rust_static_library` [illegible]. [illegible].
- [illegible] `rust_static_library` [illegible] `//third_party/rust/uwuify/v0_2:lib` [illegible].

48 [illegible]

[illegible] [illegible]

[illegible] [illegible] Chromium [illegible] [illegible] [illegible] [illegible] [illegible] CLs [illegible] [illegible].

XI [illegible]

[illegible] :Bare Metal [illegible]

49 [illegible]

[illegible] Bare Metal Rust [illegible]

[illegible] bare-metal Rust [illegible] (Rust [illegible]) [illegible] Rust [illegible] bare-metal [illegible] C [illegible].

[illegible] bare-metal' Rust' [illegible] Rust [illegible]:

- no_std Rust [illegible]
- [illegible] firmware [illegible].
- [illegible] bootloader / kernel [illegible].
- [illegible] crate [illegible] bare-metal Rust [illegible].

[illegible] BBC micro:bit v2 [illegible].

[illegible] Nordic nRF52833 [illegible] LED [illegible] I2C [illegible] SWD [illegible].

[illegible]:

```
stall gcc-aarch64-linux-gnu gdb-multiarch libudev-dev picocom pkg-config qemu-system-arm
rustup update
rustup target add aarch64-unknown-none thumbv7em-none-eabihf
rustup component add llvm-tools-preview
cargo install cargo-binutils
/github.com/probe-rs/probe-rs/releases/latest/download/probe-rs-tools-installer.sh | sh
```

[illegible] plugdev [illegible] micro:bit [illegible]:

```
SYSTEM=="hidraw", ATTRS{idVendor}=="0d28", MODE="0660", GROUP="logindev", TAG+="uaccess
sudo tee /etc/udev/rules.d/50-microbit.rules
sudo udevadm control --reload-rules
```

[illegible] MacOS:

```
xcode-select --install
brew install gdb picocom qemu
brew install --cask gcc-aarch64-embedded
rustup update
rustup target add aarch64-unknown-none thumbv7em-none-eabihf
rustup component add llvm-tools-preview
cargo install cargo-binutils
/github.com/probe-rs/probe-rs/releases/latest/download/probe-rs-tools-installer.sh | sh
```

50 □□□

`no_std`

`core`

`alloc`

`std`

- Slices, `&str`, `CStr`
- …`NonZeroU8`
- `Option`, `Result`
- …`!Display`, `Debug`, `write`
- `Iterator`
- …`!panic!`, `assert_eq`
- `NonNull` and all the usual pointer-related functions
- `async/await` □ `Future`
- …`fence`, `AtomicBool`, `AtomicPtr`, `AtomicU32`
- `Duration`

- `Box`, `Cow`, `Arc`, `Rc`
- `Vec`, `BinaryHeap`, `BtreeMap`, `LinkedList`, `VecDeque`
- `!String`, `CString`, `format`

- `Error`
- `HashMap`
- `Mutex`, `Condvar`, `Barrier`, `Once`, `RwLock`, `mpsc`
- `File` and the rest of `fs`
- `println!`, `Read`, `Write`, `Stdin`, `Stdout` and the rest of `io`
- `Path`, `OsString`
- `net`
- `Command`, `Child`, `ExitCode`
- `spawn`, `sleep` and the rest of `thread`
- `SystemTime`, `Instant`

- .□□□ □□□□□□ RNG □□ `HashMap` □□
- .`std` re-exports the contents of both `core` and `alloc`

50.1 [illegible] no_std

```rust
use core::panic::PanicInfo;

fn panic(_panic: &PanicInfo) -> ! {
    loop {}
}
```

- [illegible].
- std provides a panic handler; without it we must provide our own.
- [illegible] crate [illegible] panic-halt [illegible].
- [illegible] eh_personality [illegible] panic = "abort" [illegible].
- [illegible] main [illegible]. [illegible]. [illegible] linker [illegible] Rust [illegible].

50.2 alloc

[illegible] alloc [illegible] global (heap) allocator [illegible].

```rust
extern crate alloc;
extern crate panic_halt as _;

use alloc::string::ToString;
use alloc::vec::Vec;
use buddy_system_allocator::LockedHeap;

static HEAP_ALLOCATOR: LockedHeap<32> = LockedHeap::<32>::new();

static mut HEAP: [u8; 65536] = [0; 65536];

pub fn entry() {
    // SAFETY: `HEAP` is only used here and `entry` is only called once.
    unsafe {
        // Give the allocator some memory to allocate.
        HEAP_ALLOCATOR.lock().init(HEAP.as_mut_ptr() as usize, HEAP.len());
    }

    // Now we can do things that require heap allocation.
    let mut v = Vec::new();
    v.push("A string".to_string());
}
```

- buddy_system_allocator [illegible] third-party crate [illegible] buddy system [illegible]. [illegible] crate [illegible].
- [illegible] const LockedHeap [illegible] (allocator) [illegible]. [illegible] 32**2 [illegible].

- [illegible] [illegible] crate [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `alloc` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]-[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] binary crate [illegible] [illegible] [illegible] [illegible].
- [illegible] [illegible] [illegible] [illegible] `panic_halt` crate [illegible] [illegible] [illegible] [illegible] `extern crate panic_halt as _` [illegible] [illegible] [illegible] panic handler [illegible] [illegible] [illegible] [illegible].
- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] entry point [illegible].

51 [illegible]

[illegible]

Cortex [illegible] reset handler[illegible] ([illegible]) `cortex_m_rt` crate [illegible] M [illegible].

```
extern crate panic_halt as _;

mod interrupts;

use cortex_m_rt::entry;

fn main() -> ! {
    loop {}
}
```

[illegible] (peripherals) [illegible].

- `cortex_m_rt::entry` [illegible] `cortex_m_rt::entry` [illegible] reset handler [illegible].
- [illegible] `cargo embed --bin minimal` [illegible]

51.1 MMIO [illegible]

(peripherals) [illegible] memory-map [illegible] IO [illegible]. [illegible] LED [illegible] micro:bit [illegible]:

```
extern crate panic_halt as _;

mod interrupts;

use core::mem::size_of;
use cortex_m_rt::entry;

/// GPIO port 0 peripheral address
const GPIO_P0: usize = 0x5000_0000;
```

```rust
// GPIO peripheral offsets
const PIN_CNF: usize = 0x700;
const OUTSET: usize = 0x508;
const OUTCLR: usize = 0x50c;

// PIN_CNF fields
const DIR_OUTPUT: u32 = 0x1;
const INPUT_DISCONNECT: u32 = 0x1 << 1;
const PULL_DISABLED: u32 = 0x0 << 2;
const DRIVE_S0S1: u32 = 0x0 << 8;
const SENSE_DISABLED: u32 = 0x0 << 16;

fn main() -> ! {
    // Configure GPIO 0 pins 21 and 28 as push-pull outputs.
    let pin_cnf_21 = (GPIO_P0 + PIN_CNF + 21 * size_of::<u32>()) as *mut u32;
    let pin_cnf_28 = (GPIO_P0 + PIN_CNF + 28 * size_of::<u32>()) as *mut u32;
    // SAFETY: The pointers are to valid peripheral control registers, and no
    // aliases exist.
    unsafe {
        pin_cnf_21.write_volatile(
            DIR_OUTPUT
                | INPUT_DISCONNECT
                | PULL_DISABLED
                | DRIVE_S0S1
                | SENSE_DISABLED,
        );
        pin_cnf_28.write_volatile(
            DIR_OUTPUT
                | INPUT_DISCONNECT
                | PULL_DISABLED
                | DRIVE_S0S1
                | SENSE_DISABLED,
        );
    }

    // Set pin 28 low and pin 21 high to turn the LED on.
    let gpio0_outset = (GPIO_P0 + OUTSET) as *mut u32;
    let gpio0_outclr = (GPIO_P0 + OUTCLR) as *mut u32;
    // SAFETY: The pointers are to valid peripheral control registers, and no
    // aliases exist.
    unsafe {
        gpio0_outclr.write_volatile(1 << 28);
        gpio0_outset.write_volatile(1 << 21);
    }

    loop {}
}
```

- [illegible] GPIO 0 [illegible] LED [illegible].

[illegible]:

```
cargo embed --bin mmio
```

51.2 Crate [illegible]

[illegible] svd2rust [illegible] wrapper [illegible] Rust [illegible] memory-map [illegible] CMSIS-SVD [illegible].

```rust
extern crate panic_halt as _;

use cortex_m_rt::entry;
use nrf52833_pac::Peripherals;

fn main() -> ! {
    let p = Peripherals::take().unwrap();
    let gpio0 = p.P0;

    // Configure GPIO 0 pins 21 and 28 as push-pull outputs.
    gpio0.pin_cnf[21].write(|w| {
        w.dir().output();
        w.input().disconnect();
        w.pull().disabled();
        w.drive().s0s1();
        w.sense().disabled();
        w
    });
    gpio0.pin_cnf[28].write(|w| {
        w.dir().output();
        w.input().disconnect();
        w.pull().disabled();
        w.drive().s0s1();
        w.sense().disabled();
        w
    });

    // Set pin 28 low and pin 21 high to turn the LED on.
    gpio0.outclr.write(|w| w.pin28().clear());
    gpio0.outset.write(|w| w.pin21().set());

    loop {}
}
```

- SVD (System View Description) [illegible] XML [illegible] memory map [illegible].
  - [illegible] field [illegible] register [illegible] peripheral [illegible] value [illegible].
  - SVD [illegible] crate [illegible].
- `cortex-m-rt` [illegible].

- □□□ `cargo install cargo-binutils` □□ □□□□□□ □□□□□□ □□□□□□□□□ `cargo objdump --bin pac -- -d --no-show-raw-insn` □□ □□□□□ □□□□□ □□ □□□□□□□ □□ □□□□□ □□□□□□□.

□□□□□ □□ □□:

```
cargo embed --bin pac
```

51.3 HAL crates

[□□□crate] HAL](https://github.com/rust-embedded/wesome-embedded-rust#hal-implementation-crates) □□□□ □□□□□□ □□ □□ □□□□□□□□□□□□□□ □□□□□□□□□□□□□□ □□ □□□□□□□ □□□□□ □□□□□□□□□ □□□□□□ □□□□□□ □□□□□□□. □□□□□□ □□□□□□□□□ □□□□□□□□□□ `embedded-hal` □□ □□□□□□□□□□□□□□ □□□□□□□.

```rust
extern crate panic_halt as _;

use cortex_m_rt::entry;
use embedded_hal::digital::OutputPin;
use nrf52833_hal::gpio::{p0, Level};
use nrf52833_hal::pac::Peripherals;

fn main() -> ! {
    let p = Peripherals::take().unwrap();

    // Create HAL wrapper for GPIO port 0.
    let gpio0 = p0::Parts::new(p.P0);

    // Configure GPIO 0 pins 21 and 28 as push-pull outputs.
    let mut col1 = gpio0.p0_28.into_push_pull_output(Level::High);
    let mut row1 = gpio0.p0_21.into_push_pull_output(Level::Low);

    // Set pin 28 low and pin 21 high to turn the LED on.
    col1.set_low().unwrap();
    row1.set_high().unwrap();

    loop {}
}
```

- `set_low` and `set_high` are methods on the `embedded_hal` `OutputPin` trait.
- □□□□□□□ □□ HAL crate□□ □□□□□ □□□□□□□ □□ □□□□□□□ □□□ Cortex-M □ RISC-V □□ □□□□ □□□□□□□□□□□□□□□ AVR □MSP430 □NXP □nRF □GD32 □STM32 □ PIC □□□□□□ □□□□□ □□□□□.

□□□□□ □□ □□:

```
cargo embed --bin hal
```

51.4 Board support crates

Board support crates provide a further level of wrapping for a specific board for convenience.

```
;_ extern crate panic_halt as

;use cortex_m_rt::entry
;use embedded_hal::digital::OutputPin
;use microbit::Board

} ! <- ()fn main
;()let mut board = Board::take().unwrap

;()board.display_pins.col1.set_low().unwrap
;()board.display_pins.row1.set_high().unwrap

{} loop
{
```

- [illegible] crate [illegible].
- [illegible] crate [illegible].
  - – [illegible] microbit-v2 [illegible] LED [illegible].

[illegible]:

```
cargo embed --bin board_support
```

51.5 [illegible] state pattern

```
} ! <- ()fn main
;()let p = Peripherals::take().unwrap
;(let gpio0 = p0::Parts::new(p.P0

;let pin: P0_01<Disconnected> = gpio0.p0_01

.let gpio0_01_again = gpio0.p0_01; // Error, moved //
;()let mut pin_input: P0_01<Input<Floating>> = pin.into_floating_input
} ()if pin_input.is_high().unwrap
... //
{
let mut pin_output: P0_01<Output<OpenDrain>> = pin_input
;(into_open_drain_output(OpenDrainConfig::Disconnect0Standard1, Level::Low.
;()pin_output.set_high().unwrap
.pin_input.is_high(); // Error, moved //

let _pin2: P0_02<Output<OpenDrain>> = gpio0
p0_02.
;(into_open_drain_output(OpenDrainConfig::Disconnect0Standard1, Level::Low.
= <<let _pin3: P0_03<Output<PushPull
;(gpio0.p0_03.into_push_pull_output(Level::Low

{} loop
```

}

- □□□□□□ Copy □□ Clone □□ □□□□ □□□□□□□□□ □□□□□□□□□ □□□□ □□ □□□□□□ □□ □□ □□□□□ □□□□□□□□ □□□□ □□□□□□ □□□□□. □□□□□□□ □□ □□ pin □□ □□□□□□□ □□□□□ □□□□□ □□ □□□□□ □□□ □□ □□□□□□ □□□□ □□□□□□ □□ □□ □□□□□□.
- □□□□□□ □□□□□□□□□ pin□ □□□□□□ pin □□□□□□ □□ □□□□ □□□□□□□ □□□□□□□□□ □□□□□□□□□□ □□ □□ □□ pin □□□□□□ □□□□□□□□ □□□□□.
- □□□□ type □□ state □□□□□□ □□ □□□□ □□□□□□ □□ □□ □□ □□□□□ □□□□□: □□ □□□□□□ □□□□□. □□ □□□□ □□□□□□ □□□□□□ □□□□□□□□□ □□ □□□ GPIO. □□□□ state machine □□ □□ type system □□□□□□□□□ □□□□□□ □ □□□□□□ □□□□□□ □□ □□□□ □□□□□□ □□ pin □□ □□□□□ □□□□ □□□□□□□□ □□□□□ □□□□□ □□□□□□ □□□□□□ □□ □□ □□ □□□□□□ □□□□□□□□□ □□□□□. state transition □□□□□□□□ □□ □□□□□ □□□□□□□□ □□□□□□□□□ □□□□□□.
- □□□□□□□□□ `is_high` □□ □□ □□ □□□□ □□□□□□ □ `set_high` □□ □□ □□ □□□□□ □□□□□□ □□□□□□□□□ □□□□□□ □□□□ □□□□□□ □□□□□□ □□□□□ □□□□□.
- □□□□□□□ □□ HAL crate□□ □□ □□□□ □□□□□ □□□□□□ □□□□□□□.

51.6 embedded-hal

□□□ embedded-hal' crate' □□□□□□□ □□□□□□ □□ □□□□□□ □□□□□□ □□ □□□□□□□□□□□□□□□ □□□□□□ □□□□□ □□ □□□□ □□□□□□:

- GPIO
- PWM
- □□□□□□□ □□□□□□□□□
- □□□□□□□□□ □ □□□□□□□□□□ I2C □ SPI

□□□□□□□□□ □□□□□□ □□□□ □□□□□□□□□ □□□□□ (□□□□□□ UART)□ □□□□□□□□□□ CAN □ RNG □ □□□□□□ □□□ □□ [embedded-can](https ://crates.io/crates/embedded-can □(embedded-io □ `rand_core` □□ □□□□□□.

□□□ crate□□□ □□□□□ □□□□□□□□□□ □□ □□ □□□ □□□□ □□□□□□□□□ □□□□□□□□□□□ □□□□□□□□□ □□ □□□□□□ □□□□□. □□ □□□□□□□ □□□□□ □□□□ □□□□□ □□□□ □□ □□ □□□□□□ □□□□□□□ I2C □□ SPI □□□□□ □□□□□□ □□□□□.

- □□□ □□□□□□□□□ □□ □□□□□□□□□ □□ □□□□□□ □□□□□□(peripherals) □□□□□ □□□□□□□□□ □□□□ □□□□□ □□ □□□□□□□□□□ □□ □□□□□□ □□□□□□□□□□ □□□□□□□□□□ □□□□□ □□□□□□□□□□ □□□□□□ □ □□□□□□□□□□ □□□□□□□□□ □□ □□□□□□□ □□□□ □□□□□□ □□□□□.
- □□□□□□□□□□□□□□□□ □□□□□ □□□□□□□ □□ □□□□□□□□□□□□□□□ □ □□□□□□□ □□□□□□□□□□□ □□□□□□ □□□□□□ □□□□□□□ □□ Raspberry Pi □□□□□ □□□□□.
- □□□□`embedded-hal-async` □□□□□□□□ async □□ trait□□ □□ □□□□□□ □□ □□□□.
- □□□□□ `embedded-hal-nb` □□□□□□□ □□□□□□ □□ □□□□□ □□□□ □□□□□□ □□□□□ I/O □□□□□□ □□ □□□□ □□ □□ □□□□□ [crate (nb](https:// crates.io/crates/nb □□□.

51.7 probe-rs and cargo-embed

□□ probe-rs □□ □□□□□□□ □□□□□□ □□□□□ □□□□□ □□□□□□ □□□□□□ □□□□□□□ □□□□ □□□□□ □□□□□□ OpenOCD □□□□□ □□□□ □□□□□□□□ □□□□ □□□□.

- (SWD (Serial Wire Debug □ JTAG □□ □□□□□ □□□□□ □□□ ST-Link □CMSIS-DAP □ J-Link
- GDB stub □ Microsoft DAP (Debug Adapter Protocol) server
- □□□□□□ Cargo

buddy_system_allocator□□ third-party crate □□□ □□ □□ □□□□□□□□□□ buddy system □□ □□□□□□□□□□ □□□□□□. crate□□□ □□□□□ □□ □□□□□□ □□□□□□ □□ □□□□□□□□□ □□□□□ □□□□□□ □□ □□□ □□ □□□□□□□□□ □□ □□ □□□□□□ □□□□□□ □□□□□□ □□□□ □□□□□ □□□□□.

- CMSIS-DAP □□ □□□□□□□ □□□□□□□□□□□ ARM □□ □□□□□ USB □□□□ □□ □□□□□ □□ □□□□□□□ □□□□□ □□□□□□ □□□ □□□□□□□ □□ □□□□ CoreSight Debug Access □□ □□□□□□ □□□□□□ □□□□□□□□□□□□ Arm Cortex □□□□ □□□□□□□□ □□□□ □□□□□□ □ □□□ □□□□ □□□□□ □□□ □□ □□□□□□□ □□□□□□ □□ BBC micro:bit □□ □□ □□□□□□□□ □□□□□□.
- ST-Link □□ □□□□ □□□□□□□□□□ □□□□ □□□□ □□ ST Microelectronics □□□□ J-Link □□□□□□□ □□ SEGGER □□□.
- □□□□ □□□□□□ Debug □□□□□□ □□ □□ □□□□ JTAG 5 □□□ □□ Serial Wire Debug 2 □□□ □□□.
- probe-rs □□ □□□□□□□□□ □□□ □□ □□ □□□□□ □□□□□□ □□□□□□□□□ □□ □□ □□ □□□□□□□□□□ □□□□ □□□□□□ □□□□□.
- □□□□□□□ □□□□□□□□ Debug □□□□□□□□□□□ □□ VSCode □ □□□□ IDE□□ □□□□□□ □□□□□□ □□□□□□ □□□□□□ □□ □□ □□□□□□□□□□□ □□□□□□□□□□□□□ □□ Debug □□□□□.
- □□□ cargo-embed □□ □□□□□□ □□□ □□ □□ □□□□□□□□ □□ □□□□□□□□□ probe-rs □□□□□□ □□□ □□□.
- Real Time Transfers (RTT) □□□□□□□□□ □□□□□ □□□□□□□ □□□□□□□ □□□ debug host □ target □□ □□□□□ □□□□□□□ □□□□□ □□□□□□□ (ringbuffers) □□□□.

51.7.1 □□□□□ □□□□□□ (Debugging)

Embed.toml:

```toml
[default.general]
chip = "nrf52833_xxAA"

[debug.gdb]
enabled = true
```

□□ □□□□□□□□□ □□□ `/src/bare-metal/microcontrollers/examples`:

```
cargo embed --bin board_support debug
```

□□ □□□□□□□□ □□□□□□ □□ □□□□□ □□□□□□□□□□:

□□ gLinux □□ Debian:

```
rch target/thumbv7em-none-eabihf/debug/board_support --eval-command="target remote :1337
```

□□ MacOS:

```
gdb target/thumbv7em-none-eabihf/debug/board_support --eval-command="target remote :1337
```

□□ GDB□ □□□□□ □□□□□:

```
b src/bin/board_support.rs:29
b src/bin/board_support.rs:30
b src/bin/board_support.rs:32
c
c
c
```

51.8 □□□□□ □□□□□□□□□□

- RTIC

  - ”[illegible]”
  - [illegible] (task scheduling)[illegible] (timer queue)
- Embassy
  - [illegible] async [illegible] USB
- TockOS
  - RTOS [illegible]
- Hubris
  - [illegible] Microkernel RTOS [illegible] Oxide Computer [illegible] protection [illegible] IPC [illegible]
- Bindings for FreeRTOS
- [illegible] std [illegible]. esp-idf.
- RTIC [illegible] RTOS [illegible] (concurrency framework) [illegible].
  - [illegible] HAL [illegible].
  - [illegible] Cortex-M NVIC ([illegible] Nested Virtual Interrupt Controller) [illegible].
  - Cortex-M [illegible].
- [illegible] TockOS [illegible] Haven [illegible] Titan [illegible].
- [illegible] FreeRTOS [illegible] C [illegible] Rust [illegible].

52 □□□

□□□□□□□□□

.□□□□□□□ □□□□ □□□□□□ □□□□□ □□ □□ □□ □□□□□□□□□ □ □□□□□□□□□□ I2C □□□□□□□□□ □□ □□ □□□□ □□

□□□□□ □□□□ □□□□□□ (solutions-morning.md) [□□□□□ □□□□] □□ □□□□□□□□□□ □□□□□□□□□□ □□□□□ □□ □□
.□□□□□

□□□□□□□□   52.1

□□□□ □□□□ .□□□□□□□ □□□□ □□□□□□ □□□□□ □□ □□ □□ □□□□□□□□□ □ □□□□□□□ □□ I2C □□□□□□□□□ □□ □□ □□□□ □□
□□□□□□□□□ □□□□□□□ □□ □□□□□ □□ □□ □□□□□ □□□□□□ □□□□ □□LED □□□□ □□□□□ □□ □□ □□ □□□□□ □□□□ □□□□□□□□
.□□□□□

:□□□□□□□

- □□ □□□□□□□□□ □□□□□ □□□□□□ □□ □□ □□□crate □□□□□□ □□ microbit-v2 □ lsm303agr □□□□□□□□□
.□□□□□□□ □□□□□□ □□ micro:bit hardware
- .□□□□ □□□□□ I2C □□□□□□ bus □□ LSM303AGR □□□□□ □□ □□□□□□□□ □□□□□□□□□□□□ □□□□□
- .□□□□□□ □□□□□□□ I2C TWIM □□□□□ □□□□□□ □□□□□□□ □□□□□□□□□□ □□□□□ I2C □□□□□ □□□□□□ □□□□ TWI
- □□□□□ □□ `embedded_hal::i2c::I2c` □□□□□□ □□ □□□□□ □□□□□ □□□□□ □□ LSM303AGR □□□□□□□
.□□□□□□ □□□□□□□□□□□ □□ □□□□□ □□□□ `microbit::hal::Twim`□□□□□□□ .□□□□
- □□□□□□ □□□□□□ □□□□□□□□□ □ □□□□□□ □□□□□ □□□□□□□□□□ □□ 'microbit::Board' □□□□□□□ □□ □□□□
.□□□□□□
- □□□□□□ □□□□ □□□□□ □□□□ □□□□□□ □□□□□ □□□□ nRF52833 datasheet □□ □□□□□□□□□□ □□□□□□ □□□□□ □□
.□□□□□ □□□□□

.□□□□□ □□□□□□ `compass` □□□□□□□□□□□ □□ □□ □□□□ □□□□□□□□□ □ □□□□□ □□□□□□□ □□ □□□□□□ □□□□□□ □□□□

:*src/main.rs*

```
;_ extern crate panic_halt as

;use core::fmt::Write
;use cortex_m_rt::entry
;{use microbit::{hal::{Delay, uarte::{Baudrate, Parity, Uarte}}, Board

} ! <- ()fn main
    ;()let mut board = Board::take().unwrap
```

```
                                  .Configure serial port //
                               )let mut serial = Uarte::new
                                          ,board.UARTE0
                                     ,()board.uart.into
                                      ,Parity::EXCLUDED
                                  ,Baudrate::BAUD115200
                                                         ;(

               .Use the system timer as a delay provider //
                    ;(let mut delay = Delay::new(board.SYST

.Set up the I2C controller and Inertial Measurement Unit //
                                                    TODO //

                       ;()writeln!(serial, "Ready.").unwrap

                                                     } loop
    .Read compass data and log it to the serial port //
                                                TODO //
                                                          {
                                                              {
```

:(□□□□□□ □□□□□□ □□ □□□□□) *Cargo.toml*

```
           [workspace]

             [package]
      "name = "compass
     "version = "0.1.0
      "edition = "2021
       publish = false

        [dependencies]
 "cortex-m-rt = "0.7.3
"embedded-hal = "1.0.0
   "lsm303agr = "1.1.0
"microbit-v2 = "0.15.1
  "panic-halt = "0.2.0
```

:(□□□□□ □□□□ □□□□□□ □□ □□□□□□) *Embed.toml*

```
      [default.general]
"chip = "nrf52833_xxAA

           [debug.gdb]
        enabled = true

         [debug.reset]
halt_afterwards = true
```

:(*cargo/config.toml* (you shouldn't need to change this.

```
[build]
```

```
target = "thumbv7em-none-eabihf" # Cortex-M4F

['(("target.'cfg(all(target_arch = "arm", target_os = "none]
["rustflags = ["-C", "link-arg=-Tlink.x
```

:□□ □□□□□□□ □□ □□□□□□ □□□□□□ □□□□□□□

```
picocom --baud 115200 --imap lfcrlf /dev/ttyACM0
```

:(□□□□□ □□□□□□□ □□□□ □□□□ □□□□□ □□□□□□□ □□□□) □□ □□□□□ □□□□□ Mac □□□□□□□□□□ □□ □□

```
picocom --baud 115200 --imap lfcrlf /dev/tty.usbmodem14502
```

.□□□□□ □□□□□□□□□ Ctrl+Q □ Ctrl+A □□ picocom □□ □□□□□ □□□□□

Bare Metal Rust □□□□□□□□□ □□□□□□   52.2

□□□□□□□

(back to exercise)

```
;_ extern crate panic_halt as

;use core::fmt::Write
;use cortex_m_rt::entry
;{use core::cmp::{max, min
;use embedded_hal::digital::InputPin
}::use lsm303agr
,AccelMode, AccelOutputDataRate, Lsm303agr, MagMode, MagOutputDataRate
;{
;use microbit::display::blocking::Display
;use microbit::hal::twim::Twim
;{use microbit::hal::uarte::{Baudrate, Parity, Uarte
;{use microbit::hal::{Delay, Timer
;use microbit::pac::twim0::frequency::FREQUENCY_A
;use microbit::Board

;const COMPASS_SCALE: i32 = 30000
;const ACCELEROMETER_SCALE: i32 = 700

} ! <- ()fn main
;()let mut board = Board::take().unwrap

.Configure serial port //
)let mut serial = Uarte::new
,board.UARTE0
,()board.uart.into
,Parity::EXCLUDED
,Baudrate::BAUD115200
;(

.Use the system timer as a delay provider //
```

```
                                        ;(let mut delay = Delay::new(board.SYST

                    .Set up the I2C controller and Inertial Measurement Unit //
                                ;()IMU...").unwrap □□□□□□□ □□□" ,writeln!(serial
;(let i2c = Twim::new(board.TWIM0, board.i2c_internal.into(), FREQUENCY_A::K100
                                    ;(let mut imu = Lsm303agr::new_with_i2c(i2c
                                                           ;()imu.init().unwrap
                                                      )imu.set_mag_mode_and_odr
                                                                ,mut delay&
                                                   ,MagMode::HighResolution
                                                   ,MagOutputDataRate::Hz50
                                                                              (
                                                                     ;()unwrap.
                                                    )imu.set_accel_mode_and_odr
                                                                ,mut delay&
                                                         ,AccelMode::Normal
                                                 ,AccelOutputDataRate::Hz50
                                                                              (
                                                                     ;()unwrap.
                         ;()let mut imu = imu.into_mag_continuous().ok().unwrap

                                                   .Set up display and timer //
                                      ;(let mut timer = Timer::new(board.TIMER0
                            ;(let mut display = Display::new(board.display_pins

                                                  ;let mut mode = Mode::Compass
                                                ;let mut button_pressed = false

                                           ;()unwrap.(".□□□□□□" ,writeln!(serial

                                                                         } loop
                        .Read compass data and log it to the serial port //
                           ()while !(imu.mag_status().unwrap().xyz_new_data
                         (()imu.accel_status().unwrap().xyz_new_data &&
                                                                         {}
                       ;()let compass_reading = imu.magnetic_field().unwrap
                   ;()let accelerometer_reading = imu.acceleration().unwrap
                                                                  )!writeln
                                                                ,serial
                                                  ,"{},{},{}t\{},{},{}"
                                                ,()compass_reading.x_nt
                                                ,()compass_reading.y_nt
                                                ,()compass_reading.z_nt
                                          ,()accelerometer_reading.x_mg
                                          ,()accelerometer_reading.y_mg
                                          ,()accelerometer_reading.z_mg
                                                                          (
                                                                 ;()unwrap.

                                               ;[let mut image = [[0; 5]; 5
                                                  } let (x, y) = match mode
```

```
                                                      ) <= Mode::Compass
 (scale(-compass_reading.x_nt(), -COMPASS_SCALE, COMPASS_SCALE, 0, 4
                                                       ,as usize
  (scale(compass_reading.y_nt(), -COMPASS_SCALE, COMPASS_SCALE, 0, 4
                                                       ,as usize
                                                                      ,(
                                                ) <= Mode::Accelerometer
                                                              )scale
                                   ,()accelerometer_reading.x_mg
                                           ,ACCELEROMETER_SCALE-
                                            ,ACCELEROMETER_SCALE
                                                              ,0
                                                              ,4
                                                         ,as usize (
                                                              )scale
                                  ,()accelerometer_reading.y_mg-
                                           ,ACCELEROMETER_SCALE-
                                            ,ACCELEROMETER_SCALE
                                                              ,0
                                                              ,4
                                                         ,as usize (
                                                                      ,(
                                                                          ;{
                                                          ;image[y][x] = 255
                                       ;(display.show(&mut timer, image, 100

If button A is pressed, switch to the next mode and briefly blink all LEDs //
                                                                      .on //
                               } ()if board.buttons.button_a.is_low().unwrap
                                                    } if !button_pressed
                                                 ;()mode = mode.next
                       ;(display.show(&mut timer, [[255; 5]; 5], 200
                                                                       {
                                                  ;button_pressed = true
                                                                    } else {
                                                 ;button_pressed = false
                                                                           {
                                                                               {
                                                                                   {

                                                                         } enum Mode
                                                                        ,Compass
                                                                  ,Accelerometer
                                                                                   {

                                                                         } impl Mode
                                                         } fn next(self) -> Self
                                                                } match self
                                   ,Self::Compass => Self::Accelerometer
                                   ,Self::Accelerometer => Self::Compass
                                                                           {
                                                                               {
                                                                                   {

 } fn scale(value: i32, min_in: i32, max_in: i32, min_out: i32, max_out: i32) -> i32
                                                 ;let range_in = max_in - min_in
                                              ;let range_out = max_out - min_out
        (cap(min_out + range_out * (value - min_in) / range_in, min_out, max_out
                                                                                   {

                         } fn cap(value: i32, min_value: i32, max_value: i32) -> i32
                                             ((max(min_value, min(value, max_value
                                                                                   {
```

XII □□□

□□□ :Bare Metal □□

53 □□□

Application processors

□□□□□□□ □□□□□ .□□□□□□ □□□□□ Arm Cortex-M □□□□ □□□□□□ □□□□□□□□□□□□□□□□□ □□□□□ □□ □□□□□□ □□ 'QEMU's aarch64 'virt□□□ □□ □□□□ □□ □□□□□□□ □□□□□ .□□□□□□□□ Cortex-A □□□□□ □□□□□ □□□□□ □□□□ .□□□□□□□ □□□□

- □□ □□□□□□□ □□□□□) □□□□□□□ □□□□ □□□□□□ □□ MMU □□□□□□ □□□□□□□□□□□□□□□ □□□□□ □□□□ □□ □□□□□□ □□□□□□□ □□□□□□□□□□□□ □□ □□□□□ □□ □□□□□□□□ (x86 □□ □□□□□□□ □Arm □□□□□□□□□□□□ .□□□□□□ □□□□□□□
- □□□□□□□□□ □□□□□□□ □□ □□□□□ □□□□□□ □□□□ □□□□□□□□ □□ □□□□□□ □□□□□□□□□ □□□□□□□□□ □□ QEMU □□□□□□□□□ □□□□□ □□□□□ □□□□ □□□□□□□ □□□□□□□ □□□□□ □□□□□□ □□□□□□□□□ □□□□ □□ 'virt' □□□□ .□□□ □□ .□□□□ □□□□ □□□□□□ □□□□□□

Rust □□□□□ □□□□ □□□□□□ 53.1

□□ □□□□□□ □□□□□□□□□ □□□□□□□ □□□□□ □□□□□□ □□□□□ □□ Rust □□ □□□□□□ □□□□□□□□ □□□□□□ □□ □□□□ .□□□□□ □□□□□□

```
                                             "section .init.entry, "ax.
                                                        global entry.
                                                              :entry
                                                                  */
Load and apply the memory management configuration, ready to enable MMU and *
                                                            .caches *
                                                                  /*
                                                      adrp x30, idmap
                                                   msr ttbr0_el1, x30

                                                 mov_i x30, .Lmairval
                                                   msr mair_el1, x30

                                                  mov_i x30, .Ltcrval
                        /* .Copy the supported PA range into TCR_EL1.IPS */
                                              mrs x29, id_aa64mmfr0_el1
                                                 bfi x30, x29, #32, #4
```

```
msr tcr_el1, x30

mov_i x30, .Lsctlrval

*/
Ensure everything before this point has completed, then invalidate any *
.potentially stale local TLB entries before they start being used *
/*
isb
tlbi vmalle1
ic iallu
dsb nsh
isb

*/
Configure sctlr_el1 to enable MMU and cache and don't proceed until this *
.has completed *
/*
msr sctlr_el1, x30
isb

/* .Disable trapping floating point access in EL1 */
mrs x30, cpacr_el1
(orr x30, x30, #(0x3 << 20
msr cpacr_el1, x30
isb

/* .Zero out the bss section */
adr_l x29, bss_begin
adr_l x30, bss_end
cmp x29, x30  :0
b.hs 1f
stp xzr, xzr, [x29], #16
b 0b

/* .Prepare the stack */  :1
adr_l x30, boot_stack_end
mov sp, x30

/* .Set up exception vector */
adr x30, vector_table_el1
msr vbar_el1, x30

/* .Call into Rust code */
bl main

/* .Loop forever waiting for interrupts */
wfi  :2
b 2b
```

- [illegible] C [illegible]: [illegible]

BSS [illegible] stack pointer.
  - BSS [illegible] ([illegible]) [illegible] object file [illegible]. [illegible]. [illegible].
- [illegible] BSS [illegible].
- [illegible] MMU [illegible] cache [illegible]. [illegible]:
  - [illegible]. [illegible] Rust [illegible] aarch64-unknown-none [illegible] +strict-align [illegible].
  - [illegible] VM [illegible] cache [illegible]. [illegible] VM [illegible] cache [illegible] host [illegible] cache [illegible]. [illegible] cache [illegible] cache [illegible] cache [illegible] cache [illegible] VM, [illegible]. ([illegible] cache [illegible] VA [illegible] IPA.)
- [illegible] pagetable [illegible] ([illegible] idmap.S [illegible]) [illegible] DRAM [illegible]. [illegible] QEMU [illegible].
- [illegible] vbar_el1 (exception vector) [illegible].
- [illegible] 1 (EL1) [illegible]. [illegible] entry.S [illegible].

53.2 Inline assembly

[illegible] Rust [illegible]. [illegible] hypervisor call (HVC) [illegible] firmware [illegible]:

```rust
use core::arch::asm;
use core::panic::PanicInfo;

mod exceptions;

const PSCI_SYSTEM_OFF: u32 = 0x84000008;

extern "C" fn main(_x0: u64, _x1: u64, _x2: u64, _x3: u64) {
    // SAFETY: this only uses the declared registers and doesn't do anything
    // with memory.
    unsafe {
        asm!("hvc #0",
            inout("w0") PSCI_SYSTEM_OFF => _,
                    ,_ <= inout("w1") 0
                    ,_ <= inout("w2") 0
                    ,_ <= inout("w3") 0
                    ,_ <= inout("w4") 0
                    ,_ <= inout("w5") 0
                    ,_ <= inout("w6") 0
                    ,_ <= inout("w7") 0
                (options(nomem, nostack
                                        ;(
                                            {

                                    {} loop
                                                {
```

([illegible] crate [illegible] `smccc` [illegible] [illegible] (wrapper)[illegible] (.[illegible]

- PSCI [illegible] Arm Power State [illegible] power [illegible] CPU [illegible] EL3 [illegible] hypervisor [illegible].
- `_ <= 0` [illegible] syntax [illegible] 0 [illegible] `inout` [illegible] `in` [illegible].
- `main` [illegible] `#[no_mangle]` [illegible] `extern "C"` [illegible] (entry point) [illegible] `entry.S` [illegible].
- `x0–_x3_` [illegible] `x0–x3` [illegible] bootloader [illegible] device tree [illegible] aarch64 [illegible] `extern "C"` [illegible] `x0–x7` [illegible] `entry.S` [illegible].
- QEMU [illegible] `make qemu_psci` [illegible] `src/bare-metal/aps/examples` [illegible].

53.3 MMIO [illegible]

- `pointer::read_volatile` [illegible] `pointer::write_volatile` [illegible].
- reference [illegible].
- `addr_of!` [illegible].
- (Volatile access): [illegible].
  – [illegible] reference [illegible] memory [illegible].
- crate [illegible] (volatile access) [illegible] reference [illegible] reference [illegible] reference [illegible].

- □□□□□□□ □□ □□□□□□□□ □□ □□ struct field □□□□□□□□□□□ □□□□□□□ □□□□ !addr_of □□□□□□ □□
.□□□□ □□□□□□□□

□□□□□□□□ UART □□□□□□□ □□ □□□□□□□ 53.4

□□ □□□□□ □□□□□□□ □□ □□□□□□□ □□ □□□□□□ UART □□□□□□□□□ PL011 □□ 'QEMU 'virt □□□□□□ □□□□
.□□□□□□□□□

```
;const FLAG_REGISTER_OFFSET: usize = 0x18
;const FR_BUSY: u8 = 1 << 3
;const FR_TXFF: u8 = 1 << 5

.Minimal driver for a PL011 UART ///
} pub struct Uart
,base_address: *mut u8
{

} impl Uart
Constructs a new instance of the UART driver for a PL011 device at the ///
.given base address ///
///
Safety # ///
///
The given base address must point to the 8 MMIO control registers of a ///
PL011 device, which must be mapped into the address space of the process ///
.as device memory and not have any other aliases ///
} pub unsafe fn new(base_address: *mut u8) -> Self
{ Self { base_address
{

.Writes a single byte to the UART ///
} (pub fn write_byte(&self, byte: u8
.Wait until there is room in the TX buffer //
{} while self.read_flag_register() & FR_TXFF != 0

SAFETY: We know that the base address points to the control //
.registers of a PL011 device which is appropriately mapped //
} unsafe
.Write to the TX buffer //
;(self.base_address.write_volatile(byte
{

.Wait until the UART is no longer busy //
{} while self.read_flag_register() & FR_BUSY != 0
{

} fn read_flag_register(&self) -> u8
SAFETY: We know that the base address points to the control //
.registers of a PL011 device which is appropriately mapped //
{ ()unsafe { self.base_address.add(FLAG_REGISTER_OFFSET).read_volatile
```

```
  {
    {
```

- □□□□□ □□□□□ □□□□□□□ □□ □□□□□ □□ □□□□ unsafe □□ □□□□□□ `Uart::new` □□ □□□□□□ □□□□□□ □□□□□ □□ □□□□ □□□□□□ `Uart::new` □□□□□□□ □□□□□ □□ □□□□□□ □□ □□ □□□□ □□□□ □□□□□□□ □□□□ .□□□□□□ □□□□□ UART □□ □□□□□ □□□□□□□ □□ □□□□□□ □□ □□□□ □□□□□) □□□□ □□□□ □□□□□□□□□ □□ □□□□□□ □□□□□□□□ □□□□□□□ □□□□□□ □□ □(□□□□□□ □□ □□ □□□□□ □□□□□ □□□□□□□ □□□□ □□□□□□ □□□□ □□□□ □ □□□□□ □□□□□ .□□□□□ □□□□ □□ □□□□□ □□□□□□□□□□□ □□□□□□□□□ □□□□□ □□□□ □□□□□□□□□□ □□□□□□ □□ `write_byte`
- □□□□ □□□□□□ □□□□□ □□ `new` □□□□□) □□□□□ □□□□□□ □□□□□□ □□□□□ □□ □□ □□□□ □□□□ □□□□□□□□□□□□ □□ □□ □□□□□ □□□□□ □□□□□□ □□□□□□□ □□□□□□ □□ □□ □□□□□□□□□ □□□□ □(□□□□□ □□□□□□ □□ `write_byte` .□□□ □□□□□□□□□ safety □□ □□□□□□ □□□□□ □□ □□□□□ □□□□□□ □□□□ □□ `write_byte` □□ □□□□□□
- This is a common pattern for writing safe wrappers of unsafe code: moving the burden .of proof for soundness from a large number of places to a smaller number of places

□□□□□□ □□□□**trait** 53.4.1

.□□□□ □□□□□□ □□□□□ □□□□ □□□□□ □□□□□□ □□□□ □□□□□□ .□□□□□□ □□□□□□□□□ □□ Debug □□□□□□ □□□

```
                      ;{use core::fmt::{self, Write

                              } impl Write for Uart
} fn write_str(&mut self, s: &str) -> fmt::Result
                        } ()for c in s.as_bytes
                         ;(self.write_byte(*c
                                              {
                                         (())Ok
                                                {
                                                  {

SAFETY: `Uart` just contains a pointer to device memory, which can be //
                                           .accessed from any context //
                                            {} unsafe impl Send for Uart
```

- `Uart` □□□□□ □□ `!writeln` □ `!write` □□□□□□□□□ □□ □□□□□□ □□□□□□ □□ □□`Write` □□□□□□□□□□ .□□□□□ □□□□□□□□ □□□□
- `src/bare-metal/aps/examples` □□□ □□ `make qemu_minimal` □□ QEMU □□ □□ □□□□□ .□□□□□ □□□□□

□□□□□ UART □□□□□□□ □□ 53.5

The PL011 actually has a bunch more registers, and adding offsets to construct pointers to access them is error-prone and hard to read. Plus, some of them are bit fields which would .be nice to access in a structured way

□□□□	□□□□□□□ □□□	□□□
0x00	DR	12
0x04	RSR	4
0x18	FR	9
0x20	ILPR	8
0x24	IBRD	16

[illegible]	[illegible]	[illegible]
0x28	FBRD	6
0x2c	LCR_H	8
0x30	CR	16
0x34	IFLS	6
0x38	IMSC	11
0x3c	RIS	11
0x40	MIS	11
0x44	ICR	11
0x48	DMACR	3

- [illegible] [illegible] [illegible] ID register [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

53.5.1 [illegible] [illegible] (Bitflags)

[illegible] crate [illegible] `bitflags` [illegible] [illegible] [illegible] bitflags [illegible] [illegible].

```rust
use bitflags::bitflags;

bitflags! {
    /// Flags from the UART flag register.
    struct Flags: u16 {
        /// Clear to send.
        const CTS = 1 << 0;
        /// Data set ready.
        const DSR = 1 << 1;
        /// Data carrier detect.
        const DCD = 1 << 2;
        /// UART busy transmitting data.
        const BUSY = 1 << 3;
        /// Receive FIFO is empty.
        const RXFE = 1 << 4;
        /// Transmit FIFO is full.
        const TXFF = 1 << 5;
        /// Receive FIFO is full.
        const RXFF = 1 << 6;
        /// Transmit FIFO is empty.
        const TXFE = 1 << 7;
        /// Ring indicator.
        const RI = 1 << 8;
    }
}
```

- [illegible] `bitflags!` [illegible] [illegible] [illegible] [illegible] [illegible] `(Flags(u16` [illegible] [illegible] [illegible] [illegible] [illegible]- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] flag [illegible] [illegible] [illegible].

53.5.2 [illegible] [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] memory layout [illegible] [illegible] UART [illegible] [illegible].

```rust
struct Registers {
    dr: u16,
    _reserved0: [u8; 2],
    rsr: ReceiveStatus,
    _reserved1: [u8; 19],
    fr: Flags,
    _reserved2: [u8; 6],
    ilpr: u8,
    _reserved3: [u8; 3],
    ibrd: u16,
    _reserved4: [u8; 2],
    fbrd: u8,
    _reserved5: [u8; 3],
    lcr_h: u8,
    _reserved6: [u8; 3],
    cr: u16,
    _reserved7: [u8; 3],
    ifls: u8,
    _reserved8: [u8; 3],
    imsc: u16,
    _reserved9: [u8; 2],
    ris: u16,
    _reserved10: [u8; 2],
    mis: u16,
    _reserved11: [u8; 2],
    icr: u16,
    _reserved12: [u8; 2],
    dmacr: u8,
    _reserved13: [u8; 3],
}
```

- `#[repr(C)]` [illegible] C [illegible] Rust [illegible] ([illegible]) [illegible].

53.5.3 [illegible]

[illegible] `Registers` [illegible].

```rust
/// Driver for a PL011 UART.
pub struct Uart {
    registers: *mut Registers,
}

impl Uart {
    /// Constructs a new instance of the UART driver for a PL011 device at the
    /// given base address.
    ///
    /// # Safety
    ///
    /// The given base address must point to the 8 MMIO control registers of a
    /// PL011 device, which must be mapped into the address space of the process
    /// as device memory and not have any other aliases.
    pub unsafe fn new(base_address: *mut u32) -> Self {
        Self { registers: base_address as *mut Registers }
    }

    /// Writes a single byte to the UART.
    pub fn write_byte(&self, byte: u8) {
        // Wait until there is room in the TX buffer.
        while self.read_flag_register().contains(Flags::TXFF) {}

        // SAFETY: We know that self.registers points to the control registers
        // of a PL011 device which is appropriately mapped.
        unsafe {
            // Write to the TX buffer.
            addr_of_mut!((*self.registers).dr).write_volatile(byte.into());
        }

        // Wait until the UART is no longer busy.
        while self.read_flag_register().contains(Flags::BUSY) {}
    }

    /// Reads and returns a pending byte, or `None` if nothing has been
    /// received.
    pub fn read_byte(&self) -> Option<u8> {
        if self.read_flag_register().contains(Flags::RXFE) {
            None
        } else {
            // SAFETY: We know that self.registers points to the control
            // registers of a PL011 device which is appropriately mapped.
            let data = unsafe { addr_of!((*self.registers).dr).read_volatile() };
            // TODO: Check for error conditions in bits 8-11.
            Some(data as u8)
        }
    }

    fn read_flag_register(&self) -> Flags {
        // SAFETY: We know that self.registers points to the control registers
        // of a PL011 device which is appropriately mapped.
        unsafe { addr_of!((*self.registers).fr).read_volatile() }
    }
}
```

- [illegible] `addr_of!` / `addr_of_mut!` [illegible] pointer[illegible] reference [illegible].

53.5.4 [illegible]

[illegible] echo [illegible].

```
;mod exceptions
;mod pl011

;use crate::pl011::Uart
;use core::fmt::Write
;use core::panic::PanicInfo
;use log::error
;use smccc::psci::system_off
;use smccc::Hvc

.Base address of the primary PL011 UART ///
;_ const PL011_BASE_ADDRESS: *mut u32 = 0x900_0000 as

} (extern "C" fn main(x0: u64, x1: u64, x2: u64, x3: u64
SAFETY: `PL011_BASE_ADDRESS` is the base address of a PL011 device, and //
.nothing else accesses that address range //
;{ (let mut uart = unsafe { Uart::new(PL011_BASE_ADDRESS

;()writeln!(uart, "main({x0:#x}, {x1:#x}, {x2:#x}, {x3:#x})").unwrap

} loop
} ()if let Some(byte) = uart.read_byte
;(uart.write_byte(byte
} match byte
} <= 'b'\r
;('uart.write_byte(b'\n
{
,b'q' => break
{} <= _
{
{
{

;()writeln!(uart, "Bye!").unwrap
;()system_off::<Hvc>().unwrap
{
```

- [illegible] main [illegible] (inline assembly](../inline-assembly.md] [illegible] entry.S [illegible]
- .[illegible] src/bare-metal/aps/examples [illegible] make qemu [illegible] QEMU [illegible]

[illegible] 53.6

[illegible] crate log [illegible] logging [illegible] Log [illegible]

```
;use crate::pl011::Uart
;use core::fmt::Write
;{use log::{LevelFilter, Log, Metadata, Record, SetLoggerError
```

```
                                          ;use spin::mutex::SpinMutex

                  ;{ (static LOGGER: Logger = Logger { uart: SpinMutex::new(None

                                                              } struct Logger
                                      ,<<uart: SpinMutex<Option<Uart
                                                                            {

                                                          } impl Log for Logger
                        } fn enabled(&self, _metadata: &Metadata) -> bool
                                                                    true
                                                                         {

                                     } (fn log(&self, record: &Record
                                                                  )!writeln
                             ,()self.uart.lock().as_mut().unwrap
                                                         ,"{} [{}]"
                                                  ,()record.level
                                                    ()record.args
                                                                   (
                                                          ;()unwrap.
                                                                         {

                                                       {} (fn flush(&self
                                                                            {

                                                       .Initialises UART logger ///
} <pub fn init(uart: Uart, max_level: LevelFilter) -> Result<(), SetLoggerError
                                     ;(LOGGER.uart.lock().replace(uart

                                            ;?(log::set_logger(&LOGGER
                                      ;(log::set_max_level(max_level
                                                                 (())Ok
                                                                            {
```

- [illegible] set_logger [illegible] LOGGER [illegible] log[illegible]
  .[illegible]

[illegible] 53.6.1

.[illegible]

```
                                                         ;mod exceptions
                                                             ;mod logger
                                                              ;mod pl011

                                                  ;use crate::pl011::Uart
                                              ;use core::panic::PanicInfo
                                       ;{use log::{error, info, LevelFilter
                                           ;use smccc::psci::system_off
                                                        ;use smccc::Hvc
```

```
                                   .Base address of the primary PL011 UART ///
                          ;_ const PL011_BASE_ADDRESS: *mut u32 = 0x900_0000 as

                        } (extern "C" fn main(x0: u64, x1: u64, x2: u64, x3: u64
SAFETY: `PL011_BASE_ADDRESS` is the base address of a PL011 device, and //
                               .nothing else accesses that address range //
                        ;{ (let uart = unsafe { Uart::new(PL011_BASE_ADDRESS
                           ;()logger::init(uart, LevelFilter::Trace).unwrap

                          ;("({info!("main({x0:#x}, {x1:#x}, {x2:#x}, {x3:#x

                                                    ;(assert_eq!(x1, 42

                                          ;()system_off::<Hvc>().unwrap
                                                                          {

                                          } ! <- (fn panic(info: &PanicInfo
                                                       ;("{error!("{info
                                          ;()system_off::<Hvc>().unwrap
                                                                {} loop
                                                                          {
```

- .[illegible] panic [illegible] panic handler [illegible]
- src/bare-metal/aps/examples [illegible] make qemu_logger [illegible] QEMU [illegible] .[illegible]

[illegible] 53.7

synchronous[illegible] IRQ[illegible]) [illegible] AArch64 Ecurrent EL with SP0, current EL with SPx, lower EL using AArch64,) [illegible] (FIQ[illegible] SError [illegible] .[illegible] (lower EL using AArch32 :[illegible] stack [illegible] Rust [illegible] (volatile) [illegible]

```
                                                        ;use log::error
                                              ;use smccc::psci::system_off
                                                        ;use smccc::Hvc

                   } (extern "C" fn sync_exception_current(_elr: u64, _spsr: u64
                                          ;("error!("sync_exception_current
                                             ;()system_off::<Hvc>().unwrap
                                                                          {

                            } (extern "C" fn irq_current(_elr: u64, _spsr: u64
                                                   ;("error!("irq_current
                                             ;()system_off::<Hvc>().unwrap
                                                                          {

                            } (extern "C" fn fiq_current(_elr: u64, _spsr: u64
                                                   ;("error!("fiq_current
                                             ;()system_off::<Hvc>().unwrap
```

```
}

extern "C" fn serr_current(_elr: u64, _spsr: u64) {
error!("serr_current");
system_off::<Hvc>().unwrap();
}

extern "C" fn sync_lower(_elr: u64, _spsr: u64) {
error!("sync_lower");
system_off::<Hvc>().unwrap();
}

extern "C" fn irq_lower(_elr: u64, _spsr: u64) {
error!("irq_lower");
system_off::<Hvc>().unwrap();
}

extern "C" fn fiq_lower(_elr: u64, _spsr: u64) {
error!("fiq_lower");
system_off::<Hvc>().unwrap();
}

extern "C" fn serr_lower(_elr: u64, _spsr: u64) {
error!("serr_lower");
system_off::<Hvc>().unwrap();
}
```

- EL [illegible] EL1 [illegible].
- [illegible] SP0 [illegible] SPx [illegible] EL [illegible] AArch32 [illegible] AArch64 [illegible] EL [illegible].
- [illegible] exception [illegible] log [illegible].
- We can think of exception handlers and our main execution context more or less like different threads. Send and Sync will control what we can share between them, just like with threads. For example, if we want to share some value between exception handlers and the rest of the program, and it's Send but not Sync, then we'll need to wrap it in something like a `Mutex` and put it in a static.

53.8 [illegible]

- oreboot
  - "coreboot without the C"
  - [illegible] aarch64 [illegible] x86 [illegible] RISC-V.
  - [illegible] LinuxBoot [illegible].
- Rust RaspberryPi OS tutorial
  - [illegible] UART [illegible] bootloader [illegible] JTAG [illegible] exception [illegible] exception [illegible] page table [illegible]
  - [illegible] Rust [illegible] production [illegible].
- `cargo-call-stack`

  - [illegible] stack.

- [illegible] RaspberryPi، [illegible] Rust [illegible] MMU [illegible]. [illegible] memory [illegible] (stack [illegible]). [illegible]:
  - Without the MMU and cache, unaligned accesses will fault. It builds with `aarch64-unknown-none` which sets `+strict-align` to prevent the compiler generating unaligned accesses so it should be alright, but this is not necessarily the case in general.
  - If it were running in a VM, this can lead to cache coherency issues. The problem is that the VM is accessing memory directly with the cache disabled, while the host has cacheable aliases to the same memory. Even if the host doesn't explicitly access the memory, speculative accesses can lead to cache fills, and then changes from one or the other will get lost. Again this is alright in this particular case (running directly on the hardware with no hypervisor), but isn't a good pattern in general.

54 [illegible]

[illegible] (crates) [illegible]

[illegible] bare-metal [illegible] crate [illegible]
.[illegible]

zerocopy 54.1

[illegible] (Fuchsia [illegible]) crate zerocopy [illegible]
.[illegible]

```
;use zerocopy::AsBytes

} enum RequestType
,In = 0
,Out = 1
,Flush = 4
{

} struct VirtioBlockRequest
,request_type: RequestType
,reserved: u32
,sector: u64
{

} ()fn main
} let request = VirtioBlockRequest
,request_type: RequestType::Flush
,sector: 42
()Default::default..
;{

)!assert_eq
,()request.as_bytes
[4, 0, 0, 0, 0, 0, 0, 0, 42, 0, 0, 0, 0, 0, 0, 0]&
;(
{
```

[illegible] MMIO [illegible] ([illegible] volatile [illegible]) [illegible] DMA [illegible]. [illegible]. [illegible].

- FromBytes [illegible]-[illegible].
- [illegible] FromBytes [illegible] RequestType [illegible] u32 [illegible].
- `zerocopy::byteorder` [illegible] byte-order [illegible].
- [illegible] `cargo run` [illegible] `/src/bare-metal/useful-crates/zerocopy-example` [illegible]. ([illegible] crate [illegible] Playground [illegible].)

54.2 aarch64-paging

[illegible] aarch64-paging [illegible] crate [illegible] page table [illegible] AArch64 [illegible].

```rust
use aarch64_paging::{
    idmap::IdMap,
    paging::{Attributes, MemoryRegion},
};

const ASID: usize = 1;
const ROOT_LEVEL: usize = 1;

// Create a new page table with identity mapping.
let mut idmap = IdMap::new(ASID, ROOT_LEVEL);
// Map a 2 MiB region of memory as read-only.
idmap.map_range(
    &MemoryRegion::new(0x80200000, 0x80400000),
    Attributes::NORMAL | Attributes::NON_GLOBAL | Attributes::READ_ONLY,
).unwrap();
// Set `TTBR0_EL1` to activate the page table.
idmap.activate();
```

- [illegible] EL1 [illegible].
- [illegible] Android [illegible] ndroid/platform/superproject/+/master:packages/modules/Virtualization/pvmfw] [illegible].
- [illegible] QEMU [illegible].

54.3 buddy_system_allocator

'buddy_system_allocator' [illegible] third-party crate [illegible] buddy system allocator [illegible]. [illegible] 'LockedHeap' [illegible] [GlobalAlloc](https://doc.rust-lang.org/core/alloc/trait.GlobalAlloc.html) [illegible]. [illegible] crate [illegible] alloc [illegible] ([illegible]) [illegible]

[illegible]. [illegible] MMIO [illegible] PCI BAR [illegible]:

```
;use buddy_system_allocator::FrameAllocator
;use core::alloc::Layout

} ()fn main
    ;()let mut allocator = FrameAllocator::<32>::new
    ;(allocator.add_frame(0x200_0000, 0x400_0000

    ;()let layout = Layout::from_size_align(0x100, 0x100).unwrap
    let bar = allocator
        (alloc_aligned(layout.
        ;("expect("Failed to allocate 0x100 byte MMIO region.
    ;(println!("Allocated 0x100 byte MMIO region at {:#x}", bar
{
```

- PCI BAR [illegible].
- [illegible] cargo run [illegible] /src/bare-metal/useful-crates/allocator-example [illegible]. ([illegible] crate [illegible] Playground [illegible].)

54.4 tinyvec

[illegible] Vec [illegible] heap allocation [illegible] [tinyvec](https://crates.io/crates/tinyvec) [illegible]: [illegible] allocate [illegible] stack [illegible] panic [illegible].

```
;{use tinyvec::{array_vec, ArrayVec

} ()fn main
    ;(let mut numbers: ArrayVec<[u32; 5]> = array_vec!(42, 66
    ;("{?:println!("{numbers
    ;(numbers.push(7
    ;("{?:println!("{numbers
    ;(numbers.remove(1
    ;("{?:println!("{numbers
{
```

- `tinyvec` [illegible] `Default` [illegible].
- Rust Playground [illegible] `tinyvec` [illegible].

54.5 spin

`std::sync::Mutex` [illegible] `std::sync` [illegible] `core` [illegible] `alloc` [illegible]. [illegible] CPU [illegible].

[illegible] spin crate [illegible] spinlock [illegible].

```
                                      ;use spin::mutex::SpinMutex

                    ;(static counter: SpinMutex<u32> = SpinMutex::new(0

                                                            } ()fn main
                            ;(()println!("count: {}", counter.lock
                                            ;counter.lock() += 2*
                            ;(()println!("count: {}", counter.lock
                                                                    {
```

- [illegible] (deadlock)[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]handler [illegible] [illegible]
.[illegible]
- Once [illegible] `RwLock`, `Barrier` [illegible] .[illegible] ticket lock mutex [illegible] [illegible] [illegible] `spin`
.lazy [illegible] [illegible] [illegible] `Lazy` [illegible] .`std::sync` [illegible]
- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] 'crate 'once_cell [illegible]
.[illegible] `spin::once::Once` [illegible] [illegible] [illegible] [illegible] [illegible]
- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `spin` [illegible] Playground Rust
.[illegible]

55 □□□

□□□□□□□

□□□□ `rust_ffi_static` Soong □□ □□ □□□□ □AOSP □□ bare-metal Rust binary □□ □□□□□□ □□□□□
□□□□□□ □□□□□ linker script □□ □□ `cc_binary` □□ □□ □□□ □□□□□□ □□□□□□□□□ □□□□ Rust □□ □□□□□
□□□□□□ raw binary □□ □□ ELF □□□□□□ □□□□□ `raw_binary` □□ □□ □□□□ □ □□□□□ □□□□□□□□□ binary
.□□□□□ □□□□□□□□□ □□□□□

```
} rust_ffi_static
,"name: "libvmbase_example
,["defaults: ["vmbase_ffi_defaults
,"crate_name: "vmbase_example
,["srcs: ["src/main.rs
] :rustlibs
,"libvmbase"
,[
{

} cc_binary
,"name: "vmbase_example
,["defaults: ["vmbase_elf_defaults
] :srcs
,"idmap.S"
,[
] :static_libs
,"libvmbase_example"
,[
] :linker_scripts
,"image.ld"
,"vmbase_sections:"
,[
{

} raw_binary
,"name: "vmbase_example_bin
,"stem: "vmbase_example.bin
,"src: ":vmbase_example
,enabled: false
```

```
                        } :target
                 } :android_arm64
              ,enabled: true
                              ,{
                                  ,{
                                      {
```

vmbase 55.1

For VMs running under crosvm on aarch64, the vmbase library provides a linker script and .useful defaults for the build rules, along with an entry point, UART console logging and more

```
                                             ;{use vmbase::{main, println

                                                              ;(main!(main

       } (pub fn main(arg0: u64, arg1: u64, arg2: u64, arg3: u64
                                    ;("println!("Hello world
                                                                     {
```

- □□□□□□□□□ vmbase □□□□□□ □□□□□ □□ □□ □□□□□□ □□□□□ □□ □□□□ □□□□□ □□□□□□□!main □□□□□□ □□□□ .□□□
- main □□□□□□□ □□□□□ □□ □ □□□□□□ □□□□□□ □□ □□□□□□ □□□□□□ □□□□ □□□□□□ vmbase □□□□□□ □□□□□ .□□□□□□ □□□□□ VM □□□□□ □□□□□□ □□□□□ PSCI_SYSTEM_OFF □□ □function

56 [illegible]

[illegible]

[illegible] PL031 real-time clock [illegible].

[illegible] [illegible] [illegible].

56.1 RTC driver

The QEMU aarch64 virt machine has a PL031 real-time clock at 0x9010000. For this exercise, you should write a driver for it.

1. [illegible]. [illegible] crate `chrono` [illegible] date/time [illegible].
2. Use the match register and raw interrupt status to busy-wait until a given time, e.g. 3 seconds in the future. (Call `core::hint::spin_loop` inside the loop.)
3. [illegible]: [illegible] RTC [illegible]. [illegible] `arm-gic` crate [illegible] Arm Generic Interrupt Controller [illegible].
   - [illegible] RTC [illegible] `IntId::spi(2)` [illegible] GIC [illegible].
   - [illegible] (interrupt) [illegible] `arm_gic::wfi()` [illegible] Sleep [illegible].

[illegible] exercise template [illegible] `rtc` [illegible].

src/main.rs:

```rust
mod exceptions;
mod logger;
mod pl011;

use crate::pl011::Uart;
use arm_gic::gicv3::GicV3;
use core::panic::PanicInfo;
use log::{error, info, trace, LevelFilter};
use smccc::psci::system_off;
use smccc::Hvc;
```

```
                                              .Base addresses of the GICv3 ///
                          ;_ const GICD_BASE_ADDRESS: *mut u64 = 0x800_0000 as
                          ;_ const GICR_BASE_ADDRESS: *mut u64 = 0x80A_0000 as

                                   .Base address of the primary PL011 UART ///
                         ;_ const PL011_BASE_ADDRESS: *mut u32 = 0x900_0000 as

                      } (extern "C" fn main(x0: u64, x1: u64, x2: u64, x3: u64
SAFETY: `PL011_BASE_ADDRESS` is the base address of a PL011 device, and //
                              .nothing else accesses that address range //
                      ;{ (let uart = unsafe { Uart::new(PL011_BASE_ADDRESS
                          ;()logger::init(uart, LevelFilter::Trace).unwrap

                ;(info!("main({:#x}, {:#x}, {:#x}, {:#x})", x0, x1, x2, x3

       SAFETY: `GICD_BASE_ADDRESS` and `GICR_BASE_ADDRESS` are the base //
   addresses of a GICv3 distributor and redistributor respectively, and //
                            .nothing else accesses those address ranges //
;{ (let mut gic = unsafe { GicV3::new(GICD_BASE_ADDRESS, GICR_BASE_ADDRESS
                                                              ;()gic.setup

            .TODO: Create instance of RTC driver and print current time //

                                              .TODO: Wait for 3 seconds //

                                             ;()system_off::<Hvc>().unwrap
                                                                             {

                                             } ! <- (fn panic(info: &PanicInfo
                                                         ;("{error!("{info
                                             ;()system_off::<Hvc>().unwrap
                                                                   {} loop
                                                                             {
```

:([illegible]) *src/exceptions.rs*

```
                                               Copyright 2023 Google LLC //
                                                                         //
         ;("Licensed under the Apache License, Version 2.0 (the "License //
        .you may not use this file except in compliance with the License //
                                 You may obtain a copy of the License at //
                                                                         //
                         http://www.apache.org/licenses/LICENSE-2.0      //
                                                                         //
     Unless required by applicable law or agreed to in writing, software //
       ,distributed under the License is distributed on an "AS IS" BASIS //
.WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied //
     See the License for the specific language governing permissions and //
                                          .limitations under the License //

                                                    ;use arm_gic::gicv3::GicV3
```

```rust
use log::{error, info, trace};
use smccc::psci::system_off;
use smccc::Hvc;

extern "C" fn sync_exception_current(_elr: u64, _spsr: u64) {
    error!("sync_exception_current");
    system_off::<Hvc>().unwrap();
}

extern "C" fn irq_current(_elr: u64, _spsr: u64) {
    trace!("irq_current");
    let intid =
        GicV3::get_and_acknowledge_interrupt().expect("No pending interrupt");
    info!("IRQ {intid:?}");
}

extern "C" fn fiq_current(_elr: u64, _spsr: u64) {
    error!("fiq_current");
    system_off::<Hvc>().unwrap();
}

extern "C" fn serr_current(_elr: u64, _spsr: u64) {
    error!("serr_current");
    system_off::<Hvc>().unwrap();
}

extern "C" fn sync_lower(_elr: u64, _spsr: u64) {
    error!("sync_lower");
    system_off::<Hvc>().unwrap();
}

extern "C" fn irq_lower(_elr: u64, _spsr: u64) {
    error!("irq_lower");
    system_off::<Hvc>().unwrap();
}

extern "C" fn fiq_lower(_elr: u64, _spsr: u64) {
    error!("fiq_lower");
    system_off::<Hvc>().unwrap();
}

extern "C" fn serr_lower(_elr: u64, _spsr: u64) {
    error!("serr_lower");
    system_off::<Hvc>().unwrap();
}
```

src/logger.rs (□□□□□ □□□□ □□□ □□□□ □□ □□□□□ □□□□):

```rust
// Copyright 2023 Google LLC
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
```

```
                                     You may obtain a copy of the License at //
                                                                             //
                             http://www.apache.org/licenses/LICENSE-2.0      //
                                                                             //
         Unless required by applicable law or agreed to in writing, software //
           ,distributed under the License is distributed on an "AS IS" BASIS //
    .WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied //
         See the License for the specific language governing permissions and //
                                              .limitations under the License //

                                                                ANCHOR: main //
                                                        ;use crate::pl011::Uart
                                                          ;use core::fmt::Write
                 ;{use log::{LevelFilter, Log, Metadata, Record, SetLoggerError
                                                    ;use spin::mutex::SpinMutex

                 ;{ (static LOGGER: Logger = Logger { uart: SpinMutex::new(None

                                                                } struct Logger
                                             ,<<uart: SpinMutex<Option<Uart
                                                                              {

                                                          } impl Log for Logger
                          } fn enabled(&self, _metadata: &Metadata) -> bool
                                                                   true
                                                                          {

                                           } (fn log(&self, record: &Record
                                                              )!writeln
                                ,()self.uart.lock().as_mut().unwrap
                                                         ,"{} [{}]"
                                                    ,()record.level
                                                      ()record.args
                                                                      (
                                                             ;()unwrap.
                                                                          {

                                                         {} (fn flush(&self
                                                                              {

                                                   .Initialises UART logger ///
} <pub fn init(uart: Uart, max_level: LevelFilter) -> Result<(), SetLoggerError
                                          ;(LOGGER.uart.lock().replace(uart

                                                 ;?(log::set_logger(&LOGGER
                                             ;(log::set_max_level(max_level
                                                                     (())Ok
                                                                              {
```

:(□□□□□ □□□□□□ □□ □□□□□ □□□□ □□□□□ □□□□□□) *src/pl011.rs*

```
                                                   Copyright 2023 Google LLC //
```

```
                                                                         //
         ;("Licensed under the Apache License, Version 2.0 (the "License //
        .you may not use this file except in compliance with the License //
                                 You may obtain a copy of the License at //
                                                                         //
                         http://www.apache.org/licenses/LICENSE-2.0      //
                                                                         //
     Unless required by applicable law or agreed to in writing, software //
       ,distributed under the License is distributed on an "AS IS" BASIS //
.WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied //
     See the License for the specific language governing permissions and //
                                          .limitations under the License //


                                              ;{use core::fmt::{self, Write
                                     ;{use core::ptr::{addr_of, addr_of_mut

                                                           ANCHOR: Flags //
                                                    ;use bitflags::bitflags

                                                                } !bitflags
                                 .Flags from the UART flag register ///
                                                    } struct Flags: u16
                                                 .Clear to send ///
                                                ;const CTS = 1 << 0
                                                .Data set ready ///
                                                ;const DSR = 1 << 1
                                           .Data carrier detect ///
                                                ;const DCD = 1 << 2
                                   .UART busy transmitting data ///
                                               ;const BUSY = 1 << 3
                                         .Receive FIFO is empty ///
                                               ;const RXFE = 1 << 4
                                         .Transmit FIFO is full ///
                                               ;const TXFF = 1 << 5
                                          .Receive FIFO is full ///
                                               ;const RXFF = 1 << 6
                                        .Transmit FIFO is empty ///
                                               ;const TXFE = 1 << 7
                                                .Ring indicator ///
                                                 ;const RI = 1 << 8
                                                                      {
                                                                          {
                                                       ANCHOR_END: Flags //

                                                                } !bitflags
.Flags from the UART Receive Status Register / Error Clear Register ///
                                            } struct ReceiveStatus: u16
                                                 .Framing error ///
                                                 ;const FE = 1 << 0
                                                  .Parity error ///
```

```
                                                    ;const PE = 1 << 1
                                                      .Break error ///
                                                    ;const BE = 1 << 2
                                                    .Overrun error ///
                                                    ;const OE = 1 << 3
                                                                         {
                                                                             {

                                                          ANCHOR: Registers //
                                                            } struct Registers
                                                                  ,dr: u16
                                                      ,[reserved0: [u8; 2_
                                                       ,rsr: ReceiveStatus
                                                     ,[reserved1: [u8; 19_
                                                                ,fr: Flags
                                                      ,[reserved2: [u8; 6_
                                                                 ,ilpr: u8
                                                      ,[reserved3: [u8; 3_
                                                                ,ibrd: u16
                                                      ,[reserved4: [u8; 2_
                                                                 ,fbrd: u8
                                                      ,[reserved5: [u8; 3_
                                                                ,lcr_h: u8
                                                      ,[reserved6: [u8; 3_
                                                                  ,cr: u16
                                                      ,[reserved7: [u8; 3_
                                                                 ,ifls: u8
                                                      ,[reserved8: [u8; 3_
                                                                ,imsc: u16
                                                      ,[reserved9: [u8; 2_
                                                                 ,ris: u16
                                                     ,[reserved10: [u8; 2_
                                                                 ,mis: u16
                                                     ,[reserved11: [u8; 2_
                                                                 ,icr: u16
                                                     ,[reserved12: [u8; 2_
                                                                ,dmacr: u8
                                                     ,[reserved13: [u8; 3_
                                                                             {
                                                      ANCHOR_END: Registers //

                                                               ANCHOR: Uart //
                                                  .Driver for a PL011 UART ///
                                                             } pub struct Uart
                                                ,registers: *mut Registers
                                                                             {

                                                                   } impl Uart
Constructs a new instance of the UART driver for a PL011 device at the ///
                                                   .given base address ///
                                                                       ///
```

```
    /// # Safety
    ///
    /// The given base address must point to the MMIO control registers of a
    /// PL011 device, which must be mapped into the address space of the process
    /// as device memory and not have any other aliases.
    pub unsafe fn new(base_address: *mut u32) -> Self {
        Self { registers: base_address as *mut Registers }
    }

    /// Writes a single byte to the UART.
    pub fn write_byte(&self, byte: u8) {
        // Wait until there is room in the TX buffer.
        while self.read_flag_register().contains(Flags::TXFF) {}

        // SAFETY: We know that self.registers points to the control registers
        // of a PL011 device which is appropriately mapped.
        unsafe {
            // Write to the TX buffer.
            addr_of_mut!((*self.registers).dr).write_volatile(byte.into());
        }

        // Wait until the UART is no longer busy.
        while self.read_flag_register().contains(Flags::BUSY) {}
    }

    /// Reads and returns a pending byte, or `None` if nothing has been
    /// received.
    pub fn read_byte(&self) -> Option<u8> {
        if self.read_flag_register().contains(Flags::RXFE) {
            None
        } else {
            // SAFETY: We know that self.registers points to the control
            // registers of a PL011 device which is appropriately mapped.
            let data = unsafe { addr_of!((*self.registers).dr).read_volatile() };
            // TODO: Check for error conditions in bits 8-11.
            Some(data as u8)
        }
    }

    fn read_flag_register(&self) -> Flags {
        // SAFETY: We know that self.registers points to the control registers
        // of a PL011 device which is appropriately mapped.
        unsafe { addr_of!((*self.registers).fr).read_volatile() }
    }
}
// ANCHOR_END: Uart

impl Write for Uart {
    fn write_str(&mut self, s: &str) -> fmt::Result {
        for c in s.as_bytes() {
            self.write_byte(*c);
```

```
                                                              {
                                                         (())Ok
                                                                  {
                                                                      {

      Safe because it just contains a pointer to device memory, which can be //
                                               .accessed from any context //
                                                 {} unsafe impl Send for Uart
```

:([illegible]) *Cargo.toml*

```
                                                              [workspace]

                                                                [package]
                                                             "name = "rtc
                                                        "version = "0.1.0
                                                         "edition = "2021
                                                          publish = false

                                                           [dependencies]
                                                        "arm-gic = "0.1.1
                                                       "bitflags = "2.6.0
               { chrono = { version = "0.4.38", default-features = false
                                                          "log = "0.4.22
                                                         "smccc = "0.1.1
                                                          "spin = "0.9.8

                                                     [build-dependencies]
                                                           "cc = "1.1.15
```

:([illegible]) *build.rs*

```
                                                   Copyright 2023 Google LLC //
                                                                            //
          ;("Licensed under the Apache License, Version 2.0 (the "License //
         .you may not use this file except in compliance with the License //
                                         You may obtain a copy of the License at //
                                                                            //
                       http://www.apache.org/licenses/LICENSE-2.0        //
                                                                            //
      Unless required by applicable law or agreed to in writing, software //
       ,distributed under the License is distributed on an "AS IS" BASIS //
  .WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied //
      See the License for the specific language governing permissions and //
                                         .limitations under the License //

                                                             ;use cc::Build
                                                               ;use std::env

                                                                 } ()fn main
                  ;("env::set_var("CROSS_COMPILE", "aarch64-linux-gnu
                   ;("env::set_var("CROSS_COMPILE", "aarch64-none-elf
()Build::new
("file("entry.S.
("file("exceptions.S.
("file("idmap.S.
("compile("empty.
{
```

:([illegible]) *entry.S*

```
*/
Copyright 2023 Google LLC *
*
;("Licensed under the Apache License, Version 2.0 (the "License *
.you may not use this file except in compliance with the License *
You may obtain a copy of the License at *
*
https://www.apache.org/licenses/LICENSE-2.0     *
*
Unless required by applicable law or agreed to in writing, software *
,distributed under the License is distributed on an "AS IS" BASIS *
.WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied *
See the License for the specific language governing permissions and *
.limitations under the License *
/*

macro adr_l, reg:req, sym:req.
adrp \reg, \sym
add \reg, \reg, :lo12:\sym
endm.

macro mov_i, reg:req, imm:req.
movz \reg, :abs_g3:\imm
movk \reg, :abs_g2_nc:\imm
movk \reg, :abs_g1_nc:\imm
movk \reg, :abs_g0_nc:\imm
endm.

set .L_MAIR_DEV_nGnRE, 0x04.
set .L_MAIR_MEM_WBWA,  0xff.
(set .Lmairval, .L_MAIR_DEV_nGnRE | (.L_MAIR_MEM_WBWA << 8.

/* .KiB granule size for TTBR0_EL1 4 */
set .L_TCR_TG0_4KB, 0x0 << 14.
/* .KiB granule size for TTBR1_EL1 4 */
set .L_TCR_TG1_4KB, 0x2 << 30.
Disable translation table walk for TTBR1_EL1, generating a translation fault instead */
set .L_TCR_EPD1, 0x1 << 23.
/* .Translation table walks for TTBR0_EL1 are inner sharable */
set .L_TCR_SH_INNER, 0x3 << 12.
*/
anslation table walks for TTBR0_EL1 are outer write-back read-allocate write-allocate *
.cacheable *
```

```
                                                                                     /*
                                                          set .L_TCR_RGN_OWB, 0x1 << 10.
                                                                                      */
anslation table walks for TTBR0_EL1 are inner write-back read-allocate write-allocate *
                                                                           .cacheable *
                                                                                     /*
                                                           set .L_TCR_RGN_IWB, 0x1 << 8.
                               /* .(Size offset for TTBR0_EL1 is 2**39 bytes (512 GiB */
                                                           set .L_TCR_T0SZ_512, 64 - 39.
          set .Ltcrval, .L_TCR_TG0_4KB | .L_TCR_TG1_4KB | .L_TCR_EPD1 | .L_TCR_RGN_OWB.
           set .Ltcrval, .Ltcrval | .L_TCR_RGN_IWB | .L_TCR_SH_INNER | .L_TCR_T0SZ_512.

                            /* .Stage 1 instruction access cacheability is unaffected */
                                                          set .L_SCTLR_ELx_I, 0x1 << 12.
                    /* .SP alignment fault if SP is not aligned to a 16 byte boundary */
                                                          set .L_SCTLR_ELx_SA, 0x1 << 3.
                                   /* .Stage 1 data access cacheability is unaffected */
                                                           set .L_SCTLR_ELx_C, 0x1 << 2.
                                                  /* .EL0 and EL1 stage 1 MMU enabled */
                                                           set .L_SCTLR_ELx_M, 0x1 << 0.
               /* .Privileged Access Never is unchanged on taking an exception to EL1 */
                                                       set .L_SCTLR_EL1_SPAN, 0x1 << 23.
                               /* .SETEND instruction disabled at EL0 in aarch32 mode */
                                                         set .L_SCTLR_EL1_SED, 0x1 << 8.
                      /* .Various IT instructions are disabled at EL0 in aarch32 mode */
                                                         set .L_SCTLR_EL1_ITD, 0x1 << 7.
t .L_SCTLR_EL1_RES1, (0x1 << 11) | (0x1 << 20) | (0x1 << 22) | (0x1 << 28) | (0x1 << 29.
L_SCTLR_ELx_M | .L_SCTLR_ELx_C | .L_SCTLR_ELx_SA | .L_SCTLR_EL1_ITD | .L_SCTLR_EL1_SED.
   set .Lsctlrval, .Lsctlrval | .L_SCTLR_ELx_I | .L_SCTLR_EL1_SPAN | .L_SCTLR_EL1_RES1.

                                                                                     **/
neric entry point for an image. It carries out the operations required to prepare the *
nage to be run. Specifically, it zeroes the bss section using registers x25 and above *
e stack, enables floating point, and sets up the exception vector. It preserves x0-x3 *
                      .for the Rust entry point, as these may contain boot parameters *
                                                                                     /*
                                                              "section .init.entry, "ax.
                                                                           global entry.
                                                                                  :entry
oad and apply the memory management configuration, ready to enable MMU and caches */
                                                                     adrp x30, idmap
                                                                  msr ttbr0_el1, x30

                                                                mov_i x30, .Lmairval
                                                                   msr mair_el1, x30

                                                                 mov_i x30, .Ltcrval
                                 /* .Copy the supported PA range into TCR_EL1.IPS */
                                                           mrs x29, id_aa64mmfr0_el1
                                                               bfi x30, x29, #32, #4
```

```
                                                                   msr tcr_el1, x30

                                                              mov_i x30, .Lsctlrval

                                                                                 */
everything before this point has completed, then invalidate any potentially stale *
                                 .local TLB entries before they start being used *
                                                                                /*
                                                                                isb
                                                                       tlbi vmalle1
                                                                           ic iallu
                                                                            dsb nsh
                                                                                isb

                                                                                 */
gure sctlr_el1 to enable MMU and cache and don't proceed until this has completed *
                                                                                /*
                                                                 msr sctlr_el1, x30
                                                                                isb

                               /* .Disable trapping floating point access in EL1 */
                                                                 mrs x30, cpacr_el1
                                                         (orr x30, x30, #(0x3 << 20
                                                                 msr cpacr_el1, x30
                                                                                isb

                                                    /* .Zero out the bss section */
                                                               adr_l x29, bss_begin
                                                                 adr_l x30, bss_end
                                                                       cmp x29, x30  :0
                                                                            b.hs 1f
                                                           stp xzr, xzr, [x29], #16
                                                                               b 0b

                                                           /* .Prepare the stack */  :1
                                                          adr_l x30, boot_stack_end
                                                                        mov sp, x30

                                                     /* .Set up exception vector */
                                                          adr x30, vector_table_el1
                                                                  msr vbar_el1, x30

                                                         /* .Call into Rust code */
                                                                            bl main

                                         /* .Loop forever waiting for interrupts */
                                                                                wfi  :2
                                                                               b 2b
```

:([illegible]) *exceptions.S*

```
*/
```

```
Copyright 2023 Google LLC *
*
;("Licensed under the Apache License, Version 2.0 (the "License *
.you may not use this file except in compliance with the License *
You may obtain a copy of the License at *
*
https://www.apache.org/licenses/LICENSE-2.0     *
*
Unless required by applicable law or agreed to in writing, software *
,distributed under the License is distributed on an "AS IS" BASIS *
.WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied *
See the License for the specific language governing permissions and *
.limitations under the License *
/*

**/
Saves the volatile registers onto the stack. This currently takes 14 *
instructions, so it can be used in exception handlers with 18 instructions *
.left *
*
,On return, x0 and x1 are initialised to elr_el2 and spsr_el2 respectively *
.which can be used as the first and second arguments of a subsequent call *
/*
macro save_volatile_to_stack.
/* .Reserve stack space and save registers x0-x18, x29 & x30 */
![(stp x0, x1, [sp, #-(8 * 24
[stp x2, x3, [sp, #8 * 2
[stp x4, x5, [sp, #8 * 4
[stp x6, x7, [sp, #8 * 6
[stp x8, x9, [sp, #8 * 8
[stp x10, x11, [sp, #8 * 10
[stp x12, x13, [sp, #8 * 12
[stp x14, x15, [sp, #8 * 14
[stp x16, x17, [sp, #8 * 16
[str x18, [sp, #8 * 18
[stp x29, x30, [sp, #8 * 20

*/
Save elr_el1 & spsr_el1. This such that we can take nested exception *
.and still be able to unwind *
/*
mrs x0, elr_el1
mrs x1, spsr_el1
[stp x0, x1, [sp, #8 * 22
endm.

**/
Restores the volatile registers from the stack. This currently takes 14 *
instructions, so it can be used in exception handlers while still leaving 18 *
instructions left; if paired with save_volatile_to_stack, there are 4 *
.instructions to spare *
```

```
/*
macro restore_volatile_from_stack.
/* .Restore registers x2-x18, x29 & x30 */
[ldp x2, x3, [sp, #8 * 2
[ldp x4, x5, [sp, #8 * 4
[ldp x6, x7, [sp, #8 * 6
[ldp x8, x9, [sp, #8 * 8
[ldp x10, x11, [sp, #8 * 10
[ldp x12, x13, [sp, #8 * 12
[ldp x14, x15, [sp, #8 * 14
[ldp x16, x17, [sp, #8 * 16
[ldr x18, [sp, #8 * 18
[ldp x29, x30, [sp, #8 * 20

/* .Restore registers elr_el1 & spsr_el1, using x0 & x1 as scratch */
[ldp x0, x1, [sp, #8 * 22
msr elr_el1, x0
msr spsr_el1, x1

/* .Restore x0 & x1, and release stack space */
ldp x0, x1, [sp], #8 * 24
endm.

**/
This is a generic handler for exceptions taken at the current EL while using *
SP0. It behaves similarly to the SPx case by first switching to SPx, doing *
.the work, then switching back to SP0 before returning *
*
Switching to SPx and calling the Rust handler takes 16 instructions. To *
restore and return we need an additional 16 instructions, so we can implement *
.the whole handler within the allotted 32 instructions *
/*
macro current_exception_sp0 handler:req.
msr spsel, #1
save_volatile_to_stack
bl \handler
restore_volatile_from_stack
msr spsel, #0
eret
endm.

**/
This is a generic handler for exceptions taken at the current EL while using *
SPx. It saves volatile registers, calls the Rust handler, restores volatile *
.registers, then returns *
*
This also works for exceptions taken from EL0, if we don't care about *
.non-volatile registers *
*
Saving state and jumping to the Rust handler takes 15 instructions, and *
restoring and returning also takes 15 instructions, so we can fit the whole *
```

```
.handler in 30 instructions, under the limit of 32 *
/*
macro current_exception_spx handler:req.
save_volatile_to_stack
bl \handler
restore_volatile_from_stack
eret
endm.

"section .text.vector_table_el1, "ax.
global vector_table_el1.
balign 0x800.
:vector_table_el1
:sync_cur_sp0
current_exception_sp0 sync_exception_current

balign 0x80.
:irq_cur_sp0
current_exception_sp0 irq_current

balign 0x80.
:fiq_cur_sp0
current_exception_sp0 fiq_current

balign 0x80.
:serr_cur_sp0
current_exception_sp0 serr_current

balign 0x80.
:sync_cur_spx
current_exception_spx sync_exception_current

balign 0x80.
:irq_cur_spx
current_exception_spx irq_current

balign 0x80.
:fiq_cur_spx
current_exception_spx fiq_current

balign 0x80.
:serr_cur_spx
current_exception_spx serr_current

balign 0x80.
:sync_lower_64
current_exception_spx sync_lower

balign 0x80.
:irq_lower_64
current_exception_spx irq_lower
```

```
balign 0x80.
:fiq_lower_64
    current_exception_spx fiq_lower

balign 0x80.
:serr_lower_64
    current_exception_spx serr_lower

balign 0x80.
:sync_lower_32
    current_exception_spx sync_lower

balign 0x80.
:irq_lower_32
    current_exception_spx irq_lower

balign 0x80.
:fiq_lower_32
    current_exception_spx fiq_lower

balign 0x80.
:serr_lower_32
    current_exception_spx serr_lower
```

idmap.S ([illegible]):

```
*/
Copyright 2023 Google LLC *
*
;("Licensed under the Apache License, Version 2.0 (the "License *
.you may not use this file except in compliance with the License *
You may obtain a copy of the License at *
*
https://www.apache.org/licenses/LICENSE-2.0     *
*
Unless required by applicable law or agreed to in writing, software *
,distributed under the License is distributed on an "AS IS" BASIS *
.WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied *
See the License for the specific language governing permissions and *
.limitations under the License *
/*

set .L_TT_TYPE_BLOCK, 0x1.
set .L_TT_TYPE_PAGE,  0x3.
set .L_TT_TYPE_TABLE, 0x3.

/* .Access flag */
set .L_TT_AF, 0x1 << 10.
/* .Not global */
set .L_TT_NG, 0x1 << 11.
set .L_TT_XN, 0x3 << 53.
```

```
                        (set .L_TT_MT_DEV, 0x0 << 2         // MAIR #0 (DEV_nGnRE.
set .L_TT_MT_MEM, (0x1 << 2) | (0x3 << 8)  // MAIR #1 (MEM_WBWA), inner shareable.

          set .L_BLOCK_DEV, .L_TT_TYPE_BLOCK | .L_TT_MT_DEV | .L_TT_AF | .L_TT_XN.
          set .L_BLOCK_MEM, .L_TT_TYPE_BLOCK | .L_TT_MT_MEM | .L_TT_AF | .L_TT_NG.

                                          section ".rodata.idmap", "a", %progbits.
                                                                     global idmap.
                                                                         align 12.
                                                                            :idmap
                                                                 /* level 1 */
           quad       .L_BLOCK_DEV | 0x0          // 1 GiB of device mappings.
                      quad       .L_BLOCK_MEM | 0x40000000   // 1 GiB of DRAM.
               fill       254, 8, 0x0         // 254 GiB of unmapped VA space.
           quad       .L_BLOCK_DEV | 0x4000000000 // 1 GiB of device mappings.
              fill       255, 8, 0x0         // 255 GiB of remaining VA space.
```

:(□□□□□ □□□□□□ □□ □□□□□ □□□□ □□□□□ □□□□□□) *image.ld*

```
                                                                                */
                                                      Copyright 2023 Google LLC *
                                                                                *
                ;("Licensed under the Apache License, Version 2.0 (the "License *
               .you may not use this file except in compliance with the License *
                                        You may obtain a copy of the License at *
                                                                                *
                                https://www.apache.org/licenses/LICENSE-2.0     *
                                                                                *
            Unless required by applicable law or agreed to in writing, software *
              ,distributed under the License is distributed on an "AS IS" BASIS *
       .WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied *
            See the License for the specific language governing permissions and *
                                                 .limitations under the License *
                                                                               /*

                                                                                */
     Code will start running at this symbol which is placed at the start of the *
                                                                       .image *
                                                                               /*
                                                                      (ENTRY(entry

                                                                            MEMORY
                                                                                 }
                                      image : ORIGIN = 0x40080000, LENGTH = 2M
                                                                                 {

                                                                          SECTIONS
                                                                                 }
                                                                            */
                                                 .Collect together the code *
                                                                           /*
```

```
                                                  } (init : ALIGN(4096.
                                                    ;. = text_begin
                                                     (init.entry.)*
                                                         (*.init.)*
                                                               image< {
                                                              } : text.
                                                         (*.text.)*
                                                               image< {
                                                          ;. = text_end

                                                                     */
                                    .Collect together read-only data *
                                                                    /*
                                                } (rodata : ALIGN(4096.
                                                  ;. = rodata_begin
                                                       (*.rodata.)*
                                                               image< {
                                                               } : got.
                                                            (got.)*
                                                               image< {
                                                        ;. = rodata_end

                                                                     */
Collect together the read-write data including .bss at the end which *
                                   .will be zero'd by the entry code *
                                                                    /*
                                                  } (data : ALIGN(4096.
                                                    ;. = data_begin
                                                         (*.data.)*
                                                                 */
     The entry point code assumes that .data is a multiple of 32 *
                                                     .bytes long *
                                                                /*
                                                     ;(ALIGN(32 = .
                                                      ;. = data_end
                                                               image< {

 /* .Everything beyond this point will not be included in the binary */
                                                           ;. = bin_end

       /* .The entry point code assumes that .bss is 16-byte aligned */
                                                    }  (bss : ALIGN(16.
                                                     ;. = bss_begin
                                                          (*.bss.)*
                                                          (COMMON)*
                                                     ;(ALIGN(16 = .
                                                       ;. = bss_end
                                                               image< {

                                        } (stack (NOLOAD) : ALIGN(4096.
                                              ;. = boot_stack_begin
```

```
;4096 * 40 =+ .
;(ALIGN(4096 = .
;. = boot_stack_end
image< {

;(ALIGN(4K = .
;(. = PROVIDE(dma_region

*/
.Remove unused sections from the image *
/*
} : /DISCARD/
/* .The image loads itself so doesn't need these sections */
(gnu.hash.)*
(hash.)*
(interp.)*
(eh_frame_hdr.)*
(eh_frame.)*
(note.gnu.build-id.)*
{
{
```

:([illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]) *Makefile*

```
Copyright 2023 Google LLC #
#
;("Licensed under the Apache License, Version 2.0 (the "License #
.you may not use this file except in compliance with the License #
You may obtain a copy of the License at #
#
http://www.apache.org/licenses/LICENSE-2.0      #
#
Unless required by applicable law or agreed to in writing, software #
,distributed under the License is distributed on an "AS IS" BASIS #
.WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied #
See the License for the specific language governing permissions and #
.limitations under the License #

(UNAME := $(shell uname -s
(ifeq ($(UNAME),Linux
TARGET = aarch64-linux-gnu
else
TARGET = aarch64-none-elf
endif
OBJCOPY = $(TARGET)-objcopy

PHONY: build qemu_minimal qemu qemu_logger.

all: rtc.bin

:build
cargo build
rtc.bin: build
@$ OBJCOPY) -O binary target/aarch64-unknown-none/debug/rtc)$

qemu: rtc.bin
4 -machine virt,gic-version=3 -cpu max -serial mon:stdio -display none -kernel $< -s

:clean
cargo clean
rm -f *.bin
```

:(*cargo/config.toml* (you shouldn't need to change this.

```
[build]
"target = "aarch64-unknown-none
["rustflags = ["-C", "link-arg=-Timage.ld
```

.[illegible] [illegible] make qemu [illegible] QEMU [illegible] [illegible] [illegible]

Bare Metal Rust [illegible] 56.2

RTC driver

(back to exercise)

:*main.rs*

```
;mod exceptions
;mod logger
;mod pl011
;mod pl031

;use crate::pl031::Rtc
;{use arm_gic::gicv3::{IntId, Trigger
;{use arm_gic::{irq_enable, wfi
;{use chrono::{TimeZone, Utc
;use core::hint::spin_loop
;use crate::pl011::Uart
;use arm_gic::gicv3::GicV3
;use core::panic::PanicInfo
;{use log::{error, info, trace, LevelFilter
;use smccc::psci::system_off
;use smccc::Hvc

.Base addresses of the GICv3 ///
;_ const GICD_BASE_ADDRESS: *mut u64 = 0x800_0000 as
;_ const GICR_BASE_ADDRESS: *mut u64 = 0x80A_0000 as

.Base address of the primary PL011 UART ///
;_ const PL011_BASE_ADDRESS: *mut u32 = 0x900_0000 as
```

```
                                             .Base address of the PL031 RTC ///
                          ;_ const PL031_BASE_ADDRESS: *mut u32 = 0x901_0000 as
                                             .The IRQ used by the PL031 RTC ///
                                        ;(const PL031_IRQ: IntId = IntId::spi(2

                       } (extern "C" fn main(x0: u64, x1: u64, x2: u64, x3: u64
SAFETY: `PL011_BASE_ADDRESS` is the base address of a PL011 device, and //
                               .nothing else accesses that address range //
                       ;{ (let uart = unsafe { Uart::new(PL011_BASE_ADDRESS
                           ;()logger::init(uart, LevelFilter::Trace).unwrap

                 ;(info!("main({:#x}, {:#x}, {:#x}, {:#x})", x0, x1, x2, x3

        SAFETY: `GICD_BASE_ADDRESS` and `GICR_BASE_ADDRESS` are the base //
    addresses of a GICv3 distributor and redistributor respectively, and //
                             .nothing else accesses those address ranges //
;{ (let mut gic = unsafe { GicV3::new(GICD_BASE_ADDRESS, GICR_BASE_ADDRESS
                                                               ;()gic.setup

SAFETY: `PL031_BASE_ADDRESS` is the base address of a PL031 device, and //
                               .nothing else accesses that address range //
                     ;{ (let mut rtc = unsafe { Rtc::new(PL031_BASE_ADDRESS
                                                ;()let timestamp = rtc.read
                ;()let time = Utc.timestamp_opt(timestamp.into(), 0).unwrap
                                                      ;("{info!("RTC: {time

                                            ;(GicV3::set_priority_mask(0xff
                               ;(gic.set_interrupt_priority(PL031_IRQ, 0x80
                                ;(gic.set_trigger(PL031_IRQ, Trigger::Level
                                                              ;()irq_enable
                                     ;(gic.enable_interrupt(PL031_IRQ, true

                                 .Wait for 3 seconds, without interrupts //
                                                ;let target = timestamp + 3
                                                     ;(rtc.set_match(target
    ;(()info!("Waiting for {}", Utc.timestamp_opt(target.into(), 0).unwrap
                                                                    )!trace
                                     ,"{}=matched={}, interrupt_pending"
                                                         ,()rtc.matched
                                                ()rtc.interrupt_pending
                                                                         ;(
                                                     } ()while !rtc.matched
                                                           ;()spin_loop
                                                                          {
                                                                    )!trace
                                     ,"{}=matched={}, interrupt_pending"
                                                         ,()rtc.matched
                                                ()rtc.interrupt_pending
                                                                         ;(
                                                  ;("info!("Finished waiting
.Wait another 3 seconds for an interrupt //
;let target = timestamp + 6
;(()info!("Waiting for {}", Utc.timestamp_opt(target.into(), 0).unwrap
;(rtc.set_match(target
;()rtc.clear_interrupt
;(rtc.enable_interrupt(true
)!trace
,"{}=matched={}, interrupt_pending"
,()rtc.matched
()rtc.interrupt_pending
;(
} ()while !rtc.interrupt_pending
;()wfi
{
)!trace
,"{}=matched={}, interrupt_pending"
,()rtc.matched
()rtc.interrupt_pending
;(
;("info!("Finished waiting

;()system_off::<Hvc>().unwrap
{

} ! <- (fn panic(info: &PanicInfo
;("{error!("{info
;()system_off::<Hvc>().unwrap
{} loop
{
```

:pl031.rs

```
;{use core::ptr::{addr_of, addr_of_mut

} struct Registers
Data register ///
,dr: u32
Match register ///
,mr: u32
Load register ///
,lr: u32
Control register ///
,cr: u8
,[reserved0: [u8; 3_
Interrupt Mask Set or Clear register ///
,imsc: u8
,[reserved1: [u8; 3_
Raw Interrupt Status ///
,ris: u8
,[reserved2: [u8; 3_
Masked Interrupt Status ///
,mis: u8
```

```
    _reserved3: [u8; 3],
    /// Interrupt Clear Register
    icr: u8,
    _reserved4: [u8; 3],
}

/// Driver for a PL031 real-time clock.
pub struct Rtc {
    registers: *mut Registers,
}

impl Rtc {
    /// Constructs a new instance of the RTC driver for a PL031 device at the
    /// given base address.
    ///
    /// # Safety
    ///
    /// The given base address must point to the MMIO control registers of a
    /// PL031 device, which must be mapped into the address space of the process
    /// as device memory and not have any other aliases.
    pub unsafe fn new(base_address: *mut u32) -> Self {
        Self { registers: base_address as *mut Registers }
    }

    /// Reads the current RTC value.
    pub fn read(&self) -> u32 {
        // SAFETY: We know that self.registers points to the control registers
        // of a PL031 device which is appropriately mapped.
        unsafe { addr_of!((*self.registers).dr).read_volatile() }
    }

    /// Writes a match value. When the RTC value matches this then an interrupt
    /// will be generated (if it is enabled).
    pub fn set_match(&mut self, value: u32) {
        // SAFETY: We know that self.registers points to the control registers
        // of a PL031 device which is appropriately mapped.
        unsafe { addr_of_mut!((*self.registers).mr).write_volatile(value) }
    }

    /// Returns whether the match register matches the RTC value, whether or not
    /// the interrupt is enabled.
    pub fn matched(&self) -> bool {
        // SAFETY: We know that self.registers points to the control registers
        // of a PL031 device which is appropriately mapped.
        let ris = unsafe { addr_of!((*self.registers).ris).read_volatile() };
        (ris & 0x01) != 0
    }

    /// Returns whether there is currently an interrupt pending.
    ///
    /// This should be true if and only if `matched` returns true and the
```

```
                                                   .interrupt is masked ///
                                  } pub fn interrupt_pending(&self) -> bool
SAFETY: We know that self.registers points to the control registers //
                   .of a PL031 device which is appropriately mapped //
 ;{ ()let ris = unsafe { addr_of!((*self.registers).mis).read_volatile
                                                      ris & 0x01) != 0)
                                                                          {

                                     .Sets or clears the interrupt mask ///
                                                                        ///
  When the mask is true the interrupt is enabled; when it is false the ///
                                                 .interrupt is disabled ///
                           } (pub fn enable_interrupt(&mut self, mask: bool
                             ;{ let imsc = if mask { 0x01 } else { 0x00
SAFETY: We know that self.registers points to the control registers //
                   .of a PL031 device which is appropriately mapped //
  { (unsafe { addr_of_mut!((*self.registers).imsc).write_volatile(imsc
                                                                          {

                                    .Clears a pending interrupt, if any ///
                                        } (pub fn clear_interrupt(&mut self
SAFETY: We know that self.registers points to the control registers //
                   .of a PL031 device which is appropriately mapped //
   { (unsafe { addr_of_mut!((*self.registers).icr).write_volatile(0x01
                                                                          {
                                                                              {

       SAFETY: `Rtc` just contains a pointer to device memory, which can be //
                                                  .accessed from any context //
                                                    {} unsafe impl Send for Rtc
```

XIII □□□

□□□ :□□□□□□□

57 [illegible]

[illegible] Concurrency [illegible] Rust

[illegible] Rust [illegible] concurrency [illegible] thread [illegible] mutex [illegible] channel [illegible].

[illegible] Rust [illegible] concurrency [illegible] (*fearless concurrency*) [illegible] (runtime) [illegible].

[illegible]

[illegible]:

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
Send [illegible] Sync	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

- Rust [illegible]: thread [illegible].
- [illegible] concurrency [illegible].
- [illegible] concurrent [illegible] thread [illegible] ([illegible]) [illegible] multi-threading [illegible].

58 [illegible]

[illegible]

[illegible]:

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

58.1 [illegible]

thread [illegible] Rust [illegible] thread [illegible]:

```
use std::thread;
use std::time::Duration;

fn main() {
    thread::spawn(|| {
        for i in 0..10 {
            println!("[illegible] thread: {i}!");
            thread::sleep(Duration::from_millis(5));
        }
    });

    for i in 0..5 {
        println!("Main thread: {i}");
        thread::sleep(Duration::from_millis(5));
    }
}
```

- thread [illegible] `main` [illegible].
- Thread panic [illegible].
  - Panic [illegible] payload [illegible] «downcast_ref» [illegible].

- Rust thread API [illegible]. [illegible] C++ [illegible].
- [illegible].
  - [illegible] 5 [illegible] thread [illegible] spawned thread[illegible].
  - [illegible] spawned thread [illegible]!
  - [illegible] main [illegible] spawned thread[illegible].
    - [illegible] pthreads/C++ std::thread/boost::thread [illegible].
- [illegible] spawned thread [illegible]
- `thread::spawn` returns a `JoinHandle` [illegible].
  - `JoinHandle` [illegible] `.join()` [illegible].
- [illegible] «let handle = thread::spawn(...)» [illegible] «handle.join()» [illegible] thread [illegible].
- [illegible]
- [illegible]:
  - `thread::spawn`'s closure returns `T`
  - `JoinHandle` `.join()` returns `thread::Result<T>`
- [illegible] `Result`[illegible] «handle.join()» [illegible].
- [illegible]
  - [illegible] panic [illegible] thread. [illegible] `main` panic [illegible].
  - [illegible] panic payload. [illegible] `Any`.
- [illegible]! [illegible]
  - [illegible] reference [illegible] thread [illegible].
  - [illegible].
  - [illegible].
- [illegible] (borrow) [illegible]
  - Main [illegible] thread[illegible] return [illegible].
  - [illegible] stack [illegible] return [illegible] memory safety [illegible]!
  - [illegible].

58.2 [illegible]

thread[illegible] (borrow) [illegible]:

```
use std::thread;

fn foo() {
    let s = String::from("□□□□□");
    thread::spawn(|| {
        println!("Length: {}", s.len());
    });
}

fn main() {
    foo();
}
```

□□□□□□□□, □□□□□□□□□ □□□□□ □□□ □□□□ scoped thread □□□□□□□:

```
use std::thread;

fn main() {
    let s = String::from("□□□□□");

    thread::scope(|scope| {
        scope.spawn(|| {
            println!("Length: {}", s.len());
        });
    });
}
```

- □□□□□ □□□ □□□□ □□□ □□ □□□□□ □□□□□ «thread::scope» □□□□□ □□□□□□□ □□□□□□ □□□ thread□□ □□□□□□ □□□□□□□ □□□□□□□□□ □□□□□□□□□ □□□□□□□□ □□□□□ □□ □□□□□□□□□□□.
- □□□□□□□ □□□□□ □□□□□□□□ Rust □□□□□□ □□□□□□: □□□□ □□□□□□□□□ □□□□□□□ □□□□□□□□□□□(mutable) □□ □□□ thread □□□ □□□□□□□□ □□□□□□ (immutable) □□ □□ □□□□□□ thread □□□□ (borrow) □□□□□□□.

59 □□□

□□□□□□□

:□□□□□ □□□ .□□□□□ □□□ □□□□□□ □□ □□□□□ □□□□□ □□□ □□□

□□□□□□□	□□□□□ □□□
Receivers □ Senders	□□□□□□ □□
□□□□□□□□ □□□□□□□□□□	□□□□□□ □
□□□□□□ □□□□□□□□□□	□□□□□□ □□

Receivers □ Senders 59.1

□□□□□ □□ □□□ □□ □□□ .<Receiver<T □ <Sender<T :□□□□□□ □□□ □□ □□□□□□ Rust □□□□□□□□□ .□□□□□□□□ □□ (end-points) □□□□□□□ □□□□□ □□□ □□□ □□□ □□□□□□□□ □□□□□ □□ □□ channel

```
;use std::sync::mpsc

} ()fn main
    ;()let (tx, rx) = mpsc::channel

    ;()tx.send(10).unwrap
    ;()tx.send(20).unwrap

    ;(()rx.recv ,"{?:} :□□ □□□□□□")!println
    ;(()rx.recv ,"{?:} :□□ □□□□□□")!println

    ;()let tx2 = tx.clone
    ;()tx2.send(30).unwrap
    ;(()rx.recv ,"{?:} :□□ □□□□□□")!println
{
```

- `mpsc` stands for Multi-Producer, Single-Consumer. `Sender` and `SyncSender` implement .`Clone` (so you can make multiple producers) but `Receiver` does not
- `send()` and `recv()` return `Result`. If they return `Err`, it means the counterpart `Sender` .or `Receiver` is dropped and the channel is closed

59.2 [illegible] [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `mpsc::channel()` [illegible] [illegible] [illegible]:

```rust
use std::sync::mpsc;
use std::thread;
use std::time::Duration;

fn main() {
    let (tx, rx) = mpsc::channel();

    thread::spawn(move || {
        let thread_id = thread::current().id();
        for i in 0..10 {
            tx.send(format!("Message {i}")).unwrap();
            println!("{thread_id:?}: sent Message {i}");
        }
        println!("{thread_id:?}: [illegible] [illegible]");
    });
    thread::sleep(Duration::from_millis(100));

    for msg in rx.iter() {
        println!("Main: got {msg}");
    }
}
```

59.3 [illegible] [illegible]

[illegible] [illegible] (synchronous bounded) [illegible] [illegible] send [illegible] thread [illegible] [illegible] [illegible] [illegible]:

```rust
use std::sync::mpsc;
use std::thread;
use std::time::Duration;

fn main() {
    let (tx, rx) = mpsc::sync_channel(3);

    thread::spawn(move || {
        let thread_id = thread::current().id();
        for i in 0..10 {
            tx.send(format!("Message {i}")).unwrap();
            println!("{thread_id:?}: sent Message {i}");
        }
        println!("{thread_id:?}: [illegible] [illegible]");
    });
    thread::sleep(Duration::from_millis(100));

    for msg in rx.iter() {
        println!("Main: got {msg}");
    }
}
```

- [illegible] `send` [illegible] thread [illegible] thread [illegible].
- [illegible] `send` [illegible] error [illegible] (`Result` [illegible].
- [illegible] (bounded channel) [illegible] "[illegible]" [illegible] "rendezvous channel" [illegible] thread [illegible] `recv` [illegible].

60 □□□

Sync □ Send

:□□□□□ □□□□ □□□□ .□□□□ □□□ □□□□□ □□□□□□□ □□□ □□□□ □□□□

□□□□□□□□	□□□□□ □□□□
□□□□□□□ □□□□□□□□□□	□□□□□□ □
Send	□□□□□□ □
Sync	□□□□□□ □
□□□□□□□□	□□□□□□ □□□

□□□□□□□ □□□□□□□□□□ 60.1

□□ □□ □□□□□ □□□□□□□□ □□□□□□ □□ □□thread □□□□□□ □□ □□□□□□ □□□□□□□ □□ □□□□□□□ □□□□□□ Rust :□□□ trait

- □□□□ □□□ T □□□□□ □□□□□□ □□□□□ thread boundary □□□□□□□ □□□ T □□□□□□□□□ □□□ □□□□□□ □□□ :Send .□□□ Send
- T □□□□□ □□□ □□□□□□ □□□□□ thread boundary □□ □□□□□□ □□□ T& □□ □□□□□□□□□ □□□ □□□□□□ □□□ :Sync .□□□ Sync □□□□ □□□

□□□□□□ □□□□□□□□□□ .(unsafe-rust/unsafe-traits.md/../..) □□□□□□ [□□□□□□ □□□□□□□□□□] Sync □ Send Sync □ Send □□□□□□ □□□□□□ □□□□ □□□ □□□□□□ □□□ □□□□□□ □□□□□ □□□□ □□□□□□□□□ □□□□□ □□□ □□□□□ □□□□□□□ □□□ □□□ □□□□□□ □□□□□□□ □□□□□ □□□□□ □□□□□□ □□□□□ □□□□□ □□□ □□□ □□□□□ □□□□□□□ □□□ □□□□□□□ □□□□ .□□□□□□ .□□□□ □□□□□□ □□□ □□□□□□ □□□□□□

- thread- □□□□□□ □□□□□ □□□ □□□□□ □□□□ □□□ □□□□□□□□□□□□ □□□□□□ □□□ □□□ □□trait □□□□ □□□□□□□ .□□□□□ □□□ □□□□□ safety
- .□□□□ □□□□□□□□□ □□□□□ trait □□□□□□ □□□ generic □□□□□□□□□□□ □□□ □□□□□□□ □□□ □□□□□

Send 60.2

.□□□□ Send □□□ T □□□□□ □□□□□□ □□□□ □□□□□□ thread □□□ T □□□□□□ □□□□□□□ □□□□

) □□□□□□□□□□ □□□ □□□□ □□□□ □□□□□ thread □□□ □□□ (moving ownership) □□□□□□□ □□□□□□□ □□□□□□ □□□ □□□□□□□□□□ □□□□□□ □□□ □□□ □□□□ □□□□ □□□□□ □□□□□□□□□□ .□□□□□ □□□ □□□□□ thread □□□ □□□ (*destructors* .□□□□□ □□□□□□ □□□□□ thread □□□ □□□ □□□ □ □□□□□ □□□□□□ thread □□□ □□□ □□□ □□□□□□

[illegible] SQLite [illegible] thread [illegible].

60.3 Sync

[illegible] T [illegible] Sync [illegible] T[illegible].

[illegible]:

T [illegible] Sync [illegible] T& [illegible] Send [illegible]

[illegible] (pass references) [illegible] thread[illegible].

[illegible] Sync [illegible] thread [illegible] Sync [illegible] thread[illegible]. [illegible] thread[illegible] thread[illegible].

60.4 [illegible]

Send + Sync

[illegible] Send + Sync [illegible]:

- `i8`, `f32`, `bool`, `char`, `&str`, ...
- `(T1, T2)`, `[T; N]`, `&[T]`, `struct { x: T }`, ...
- `(T1, T2)`, `[T; N]`, `&[T]`, `struct { x: T }`, ...
- `String`, `Option<T>`, `Vec<T>`, `Box<T>`, ...
- `Arc<T>`: [illegible] atomic reference [illegible] thread-safe.
- `mpsc::Sender<T>`: [illegible] 1.72.0.
- `AtomicU8`, `AtomicBool`, ...: [illegible] atomic [illegible].

[illegible] generic [illegible] `Send + Sync` [illegible] `Send + Sync`.

Send + !Sync

[illegible]. [illegible] (interior mutability)[illegible]:

- `mpsc::Receiver<T>`
- `Cell<T>`
- `RefCell<T>`

!Send + Sync

These types are safe to access (via shared references) from multiple threads, but they cannot be moved to another thread:

- `MutexGuard<T: Sync>`: Uses OS level primitives which must be deallocated on the thread which created them. However, an already-locked mutex can have its guarded .variable read by any thread with which the guard is shared

`Send + !Sync!`

:□□□□ □□□□□□□ □□□□□ □□□□ □□□□□ □□ □□ □□□□□ □□□□□ □□□□ □ □□□□□□□□ □□□□□ thread □□□□ □□ □□□□□□□ □□□□

- Rc”

  Rc” □□ :”

  RcBox” □□ □□□□□□ □□ □□□□□□ ”

  .□□□□ atomic □□□□ □□□□□□ □□□□□□ □□□□□ □□ □□□□ ”

- □□□□ □□□□□ □□□□ □□□□□□ □□□□□□ □□ □□□□ □□ Rust □□□□ □□□□□ `:const T, *mut T*` □□□□□□□ .□□□□□□ □□□□□□ □□□□□ □□□□□□□□ □□□□□□□□

61 [illegible]

[illegible] [illegible]

[illegible]:

[illegible]	[illegible]
Arc	[illegible]
Mutex	[illegible]
[illegible]	[illegible]

61.1 Arc

`Arc<T>` [illegible] read-only [illegible] `Arc::clone` [illegible]:

```
use std::sync::Arc;
use std::thread;

fn main() {
    let v = Arc::new(vec![10, 20, 30]);
    let mut handles = Vec::new();
    for _ in 0..5 {
        let v = Arc::clone(&v);
        handles.push(thread::spawn(move || {
            let thread_id = thread::current().id();
            println!("{thread_id:?}: {v:?}");
        }));
    }

    handles.into_iter().for_each(|h| h.join().unwrap());
    println!("v: {v:?}");
}
```

- ”Arc” [illegible] ”Atomic Reference Counted” [illegible] `Rc` [illegible] atomic [illegible].
- Arc”

” [illegible] ”Clone” [illegible] T [illegible]. Send و Sync [illegible] و [illegible] T [illegible].

- `Arc::clone()` [illegible] atomic [illegible] 'T' [illegible].
- [illegible] reference cycle [illegible] Arc [illegible] garbage collector [illegible].
  - `std::sync::Weak` [illegible].

61.2 Mutex

`Mutex<T>` [illegible] و [illegible] (mutable) [illegible] ”T” [illegible] read-only interface [illegible] ([illegible] (mutable) [illegible]) [illegible].

```rust
use std::sync::Mutex;

fn main() {
    let v = Mutex::new(vec![10, 20, 30]);
    println!("v: {:?}", v.lock().unwrap());

    {
        let mut guard = v.lock().unwrap();
        guard.push(40);
    }

    println!("v: {:?}", v.lock().unwrap());
}
```

[illegible] `impl<T: Send> Sync for Mutex<T>` [illegible].

- «Mutex» [illegible] Rust [illegible] --- [illegible] (protected) [illegible].
  - [illegible] protected [illegible] mutex [illegible].
- [illegible] lock [illegible] `&mut T` [illegible] `&Mutex<T>` [illegible]. `MutexGuard` [illegible] `&mut T` [illegible].
- `Mutex<T>` [illegible] Send و Sync iff ([illegible] و [illegible]) T [illegible] Send [illegible].
- [illegible] و [illegible]: RwLock.
- [illegible] `lock()` [illegible] Result [illegible]
  - [illegible] thread [illegible] Mutex [illegible] panic [illegible] Mutex «poisoned/[illegible]» [illegible]. [illegible] `lock()` [illegible] mutex [illegible] [PoisonError](https://doc.rust-lang.org/std/sync/struct.PoisonError.html) [illegible]. [illegible]- `into_inner()` [illegible].

61.3 [illegible]

[illegible] Arc و Mutex [illegible]:

```
;use std::thread
;{use std::sync::{Arc, Mutex //

} ()fn main
;[let v = vec![10, 20, 30
} ||)let handle = thread::spawn
;(v.push(10
;({
;(v.push(1000

;()handle.join().unwrap
;("{?:println!("v: {v
{
```

:□□□□□ □□□□□□

```
;{use std::sync::{Arc, Mutex
;use std::thread

} ()fn main
;(([let v = Arc::new(Mutex::new(vec![10, 20, 30

;(let v2 = Arc::clone(&v
} || let handle = thread::spawn(move
;()let mut v2 = v2.lock().unwrap
;(v2.push(10
;({

}
;()let mut v = v.lock().unwrap
;(v.push(1000
{

;()handle.join().unwrap

;("{?:println!("v: {v
{
```

:□□□□□ □□□□□ □□□□□□□□

- .□□□□ □□ □□ □□□□□ □□□□□ □□□□□□ □□□□□ □□□□□□□ □□□□□□ Mutex □ Arc □□ v
  □□□□□ □□□□□ □□□□□□□□ □□□□□□□ □□ □□□□□ □□□□□ □□□□□□ □□ Arc □□ □□ Mutex □□ □□□□□ □□□□□ –
  .□□□□ □□thread □□□ (mutable) □□□□□□
- .□□□□ □□□□□□ □□□□□□ thread □□ □□ □□ □□□□□□ □□ □□□□ □□□□□ v2 □□□□□□ □□ □□□□□ <_>v: Arc
  .□□ □□□□□□ lambda signature □□ move □□□□
- .□□□□□□□ □□□□□□ □□□□□□ □□ □□ LockGuard □□□□□□ □□□□□ □□□□□□ □□□□□ □□□□□□□

62 [illegible]

[illegible]

[illegible]:

[illegible]	[illegible]
Dining [illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

62.1 Dining [illegible]

[illegible] concurrency [illegible]:

> [illegible].

[illegible] Cargo installation [illegible] src/main.rs [illegible] `cargo run` [illegible] (deadlock) [illegible]:

```
use std::sync::{mpsc, Arc, Mutex};
use std::thread;
use std::time::Duration;

struct Fork;

struct Philosopher {
    name: String,
    // left_fork: ...
    // right_fork: ...
```

```rust
    // thoughts: ...
}

impl Philosopher {
    fn think(&self) {
        self.thoughts
            .send(format!("[illegible]! {} [illegible]!", &self.name))
            .unwrap();
    }

    fn eat(&self) {
        // Pick up forks...
        println!("{} is eating...", &self.name);
        thread::sleep(Duration::from_millis(10));
    }
}

static PHILOSOPHERS: &[&str] =
    &["Socrates", "Hypatia", "Plato", "Aristotle", "Pythagoras"];

fn main() {
    // Create forks

    // Create philosophers

    // Make each of them think and eat 100 times

    // Output their thoughts
}
```

[illegible] `Cargo.toml` [illegible]:

```toml
[package]
name = "dining-philosophers"
version = "0.1.0"
edition = "2021"
```

62.2 [illegible]

[illegible] multi-thread [illegible].

[illegible] HTTP [illegible] `reqwest` [illegible]. [illegible] `reqwest` [illegible]. [illegible] `thiserror` [illegible].

Create a new Cargo project and `reqwest` it as a dependency with:

```shell
cargo new link-checker
cd link-checker
```

```
cargo add --features blocking,rustls-tls reqwest
cargo add scraper
cargo add thiserror
```

[illegible] [illegible] [illegible] [illegible] error: no such subcommand [illegible] cargo add[illegible]
.[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] Cargo.toml

:[illegible] [illegible] [illegible] [illegible] Cargo.toml [illegible] [illegible] [illegible] cargo add [illegible]

```
[package]
"name = "link-checker
"version = "0.1.0
"edition = "2021
publish = false

[dependencies]
{ ["reqwest = { version = "0.11.12", features = ["blocking", "rustls-tls
"scraper = "0.13.0
"thiserror = "1.0.37
```

/https://www.google.org [illegible] [illegible] [illegible] [illegible] [illegible] .[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
.[illegible] [illegible]

:[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] src/main.rs [illegible]

```
;use reqwest::blocking::Client
;use reqwest::Url
;{use scraper::{Html, Selector
;use thiserror::Error

} enum Error
,(ReqwestError(#[from] reqwest::Error
,(BadResponse(String
{

} struct CrawlCommand
,url: Url
,extract_links: bool
{

} <fn visit_page(client: &Client, command: &CrawlCommand) -> Result<Vec<Url>, Error
;(command.url ,"{#:} [illegible]")!println
;?()let response = client.get(command.url.clone()).send
} ()if !response.status().is_success
;((()return Err(Error::BadResponse(response.status().to_string
{

;()let mut link_urls = Vec::new
} if !command.extract_links
;(return Ok(link_urls
{

;()let base_url = response.url().to_owned
;?()let body_text = response.text
```

```
;(let document = Html::parse_document(&body_text

;()let selector = Selector::parse("a").unwrap
let href_values = document
(select(&selector.
;(("filter_map(|element| element.value().attr("href.
} for href in href_values
} (match base_url.join(href
} <= (Ok(link_url
;(link_urls.push(link_url
{
} <= (Err(err
;("{println!("On {base_url:#}: ignored unparsable {href:?}: {err
{
{
{
(Ok(link_urls
{

} ()fn main
;()let client = Client::new
;()let start_url = Url::parse("https://www.google.org").unwrap
;{ let crawl_command = CrawlCommand{ url: start_url, extract_links: true
} (match visit_page(&client, &crawl_command
,("{?#:Ok(links) => println!("Links: {links
,("{#:err} :[illegible]")!Err(err) => println
{
{
```

[illegible] `src/main.rs` [illegible]

```
cargo run
```

Task

- [illegible] URL: [illegible] thread [illegible] URL [illegible] thread [illegible] channel [illegible].
- [illegible] «www.google.org» [illegible].

62.3 [illegible]

Dining [illegible]

```
;{use std::sync::{mpsc, Arc, Mutex
;use std::thread
;use std::time::Duration

;struct Fork
```

```
                                                   } struct Philosopher
                                                      ,name: String
                                       ,<<left_fork: Arc<Mutex<Fork
                                      ,<<right_fork: Arc<Mutex<Fork
                                ,<thoughts: mpsc::SyncSender<String
                                                                      {

                                                     } impl Philosopher
                                                  } (fn think(&self
                                                  self.thoughts
       ((self.name& ,"!□□□□ □□□□ □□ {} !□□□□")!send(format.
                                                 ;()unwrap.
                                                                  {

                                                    } (fn eat(&self
                   ;(println!("{} is trying to eat", &self.name
                    ;()let _left = self.left_fork.lock().unwrap
                  ;()let _right = self.right_fork.lock().unwrap

                       ;(println!("{} is eating...", &self.name
                      ;((thread::sleep(Duration::from_millis(10
                                                                  {
                                                                      {

                                         = [static PHILOSOPHERS: &[&str
      ;["Socrates", "Hypatia", "Plato", "Aristotle", "Pythagoras"]&

                                                            } ()fn main
                             ;(let (tx, rx) = mpsc::sync_channel(10

                                (()let forks = (0..PHILOSOPHERS.len
                           (((map(|_| Arc::new(Mutex::new(Fork.
                                          ;()<<_>collect::<Vec.

                                          } ()for i in 0..forks.len
                                           ;()let tx = tx.clone
                     ;([let mut left_fork = Arc::clone(&forks[i
;([()let mut right_fork = Arc::clone(&forks[(i + 1) % forks.len

          To avoid a deadlock, we have to break the symmetry //
  somewhere. This will swap the forks without deinitializing //
                                             .either of them //
                                      } if i == forks.len() - 1
           ;(std::mem::swap(&mut left_fork, &mut right_fork
                                                              {

                                } let philosopher = Philosopher
                         ,()name: PHILOSOPHERS[i].to_string
                                              ,thoughts: tx
                                                 ,left_fork
```

```
,right_fork
;{

} || thread::spawn(move
} for _ in 0..100
;()philosopher.eat
;()philosopher.think
{
;({
{

;(drop(tx
} for thought in rx
;("{println!("{thought
{
{
```

Link □□□□□□□□

```
;{use std::sync::{mpsc, Arc, Mutex
;use std::thread

;use reqwest::blocking::Client
;use reqwest::Url
;{use scraper::{Html, Selector
;use thiserror::Error

} enum Error
,(ReqwestError(#[from] reqwest::Error
,(BadResponse(String
{

} struct CrawlCommand
,url: Url
,extract_links: bool
{

} <fn visit_page(client: &Client, command: &CrawlCommand) -> Result<Vec<Url>, Error
;(command.url ,"{#:} □□□□□")!println
;?()let response = client.get(command.url.clone()).send
} ()if !response.status().is_success
;((()return Err(Error::BadResponse(response.status().to_string
{

;()let mut link_urls = Vec::new
} if !command.extract_links
;(return Ok(link_urls
{

;()let base_url = response.url().to_owned
;?()let body_text = response.text
```

```
                                 ;(let document = Html::parse_document(&body_text

                                    ;()let selector = Selector::parse("a").unwrap
                                                       let href_values = document
                                                           (select(&selector.
                         ;(("filter_map(|element| element.value().attr("href.
                                                        } for href in href_values
                                                  } (match base_url.join(href
                                                        } <= (Ok(link_url
                                            ;(link_urls.push(link_url
                                                                        {
                                                            } <= (Err(err
     ;("{println!("On {base_url:#}: ignored unparsable {href:?}: {err
                                                                        {
                                                                            {
                                                                                {
                                                                    (Ok(link_urls
                                                                                    {

                                                                  } struct CrawlState
                                                                  ,domain: String
                                ,<visited_pages: std::collections::HashSet<String
                                                                                    {

                                                                    } impl CrawlState
                                          } fn new(start_url: &Url) -> CrawlState
                    ;()let mut visited_pages = std::collections::HashSet::new
                        ;(()visited_pages.insert(start_url.as_str().to_string
{ CrawlState { domain: start_url.domain().unwrap().to_string(), visited_pages
                                                                                {

           .Determine whether links within the given page should be extracted ///
                              } fn should_extract_links(&self, url: &Url) -> bool
                                   } let Some(url_domain) = url.domain() else
                                                            ;return false
                                                                           ;{
                                                    url_domain == self.domain
                                                                                {

            Mark the given page as visited, returning false if it had already ///
                                                                .been visited ///
                                  } fn mark_visited(&mut self, url: &Url) -> bool
                          (()self.visited_pages.insert(url.as_str().to_string
                                                                                {
                                                                                    {

                                   ;<(type CrawlResult = Result<Vec<Url>, (Url, Error
                                                            )fn spawn_crawler_threads
                                  ,<command_receiver: mpsc::Receiver<CrawlCommand
                                        ,<result_sender: mpsc::Sender<CrawlResult
                                                               ,thread_count: u32
                                        } (
        ;((let command_receiver = Arc::new(Mutex::new(command_receiver

                          } for _ in 0..thread_count
                ;()let result_sender = result_sender.clone
             ;()let command_receiver = command_receiver.clone
                          } || thread::spawn(move
                      ;()let client = Client::new
                                } loop
                      } = let command_result
    ;()let receiver_guard = command_receiver.lock().unwrap
                     ()receiver_guard.recv
                                ;{
           } let Ok(crawl_command) = command_result else
    .The sender got dropped. No more commands coming in //
                            ;break
                                ;{
  } (let crawl_result = match visit_page(&client, &crawl_command
                ,(Ok(link_urls) => Ok(link_urls
        ,((Err(error) => Err((crawl_command.url, error
                                ;{
            ;()result_sender.send(crawl_result).unwrap
                                   {
                                    ;({
                                       {
                                         {

                                 )fn control_crawl
                                ,start_url: Url
                  ,<command_sender: mpsc::Sender<CrawlCommand
                 ,<result_receiver: mpsc::Receiver<CrawlResult
                                  } <Vec<Url <- (
              ;(let mut crawl_state = CrawlState::new(&start_url
   ;{ let start_command = CrawlCommand { url: start_url, extract_links: true
                 ;()command_sender.send(start_command).unwrap
                           ;let mut pending_urls = 1

                        ;()let mut bad_urls = Vec::new
                           } while pending_urls > 0
            ;()let crawl_result = result_receiver.recv().unwrap
                            ;pending_urls -= 1

                           } match crawl_result
                          } <= (Ok(link_urls
                      } for url in link_urls
              } (if crawl_state.mark_visited(&url
;(let extract_links = crawl_state.should_extract_links(&url
 ;{ let crawl_command = CrawlCommand { url, extract_links
       ;()command_sender.send(crawl_command).unwrap
                    ;pending_urls += 1
                               {
```

```
                                                            {
                                                                {
                                           } <= ((Err((url, error
                                          ;(bad_urls.push(url
            ;(error ,"{#:} :□□ □□□□□□□ crawling □□□□")!println
                                                    ;continue
                                                                {
                                                                    {
                                                                        {
                                                                 bad_urls
                                                                            {

                                 } <fn check_links(start_url: Url) -> Vec<Url
   ;()<let (result_sender, result_receiver) = mpsc::channel::<CrawlResult
;()<let (command_sender, command_receiver) = mpsc::channel::<CrawlCommand
              ;(spawn_crawler_threads(command_receiver, result_sender, 16
                (control_crawl(start_url, command_sender, result_receiver
                                                                            {

                                                                  } ()fn main
  ;()let start_url = reqwest::Url::parse("https://www.google.org").unwrap
                                   ;(let bad_urls = check_links(start_url
                                   ;(println!("Bad URLs: {:#?}", bad_urls
                                                                            {
```

XIV [illegible]

[illegible] :[illegible]

63 [illegible]

[illegible]

”Async” [illegible] concurrency [illegible] task [illegible] I/O [illegible].

[illegible] asynchronous Rust [illegible] ”futures” [illegible] future. [illegible] «polled» [illegible].

Future [illegible].

[illegible]

- [illegible] «asyncio» [illegible]. [illegible] «Future» [illegible] callback [illegible] poll [illegible]. [illegible] Async Python [illegible] «[illegible]» [illegible] runtime [illegible] Rust [illegible].
- [illegible] ”Promise” [illegible] callback [illegible]. runtime [illegible] (event loop) [illegible] Promise [illegible].

[illegible]

[illegible]. [illegible]:

[illegible]	[illegible]
Async [illegible]	[illegible]
Control Flow [illegible]	[illegible]
Pitfall[illegible]	[illegible]
[illegible]	[illegible]

64 [illegible]

Async [illegible]

[illegible]:

[illegible]	[illegible]
async/await	[illegible]
Futures	[illegible]
Runtimes	[illegible]
Task	[illegible]

64.1 async/await

At a high level, async Rust code looks very much like ”normal” sequential code:

```rust
use futures::executor::block_on;

async fn count_to(count: i32) {
    for i in 0..count {
        println!("Count is: {i}!");
    }
}

async fn async_main(count: i32) {
    count_to(count).await;
}

fn main() {
    block_on(async_main(10));
}
```

[illegible]:

- [illegible] syntax [illegible]. [illegible] (concurrency) [illegible]!
- [illegible] async call [illegible]

main ;(let future: () = async_main(10 type   –
.

- •   .   Rust   syntax ”async”
.   future

- •
. async main future

- • block_on.   (executor)
.   future   thread

- •   .   (asyn)   await
.   thread await block_on

- •   block) async   await.
.(

Futures 64.2

                    object   trait Future
. Poll poll   poll future .

```
;use std::pin::Pin
;use std::task::Context

} pub trait Future
;type Output
;<fn poll(self: Pin<&mut Self>, cx: &mut Context<'_>) -> Poll<Self::Output
{

} <pub enum Poll<T
,(Ready(T
,Pending
{
```

 ( )   . impl Future async
        JoinHandle .   Future
.   (joining)   tokio::spawn Future

     async   Future   await.
.   Future

- • .   Poll Future
.

- •   async   Context Pin
:   .   async

–   Future Context
.   poll

– pointer   Future   Pin
    reference   future
.   await.

64.3 Runtimes

[illegible] *runtime* [illegible] (_a_reactor) [illegible] future [illegible] (an *executor*) [illegible]. Rust [illegible] runtime [illegible]:

- Tokio: [illegible](performant)[illegible] Hyper [illegible] HTTP [illegible] [Tonic](https://github.com/hyperium/tonic) [illegible] gRPC.
- async-std: [illegible] "std for async" [illegible] runtime [illegible] `async::task` [illegible].
- smol: [illegible]

[illegible] (runtime) [illegible]. [illegible] Fuchsia [illegible] runtime [illegible].

- [illegible] Tokio [illegible] playground [illegible] Rust [illegible] playground. [illegible] (I/O) [illegible] async [illegible] playground [illegible].

- Future [illegible] «(inert)[illegible]» [illegible] ([illegible] I/O [illegible]) [illegible] (executor)[illegible] polling [illegible]. [illegible] JS Promises [illegible].

64.3.1 Tokio

Tokio provides:

- [illegible] runtime [illegible] multi-thread [illegible](asynchronous).
- [illegible] asynchronous version [illegible].
- [illegible].

```rust
use tokio::time;

async fn count_to(count: i32) {
    for i in 0..count {
        println!("[illegible] task: {i}!");
        time::sleep(time::Duration::from_millis(5)).await;
    }
}

async fn main() {
    tokio::spawn(count_to(10));

    for i in 0..5 {
        println!("Main task: {i}");
        time::sleep(time::Duration::from_millis(5)).await;
    }
}
```

- [illegible] `tokio::main` [illegible] `main` [illegible] async [illegible].

- [illegible] spawn [illegible] "task" [illegible].

- [illegible]: spawn [illegible] Future [illegible] `.await` [illegible] `count_to` [illegible].

Further □□□□□□:

- .□□□□ async □□□□ □□ □□□□□□□□ □□□□ □□□□□□□□ □□ □□□□□□ □□ (□□□□□□□□) `count_to` □□□□ .□□□□ □□□□□ □□ □□□□□ □□□□□□ □□□□□ □□□□□□□ □□ □□□□□□□□□□□ □□ handle □□ `tokio::spawn`
- .□□□□□ □□□□□□□□ □□ «`count_to(10).await` □□□□□ spawn □□□□ □□□
- .□□□□□□ `tokio::spawn` □□ □□□□□□□ □□□□ □□□□□□

Task 64.4

.□□□□ □□□□ □□□□ thread □□□□□ □□ □□□□□ task system □□ Rust

-□□ poll □□ □□ □□□□ □□□□□□ □□□□□ (executor) □□□□□□□□□□ □□ □□□□□ □□□□□ □□□□ □□ future □□ task □□ `poll` □□ □□ `poll` □□□□ □□ □□□□□ □□□□□□ □□□□□□□ future □□□□ □□ □□ □□□□ □□□□□ future □□ .□□□□ □□ task □□ □□ □□□□□□□□□ .□□□□□ □□□□□□□ □□□□ □□□□□□□□□□ stack □□ □□ □□□□□□□□□□□ □□□□ □□ □□ □□□□□□□ .□□□□ □□□□□□□□□□□ I/O □□□□□□□ □□ □ □□□□□□ □□ □□□□□□ □□□□□□ □child future □□□□□□ □□ poll

```
                            ;{use tokio::io::{self, AsyncReadExt, AsyncWriteExt
                                                   ;use tokio::net::TcpListener

                                            } <()>async fn main() -> io::Result
                    ;?let listener = TcpListener::bind("127.0.0.1:0").await
           ;(()println!("listening on port {}", listener.local_addr()?.port

                                                                     } loop
                     ;?let (mut socket, addr) = listener.accept().await

                                  ;("{?:println!("connection from {addr

                                              } tokio::spawn(async move
  ;("socket.write_all(b"Who are you?\n").await.expect("socket error

                                       ;[let mut buf = vec![0; 1024
;("let name_size = socket.read(&mut buf).await.expect("socket error
 ;()let name = std::str::from_utf8(&buf[..name_size]).unwrap().trim
           ;("name}!\n} □□□□□□□□□ □□□□□□ □□□□□ □□")!let reply = format
   ;("socket.write_all(reply.as_bytes()).await.expect("socket error
                                                                    ;({
                                                                          {
                                                                              {
```

.□□□□□ □□□□□ □□□□□ □□ □□ □□ □ □□□□□ □□□□ □□□□ □□□□ □□□□□□ `src/main.rs` □□ □□ □□□□□ □□□□

.□□□□□ □□□□□ □□ □□ telnet □□ nc □□□□□□ TCP □□□□□□ □□□□□□ □□ □□ □□□□□ □□□□

- Ask students to visualize what the state of the example server would be with a few ?connected clients. What tasks exist? What are their Futures
- This is the first time we've seen an `async` block. This is similar to a closure, but does not .take any arguments. Its return value is a Future, similar to an `async fn`
- .□□□□□□ □□□□□□ ? □□ □□□□□□□□ □□ □□ □□□□ □□□□□□□ □ □□□□□ □□□□□□ □□□□□ □□ □□ □□ async □□□□□

65 □□□

Control Flow □ □□□□□□□

:□□□□ □□□ .□□□□ □□□ □□□□□□ □□ □□□□ □□□□ □□□ □□□

□□□□□□	□□□□ □□□
Async □□□□□□□□	□□□□□ □□
Join	□□□□□ □
Select	□□□□□ □

Async □□□□□□□□□ 65.1

:tokio □□□□ □□□□□ □□ .□□□□□□□ □□□□□□□□□ asynchronous channel □□ crate □□□□□

```
                         ;{use tokio::sync::mpsc::{self, Receiver

                 } (<()>async fn ping_handler(mut input: Receiver
                                    ;let mut count: usize = 0

                     } while let Some(_) = input.recv().await
                                              ;count += 1
     ;(".□□□ □□□ □□□□□□ ping □□□ {count} □□□□□□")!println
                                                        {

                         ;("□□□□□□□□□□ println!("ping_handler
                                                                {

                                                } ()async fn main
                  ;(let (sender, receiver) = mpsc::channel(32
;((let ping_handler_task = tokio::spawn(ping_handler(receiver
                                             } for i in 0..10
    ;(".sender.send(()).await.expect("Failed to send ping
                ;(println!("Sent {} pings so far.", i + 1
                                                        {

                                                ;(drop(sender
    ping_handler_task.await.expect("Something went wrong in ping handler task.");
}
```

- [illegible] 3 [illegible].
- [illegible] interface [illegible] channel [illegible] sync [illegible].
- [illegible] `std::mem::drop` [illegible]. [illegible]
- [illegible] crate [illegible] Flume [illegible] `sync` [illegible] `async` `send` [illegible] `recv` [illegible]. [illegible] task [illegible] IO [illegible] CPU [illegible].
- [illegible] `async` [illegible] `future` [illegible].

65.2 Join

[illegible] (join) [illegible] future [illegible] (collection) [illegible]. [illegible] `Promise.all` [illegible] JavaScript [illegible] `asyncio.gather` [illegible].

```
use anyhow::Result;
use futures::future;
use reqwest;
use std::collections::HashMap;

async fn size_of_page(url: &str) -> Result<usize> {
    let resp = reqwest::get(url).await?;
    Ok(resp.text().await?.len())
}

async fn main() {
    let urls: [&str; 4] = [
        "https://google.com",
        "https://httpbin.org/ip",
        "https://play.rust-lang.org/",
        "BAD_URL",
    ];
    let futures_iter = urls.into_iter().map(size_of_page);
    let results = future::join_all(futures_iter).await;
    let page_sizes_dict: HashMap<&str, Result<usize>> =
        urls.into_iter().zip(results.into_iter()).collect();
    println!("{:?}", page_sizes_dict);
}
```

[illegible] `src/main.rs` [illegible].

- [illegible] future [illegible] `std::future::join!` [illegible] future [illegible]. [illegible] crate) [illegible] `futures` [illegible] `std::future` [illegible].
- [illegible] `join` [illegible] future [illegible] resolve [illegible].

- [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `!join` [illegible] [illegible] `join_all` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] http [illegible] [illegible] [illegible] [illegible] `(!join)` [illegible] .[illegible] [illegible] future [illegible] `!futures::join` [illegible] [illegible] [illegible] [illegible] `tokio::time::sleep` [illegible] ([illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]`!select` [illegible] [illegible]) [illegible] timeout [illegible] .[illegible] [illegible] [illegible] `!join`

Select 65.3

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]future [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `Promise.race` [illegible] [illegible] [illegible] [illegible] JavaScript [illegible] .[illegible] [illegible] future [illegible] [illegible] [illegible] `(asyncio.wait(task_set, return_when=asyncio.FIRST_COMPLETED` [illegible] .[illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `pattern` [illegible] [illegible](match statement) [illegible] [illegible] [illegible] [illegible] [illegible] `future` [illegible] [illegible] .[illegible] `pattern = future => statement` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `statement` [illegible] .[illegible] [illegible] `pattern` [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] .[illegible] `!select` [illegible] [illegible] `statement` [illegible] [illegible] .[illegible] [illegible]

```
                                                    ;use tokio::sync::mpsc
                                          ;{use tokio::time::{sleep, Duration

                                                        } ()async fn main
                                    ;(let (tx, mut rx) = mpsc::channel(32
                                  } let listener = tokio::spawn(async move
                                                  } !tokio::select
                  ,("{Some(msg) = rx.recv() => println!("got: {msg
         ,("sleep(Duration::from_millis(50)) => println!("timeout = _
                                                              ;{
                                                                ;({
                                   ;sleep(Duration::from_millis(10)).await
;("await.expect("Failed to send greeting.(("![illegible]")tx.send(String::from

                                 ;("[illegible] [illegible] [illegible]")listener.await.expect
                                                                   {
```

- [illegible] async [illegible] [illegible] [illegible] :[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] async `listener` [illegible] [illegible] [illegible] [illegible] [illegible] sleep [illegible] [illegible] `sleep` .[illegible] [illegible]timeout [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] `send` [illegible] .[illegible] [illegible] [illegible] [illegible]
- `select!` is also often used in a loop in ”actor” architectures, where a task reacts to .events in a loop. That has some pitfalls, which will be discussed in the next segment

66 □□□

□□Pitfall

□□□□□□ concurrent asynchronous □□□□□□ □□□□□□□□ □□□□□ □□ □□□□□□□□□ □ □□□□□ □□□□□□□ Async / await
□□footgun □ □□pitfall □ □□□□□□□ □□ □□□□ □□□□ □□ □□□□ Rust □□ async/wait □□□□ □□□□□□□□□□□□ .□□□□□□
.□□□□□□□ □□□□□□ □□□□ □□□□ □□ □□ □□□□□ □□ □□□□□ .□□□□ □□□□□□

:□□□□□ □□ .□□□□□ □□□□ □□□□□□ □□ □□□□□ □□□□□ □□□□ □□□□

□□□□□□□□	□□□□□ □□□□
Executor □□□□□ □□□□□□	□□□□□□ □□
Pin	□□□□□□ □□
Async □□□□□	□□□□□□ □
□□□	□□□□□□ □□

executor □□□□□ □□□□□□ 66.1

□□□□□ (concurrent) □□□□□□□ □□□□□ □□ □□ □□□□□□□ □□□□□□ IO task □□ □□□□□ □□□async runtime □□□□□
□□□□□□□□ □ executor □□□ □□□□□□ □□□□□ CPU □□□□□ block □□□□□□□ □□ □□□□ □□□□□ □□□□□ □□□□ .□□□□□
□□□□□□ □□□□□□□ □□□ □□□□□□ □□□□□ □□ □□ □□□□ □□□□ □□□□□ □□ □□□□ □□ .□□□□□□ □□□□□□ □□□□□ □□□□□□ □□□
.□□□□□ □□□□□□□□ async

```
                                  ;use futures::future::join_all
                                         ;use std::time::Instant

          } (async fn sleep_ms(start: &Instant, id: u64, duration_ms: u64
 ;((std::thread::sleep(std::time::Duration::from_millis(duration_ms
                                                          )!println
  ,"future {id} slept for {duration_ms}ms, finished after {}ms"
                                        ()start.elapsed().as_millis
                                                                 ;(
                                                                   {

                                                   } ()async fn main
                                          ;()let start = Instant::now
 ;((let sleep_futures = (1..=10).map(|t| sleep_ms(&start, t, t * 10
    join_all(sleep_futures).await;
}
```

- [illegible] sleep [illegible] (concurrent).
- [illegible] "current_thread" [illegible] task [illegible] thread [illegible]. [illegible] multi-threaded [illegible].
- `std::thread::sleep` [illegible] `tokio::time::sleep` [illegible].
- [illegible] `tokio::task::spawn_blocking` [illegible] thread [illegible] handle [illegible] executor [illegible] future [illegible].
- [illegible] task [illegible] thread [illegible]. [illegible] executor [illegible] task [illegible] thread. [illegible] FFI [illegible] thread [illegible] thread [illegible] (map) [illegible] thread [illegible] (CUDA [illegible]) [illegible]. [illegible] `tokio::task::spawn_blocking` [illegible].
- [illegible] mutex [illegible]. [illegible] mutex [illegible] `.await` [illegible] task [illegible] task [illegible] thread [illegible].

66.2 Pin

[illegible] Async [illegible] `Future` [illegible]. [illegible] future [illegible].

[illegible]. [illegible] future [illegible] pointer [illegible].

[illegible] future [illegible] pointer [illegible]. `Pin` [illegible] wrapper [illegible] reference [illegible].

```
use tokio::sync::{mpsc, oneshot};
use tokio::task::spawn;
use tokio::time::{sleep, Duration};

// A work item. In this case, just sleep for the given time and respond
// with a message on the `respond_on` channel.
struct Work {
    input: u32,
    respond_on: oneshot::Sender<u32>,
}

// A worker which listens for work on a queue and performs it.
async fn worker(mut work_queue: mpsc::Receiver<Work>) {
    let mut iterations = 0;
    loop {
                                                   } !tokio::select
                            } <= ()Some(work) = work_queue.recv
.sleep(Duration::from_millis(10)).await; // Pretend to work
                                            work.respond_on
                               (send(work.input * 1000.
                    ;("expect("failed to send response.
                                           ;iterations += 1
                                                              {
               TODO: report number of iterations every 100ms //
                                                                  {
                                                                      {
                                                                          {

           .A requester which requests work and waits for it to complete //
     } async fn do_work(work_queue: &mpsc::Sender<Work>, input: u32) -> u32
                                     ;()let (tx, rx) = oneshot::channel
                                                             work_queue
                              ({ send(Work { input, respond_on: tx.
                                                             await.
                           ;("expect("failed to send on work queue.
                         ("rx.await.expect("failed waiting for response
                                                                          {

                                                          } ()async fn main
                                      ;(let (tx, rx) = mpsc::channel(10
                                                     ;((spawn(worker(rx
                                                      } for i in 0..100
                                  ;let resp = do_work(&tx, i).await
                     ;("{i}: {resp}  [illegible]")!println
                                                                      {
                                                                          {
```

- .[illegible] (actor pattern) [illegible]
  .[illegible] !select [illegible]
- [illegible]
  .[illegible]
  - (..)!sleep(Duration::from_millis(100)) => { println = _[illegible]
    [illegible] .[illegible] .[illegible] !select [illegible] {
  - :[illegible] loop [illegible] future [illegible] timeout_fut [illegible]

```
;((let timeout_fut = sleep(Duration::from_millis(100
                                              } loop
                                       } !select
                                         ,..
       ,{ ;(..)!timeout_fut => { println = _
                                               {
                                                   {
```

  - This still doesn't work. Follow the compiler errors, adding &mut to the timeout_fut
    :in the select! to work around the move, then using Box::pin

```
;(((let mut timeout_fut = Box::pin(sleep(Duration::from_millis(100
                                                            } loop
                                                       } !select
                                                            ,..
             ,{ ;(..)!mut timeout_fut => { println& = _
                                                               {
                                                                  {
```

□□ □□□□□□ □□□□□□ □□ □□ □ timeout □□□□□□□ □□ □□ □□□□ □□□□□□□ □□□□□□□□□ □□□□□ □□□□ – □□□□□ □□□□□□□□□□□□□ .(□□□□□□ □□□□ □□□□□□ □□□□ □□ □□□□□□□ future) □□□□ `Poll::Ready` :□□□□□□ □□□□□□ □□ □□□□ □□ `timeout_fut` □□□□□□□□□□

```
;(((let mut timeout_fut = Box::pin(sleep(Duration::from_millis(100
                                                            } loop
                                                       } !select
                             } <= mut timeout_fut& = _
                                         ;(..)!println
;(((timeout_fut = Box::pin(sleep(Duration::from_millis(100
                                                         ,{
                                                               {
                                                                  {
```

- Box allocates on the heap. In some cases, `std::pin::pin!` (only recently stabilized, with older code often using `tokio::pin!`) is also an option, but that is difficult to use .for a future that is reassigned
- □□□□□□ □□□□□□ task □□□□□ □□□□□□□ □□□□□□□□□ `pin` □□ □□□□ □□□□ □□ □□ □□□□ □□□□ □□□□□ □□□□□□□□□ .□□□□□□ □□□□□□ `oneshot` □□□□□□ □□ □□ □□□□□□□□□□ 100 □□ □□ □□□□□
- □□□□□□□ (self-referential) □□□□□□□ □□□□ □□□□□□□□ □□□□ □□ □□□□□□□□□□□□□ □□□□□ □□ □□□□□□□□□□ □□□□□□□□ □□□□□□□□□□□ □□□□□□□□ □□□□□□□□□ □□ Rust borrow checker □□□□□□□ □□□□ □□ .□□□□□ □□ □□□□□ □□□□□□□ □□□□□□ □□□□□ □□ □□ □□□□□□□□□□ □□ □□□□□□ □□□□□□□□□□□ □□□□□□ □□□□□ □□□□□□□ □□□□□□ borrow checker □□□□□ async □□□□□□ □ □□□□□□□ □□□□□ □□ □□□□□□ □□□□□□□□□□□ .□□□□□□□ .□□□□□□□
- □□ □□ □□□□□□□□ □□ □□□□□□□□□ □□ object □□ .□□□□ reference □□ □□□□□□ □□ wrapper □□ `Pin` □□ □□□□□ □□ □□ □□ □□□□□□□□ □□ □□□□□ □□□□□ □□□□ □□ .□□□□ □□□□□ □□□□ □□□□ □□ □□□□ □□□□ pointer .□□□□ □□□□□□ □□□□ □□□□□ pointer
- □□ □□□□□□ □□□□□ `mut Self&` □□□□ □□ `<Pin<&mut Self` □□ `Future` □□□□□□ □□ `poll` □□□□ □□ □□□□ □□ □□ □□□□□□□ □□□□ □□ □□□□ □□□□□ □□□□□ □□ .□□□□□□ □□□□□□□□□ (instance) □□□□□□ .□□□□ □□□□□□□□□□ □□□□ □□□□ □□□□□□□□□

Async □□□□□ 66.3

□□□□□ □□□□□□□□□□ □□ □□□□□ □□□□ .□□□□□□□ □□□□□□ 1.75 □□□□□□□ □□ □□□□□□□ □□trait □□ Async □□□□□□□ □□□□□□□□□□□□ □□□□□ □□□□□□ □□ □□trait □□ (`impl Trait` (RPIT □□□□□□□□ □□□□□□□ □□ □□□□□□□□□ .□□□ `<... = impl Future<Output <-` □□□□□ `async fn` □□□□□ (desugaring)

□□ RPIT □ `async fn` □□□□□ □□ □□□□□□□ □□ □□□□□ □□□□□□□ native □□□□□□□□□□ □□ □□□□ □□□□□□□□□ □□ :□□□□□ □□□□□ □□□□□□□□□

- □□□□□□□□□□) □□□□ □□ □□□□ □□ □□□□□□□ □□□□□ □□□□□□□ □□□□ □□□□□ Return-position impl Trait (□□□□□ □□□□□ □□□□□□□ □□□□ (borrowing) □□□□□ □□□□ □□□□□□□□ □□ □□□□□

- [illegible] `impl trait` [illegible] `async` [illegible] `dyn` [illegible].

If we do need dyn support, the crate async_trait provides a workaround through a macro, with some caveats:

```rust
use async_trait::async_trait;
use std::time::Instant;
use tokio::time::{sleep, Duration};

trait Sleeper {
    async fn sleep(&self);
}

struct FixedSleeper {
    sleep_ms: u64,
}

impl Sleeper for FixedSleeper {
    async fn sleep(&self) {
        sleep(Duration::from_millis(self.sleep_ms)).await;
    }
}

async fn run_all_sleepers_multiple_times(
    sleepers: Vec<Box<dyn Sleeper>>,
    n_times: usize,
) {
    for _ in 0..n_times {
        println!("running all sleepers..");
        for sleeper in &sleepers {
            let start = Instant::now();
            sleeper.sleep().await;
            println!("slept for {}ms", start.elapsed().as_millis());
        }
    }
}

async fn main() {
    let sleepers: Vec<Box<dyn Sleeper>> = vec![
        Box::new(FixedSleeper { sleep_ms: 50 }),
        Box::new(FixedSleeper { sleep_ms: 100 }),
    ];
    run_all_sleepers_multiple_times(sleepers, 5).await;
}
```

- [illegible] `async_trait` [illegible] heap allocation [illegible] heap allocation [illegible].
- [illegible] `async trait` [illegible] Rust [illegible] Niko Matsakis [illegible].

- [illegible] sleep [illegible] Vec [illegible].

66.4 [illegible]

[illegible] future [illegible] poll [illegible]. *cancellation* [illegible] await [illegible]. [illegible] future [illegible]. [illegible] (deadlock) [illegible].

```rust
use std::io::{self, ErrorKind};
use std::time::Duration;
use tokio::io::{AsyncReadExt, AsyncWriteExt, DuplexStream};

struct LinesReader {
    stream: DuplexStream,
}

impl LinesReader {
    fn new(stream: DuplexStream) -> Self {
        Self { stream }
    }

    async fn next(&mut self) -> io::Result<Option<String>> {
        let mut bytes = Vec::new();
        let mut buf = [0];
        while self.stream.read(&mut buf[..]).await? != 0 {
            bytes.push(buf[0]);
            if buf[0] == b'\n' {
                break;
            }
        }
        if bytes.is_empty() {
            return Ok(None);
        }
        let s = String::from_utf8(bytes)
            .map_err(|_| io::Error::new(ErrorKind::InvalidData, "not UTF-8"))?;
        Ok(Some(s))
    }
}

async fn slow_copy(source: String, mut dest: DuplexStream) -> std::io::Result<()> {
    for b in source.bytes() {
        dest.write_u8(b).await?;
        tokio::time::sleep(Duration::from_millis(10)).await
    }
    Ok(())
}

async fn main() -> std::io::Result<()> {
```

```
                                       ;(let (client, server) = tokio::io::duplex(5
           ;((let handle = tokio::spawn(slow_copy("hi\nthere\n".to_owned(), client

                                          ;(let mut lines = LinesReader::new(server
              ;((let mut interval = tokio::time::interval(Duration::from_millis(60
                                                                           } loop
                                                              } !tokio::select
                                  ,("!interval.tick() => println!("tick = _
                            } ?line = lines.next() => if let Some(l) = line
                                                        (print!("{}", l
                                                                  } else {
                                                                 break
                                                                        ,{
                                                                             {
                                                                                 {
                                                           ;?()handle.await.unwrap
                                                                           (())Ok
                                                                                     {
```

- □ □□□□□□□□ □□ API □□□□□□□□ □□□□ .□□□□□□□ □□□□ cancellation-safety □□□□ □□ □□□□□□□□□□
.□□□□ □□□□□□□ □□ □□□□ `async fn` □□ □□□□□□□ □□□□ □□

- (□□□ □□ □□□□□□□ □) □□□□ □□□□□□ □□□□□□ □□ □□□□ cancellation □□ □□□□ □? □ panic □□□□□□□
.□□□

- .□□□□□□ □□□□ □□ □□ string □□ □□□□□□□□ □□□□□ □□□□
  - .□□□□□ □□ □□□□ □□ buf □ ()next □□□□□ □□□□□ □□□□ ()tick □□□□□ □□ □□□□□ □□
  - :□□□□ □□□□□ □□□□□□□□ □□ □□□□□ □□ buf □□□□□□ □□ □□□□□□□ □□ LinesReader

```
                                          } struct LinesReader
                                    ,stream: DuplexStream
                                          ,<bytes: Vec<u8
                                            ,[buf: [u8; 1
                                                          {

                                            } impl LinesReader
                     } fn new(stream: DuplexStream) -> Self
           { [Self { stream, bytes: Vec::new(), buf: [0
                                                      {
      } <<async fn next(&mut self) -> io::Result<Option<String
                     .prefix buf and bytes with self //
                                                ... //
             ;(let raw = std::mem::take(&mut self.bytes
                        (let s = String::from_utf8(raw
;?(("map_err(|_| io::Error::new(ErrorKind::InvalidData, "not UTF-8.
                                                ... //
                                                      {
                                                          {
```

- □□□□□□ tick □□ □□□□ □□ □□□□□□ □□□□□□□□ □□□□□ □□□□ cancellation-safe □□□□□ 'Interval::tick'
.□□□□ □□□□ □□□□□

- □□ □□□□□□□□□□□□ □□ □□□□□□□ □□□□□ □□□□ cancellation-safe □□□□□ `AsyncReadExt::read`

[illegible].

- `AsyncBufReadExt::read_line` [illegible] [illegible] [illegible] [illegible] [illegible] cancellation-safe [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

67 □□□

□□□□□□□

□□□□ □□□□ □□□□□□□ □□□□□□ □□□□ □□□□ .□□□□□ □□□□ □□□□□□□ □□□ □ □□□□□ □ □□□□□ □□□□□ □□□□ □□□□:

□□□□□□□□	□□□□□ □□□□
Dining □□□□□□	□□□□□□ □□
□□ □□□□□□□□ □□□□	□□□□□□ □□
□□□□□ □□□□	□□□□□□ □□

67.1 Dining Philosophers --- Async

□□□□□ □□□□□□□□ dining philosophers □□ □□□□□ □□□□□□ □□□□□.

□□□□□□ □□□□□ (Cargo installation](../../cargo/running-locally.md □□ □□□□□□□ □□□□ □□□□□ □□□□□ □□□□□□□.
□□ □□□□□ □□□□ □ □□□□□ □□ □□ □□□□□ □□□□□□□ □□□□□□□ □□□□ `src/main.rs` □□□□ □□ □□□□□□□ □□ □□ □□□□ □□
□□□□□□□□□ (`src/main.rs`) □□□□ □□ □□ `cargo run`:

```rust
use std::sync::Arc;
use tokio::sync::mpsc::{self, Sender};
use tokio::sync::Mutex;
use tokio::time;

struct Fork;

struct Philosopher {
    name: String,
    // left_fork: ...
    // right_fork: ...
    // thoughts: ...
}

impl Philosopher {
    async fn think(&self) {
        self.thoughts
            .send(format!("!□□□□□ □□□□□ □□ {} !□□□□□", &self.name))
            .await
```

```
                                   ;()unwrap.
                                            {

                              } (async fn eat(&self
               Keep trying until we have both forks //
              ;(println!("{} is eating...", &self.name
         ;time::sleep(time::Duration::from_millis(5)).await
                                            {
                                              {

                             = [static PHILOSOPHERS: &[&str
     ;["Socrates", "Hypatia", "Plato", "Aristotle", "Pythagoras"]&

                                         } ()async fn main
                                          Create forks //

                                   Create philosophers //

                                 Make them think and eat //

                                  Output their thoughts //
                                              {
```

[illegible] Async Rust [illegible] tokio [illegible].
[illegible] Cargo.toml [illegible]:

```
                       [package]
 "name = "dining-philosophers-async-dine
                  "version = "0.1.0
                   "edition = "2021

                  [dependencies]
[ ["tokio = { version = "1.26.0", features = ["sync", "time", "macros", "rt-multi-thread
```

[illegible] Mutex [illegible] mpsc [illegible] tokio crate [illegible]
[illegible].

- [illegible] thread [illegible]

67.2 [illegible]

[illegible] broadcast chat [illegible]. [illegible]. [illegible]. [illegible].

[illegible] broadcast channel [illegible] tokio_websockets [illegible]
[illegible].

[illegible] Cargo [illegible]:

Cargo.toml:

```toml
[package]
name = "chat-async"
version = "0.1.0"
edition = "2021"

[dependencies]
futures-util = { version = "0.3.30", features = ["sink"] }
http = "1.1.0"
tokio = { version = "1.40.0", features = ["full"] }
ebsockets = { version = "0.9.0", features = ["client", "fastrand", "server", "sha1_smol
```

□□□□□ □□□□□ □□□API

□□□□□□□ □□□□□ □□ □□□□□□ □□□□ .□□□□□□ □□□□□ tokio_websockets □ tokio □□ □□□□ □□□□□□ □□ □□□□ .□□□□□ □□□□□□□ API □□

- ()StreamExt::next □□□□□ WebSocketStream: □□□□□ □□□□□□□ □□□□□□□□□□ □□□□□□□ □□ □□ .□□□□□ □□ □□□□□□
- ()SinkExt::send □□□□□□□□□□ □□□□ □□□□ WebSocketStream: □□□□□ □□□□□□ □□□□□□□□□ □□□□□□□ .Websocket Stream □□ □□
- ()Lines::next_line: □□□□□ □□□□□□□ □□□□□□□□□ □□□□□□□□ □□□□□□ □□ □□□□□□ □□□□□□□□□□.
- ()Sender::subscribe: □□□□□ □□□□□□□ □□ □□ broadcast channel.

□□□□□□□ □□

□□ □□□□ □□□□□□ □□ □□ □□□□□□ Cargo□ □□□□ □□□□□□□ □□□□ □□ □□□□□ □□□□□□□ □ □□ □□□□ src/main.rs □□□□□□ □□□□□□ □□ .□□□□□□ □□ □□ □□□□□□□ □□□□□ □□□□□□. □□□□ □□□□□ □□□□□□□ □ □□□□□ □□□□□ □□□□□□. □□□□ □□ □□□□ □□□□□□□ □□□□□□□□□□ □□□□□ □□ □□ □□ □□□□□□ Cargo □□□□□□□□ □□□□□□□□□ □□□□ □□□□ □□ □□ □□ □□□□□□ Cargo □□□□□ □□ □□ □□□□□□□□ □□□□□ .□□□□□□□ □□□□ □□□□□ □□□□□□□ □ □□ □□□□□ □□□□□ □□□□ src/bin □□□□□ □□□□□□ (□□documentation □□□□□□□ □□□□□).

□□ □□□□□ □ □□□□□□□□ □□□□ □□ □□ □□□□□□ src/bin/server.rs□□ □ src/bin/client.rs □□□□ □□□□□. □□□□□□ □□□□ □□□□ □□ □□ □□□□□□□ □□ □□□□□□□□□ □□ □□ □□□□ □□□□□□ □□□□□ □□□□ □□□□ □□□□□□ □□□□□.

src/bin/server.rs:

```rust
use futures_util::sink::SinkExt;
use futures_util::stream::StreamExt;
use std::error::Error;
use std::net::SocketAddr;
use tokio::net::{TcpListener, TcpStream};
use tokio::sync::broadcast::{channel, Sender};
use tokio_websockets::{Message, ServerBuilder, WebSocketStream};

async fn handle_connection(
    addr: SocketAddr,
    mut ws_stream: WebSocketStream<TcpStream>,
    bcast_tx: Sender<String>,
) -> Result<(), Box<dyn Error + Send + Sync>> {

    // TODO: For a hint, see the description of the task below.
```

```
                                                                                                  {

                     } <<async fn main() -> Result<(), Box<dyn Error + Send + Sync
                                                        ;(let (bcast_tx, _) = channel(16

                    ;?let listener = TcpListener::bind("127.0.0.1:2000").await
                                                  ;("println!("listening on port 2000

                                                                                   } loop
                             ;?let (socket, addr) = listener.accept().await
                                          ;("{?:addr} □□ □□□□□ □□□□□□")!println
                                              ;()let bcast_tx = bcast_tx.clone
                                                      } tokio::spawn(async move
                            .Wrap the raw TCP stream into a websocket //
        ;?let ws_stream = ServerBuilder::new().accept(socket).await

                    handle_connection(addr, ws_stream, bcast_tx).await
                                                                            ;({
                                                                                  {
                                                                                      {
```

:*src/bin/client.rs*

```
                                                        ;use futures_util::stream::StreamExt
                                                                  ;use futures_util::SinkExt
                                                                              ;use http::Uri
                                             ;{use tokio::io::{AsyncBufReadExt, BufReader
                                          ;{use tokio_websockets::{ClientBuilder, Message

                               } <async fn main() -> Result<(), tokio_websockets::Error
                                                                = (_ ,let (mut ws_stream
       (("ClientBuilder::from_uri(Uri::from_static("ws://127.0.0.1:2000
                                                                     ()connect.
                                                                       ;?await.

                                                         ;()let stdin = tokio::io::stdin
                                         ;()let mut stdin = BufReader::new(stdin).lines


                      .TODO: For a hint, see the description of the task below //

                                                                                      {
```

□□□□□□ □□□□□□□□□

:□□ □□□□□□□□ □□ □□□□□ □□□□□□□□□□□ □□ □□□□□

```
cargo run --bin server
```

:□□ □□□□□□□ □□□ □

```
cargo run --bin client
```

Task

- [illegible] handle_connection [illegible] src/bin/server.rs [illegible].
  – [illegible]: [illegible] tokio::select! [illegible] task [illegible]. [illegible] task [illegible] (broadcast) [illegible].
- [illegible] src/bin/client.rs [illegible].
  – [illegible]: [illegible] tokio::select! [illegible] task [illegible]: ([illegible]) [illegible] ([illegible]) [illegible].
- [illegible]: [illegible].

67.3 [illegible]

Dining Philosophers --- Async

```
;use std::sync::Arc
;{use tokio::sync::mpsc::{self, Sender
;use tokio::sync::Mutex
;use tokio::time

;struct Fork

} struct Philosopher
,name: String
,<<left_fork: Arc<Mutex<Fork
,<<right_fork: Arc<Mutex<Fork
,<thoughts: Sender<String
{

} impl Philosopher
} (async fn think(&self
self.thoughts
((self.name& ,"![illegible] {} ![illegible]")!send(format.
await.
;()unwrap.
{

} (async fn eat(&self
Keep trying until we have both forks //
} let (_left_fork, _right_fork) = loop
...Pick up forks //
;()let left_fork = self.left_fork.try_lock
;()let right_fork = self.right_fork.try_lock
} let Ok(left_fork) = left_fork else
If we didn't get the left fork, drop the right fork if we //
.have it and let other tasks make progress //
;(drop(right_fork
;time::sleep(time::Duration::from_millis(1)).await
```

```
                                                        ;continue
                                                                   ;{
                               } let Ok(right_fork) = right_fork else
   If we didn't get the right fork, drop the left fork and let //
                                    .other tasks make progress //
                                                 ;(drop(left_fork
               ;time::sleep(time::Duration::from_millis(1)).await
                                                        ;continue
                                                                   ;{
                                       ;(break (left_fork, right_fork
                                                                       ;{

                                 ;(println!("{} is eating...", &self.name
                       ;time::sleep(time::Duration::from_millis(5)).await

                                            The locks are dropped here //
                                                                            {
                                                                                {

                                                   = [static PHILOSOPHERS: &[&str
                ;["Socrates", "Hypatia", "Plato", "Aristotle", "Pythagoras"]&

                                                                } ()async fn main
                                                              Create forks //
                                                      ;[]!let mut forks = vec
;((((PHILOSOPHERS.len()).for_each(|_| forks.push(Arc::new(Mutex::new(Fork..0)

                                                       Create philosophers //
                                               } = (let (philosophers, mut rx
                                           ;[]!let mut philosophers = vec
                                        ;(let (tx, rx) = mpsc::channel(10
                       } ()for (i, name) in PHILOSOPHERS.iter().enumerate
                               ;([let left_fork = Arc::clone(&forks[i
   ;([()let right_fork = Arc::clone(&forks[(i + 1) % PHILOSOPHERS.len
                                      } philosophers.push(Philosopher
                                          ,()name: name.to_string
                                                       ,left_fork
                                                      ,right_fork
                                            ,()thoughts: tx.clone
                                                                  ;({
                                                                        {
                                                       (philosophers, rx)
      tx is dropped here, so we don't need to explicitly drop it later //
                                                                           ;{

                                                   Make them think and eat //
                                                   } for phil in philosophers
                                                } tokio::spawn(async move
                                                    } for _ in 0..100
                                              ;phil.think().await
                                                ;phil.eat().await
```

```
{
;({
{

Output their thoughts //
} while let Some(thought) = rx.recv().await
;("{thought} :□□□□□ □□□□□ □□□□□ □□ □□□□□□ □□")!println
{
{
```

□□ □□□□□□□ □□□

:src/bin/server.rs

```
;use futures_util::sink::SinkExt
;use futures_util::stream::StreamExt
;use std::error::Error
;use std::net::SocketAddr
;{use tokio::net::{TcpListener, TcpStream
;{use tokio::sync::broadcast::{channel, Sender
;{use tokio_websockets::{Message, ServerBuilder, WebSocketStream

)async fn handle_connection
,addr: SocketAddr
,<mut ws_stream: WebSocketStream<TcpStream
,<bcast_tx: Sender<String
} <<Result<(), Box<dyn Error + Send + Sync <- (

ws_stream
((()to_string."□□□□□ □□□□□ □□□□□ □□ !□□□□□□ □□□ chat □□")send(Message::text.
;?await.
;()let mut bcast_rx = bcast_tx.subscribe

A continuous loop for concurrently performing two tasks: (1) receiving //
messages from `ws_stream` and broadcasting them, and (2) receiving //
.messages on `bcast_rx` and sending them to the client //
} loop
} !tokio::select
} <= ()incoming = ws_stream.next
} match incoming
} <= ((Some(Ok(msg
} ()if let Some(text) = msg.as_text
;("{?:println!("From client {addr:?} {text
;?(()bcast_tx.send(text.into
{
{
,(()Some(Err(err)) => return Err(err.into
,(())None => return Ok
{
{
} <= ()msg = bcast_rx.recv
```

```
             ;?ws_stream.send(Message::text(msg?)).await
                                                           {
                                                               {
                                                                   {
                                                                       {

           } <<async fn main() -> Result<(), Box<dyn Error + Send + Sync
                                    ;(let (bcast_tx, _) = channel(16

          ;?let listener = TcpListener::bind("127.0.0.1:2000").await
                                 ;("println!("listening on port 2000

                                                              } loop
                  ;?let (socket, addr) = listener.accept().await
                             ;("{?:addr} □□ □□□□ □□□□□")!println
                                ;()let bcast_tx = bcast_tx.clone
                                       } tokio::spawn(async move
                .Wrap the raw TCP stream into a websocket //
 ;?let ws_stream = ServerBuilder::new().accept(socket).await

          handle_connection(addr, ws_stream, bcast_tx).await
                                                             ;({
                                                                   {
                                                                       {
```

:src/bin/client.rs

```
                                    ;use futures_util::stream::StreamExt
                                              ;use futures_util::SinkExt
                                                          ;use http::Uri
                            ;{use tokio::io::{AsyncBufReadExt, BufReader
                         ;{use tokio_websockets::{ClientBuilder, Message

                } <async fn main() -> Result<(), tokio_websockets::Error
                                            = (_ ,let (mut ws_stream
(("ClientBuilder::from_uri(Uri::from_static("ws://127.0.0.1:2000
                                                  ()connect.
                                                    ;?await.

                                     ;()let stdin = tokio::io::stdin
                      ;()let mut stdin = BufReader::new(stdin).lines

 .Continuous loop for concurrently sending and receiving messages //
                                                              } loop
                                                } !tokio::select
                            } <= ()incoming = ws_stream.next
                                        } match incoming
                                  } <= ((Some(Ok(msg
             } ()if let Some(text) = msg.as_text
          ;(println!("From server: {}", text
                                               {
                                                  ,{
```

```
                           ,(()Some(Err(err)) => return Err(err.into
                                              ,(())None => return Ok
                                                                       {
                                                                           {
                                                } <= ()res = stdin.next_line
                                                             } match res
                                          ,(())Ok(None) => return Ok
Some(line)) => ws_stream.send(Message::text(line.to_string())).await
                                 ,(()Err(err) => return Err(err.into
                                                                       {
                                                                           {

                                                                               {
                                                                                   {
                                                                                       {
```

XV [illegible]

[illegible] [illegible]

[illegible] 68

[illegible]!

[illegible] [illegible] [illegible] [illegible] [illegible] Comprehensive Rust 🦀! [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] GitHub [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

69 [illegible]

[illegible]

[illegible] Rust [illegible].

- allocate:
  [illegible] (heap](memory-management/review.md] [illegible].
- argument:
  [illegible].
- [illegible]: [illegible] type [illegible].
- Bare-metal Rust:
  [illegible] Rust [illegible]. Bare-metal Rust [illegible].
- block:
  [Blocks](control-flow-basics/blocks-and-scopes.md) [illegible] *scope* [illegible].
- borrow:
  Borrowing [illegible].
- borrow checker:
  [illegible] Rust [illegible] (borrows) [illegible].
- brace:
  `{` and `}`. Also called *curly brace*, they delimit *blocks*.
- build:
  [illegible].
- call:
  [illegible].
- channel:
  [illegible] [thread [illegible]](concurrency/channels.md) [illegible].
- Comprehensive Rust 🦀:
  [illegible] Comprehensive Rust 🦀 [illegible].

- concurrency:
  [illegible] task [illegible] process [illegible].
- Concurrency [illegible] Rust [illegible] Concurrency in Rust [illegible].
- constant:
  [illegible].
- control flow:
  [illegible].
- crash:
  [illegible] (failure) [illegible].
- enumeration:
  [illegible].
- error:
  [illegible].
- error handling:
  [illegible].
- exercise:
  [illegible] task [illegible].
- function:
  [illegible].
- garbage collector:
  [illegible].
- generics:
  [illegible].
- immutable:
  [illegible].
- integration test:
  [illegible].
- keyword:
  [illegible].
- library:
  [illegible] routine [illegible].
- macro:
  [illegible] Rust [illegible] ! [illegible]. [illegible] `format!` [illegible] Rust [illegible].
- `main` function:
  [illegible] Rust [illegible] `main` [illegible].

- match:
  [illegible] Rust [illegible].
- memory leak:
  [illegible].
- method:
  [illegible] object [illegible] type [illegible] Rust.
- module:
  [illegible] Rust [illegible].
- move:
  [illegible] (ownership) [illegible] Rust.
- mutable:
  [illegible] Rust [illegible].
- ownership:
  [illegible] Rust [illegible].
- panic:
  [illegible] Rust [illegible].
- parameter:
  [illegible].
- pattern:
  [illegible] Rust [illegible].
- payload:
  [illegible].
- program:
  [illegible].
- [illegible].
- receiver:
  [illegible] Rust [illegible].
- reference counting:
  [illegible] object [illegible] object [illegible].
- return:
  [illegible] Rust [illegible].
- Rust:
  [illegible] safety [illegible] concurrency [illegible].

- Rust Fundamentals: [illegible].
- Rust [illegible] Android: [illegible] Rust in Android [illegible].
- Rust [illegible] Chromium: [illegible] Rust in Chromium [illegible].
- safe: [illegible] (ownership) [illegible] Rust [illegible] (borrowing) [illegible].
- scope: [illegible].
- standard library: [illegible] Rust [illegible].
- static: [illegible] Rust [illegible] `'static` [illegible].
- string: [illegible]. [illegible] Strings [illegible].
- struct: [illegible] Rust [illegible].
- test: [illegible] Rust [illegible].
- thread: [illegible].
- thread safety: [illegible] multithread [illegible].
- trait: [illegible] type [illegible] polymorphism [illegible] Rust [illegible].
- trait bound: An abstraction where you can require types to implement some traits of your interest.
- tuple: [illegible] data type [illegible]. [illegible] Tuple [illegible].
- type: [illegible] Rust [illegible].
- type inference: [illegible] Rust [illegible].
- undefined behavior: [illegible] Rust [illegible].

- union:
  [illegible] data type [illegible]
  [illegible].
- unit test:
  [illegible] Rust [illegible] unit test [illegible] integration test [illegible]
  [illegible]. Unit Tests [illegible].
- unit type:
  [illegible] tuple [illegible]
  [illegible].
- unsafe:
  [illegible] (subset) [illegible] Rust [illegible].
  [Unsafe Rust] (unsafe-rust/unsafe.md) [illegible].
- variable:
  [illegible] *scope* [illegible].

70 □□□

Rust □□□□□ □□□□□□ □□□□□

.□□□ □□□□□ □□□□□□ □□□□□□□ □□□□□ □□ □□ □□□□□□□ □ □□□□□□□□ □□□□□□ □□□□□□ Rust □□□□□□

□□□□□ □□□□□□□□

:□□□□□□□ □□□□□ □□□□□ □□□□ □□ □□ Rust □□□□□□□ □□□□ .□□□□ □□□□□□□ □□□□□□ □□□□□□□ Rust □□□□□□

- □□ □□□□□□ □ □□□□□□□ □□□□□ :(/Rust] (https://doc.rust-lang.org/book □□□□□□ □□□□□□□ □□□□□] □□□□□ □□□□□ □□□□□□ □□□□ □□□□□ □ □□□□□□ □□□□□ □□□□□□ □□□□□□□ □□ □□ □□□□□ □□□□ □□□ Rust □□□□□ .□□□□□□ □□□□□□□□□□
- □□ □□□□□□ □□□□□ □□□□□□□ □□ □□□□ □□ □□□□ □□ □□ Rust syntax □□□□□ □□ :Rust By Example □□ □□□□□□ □□□□□□ □□□□□□□□□ □□□□□ □□□□□□ □□□□□ .□□□□□□□□ □□□□□□ □□ □□ □□□□□□ □□□□□□□□□□ .□□□□□ □□□□□□ □□□□□□□ □□ □□ □□ □□□□□□ □□□□□□□ □□□□ □□
- □□□□□□□□□ □□□□□ □□□□□□□□ :(/Rust Standard Library](https://doc.rust-lang.org/std] .□□□□□□□ Rust □□□□□ □□□□□□□□□□
- .□□□□□□ □□□□□□ □□□ Rust □□□□□□ □□□□ □ □□□□□□ □□□ □□□□□□ □□□□□ :The Rust Reference

:Rust □□□□□ □□□□□ □□□ □□□□ □□□□□□□□ □□□□□□ □□□□□□ □□□□□□□□□□

- □ □□□□ □□□□pointer □□ □□□□ □□□□□ □□ □□□□□□□ □□□□□ □□ □□□□□□ unsafe Rust □□ :Rustonomicon .□□□□□□ □□□□□□ □□ (FFI) □□□□□ □□□□ □□□□□ □□ □□□□interface
- asynchronous) □□□□□□□□□ □□□□□□□□□□□□ □□□□ :Rust □□ □□□□□□□□□ □□□□□□ □□□□□□□ .□□□□ □□□□ □□□□□□ Rust □□□□□ □□□□□□ □□ □□ □□ □□□□□□ □□□□□ □□ □□□□□□ (programming
- □□ □□embedded device □□ Rust □□ □□□□□□□□ □□ □□□□□□□□□ :The Embedded Rust Book .□□□□□□ □□□□□ □□ □□□□□□ □□□□□□□□□□□ □□□□□

□□□□□□□□ □□□□□□□ □□□□□□

:Rust □□□□□ □□□□□ □□□□□□□□□ □ □□□□□□□□□ □□ □□□□□□ □□□□□□□

- □□□□□□ □□□□ □□□□□□□□□□□□□□ □□□□□□□ □□ □□ Rust □□□□□□□ :Learn Rust the Dangerous Way .□□□□□□ □□□□□ C
- -□□□□□□ □□ □□□□□□□□□□□□□□□ □□□□□□□ □□ □□ Rust □□ :Rust for Embedded C Programmers .□□□□□□ □□□□□ □□□□□□□□□ C □□□□□ □□ □□ □□□□□
- □□ □□□□□□ □□□□□□□□□□ □□□□ □□ □□ Rust □□□□□□□□ □□□□□ syntax □□ :Rust for Professionals .□□□□□□ □□□□□ Python □ C□ C++□ Java□ JavaScript □□□□□□ □□□□□ □□□□□□□□

- .□□□□□□ □□□□□ □□ Rust □□□□□□□□ □□ □□□□ □□□□□ □□□□□□ □□□□ □□ □□□□ :Rust on Exercism
- □□□□□□□□ □□ □ □□□□□ □□□□ □□ □□ □□□□□ □□□□□□□□□ □□ □□□□□□□□□ :Ferrous Teaching Material
  □□□□□ □□□□ async/wait □ WebAssembly □□□□□□ □□□□□□ □□□□□□□□ .□□□□□□ □□□□□ □□ Rust □□□□□
  .□□□ □□□□ □□□□□
- Advanced testing for Rust applications: a self-paced workshop that goes beyond Rust's
  built-in testing framework. It covers `googletest`, snapshot testing, mocking as well as
  .how to write your own custom test harness
- https://docs.microsoft.)[□□□□□□□□ Rust □□ □□ □□□□ □□□□ □□□□ □□□□□□] □ Beginner's Series to Rust
  □□□□□ □□□□□□□□□□□□□□ □□□□ □□ Rust □□□□□□□□ □□ :(/com/en-us/learn/paths/rust-first-steps
  □□□□□□ □□ □□□□ □□□□□□ □□ □□ □□□□□□□□□□ □□□□□ □ □□□□□□ □□ □□ □□□□□□□□□□ □□□□□ .□□□□□□□
  .□□□□□□ □□□□□ □□ □□□□□□ □□□□□□□□□□□ □ Rust
- Learn Rust With Entirely Too Linked Lists](https://rust-unofficial.github.io/too-many-]
  list □□□□□□ □□□□ □□□□ □□□□□□ □□□□□ □□ □Rust □□□□□□ □□□□□□□ □□□□□□□ □□□□□ □□□□□ :(/lists
  .structure

□□ □□□□□□ □□□□□□□□ □□□□□ □□ (/Little Book of Rust Books](https://lborb.github.io/book] □□□□□□
.□□□□□□□ Rust □□□□□

□□□ 71

□□□□□□□□□□

□□□□□□ □□ □□□□□□ □□ □□□□ □□□□□□□ □□ □□□□□□ □□□□□ □□□□□□□□ Rust □□□□□□ □□□□ □□□□. □□□□□ □□□□□□ □□□□□ □□□□□□ □□□□□ □□ □□□□□ [□□□□□ □□□□□□](other-resources.md) □□□□□□□ □□□□□.

□□□□□□□ Comprehensive Rust □□□□ □□□□□ Apache 2.0 □□□□□ □□□□□□□ □□□□□□ □□□□□ □□□□□□□ □□□□□□ □□ 'LICENSE' □□□□□□□ □□□□□.

Rust □□□□□□□□□ □□□□□

□□□□□ □□ □□□□□□□ □ □□□□□□□□ □□ [Rust by Example](https://doc.rust-lang.org/rust-by-example/) □□□□ □ □□□□□□□ □□□□□□□. □□□□□□ □□□□□ □□□□□□□□ □□ □□□□□ □□□□□□ license□ □□ □□□□□□□□□□ `/third_party/rust-by-example` □□□□□□□ □□□□□.

Rust □□ □□□□□□□□

□□□□□ □□□□□□□ □□ [Rust on Exercism](https://exercism.org/tracks/rust) □□□□ □ □□□□□□□ □□□-□□□□. □□□□□□ □□□□□ □□□□□□□□ □□ □□□□□ □□□□□□ license□ □□ □□□□□□□□□□ `third_party/rust-on-exercism/` □□□□□□□ □□□□□.

CXX

□□□□ Interoperability with C++ □□ □□□□□□□ □□ CXX □□□□□□□□□ □□□□□□. □□□□□□ □□□□□ □□□□□□□□□ □□ □□□□□ □□□□□□ license□ □□□□□□□□□□ `third_party/cxx/` □□□ □□□□□□□.