🦀 Comprehensive Rust

Martin Geisler

فهرست

به Comprehensive Rust خوش آمدید

این یک دوره رایگان Rust است که توسط تیم اندرویید در گوگل توسعه یافته است. این این دوره طیف کاملی از Rust را پوشش می‌دهد, از مباحث پایه تا مباحث پیشرفته مانند جنریک و مدیریت خطاها.

> آخرین نسخه از دوره را می‌توان در https://google.github.io/comprehensive-rust/ پیدا کنید. اگر از جای دیگری می‌خوانید, لطفا برای بروز رسانی‌ها منبع اصلی را نیز بررسی کنید.
>
> این دوره به زبان های دیگر موجود است. زبان مورد نظر خود را در گوشه سمت راست بالای صفحه انتخاب کنید یا صفحه ترجمه‌ها را برای فهرستی از تمام ترجمه‌های موجود را بررسی کنید.
>
> این دوره نیز به عنوان یک PDF در دسترس است.

هدف از این دوره آموزش Rust به شماست. ما فرض می کنیم شما چیزی از درباره Rust نمی دانید :

- درک جامعی از syntax و زبان Rust به شما می دهد.
- شما را قادر می سازد تا برنامه های موجود را تغییر دهید و برنامه های جدید را در Rust بنویسید.
- اصطلاحات رایج Rust را به شما یاد می دهد.

ما چهار روز اول دوره را اصول Rust می نامیم.

با تکیه بر این، از شما دعوت می شود تا به یک یا چند موضوع تخصصی بپردازید:

- Android: یک دوره نیم روزه استفاده از Rust برای توسعه پلت فرم اندروید (AOSP). این شامل قابلیت همکاری با C، C ++و Java است.
- Chromium: یک دوره نیم روزه در مورد استفاده از Rust در مرورگرهای مبتنی بر Chromium. این شامل قابلیت همکاری با C ++ و نحوه قرار دادن جعبه های(crates) شخص ثالث در کروم است.
- Bare-metal: یک کلاس تمام روز در مورد استفاده از Rust برای توسعهbare-metal (تعریف شده). هم میکروکنترلرها و هم پردازنده های برنامه پوشش داده شده اند.
- همروندی: یک کلاس یک روزه در مورد concurrency در زبان Rust است. ما در دو مورد concurrency کلاسیک (زمانبندی preemptively با استفاده از threadها و mutexها) و async/await concurrency (multitasking مشارکتی) با استفاده از futures را پوشش خواهیم داد.

اهداف خارج از این دوره

زبان Rust, یک زبان بزرگ است و ما نمی‌توانیم طی چند روز همه موارد را پوشش دهیم.چندتا از اهداف خارج از این دوره عبارتند از:

- برای آموزش چگونه‌گی توسعه Macro ها: لطفا Chapter 19.5 in the Rust Book و Rust by Example را بررسی کنید.

فرض می‌شود

این دوره فرض می‌کند که شما دانش برنامه‌نویسی دارید. Rust یک زبان اسات‌ای‌پی است و ما گاهی اوقات زبان Rust را با C و C++ مقایسه می کنیم تا رویکرده‌ای Rust را بهتر توضیح دهیم.

اگر می‌دانید چگونه به زبانی با دینامیک تایپ مانند پایتون یا جاوا اسکریپت برنامه‌نویسی کنید می‌توانید به خوبی این روش را دنبال کنید.

این یک نمونه از *speaker note* هست. ما از این‌ها استفاده خواهیم کرد تا اطلاعات بیشتری را ارائه دهیم.. این مورد می‌تواند شامل نکات کلیدی باشد که مدرس باید آن را پوشش دهد و همچنین پاسخ به سوالات رایجی که در کلاس مطرح می شود.

فصل 1

اجرای دوره

این صفحه برای مدرس دوره است.

این بخشی از پیشنهاد نحوه برگزاری دوره توسط گوگل به صورت درون سازمانی است.

ما معمولا کلاس‌ها را از ساعت ۱۰:۰۰ تا ۱۶:۰۰ برگزار می کنیم، با یک ساعت استراحت ناهار در میانه روز با این روی ۲.۵ ساعت برای کلاس صبح و ۲.۵ ساعت برای کلاس بعدازظهر باقی می‌گذارد. توجه داشته باشید که این فقط یک توصیه است: شما می‌توانید ۳ ساعت از چلسه صبح را صرف تمرین بیشتر برای افراد کنید. نکته منفی این کار این است که با چلسه طولانی تر افراد بعد از ۶ ساعت کلاس در بعد از ظهر خیلی خسته می‌شوند.

قبل از اجرای دوره، شما می‌خواهید:

1. با مطالب دوره آشنا شوید. ما یادداشت های سخنرانی را برای کمک به برجسته کردن نکات کلیدی گنجانده‌ایم (لطفا با مشارکت بیشتر در یادداشت‌های سخنران به ما کمک کنید!). هنگام ارائه، باید مطمئن شوید که یادداشت‌های سخنران را در یک پنجره پاپ‌آپ باز کنید (روی پیوند با یک فلش کوچک در کنار «یادداشت‌های سخنران» کلیک کنید). به این ترتیب یک صفحه نمایش تمیز برای ارائه به کلاس خواهید داشت.
2. در مورد زمان‌بندی دوره تصمیم بگیرید. از آنجایی که دوره حداقل سه روز کامل طول می‌کشد، توصیه می‌کنیم که دوره را در دو هفته برنامه‌ریزی کنید. شرکت کنندگان در دوره گفته‌اند که داشتن فاصله‌ای در دوره مفید است، زیرا به آن‌ها کمک می‌کند تا تمام اطلاعاتی را که به آن‌ها می‌دهیم پردازش کنند.
3. یک اتاق بزرگ برای حضور شرکت کنندگان پیدا کنید. ما کلاسی با گنجایش ۱۵ الی ۲۵ نفر را پیشنهاد می‌کنیم. افراد در این تعداد می‌توانند به راحتی سوال بپرسند --- همچنین مدرس وقت کافی برای پاسخ دادن به سوالات را نیز دارد. مطمئن شوید که اتاق مورد نظر نیز برای شما و دانشجویان دارد: شما همگی نیاز دارید که بتوانید بشنید و با لپ‌تاپ های خود کار کنید. به خصوص شما به عنوان مدرس کلی live-coding انجام خواهید داد پس صرفا یک میز بدون جا برای لپ‌تاپ برای شما مناسب نخواهد بود.
4. در روز برگزاری دوره، کمی زودتر به کلاس بیایید تا همه چیز را آماده کنید. ما توصیه می کنیم مستقیماً با استفاده از `mdbook serve`را در لپ‌تاپ خود اجرا کنید. (راهنمای نصب را ببینید). با این کار عملکرد بدون تاخیر در هنگام تغییر صفحات تضمین می شود. استفاده از لپ تاپ همچنین به شما امکان می دهد اشتباهات تایپی ا در صورت مشاهده شما یا شرکت کنندگان در دوره اصلاح کنید.
5. بگذارید افراد خودشان یا در گروه های کوچک تمرینات را حل کنند. مابه طور معمول ۳۰ الی ۴۵ دقیقه را برای تمرینات در صبح و بعدازظهر (از جمله زمان بررسی راه حل ها) صرف می‌کنیم. حتما از

افراد بخواهید که اگر گیر کرده‌اند یا چیزی وجود دارد که می‌توانید به آن‌ها کمک کنید. وقتی که می‌بینید چندین فرشکل کسانی دارند, خطاب به کل اتاق راه‌حل را پیشنهاد دهید؛ به عنوان مثال، با نشان دادن جایی که می‌توانند اطلاعات مربوطه را در کتابخانه استاندارد (standard library) پیدا کنند.

همه چیز خوب بود! در تدریس دوره موفق باشید! امیدوارم که برای شما هم به همان اندازه که برای ما لذت‌بخش بوده، لذت‌بخش باشد!

لطفاً بازخورد خود را ارائه دهید تا در آینده بتوانیم به بهبود دوره ادامه دهیم. ما دوست داریم بشنویم چه چیزی برای شما خوب بوده و چه چیزی می توانید بهتر شود. همینطور شما دانش‌آموزان نیز بسیار خوش آمدید برای ما بازخورد ارسال کنید !

1.1 مباحث دوره

این صفحه برای مدرس دوره است.

مبانی Rust

سه روز اول دوره را مبانی Rust تشکیل می‌دهند. این این سه روز با سرعت بالایی پیش می‌روند و ما موارد زیادی را پوشش می‌دهیم!

مباحث دوره:

- روز ۱ صبح (۲ ساعت و ۵ دقیقه با استراحت‌های کوتاه)

بخش	مدت زمان
خوش آمدید	۵ دقیقه
سلام, دنیا	۱۵ دقیقه
تایپ‌ها و متغیرها	۴۰ دقیقه
مبانی پایه کنترل جریان	۴۰ دقیقه

- روز ۱ بعد از ظهر (۲ ساعت و ۳۵ دقیقه،شامل وقت استراحت)

بخش	مدت زمان
تاپل ها و آرایه ها	۳۵ دقیقه
مراجع	۵۵ دقیقه
تایپ‌های تعریف شده توسط کاربر	۵۰ دقیقه

- روز ۲ صبح (۲ ساعت و ۱۰ دقیقه، شامل وقت استراحت)

بخش	مدت زمان
خوش آمدید	۳ دقیقه
تطبیق	۱ ساعت
متدها و تریت‌ها	۵۰ دقیقه

- روز ۲ بعد از ظهر (۴ ساعت و ۵ دقیقه، شامل وقت استراحت)

بخش	مدت زمان
Generics	۴۵ دقیقه
کتابخانه استاندارد تایپ‌ها	۱ ساعت
کتابخانه استاندارد Traits	۱ ساعت و ۱۰ دقیقه

- روز ۳ صبح (۲ ساعت و ۲۰ دقیقه، شامل وقت استراحت)

بخش	مدت زمان
خوش آمدید	۳ دقیقه
مدیریت حافظه	۱ ساعت
اشاره‌گره‌های هوشمند	۵۵ دقیقه

- روز ۳ بعدازظهر(۱ ساعت و ۵۵ دقیقه، شامل وقت استراحت)

بخش	مدت زمان
قرض‌گیری (Borrowing)	۵۵ دقیقه
طول‌عمر	۵۰ دقیقه

- روز ۴ صبح (۲ ساعت و ۴۰ دقیقه، شامل وقت استراحت)

بخش	مدت زمان
خوش آمدید	۳ دقیقه
Iterators	۴۵ دقیقه
ماژول‌ها	۴۰ دقیقه
تست‌کردن	۴۵ دقیقه

- روز ۴ بعدازظهر (۲ ساعت و ۱۰ دقیقه، شامل وقت استراحت)

بخش	مدت زمان
مدیریت خطا	۱ ساعت
Rust ناایمن	ساعت و ۵ دقیقه

عمیق تر شدن

علاوه بر کلاس 4 روزه Rust Fundamentals، موضوعات تخصصی تری را نیز پوشش می دهیم:

در اندروید **Rust**

در Rust در اندروید توی دوره یک دوره نیم روزه در مورد استفاده از Rust برای توسعه پلتفرم اندروید عمیق می‌شیم. این شامل قابلیت تعامل با C، C++ و جاوا می‌شود.

شما نیاز دارید که یک نسخه از مخزن ASOP بگیرید, همچنین یک نسخه از مخزن دوره بگیرید و روی همون ماشین در مسیر /src/android مخزن ASOP قرار دهید. با این کار طمین‌ان حاصل می‌کنید که سیستم build اندروید فایل های Android.bp را در /src/android می‌بینید.

اطمینان حاصل کنید که `adb sync` با شبیه‌ساز یا دستگاه واقعی شما کار می‌کند و همه نمونه‌های Android را با استفاده از `src/android/build_all.sh` از قبل بسازید. اسکریپت را بخوانید تا دستوراتی را که اجرا می‌کند ببینید و مطمئن شوید که وقتی آن‌ها را اجرا می‌کنید به درستی کار می‌کنند.

در اندروید **Rust**

Rust in Chromium یک دوره نیم روزه برای استفاده از Rust به عنوان بخشی از مرورگر Chromium است. این شامل استفاده از Rust در سیستم ساخت gn Chromium، آوردن کتابخانه‌های شخص ثالث ("crates") و قابلیت همکاری C++ است.

شما باید بتوانید Chromium را بسازید --- یک اشکال زدایی، ساخت کامپوننت برای سرعت [توصیه می شود] (chromium/setup.md/..) است، اما در ساختی کار می کند. مطمئن شوید که می‌توانید مرورگر Chromium را که ساخته‌اید اجرا کنید.

بر روی سخت افزار بدون سیستم عامل **Rust**

دوره آموزشی Rust بر روی سخت افزار بدون سیستم عامل یک دوره یک روزه با تمرکز بر استفاده ازRust برای توسعه بر روی سخت افزار بدون سیستم عامل (embedded) است. این دوره هم میکروکنترلرها و هم پردازشگر های با کارای خاص را پوشش می دهد.

برای قسمت میکروکنترلر، باید برد توسعه BBCmicro:bit v2 را خریداری کنید. همه باید تعدادی بسته را همانطور که در welcome page توضیح داده شده نصب کنند.

همزمانی در **Rust**

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

شما به یک crate جدید نیاز خواهید داشت و وابستگی ها دانلود و آماده استفاده باشند. سپس می‌توانید نمونه‌ها را در `src/main.rs` کپی/پیست کنید تا با آن‌ها آزمایش کنید:

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

مباحث دوره:

- صبح (۳ ساعت و ۲۰ دقیقه، شامل وقت استراحت)

بخش	مدت زمان
تردها	۳۰ دقیقه
کانال‌ها	۲۰ دقیقه
Send و Sync	۱۵ دقیقه
ناحیه‌های مشترک	۳۰ دقیقه
تمرین‌ها	۱ ساعت و ۱۰ دقیقه

- بعدازظهر (۳ ساعت و ۲۰ دقیقه، شامل وقت استراحت)

بخش	مدت زمان
مبانی Async	۳۰ دقیقه
کانال‌ها و Control Flow	۲۰ دقیقه
Pitfallها	۵۵ دقیقه
تمرین‌ها	۱ ساعت و ۱۰ دقیقه

تمرین

این دوره قرار است بسیار تعاملی باشد و توصیه می کنیم اجازه دهید حس کنجکاوی Rust را هدایت کنند!

1.2 نحوه صفحه کلید

چندین میانبر صفحه کلید مفید کلی در mdBook وجود دارد:

- Arrow-Left: به صفحه قبلی هدایت می‌کند.
- Arrow-Right: به صفحه بعدی هدایت می‌کند.
- Ctrl + Enter: Execute the code sample that has focus.
- s: نوار جستجو را فعال می‌کند.

1.3 ترجمه

این دوره توسط مجموعه ای از داوطلبان فوق العاده به زبان های دیگر ترجمه شده است:

- پرتغالی برزیلی توسط @joaovicmendes, @hugojacob, rastringer و @henrif75.
- Chinese (Simplified) by @suetfei, @wnghl, @anlunx, @kongy, @noahdragon, @superwhd, @SketchK, and @nodmp.
- چینی (سنتی) توسط @kuanhungchen, @mingyc, hueich, @victorhsieh و @johnathan79717.
- Japanese by @CoinEZ-JPN, @momotaro1105, @HidenoriKobayashi and @kantasv.
- کره ای توسط @jiyongp, keispace و @jooyunghan.
- اسپانیایی توسط @deavid.
- Ukrainian by @git-user-cpp, @yaremam and @reta.

از انتخابگر زبان در گوشه بالا سمت راست برای جابه‌جایی بین زبان‌ها استفاده کنید.

ترجمه‌های ناقص

تعداد زیادی ترجمه در حال انجام وجود دارد. ما به آخرین ترجمه های به روز شده پیوند می دهیم:

- Arabic by @younies
- بنگالی توسط @raselmandol.
- Farsi by @Alix1383, @DannyRavi, @hamidrezakp, @javad-jafari and @moaminsharifi.
- فرانسوی توسط @KookaS و @vcaen.
- آلمانی توسط @Throvn و @ronaldfw.
- ایتالیایی توسط @Throvn و @ronaldfw.

فهرست کامل ترجمه‌ها با وضعیت فعلی‌شان نیز در آخرین به‌روزرسانی یا همگام‌سازی شده با آخرین نسخه دوره.

اگر می‌خواهید به این کار کمک کنید، لطفاً دست‌ورالعمل‌های ما را برای چگونگی ادامه کار ببینید. ترجمه‌ها در issue tracker هماهنگ و کنترل می شوند.

فصل 2

استفاده از cargo

وقتی شروع به خواندن درباره Rust می کنید، خیلی سریع با Cargo ، ابزار استانداردی که در اکوسیستم Rust برای ساخت و اجرای برنامه های Rust استفاده می شود، آشنا خواهید شد. در اینجا میخواهیم یک مرور مختصر از مورد کارگو و نحوه انطباق آن با اکوسیستم Rust و برنامه‌های آن را در این آموزش ارائه دهیم.

راه‌اندازی

لطفا دستورالعمل را دنبال کنید **https://rustup.rs**.

این کار به شما امکان استفاده از ابزار ساخت `cargo` (Cargo) و کامپایلر `rustc` (Rust) را می دهد. شما همچنین `rustup` را دریافت خواهید کرد، یک ابزار خط فرمان(CLI) که می توانید از آن برای نصب نسخه های مختلف کامپایلر استفاده کنید.

پس از نصب Rust، باید ویرایشگر یا IDE خود را برای کار با Rust پیکربندی کنید. اکثر ویرایشگرها این کار را با ارتباط گرفتن با rust-analyzer انجام میدهند، که قابلیت تکمیل خودکار و پرش به تعریف را برای Vim/Neovim، Emacs، VS Code و بسیاری دیگر فراهم می کند. همچنین یک IDE مختص به نام RustRover در دسترس است.

- در دبیان/اوبونتو، میتوانید Cargo و Rust و Rust formatter را نیز از طریق `apt` نصب کنید. با این حال، این به شما یک نسخه قدیمی Rust را جهت نصب می دهد و ممکن است منجر به رفتار غیرمنتظره برنامه شود. command مورد نظر این خواهد بود:

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

- در macOS، میتوانید از Homebrew برای نصب Rust استفاده کنید، اما ممکن است یک نسخه قدیمی باشد. بنابراین توصیه می شود Rust را از سایت رسمی نصب کنید.

2.1 اکوسیستم Rust

اکوسیستم Rust از تعدادی ابزار تشکیل شده است که مهمترین آنها عبارتند از:

- `rustc`: کامپایلر Rust که فایل‌های `.rs` را به باینری و سایر فرمت‌های میانی تبدیل می‌کند.
- `cargo`: مدیر وابستگی Rust و build tool آن است. Cargo می داند که چگونه وابستگی ها را که معمولاً در https://crates.io میزبانی می شوند دانلود کند و هنگام ساخت پروژه آنها را به

`rustc` مترجم می‌کند. Cargo همچنین دارای یک دستگاه تست داخلی است که برای اجرای unit test استفاده می شود.

- 'rustup': نصب کننده و به روز رسانی rustchain ابزار. این ابزار برای نصب و به روز رسانی "rustc" و "cargo" در هنگام انتشار نسخه های جدید Rust استفاده می شود. علاوه بر این، "rustup" همچنین می تواند اسناد را برای کتابخانه استاندارد دانلود کند. شما می توانید چندین نسخه از Rust را در یک زمان نصب کنید و "rustup" به شما اجازه می دهد تا در صورت نیاز بین انها تغییر دهید.

نکات کلیدی:

- Rust یک برنامه سریع برای انتشار نسخه‌های جدید دارد و هر شش هفته یک نسخه جدید مترشح می شود. نسخه‌های جدید سازگاری با نسخه‌های قدیمی را حفظ می‌کنند --- به علاوه قابلیت‌های جدید را فعال می‌کنند.
- سه کانال انتشار وجود دارد: "beta"، "stable"، و "nightly".
- ویژگی های جدید در "nightly" آزمایش می شوند ، "beta" چیزی است که هر شش هفته "stable" می شود.
- همچنین می‌توان وابستگی‌ها را از registries، پوشه‌ها و git و موارد دیگر برطرف کرد.
- Rust همچنین نسخه editions دارد: نسخه فعلی Rust 2021 است. نسخه های قبلی Rust 2015 و Rust 2018 بودند.
  - نسخه ها مجاز به ایجاد تغییرات backwards incompatible در زبان هستند.
  - برای جلوگیری از breaking code، نسخه‌ها اختیاری انتخاب می‌شوند که: شما نسخه مورد نظر برای crate خود از طریق فایل `Cargo.toml` انتخاب می‌کنید.
  - برای جلوگیری از شکاف در اکوسیستم، کامپایلرهای Rust می توانند کدهای نوشته شده برای نسخه های مختلف را ترکیب کنند.
  - لازم به ذکر است که استفاده از کامپایلر به طور مستقیم(rustc) و نه از طریق `cargo` بسیار غیرمعمول است (اکثر کاربران هرگز این کار را نمی کنند).
  - ممکن است لازم به ذکر باشد که Cargo خود یک ابزار بسیار قدرتمند و جامع است. این است که قادر به بسیاری از ویژگی های پیشرفته از جمله اما نه محدود به:
    * ساختار پروژه/بسته
    * workspaces
    * وابستگی های Dev و وابستگی‌های Runtime Management/Caching
    * build scripting
    * global installation
    * همچنین با command plugin فرعی (مانند cargo clippy) قابلیت توسعه است.
  - در official Cargo Book بیشتر بخوانید.

2.2 نمونه کد در این آموزش

برای این آموزش، بیشتر زبان Rust را از طریق مثال های که می توان از طریق مرورگر شما اجرا کرد، بررسی می کنیم. این کار راه اندازی را بسیار ساده تر می کند و تجربه ای ثابت را برای همه تضمین می کند.

نصب Cargo همچنان پیشنهاد می شود: چونکه انجام تمرینات را برای شما آسان تر می کند. در روز آخر، تمرین بزرگتری را انجام خواهیم داد که به شما نشان می دهد چگونه با وابستگی ها کار کنید و برای این کار شما به Cargo نیاز دارید.

بل‌وک ه‌ای کد در این دوره کاملاً تعاملی(interactive) هستند:

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

.You can use Ctrl + Enter to execute the code when focus is in the text box

اکثر نمونه ه‌ای کد مانند نشان داده شده در بالا قابل ویرایش هستند. چند نمونه کد به دلایل مختلف قابل ویرایش نیستند:

- همین‌طورembedded playground نمی توانند unit tests را اجرا کنند. کد را کپی کنید و آن را در Playground واقعی باز کنید تا unit tests د را نشان دهید.
- در واقع embedded playgrounds در لحظه ای که از صفحه دور می شوید حالت پای‌دار خود را از دست می دهند! به همین دلیل است که دانش آموزان باید تمرینات را با استفاده از local Rust ی ا از طریق installation Playground حل کنند.

2.3 اجرای کد به صورت لوکال با Cargo

اگر می خواهید کد را روی سیستم خود آزمایش کنید، ابتدا باید Rust را نصب کنید. این کار را با دنبال کردن instructions in the Rust Book انجام دهید. این باید به شما یک `rustc` و `cargo` کاربردی بدهد. در زمان نگارش، آخرین نسخه پای‌دار Rust دارای این versionها است:

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

شما همچنین می توانید از هر نسخه بعدی استفاده کنید، زیرا Rust سازگاری با نسخه های قبلی را حفظ می‌کند.

با این کار، این مراحل را دنبال کنید تا از یکی از مثال‌های این آموزش، یک باینری Rust بسازید:

1. روی دکمه "کپی در کلیپ بورد" در نمونه ای که می خواهید کپی کنید؛ کلیک کنید.
2. از `cargo new exercise` برای ایجاد دایرکتوری `/excerise` جدید برای کد خود استفاده کنید:

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

3. به `/exercise` بروید و از `cargo run` برای ساخت و اجرای باینری خود استفاده کنید:

```
$ 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. کد صفحه دیگر را در `src/main.rs` با کد خود جایگزین کنید. برای مثال، با استفاده از مثال در صفحه قبل، `src/main.rs` را شبیه به آن کنید.

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

5. برای ساختن و اجرای باینری به روز شده خود از `cargo run` استفاده کنید:

```
$ 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. از `cargo check` برای بررسی سریع پروژه خود برای خطاها استفاده کنید، از `cargo build` برای کامپایل، بدون اجرای آن استفاده کنید. خروجی را در `/target/debug` برای ساخت اشکال زدایی معمولی خواهید یافت. برای تولید نسخه بهینه سازی شده در `/target/release` از `cargo build --release` استفاده کنید.
7. با ویرایش `Cargo.toml` میتوانید وابستگیهایی برای پروژه خود اضافه کنید. هنگامی که دستورات `cargo` را اجرا می کنید، به طور خودکار وابستگی های مورد نیاز را برای شما دانلود و کامپایل میکنند.

سعی کنید شرکت کنندگان کلاس را تشویق کنید تا Cargo را نصب کنند و از یک ویرایشگر محلی استفاده کنند. این زندگی آنها را آسان تر می کند زیرا آنها یک محیط توسعه عادی خواهند داشت.

بخش I

روز ۱: صبح

فصل 3

به روز اول خوش آمدید

این اولین روز از مبانی Rust است. ما امروز بخش‌های فابل توجه‌ای را پوشش خواهیم داد:

- سینتکس‌های مقدماتی: متغیرها, تایپ‌های عددی و تایپ‌های مرکب, enums, structs, مراجع, توابع, و متدها.
- Types and type inference.
- ساختارهای جریان کنترل: حلقه ها، شرط ها و غیره.
- تایپ های تعریف شده توسط کاربر: ساختارها و enums.
- تطابق الگو: تجزیه و تحلیل enums, structs و آرایه‌ها.

برنامه زمانی

با احتساب 10 دقیقه استراحت، این جلسه باید حدود 2 ساعت و 5 دقیقه طول بکشد. آن شامل:

بخش	مدت زمان
خوش آمدید	۵ دقیقه
سلام, دنیا	۱۵ دقیقه
تایپ‌ها و مقادیر	۴۰ دقیقه
مبانی پایه کنترل جریان	۴۰ دقیقه

This slide should take about 5 minutes.

لطفا به دانشجویان یادآوری کنید:

- آن‌ها باید سؤالاتی را که به دست آوردند بپرسند، آن‌ها را تا انتها ذخیره نکنند.
- کلاس قرار است تعاملی باشد و بحث‌ها بسیار مورد تشویق قرار می گیرند!
  - به‌عنوان یک مربی، باید سعی کنید بحث‌ها را مرتبط نگه دارید، به عنوان مثال، بحث‌های مرتبط با نحوه انجام کارها توسط Rust در مقابل برخی زبان‌های دیگر را حفظ کنید. پیدا کردن تعادل مناسب می تواند سخت باشد، اما در مورد اجازه دادن به بحث اشتباه کنید، زیرا آن‌ها بیشتر از ارتباط یک طرفه افراد را درگیر می کنند.
- احتمالاً سؤالات به این معنی است که ما در مورد چیزهایی قبل از اسلایدهای صحبت می کنیم.
  - این اصلاً اشکالی ندارد! تکرار بخش مهمی از یادگیری است. به یاد داشته باشید که اسلایدها فقط یک پشتیبان هستند و شما می توانید هر طور که دوست دارید از آن‌ها صرف نظر کنید.

ای‌ده روز اول ن‌شان دادن چی‌زه‌ای "پای‌ه" در Rust اس‌ت که با‌ی‌د در زبان ه‌ای دی‌گر م‌شاب‌ه‌ت ه‌ای ف‌وری داش‌ته با‌ش‌ن‌د. ق‌س‌م‌ت ه‌ای پی‌ش‌رف‌ته تر Rust در روزه‌ای ب‌ع‌د ع‌رض‌ه م‌ی ش‌ون‌د.

اگر ای‌ن را در ک‌ل‌اس درس ت‌دری‌س م‌ی ک‌ن‌ی‌د، ای‌ن‌جا م‌کان خ‌وب‌ی ب‌رای م‌رور ب‌رن‌ام‌ه اس‌ت. ت‌وج‌ه داش‌ته با‌ش‌ی‌د که در پ‌ای‌ان ه‌ر ب‌خ‌ش ی‌ک ت‌م‌ری‌ن و س‌پ‌س ی‌ک اس‌ت‌راح‌ت وج‌ود دارد. ب‌رای پ‌وش‌ان‌دن م‌ح‌ل‌ول ت‌م‌ری‌ن ب‌ع‌د از اس‌ت‌راح‌ت ب‌رن‌ام‌ه ری‌زی ک‌ن‌ی‌د. زم‌ان ه‌ای ذک‌ر ش‌ده در ای‌ن‌جا ی‌ک پی‌ش‌ن‌ه‌اد ب‌رای ح‌ف‌ظ دوره در ب‌رن‌ام‌ه اس‌ت. ب‌ا خ‌ی‌ال راح‌ت ان‌ع‌ط‌اف پ‌ذی‌ر با‌ش‌ی‌د و در ص‌ورت ل‌زوم ت‌ن‌ظی‌م ک‌ن‌ی‌د!

فصل 4

سلام, دنیا

این بخش ۱۵ دقیقه زمان می برد. این بخش شامل:

اسلاید	مدت زمان
زبان Rust چیست؟	۱۰ دقیقه
مزیت‌های زبان Rust	۳ دقیقه
Playground	۲ دقیقه

4.1 زبان Rust چیست؟

Rust یک زبان برنامه‌نویسی جدید است که نسخه 1.0 آن در سال 2015 منتشر شد:

- زبان Rust, یک زبان کامپایل شده ایستا است که نقشی مشابه C++ دارد
  - `rustc` از LLVM به عنوان بک‌اند خود استفاده می‌کند.
- راست از بسیاری از بسته‌ها و معماری‌ها پشتیبانی می کند :
  - x86, ARM, WebAssembly, ...
  - Linux, Mac, Windows, ...
- زبان Rust برای طیف گسترده‌ای از دستگاه‌ها استفاده می‌شود:
  - میان‌افزار (firmware) و بوت‌لودرها (boot loaders)
  - نمایشگره‌ای هوشمند,
  - تلفن‌های همراه,
  - رایانه‌های رومیزی,
  - سرورها.

This slide should take about 10 minutes.

Rust در همان حوزه C++ قرار می‌گیرد:

- انعطاف پذیری بالا.
- سطح کنترل بالا.
- می‌تواند به دستگاه‌های بسیار محدود منابع کروکنترلرها متصل شود.
- فاقد زمان اجرا (runtime) یا جمع‌آوری زباله (garbage collection) است.
- بر قابلیت اطمینان و ایمنی بدون قربانی کردن عملکرد تمرکز دارد.

4.2 مزیت‌های زبان Rust

برخی از نقاط قوت مرتبط به فرد زبان Rust:

- ایمنی حافظه زمان کامپایل - کل کلاس های باگ حافظه در زمان کامپایل جلوگیری می شود
  - هیچ متغیر مقداردهی نشده‌ای (uninitialized) وجود ندارد.
  - هیچ آزادسازی دوباره‌ای وجود ندارد.
  - هیچ استفاده‌ای پس از آزادسازی وجود ندارد.
  - هیچ اشاره‌گر NULL وجود ندارد.
  - هیچ موتکس قفل شده‌ای وجود ندارد.
  - هیچ وضعیت رقابتی (data races) بین رشته‌ها وجود ندارد.
  - تکرارکننده‌ها (iterators) هیچگاه نامعتبر نمی‌شوند..
- بدون رفتار زمان اجرا - نشده فعالیت اجرا که کاری دست‌ور Rust انجام می دهد مگر خصوصاً با علامت بیان می‌شود.
  - دسترسی به آرایه‌ها با بررسی محدوده چک می‌شود.
  - سرریز عدد صحیح فعالیت شده است (پیش‌فرض wrap-around).
- ویژگی های زبان مدرن - به اندازه زبان های سطح بالاتر گوگل و ارگونومیک است
  - Enumها و تطابق الگوها.
  - جنریک‌ها.
  - FFI بدون سربار.
  - انتزاع‌های بدون هزینه.
  - خطاهای کامپایل تسلیم.
  - مدیریت‌پذیری درون-ساختی.
  - پشتیبانی درون-ساختی از تست نویسی.
  - پشتیبانی عالی از LSP.

This slide should take about 3 minutes.

وقت زیادی را اینجا صرف نکنید. تمام این نکات باید با عمق بیشتری پوشش داده شود.

حتما از کلاس بپرسید که با چه زبان‌هایی تجربه دارند. بسته به پاسخ، می توانید ویژگی‌های مختلف Rust را برجسته کنید::

- تجربه با C یا C++ : زبان Rust با استفاده از بررسی کننده قرضگیری (اشاره به بحث قرض گرفتن یا borrow) ، یک سری کامل از خطاهای زمان اجرا را از بین می‌برد .t. عملکردی مانند C و C++ را دارد اما مشکلات عدم ایمنی حافظه را ندارد. علاوه بر این، شما یک زبان مدرن با ساختارهای مانند تطابق الگو و مدیریت پذیرفته داخلی دریافت می‌کنید.
- تجربه با Java، Go، Python، JavaScript...: شما همان ایمنی حافظه (memory safety) را مانند آن زبان‌ها دریافت می‌کنید، به علاوه یک احساس زبان سطح بالا شبیه را تجربه خواهید کرد. علاوه بر این، شما عملکرد سریع و قابل پیش‌بینی مانند C و C++ (بدون garbage collector) و همچنین دسترسی به سخت‌افزار سطح پایین (در صورت نیاز) دریافت می‌کنید.

4.3 Playground

Rust Playground یک راه آسان برای اجرای برنامه های Rust کوتاه ارائه می دهد و پایه ای برای مثال ها و تمرین های این دوره است. برنامه "Hello-world" را که با آن شروع می شود اجرا کنید. دارای چند ویژگی مفید است:

- در زیر "ابزارها"، از گزینه "rustfmt" برای قالب بندی کد خود به روش "استاندارد" استفاده کنید.

- Rust دارای دو "نمایه" اصلی برای تولید کد است: Debug (بررسی های زمان اجرا اضافی، به‌ینه سازی کمتر) و Release (بررسی های زمان اجرا کمتر، به‌ینه سازی زیاد). این‌ها در قسمت «اشکال-زدایی» در بالا قابل دسترسی هستند.
- اگر علاقه مند هستید، از "ASM" در زیر "..." برای دیدن کد اسمبلی تولید شده استفاده کنید.

This slide should take about 2 minutes.

هنگامی که دانش آموزان به سمت استراحت می روند، آن‌ها را تشویق کنید تا playground را باز کنند و کمی تجربه کنند. آن‌ها را تشویق کنید که برگه را باز نگه دارند و در بقیه دوره چیزهایی را امتحان کنند. این به ویژه برای دانش‌آموزان پیشرفته که می‌خواهند درباره به‌ینه‌سازی‌های Rust یا مونتاژ تولید شده بیشتر بدانند مفید است.

فصل 5

تایپ‌ها و مقادیر

این بخش باید حدود ۴۰ دقیقه طول بکشد. آن شامل:

اسلاید	مدت زمان
سلام, دنیا	۵ دقیقه
متغیرها	۵ دقیقه
مقادیر	۵ دقیقه
عملگرهای ریاضی	۳ دقیقه
تعیین تایپ ضمنی	۳ دقیقه
تمرین: دنباله فیبوناچی	۱۵ دقیقه

5.1 سلام, دنیا

بیایید به ساده ترین برنامه Rust ممکن یعنی یک برنامه Hello World کلاسیک بپردازیم:

```
fn main() {
    println!("سلام 🌍!");
}
```

آنچه شما می‌بینید:

- توابع با `fn` معرفی می‌شوند.
- بلوک‌ها با پرانتزهای باز و بسته مانند C و ++C محدود می‌شوند.
- تابع `main` نقطه ورود برنامه است.
- زبان Rust دارای ماکروهای hygienic است، `println!` یک نمونه از این است.
- رشته‌های Rust دارای انکودینگ UTF-8 هستند و می‌توانند شامل هر کاراکتر یونیکد باشند.

This slide should take about 5 minutes.

این اسلاید سعی می کند دانشجویان با کد Rust احساس راحتی کنند. آن‌ها در سه روز آینده خیلی از این کدها خواهند دید، بنابراین با یک چیز آشنا شروع می کنیم..

نکات کلیدی:

- زبان Rust, زبان بسیار شبیه به سایر زبان‌های خانواده Java/C++/C است.یک زبان امری است (imperative) و سعی نمی‌کند چیزی را مگر اینکه کاملاً ضروری باشد، دوباره اختراع کند.

- زبان Rust, یک زبان مدرن با پشتیبانی کامل از چیزهای مانند دیوید است.
- Rust از ماکروها برای موقعیت‌های استفاده می‌کند که می‌خواهید تعداد متغیری از آرگومان‌ها داشته باشید (بدون اورلودینگ تابع).
- «هاجنیک» (hygienic) بودن ماکرو به این معنی است که آن‌ها به طور تصادفی شناسه‌ها را از محدوده‌ای که در آن استفاده می‌شوند، ذخیره نمی‌کنند. ماکروهای Rust در واقع فقط [تا حدی هاجنیک](https://veykril.github.io/tlborm/decl-macros/minutiae/hygiene.html) هستند.
- زبان Rust, یک زبان چند پارادایمی است. به عنوان مثال، دارای ویژگی‌های قدرتمند برنامه نویسی شیگرا است و در حالی که یک زبان فانکشنال(functional) نیست، شامل طیف وسیعی از مفاهیم فانکشنال است.

5.2 متغیرها

زبان Rust از طریق سیستم تایپ استاتیک, ایمنی نوع را فراهم می‌کند. به صورت پیش‌فرض تعریف متغیر ها از نوع «غیر قابل تغییر» (immutable) است:

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

This slide should take about 5 minutes.

- برای نشان دادن اینکه متغیرها به طور پیش‌فرض تغییرناپذیر هستند، کامنت "x = 20" را حذف کنید. برای اجازه دادن به تغییرات، کلمه کلیدی «mut» را اضافه کنید.
- «i32» در اینجا نوع متغیر است. این باید در زمان کامپایل شناخته شود، اما استنتاج نوع (که بعداً پوشش داده می شود) به برنامه نویس اجازه می دهد تا در بسیاری از موارد آن را حذف کند.

5.3 ثابت‌ها

در اینجا چند نوع پایه داخلی و نحو برای ثابت‌ها تحت الفظی هر نوع آورده شده است.

انواع	ثابت‌ها	
اعداد صحیح علامت-دار	i8, i16, i32, i64, i128, isize	-10, 0, 1_000, 123_i64
اعداد صحیح مثبت	u8, u16, u32, u64, u128, usize	0, 123, 10_u16
اعداد با ممیز شناور	f32, f64	3.14, -10.0e20, 2_f32

انواع	مقادیر ثابت
مقادیر عددی یونیکد	`char`	`'∞'`, `'a'`, `'α'`
بولین-ها	`bool`	`true`, `false`

اندازه تایپ‌ها به شرح زیر است:

- `uN`, `iN` و `fN` به اندازه N حافظه اشغال می‌کنند.
- `isize` و `usize` به اندازه یک اشاره‌گر حافظه اشغال می‌کنند.
- `char` به اندازه 32 بیت حافظه اشغال می‌کنند.
- `bool` به اندازه 8 بیت حافظه اشغال می‌کنند.

This slide should take about 5 minutes.

موارد اندکی وجود دارند که در بالا نشان داده نشده است:

- می‌توان همه خطوط زیرین _ را در اعداد حذف کرد، آن‌ها فقط برای خوانایی هستند. «`1_000`» می‌تواند به صورت `1000` (یا `10_00`) نوشته شود و `123_i64` می‌تواند به صورت `123i64` نوشته شود».

5.4 عملگرهای ریاضی

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

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

This slide should take about 3 minutes.

این اولین بار است که تابعی غیر از "main" می‌بینیم، اما معنی آن باید واضح باشد: سه عدد صحیح می‌گیرد و یک عدد صحیح برمی‌گرداند. توابع با جزئیات بیشتر پوشش داده خواهد شد.

حساب‌های بسیار شبیه به زبان‌های دیگر است، با تقدم مشابه.

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» را به «i16» تغییر دهید تا یک سرریز عدد صحیح را ببینید، که در یک ساخت اشکال‌زدایی وحشت می‌کند (بررسی می‌شود) و در یک نسخه انتشار می‌پیچد. گزینه‌های دیگری مانند سرریز، اشباع و محل وجود دارد. این‌ها با نحو متدقابل دسترسی هستند، به عنوان مثال، `(a * b).saturating_add(b)` `(* c).saturating_add(c *a`.

در واقع، کامپایلر سرریز عبارات ثابت را تشخیص می دهد، به همین دلیل است که مثال به یک تابع جداگانه نیاز دارد.

5.5 تعریف پایتون

زبان Rust برای تعریف نوع بازگشته از آن نگاه می‌کند:

```
} (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 //
{
```

.This slide should take about 3 minutes

این اسلایدها نشان می‌دهد که چگونه کامپایلر Rust با توجه به اعلان‌ها و استفاده‌ای متغیر، انواع را استنتاج می‌کند.

بسیار مهم است که تاکید کنیم متغیرهایی که به این صورت تعریف می‌شوند از «نوع داده پیوی any» می‌توانند که بتوانند هر نوعی باشند. وقتی که ما از تعیین تایپ ضمنی استفاده می‌کنیم در واقع مشابه زمانی هست که نوع داده را به صورت صریح اعلام می‌کنیم و کد های ماشین آن‌ها دقیقا یکسان هستند. فقط با استفاده از تعیین تایپ ضمنی می‌توانید کد ها رو به صورت مختصرتر بنویسیم.

هنگامی که هیچ چیز نوع یک عدد صحیح را محدود نمی کند، Rust به طور پیش فرض روی «i32» قرار می گیرد. گاهی اوقات در پیام های خطا به صورت «{integer}» نشان داده می شود. به طور مشابه، تایپ ممیز شناور پیش‌فرض «f64» است.

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

5.6 تمرین: دنباله فیبوناچی

دنباله فیبوناچی با «[0،1]» شروع می شود. برای n>1، عدد فیبوناچی n به صورت بازگشتی به عنوان مجموع اعداد فیبوناچی n-1 و n-2 محاسبه می شود.

یک تابع fib(n) بنویسید که عدد فیبوناچی n را محاسبه کند. چه زمانی این عملکرد panic می شود؟

```
} fn fib(n: u32) -> u32
} if n < 2
//حالت پایه
("این را پیاده‌سازی کن")!todo
} else {
// حالت بازگشتی
("این را پیاده‌سازی کن")!todo
{
```

```
                                        {

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

5.6.1 لحاد

```
                        } 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
                                        {
```

فصل 6

مبانی برنامه‌نویسی

این بخش حدود ۴۰ دقیقه طول می‌کشد. آن شامل:

اسلاید	مدت زمان
عبارات if	۴ دقیقه
حلقه‌ها	۵ دقیقه
break و continue	۴ دقیقه
بلوک‌ها و محدوده‌ها	۵ دقیقه
توابع	۳ دقیقه
اکرام‌ها	۲ دقیقه
تمرین: دنباله Collatz	۱۵ دقیقه

6.1 عبارات `if`

شما عبارت `if` را به هر دیگر زبان‌های برنامه‌نویسی می‌بینید:

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

در کنار این موضوع، می‌توانید از if به عنوان یک عبارت با قابلیت بازگشت مقدار هم استفاده کنید. آخرین عبارت توی هر بلاک if اون مقدار و نوع بازگشتی است:

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

.This slide should take about 4 minutes

از آن‌جایی که `if` یک عبارت است و باید نوع خاصی داشته باشد، هر دو بلاک (`if` و `else`) باید از نوع یکسانی را باز گردانند. در نظر بگیرید که اگر بعد از `x / 2` در مثال دوم `;` اضافه کنید، چه اتفاقی می افتد.

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 حلقه‌ها

سه کلمه کلیدی حلقه ای در Rust وجود دارد: "loop"، "while" و "for":

حلقه‌های `while`

کلمه‌کلیدی`while` بسیار شبیه به سایر زبان‌ها عمل می‌کند.

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

6.2.1 `for`

حلقه for در محدوده‌ای از مقادیر یا موارد موجود در یک مجموعه تکرار می‌شود:

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

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

- حلقه‌های «for» در اکثر مواقع برای «تکرار کردن» روی مجموعه‌ها استفاده می‌شوند. Iterators بعداً با جزئیات بیشتر مورد بحث قرار خواهند گرفت.
- توجه داشته باشید که حلقه `for` فقط تا 4 تکرار می شود. نحو `1..=5` را برای یک محدوده فراگیر نشان دهید.

6.2.2 `loop`

loop برای همیشه، تا زمانی که یک «break» ایجاد شود، حلقه می‌شود.

```
fn main() {
    let mut i = 0;
```

```
    loop {
        i += 1;
        println!("{i}");
        if i > 100 {
            break;
        }
    }
}
```

6.3 break و continue

اگر می‌خواهید بلافاصله تکرار بعدی را شروع کنید، از continue استفاده کنید.

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);
    }
}
```

This slide and its sub-slides should take about 4 minutes.

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).

6.3.1 برچسب‌ها

کلیدواژه‌های continue و break هر دو می‌توانند به صورت اختیاری یک آرگومان برچسب (label) بگیرند که می‌توان برای خروج از حلقه‌های تو در تو استفاده می‌کرد:

```
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 بلوک‌ها و محدوده‌ها

بلوک‌ها

یک بلوک در Rust حاوی دنباله‌ای از عبارات است که با پرانتزهای «{}» محصور شده است. هر بلوک دارای یک مقدار و یک نوع است که آخرین عبارت بلوک است:

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

اگر آخرین عبارت با ; پایان یابد، مقدار و نوع بازگشتی () است.

This slide and its sub-slides should take about 5 minutes.

- می‌توانید نشان دهید که چگونه با تغییر خط آخر بلاک مقدار و نوع بازگشتی تغییر می‌کند. به عنوان مثال با اضافه کردن یا حذف کردن یک ; یا با استفاده از کلیدواژه `return` تغییرات را اعمال کنید.

6.4.1 محدوده‌ها و سایه‌گذاری

محدوده (scope) یک متغیر محدود به بلاک محاصره‌کننده آن است.

شما می‌توانید متغیرها را سایه بزنید، هم آن‌هایی که از اسکوپ‌های بیرونی هستند و هم متغیرهایی که از اسکوپ یکسان هستند:

```
                                                } ()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
                                                {
```

- با افزودن یک «b» در بلوک داخلی در آخرین مثال، و سپس تلاش برای دسترسی به آن در خارج از بلوک، نشان دهید که دامنه یک متغیر محدود است.

- 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.
- یک متغیر سایه‌دار می تواند انواع داده‌ای متفاوتی داشته باشد.
- سایه زدن در ابتدا مبهم به نظر می رسد، اما برای نگه داشتن متغیر پس از `unwrap()` . مناسب است.

6.5 توابع

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

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

This slide should take about 3 minutes.

- بعد اعلان تابع پارامترهای ورودی و نوع آن و سپس یک نوع برگشتی هستند (برخلاف برخی از زبان‌های برنامه‌نویسی).
- آخرین عبارت در بدنه تابع (یا هر بلوک دیگری) به عنوان مقدار برگشتی در نظر گرفته می‌شود. به همین سادگی `;` را می‌توان در انتهای عبارت حذف کنید.
- Some functions have no return value, and return the 'unit type', `()`. The compiler will infer this if the return type is omitted.
- بارگذاری مجدد (overloading) پشتیبانی نمی‌شود -- هر تابع فقط یک پیاده‌سازی دارد.
  - همیشه تعداد ثابتی از پارامترها را می گیرد. آرگومان های پیش فرض پشتیبانی نمی شوند. ماکروها را می توان برای پشتیبانی از توابع متغیر استفاده کرد.
  - همیشه یک موجه واحد از انواع آرگومان‌ها را می‌گیرد.

6.6 ماکروها

ماکروها در طول کامپایل به کد Rust گسترش می‌یابند و می‌توانند تعداد متغیری از آرگومان‌ها را بگیرند. آن‌ها در پایان با یک «!» متمایز می شوند. کتابخانه استاندارد Rust شامل مجموعه ای از ماکروهای مفید است.

- `println!(format, ...)` یک خط را در خروجی استاندارد چاپ می کند و قالب بندی شرح داده شده در [std::fmt](https://doc.rust-lang.org/std/fmt/index.html) را اعمال می کند. .
- `format!(format, ...)` درست مانند `println!` کار می کند، اما نتیجه را به صورت یک رشته برمی گردانند.
- `dbg!(expression)` مقدار عبارت را ثبت کرده و آن را برمی گردانند.
- `todo!()` مقداری از کد را به عنوان هنوز پیاده‌سازی نشده علامت‌گذاری می‌کند. panic می کند.
- `unreachable!()` مقداری از کد را غیرقابل دسترسی علامت گذاری می کند. اگر اعدام شود وحشت می کند.

```
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
                                {
```

.This slide should take about 2 minutes

نکته مهم این بخش این است که این امکانات مشترک و نحوه استفاده از آن‌ها وجود دارد. این‌که چرا آن‌ها به عنوان ماکرو تعریف می شوند و به چه چیزی گسترش می یابند، بسیار مهم نیست.

این دوره شامل تعریف ماکروها نمی شود، اما در بخش بعدی استفاده از ماکروهای مشتق شده توضیح داده خواهد شد.

6.7 تمرین: دنباله Collatz

:The Collatz Sequence is defined as follows, for an arbitrary n1 greater than zero

- اگر ni = ۱ باشد، دنباله (sequence) در ni پایان می‌یابد.
- اگر ni زوج باشد، آنگاه ni/۲ = ni+۱.
- اگر ni فرد باشد، آنگاه ni * ۳ + ۱ = ni+۱.

به عنوان مثال، با شروع از ni = ۳:

- ۳ فرد است، پس $n2$ = ۳ * ۳ + ۱ = 10;
- ۱۰ زوج است، پس $n3$ = ۱۰ / ۲ = ۵;
- ۵ فرد است، پس $n4$ = ۳ * ۵ + ۱ = 16;
- ۱۶ زوج است، پس $n5$ = ۱۶ / ۲ = 8;
- ۸ زوج است، پس $n6$ = ۸ / ۲ = 4;
- ۴ زوج است، پس $n7$ = ۴ / ۲ = ۲;
- ۲ زوج است، پس $n8$ = ۱; و
- دنباله به پایان می‌رسد.

یک تابع بنویسید تا طول دنباله Collatz برای یک n اولیه داده شده را محاسبه کند.

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

                    } ()fn main
                ("این را پیاده‌سازی کن")!todo
                                {
```

6.7.1 راحل

```
.`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
                                                             {

                                                    [test]#
                                          } ()fn test_collatz_length
                                    ;(assert_eq!(collatz_length(11), 15
                                                             {

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

بخش II

روز ۱: بعد از ظهر

فصل 7

خوش آمد

با احتساب 10 دقیقه استراحت، این جلسه بایـد حدود 2 ساعت و 35 دقیقه طول بکشد. آن شامل:

بخش	مدت زمان
تاپل ها و آرایه ها	۳۵ دقیقه
مراجع	۵۵ دقیقه
تایپ‌های تعریف شده توسط کاربر	۵۰ دقیقه

فصل 8

تاپل ها و آرایه ها

این بخش بایـد حدود 35 دقیقه طول بکشد. این شامل:

اسلایـد	مدت زمان
آرایه‌ها	۵ دقیقه
تاپل‌ها	۵ دقیقه
تکرار ارایه	۳ دقیقه
الگوها و ضدِ ساخت‌اره‌ها	۵ دقیقه
تمرین: آرایه‌های تو در تو	۱۵ دقیقه

8.1 آرایه‌ها

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

.This slide should take about 5 minutes

- یک مقدار از نوع آرایه `[T; N]` دارای `N` (یک ثابت زمان کامپایل) عنصر از نوع یکسان `T` است. توجه داشته باشید که طول آرایه بخشی از نوع آن است، به این معنی که `[u8; 3]` و `[u8; 4]` دو نوع متفاوت در نظر گرفته می‌شوند.
- سعی کنید به یک عنصر آرایه خارج از محدوده دسترسی داشته باشید. دسترسی های آرایه در زمان اجرا بررسی می شود. زنگ معمولاً می‌تواند این بررسی‌ها را از بین ببرد و با استفاده از Rust ناامن از آن‌ها جلوگیری کرد.
- ما می‌توانیم از مقادیر ثابت برای اندازه‌ها استفاده کنیم.
- ماکرو `println!` با پارامتر فرمت `?` نیازمند پیاده سازی دیباگ است: `{}` خروجی پیش فرض را می‌دهد، `{:?}` خروجی دیباگ را می‌دهد. انواع‌ای مانند اعداد صحیح و رشته‌ها خروجی پیش فرض را پیاده سازی می‌کنند، اما آرایه‌ها فقط خروجی دیباگ را پیاده سازی می‌کنند. این بدان معناست که ما باید در اینجا از خروجی دیباگ استفاده کنیم.

- اضافه کردن #، مانند {a:#?}، یک فرمت «چاپ زیبا» را فراخوانی می‌کند که می‌تواند خواندن آن را آسان‌تر کند.

8.2 تاپل‌ها

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

.This slide should take about 5 minutes

- مانند آرایه‌ها، تاپل‌ها نیز دارای طول ثابت هستند.
- تاپل‌ها مقادیر انواع مختلف را در یک نوع مرکب کنار هم قرار می‌دهند.
- می‌توان به فیلدهای یک تاپل با استفاده از نقطه و شماره اندیس مقدار، مانند `t.0`، `t.1` دسترسی پیدا کرد.
- تاپل خالی `()` به عنوان `unit type` نامیده می‌شود و نشان‌دهنده عدم وجود مقدار بازگشتی است، مشابه `void` در زبان‌های دیگر.

8.3 تکرار آرایه

عبارت `for` از تکرار روی آرایه‌ها (اما نه تاپل‌ها) پشتیبانی می‌کند.

```
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);
        }
    }
}
```

.This slide should take about 3 minutes

این قابلیت از ویژگی `IntoIterator` استفاده می‌کند، اما ما هنوز به آن پرداخته‌ایم.

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 الگوها و ضدِ ساختارها

هنگام کار با تاپل‌ها و سایر مقادیر ساختاری‌افته، معمول است که بخواهید مقادیر داخلی را در متغیرهای محلی استخراج کنید. این را می‌توان به صورت دستی با دسترسی مستقیم به مقادیر داخلی انجام داد:

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

```
    println!("left: {left}, right: {right}");
}
```

با این حال، Rust همچنین از استفاده از تطبیق الگو برای تخریب یک مقدار بزرگتر در بخش های تشکیل دهنده آن پشتیبانی می کند:

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

This slide should take about 5 minutes.

- الگوهای استفاده شده در اینجا "irrefutable" هستند، به این معنی که کامپایلر می تواند به طور استاتیکی تأیید کند که مقدار سمت راست = ساختاری مشابه الگو دارد.
- نام متغیر یک الگوی انکارناپذیر است که همیشه با هر مقداری مطابقت دارد، از این رو می‌توانیم از «let» برای اعلام یک متغیر استفاده کنیم.
- Rust همچنین از استفاده از الگوها در شرطی‌ها پشتیبانی می‌کند و امکان مقایسه برابری و تخریب ساختار را در همان زمان فراهم می‌کند. این شکل از تطبیق الگو بعداً با جزئیات بیشتری مورد بحث قرار خواهد گرفت.
- متغیرهای بالا را ویرایش کنید تا خطای کامپایلر در زمانی که الگو با مقدار مطابقت‌شده مطابقت ندارد نشان داده شود.

8.5 تمرین: آرایه‌ای تو در تو

آرایه ها می توانند آرایه های دیگری نیز داشته باشند:

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

نوع این متغیر چیست؟

از آرایه‌ای مشابه مثال بالا برای نوشتن عبارت `transpose` استفاده کنید که یک ماتریس را جابجا می‌کند (ردیف‌ها به ستون‌ها تبدیل می‌کند):

```
|1 4 7]      |1 2 3]
|2 5 8|  ==  |4 5 6|  transpose
|3 6 9]      |7 8 9]
```

کد زیر را در https://play.rust-lang.org/ کپی کرده و توابع را پیاده‌سازی کنید:

```
// TODO: این را زمانی که پیاده‌سازی‌تان تمام شد حذف کنید.
#![allow(unused_variables, dead_code)]

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

#[test]
fn test_transpose() {
    let matrix = [
        [101, 102, 103], // <-- the comment makes rustfmt add a newline
        [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
.دنک هفاض دیدج طخ کی rustfmt دوشیم ثعاب تنماک نیا --> // ,[101, 102, 103]
                                    ,[201, 202, 203]
                                    ,[301, 302, 303]
                                               ;[

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

8.5.1 لحارا

```
} [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
                                            {

                                          [test]#
                                    } ()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
.دنک هفاض دیدج طخ کی rustfmt دوشیم ثعاب تنماک نیا --> // ,[101, 102, 103]
                                                ,[201, 202, 203]
                                                ,[301, 302, 303]
                                                            ;[

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

فصل 9

مراجع

این بخش دایره حدود ۵۵ دقیقه طول بکشد. آن شامل:

زمان مدت	اسلایدها
۱۰ دقیقه	مراجع اشتراکی
۱۰ دقیقه	مراجع انحصاری
۱۰ دقیقه	برش‌ها
۱۰ دقیقه	رشته‌ها
۱۵ دقیقه	تمرین: هندسه

9.1 مراجع اشتراکی

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);
}
```

یک مرجع یک نوع T دارای نوع &T است. یک مقدار مرجع با عملگر & ساخته می شود. عملگر * یک مرجع را "ارجاع مجدد" می کند و مقدار آن را به دست می دهد.

راست بطور استاتیک مراجع تعلیق شده (dangling) را ممنوع می‌کند:

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

.This slide should take about 10 minutes

- گفته می‌شود که مرجع مقداری را که به آن ارجاع می‌دهد "borrow" (قرض) می‌کند، و این مدل خوبی برای دانش‌آموزانی است که با اشاره‌گرها آشنا نیستند: کد می‌تواند از مرجع برای دسترسی به مقدار استفاده کند، اما همچنان متعلق به متغیر اصلی است. این دوره در روز 3 به جزئیات بیشتری در مورد مالکیت خواهد پرداخت.
- مراجع به عنوان اشاره‌گر پیاده‌سازی می‌شوند و یک مزیت کلیدی این است که می‌توانند بسیار کوچک‌تر از چیزی باشند که به آن اشاره می‌کنند. دانش‌آموزانی که با C یا C++ آشنا هستند، مراجع را به عنوان اشاره‌گر تشخیص می‌دهند. بخش‌های بعدی دوره به این موضوع می‌پردازد که چگونه Rust از اشکال‌زدایی حافظه نشانگرهای خام جلوگیری می‌کند.
- Rust به طور خودکار برای شما مراجع ایجاد نمی‌کند - & همیشه مورد نیاز است.
- راست در برخی موارد به‌طور خودکار از Dereference می‌کند، به‌ویژه هنگام فراخوانی متدها (`ref_x.count_ones()`) را امتحان کنید.
- در این مثال، r قابل تغییر است تا بتواند آن را مجدداً اختصاص داد (r = &b;). توجه داشته باشید که این r را دوباره متصل می‌کند، به طوری که به چیز دیگری اشاره می‌کند. این با C++ متفاوت است، جایی که انتساب به یک مرجع مقدار مرجع را تغییر می‌دهد.
- یک مرجع مشترک اجازه تغییر مقداری را که به آن ارجاع می‌دهد را نمی‌دهد، حتی اگر آن مقدار قابل تغییر باشد. "r = "X*" را امتحان کنید.
- Rust طول عمر همه مراجع را ردیابی می‌کند تا اطمینان حاصل شود که آن‌ها به اندازه کافی عمر می‌کنند. ارجاعات آویزان نمی‌توانند در Rust ایمن رخ دهند. x_axis یک ارجاع به point برمی‌گرداند، اما «نقطه» زمانی که تابع برمی‌گردد، تخصیص داده می‌شود، بنابراین کامپایل نمی‌شود.
- هنگامی که به مالکیت در زبان راست رسیدیم، بیشتر در مورد «قرض دادن» صحبت خواهیم کرد.

9.2 مراجع انحصاری

مراجع انحصاری، همچنین به عنوان مراجع قابل تغییر شناخته می‌شوند، اجازه می‌دهند مقداری را که به آن ارجاع می‌دهند تغییر دهند. آن‌ها نوع &mut T دارند.

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

.This slide should take about 10 minutes

نکات کلیدی:

- "انحصاری" به این معنی است که فقط از این مرجع می‌توان برای دسترسی به مقدار استفاده کرد. هیچ مرجع دیگری (اشتراک‌گذاری شده یا انحصاری) نمی‌تواند همزمان وجود داشته باشد، و تا زمانی که مرجع انحصاری وجود دارد، نمی‌توان به مقدار ارجاع‌شده دسترسی داشت. زمانی که x_coord زنده است، &point.0 بسازید یا point.0 را تغییر دهید.
- حتماً تفاوت بین «let mut x_coord: &i32» و «let x_coord: &mut i32» را ادداشت کنید. مورد اول یک مرجع مشترک را نشان می‌دهد که می‌تواند به مقادیر مختلف متصل شود، در حالی که دومی نشان‌دهنده یک مرجع انحصاری به یک مقدار قابل تغییر است.

9.3 برش‌ها

یک برش به شما امکان می‌دهد نما (view) از یک مجموعه بزرگ‌تر داشته باشید:

```
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
}
```

- برش‌ها داده‌ها را از نوع برش‌شده قرض می‌گیرند.
- پرسش: اگر a[3] را درست قبل از چاپ s تغییر دهید چه اتفاقی می‌افتد؟

This slide should take about 10 minutes.

- ما با قرض گرفتن a و مشخص کردن شاخص‌های شروع و پایان در براکت‌ها، برش (slice) ایجاد می‌کنیم.
- اگر برش از شاخص ۰ شروع شود، سینتکس راست به ما اجازه می‌دهد شاخص شروع را حذف کنیم (یعنی عدد صفر را ننویسیم)، به این معنی که &a[0..a.len()] و &a[..a.len()] یکسان هستند.
- در مورد شاخص آخر نیز همین‌طور است، بنابراین &a[2..a.len()] و &a[2..] یکسان هستند.
- یک روش ساده برای برش کل آرایه، این است که از &a[..] استفاده کنیم.
- s یک مرجع به برش i32 است. توجه داشته باشید که نوع s (&[i32]) دیگر طول آرایه را ذکر نمی‌شود. این به ما امکان می‌دهد محاسباتی را روی برش‌هایی با اندازه‌های مختلف انجام دهیم.
- برش‌ها همیشه از یک شیء دیگر قرض می‌گیرند. در این مثال، a باید حداقل به اندازه طول مورد برش ما، زنده (در محدوده) باقی بماند.
- پرسش در مورد تغییر a[3] می‌تواند یک بحث جالب را شروع کند، اما پاسخ‌اش این است که به دلایلی نمی‌خواهیم حافظه، نمی‌توانیم این کار را از طریق a در این مرحله از اجرا انجام دهیم، اما می‌توانیم داده‌ها را از هر دو a و s به طور امن بخوانیم. این کار قبل از ایجاد برش و دوباره بعد از println! کار می‌کند، زمانی که برش دیگر استفاده نمی‌شود. جزئیات بیشتری در بخش بررسی‌کننده-قرض (the borrow checker) توضیح خواهیم داد.

9.4 رشته‌ها

حالا می‌توانیم دو نوع رشته‌ای را در راست درک کنیم:

- &str تکه‌ای از بایت‌های رمزگذاری‌شده UTF-8، شبیه به &[u8] است.
- String is an owned buffer of UTF-8 encoded bytes, similar to Vec<T>.

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

    let mut s2: String = String::from("دنیا");
    println!("s2: {s2}");
    s2.push_str(s1);
    println!("s2: {s2}");
```

```
;[..()let s3: &str = &s2[s2.len() - s1.len
                        ;("{println!("s3: {s3
                                            {
```

.This slide should take about 10 minutes

- `&str` یک برش رشته‌ای را مرجع غیرقابل تغییر به داده‌های رشته‌ای رمزشده UTF-8 است که در یک بلوک حافظه ذخیره شده است. لیترال های رشته‌ای `String` ("Hello") در باینری برنامه ذخیره می‌شوند.
- در راست نوع `String` یک wrapper بر روی یک بردار از بایت‌هاست. مانند `Vec<T>`، یک نوع Owned است.
- مانند بسیاری از انواع دیگر، `String::from()` یک رشته از یک لیترال رشته ایجاد می‌کند. `String::new()` که رشته خالی جدید ایجاد می‌کند که داده های رشته‌ای می‌توانند با استفاده از متدهای `push()` و `push_str()` به آن اضافه شوند.
- ماکرو `format!()` یک راه راحت برای ایجاد یک رشته Owned از مقادیر پویا است. مثل فرمت قابل پذیرش توسط ماکرو `println!()` است.
- می‌توانید برش‌های `str&` را از `String` از طریق & و انتخابی محدوده انتخاب کنید. اگر محدوده بایتی را انتخاب کنید که با مرزهای نویسه تراز نباشد، عبارت وحشت می کند. تکرار کننده `chars` روی کاراکترها تکرار می شود و بر تلاش برای درست کردن مرزهای کاراکتر ترجیح داده می شود.
- برای برنامه‌نویسان C++: `&str` را به عنوان `const char*` در C++ درنظر بگیرید، اما یک فرق مهم این است که در راست که همیشه به یک رشته معتبر در حافظه اشاره می کند. راست نوع `String`معادل تقریبی `std::string` در C++ است (با این تفاوت که فقط می‌تواند حاوی بایت-های رمزشده UTF-8 باشد و هرگز از بهینه‌سازی Small-String استفاده نمی کند).
- رشته‌های بایت به شما امکان می دهند مستقیماً یک مقدار `u8[]` & ایجاد کنید:

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

- رشته‌های خام به شما امکان می دهند یک مقدار &str با غیرفعال کردن فرارها ایجاد کنید: `r"\n" == "\\n"`.شما می‌توانید با استفاده از تعداد برابر # در دو طرف دابل‌کوت دابل‌کوت‌ها، دابل‌کوت‌ها را جاسازی کنید:

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

9.5 تمرین: هندسه

ما چند توابع کاربردی برای هندسه سه بعدی ایجاد خواهیم کرد که نقطه ای را به عنوان `[f64;3]` نشان می دهد. تعیین امضاهای عملکرد به عهده شماست.

```
// اندازه یک بردار را با جمع مربعات مختصات آن محاسبه کنید
// و سپس جذر آن را بگیرید. از متد `()sqrt` برای محاسبه جذر استفاده کنید، مثل `v.sqrt()`
```

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

// یک بردار را با محاسبه اندازه‌ی آن و تقسیم تمام مختصات آن
// بر آن اندازه، نرمال‌سازی کنید.
fn normalize(...) {
    todo!()
}

// از `main` زیر برای تست کار خود استفاده کنید.

fn main() {
    println!("اندازه‌ی یک بردار واحد: {}", magnitude(&[0.0, 1.0, 0.0]));

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

9.5.1 راه‌حل

```rust
/// اندازه‌ی بردار داده شده را محاسبه کنید.
fn magnitude(vector: &[f64; 3]) -> f64 {
    let mut mag_squared = 0.0;
    for coord in vector {
        mag_squared += coord * coord;
    }
    mag_squared.sqrt()
}

/// اندازه‌ی بردار را به 1.0 تغییر دهید بدون اینکه جهت آن تغییر کند.
fn normalize(vector: &mut [f64; 3]) {
    let mag = magnitude(vector);
    for item in vector {
        *item /= mag;
    }
}

fn main() {
    println!("اندازه‌ی یک بردار واحد: {}", magnitude(&[0.0, 1.0, 0.0]));

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

فصل 10

تایپ‌های تعریف شده توسط کاربر

این بخش حدود ۵۰ دقیقه طول خواهد کشید. شامل موارد زیر است:

اسلاید	مدت زمان
ساختارهای نامدار	۱۰ دقیقه
ساختار تاپل‌ها	۱۰ دقیقه
Enums	۵ دقیقه
Static	۵ دقیقه
نام‌های مستعار تایپ	۲ دقیقه
تمرین: رویدادهای آسانسور	۱۵ دقیقه

10.1 ساختارهای نامدار

مانند C و C++، زبان Rust از ساختارهای سفارشی پشتیبانی می‌کند:

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

fn describe(person: &Person) {
    println!("{} {} ساله است", person.name, person.age);
}

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

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

    let name = String::from("ایوری");
    let age = 39;
    let avery = Person { name, age };
```

```
;(describe(&avery

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

.This slide should take about 10 minutes

نکات کلیدی:

- ساختارها (Structs) در Rust مانند C یا C++ عمل می‌کنند.
  - مانند C++ و برخلاف C، برای تعریف یک نوع نیازی به `typedef` نیست.
  - برخلاف C++، در Rust بین ساختارها ارث‌بری وجود ندارد.
- این زمان مناسبی است تا به مردم اطلاع دهیم که انواع مختلفی از ساختارها وجود دارد.
  - ساختارهای بدون اندازه (مانند `struct Foo;`) ممکن است زمانی استفاده شوند که می‌خواهید یک صفت (trait) را بر روی یک نوع پیاده‌سازی کنید، اما داده‌ای ندارید که بخواهید در خود مقدار ذخیره کنید.
  - اسل‌های بعدی ساختارهای تاپل (Tuple structs) را معرفی خواهد کرد، که زمانی استفاده می‌شوند که نام فیلدها مهم نیستند.
- اگر از قبل متغیرهایی با نام‌های مناسب دارید، می‌توانید ساختار را با استفاده از یک روش میان‌بر ایجاد کنید.
- سینتکس `avery..` به ما اجازه می‌دهد که اکثر فیلدها را از ساختار قدیمی کپی کنیم بدون این‌که همه آن‌ها را صریحاً تایپ کنیم. این بایدهمیشه آخرین عنصر باشد.

10.2 ساختارهای تاپل

اگر نام فیلدها بی‌اهمیت هستند، می‌توانید از ساختار tuple استفاده کنید:

```
;(struct Point(i32, i32

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

این اغلب برای single-field wrapper (که newtypes نامیده می‌شوند) استفاده می‌شود:

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

} fn compute_thruster_force() -> PoundsOfForce
("برسپب اسان در کشوم حوزه دنمشدان کی از")!todo
{

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

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

.This slide should take about 10 minutes

- Newtypes یک راه من‌اسب‌ب برای رمزگذاری اطلاعات اضافی در مورد مقدار در یک نوع اولیه (primitive type) است، به عنوان مثال:
  - این عدد در برخی واحدها اندازه گیری می‌شود: `Newtons` در مثال بالا.
  - مقدار زمانی که ایجاد شد مقداری اعتبارسنجی را دریافت کرد، بنابراین دیگر لازم نیست در هر بار استفاده دوباره آن را تأیید کنید: `PhoneNumber(String)` یا `OddNumber(u32)`.
- نحوه افزودن مقدار `f64` به نوع `Newtons` را با دسترسی به single field در نوع جدید نشان دهید.
  - Rust معمولاً چیزهای غیر واضح را دوست ندارد، مانند automatic unwrapping یا به‌عنوان-مثال استفاده از `boolean`ها به عنوان اعداد صحیح.
  - مبحث Operator overloading در روز سوم مورد بحث قرار می گیرد (generics).
- این مثال، اشاره ظریفی به شکست مدارگرد آب‌وهوای مریخ است.

Enums 10.3

کلمه کلیدی enum اجازه ایجاد نوع داده‌ای را می دهد که دارای چندین گونه مختلف است:

```
#[derive(Debug)]
enum Direction {
    Left,
    Right,
}

#[derive(Debug)]
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!("در این پیچ: {:?}", m);
}
```

.This slide should take about 5 minutes

نکات کلیدی:

- Enumها به شما امکان می دهند مجموعه‌ای از مقادیر مختلف با نوع‌های مختلف را تحت یک نوع جمع آوری کنید.
- `Direction` یک type با گونه‌های مختلف است. دو مقدار `Direction` وجود دارد: `Direction::Left` و `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.
- الان زمان خوبی برای مقایسه ساختارها و Enumهاست:
  - در هر دو، می توانید یک نسخه ساده بدون فیلد (unit struct) یا یکی با انواع مختلف فیلد (variant payloads) داشته باشید.
  - شما حتی می توانید انواع مختلف یک `Enum` را با ساختارهای جداگانه پیاده سازی کنید، اما در آن صورت آنها از همان نوعی که در ابتدا تعریف کردید نمی `Enum` نخواهند بود.
- Rust از حداقل فضای برای ذخیره‌سازی متمایزکننده (discriminant) استفاده می‌کند.
  - در صورت لزوم، یک عدد صحیح با کوچکترین اندازه مورد نیاز را ذخیره می‌کند

- اگر مقادیر متغیر مجاز همه الگوهای bit را پوشش ندهند، از الگوهای bit نامعتبر برای رمزگذاری متمایز کننده (یک "niche optimization") استفاده می‌کند. برای مثال، Option<&u8> یک اشاره‌گر به یک عدد صحیح یا NULL را برای نوع None ذخیره می‌کند.
- شما می توانید در صورت نیاز (به عنوان مثال، برای سازگاری با C) discriminant را کنترل کنید:

```
#[repr(u32)]
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);
}
```

بدون repr، نوع discriminant دو بایت حافظه اشغال می‌کند، زیرا 10001 در 2 بایت جا می‌شود.

برای کاوش بیشتر

زبان Rust دارای چندین به‌ینه‌سازی دارد که می‌تواند برای کاهش فضای اشغال شده توسط Enumها استفاده کند.

- به‌ینه‌سازی اشاره‌گر NULL: برای برخی از انواع، Rust تضمین می‌کند که size_of::<T>() برابر با size_of::<Option<T>>() است.

کد نمونه، اگر می‌خواهید نشان دهید که نمایش بیت به بیت در عمل چگونه ممکن است به نظر برسد. مهم است توجه داشته باشید که کامپایلر هیچ تضمینی در مورد این نمایش نمی‌دهد، بنابراین این کاملاً ناامن است.

```
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
                                                                    {
```

طبق کتاب Rust RFC، این موارد هنگام استفاده درج می شوند.

فقط توابعی که با const علامت گذاری شده اند می توانند در زمان کامپایل برای تولید مقادیر const فراخوانی شوند. با این حال، توابع const را می توان در زمان اجرا فراخوانی کرد (بر خلاف عبارت ثابت)

- به این نکته اشاره کنید که شبه const constexpr در C++ عمل می‌کند.
- با این‌که خیلی رایج نیست که اگر کسی به یک مقدار ثابت که در زمان اجرا ارزیابی می‌شود از const استفاده کند اما مفید تر و ایمن تر از استفاده static هستند.

static 10.5

متغیرهای ایستا در طول عمر کل اجرای برنامه خواهند ماند و بنابراین قلمرو نمی‌شوند:

```
;"دیدآم وش خ RustOS 3.14 به" = 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.

This slide should take about 5 minutes.

- `static` is similar to mutable global variables in C++.
- `static` هویت شی را فراهم می‌کند: آدرسی در حافظه و حالتی که توسط انواع با تغییرپذیری داخلی مانند `Mutex<T>` را نیاز دارد.

برای کاوش بیشتر

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 پایان راهنمای مبتدی

تایپ alias، نامی برای نوع دیگر ایجاد می کند. این دو نوع را می توان به جای هم استفاده کرد.

```
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>>>>;
```

This slide should take about 2 minutes.

برنامه‌نویسان C این را شبیه به `typedef` تشخیص می‌دهند.

10.7 تمرین: رویدادهای آسانسور

ما یک ساختار داده برای نمایش یک رویداد در یک سیستم کنترل آسانسور ایجاد خواهیم کرد. این به شما بستگی دارد که انواع و عملکردها را برای ساخت رویدادهای مختلف تعریف کنید. از `#[derive(Debug)]` استفاده کنید تا اجازه دهید انواع با `{:?}` قالب‌بندی شوند.

این تمرین فقط به ایجاد و پر کردن ساختارهای داده نیاز دارد تا `main` بدون خطا اجرا شود. بخش بعدی این دوره دریافت داده‌ها از این ساختارها را پوشش می‌دهد.

```
#[derive(Debug)]
/// An event in the elevator system that the controller must react to.
enum Event {
    // TODO: add required variants
}
```

```
                                                .A direction of travel ///
                                                        [(derive(Debug]#
                                                        } 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
                                                                       ;(
      ;((car_arrived(0 ,"{?:} :است هدیسر فکمه طبقه هب نیشام")!println
                     ;(()car_door_opened ,"{?:} :در ماشین باز شد")!println
                                                                 )!println
                       ,"{?:} :است هداد رارشف 3 را هبقط همدک رافسم کی"
                                              (car_floor_button_pressed(3
                                                                       ;(
             ;(()car_door_closed ,"{?:} :در ماشین بسته شد")!println
          ;((car_arrived(3 ,"{?:} :است هدیسر ۳ هبقط هب نیشام")!println
                                                                    {
```

10.7.1 راه‌حل

```
[(derive(Debug]#
.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 ///
                                [(derive(Debug]#
                                } enum Direction
                                    ,Up
                                ,Down
                                {

    .A user-accessible button ///
                                [(derive(Debug]#
                                    } 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 ,"{?:} :است هدیرس فکمه هقبط هب نیشام")!println
                     ;(()car_door_opened ,"{?:} :دش زاب نیشام در")!println
                                                                )!println
                        ,"{?:} :است داده رارشف را 3 هقبط همکد رافسم کی"
                                                (car_floor_button_pressed(3
                                                                       ;(
              ;(()car_door_closed ,"{?:} :دش هتسب نیشام در")!println
          ;((car_arrived(3 ,"{?:} :است هدیرس ۳ هقبط هب نیشام")!println
                                                                        {
```

بخش III

روز ۲: صبح

فصل 11

به روز ۲ خوش آمدید

اکنون که مقدار زیادی از Rust را دیده ایم، امروز بر روی سیستم تایپ Rust تمرکز خواهیم کرد:

- تطبیق الگو: استخراج داده از ساختارها.
- متدها: ارتباط توابع با تایپ ها.
- Traits: رفتارهایی که توسط چندین تایپ مشترک هستند.
- Generics: پارامتریسازی تایپها بر اساس تایپهای دیگر.
- کتابخانهاستاندارد تایپها و traits: یک گردش در کتابخانه استاندارد و ارزشمند Rust.

برنامه زمانی

با احتساب ۱۰ دقیقه استراحت، این جلسه باید حدود ۲ ساعت و ۱۰ دقیقه طول بکشد. این شامل:

بخش	مدت زمان
خوش آمدید	۳ دقیقه
تطبیق	۱ ساعت
متدها و تریتها	۵۰ دقیقه

فصل 12

قیبطت

این بخش بای دحدود ۱ ساعت طول بکشد. این شامل:

اسلید	مدت زمان
ریادقم قباطبت	۱۰ دقیقه
تخریب ساختارها	۴ دقیقه
تخریب ساختار Enums	۴ دقیقه
کنترل جریان Let	۱۰ دقیقه
تمرین: ارزیابی عبارت	۳۰ دقیقه

12.1 ریادقم قباطبت

کلمه کلیدی match به شما اجازه می‌دهد یک مقدار را با یک یا چند الگو مطابقت دهید. مقایسه‌ها از بالا به پایین انجام می‌شوند و اولین تطابق انتخاب می‌شود.

الگوها می‌توانند مقادیر ساده ای باشند، شبیه به switch در C و C++:

```
[rustfmt::skip]#
    } ()fn main
;'let input = 'x
   } match input
 => println'
 => println'
  '9'=..'0'
,("کردن کرت")!q'
,("حرکت در اطراف")!a' | 's' | 'w' | 'd'
,("ورودی شما")!println <=
,("حروف کوچک: {key}")!key if key.is_lowercase() => println
,("یک چیز دیگر")!println <=
                _
    {
{
```

الگوی _ یک الگوی عام (Wildcard) است که با هر مقداری مطابقت دارد. عبارت‌ها باید جامع باشند، به این معنی که همه احتمالات را پوشش دهند، بنابراین _ اغلب به عنوان آخرین حالت برای پوشش تمامی موارد استفاده می‌شود.

match می‌تواند به عنوان یک عبارت استفاده شود. دقیقاً مانند if، هر شاخه match باید از یک تایپ باشد. تایپ بازگشتی، تایپ آخرین عبارت در بلاک است، اگر وجود داشته باشد. در مثال بالا، تایپ

بازگشتی ( ) است.

یک متغیر در الگو (key در این مثال) یک اتصال ایجاد می‌کند که می‌توان از آن در بخش مطابقت استفاده کرد.

یک guard در عبارت match باعث می‌شود که آن شاخه تنها در صورتی مطابقت داشته باشد که شرط برقرار باشد.

.This slide should take about 10 minutes

نکات کلیدی:

- بهتر است که اشاره کنید چطوری می‌توان از کاراکترهای خاص در الگو استفاده کرد
  - | به عنوان or
  - .. برای تعیین همه محدوده‌ها تا جایی که می‌توان گسترش یابد
  - 1..=5 نمایانگر یک محدوده خاص است
  - _ نمایانگر هر حالتی است
- guard های تطبیق یک ویژگی سینتکس جداگانه دستبندی می‌شوند, زمانی مهم و ضروری هستند که بخواهیم ایده‌های پیچیده‌تر از الگوهای ساده بیان کنیم.
- آنها با عبارت if جداگانه در داخل یک شاخه تطبیق هستند یکسان نیستند. یک عبارت if در داخل بلاک شاخه (پس از =>) پس از ورود به آن شاخه خاص صدا زده می‌شود. اگر شرط if برقرار نباشد کاری به سایر شاخه‌های عبارت match اصلا ندارد.
- شرط تعریف شده در guard با کمک | به شرط‌های تطبیق الگو اضافه می‌شود.

12.2 ساختارها

مانند tuple ها، ساختار را نیز می توان با تطبیق تخریب کرد:

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

[rustfmt::skip]#
} ()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
,("دندش هتفرگ دهیدان اهدلیف رسای ، {y = {y در")!Foo { y, .. }  => println
{
{
```

.This slide should take about 4 minutes

- تغییر متغیرهای لیترال در foo برای مطابقت با سایر الگوها.
- اضافه کردن یک فیلد جدید به Foo و ایجاد تغییرات مورد نیاز.
- تشخیص تفاوت بین یک گرفتن متغیر و یک عبارت ثابت می‌تواند دشوار باشد. سعی کنید 2 را در شاخه دوم به یک متغیر تغییر دهید و خواهید دید که کار نمی‌کند. آن را به یک const تغییر دهید و خواهید دید که دوباره کار می‌کند.

Enums 12.3

مانند tuple ها، enum ها را نیز می توان با تطبیق تخریب کرد:

الگوها همچنین می‌توانند برای مقدار شما استفاده شوند. این روش به شما اجازه می‌دهد ساختار انواع خود را بررسی کنید. بیایید با یک نوع ساده enum شروع کنیم:

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

} fn divide_in_two(n: i32) -> Result
                } if n % 2 == 0
            (Result::Ok(n / 2
                } else {
(("درک می‌شود تقسیم دو به را  {n}  توان یمن")!Result::Err(format
                    {
                        {

                } ()fn main
            ;let n = 100
        } (match divide_in_two(n
    ,("{half} است دو به می‌شود تقسیم {Result::Ok(half) => println!("{n
,("{msg} :است داده خ یخطای ،شوزپ‌ضرع با")!Result::Err(msg) => println
                    {
                        {
```

این‌جا از شاخه‌ها برای تجزیه مقدار Result استفاده کرده‌ایم. در شاخه‌ی اول، half به مقداری که درون حالت Ok قرار دارد متصل شده است. در شاخه‌ی دوم، msg به پیام خطا متصل شده است.

.This slide should take about 4 minutes

- عبارت if/else یک نوع enum بازمی‌گردانند که عدتاً با استفاده از match هم باز می‌شود.
- می توانید با اضافه کردن یک فیلد دیگر None به enum اضافه کنید و نمایش خطاها هنگام اجرای کد، را تست کنید. مکان هایی را که کد شما اکنون ناقص است و نحوه تلاش کامپایلر برای ارائه نکاتی به شما را نشان دهید.
- مقادیر در حالات enum تنها پس از تطبیق الگو قابل دسترسی هستند.
- نشان دهید چه اتفاقی می‌افتد وقتی جستجو (مطابقت) ناقص است. به مزیتی که کامپایلر Rust فراهم می‌کند، اشاره کنید که تأیید می‌کند همه حالات پوشش داده شده‌اند.
- نتیجه‌ی عبارت divide_in_two را در متغیر result ذخیره کنید و آن را در یک حلقه با استفاده از match بررسی کنید. این کد کامپایل نمی‌شود زیرا msg هنگام مطابقت مصرف می‌شود. برای رفع این مشکل، به جای result از &result استفاده کنید. این کار باعث می‌شود msg به صورت یک ارجاع باشد و مصرف نشود. این ویژگی که به نام "match ergonomics" شناخته می‌شود، در Rust 2018 معرفی شده است. اگر می‌خواهید از نسخه‌های قدیمی‌تر Rust پشتیبانی کنید، به جای msg از ref msg در الگو استفاده کنید.

12.4 کنترل جریان Let

زبان Rust چند ساختار کنترل جریان دارد که با سایر زبان‌ها متفاوت است. این ساختارها برای مطابقت الگو استفاده می‌شوند:

- عبارت if let

- عبارت let else
- عبارت while let

عبارت if let

عبارت if let به شما امکان می‌دهد بسته به این‌که آیا یک مقدار با یک الگو مطابقت دارد، کدهای مختلف را اجرا کنید:

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

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

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

عبارت let else

برای حالت رایج مطابقت با یک الگو و بازگشت از تابع، از let else استفاده کنید. در این‌جا، حالت "else" با دیورژن می‌شود (مانند return، break یا panic - به غیر از این‌که از انتهای بلوک خارج شود).

```
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("hex digit نیست"));
            }
        } else {
            return Err(String::from("رشته خالی است"));
        }
    } else {
        return Err(String::from("هیچ‌چیز"));
    }
}

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

مانند if let، یک دستور while let وجود دارد که مقدار مقابل الگو به طور مکرر (تکرار شونده‌ای) بررسی می‌کند:

```
} ()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) //
{
```

در این‌جا `String::pop` تا زمانی که رشته خالی نشده است، `Some(c)` را برمی‌گرداند و پس از آن `None` را باز می‌گرداند. استفاده از `while let` به ما این امکان را می‌دهد که به طور مداوم از میان همه موارد عبور کنیم.

.This slide should take about 10 minutes

if-let

- برخلاف دستور `match` همه حالت‌های ممکن را برای تطبیق الگو پشت‌بانی نمی‌کند. اگر می‌خواهید همه حالت را پوشش دهید از دستور `match` استفاده کنید.
- یک استفاده رایج از دستور `if let`، رسیدگی به مقادیر `Some` هنگام کار با `Option` است.
- برخلاف دستور `match` دستور `if let` از `=>` برای تطبیق الگو استفاده نمی‌کند.

let-else

`if-let`ها می‌توانند به صورت تو در تو انباشته شوند، همان‌طور که در این‌جا نشان داده شده است. ساختار `let-else` از قلت کردن این کدهای تو در تو پشت‌بانی می‌کند. نسخه‌ی پیچیده را برای دانش‌آموزان بازنویسی کنید تا تبدیل آن را متوجه شوند.

نسخه‌ی بازنویسی شده به صورت زیر است:

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

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

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

    ;(return Ok(digit
{
```

while-let

- توجه داشته باشید که حلقه `while let` تا زمانی که مقادیر مقابل الگو تطبیق داشته باشد (شرط برقرار باشد)، ادامه خواهد داشت.

- شما می‌توانید حلقه‌ی `while let` را به صورت یک حلقه بی‌پایان با یک دستور `if` بازنویسی کنید که در صورت عدم وجود مقداری برای باز کردن (unwrap) از `name.pop()`، شکسته می‌شود. `while let` یک Syntactic sugar برای این سناریو ارائه می‌دهد.

12.5 تمرین: ارزیابی عبارت

بیایید یک ارزیاب ساده بازگشتی برای عبارات حسابی بنویسیم.

نوع `Box` در اینجا یک اشاره‌گر هوشمند است و در ادامه دوره به طور مفصل مورد بررسی قرار خواهد گرفت. یک عبارت می‌تواند با استفاده از `Box::new` "باکس" شود، همان‌طور که در تست‌ها مشاهده می‌شود. برای ارزیابی یک عبارت باکس‌شده، از عملگر deref (`*`) برای "باز کردن باکس" استفاده کنید: `eval(*boxed_expr)`.

برخی از عبارات نمی‌توانند ارزیابی شوند و خطا برمی‌گردانند. نوع `Result<Value, String>` یک enum است که یا نمایانگر یک مقدار موفقیت‌آمیز (`Ok(Value)`) یا یک خطا (`Err(String)`) است. ما این نوع را به‌طور مفصل‌تر در آینده پوشش خواهیم داد.

کد را کپی و در Rust Playground پیست کنید و پیاده‌سازی تابع `eval` را آغاز کنید. محصول نهایی باید تست‌ها را پاس کند. ممکن است استفاده از `todo!()` و گذراندن تست‌ها به صورت تک به تک مفید باشد. همچنین می‌توانید به طور موقت یک تست را با استفاده از `#[ignore]` نادیده بگیرید:

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

اگر زودتر تمام کردید، سعی کنید یک تست بنویسید که منجر به تقسیم بر صفر یا سرریز عدد صحیح شود. چگونه می‌توانید این را با استفاده از `Result` به جای panic مدیریت کنید؟

```
/// An operation to perform on two subexpressions.
#[derive(Debug)]
enum Operation {
    Add,
    Sub,
    Mul,
    Div,
}

/// An expression, in tree form.
#[derive(Debug)]
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!()
}

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

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

                                                [test]#
                                            } ()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
                                                    ;(
                                                {

                                                [test]#
                                         } ()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
                                                    ;(
                                                {

                                                [test]#
                                        } ()fn test_error
                                            )!assert_eq
                                } eval(Expression::Op
                            ,op: Operation::Div
  ,((left: Box::new(Expression::Value(99
  ,((right: Box::new(Expression::Value(0
                                                ,({
       (("رفص رب میسقت")Err(String::from
                                                    ;(
```

```
{
```

12.5.1 راه‌حل

```
.An operation to perform on two subexpressions ///
[(derive(Debug]#
} enum Operation
,Add
,Sub
,Mul
,Div
{

.An expression, in tree form ///
[(derive(Debug]#
} 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
;(("رفص رب میسقت")return Err(String::from
} else {
left / right
{
{
({
{
,(Expression::Value(v) => Ok(v
{
{
```

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

                                                    [test]#
                                                } ()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
                                                        ;(
                                                        {

                                                    [test]#
                                                } ()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
                                                        ;(
                                                        {

                                                    [test]#
                                                } ()fn test_error
                                                )!assert_eq
                                            } eval(Expression::Op
                                        ,op: Operation::Div
  ,((left: Box::new(Expression::Value(99
  ,((right: Box::new(Expression::Value(0
                                                    ,({
```

```
                                        (("رفص رب میسقت")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 ,"{?:} :هجیتن")!println
                                                                        {
```

فصل 13

متدها و تریتها

این بخش حدود ۵۰ دقیقه طول خواهد کشید. شامل موارد زیر است:

اسلاید	مدت زمان
متدها	۱۰ دقیقه
Traits	۱۵ دقیقه
Deriving	۳ دقیقه
تمرین: توابع Generic	۲۰ دقیقه

13.1 متدها

Rust به شما این امکان را می‌دهد که توابعی را با تایپ جدید خود مرتبط کنید. این کار را با استفاده از بلوک `impl` انجام می‌دهید:

```rust
#[derive(Debug)]
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!("ضبط {} دور برای {}:", self.name, self.laps.len());
```

```rust
        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();
        println!("مسابقه {} به پایان رسید، کل زمان دور: {}", self.name, total);
    }
}

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

آرگومان‌های `self` ”گیرنده“ را مشخص می‌کنند - شیئی که متد بر روی آن عمل می‌کند. چندین گیرنده رایج برای یک متد وجود دارد:

- `&self`: شیئی را از فراخوانندە با استفاده از یک مرجع مشترک و غیرقابل تغییر قرض می‌گیرد. شیء می‌تواند بعداً دوباره استفاده شود.
- `&mut self`: شیء را از فراخوانندە با استفاده از یک مرجع منحصر به فرد و قابل تغییر قرض می‌گیرد. شی پس از آن نمی‌تواند دوباره استفاده شود تا زمانی که مرجع به پایان برسد.
- `self`: مالکیت شیء را به عهده می‌گیرد و آن را از فراخوانندە منتقل می‌کند. متد مالک شیء می‌شود. شیء هنگامی که متد باز می‌گردد حذف خواهد شد، مگر اینکه مالکیت آن به‌طور صریح منتقل شود. مالکیت کامل به‌طور خودکار به معنای قابلیت تغییر نیست.
- `mut self`: مشابه مورد بالا، اما متد می‌تواند شیء را تغییر دهد.
- بدون گیرنده: این تبدیل به یک متد استاتیک در ساختار می‌شود. معمولاً برای ایجاد سازنده‌ها استفاده می‌شود که به‌طور معمول `new` نامیده می‌شوند.

This slide should take about 8 minutes.

نکات کلیدی:

- مفید است که متدها را با مقایسه آن‌ها با توابع معرفی کنیم.
  - متدها بر روی یک نمونه از تایپی (مانند struct یا enum) فراخوانی می‌شوند، و پارامتر اول نمونه را به‌عنوان نمونه `self`.
  - توسعه‌دهندگان ممکن است تصمیم بگیرند از متدها استفاده کنند تا از نحو گیرنده متد بهره‌برداری کنند و به سازمان‌دهی بهتر کد کمک کنند. با استفاده از متدها، می‌توانیم تمامی کدهای پیاده‌سازی را در یک مکان قابل پیش‌بینی نگه داریم.
- استفاده از کلمه کلیدی `self`، که به‌عنوان گیرنده متد عمل می‌کند، را مشخص کنید.
  - نشان دهید که `self` یک اصطلاح کوتاه‌شده برای `self: Self` است و شاید نشان دهید که چگونه نام struct نیز می‌تواند استفاده شود.
  - توضیح دهید که `Self` یک نام مستعار نوع برای تایپ است که بلوک `impl` در آن قرار دارد و می‌تواند در سایر بخش‌های بلوک استفاده شود.
  - Note how `self` is used like other structs and dot notation can be used to refer to

.individual fields
- این ممکن است زمان مناسبی باشد برای نشان دادن تفاوت بین `&self` و `self` با تلاش برای اجرای متد `finish` دو بار.
- فراتر از حالت‌های مختلف `self`، تایپ‌های special wrapper types نیز وجود دارند که به‌عنوان تایپ‌های گیرنده مجاز هستند، مانند `Box<Self>`.

Traits 13.2

راست به شما این امکان را می‌دهد که با استفاده از traits بر روی تایپ‌ها انتزاع ایجاد کنید. آن‌ها شبیه interface ها هستند:

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

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

.This slide and its sub-slides should take about 15 minutes

- یک trait مجموعه‌ای از متدها را تعریف می‌کند که تایپ‌ها باید آن‌ها را داشته باشند تا بتوانند آن trait را پیاده‌سازی کنند.
- در بخش "Generics"، در ادامه خواهید دید که چگونه می‌توانید عملکردی بسازید که generic بر روی تمام تایپ‌های که یک trait را پیاده‌سازی کرده‌اند باشد.

Traits پیاده سازی 13.2.1

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

} (fn greet(&self
;(()self.talk ,"{} ؟هچ تمسا !یتسه ابیز ردچ و تو هوا")!println
{
{

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

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

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

- برای پیاده‌سازی `Trait` برای `Type`، از بلوک `impl Trait for Type { .. }` استفاده می‌کنید.
- برخلاف رابط‌های Go، داشتن فقط متدهای مطابقت‌دهنده کافی نیست: نوع `Cat` با متد `talk()` به‌طور خودکار `Pet` را برآورده نمی‌کند، مگر این‌که در یک بلوک `impl Pet` قرار داشته باشد.
- Traits ممکن است پیاده‌سازی‌های پیش‌فرض برای برخی از متدها ارائه دهند. پیاده‌سازی‌های پیش‌فرض می‌توانند به تمامی متدهای trait وابسته باشند. در این مورد، `greet` ارائه شده است و به `talk` وابسته است.

13.2.2 Supertraits

یک trait می‌تواند نیاز داشته باشد که تایپ‌هایی که آن را پیاده‌سازی می‌کنند، همچنین traits دیگری به نام *supertraits* را نیز پیاده‌سازی کنند. در اینجا، هر نوعی که `Pet` را پیاده‌سازی کند، باید `Animal` را نیز پیاده‌سازی کند.

```
} 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
                            {
```

این گاهی اوقات "trait inheritance" نامیده می‌شود، اما دانش‌آموزان نباید انتظار داشته باشند که این رفتار مشابه وراثت در برنامه‌نویسی شیءگرا (OO) باشد. این تنها یک الزام اضافی بر روی پیاده‌سازی‌های یک trait را مشخص می‌کند.

13.2.3 تایپ‌های وابسته

تایپ‌های مرتبط تایپ‌هایی جایگزین هستند که توسط پیاده‌سازی trait تأمین می‌شوند.

```
#[derive(Debug]
;(struct Meters(i32
```

```
#[derive(Debug)]
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)));
}
```

- تای‌پ‌های مرتبط گاهی اوقات "تای‌پ‌های خروجی" نیز نامیده می‌شوند. نکته کلیدی این است که پیاده‌ساز، نه فراخواننده، این تای‌پ را انتخاب می‌کند.
- بسیاری از traitهای کتابخانه استاندارد دارای نوع‌های مرتبط هستند، از جمله اپراتورهای حسابی و Iterator.

Deriving 13.3

Traitهای پشتیبانی‌شده می‌توانند به‌طور خودکار برای تای‌پ‌های سفارشی شما پیاده‌سازی شوند، به شرح زیر:

```
#[derive(Debug, Clone, Default)]
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");
    println!("{:?} vs. {:?}", p1, p2); // `{?:}` Debug trait adds support for printing with //
}
```

This slide should take about 3 minutes.

انتزاع (Derivation) با استفاده از ماکروها پیاده‌سازی می‌شود و بسیاری از crateها ماکروهای مفیدی برای اضافه کردن قابلیت‌های کاربردی ارائه می‌دهند. به عنوان مثال، serde می‌تواند پشتیبانی از ترتیب را برای یک ساختار با استفاده از #[derive(Serialize)] فراهم کند.

13.4 تمرین: Trait Logger

بیایید یک ابزار لاگ ساده طراحی کنیم که از یک trait به نام Logger با متد log استفاده کند. کدی که ممکن است پیشرفت خود را لاگ کند می‌تواند یک &impl Logger دریافت کند. در زمان تست، این ممکن است پیام‌ها را در فایل لاگ تست قرار دهد، در حالی که در نسخه تولید، پیام‌ها به یک سرور لاگ ارسال می‌شود.

با این حال، StderrLogger که در زیر داده شده است، تمامی پیام‌ها را بدون توجه به سطح جزئیات لاگ می‌کند. وظیفه شما این است که نوع VerbosityFilter را بنویسید که پیام‌هایی با سطح جزئیات بالاتر از حداکثر سطح تعیین‌شده را نادیده بگیرد.

این الگو الگوی رایجی است: یک ساختار که یک پیاده‌سازی trait را در بر می‌گیرد و همان trait را پیاده‌سازی می‌کند و در این فرآیند به آن رفتار اضافی می‌دهد. چه نوع‌های دیگری از پوشش‌دهنده‌ها ممکن است در یک ابزار لاگ مفید باشند؟

```
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!("{verbosity}: {message}=رتشیب اطلاعات");
    }
}

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

// TODO: Define and implement `VerbosityFilter`.

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

13.4.1 راه‌حل

```
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
;("{verbosity}: {message}=رتشیب تاعلاطا")!eprintln
                                                        {
                                                            {

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

بخش IV

روز دوم: عصر

فصل 14

خوش آمد

با احتساب ۱۰ دقیقه استراحت، این جلسه بایـد حدود ۳ ساعت و ۱۵ دقیقه طول بکشد. آن شامل:

بخش	مدت زمان
Generics	۴۵ دقیقه
کتابخانه استاندارد	۱ ساعت
تایپ‌ها	
کتابخانه استاندارد	۱ ساعت و ۱۰ دقیقه
Traits	

فصل 15

Generics

این بخش بای د حدود ۴۵ دقی قه طول بکشد. آن ش امل:

اس ل ای د	م دت زم ان
ت وابع Generic	۵ دقی قه
دی تا ت ای پ ه ای Generic	۱۰ دقی قه
Trait Bounds	۱۰ دقی قه
impl Trait	۵ دقی قه
dyn Trait	۵ دقی قه
ت مری ن: Generic min	۱۰ دقی قه

15.1 ت وابع Generic

پ شت ی بان ی م ی ک ن د ک ه ب ه ش ما امک ان م ی د ه د ال گوری تم ا ی ا س اخ تاره ای داده (م ان ن د م رتب س ازی ی ا درخت دودوی ی) را ب ر روی ت ای پ ه ای اس تف اده ش ده ی ا ذخی ره ش ده ت خ ص ی ص ده ی د. از Rust generics

```
.`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 ,"{?:} :ش ماره ای را ان تخ اب ک رد")!println
;(((2 ,"گ ربه") ,(1 ,"س گ") ,pick(28 ,"{?:} :ک ی ت اپ ل ان تخ اب ک رد")!println
                                                                        {
```

This slide should take about 5 minutes.

- Rust ت ای پ T را ب ر اس اس ت ای پ آرگوم ان ه ا و م قدار ب ازگش ت ی اس تن ب اط م ی ک ن د.

- این شبیه به الگوهای در C++ است، اما Rust توابع generic را با الزامات جزئی کامپایل می‌کند، بنابراین آن توابع باید برای تمام تایپ‌هایی که با محدودیت‌ها مطابقت دارند معتبر باشد. به عنوان مثال، می‌توان تابع `pick` را طوری تغییر داد که اگر `n == 0` باشد، مقدار `even + odd` را برگرداند. حتی اگر فقط نمونه‌سازی تابع `pick` با اعداد صحیح استفاده شود، Rust همچنان آن را نامعتبر در نظر می‌گیرد. اما C++ اجازه این کار را به شما می‌دهد.
- کد generic بر اساس محل‌های فراخوانی به کد non-generic تبدیل می‌شود. این یک انتزاع بدون هزینه است: شما دقیقاً همان نتیجه‌ای را دریافت می‌کنید که گویی ساختاره‌ای داده را بدون انتزاع به صورت دستی کدنویسی کرده‌اید.

15.2 دیتا تایپ‌های Generic

می‌توانید از generic ها برای انتزاع نوع در فیلد هشخص استفاده کنید:

```
#[derive(Debug)]
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} و {float}");
    println!("coords: {:?}", integer.coords());
}
```

.This slide should take about 10 minutes

- سوال: چرا T در عبارت `impl<T> Point<T> {}` دوبار هشخص شده است؟ آیا این تکراری نیست؟
  - این به این دلیل است که این یک بخش پیاده‌سازی generic برای تایپ generic است. آن‌ها به‌طور مستقل generic هستند.
  - این به این معناست که این متدها برای هر نوع T تعریف شده‌اند.
  - این امکان وجود دارد که `impl Point<u32> { .. }` را بنویسید.
    `Point` هنوز هم generic است و می‌توانید از `Point<f64>` استفاده کنید، اما متدهای موجود در این بلوک تنها برای `Point<u32>` در دسترس خواهند بود.
- سعی کنید یک متغیر جدید با `let p = Point { x: 5, y: 10.0 };` بسازید. کد را به‌روزرسانی کنید تا نقاطی که دارای عناصر با تایپ‌های مختلف هستند را مجاز کند، با استفاده از دو تایپ متغیر، مانند T و U.

Generic Traits 15.3

Traits نیز می‌توانند generic باشند، درست مانند تایپ و توابع. پارامترهای یک trait زمانی که استفاده می‌شود، تایپ‌های مشخصی پیدا می‌کنند.

```
#[derive(Debug)]
struct Foo(String);

impl From<u32> for Foo {
    fn from(from: u32) -> Foo {
        Foo(format!("{integer: {from از ده شش لدیبت}"))
    }
}

impl From<bool> for Foo {
    fn from(from: bool) -> Foo {
        Foo(format!("{bool: {from از ده‌شلدیبت}"))
    }
}

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

- From trait در ادامه دوره پوشش داده خواهد شد، اما تعریف آن در مستندات std ساده است.
- پیاده‌سازی‌های trait نیازی به پوشش تمام پارامترهای تایپ ممکن ندارند. در اینجا، "Foo::from("hello") کامپایلر نخواهد شد زیرا پیاده‌سازی From<&str> برای Foo وجود ندارد.
- traitهای Generic تایپ‌ها را به‌عنوان "ورودی" می‌پذیرند، در حالی که تایپ طبیعت از "خروجی" هستند. یک trait می‌تواند پیاده‌سازی‌های مختلفی برای تایپ‌های ورودی متفاوت داشته باشد.
- در واقع، Rust نیاز دارد که حداکثر یک پیاده‌سازی از یک trait برای هر تایپ T تطابق داشته باشد. برخلاف برخی زبان‌های دیگر، Rust هیچ قاعده‌ای برای انتخاب "مشخص‌ترین" تطابق را ندارد. در حال حاضر، کارهایی برای اضافه کردن این پشتیبانی وجود دارد که به آن ویژه‌سازی می‌گویند.

Trait Bounds 15.4

هنگام کار با genericها، معمولاً می‌خواهید نیاز داشته باشید که تایپ، trait دیگر خاص را پیاده‌سازی کنند، تا بتوانید متدهای آن trait را فراخوانی کنید.

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
{
```

.This slide should take about 8 minutes

- سعی کنید یک NonClonable بسازید و آن را به duplicate پاس دهید.
- زمانی که چندین دیرت لازم است، از + برای ترکیب آن‌ها استفاده کنید.
- یک عبارت where را نشان دهید، زیرا دانش‌آموزان هنگام خواندن کد با آن مواجه خواهند شد.

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

  - اگر تعداد پارامترها زیاد باشد، استفاده از عبارت where باعث می‌شود که امضای تابع مرتب‌تر و خواناتر باشد.
  - این ویژگی‌های اضافی دارد که آن را قدرتمندتر می‌کند.
    * اگر کسی بپرسد، ویژگی اضافی این است که تایپ در سمت چپ : می‌تواند دلخواه باشد، مانند Option<T>.

- توجه داشته باشید که Rust (هنوز) پشتیبانی از ویژه‌سازی را ندارد. به عنوان مثال، با توجه به اصلی duplicate، اضافه کردن یک پیاده‌سازی ویژه‌شده مانند duplicate(a: u32) برعکس است.

impl Trait 15.5

مشابه با محدودیت‌های trait، می‌توان از impl Trait syntax در آرگومان‌های تابع و مقادیر بازگشتی استفاده کرد:

```
                                    :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
                        ;("{?:debuggable} :اگ‌بی لبا‌ق")!println
                                                    {
```

.This slide should take about 5 minutes

`impl Trait` به شما اجازه می‌دهد با تایپ‌هایی کار کنید که نمی‌توانید نام ببرید. معنی `impl` در موقعیت‌های مختلف کمی متفاوت است.

- برای یک پارامتر، `impl Trait` شبیه به یک پارامتر generic ناشناخته با یک محدودیت trait است.
- برای تایپ بازگشتی، به این معنا است که تایپ بازگشتی تایپ مشخصی است که trait را پیاده‌سازی می‌کند، بدون این‌که تایپ را نام ببرید. این می‌تواند زمانی مفید باشد که نمی‌خواهید تایپ مشخص را در یک API عمومی افشا کنید.

  Inference در موقعیت بازگشتی دشوار است. تابعی که `impl Foo` را برمی‌گرداند، تایپ مشخصی را که برمی‌گرداند انتخاب می‌کند، بدون این‌که آن را به طور صریح در منبع بنویسد. تابعی که generic مانند `collect<B>() -> B` را برمی‌گرداند، می‌تواند هر تایپ که B را برآورده می‌کند بازگرداند، و ممکن است فراخوانی‌کننده نیاز به انتخاب یکی از آن‌ها داشته باشد، مانند `let x: Vec<_> = foo.collect()` یا با استفاده از turbofish، `foo.collect::<Vec<_>>()`.

نوع debuggable چیست؟ سعی کنید .. = let debuggable: ()‘ را امتحان کنید تا بببینید پیام خطا چه چیزی را نشان می‌دهد.

dyn Trait 15.6

علاوه بر استفاده از تری‌ها برای فراخوانی استاتیک از طریق genericها، Rust همچنین از استفاده از آن‌ها برای فراخوانی دینامیک با تایپ‌های حذف‌شده از طریق اشاره‌گر trait پشتیبانی می‌کند:

```
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, my name is {}!", self.name)
    }
}

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

// Uses generics and static dispatch.
fn generic(pet: &impl Pet) {
```

```
    println!("سلام، من یک {} هستم؟", pet.talk());
}

// Uses type-erasure and dynamic dispatch.
fn dynamic(pet: &dyn Pet) {
    println!("سلام، من یک {} هستم؟", 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);
}
```

This slide should take about 5 minutes.

- Genericها، از جمله `impl Trait`، از monomorphization برای ایجاد یک نمونه تخصصی از توابع برای هر تایپ مختلفی که با آن نمونه‌سازی شده استفاده می‌کنند. این بدان معناست که فراخوانی یک متد trait از درون یک تابع generic همچنان از فراخوانی استاتیک استفاده می‌کند، زیرا کامپایلر اطلاعات کامل تایپ را دارد و می‌تواند پیاده‌سازی trait مربوط به تایپ را مشخص کند.
- زمانی که از `dyn Trait` استفاده می‌شود، به‌جای آن از فراخوانی دینامیک از طریق یک virtual method table (vtable) استفاده می‌کند. این بدان معناست که یک نسخه واحد از `fn dynamic` وجود دارد که بدون توجه به تایپ `Pet` که وارد می‌شود، استفاده می‌شود.
- زمانی که از `dyn Trait` استفاده می‌شود، شی trait باید پشت یک تایپ واسط قرار داشته باشد. در این مورد، این تایپ واسط یک ارجاع است، اگرچه تایپ‌های اشاره‌گره‌ای هوشمند مانند `Box` نیز می‌توانند استفاده شوند (این موضوع در روز سوم نشان داده خواهد شد).
- در زمان اجرا، یک `&dyn Pet` به‌صورت یک "اشاره‌گر چاق" (fat pointer) نمایان می‌شود، یعنی یک جفت از دو اشاره‌گر: یکی از اشاره‌گرها به شیء مشخصی که `Pet` را پیاده‌سازی می‌کند اشاره دارد و دیگری به vtable برای پیاده‌سازی تریدی آن نوع اشاره می‌کند. هنگام فراخوانی متد `talk` بر روی `&dyn Pet`، کامپایلر آدرس تابع `talk` را در vtable جست‌وجو کرده و سپس تابع را فراخوانی می‌کند و اشاره‌گر به `Dog` یا `Cat` را به آن تابع پاس می‌دهد. کامپایلر نیازی به دانستن تایپ مشخص `Pet` برای انجام این کار ندارد.
- یک `dyn Trait` به‌عنوان "تایپ حذف شده" (type-erased) در نظر گرفته می‌شود، زیرا دیگر در زمان کامپایل اطلاعاتی درباره تایپ مشخص نداریم.

15.7 تمرین: Generic `min`

در این تمرین کوتاه، شما یک تابع `mingeneric` را پیاده‌سازی خواهید کرد که حداقل از دو مقدار را تعیین می‌کند، با استفاده از trait `Ord`.

```
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
                                                {
```

.This slide and its sub-slides should take about 10 minutes

- trait Ord و enum Ordering را به دانش‌آموزان نشان دهید.

15.7.1 راه‌حل

```
;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
                                                {
```

فصل 16

کتابخانه استاندارد ۱ه‌پای

این بخش بای‌د حدود ۱ ساعت طول بکشد. این شامل:

اسل‌ای‌د	مدت زمان
کتابخانه استاندارد	۳ دقیقه
مستندات	۵ دقیقه
Option	۱۰ دقیقه
Result	۵ دقیقه
String	۵ دقیقه
Vec	۵ دقیقه
HashMap	۵ دقیقه
تمرین: شمارنده	۲۰ دقیقه

برای هر یک از اسل‌ای‌ده‌ای این بخش، کمی زمان صرف مرور صفحات مستندات کنید و برخی از متده‌ای رایج‌تر را برجسته کنید.

16.1 کتابخانه استاندارد

Rust دارای یک کتابخانه استاندارد است که به ایجاد مجموعه‌ای از تایپ‌های رایج استفاده‌شده توسط کتابخانه‌ها و برنامه‌های Rust کمک می‌کند. به این ترتیب، دو کتابخانه می‌توانند به‌راحتی با هم کار کنند زیرا هر دو از تایپ `String` یکسانی استفاده می‌کنند.

در واقع، Rust شامل چندین لایه از کتابخانه استاندارد است: `core`، `alloc` و `std`.

- `core` شامل ابتدایی‌ترین تایپ‌ها و توابع است که به `libc`، تخصیص‌دهنده حافظه یا حتی وجود یک سیستم‌عامل وابسته نیستند.
- `alloc` شامل تایپ‌های است که به یک تخصیص‌دهنده حافظه سراسری نیاز دارند، مانند `Vec`، `Box` و `Arc`.
- برنامه‌های Rust تعبیه‌شده اغلب تنها از `core` و گاهی اوقات از `alloc` استفاده می‌کنند.

16.2 مستندات

Rust دارای مستندات گسترده‌ای است. به عنوان مثال:

- تمام جزئیات مربوط به حل‌ه‌ا.
- تای‌پ‌های ابتدایی مانند u8.
- تای‌پ‌های کتابخانه استاندارد مانند Option یا BinaryHeap.

در واقع، شما می‌توانید کد خود را مستند کنید:

```
.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
                                                        {
```

مستندات به‌عنوان Markdown پردازش می‌شوند. تمام crate‌های کتابخانه‌ای من‌تش‌رش‌ده Rust به‌طور خودکار در docs.rs با استفاده از ابزار rustdoc مستند می‌شوند. مستند کردن تمام آیتم‌های عمومی در یک API با استفاده از این الگو به‌طور رایج مرسوم است.

برای مستند کردن یک آیتم از درون خود آیتم (مانند درون یک ماژول)، از `//!` یا `/*! .. */` استفاده کنید که به آن "کامنت‌های مستندات داخلی" می‌گویند:

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

.This slide should take about 5 minutes

- مستندات تولیدشده برای rand crate را در https://docs.rs/rand به دانش‌آموزان نشان دهید.

Option 16.3

ما قبلاً برخی استفاده‌ها از `Option<T>` را مشاهده کرده‌ایم. این تای‌پ یا مقداری از تای‌پ T را ذخیره می‌کند یا هیچ چیزی را ذخیره نمی‌کند. به عنوان مثال، `String::find` یک `Option<usize>` را برمی‌گرداند.

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

.This slide should take about 10 minutes

- `Option` به‌طور گسترده‌ای استفاده می‌شود و تنها در کتابخانه استاندارد محدود نمی‌شود.
- `unwrap` مقدار موجود در یک `Option` را برمی‌گرداند یا باعث `panic` می‌شود. `expect` مشابه است اما پیامی برای خطا می‌پذیرد.
  - می‌توانید در مواجهه با `panic None` کنید، اما نمی‌توانید به‌طور "تصادفی" فراموش کنید که `None` بررسی کنید.
  - استفاده از `unwrap/expect` در همه‌جا نشانه سریع چیزی رایج است، اما کد تولیدی معمولاً `None` را به‌شیوه‌ای مناسب‌تر مدیریت می‌کند.
- بهینه‌سازی niche به این معناست که `Option<T>` اغلب اندازه‌ای مشابه با T در حافظه دارد.

Result 16.4

Result مشابه Option است، اما موفقیت یا شکست یک عملیات را نشان می‌دهد، هرکدام با یک نوع متغیر enum متفاوت. این نوع جنریک است: Result<T, E> که در آن T در متغیر Ok ذخیره می‌شود و E در متغیر Err ظاهر می‌شود.

```
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!("فایل خاطرات خوانده شد: {bytes} بایت\n{contents}");
            } else {
                println!("فایل خاطرات خوانده شد اما نمی‌توان به عنوان متن خواند");
            }
        }
        Err(err) => {
            println!("خطا در باز شدن فایل خاطرات: {err}");
        }
    }
}
```

This slide should take about 5 minutes.

- Option نشان‌دهنده مقدار موفقیت‌آمیز درون Result قرار دارد و توسعه‌دهنده را ملزم به استخراج صریح آن می‌کند. این به بررسی خطاها تشویق می‌کند. در صورتی که خطا هرگز نباید رخ دهد، می‌توان از unwrap() یا expect() استفاده کرد که این نیز نشان‌دهنده نیت توسعه‌دهنده است.
- مستندات Result اطلاعات خوبی هستند. نه در طول دوره، اما ذکر آن ارزشمند است. این مستندات شامل بسیاری از متدها و توابع کاربردی است که به برنامه‌نویسی به استایل تابع-محور کمک می‌کنند.
- Result نوع استاندارد برای پیاده‌سازی مدیریت خطاها است که در روز چهارم دوره خواهیم دید.

String 16.5

String یک رشته قابل رشد با کدگذاری UTF-8 است:

```
fn main() {
    let mut s1 = String::new();
    s1.push_str("سلام");
    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 پیاده‌سازی‌کننده <Deref<Target = str> است، که به این معناست که می‌توانید تمام متدهای str را بر روی String فراخوانی کنید.

.This slide should take about 5 minutes

- String::new یک رشته جدید خالی برمی‌گرداند. از String::with_capacity استفاده کنید زمانی که می‌دانید چقدر داده می‌خواهید به رشته اضافه کنید.
- String::len اندازه رشته String را به‌صورت بایت برمی‌گرداند (که ممکن است با طول آن به‌صورت کاراکتر متفاوت باشد).
- String::chars یک تکرارگر (iterator) از روی کاراکترهای واقعی برمی‌گرداند. توجه داشته باشید که یک char ممکن است با آنچه که یک انسان به عنوان "کاراکتر" در نظر می‌گیرد، متفاوت باشد به دلیل grapheme clusters.
- زمانی که مردم به رشته‌ها اشاره می‌کنند، ممکن است منظورشان &str یا String باشد.
- زمانی که یک تایپ، Deref<Target = T> را پیاده‌سازی می‌کند، کامپایلر به شما این امکان را می‌دهد که به‌طور شفاف متدهای T را فراخوانی کنید.
  - ما هنوز Deref trait را بررسی نکرده‌ایم، بنابراین در این مرحله این بی‌شتر توضیح‌دهنده ساختار نوار کناری در مستندات است.
  - String پیاده‌سازی‌کننده Deref<Target = str> است که به‌طور شفاف دسترسی به متدهای str را فراهم می‌کند.
  - let s3 = s1.deref(); و let s3 = &*s1; بنویسید و مقایسه کنید .
- بنویسید و مقایسه کنید let s3 = s1.deref(); و let s3 = &*s1;.
- راه‌های مختلف برای ایندکس‌گذاری یک String را مقایسه کنید:
  - به یک کاراکتر با استفاده از s3.chars().nth(i).unwrap()، جایی که i در محدوده است یا خارج از محدوده.
  - به یک زیررشته با استفاده از s3[0..4]، جایی که این برش در مرزهای کاراکترها است یا نباشد.
- بسیاری از تایپ‌ها می‌توانند با استفاده از متد to_string به رشته تبدیل شوند. این تریت به‌طور خودکار برای تمام تایپ‌هایی که Display را پیاده‌سازی می‌کنند، پیاده‌سازی شده است، بنابراین هر چیزی که می‌تواند قالب‌بندی شود، همچنین می‌تواند به رشته تبدیل شود.

Vec 16.6

این Vec با فرقابل تغییر اندازه و heap-allocated است:

```
                                                        } ()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
{
```

Vec پیاده‌سازی‌کننده Deref<Target = [T]> است، به این معنی که می‌توانید متدهای برش را بر روی یک Vec فراخوانی کنید.

.This slide should take about 5 minutes

- Vec نوعی مجموعه است، به همراه String و HashMap. داده‌های آن در حافظه heap ذخیره می‌شوند. به این معنی که مقدار داده‌ها نیازی به دانستن در زمان کامپایل ندارد و می‌توانند در زمان اجرا رشد یا کوچک شوند.
- توجه داشته باشید که Vec<T> نیز یک تایپ generic است، اما نیازی به تعیین صریح T ندارید. همان‌طور که همیشه با استنتاج تایپ در Rust، T در زمان اولین فراخوانی push مشخص شده است.
- vec![...] یک ماکرو است‌اندارد برای استفاده به‌جای Vec::new() است و از افزودن عناصر اولیه به vector پشتیبانی می‌کند.
- برای ایندکس‌گذاری vector از [ ] استفاده می‌کنید، اما اگر از محدوده خارج شود، باعث panic می‌شود. به‌طور جایگزین، استفاده از get یک Option را برمی‌گرداند. تابع pop آخرین عنصر را حذف می‌کند.
- برش‌ها در روز سوم پوشش داده می‌شوند. در حال حاضر، دانش‌آموزان تنها باید بدانند که یک مقدار از تایپ Vec به تمام متدهای مستند شده برش‌ها نیز دسترسی دارد.

HashMap 16.7

نقشه hash است‌اندارد با حفاظت در برابر حملات HashDoS:

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

} ()fn main
    ;()let mut page_counts = HashMap::new
    ;(207 ,"نیک فیلم‌های ابرقهرمانی")page_counts.insert
    ;(751 ,"زمین گیری در ورزشگاه")page_counts.insert
    ;(303 ,"غرور و تعصب")page_counts.insert

} ("if !page_counts.contains_key("Les Misérables
        )!println
"درباره {} کتاب می‌دانیم، اما Les Misérables نه.",
        ()page_counts.len
        ;(
        {

} ["بیگانه نیمرز در سیاره‌ای دوردست" ,"غرور و تعصب"] for book in
        } (match page_counts.get(book
    ,("صفحه‌ها {Some(count) => println!("{book}: {count
        ,(".است ناشناخته {None => println!("{book
        {
        {

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

```
for book in ["غرور و تعصب", "ماجراجویی آلیس در سرزمین عجایب"] {
    let page_count: &mut i32 = page_counts.entry(book).or_insert(0);
    *page_count += 1;
}

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

This slide should take about 5 minutes.

- HashMap در prelude تعریف نشده و باید به scope وارد شود.
- سطرهای کد زیر را امتحان کنید. سطر اول بررسی می‌کند که آیا یک کتاب در HashMap وجود دارد یا خیر و اگر وجود نداشت، یک مقدار جایگزین برمی‌گرداند. سطر دوم مقدار جایگزین را در HashMap وارد می‌کند اگر کتاب پیدا نشد.

```
let pc1 = page_counts
    .get("هری پاتر و سنگ جادو")
    .unwrap_or(&336);
let pc2 = page_counts
    .entry("The Hunger Games".to_string())
    .or_insert(374);
```

- برخلاف `vec!`، متأسفانه ماکروی استاندارد `hashmap!` وجود ندارد.
  - از نسخه Rust 1.56 به بعد، HashMap پیاده‌سازی‌کننده `From<[(K, V); N]>` است که به ما اجازه می‌دهد به‌راحتی یک HashMap را از یک آرایه مقداردهی اولیه کنیم:

```
let page_counts = HashMap::from([
    ("هری پاتر و سنگ جادو".to_string(), 336),
    ("The Hunger Games".to_string(), 374),
]);
```

- به‌طور جایگزین، HashMap می‌تواند از هر `Iterator` که جفت‌های key-value را تولید می‌کند، ساخته شود.
- ما `HashMap<String, i32>` را نمایش می‌دهیم و از استفاده از `&str` به‌عنوان کلید اجتناب می‌کنیم تا مثال‌ها ساده‌تر شوند. استفاده از ارجاعات در مجموعه‌ها البته ممکن است، اما می‌تواند به مشکلاتی با borrow checker منجر شود.
  - حذف `to_string()` از مثال بالا را امتحان کنید و ببینید آیا هنوز کامپایل می‌شود یا خیر. فکر می‌کنید ممکن است با چه مشکلاتی مواجه شویم؟
- این چندین تایپ "تایپ بازگشتی خاص متد" دارد، مانند `std::collections::hash_map::Keys`. این تایپ‌ها معمولاً در جستجوهای مستندات Rust ظاهر می‌شوند. مستندات این تایپ را به دانش‌آموزان نشان دهید و پیوند مفید بازگشتی به متد `keys` را نیز نمایش دهید.

16.8 تمرین: شمارنده

در این تمرین، شما یک ساختار داده بسیار ساده را به‌صورت generic خواهید کرد. این ساختار از `std::collections::HashMap` برای پیگیری اینکه چه مقادیری مشاهده شده‌اند و هرکدام چند بار ظاهر شده‌اند، استفاده می‌کند.

نسخه اولیه `Counter` به‌طور سخت‌افزاری برای مقادیر `u32` کدگذاری شده است. ساختار و متدهای آن را به‌صورت generic بر اساس تایپ مقداری که در حال پیگیری است، تغییر دهید، به‌طوری که `Counter` بتواند هر تایپ مقداری را پیگیری کند.

اگر زود تمام کردید، سعی کنید از متد entry استفاده کنید تا تعداد جستجوهای مورد نیاز برای پیاده‌سازی متد count را به نصف کاهش دهید.

```rust
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 {
        println!("مقادیر {} برابر با {} را دیده‌شده", i, ctr.times_seen(i));
    }

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

```
{
```

16.8.1 راه‌حل

```
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 {
        println!("عدد {} به دفعات {} دیده شده است", i, ctr.times_seen(i));
    }

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

فصل 17

کتابخانه استاندارد Traits

این بخش بایدحدود ۱ ساعت و ۱۰ دقیقه طول بکشد. این بخش شامل موارد زیر است:

اسلاید	مدت زمان
مقایسه	۵ دقیقه
اپراتورها	۵ دقیقه
From and Into	۵ دقیقه
Casting	۵ دقیقه
Read and Write	۵ دقیقه
Default, struct update syntax	۵ دقیقه
Closures	۱۰ دقیقه
تمرین: ROT13	۳۰ دقیقه

همانند تایپ‌های موجود در کتابخانه استاندارد، زمانی را صرف مرور مستندات هرtrait کنید.

این بخش طولانی است. در میانه‌ی آن یک استراحت کنید.

17.1 مقایسه

این traitها از مقایسه بین مقادیر پشتیبانی می‌کنند. همه‌ی این traitها را می‌توان برای تایپ‌هایی که شامل فیلدهایی هستند که این traitها را پیاده‌سازی می‌کنند، به‌دست آورد.

PartialEq and Eq

PartialEq یک رابطه همارزی جزئی است که دارای متد الزامی eq و متد ارائه‌شده ne می‌باشد. عملگرهای == و =! این متدها را فراخوانی می‌کنند.

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

```
        self.id == other.id
    }
}
```

Eq یک رابطه هم‌ارزی کامل است (بازتابی، متقارن، و transitive) و شامل PartialEq می‌شود. تو این‌طوری که به هم‌ارزی کامل نیاز دارند، از Eq به‌عنوان یک trait bound استفاده می‌کنند.

PartialOrd and Ord

PartialOrd یک ترتیب جزئی را تعریف می‌کند و دارای متد partial_cmp است. این ویژگی برای پیاده‌سازی عملگرهای <، <=، >= و > استفاده می‌شود.

```
use std::cmp::Ordering;
#[derive(Eq, PartialEq)]
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,
        }
    }
}
```

Ord یک ترتیب کامل است که در آن متد cmp مقدار Ordering را برمی‌گرداند.

.This slide should take about 5 minutes

PartialEq می‌تواند بین تایپ‌های مختلف پیاده‌سازی شود، اما Eq نمی‌تواند، زیرا بازتابی است:

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

در عمل، معمولاً این traitها به‌طور خودکار به‌دست می‌آیند، اما کمتر پیش می‌آید که آن‌ها به‌طور دستی پیاده‌سازی شوند.

17.2 اپراتورها

بارگذاری مجدد عملگرها از طریق traits در std::ops پیاده‌سازی شده است:

```
#[derive(Debug, Copy, Clone)]
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
                                                {
```

.This slide should take about 5 minutes

نکات بحث:

- می‌توانید Add را برای &Point پیاده‌سازی کنید. در چه موقعیت‌هایی این کار مفید است؟
  - پاسخ: Add:add خود self را مصرف می‌کند. اگر تایپ T که برای آن عملگر را بارگذاری می‌کنید، Copy نباشد، باید پیاده‌سازی عملگر را برای &T نیز در نظر بگیرید. این کار از ایجاد کپی‌های غیرضروری در محل فراخوانی جلوگیری می‌کند.
- چرا Output یک تایپ مرتبط است؟ آیا می‌توان آن را به‌عنوان یک پارامتر تایپ برای متد تعریف کرد؟
  - پاسخ کوتاه: پارامترهای تایپ توسط فراخوانی‌کننده کنترل می‌شوند، اما تایپ‌های مرتبط (مانند Output) توسط پیاده‌سازی trait کنترل می‌شوند.
- شما می‌توانید Add را برای دو تایپ مختلف پیاده‌سازی کنید، به‌عنوان مثال impl Add<(i32, i32)> for Point می‌تواند یک tuple را به یک Point اضافه کند.

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.

From and Into 17.3

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 به‌طور خودکار زمانی پیاده‌سازی می‌شود که From پیاده‌سازی شده باشد:

```
} ()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
{
```

.This slide should take about 5 minutes

- به همین دلیل معمولاً From ها نسبت به Into ها پیاده‌سازی می‌شوند، زیرا تایپ شما به‌طور خودکار پیاده‌سازی Into را نیز دریافت می‌کند.
- هنگام اعلام تایپ ورودی معادل توابع می‌توانند به هر چیزی که می‌توان به یک String تبدیل شود”، قاعده برعکس است، باید از Into استفاده کنید. توابع شما تایپ‌های را قبول می‌کنند که پیاده‌سازی From دارند و همچنین تایپ‌هایی که قطع Into را پیاده‌سازی کرده‌اند.

Casting 17.4

Rust هیچ *implicit* ندارد، اما از تبدیل‌های صریح با استفاده از as پشتیبانی می‌کند. این تبدیل‌ها معمولاً پیرو معنای C هستند که در آنجا تعریف شده‌اند.

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

تبدیل استفاده از as همیشه در Rust تعریف شده و در تمامی پلتفرم‌ها ثابت هستند. این ممکن است با شهود شما برای امعال ریخته‌گری تایپ کوچکتر مطابقت نداشته باشد -- مستندات را بررسی کنید و برای وضوح بیشتر نظر دهید.

می‌توانند از آن آسان است و تبدیل تایپ‌ها با استفاده از as ابزاری نسبتاً حساس است که استفاده نادرست از تبدیل تایپ‌های ریخته‌گری از اشکالات ظریف باشد، به خصوص زمانی که کار نگه‌داری آینده باعث ریخته‌گری‌های مورد استفاده امنی در تایپ‌ها شود. تبدیل‌های بهتر است زمانی استفاده شوند که صریحاً نشان دادن برش بدون قید و شرط باشد (مثلاً انتخاب 32 بیت پایین از یک u64 با as u32، بدون توجه به آنچه در بیت‌های بالا وجود دارد).

برای تبدیل‌های بدون خطا (مانند تبدیل u32 به u64)، استفاده از From یا Into بر as ارجحیت دارد شود که تبدیل در واقع بدون خطا است. برای تبدیل‌های با احتمال خطا، TryFrom و TryInto در دسترس هستند وقتی که می‌خواهید تبدیل‌هایی را که به شیوه‌ای متفاوت از آنهایی که مطابقت ندارند، مدیریت کنید.

.This slide should take about 5 minutes

در نظر داشته باشید که پس از این اسلایدها استراحت کنید.

as مشابه به static_cast در C++ است. استفاده از as در مواردی که ممکن است داده‌ها از دست بروند، معمولاً توصیه نمی‌شود حداقل نیازی به توضیحی کامنتی دارد.

این موضوع در تبدیل اعداد صحیح به usize برای استفاده به عنوان ایندکس رایج است.

Read and Write 17.5

با استفاده از Read و BufRead، می‌توانید بر روی منابع u8 انتزاع کنید:

```
;{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
                                        {
```

به طور مشابه، Write به شما امکان می‌دهد که بر روی عبارت u8 انتزاع کنید:

```
                                ;{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
                                      ;?("سلام" ,log(&mut buffer
                                      ;?("دنیا" ,log(&mut buffer
                                    ;(println!("Logged: {:?}", buffer
                                                        ())Ok
                                                        {
```

The `Default` Trait 17.6

ویژگی Default یک مقدار پیش‌فرض برای یک نوع داده‌ای پیاده‌سازی می‌کند.

```
                                        [(derive(Debug, Default]#
                                                } struct Derived
                                                    ,x: u32
                                                  ,y: String
                                                ,z: Implemented
                                                        {

                                                    [(derive(Debug]#
                                            ;(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
{
```

.This slide should take about 5 minutes

- این ویژگی می‌تواند به طور مستقیم پیاده‌سازی شود یا می‌تواند از طریق `#[derive(Default)]` به صورت خودکار تولید شود.
- یک پیاده‌سازی خودکار، مقداری تولید می‌کند که در آن تمام فیلدهای مقدار به پیش‌فرض خود تنظیم شده‌اند.
  - این بدان معناست که تمام انواع داخل ساختار نیز باید `Default` را پیاده‌سازی کنند.
- انواع‌های استاندارد Rust اغلب `Default` را با مقادیر پیاده‌سازی می‌کنند (مثل `0`، `""` و غیره).
- مقدار داده‌ی جزئی با `Default` به خوبی کار می‌کند.
- کتابخانه استاندارد Rust آگاه است که تایپ‌های مختلف می‌توانند `Default` را پیاده‌سازی کنند و روش‌های کمکی را فراهم می‌کند که از آن استفاده می‌کنند.
- سینتکس `..` به نام سینتکس به‌روزرسانی ساختار شناخته می‌شود.

Closures 17.7

با استفاده از عبارات لامبدا، آن‌ها پیاده‌سازی‌های داخلی دارند که نمی‌توان نام‌گذاری کرد. با این حال، آن‌ها traits `Fn`، `FnMut` و `FnOnce` هستند:

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

                    } ()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
                                        ;{
            ;(4 ,"{} :نتیجه" ,apply_and_log(&mut accumulate
            ;(5 ,"{} :نتیجه" ,apply_and_log(&mut accumulate
```

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

.This slide should take about 10 minutes

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

FnMut یک زیرتایپ از Fn است. نیز یک زیرتایپ از FnMut و FnOnce است. به عبارت دیگر، می‌توانید از FnMut در جایی که FnOnce نیاز است استفاده کنید و از Fn در جایی که FnMut یا FnOnce نیاز است استفاده کنید.

زمانی که تابع تعریف می‌کنید که یک closure را می‌گیرد، باید از FnOnce استفاده کنید اگر قطع آن را فراخوانی می‌کنید (یعنی یک بار استفاده می‌شود)، یا از FnMut در غیر این صورت، و در نهایت از Fn. این کار بیشترین انعطاف‌پذیری را برای فراخوانی‌کننده فراهم می‌کند.

در مقابل، زمانی که یک closure دارید، بیشترین انعطاف‌پذیری که می‌توانید داشته باشید Fn است (که می‌تواند در هر جایی استفاده شود)، سپس FnMut و در نهایت FnOnce.

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

به صورت پیش‌فرض، بسته‌بندی‌ها (closures) هر متغیر از یک دامنه بیرونی را با کمترین سطح دسترسی ممکن (با ارجاع مشترک اگر ممکن باشد، سپس ارجاع انحصاری، سپس با انتقال) capture می‌کنند. کلیدواژه move یا انتقال، capture را به صورت value اجباری می‌کند.

```
} (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
                                                    {
```

17.8 تمرین: ROT13

در این مثال، شما الگوریتم کلاسیک رمزگذاری "ROT13" را پیاده‌سازی خواهید کرد. این کد را به محیط Playground کپی کرده و بخش‌های ناقص آن را پیاده‌سازی کنید. تنها حروف الفبای ASCII را بچرخانید تا نتیجه همچنان UTF-8 معتبر باقی بماند.

```
;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
                                                                {

                                                     [(cfg(test]#
                                                       } mod test
                                                   ;*::use super

                                                         [test]#
                                                      } ()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
                                                                    {

                                                         [test]#
                                                    } ()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
                                                          {
                                                              {
                                                                  {
                                                                      {
```

چه اتفاقی می‌افتد اگر دو نمونه از RotDecoder را به هم متصل کنیم که هر کدام ۱۳ را رتکارا برچخانند؟

17.8.1 راه‌حل

```
                                                    ;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
                                                                              {

                                                                  [(cfg(test]#
                                                                    } mod test
                                                                ;*::use super

                                                                       [test]#
                                                                  } ()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
                                                                          {

                                                                       [test]#
                                                                } ()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
                                                                      {
                                                                          {
                                                                              {
                                                                                  {
```

بخش V

روز ۳: صبح

فصل 18

به روز ۳ خوش آمدید

امروز، ما به بررسی خواهیم پرداخت:

- مدیریت حافظه، طول عمرها و بررسی‌کننده قرض‌گیری (borrow checker): چگونه زبان Rust از ایمنی حافظه اطمینان حاصل می‌کند.
- اشاره‌گر هوشمند: تای‌په‌ای اشاره‌گر در کتابخانه استاندارد.

برنامه زمانی

با احتساب استراحت‌های ۱۰ دقیقه‌ای، این جلسه باید حدود ۲ ساعت و ۲۰ دقیقه طول بکشد. این جلسه شامل:

بخش	مدت زمان
خوش آمدید	۳ دقیقه
مدیریت حافظه	۱ ساعت
اشاره‌گرهای هوشمند	۵۵ دقیقه

فصل 19

مدیریت حافظه

این بخش بایدها حدود ۱ ساعت طول بکشد. این شامل:

ایدال‌س	مدت زمان
بررسی حافظه برنامه‌ها	۵ دقیقه
رویکردهای مدیریت حافظه	۱۰ دقیقه
مالکیت	۵ دقیقه
مفاهیم جابه‌جایی	۵ دقیقه
Clone	۲ دقیقه
کپی کردن تایپ‌ها	۵ دقیقه
Drop	۱۰ دقیقه
تمرین: تایپ‌های سازنده	۲۰ دقیقه

19.1 بررسی حافظه برنامه‌ها

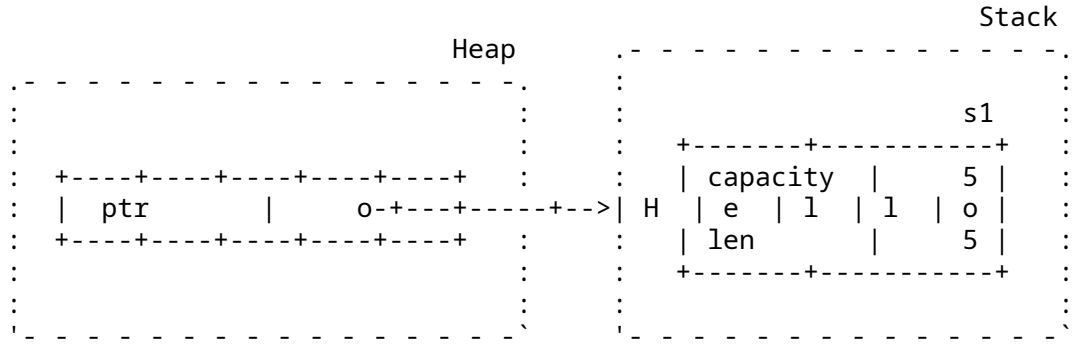
برنامه‌ها حافظه را به دو روش تخصیص می‌دهند:

- Stack: بلوک پیوسته‌ای از حافظه که برای متغیرهای محلی (داخل یک تابع) استفاده می‌شود.
  - مقدار دارای اندازه‌ای ثابت هستند که در زمان کامپایل شناخته می‌شوند.
  - بسیار سریع: فقط یک اشاره‌گر stack را جابجا کنید.
  - مدیریت آسان: پیروی فراخوانی‌های تابع است.
  - به‌روزی عالی از حافظه.
- Heap: ذخیره‌سازی مقدار خارج از فراخوانی‌های تابع.
  - مقدار دارای اندازه‌ای پویا هستند که در زمان اجرا تعیین می‌شوند.
  - کمی کندتر از stack: نیاز به برخی از عملیات‌های مدیریتی دارد.
  - هیچ تضمینی برای به‌روزی بالا از حافظه ندارند.

مثال

ساختن یک `String` metadata با اندازه ثابت را روی stack و داده با اندازه پویا، یعنی رشته واقعی، را روی heap قرار می‌دهد:

```
} ()fn main
;("سلام")let s1 = String::from
{
```



.This slide should take about 5 minutes

- ذکر کنید که یک String توسط یک Vec پشتیبانی می‌شود، بنابراین دارای ظرفیت و طول است و در صورت تغییرپذیری، می‌تواند از طریق اختصاص مجدد حافظه روی heap رشد کند.
- اگر دانش‌آموزان درباره آن سوال کنند، می‌توانید اشاره کنید که حافظه زیرین با استفاده از System Allocator بر روی heap اختصاص داده شده و تخصیص‌دهنده‌ای سفارشی می‌تواند با استفاده از Allocator API پیاده‌سازی شود

برای کاوش بیشتر

می‌توانید با استفاده از Rust ناامن (unsafe) نحوه چیدمان حافظه را بررسی کنید. با این حال، باید اشاره کنید که این کار به درستی ناامن است!

```
} ()fn main
;("سلام")let mut s1 = String::from
;(' ')s1.push
;("دنیا")s1.push_str
.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 حافظه مدیریت‌شده روی

به طور سنتی، زبان‌ها به دو دسته تقسیم شده‌اند:

- کنترل کامل از طریق مدیریت دستی حافظه: C، C++، پاسکال، ...
  - برنامه‌نویس می‌بایست به صراحت حافظه heap را آزاد کند.
  - برنامه‌نویس باید تعیین کند که آیا یک اشاره‌گر به موقعیت حافظه اشاره می‌کند یا نه.
  - اطلاعات نشان می‌دهد که برنامه‌نویسان اشتباهاتی مرتکب می‌شوند.
- ایمنی کامل از طریق مدیریت خودکار حافظه در زمان اجرا: جاوا، پایتون، گو، هسکل، ...

– یک سیستم زمان اجرا اطمینان می‌یابد که حافظه تا زمانی که دیگر نتوانند به آن ارجاع داده شود، آزاد نمی‌شود.
– Typically implemented with reference counting or garbage collection.

Rust یک ترکیب جدید ارائه می‌دهد:

کنترل کامل و ایمنی از طریق اجرای صحیح مدیریت حافظه در زمان کامپایل.

این کار را با استفاده از مفهوم مالکیت صریح انجام می‌دهد.

This slide should take about 10 minutes.

این اسلاید به منظور کمک به دانش‌آموزانی است که از زبان‌های دیگر می‌آیند تا Rust را در زمینه مناسب قرار دهند.

- C باید حافظه heap را به‌طور دستی با استفاده از `malloc` و `free` مدیریت کند. خطاهای رایج شامل فراموش کردن فراخوانی `free`، فراخوانی آن چندین بار برای یک اشاره‌گر، یا dereference کردن یک اشاره‌گر پس از آزاد شدن حافظه‌ای است که به آن اشاره می‌کند.
- C++ ابزارهایی مانند اشاره‌گرهای هوشمند (`unique_ptr`, `shared_ptr`) دارد که از تضمین‌های زبانی درباره فراخوانی ویرانگرها (destructor) برای اطمینان از آزاد شدن حافظه هنگام بازگشت از تابع استفاده می‌کنند. با این حال، هنوز هم بسیار آسان است که از این ابزارها به اشتباه استفاده کرده و باگ‌هایی مشابه به C ایجاد کرد.
- جاوا، گو و پایتون به جمع‌آوری‌کننده زباله (garbage collector) برای شناسایی حافظه‌ای که دیگر در دسترس نیست و دور ریختن آن متکی هستند. این امر تضمین می‌کند که هر اشاره‌گری می‌تواند dereference شود و از بروز خطاهای استفاده پس از آزادسازی (use-after-free) و سایر دست‌درازی‌های باگ جلوگیری می‌کند. اما، GC هزینه‌ای در زمان اجرا دارد و تنظیم مناسب آن دشوار است.

مدل مالکیت و قرضگیری Rust (ownership and borrowing) می‌تواند در بسیاری از موارد عملکرد C را با عملیات‌های تخصیص و آزادسازی دقیقاً در مکان‌های مورد نیاز -- با هزینه صفر -- به دست آورد. همچنین ابزارهایی مشابه به اشاره‌گرهای هوشمند C++ را فراهم می‌کند. در صورت نیاز، گزینه‌های دیگری مانند شمارش ارجاع نیز در دسترس هستند و حتی crates شخص ثالثی برای پشتیبانی از جمع‌آوری زباله در زمان اجرا موجود است (که در این کلاس پوشش داده نمی‌شود).

19.3 مالکیت

تمام پیوندهای متغیر دارای یک دامنه هستند که در آن معتبر هستند و استفاده از متغیر خارج از دامنه‌اش یک خطاست:

```
;(struct Point(i32, i32

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

می‌گوییم که متغیر مالک مقدار است. هر مقدار در Rust در هر لحظه دقیقاً یک مالک دارد.

در پایان دامنه، متغیر حذف می‌شود و داده‌ها آزاد می‌شوند. یک ویرانگر (destructor) می‌تواند در اینجا اجرا شود تا منابع را آزاد کند.

This slide should take about 5 minutes.

دانش‌آموزانی که با پیاده‌سازی‌های جمع‌آوری زباله آشنا هستند، خواهند دانست که یک جمع‌آوری‌کننده زباله با مجموعه‌ای از "ریشه‌ها" برای یافتن تمام حافظه‌های قابل دسترسی شروع می‌کند. اصول "مالکیت تک‌گانه" Rust ایده مشابهی است.

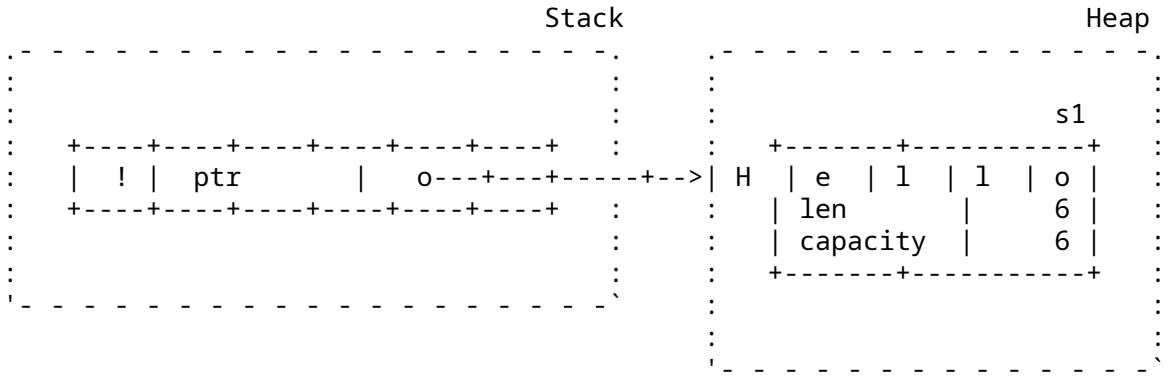
19.4 مفاهیم جایگزین

انتساب، مالکیت را بین متغیرها می‌گرداند:

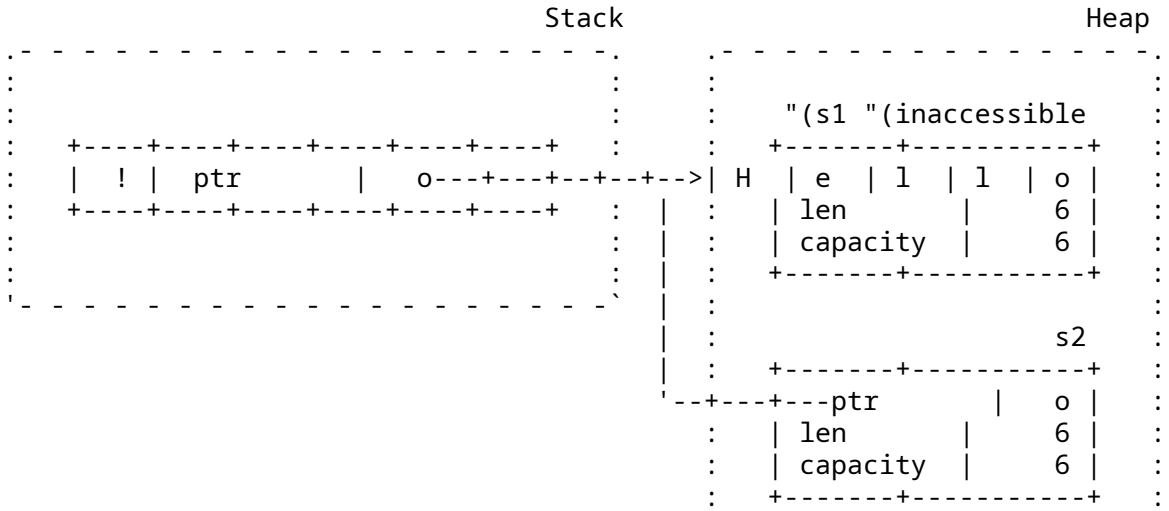
```
fn main() {
    let s1: String = String::from("سلام!");
    let s2: String = s1;
    println!("s2: {s2}");
    println!("s1: {s1}"); //
}
```

- انتساب s1 به s2 مالکیت را می‌گرداند.
- زمانی که دیگر در اسکوپ s1 نیستیم، هیچ اتفاقی نمی‌افتد: چون s1 مالک چیزی نیست.
- زمانی که دیگر در اسکوپ s2 نیستیم، داده‌های رشته آزاد می‌شوند.

قبل از انتقال به s2:



بعد از انتقال به s2:



هنگامی که یک مقدار را به یک تابع منتقل می‌کنید، مقدار به آرگمان تابع اختصاص داده می‌شود. به این شکل مالکیت را منتقل می‌کنند:

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

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

.This slide should take about 5 minutes

- اشاره کنید که این رویه راست بر خلاف پیش‌فرض زبان C++ است که در آن مقدار کپی می‌شود مگر که از `std::move` استفاده کنیم (تا یک مقدار را جا به جا کنیم!)
- این رویه فقط برای انتقال مالکیت است. این‌که آیا هیچ کد ماشینی برای دستکاری خود داده‌ها تولید می‌شود یا خیر، موضوعی برای بهینه‌سازی است و چنین کپی‌هایی به‌طور تهاجمی (aggressively) بهینه‌سازی می‌شوند.
- مقادیر ساده (مانند اعداد صحیح) را می‌توان Copy کرد (اساساً ارزان‌تر از ارجاع به آن‌ها).
- در Rust، کلون‌ها واضح بیان می‌شوند (با استفاده از `clone`).

در مثال `say_hello`:

- با اولین فراخوانی `say_hello`، تابع `main` مالکیت `name` را انتقال می‌دهد. پس از آن، `name` دیگر نمی‌تواند در `main` استفاده شود.
- حافظه این‌باشت اختصاص داده شده برای `name` در انتهای تابع `say_hello` آزاد خواهد شد.
- تابع `main` می‌تواند مالکیت `name` را حفظ کند اگر آن را به عنوان یک مرجع (&name) منتقل کند و صد البته که `say_hello` یک مرجع را به عنوان پارامتر بایرد بپذیرد.
- به عنوان گزینه دیگر، `main` می‌تواند یک کلون از `name` را در فراخوانی اولیه تابع ای که در نظر دارید (`name.clone()`) منتقل کند.
- در زبان Rust سخت‌تر از C++ است که سه‌وا کپی ایجاد کنید, زیرا به صورت پیش‌فرض از مفهوم «انتقال» استفاده می‌کنیم و برنامه نویس مجبور است هر جا که لازم هست به صورت صریح کلون را ایجاد کند.

برای کاوش بیشتر

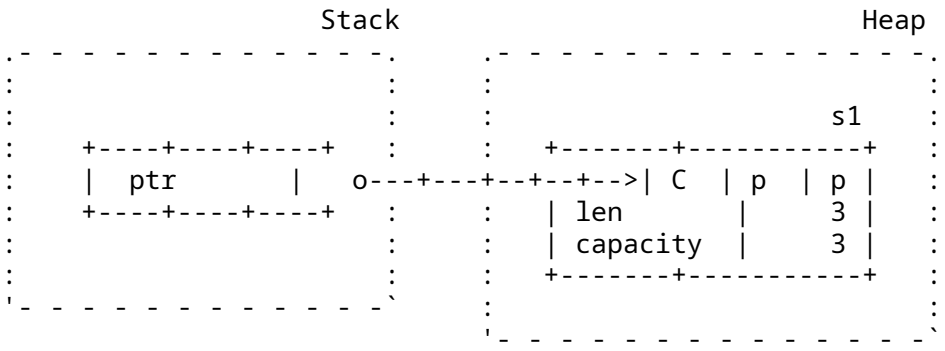
نسخه‌های دفاعی در C++ مدرن

C++ مدرن این مشکل را به شیوه متفاوتی حل می‌کند:

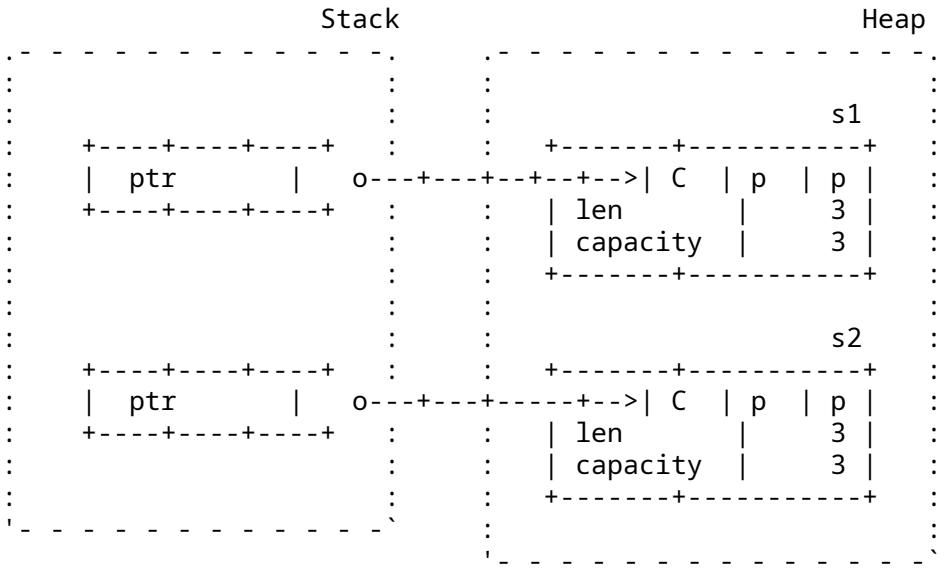
```
;"std::string s1 = "Cpp
.std::string s2 = s1;  // Duplicate the data in s1
```

- داده‌های انباشت از داده‌های `s1` یک کپی برابر اصل برای `s2` گرفته می‌شود که این کپی به صورت مستقل است.
- حالا هر موقع `s1` یا `s2` از اسکوپ مورد نظرشون خارج شوند هر کدام به صورت جداگانه‌ای حافظه خود را آزاد می‌کنند.

قبل از انتساب همراه کپی:



بعد از انتساب همراه کپی:



نکات کلیدی:

- زبان C++ انتخاب کمی متفاوت نسبت به زبان Rust انجام داده است. زیرا = داده‌ها را کپی می‌کند، داده‌های رشته بایدکلون شوند. در غیر این صورت، در موقع از اسکوپ یکی از آن‌ها خارج شویم امکان به وجود آمدن اشتباه آزادسازی مجدد حافظه رخ دهد.
- البته که زبان C++ دارای `std::move` است که برای انتقال یک متغیر استفاده می‌شود. اگر مثال ما `s2 = std::move(s1)` بود هیچ تخصیص انباشتی صورت نمی‌گرفت بلکه `s1` در یک وضعیت معتبر البته نامشخص قرار می‌گرفت و برخلاف زبان Rust, توی زبان C++ برنامه‌نویس مجاز است که دوباره از `s1` استفاده کند.
- برخلاف Rust، = در C++ می‌تواند برای کپی کردن و هم انتقال دادن استفاده شود.

19.5 Clone

گاهی اوقات شما می‌خواهید یک نسخه از مقدار بسازید. ویژگی `Clone` این کار را انجام می‌دهد.

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

```
{

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

.This slide should take about 2 minutes

- ای‌ده‌ی Clone این است که شناسایی مکانه‌ای تخصیص حافظه heap آسان‌تر شود. به دنبال ()clone. و چند مورد دیگر مانند vec! یا Box::new بگردید.
- معمولاً برای حل مشکلات مربوط به بررسی‌کننده قرض‌گیری (borrow checker) از کپی کردن اجتناب داده می‌شود و سپس در آینده تلاش می‌شود تا آن کپی‌ها به‌ینه‌سازی شوند.
- clone معمولاً یک کپی عمیق از مقدار را انجام می‌دهد، به این معنی که اگر به عنوان مثال یک آرایه را کپی کنید، تمام عناصر آن آرایه نیز کپی خواهند شد.
- رفتار clone توسط کاربر تعریف می‌شود، بنابراین می‌توانند در صورت نیاز، منطق کپی‌برداری سفارشی را اجرا کنند.

19.6 کپی کردن انتزاعات

در حالی که مفهوم انتقال به صورت پیش‌فرض است، در زبان راست چند نوع خاص به صورت پیش‌فرض کپی می‌شوند:

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

این انواع داده ویژگی Copy را پیاده‌سازی کرده‌اند.

البته که می‌توان برای نوع‌های که می‌سازید هم مفهوم کپی را داشته باشید:

```
[(derive(Copy, Clone, Debug]#
;(struct Point(i32, i32

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

- پس از انتساب، هر دو p1 و p2 داده‌های خود مستقل خود را دارند.
- همچنین می‌توانیم از p1.clone() برای کپی صریح داده‌ها استفاده کنیم.

.This slide should take about 5 minutes

کپی‌برداری و کلون‌سازی یکسان نیستند:

- کپی‌برداری به کپی‌های بیت به بیت از مناطق حافظه اشاره دارد و روی همه انواع تعریف شده توسط شما کار نمی‌کند.
- کپی‌برداری اجازه منطق سفارشی را نمی‌دهد (بر خلاف کپی constructors در C++).
- کلونسازی یک عملیات عمومی‌تر است و همچنین با پیاده‌سازی ویژگی Clone امکان رفتار سفارشی را فراهم می‌کند.
- کپی‌برداری روی انواع داده‌ای که ویژگی Drop را پیاده سازی کرده اند کار نمی‌کند.

در مثال بالا، موارد زیر را امتحان کنید:

- یک فیلد String به struct Point اضافه کنید. کامپایل نمی‌شود زیرا String یک نوع Copy نیست.
- ویژگی Copy را از صفت derive حذف کنید. اکنون خطای کامپایلر در println! برای p1 قرار دارد.
- نشان دهید که اگر p1 را به جای کپی آن کلون کنید، کار می‌کند.

برای کاوش بیشتر

- ارجاعات مشترک (shared references) دارای ویژگی Copy/Clone هستند، اما ارجاعات قابل تغییر (mutable references) اینطور نیستند. این به این دلیل است که Rust نیاز دارد که ارجاعات قابل تغییر منحصر به فرد باشند، بنابراین در حالی که کپی کردن یک ارجاع مشترک معتبر است، ایجاد یک کپی از یک ارجاع قابل تغییر قوانین قرض‌گیری Rust را نقض می‌کند.

19.7 ویژگی Drop

مقادیری که ویژگی Drop را پیاده‌سازی می‌کنند می‌توانند کدی را مشخص کنند که هنگام خروج از دامنه اجرا شود:

```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");
}
```

.This slide should take about 8 minutes

- توجه داشته باشید که `std::mem::drop` با `std::ops::Drop::drop` یکسان نیست.
- مقادیر به طور خودکار زمانی که از دامنه خارج می‌شوند، حذف می‌شوند.
- زمانی که یک مقدار حذف می‌شود، اگر آن مقدار ویژگی `std::ops::Drop` را پیاده‌سازی کرده باشد، پیاده‌سازی `Drop::drop` آن فراخوانی خواهد شد.
- تمام فیلدهای آن نیز سپس حذف خواهند شد، چه آن مقدار ویژگی `Drop` را پیاده‌سازی کرده باشد یا نه.
- `std::mem::drop` یک تابع خالی است که هر مقداری را می‌پذیرد. اهمیت آن در این است که مالکیت مقدار را به عهده می‌گیرد، بنابراین در پایان دامنه‌اش حذف می‌شود. این ویژگی آن را به روشی مناسب برای حذف صریح مقادیر پیش از آنچه که معمولاً از دامنه خارج می‌شوند، تبدیل می‌کند.
  - این می‌تواند برای اشیائی که در هنگام `drop` کاری انجام می‌دهند مفید باشد: آزاد کردن قفل‌ها، بستن فایل‌ها و غیره.

نکات بحث:

- چرا `Drop::drop self` را نمی‌گیرد؟
  - پاسخ کوتاه: اگر این‌طور بود، `std::mem::drop` در پایان بلوک فراخوانی می‌شد که منجر به فراخوانی مجدد `Drop::drop` و ایجاد خطای سرریز (stack overflow) می‌شد!
- سعی کنید `drop(a)` را با `a.drop()` جایگزین کنید.

19.8 تمرین: تابع‌های سازنده

در این مثال، ما یک نوع داده پیچیده را پیاده‌سازی خواهیم کرد که مالک تمام داده‌های خود است. ما از "الگوی سازنده" برای پشتیبانی از ساخت یک مقدار جدید به صورت قطعه‌قطعه، با استفاده از توابع کمکی، استفاده خواهیم کرد.

جاهای خالی را پر کنید.

```
#[derive(Debug)]
enum Language {
    Rust,
    Java,
    Perl,
}

#[derive(Clone, Debug)]
struct Dependency {
    name: String,
    version_expression: String,
}

/// A representation of a software package.
#[derive(Debug)]
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 راحل

```
                                                      [(derive(Debug]#
                                                       } enum Language
                                                             ,Rust
                                                             ,Java
                                                             ,Perl
                                                                    {

                                                [(derive(Clone, Debug]#
                                                    } struct Dependency
                                                      ,name: String
                                                ,version_expression: String
                                                                    {

                                           .A representation of a software package ///
                                                      [(derive(Debug]#
                                                      } 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
                                                                   {
```

فصل 20

اشاره‌گره‌ای هوشمند

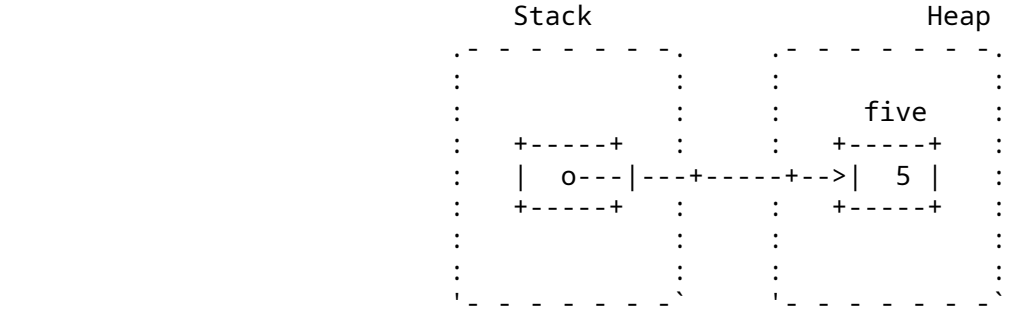
این بخش باید حدود ۵۵ دقیقه طول بکشد. آن شامل:

اسلاید	مدت زمان
Box<T>	۱۰ دقیقه
Rc	۵ دقیقه
Owned Trait Objects	۱۰ دقیقه
تمرین: درخت بایناری	۳۰ دقیقه

20.1 Box<T>

Box یک اشاره‌گر مالک به داده‌ای روی heap است:

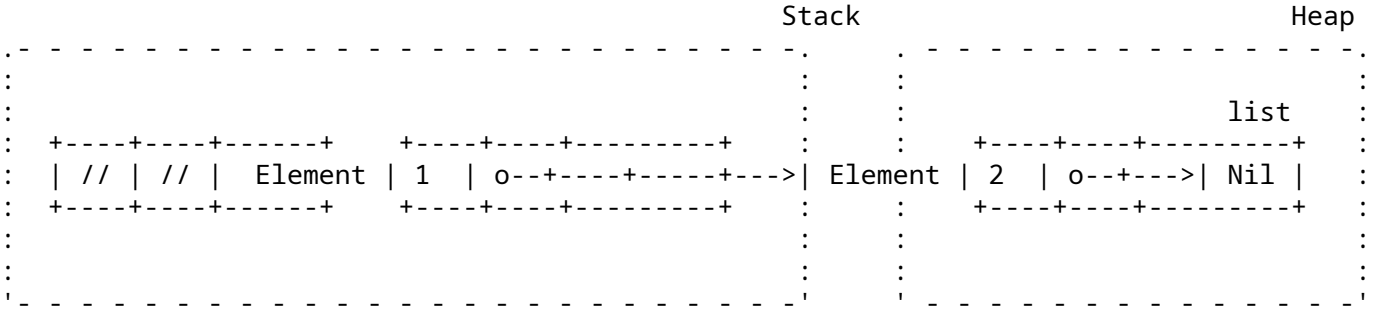
```
fn main() {
    let five = Box::new(5);
    println!("five: {}", *five);
}
```



Box<T> ویژگی Deref<Target = T> را پیاده‌سازی می‌کند، که به این معناست که می‌توانید متدهای روش‌های T را روی Box<T> فراخوانی کنید.

تایپ‌های داده‌ای بازگشتی یا انواع داده با اندازه‌های دینامیک را نمی‌توان به صورت inline بدون pointer indirection ذخیره کرد، که می‌توان با استفاده از Box آن کار را کرد:

```
[(derive(Debug]#
  } <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
{
```



.This slide should take about 8 minutes

- Box مانند `std::unique_ptr` در C++ است، با این تفاوت که تضمین شده است که همیشه تهی (null) نخواهد بود.
- Box می‌تواند زمانی مفید باشد که شما:
  - یک تایپ دارید که اندازه آن در زمان کامپایل مشخص نیست، اما کامپایلر Rust نیاز به دانستن اندازه دقیق آن دارد.
  - می‌خواهید که مقدار زیادی داده را انتقال دهید. برای جلوگیری از کپی کردن حجم زیادی از داده‌ها در پشته، به جای آن داده‌ها را در heap در یک Box ذخیره کنید تا فقط اشاره‌گر منتقل شود.
- اگر از Box استفاده نمی‌کردیم و سعی می‌کردیم یک `List` را مستقیماً در داخل `List` قرار دهیم، کامپایلر نمی‌توانست اندازه ثابتی برای ساختار در حافظه محاسبه کند (زیرا `List` اندازه‌ای بی‌نهایت پیدا می‌کرد).
- Box این مشکل را حل می‌کند زیرا اندازه‌ای برابر با یک اشاره‌گر عادی دارد و فقط به صرف به `List` در heap اشاره می‌کند.
- Box را از تعریف `List` حذف کنید و خطای کامپایلر را نمایش دهید. پیام خطا "recursive without indirection" را دریافت خواهیم کرد، زیرا برای رکورسیون داده‌ها باید از یک روش غیرمستقیم، مانند Box یا ارجاعی از نوعی، به جای ذخیره مستقیم مقدار استفاده کنیم.

برای کاوش بیشتر

بهینه سازی Niche

اگرچه Box مشابه std::unique_ptr در C++ به نظر می‌رسد، اما نمی‌توانند خالی/null باشند. این ویژگی باعث می‌شود که Box یکی از تایپ‌های باشد که به کامپایلر اجازه می‌دهد ذخیره‌سازی برخی از enumها را بهینه‌سازی کند.

برای مثال، Option<Box<T>> همان اندازه را دارد که Box<T>، زیرا کامپایلر از مقدار NULL برای تمایز بین variant ها به جای استفاده از تگ صریح استفاده می‌کند ("بهینه‌سازی اشاره‌گر خالی"):

```
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));
}
```

Rc 20.2

Rc یک اشاره‌گر مشترک با شمارش ارجاع است. از این هنگام استفاده کنید که نیاز دارید به داده‌ای یکسان از مکان‌های متعدد اشاره کنید:

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

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

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

- اگر در یک محیط چند-رشته‌ای (multi-threaded) هستید، به Arc و Mutex نگاه کنید.
- شما می‌توانید یک اشاره‌گر مشترک را به یک اشاره‌گر Weak تغییر دهید تا دوره‌ای ایجاد کنید که در نهایت حذف خواهند شد.

This slide should take about 5 minutes.

- شمارش Rc تضمین می‌کند که مقدار درون آن به مدت زمانی که ارجاع‌هایی وجود دارد، معتبر خواهد بود.
- Rc در Rust مشابه std::shared_ptr در C++ است.

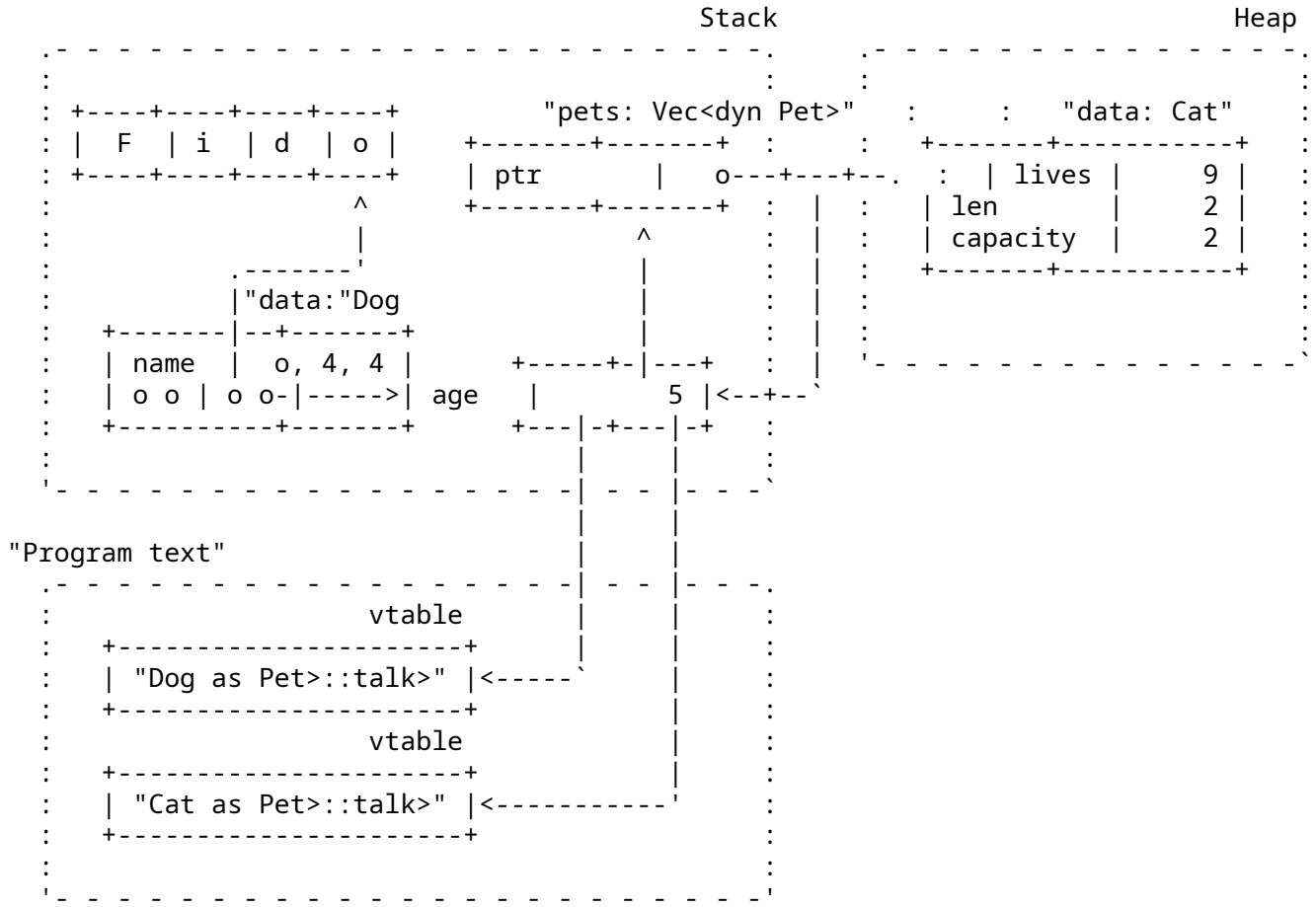
- `Rc::clone` ارزان است: این تابع یک اشاره‌گر به همان تخصیص (allocation) ایجاد می‌کند و شمارش ارجاع را افزایش می‌دهد. این عمل کپی عمیق (deep clone) انجام نمی‌دهد و به طور کلی هنگام جست‌وجو برای مسائل عملکردی در کد می‌توان آن را نادیده گرفت.
- `make_mut` در واقع در صورت نیاز مقدار درونی را کپی می‌کند (”clone-on-write“) و یک ارجاع قابل تغییر (mutable reference) برمی‌گرداند.
- از `Rc::strong_count` برای بررسی شمارش ارجاع‌ها استفاده کنید.
- `Rc::downgrade` یک شیء با شمارش ارجاع ضعیف به شما می‌دهد تا دوره‌ای ایجاد کنید که به درستی حذف خواهند شد (احتمالاً به همراه `RefCell`).

Owned Trait Objects 20.3

پیش‌تر دیدیم که چگونه می‌توان ازویژگی اشیاء (trait objects) با ارجاعات استفاده کرد، مثلاً dyn Pet&. با این حال، می‌توانیم از اشیاء ویژگی با اشاره‌گرهای هوشمند مانند Box نیز استفاده کنیم تا یک شیء ویژگی مالک (owned trait object) ایجاد کنیم: Box<dyn Pet>.

```
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! نام من {} است", self.name)
    }
}

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

fn main() {
    let pets: Vec<Box<dyn Pet>> = vec![
        Box::new(Dog { name: String::from("Fido"), age: 5 }),
        Box::new(Cat { lives: 9 }),
    ];
    for pet in pets {
        println!("{} ؟دیتسه یک امش ،امل‌س", pet.talk());
    }
}
```

چیدمان حافظه پس از تخصیص pets:



.This slide should take about 10 minutes

- تای‌پ‌های‌ی که ویژگی مع‌ین را پی‌اده‌سازی می‌کنند ممکن است اندازه‌های مختلف‌ی داشته باشند. این موضوع باعث می‌شود که داشتن متغیرهای از نوع `Vec<dyn Pet>` در مثال بالا غیرممکن باشد.
- `dyn Pet` راه‌ی است برای اطلاع دادن به کامپایلر درباره یک تای‌پ با اندازه‌ای که ویژگی `Pet` را پی‌اده‌سازی می‌کند.
- در این مثال، `pets` در stack تخصیص داده می‌شود و داده‌های vector در heap هستند. دو عنصر اشاره‌گرهای چاق (fat pointers) vector هستند:
  - اشاره‌گر چاق (fat pointer) یک اشاره‌گر با عرض دو برابر است. این اشاره‌گر دو مؤلفه دارد: یک اشاره‌گر به شیء واقعی و یک اشاره‌گر به روش‌های جدول مجازی (vtable) برای پی‌اده‌سازی `Pet` آن شیء خاص.
  - داده‌های مربوط به `Dog` به نام Fido شامل فیلدهای `name` و `age` است. `Cat` دارای فیلد `lives` است.
- خروجی‌های زیر را در مثال بالا مقایسه کنید:

```
    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 تمرین: درخت باینری

یک درخت باینری (binary tree) یک ساختار داده درختی است که در آن هر گره دو فرزند (چپ و راست) دارد. ما درختی خواهیم ساخت که در آن هر گره یک مقدار را ذخیره می‌کند. برای یک گره معین N، تمام گره‌های زیر درخت چپ N دارای مقادیر کوچکتر خواهند بود و تمام گره‌های زیر درخت راست N دارای مقادیر بزرگتر خواهند بود.

تابع‌های زیر را پیاده‌سازی کنید تا آزمایش‌های داده شده موفقیت‌آمیز باشند.

اختیاری: یک تکرارگر (iterator) بر روی درخت باینری پیاده‌سازی کنید که مقادیر را به ترتیب (in-order) برگرداند.

```
/// A node in the binary tree.
#[derive(Debug)]
struct Node<T: Ord> {
    value: T,
    left: Subtree<T>,
    right: Subtree<T>,
}

/// A possibly-empty subtree.
#[derive(Debug)]
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.
#[derive(Debug)]
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`.
```

```
                                                    [(cfg(test]#
                                                     } mod tests
                                                ;*::use super

                                                       [test]#
                                                   } ()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
                                                                    {

                                                       [test]#
                                                   } ()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
                                                                    {

                                                       [test]#
                                                     } ()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 راه‌حل

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

```
                                        .A node in the binary tree ///
                                                    [(derive(Debug]#
                                                  } <struct Node<T: Ord
                                                          ,value: T
                                                        ,<left: Subtree<T
                                                       ,<right: Subtree<T
                                                                        {

                                        .A possibly-empty subtree ///
                                                    [(derive(Debug]#
                                  ;(<<<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 ///
                                                    [(derive(Debug]#
                                                } <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
                                                                        {

                                                                    [(cfg(test]#
                                                                     } mod tests
                                                            ;*::use super

                                                                     [test]#
                                                                   } ()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
                                                                        {
```

```
                                                        [test]#
                                                    } ()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
                                                               {

                                                        [test]#
                                                  } ()fn unbalanced
                                ;()let mut tree = BinaryTree::new
                                              } for i in 0..100
                                           ;(tree.insert(i
                                                           {
                                    ;(assert_eq!(tree.len(), 100
                                         ;((assert!(tree.has(&50
                                                           {
                                                               {
```

بخش VI

روز ۳: بعد از ظهر

فصل 21

خوش آمد

با احتساب استراحت‌های ۱۰ دقیقه‌ای، این جلسه باید حدود ۱ ساعت و ۵۵ دقیقه طول بکشد. این شامل:

بخش	مدت زمان
قرض‌گیری (Borrowing) طول‌عمر	۵۵ دقیقه ۵۰ دقیقه

فصل 22

قرض‌گیری (Borrowing)

این بخش بای حدود ۵۵ دقیقه طول بکشد. آن شامل:

اسلاید	مدت زمان
قرض‌گیری یک مقدار	۱۰ دقیقه
چک کردن قرض	۱۰ دقیقه
خطاهای قرض‌گیری	۳ دقیقه
تغییرپذیری داخلی	۱۰ دقیقه
تمرین: آمار سلامتی	۲۰ دقیقه

22.1 قرض‌گیری یک مقدار

همان‌طور که پیش‌تر دیدیم، به جای انتقال مالکیت هنگام فراخوانی یک تابع، می‌توانید به تابع اجازه دهید که ارجاعی به مقدار داشته باشد:

```
#[derive(Debug]
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:?}");
}
```

- تابع add ارجاع می‌گیرد به دو نقطه و یک نقطه جدید برمی‌گرداند.
- فراخوانی‌کننده مالکیت ورودی‌ها را حفظ می‌کند.

This slide should take about 10 minutes.

این اسل‌ای‌د مرور مطال‌ب مربوط به ارجاعات از روز اول است که به طور جزئی به شامل آرگوم‌ان‌های تابع و مقادیر بازگشتی گسترش یافته است.

برای کاوش بیشتر

یادداشت‌های در مورد بازگشته‌ای stack و درون‌ریزی (inlining):

- برای نشان دادن این که بازگشت از add ارزان است، زیرا کامپایلرها می‌توانند عملیات کپی را حذف کنند، با درون‌ریزی (inlining) فراخوانی add به تابع main، کد فوق را تغییر دهید تا آدرس‌های پشته چاپ شوند و آن را در Playground اجرا کنید یا به اسمبلی در Godbolt نگاه کنید. در سطح بهینه‌سازی "DEBUG"، آدرس‌ها باید تغییر کنند، در حالی که با تغییر به تنظیم "RELEASE" ثابت می‌مانند:

```
#[derive(Debug)]
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:?}");
}
```

- کامپایلر Rust می‌تواند به‌طور خودکار درون‌ریزی (inlining) انجام دهد، که می‌تواند در سطح تابع با #[inline(never)] غیرفعال شود.
- پس از غیرفعال کردن درون‌ریزی، آدرس‌های چاپ شده در تمامی سطوح بهینه‌سازی تغییر خواهند کرد. با نگاه کردن به Godbolt یا Playground، می‌توان دید که در این حالت، بازگشت مقدار به ABI بستگی دارد، به عنوان مثال در amd64، دو i32 که نقطه را تشکیل می‌دهند، در دو رجیستر (مانند eax و edx) بازگردانده خواهند شد.

22.2 چک کردن قرض

بررسی‌کننده ارجاع (borrow checker) در Rust محدودیت‌های بر روی روش‌های ارجاع به مقادیر اعمال می‌کند. برای یک مقدار خاص، در هر زمان:

- شما می‌توانید یک یا چند ارجاع اشتراکی به مقدار داشته باشید، یا
- می‌توانید دقیقاً یک ارجاع انحصاری به مقدار داشته باشید.

```
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
            {
```

.This slide should take about 10 minutes

- توجه داشته باشید که نیاز این است که ارجاعات متضاد در همان نقطه وجود نداشته باشند. هم نیست که ارجاع در کجا dereferenced شود.
- کد بالا کامپایل نمی‌شود زیرا a به‌طور همزمان به‌صورت قابل تغییر (از طریق c) و غیرقابل تغییر (از طریق b) ارجاع داده شده است.
- برای اینکه کد کامپایل شود، دستور println! مربوط به b را قبل از محدوده‌ای که c را عرفی می‌کند، منتقل کنید.
- پس از آن تغییر، کامپایلر متوجه می‌شود که b تنها پیش از ارجاع جدید قابل تغییر به a از طریق c استفاده می‌شود. این ویژگی بررسی‌کننده ارجاع به نام non-lexical lifetimes است.
- محدودیت ارجاع انحصاری بسیار قوی است. Rust از آن برای اطمینان از عدم وقوع داده‌های رقابتی (data races) استفاده می‌کند. Rust همچنین متکی به این محدودیت برای به‌ینه‌سازی کد است. به عنوان مثال، مقدار پشت یک ارجاع اشتراکی می‌تواند به‌طور ایمن در یک رجیستر برای طول عمر آن ارجاع کش شود.
- بررسی‌کننده ارجاع به‌گونه‌ای طراحی شده است که بسیاری از الگوهای رایج را پشتیبانی کند، مانند گرفتن ارجاعات انحصاری به فیلدهای مختلف در یک ساختار به‌طور همزمان. اما، در برخی موقعیت‌ها، ممکن است که به‌طور کامل متوجه وضعیت نشود و این اغلب منجر به "درگیری با borrow checker" می‌شود.

22.3 خطاهای قرضگیری

به عنوان یک مثال ملموس از چگونگی جلوگیری از خطاهای حافظه توسط این قوانین ارجاع، به حالت تغییر یک مجموعه در حالی که ارجاعاتی به عناصر آن وجود دارد، توجه کنید:

```
} ()fn main
;[let mut vec = vec![1, 2, 3, 4, 5
;[let elem = &vec[2
;(vec.push(6
;("{println!("{elem
                {
```

به طور مشابه، به وضعیت نامعتبر شدن تکرارگر (iterator) توجه کنید:

```
} ()fn main
;[let mut vec = vec![1, 2, 3, 4, 5
    } for elem in &vec
;(vec.push(elem * 2
                {
                {
```

.This slide should take about 3 minutes

- در هر دو مورد، تغییر مجموعه با اضافه کردن عناصر جدید به آن می‌تواند به‌طور بالقوه ارجاعات موجود به عناصر مجموعه را نامعتبر کند، اگر مجموعه نیاز به تخصیص مجدد حافظه داشته باشد.

22.4 تغییرپذیری داخلی

در برخی موقعیت‌ها، لازم است که داده‌ای پشت یک ارجاع اشتراکی (فقط خواندنی) را تغییر دهید. به عنوان مثال، یک ساختار داده اشتراکی ممکن است دارای یک کش داخلی باشد و بخواهد این کش را از روش‌های خواندنی به‌روزرسانی کند.

الگوی "تغییرپذیری داخلی" (interior mutability) اجازه می‌دهد که دسترسی انحصاری (قابل تغییر) پشت یک ارجاع اشتراکی وجود داشته باشد. کتابخانه استاندارد روش‌های متعددی برای انجام این کار فراهم می‌کند، در حالی که همچنان می‌تضمین را نمی‌کنند، معمولاً با انجام یک بررسی در زمان اجرا.

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.

این type با استفاده از یک شمارنده در RefCell بررسی‌های dynamic را انجام می‌دهند تا از وجود RefMut در کنار Ref/RefMut دیگر جلوگیری کنند.

با پیاده‌سازی Deref (و DerefMut برای RefMut)، این تایپ‌ها امکان فراخوانی متدها روی مقدار داخلی را بدون اجازه خروج ارجاع‌ها فراهم می‌کنند.

```
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
{
```

.This slide should take about 10 minutes

مه‌م‌تری‌ن ن‌کته‌ای که با‌ید از ای‌ن اس‌لای‌د ب‌رداشت ک‌رد ای‌ن اس‌ت که Rust روش‌ه‌ای ای‌من ب‌رای ت‌غ‌ی‌ر داده-های پ‌شت ی‌ک ارجاع اش‌ت‌راک‌ی ارائه می‌دهد. راه‌ه‌ای مخ‌ت‌ل‌ف‌ی ب‌رای ت‌ض‌م‌ی‌ن ای‌ن ای‌من‌ی وجود دارد، و `RefCell` و `Cell` دو مورد از آن‌ه‌ا ه‌س‌ت‌ند.

- `RefCell` ق‌وان‌ی‌ن مع‌مول ارجاع Rust (ی‌ا چ‌ن‌دی‌ن ارجاع اش‌ت‌راک‌ی ی‌ا ی‌ک ارجاع ان‌ح‌ص‌اری) را ب‌ا ی‌ک ب‌ررس‌ی در زمان اج‌را اع‌مال می‌ک‌ن‌د. در ای‌ن ح‌ال‌ت، ه‌مه ارجاع‌ات ب‌س‌ی‌ار ک‌وتاه ه‌س‌ت‌ند و ه‌رگ‌ز ه‌م‌پ‌وش‌ان‌ی ن‌دارند، ب‌ن‌اب‌ران‌ی‌ن ب‌ررس‌ی‌ه‌ا ه‌م‌ی‌ش‌ه مو‌ف‌ق‌ی‌ت‌آم‌ی‌ز ه‌س‌ت‌ند.

  – ب‌ل‌وک اض‌اف‌ی در م‌ث‌ال `RefCell` ب‌رای پ‌ای‌ان دادن ب‌ه ارجاع‌ی ک‌ه ت‌وس‌ط ف‌راخ‌وان‌ی `borrow_mut` ای‌جاد ش‌ده اس‌ت، ق‌ب‌ل از چ‌اپ `cell` اس‌ت. ت‌ل‌اش ب‌رای چ‌اپ ی‌ک `RefCell` ک‌ه در ح‌ال ح‌اض‌ر ارجاع داده ش‌ده اس‌ت، ف‌ق‌ط پ‌ی‌غ‌ام `"{borrowed}"` را ن‌ش‌ان می‌ده‌د.

- `Cell` ی‌ک روش س‌اده‌ت‌ر ب‌رای ت‌ض‌م‌ی‌ن ای‌من‌ی اس‌ت: ای‌ن ن‌وع دارای م‌ت‌دی ب‌ه ن‌ام `set` اس‌ت ک‌ه `&self` را می‌پ‌ذی‌رد. ای‌ن روش ن‌ی‌از ب‌ه ب‌ررس‌ی در زمان اج‌را ن‌دارد، ام‌ا ن‌ی‌از ب‌ه ان‌ت‌ق‌ال م‌ق‌ادی‌ر دارد ک‌ه می‌توان‌د ه‌زی‌ن‌ه‌ای خ‌ود را داش‌ته ب‌اش‌د.

- ه‌ر دو `RefCell` و `Cell` دارای `Sync!` ه‌س‌ت‌ند، ب‌ه ای‌ن مع‌ن‌ی ک‌ه `&RefCell` و `Cell&` ن‌می‌توان‌ند ب‌ی‌ن ن‌خ‌ه‌ا م‌ن‌ت‌ق‌ل ش‌ون‌د. ای‌ن ام‌ر ام‌ن‌ی‌ع از دس‌ت‌رس‌ی م‌ه‌زم‌ان دو ن‌خ ب‌ه س‌ل‌ول می‌ش‌ود.

22.5 ت‌م‌ری‌ن: آم‌ار س‌ل‌ام‌ت‌ی

ش‌ما در ح‌ال پ‌ی‌اده‌س‌ازی ی‌ک س‌ی‌س‌ت‌م پ‌ای‌ش س‌ل‌ام‌ت ه‌س‌ت‌ی‌د. ب‌ه ع‌ن‌وان ب‌خ‌ش‌ی از ای‌ن ک‌ار، ن‌ی‌از دارید ت‌ا آم‌ار س‌ل‌ام‌ت ک‌ارب‌ران را دن‌ب‌ال ک‌ن‌ی‌د.

ش‌ما ب‌ا ی‌ک ت‌ابع اب‌ت‌دای‌ی در ب‌ل‌وک `impl` و ه‌م‌چ‌ن‌ی‌ن ی‌ک ت‌ع‌ری‌ف `struct` ب‌ه ن‌ام `User` ش‌روع خ‌واه‌ی‌د ک‌رد. ه‌دف ش‌ما پ‌ی‌اده‌س‌ازی م‌ت‌د اب‌ت‌دای‌ی در `User` اس‌ت ک‌ه در ب‌ل‌وک `impl` ت‌ع‌ری‌ف ش‌ده اس‌ت.

ک‌د زی‌ر را ب‌ه https://play.rust-lang.org/ ک‌پ‌ی ک‌ن‌ی‌د و م‌ت‌ده‌ای ن‌اق‌ص را ت‌ک‌م‌ی‌ل ک‌ن‌ی‌د:

```
// TODO: ای‌ن را زم‌ان‌ی ک‌ه پ‌ی‌اده‌س‌ازی ت‌ام‌ت ش‌د ح‌ذف ک‌ن‌ی‌د.
#![allow(unused_variables, dead_code)]


#[allow(dead_code)]
struct User {
    name: String,
    age: u32,
    height: f32,
    visit_count: usize,
    last_blood_pressure: Option<(u32, u32)>,
}

struct Measurements {
    height: f32,
    blood_pressure: (u32, u32),
}

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!("به‌روزرسانی آمار یک کاربر بر اساس اندازه‌گیری‌های معیار بازدید از پزشک")
    }
}

fn main() {
    let bob = User::new(String::from("Bob"), 32, 155.2);
    println!("من {} هستم و سن من {} است", bob.name, bob.age);
}

#[test]
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 راه‌حل

```
#![allow(dead_code)]
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 ,"است {} من سن و مست‌ه {} من")!println
                                                                          {

                                                                    [test]#
                                                                } ()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
                                                                                {
```

فصل 23

طول‌عمر

این بخش حدود ۵۰ دقیقه طول خواهد کشید. شامل موارد زیر است:

اسلاید	مدت زمان
تفسیری‌ره‌ای طول عمر	۱۰ دقیقه
حذف طول عمر	۵ دقیقه
طول عمر ساختارها	۵ دقیقه
تمرین: تجزیه Protobuf	۳۰ دقیقه

23.1 تفسیری‌ره‌ای طول عمر

یک مرجع دارای طول عمر است که نباید از ارزش مورد اشاره بیشتر باشد. این موضوع توسط بررسی-کننده قرض‌ها تایید می‌شود.

طول عمر می‌تواند ضمنی باشد - این همان چیزی است که تا کنون مشاهده کرده‌ایم. طول عمرها می‌توانند صریح نیز باشند: `&'a Point`، `&'document str`. طول عمرها با `'` شروع می‌شوند و `'a` نام پیش-فرض معمولی است. `&'a Point` را به عنوان "یک `Point` قرضی که برای حداقل طول عمر `a` معتبر است" بخوانید.

طول عمرها همیشه توسط کامپایلر استنتاج می‌شوند: شما نمی‌توانید به طور دستی طول عمر را اختصاص دهید. انتساب‌های صریح طول عمر محدودیت‌هایی ایجاد می‌کنند که در صورت وجود ابهام است؛ کامپایلر تأیید می‌کند که یک راه‌حل معتبر وجود دارد.

طول عمرها وقتی که به عبور مقادیر به توابع و بازگشت مقادیر از توابع می‌پردازیم پیچیده‌تر می-شوند.

```rust
#[derive(Debug)]
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
{
```

.This slide should take about 10 minutes

در این مثال، کامپایلر نمی‌داند که طول عمر p3 را چگونه استنباط کند. نگاه کردن به بدنه تابع نشان می‌دهد که تنها به‌طور ایمن می‌توان فرض کند که طول عمر p3 کوتاه‌تر از p1 و p2 است. اما مانند تایپ‌ها، راست نیاز به توضیحات صریح طول عمرها در آرگومان‌های تابع و مقادیر بازگشتی دارد.

به تابع left_most به صورت من‌اسب 'a را اضافه کنید:

```
} fn left_most<'a>(p1: &'a Point, p2: &'a Point) -> &'a Point
```

این به این معناست که "با توجه به این‌که p1 و p2 هر دو از 'a بیشتر عمر می‌کنند، مقدار بازگشتی برای مدت عمر 'a خواهد بود".

در موارد معمول، عمر متغیرها می‌تواند نادیده گرفته شود، همان‌طور که در اسلایدهای قبلی توضیح داده شده است.

23.2 طول عمر در فراخوانی توابع

عمرهای مربوط به آرگومان‌های تابع و مقادیر بازگشتی باید به طور کامل مشخص شوند، اما Rust اجازه می‌دهد عمرها در بیشتر موارد با چند قانون ساده نادیده گرفته شوند. این مسئله استنتاج نیست -- بلکه تنها یک اصطلاح نوشتاری کوتاه است.

- هر آرگومان که فاقد یک lifetime annotation است، یک عمر به آن اختصاص داده می‌شود.
- اگر تنها یک عمر برای آرگومان وجود داشته باشد، به تمام مقادیر بازگشتی که حاشیه‌نویسی نشده‌اند، اختصاص داده می‌شود.
- اگر چندین عمر آرگومان وجود داشته باشد و اولین آن برای self باشد، آن عمر به تمام مقادیر بازگشتی که حاشیه‌نویسی نشده‌اند، اختصاص داده می‌شود.

```
[(derive(Debug]#
;(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&
                                                          (
                                                             ;(
                                                                {
```

.This slide should take about 5 minutes

در این مثال، `cab_distance` به طور خودکار حذف می‌شود.

تابع `nearest` مثال دیگری از تابعی است که با ارجاعات متعدد در آرگومان‌هایش نیاز به حاشیه‌نویسی صریح دارد.

امضا را طوری تنظیم کنید که "دروغ" بگوید درباره طول عمر مقادیری که برگشت داده می‌شوند:

```
} <fn nearest<'a, 'q>(points: &'a [Point], query: &'q Point) -> Option<&'q Point
```

این کد کامپایل نخواهد شد، که نشان‌دهنده این است که برچسب‌های طول عمر توسط کامپایلر برای اعتبارسنجی بررسی می‌شوند. توجه داشته باشید که این وضعیت برای اشاره‌گرهای خام (ناامن) صدق نمی‌کند و این یکی از معایب رایج خطاها در Rust ناامن است.

دانش‌آموزان ممکن است بپرسند که چه زمانی باید از طول عمرها استفاده کرد. در Rust، همیشه برای قرض‌ها طول عمر وجود دارد. بیشتر مواقع، حذف و استنباط تایپ به این معنی است که نیازی به نوشتن این طول عمرها نیست. در موارد پیچیده‌تر، برچسب‌های طول عمر می‌توانند به حل ابهام کمک کنند. اغلب، به‌ویژه در هنگام پروتوتایپ‌سازی، راحت‌تر است که با داده‌های مالکیت‌شده کار کنید و مقادیر را در صورت لزوم کلون کنید.

Lifetimes in Data Structures 23.3

اگر یک تایپ داده داده‌ای قرضی را ذخیره کند، باید با یک طول عمر مشخص شود:

```
                                                    [(derive(Debug]#
                                                ;(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
{
```

.This slide should take about 5 minutes

- در مثال بالا، حاشیه‌نویسی بر روی `Highlight` تضمین می‌کند که داده‌ای زیرین `&str` به مدت حداقل برابر با هر نمونه از `Highlight` که از آن داده استفاده می‌کند، زنده بماند.
- اگر `text` قبل از پایان عمر `fox` (یا dog) مصرف شود، بررسی‌کننده‌ی قرض (borrow checker) خطا می‌دهد.
- تایپ‌های دارای داده‌ای قرضی (borrowed data) کاربران را مجبور می‌کنند تا داده‌های اصلی را نگه دارند. این می‌تواند برای ایجاد نمایه‌های سبک مفید باشد، اما معمولاً استفاده از آن‌ها را تا حدی دشوارتر می‌کند.
- در صورت امکان، داده‌های ساختارها را به طور مستقیم مالکیت کنید.
- برخی از ساختارهای داده که شامل چندین ارجاع هستند، ممکن است نیاز به چندین نشانه‌گذاری عمر داشته باشند. این امر می‌تواند ضروری باشد اگر بخواهید روابط عمری بین ارجاعات مختلف را به علاوه عمر ساختار خود توصیف کنید. این موارد بسیار پیشرفته هستند.

23.4 تمرین: تجزیه Protobuf

در این تمرین، شما یک تجزیه‌کننده برای رمزگذاری بایناری پروتوباف خواهید ساخت. نگران نباشید، این کار ساده‌تر از آن است که به نظر می‌رسد! این الگو نشان‌دهنده یک الگوی رایج در تجزیه داده‌ها است که شامل عبور برش‌های داده است. داده‌ای اصلی هرگز کپی نمی‌شوند.

تجزیه کامل یک پیام پروتوباف نیاز به دانستن تایپ‌های این فیلدها دارد که بر اساس شماره‌های فیلد ایندکس شده‌اند. این اطلاعات معمولاً در یک فایل `proto` ارائه می‌شود. در این تمرین، ما این اطلاعات را به صورت عبارات `match` در توابعی که برای هر فیلد فراخوانی می‌شوند، کدگذاری خواهیم کرد.

ما از پروتوباف زیر استفاده خواهیم کرد:

```
} message PhoneNumber
;optional string number = 1
;optional string type = 2
{

} message Person
;optional string name = 1
;optional int32 id = 2
;repeated PhoneNumber phones = 3
{
```

یک پیام پروتوباف به عنوان مجموعه‌ای از فیلدها، یکی پس از دیگری، کدگذاری می‌شود. هر فیلد به صورت یک "تگ" به همراه مقدار آن پیاده‌سازی شده است. تگ شامل شماره فیلد (مانند 2 برای فیلد `id` در پیام `Person`) و wire type است که نحوه تعیین بایت مشخص می‌کند.

اعداد، از جمله تگ، با استفاده از کدگذاری با طول متغیر به نام VARINT نمایندگی می‌شوند. خوشبختانه، توابع `parse_varint` برای شما تعریف شده است. کد داده شده همچنین بازخوانی‌های برای مدیریت فیلدهای `Person` و `PhoneNumber` و تجزیه یک پیام به مجموعه‌ای از فراخوانی‌ها به آن بازخوانی‌ها را تعریف می‌کند.

برای شما باقی‌مانده است که توابع `parse_field` و ویژگی `ProtoMessage` را برای `Person` و `PhoneNumber` پیاده‌سازی کنید.

```
.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//
                                                                      {

                                                      [(derive(Debug]#
                            .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//
                                                                      {

                                                      [(derive(Debug]#
                         .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} :ربتعمان میس عون")!panic <= _
                                                                  {
                                                                      {
                                                                      {

                                                  } <impl<'a> FieldValue<'a
                                           } fn as_str(&self) -> &'a str
                                } let FieldValue::Len(data) = self else
                    ;("دشاب Len دلیف کی هتشر هک دورمی رارقتان")!panic
                                                                 ;{
                        ("string ربتعمان")std::str::from_utf8(data).expect
```

```
                                                                {

                                                } [fn as_bytes(&self) -> &'a [u8
                                            } let FieldValue::Len(data) = self else
                                ;("دنشاب `Len` دلیف کی رارظتنا درمو ایەتایب")!panic
                                                                    ;{
                                                                  data
                                                                    {

                                                    } fn as_u64(&self) -> u64
                                        } let FieldValue::Varint(value) = self else
                                ;("دشاب `Varint` دلیف کی `u64` دورین رارظتنا")!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
                                        ;("تسین varint ارب یفاک تایب")!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 ارب یدایز ایەتایب دادعت")!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
```

```
ا مزال هک تیاب دادعت ره فرصم اب ،دیزاسب دلیف کی ،میس عون ساسا رب")!todo <= _
                                                                        ;{
                                                  ("دیناردگرب را هدش فرصم تیاب ره و دلیف")!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
                                                                          {



                                                        [(derive(Debug, Default]#
                                                              } <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
                                                                          {
```

.This slide and its sub-slides should take about 30 minutes

- در این تمرین موارد مختلفی وجود دارد که ممکن است تجزیه protobuf با شکست مواجه شود، مثلاً اگر بخواهید یک i32 را هنگامی که کمتر از ۴ بایت در بافر داده باقی‌مانده است، تجزیه کنید. در کد Rust معمولاً این را با استفاده از Result مدیریت می‌کنیم، اما برای سادگی در این تمرین، اگر با هرگونه خطا مواجه شویم، به جای آن که با Result برخورد کنیم، برنامه را متوقف خواهیم کرد. در روز چهارم، به بررسی دقیق‌تر مدیریت خطا در Rust خواهیم پرداخت.

23.4.1 راهاحل

```
                                        .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//
                                                                 {

                                                    [(derive(Debug]#
                                .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//
                                                                 {

                                                    [(derive(Debug]#
                                .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} :ربتعمان میس عون")!panic <= _
                                                                    {
                                                                        {
                                                                        {

                                                    } <impl<'a> FieldValue<'a
                                             } fn as_str(&self) -> &'a str
```

```
                    } let FieldValue::Len(data) = self else
        ;("دشاب Len دلیف کی هتشر هک دورمی راظتنا")!panic
                                                    ;{
            ("string ربتعمان")std::str::from_utf8(data).expect
                                                        {

                                    } [fn as_bytes(&self) -> &'a [u8
                    } let FieldValue::Len(data) = self else
        ;("دنشاب `Len` دلیف کی راظتنا دروم ایهتیاب")!panic
                                                    ;{
                                                      data
                                                        {

                                        } fn as_u64(&self) -> u64
                } let FieldValue::Varint(value) = self else
        ;("دشاب `Varint` دلیف کی `u64` دورمی راظتنا")!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
                    ;("تسین varint یارب یفاک تیاب")!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 یارب یدایز ایهتیاب دادعت")!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
س‌ین ربتعم `usize` در len از ده‌افتسا")let len: usize = len.try_into().expect
                                                    } if remainder.len() < len
                                                ;("هرظتنم رغی 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
                                                                                {

                                                                    [(derive(PartialEq]#
                                                                [(derive(Debug, Default]#
                                                                 } <struct PhoneNumber<'a
                                                                     ,number: &'a str
                                                                      ,type_: &'a str
                                                                                {

                                                                    [(derive(PartialEq]#
                                                                [(derive(Debug, Default]#
                                                                      } <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
                                                                {

                                                            [(cfg(test]#
                                                             } mod tests
                                                    ;*::use super

                                                            [test]#
                                                        } ()fn test_id
                        ;([let person_id: Person = parse_message(&[0x10, 0x2a
        ;({ []!assert_eq!(person_id, Person { name: ".", id: 42, phone: vec
                                                                {

                                                            [test]#
                                                    } ()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
                                                                ;(
```

```
                                                                                {

                                                                            [test]#
                                                                    } ()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
                                                                                {

                                                                            [test]#
                                                                  } ()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
,[,{ "هان" :_phone: vec![PhoneNumber { number: "+1234-777-9090", type
                                                                    {
                                                                       ;(
                                                                            {
                                                                                {
```

بخش VII

روز چهارم: صبح

فصل 24

Welcome to Day 4

امروز ما موضوعات مربوط به ساخت نرم افزار در مقیاس بزرگ در Rust را پوشش خواهیم داد:

- Iterators: شیرجه عمیق در ویژگی `Iterator`.
- ماژول ها و قابلیت مشاهده.
- تست‌کردن.
- رسیدگی به خطا: panics در `Result` و اپراتور تلاش مجدد `?`.
- با Unsafe Rust: پنجره فرار مبتنی بر زمان که نمی‌توانید خود را در safe Rust بیان کنید.

برنامه زمانی

با احتساب ۱۰ دقیقه استراحت، این جلسه باید حدود ۲ ساعت و ۴۰ دقیقه طول بکشد. شامل:

بخش	مدت زمان
خوش آمدید	۳ دقیقه
Iterators	۴۵ دقیقه
ماژول‌ها	۴۰ دقیقه
تست‌کردن	۴۵ دقیقه

فصل 25

Iterators

این بخش باید حدود ۴۵ دقیقه طول بکشد. آن شامل:

اسلاید	مدت زمان
Iterator	۵ دقیقه
IntoIterator	۵ دقیقه
FromIterator	۵ دقیقه
تمرین: روش Iterator Chaining	۳۰ دقیقه

25.1 `Iterator`

ویژگی 'Iterator' از تکرار بیش از مقادیر در یک مجموعه پشتیبانی می‌کند. این به یک متد next نیاز دارد و متدهای زیادی را ارائه می‌کند. بسیاری از انواع کتابخانه استاندارد Iterator را پیاده‌سازی می‌کنند و شما نیز می‌توانید آن را خودتان پیاده‌سازی کنید:

```
} 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
                                                                      {
                                                                          {
```

.This slide should take about 5 minutes

- ویژگی `Iterator` بسیاری از عملیات برنامه‌نویسی رایج را روی collectionها پیاده‌سازی می‌کند (مانند `map`, `filter`, `reduce`و غیره). این ویژگی است که در آن می‌توانید تمام این‌ها را پیدا کنید. در Rust، این توابع بایدک را به اندازه پیاده‌سازی‌های ضروری مربوط به آن‌ها بهینه تولید کنند.
- ویژگی `IntoIterator` است که باعث می‌شود حلقه‌ها کار کنند. این مجموعه (collection) توسط پیاده‌سازی می‌شود. `Vec<T>` و ارجاعاتی به آن‌ها مانند `Vec<T>&` و `[T]&` Ranges نیز آن را اجرا می‌کنند. به همین دلیل است که می‌توانید روی یک بردار با `for i in some_vec { .. }` تکرار کنید، اما درنهایت `some_vec.next()` وجود ندارد.

`IntoIterator` 25.2

ویژگی `Iterator` به شما می‌گوید که چگونه پس از ایجاد یک تکرار کننده، *iterate* کنید. ویژگی `IntoIterator` مرتبط نحوه ایجاد یک تکرار کننده برای یک نوع را مشخص می‌کند. به طور خودکار توسط حلقه `for` استفاده می‌شود.

```
                                                } 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
                                                              {
                                                                  {
```

This slide should take about 5 minutes.

روی مستندات `IntoIterator` کلیک کنید. هر پیاده‌سازی `IntoIterator` باید دو نوع را اعلام کند:

- `Item`: نوعی که باید تکرار شود، مانند `i8`،
- `IntoIter`: یک «Iterator» تایپ است که با متد `into_iter` برگردانده شده است.

توجه داشته باشید که `IntoIter` و `Item` به هم `link` شده‌اند: تکرارکننده (iterator) باید همان `Item` type را داشته باشد، به این معنی که `Option<Item>` را برمی‌گردانند.

مثال روی تمام ترکیبات مختصات `x` و `y` تکرار می‌شود.

سعی کنید دو بار روی شبکه در `main` تکرار کنید. چرا این گزینه شکست می‌خورد؟ توجه داشته باشید که `IntoIterator::into_iter` مالکیت `self` را می‌گیرد.

این مشکل را با اجرای `IntoIterator` برای `&Grid` و ذخیره یک reference به این `Grid` در `GridIter` برطرف کنید.

همین مشکل می‌تواند برای انواع کتابخانه است‌اندارد رخ دهد: برای `e` در `some_vector` مالکیت `some_vector` را در اختیار می‌گیرد و روی عناصر متعلق به آن بردار تکرار می‌شود. به جای آن از `e` در `some_vector&` برای تکرار بر روی ارجاعات به عناصر `some_vector` استفاده کنید.

FromIterator 25.3

گزینه `FromIterator` به شما امکان می دهد از یک [Iterator](https://doc.rust-lang .org/std/iter/trait.Iterator.html) یک مجموعه یا collection بسازید.

```
                                                                  } ()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
                                                                  {
```

This slide should take about 5 minutes.

پیاده‌سازی `Iterator`

```
                                                        fn collect<B>(self) -> B
                                                                       where
                                                    ,<B: FromIterator<Self::Item
                                                                Self: Sized
```

دو راه برای تعیین `B` برای این روش وجود دارد:

- با کمک "turbofish”: some_iterator.collect::<COLLECTION_TYPE>()، همان‌طور که نشان داده شده است و به طور خاص‌تر _ استفاده شده در این‌جا به Rust امکان می‌دهد نوع صریح را استنتاج کند.
- با نوع inference: let prime_squares: Vec<_> = some_iterator.collect() این مثال را برای استفاده از این فرم بازنویسی کنید.

پیاده‌سازی‌های اولیه FromIterator برای Vec, HashMap و غیره وجود دارد. همچنین پیاده‌سازی‌های تخصصی‌تری وجود دارد که به شما امکان می‌دهد کارهای جالبی مانند تبدیل Iterator<Item = Result<V, E>> به یک Result<Vec<V>, E> انجام دهید.

25.4 تمرین: روش Iterator Chaining

در این تمرین، با یک برخی از روش های ارائه شده در Iterator را برای پیاده‌سازی یک ویژگی پیدا کنید و از آن‌ها برای محاسبه پیچیده استفاده کنید.

کد زیر را در https://play.rust-lang.org/ کپی کنید و تست ها را قبول کنید. از یک عبارت تکرارکننده (iterator) استفاده کنید و نتیجه را جمع‌آوری (collect) کنید تا مقدار بازگشتی را بسازید.

```
,`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
                                                                        {

                                                                    [test]#
                                                    } ()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
                                                                        {

                                                                    [test]#
                                                  } ()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
                                                                        {

                                                                    [test]#
                                                    } ()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
                                                                       ;(
                                                                        {
```

```
                                                                [test]#
                                                    } ()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 راجل

```
,`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
                                                                          {

                                                                [test]#
                                                         } ()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
                                                                          {

                                                                [test]#
                                                       } ()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
                                                                          {

                                                                [test]#
                                                     } ()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
                                                                    ;(
                                                                          {

                                                                [test]#
                                                    } ()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
```

فصل 26

ماژول‌ها

این بخش بای حدود ۴۰ دقیقه طول بکشد. آن شامل:

اسلاید	مدت زمان
ماژول‌ها	۳ دقیقه
سلسله مراتب فایل‌سیستم	۵ دقیقه
قابلیت دید	۵ دقیقه
use, super, self	۱۰ دقیقه
تمرین: ماژول‌هایی برای کتابخانه رابط کاربری گرافیکی	۱۵ دقیقه

26.1 ماژول‌ها

دیده‌ایم که چگونه بلوک‌های `impl` به ما اجازه می‌دهند تا namespace functions را به یک type تبدیل کنیم.

به طور مشابه، `mod` به ما اجازه می‌دهد تا توابع و namespace type به این صورت داشته باشیم:

```
mod foo {
    pub fn do_something() {
        println!("در ماژول foo");
    }
}

mod bar {
    pub fn do_something() {
        println!("در ماژول نواری");
    }
}

fn main() {
    foo::do_something();
    bar::do_something();
}
```

.This slide should take about 3 minutes

- بسته‌های Packageای عمل‌کردی را ارائه می‌کنند و شامل یک فایل `Cargo.toml` می‌شوند که نحوه ساخت بسته‌ای از crateهای 1+ را شرح می‌دهد.
- در واقع Crateها درختی از ماژول‌ها هستند که در آن یک crate باینری یک فایل اجرایی ایجاد می‌کند و یک crate کتابخانه در یک کتابخانه کامپایل می‌شود.
- ماژول‌های scope، organization، و تمرکز این بخش را تعریف می‌کنند.

26.2 سلسله مراتب فایل‌سیستم

حذف محتوای ماژول به Rust می‌گوید که آن را در فایل دیگری جست‌جو کند:

```
mod garden;
```

این به rust می‌گوید که محتوای ماژول `garden` در `src/garden.rs` یافت می‌شود. به طور مشابه، ماژول `garden::vegetables` را می‌توان در `src/garden/vegetables.rs` یافت.

ریشه `crate` در:

- `src/lib.rs` (for a library crate)
- `src/main.rs` (for a binary crate)

ماژول‌های تعریف‌شده در فایل‌ها را نیز می‌توان با استفاده از «کامنت‌های مستند داخلی» مستند کرد. این‌ها موردی را که حاوی آن‌ها است - در این مورد، یک ماژول مستند می‌کنند.

```
//! 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!()
}
```

.This slide should take about 5 minutes

- قبل از Rust 2018، ماژول‌ها باید به جای `module.rs` در `module/mod.rs` قرار می‌گرفتند و این هنوز یک جایگزین کارآمد برای نسخه‌های بعد از 2018 است.
- دلیل اصلی معرفی `filename.rs` به‌عنوان جایگزین `filename/mod.rs` این بود که تشخیص بسیاری از فایل‌ها با نام `mod.rs` در IDEها دشوار است.
- لانه‌گزینی(nesting) عمیق‌تر می‌توانند از folderها استفاده کنند، حتی اگر ماژول اصلی یک فایل باشد:

```
/src
  main.rs ──┤
  top_module.rs ──┤
```

```
/top_module ──└
sub_module.rs ──└
```

- مکانی که rust به دنبال ماژول‌ها می‌گردد را می‌توان با دستور کامپایلر تغییر داد:

```
#[path = "some/path.rs"]
mod some_module;
```

برای مثال، اگر می‌خواهید تست‌هایی را برای یک ماژول در فایلی به نام `some_module_test.rs` قرار دهید، مفید است، شبیه به قرارداد (convention) در Go.

26.3 قابلیت دید

ماژول‌ها یکی از چند خصوصیت زیر را دارند:

- گزینه‌های ماژول به طور پیش‌فرض private هستند (جزئیات پیاده سازی را پنهان می‌کنند).
- گزینه‌های sibling و sibling همیشه قابل مشاهده است.
- به عبارت دیگر، اگر یک مورد در ماژول `foo` قابل مشاهده باشد، در همه فرزندان`foo` قابل مشاهده است.

```
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();
}
```

This slide should take about 5 minutes.

- از کلمه کلیدی pub برای public کردن ماژول‌ها استفاده کنید.

علاوه بر این، مشخص‌کننده‌ای پیش‌فرته pub(...) برای محدود کردن دامنه دید عمومی وجود دارد.

- این آدرس با بپینید Rust Reference.
- پیکربندی قابل نمایش بودن pub(crate) یک الگوی رایج است.
- این مورد کمتر متداول است، شما می‌توانید به یک ریسمان خاص دید بدهید.

- در هر صورت، قابلیت دیدن باید به یک ماژول والد (و همه فرزندان آن) داده شود.

use, super, self 26.4

یک ماژول می‌تواند نام‌ها را از ماژول دیگری با use وارد محدوده کند. شما معمولاً چیزی شبیه به این را در بالای هر ماژول خواهید دید:

```
use std::collections::HashSet;
use std::process::abort;
```

مسیر

مسیرها (Paths) به شرح زیر حل می‌شوند:

1. به عنوان یک path نسبی:
   - در واقع foo یا foo به self::foo در ماژول فعلی اشاره دارد،
   - super::foo refers to foo in the parent module.
2. به عنوان یک path مطلق:
   - crate::foo به foo در ریشه جعبه فعلی اشاره دارد,
   - یک bar::foo به foo در bar crate اشاره دارد.

This slide should take about 8 minutes.

- این"re-export" نام‌ها در مسیر کوتاه‌تری معمول است. برای مثال، lib.rs سطح بالا در یک crate ممکن است داشته باشد

```
mod storage;

pub use storage::disk::DiskStorage;
pub use storage::network::NetworkStorage;
```

در دسترس قرار دادن DiskStorage و NetworkStorage برای سایر crateها با یک مسیر راحت و کوتاه.

- در بیشتر موارد، فقط مواردی که در یک ماژول ظاهر می‌شوند باید use شوند. با این‌حال، یک ویژگی (trait) باید در محدوده باشد تا بتوان هر method ای را روی آن ویژگی فراخوانی کرد، حتی اگر نوعی که آن ویژگی را اجرا می کند قبلاً در محدوده باشد. به عنوان مثال، برای استفاده از متد read_to_string در نوعی که ویژگی Read را اجرا می‌کند، باید از use std::io::Read استفاده کنید.
- عبارت use می‌تواند دارای علامت عام باشد: use std::io::*;. از این کار منع شده است زیرا مشخص نیست کدام موارد import می‌شوند و ممکن است در طول زمان تغییر کنند.

26.5 تمرین: ماژول‌های برای برنامه کاربری گرافیکی

در این تمرین، یک پیاده‌سازی کتابخانه GUI کوچک را دوباره سازمان‌دهی خواهید کرد. این کتابخانه یک ویژگیWidget و چند پیاده‌سازی از آن ویژگی و همچنین یک تابع main را تعریف می‌کنند.

معمول است که هر نوع یا مجموعه‌ای از انواع مرتبط نزدیک را در ماژول خود قرار دهید، بنابراین هر نوع ویجت باید ماژول خاص خود را داشته باشد.

Cargo Setup

ی‌ک Rust playground ف‌ط‌ق از ی‌ک ف‌ای‌ل پ‌ش‌ت‌ی‌ب‌ان‌ی می‌ک‌ن‌د، ب‌ن‌اب‌ر‌ای‌ن ی‌ک پ‌روژه Cargo را در س‌ی‌س‌ت‌م ف‌ای‌ل م‌ح‌ل‌ی خ‌ود ای‌ج‌اد ک‌ن‌ی‌د:

```
cargo init gui-modules
cd gui-modules
cargo run
```

ای‌ن src/main.rs ح‌اص‌ل را وی‌را‌ی‌ش ک‌ن‌ی‌د ت‌ا ع‌ب‌ارات mod را اض‌اف‌ه ک‌ن‌ی‌د و ف‌ای‌ل‌ه‌ای اض‌اف‌ی را در دای‌رک‌ت‌وری src اض‌اف‌ه ک‌ن‌ی‌د.

ع‌ب‌ن‌م

در ای‌ن‌ج‌ا اج‌را‌ی ت‌ک م‌اژول ک‌ت‌اب‌خ‌ان‌ه GUI آم‌ده اس‌ت:

```
                                    } 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
اس کچوک ینتم رایب GUI یشیامن خسن ک ین")window.add_widget(Box::new(Label::new
                    ;((("!window.add_widget(Box::new(Button::new("Click me
                                                    ;()window.draw
                                                            {
```

.This slide and its sub-slides should take about 15 minutes

دانش‌آموزان را تشویق کنید تا کد را به‌گونه‌ای تقسیم کنند که برای‌شان طبیعی است و به اعلان‌های mod, use و pub عادت کنند. پس از آن، در مورد این‌که چه organizationهای idiomatic دست‌به تحت کنید.

26.5.1 راه‌حل

```
                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
ک ینتم یارب GUI یشیامن هخسن ک ین")add_widget(Box::new(widgets::Label::new.
                        ;(("!window.add_widget(Box::new(widgets::Button::new("Click me
                                                                ;()window.draw
                                                                        {
```

فصل 27

تست‌کردن

این بخش باید حدود ۴۵ دقیقه طول بکشد. آن شامل:

اسلاید	مدت زمان
تست ماژول‌ها	۵ دقیقه
انواع دیگر تست‌ها	۵ دقیقه
کامپایلر Lints و Clippy	۳ دقیقه
تمرین: الگوریتم Luhn	۳۰ دقیقه

27.1 تست‌های واحد (Unit Tests)

Rust and Cargo با یک چارچوب تست واحد ساده ارائه می شود:

- Unit tests are supported throughout your code.
- تست‌های یکپارچه‌سازی از طریق دایرکتوری `tests/` پشتیبانی می‌شوند.

تست‌ها با `#[test]` علامت‌گذاری شده‌اند. تست‌های واحد اغلب در یک ماژول `tests` تودرتو قرار می‌گیرند و از `#[cfg(test)]` استفاده می‌کنند تا آن‌ها را به صورت مشروط تنها در هنگام build تست‌ها کامپایل کنند.

```
} fn first_word(text: &str) -> &str
        } (' ')match text.find
 ,[Some(idx) => &text[..idx
            ,None => &text
                        {
                            {

                [(cfg(test]#
                 } mod tests
              ;*::use super

                      [test]#
                  } ()fn test_empty
        ;("." ,(".")assert_eq!(first_word
```

```
{

[test]#
} ()fn test_single_word
;("سلام" ,("سلام")assert_eq!(first_word
{

[test]#
} ()fn test_multiple_words
;("سلام" ,("سلام دنیا!")assert_eq!(first_word
{
{
```

- این به شما امکان می‌دهد تا private helper را آزمایش کنید.
- ویژگی `#[cfg(test)]` تنها زمانی فعال است که `cargo test` را اجرا کنید.

This slide should take about 5 minutes.

تست‌ها را در playground اجرا کنید تا نتیجه‌ی آن‌ها را نشان دهید.

27.2 انواع دیگر تست‌ها

Integration Tests

اگر می‌خواهید کتابخانه خود را به عنوان یک سرویس‌گیرنده آزمایش کنید، از تست یکپارچه‌سازی (integration test) استفاده کنید.

یک فایل `.rs` در زیر `/tests` بسازید:

```
// tests/my_library.rs
use my_library::init;

#[test]
fn test_init() {
    assert!(init().is_ok());
}
```

این آزمایش‌ها فقط به public API مربوط به crate شما دسترسی دارند.

تست‌نده‌ها

زبان Rust دارای پشتیبانی داخلی برای تست‌های مستندسازی است:

```
/// 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())]
}
```

- بل‌وک‌های کد در commentهای `///` به طور خودکار به عنوان کد Rust دیده می‌شوند.
- این کد به عنوان بخشی از `cargo test` کامپایل و اجرا می‌شود.
- افزودن `#` به کد، آن را از مستندات پنهان می‌کند، اما همچنان آن را کامپایل/اجرا می‌کند.
- کد بالا را در Rust Playground تست کنید.

27.3 کامپایلر Lints و Clippy

کامپایلر Rust پیام‌های خطای جالب و همچنین built-in lint مناسبی تولید می‌کند. Clippy که linterای بیشتری را ارائه می‌دهد، که در گروه‌های سازمان‌دهی شده‌اند که می‌توانند در هر پروژه فعال شوند.

```
#[deny(clippy::cast_possible_truncation)]
fn main() {
    let x = 3;
    while (x < 70000) {
        x *= 2;
    }
    println!("X{}X" حاصل‌ضرب در یک u16 مناسب است، درست است؟ در یک u16 حاصل‌ضرب مناسب است);
}
```

This slide should take about 3 minutes.

نمونه کد را اجرا کنید و پیام خطا را بررسی کنید. linterای زیر در اینجا قابل مشاهده نیستند، اما پس از کامپایل شدن کد، آن‌ها نشان‌داده نمی‌شوند. برای نمایش آن linterها به سایت Playground بروید.

پس از رفع linterها، `clippy` را در سایت playground اجرا کنید تا هشداره‌ای clippy نشان داده شود. Clippy مستندات گسترده‌ای از linterای خود دارد و همیشه linterای جدید (از جمله default-deny lint) را اضافه می‌کند.

توجه داشته باشید که خطاها یا هشداره‌ای مربوط به `help: ...` را می‌توان با `cargo fix` یا از طریق ویرایشگر خود برطرف کرد.

27.4 تمرین: الگوریتم Luhn

الگوریتم Luhn

الگوریتم Luhn برای اعتبارسنجی شماره‌ای کارت اعتباری استفاده می‌شود. این الگوریتم یک رشته را به عنوان ورودی دریافت می‌کند و برای اعتبارسنجی شماره کارت اعتباری مراحل زیر را انجام می‌دهد:

- Ignore all spaces. Reject numbers with fewer than two digits.
- با حرکت از سمت راست رشته به چپ، هر دومین رقم را دوبل کنید: برای شماره 1234، 3 و 1 را دوبل می‌کنیم. برای شماره 98765، 6 و 8 را دوبل می‌کنیم.
- پس از دوبل کردن یک عدد، اگر اون جفت خروجی بیش از 9 باشد، ارقام را جمع کنید. بنابراین، دوبل کردن 7 به 14 تبدیل می‌شود که به `5 + 4 = 1` تبدیل می‌شود.
- تمام ارقام دو برابر نشده و دو برابر شده را جمع کنید.
- اگر مجموع با 0 خاتمه یابد، شماره کارت اعتباری معتبر است.

کد ارائه شده یک پیاده‌سازی باگ از الگوریتم luhn را به همراه دو unit test پایه ارائه می‌کند که تأیید می‌کند بیشتر الگوریتم به درستی پیاده‌سازی شده است.

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
{

[(cfg(test]#
} mod test
;*::use super

[test]#
} ()fn test_valid_cc_number
;("assert!(luhn("4263 9826 4026 9299
;("assert!(luhn("4539 3195 0343 6467
;("assert!(luhn("7992 7398 713
{

[test]#
} ()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 راەحل

```
.This is the buggy version that appears in the problem //
[(cfg(never]#
} pub fn luhn(cc_number: &str) -> bool
;let mut sum = 0
;let mut double = false
```

```rust
    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
}
```

```rust
// 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!(
        "آیا {cc_number} یک شماره کارت اعتباری معتبر است؟ {}",
        if luhn(cc_number) { "بله" } else { "نه" }
```

```
                                                    ;(
                                                        {

                                                [(cfg(test]#
                                                  } mod test
                                              ;*::use super

                                                    [test]#
                                                } ()fn test_valid_cc_number
                                     ;(("assert!(luhn("4263 9826 4026 9299
                                     ;(("assert!(luhn("4539 3195 0343 6467
                                           ;(("assert!(luhn("7992 7398 713
                                                    {

                                                    [test]#
                                                } ()fn test_invalid_cc_number
                                    ;(("assert!(!luhn("4223 9826 4026 9299
                                    ;(("assert!(!luhn("4539 3195 0343 6476
                                    ;(("assert!(!luhn("8273 1232 7352 0569
                                                    {

                                                    [test]#
                                                } ()fn test_non_digit_cc_number
                                                     ;(("assert!(!luhn("foo
                                                 ;(("assert!(!luhn("foo 0 0
                                                    {

                                                    [test]#
                                                  } ()fn test_empty_cc_number
                                                     ;((".")assert!(!luhn
                                                     ;((" ")assert!(!luhn
                                                   ;(("  ")assert!(!luhn
                                                  ;(("    ")assert!(!luhn
                                                    {

                                                    [test]#
                                            } ()fn test_single_digit_cc_number
                                                     ;(("assert!(!luhn("0
                                                    {

                                                    [test]#
                                                  } ()fn test_two_digit_cc_number
                                                     ;((" assert!(luhn(" 0 0
                                                    {
                                                        {
```

بخش VIII

روز چهارم: بعد از ظهر

فصل 28

خوش آمد

Including 10 minute breaks, this session should take about 2 hours and 15 minutes. It contains:

بخش	مدت زمان
مدیریت خطا	۱ ساعت
ایمنی Rust	ساعت و ۵ دقیقه

فصل 29

مدیریت خطا

این بخش برای حدود ۱ ساعت طول بکشد. این شامل:

اسلاید	مدت زمان
در مورد Panicها	۳ دقیقه
Result	۵ دقیقه
عملگرد Try	۵ دقیقه
این تبدیل‌ها (Conversions) را امتحان کنید	۵ دقیقه
Error Trait	۵ دقیقه
thiserror و anyhow	۵ دقیقه
تمرین: بازنویسی با Result	۳۰ دقیقه

29.1 در مورد Panicها

Rust خطاهای مهلک را با "panic" کنترل می‌کند.

اگر یک خطای مرگبار در زمان اجرا رخ دهد، Rust باعث panic می‌شود:

```
fn main() {
    let v = vec![10, 20, 30];
    println!("v[100]: {}", v[100]);
}
```

- استفاده از Panicها برای خطاهای غیرقابل جبران و غیرمنتظره است.
  - پانیک‌ها عالیم باگ در برنامه‌ها هستند.
  - خرابی‌های زمان اجرا مانند failed bounds checkها می‌توانند باعث panic شوند
  - Assertions (such as `assert!`) panic on failure
  - پنیک‌های خاص می‌توانند از ماکرو `panic!` استفاده کنند.
- یک panic را "باز" می‌کند و مقادیر را حذف می‌کند درست مثل این‌که توابع برگشته باشند.
- اگر خرابی قابل قبول نیست، از APIهای بدون panic (مانند `Vec::get`) استفاده کنید.

This slide should take about 3 minutes.

به طور پیش‌فرض، panic باعث unwind شدن stack می‌شود. unwinding را می‌توان گرفت(در واقع منظور این است که می‌توان آن را caught کرد):

```
use std::panic;

fn main() {
    let result = panic::catch_unwind(|| {
        println!("{result?}");
    });

    let result = panic::catch_unwind(|| {
        panic!("oh no!");
    });
    println!("{result?}");
}
```

- گرفتن (Catching) غیرمعمول است. سعی نکنید exceptionها را با `catch_unwind` پیاده‌سازی کنید!
- این کار می‌تواند در سرورهای مفید باشد که حتی در صورت خراب شدن یک درخواست، باید به کار خود ادامه دهند.
- اگر `panic = 'abort'` در `Cargo.toml` شما تنظیم شده باشد، این مورد کار نمی‌کند.

Result 29.2

مکانیسم اصلی ما برای مدیریت خطا در Rust، فه‌رست `Result` است، که به طور خلاصه هنگام بحث در مورد تای‌پ‌های کتابخانه استاندارد مشاهده کردیم.

```
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!("فایل دفترچه خاطرات با {bytes} بایت خوانده شد: {contents}");
            } else {
                println!("فایل را خواند اما نتوانست محتوای آن را به رشته تبدیل کند");
            }
        }
        Err(err) => {
            println!("باز کردن فایل دفترچه خاطرات با خطا مواجه شد: {err}");
        }
    }
}
```

This slide should take about 5 minutes.

- `Result` has two variants: `Ok` which contains the success value, and `Err` which contains an error value of some kind.
- این‌که آیا یک تابع می‌تواند خطا ایجاد کند یا نه، در signature نوع تابع برگرداندن با نوع تابع کدگذاری می‌شود.
- مانند `Option`، هیچ راهی برای فراموش کردن خطا وجود ندارد: شما نمی‌توانید به مقدار موفقیت‌آمیز مقدار خطا بدون تطبیق الگوی اولیه در `Result` دست پیدا کنید تا بررسی کنید که نوع مدام خطا

را دارید. روش‌هایی مانند unwrap نوشتن کده‌ای سریع و کثیف را آسان‌تر می‌کنند که مدیریت خطا را به خوبی انجام نمی‌دهد، اما به این معنی است که همیشه می‌توانید در کد منبع خود ببینید که در کجا مدیریت صحیح خطا نادیده گرفته می‌شود.

برای کاوش بیشتر

مقایسه مدیریت خطا در Rust با قراردادهای مدیریت خطا که دانش‌آموزان ممکن است با سایر زبان‌های برنامه‌نویسی آشنا باشند، ممکن است مفید باشد.

استثناها

- بسیاری از زبان‌ها از exception‌ها استفاده می‌کنند، به عنوان مثال. C++، جاوا، پایتون.
- در اکثر زبان‌های دارای exception، این که آیا یک تابع می‌تواند استثنا ایجاد کند یا نه، به عنوان بخشی از نوع امضای(signature) آن قابل مشاهده نیست. این به طور کلی به این معنی است که هنگام فراخوانی یک تابع نمی‌توانید بگویید که آیا ممکن است یک exception ایجاد کند یا خیر.
- استثناها معمولاً call stack را باز می‌کنند و تا رسیدن به بلوک try به سمت بالا منتشر می‌شوند. خطایی که در اعماق call stack ایجاد می‌شود ممکن است بر عملکرد نامرتبط بی‌تأثیر بگذارد.

شماره‌ای خطا

- برخی از زبان‌ها دارای تابع‌هستند که یک عدد خطا (یا مقداری خطای دیگر) را جدا از مقدار بازگشت موفقیت‌آمیز تابع برمی‌گردانند. به عنوان مثال می توان به C و Go اشاره کرد.
- بسته به زبان ممکن است فراموش کنید مقدار خطا را بررسی کنید، در این صورت ممکن است به یک مقدار موفقیت نامعتبر یا نامعتبر دسترسی داشته باشید.

29.3 عملگر Try

خطاهای زمان اجرا مانند connection-refused یا file-not-found با نوع «نتیجه» مدیریت می‌شوند، اما تطبیق این نوع در هر تماس می‌تواند دشوار باشد. اپراتور ? برای برگرداندن خطاها به تماس گیرنده استفاده می‌شود. این به شما امکان می‌دهد تا موارد مشترک را بازگردانید.

```
} match some_expression
,Ok(value) => value
,(Err(err) => return Err(err
{
```

بسیار ساده‌تر

```
?some_expression
```

می‌توانم از این برای ساده‌سازی کد رسیدگی به خطا استفاده کنم:

```
;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} :اطخ ای یربارکمان")!println
                                                    {
```

.This slide should take about 5 minutes

ناب `read_username` را برای استفاده از ? ساده کنید.

نکات کلیدی:

- متغیر `username` می تواند `Ok(string)` یا `Err(error)` باشد.
- از فراخوانی `fs::write` برای آزمایش سناریوهای مختلف استفاده کنید: بدون فایل، فایل خالی، فایل با نام کاربری.
- توجه داشته باشید که `main` تا زمانی که `std::process::Termination` را پیاده‌سازی کند، می‌تواند نتیجه‌ای `<E> ,()` را برگرداند. در عمل، این بدان معنی است که `E` پیاده‌سازی `Debug` را انجام‌می‌دهد. فایل اجرایی، نوع `Err` را چاپ می‌کند و در صورت خطا، وضعیت خروج غیر صفر (nonzero) را برمی‌گرداند.

29.4 این تبدیل‌ها (Conversions) را امتحان کنید

گسترش مؤثر ? کمی پیچیده‌تر از آنچه قبلاً ذکر شد است:

```
expression?
```

به طور مشابه کار می‌کند

```
                                } match expression
                                ,Ok(value) => value
                        ,((Err(err)  => return Err(From::from(err
                                                            {
```

فراخوانی `From::from` در اینجا به این معنی است که ما سعی می کنیم نوع خطا را به نوع بازگشتی توسط تابع تبدیل کنیم. این باعث می‌شود که خطاهای سطح بالاتر کپسوله شوند.

مثال

```
                                        ;use std::error::Error
                        ;{use std::fmt::{self, Display, Formatter
                                        ;use std::fs::File
                                ;{use std::io::{self, Read
```

```
[(derive(Debug]#
} 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
"دشن تفای یربارک مان {path}  نین رد" ,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} :اطخ ای یربارکمان")!println
{
```

.This slide should take about 5 minutes

ع‌مل‌گر ? با‌ید مق‌داری س‌ازگار با نوع با‌زگش‌تی تا‌ب‌ع ب‌رگ‌ردان‌د. ب‌رای Result، ب‌ه ای‌ن مع‌نی ا‌س‌ت ک‌ه ان‌واع خ‌طا با‌ید س‌ازگار با‌ش‌ن‌د. تا‌ب‌ع‌ی ک‌ه Result<T, ErrorOuter> را ب‌رمی‌گ‌ردان‌د، ت‌ن‌ه‌ا می‌ت‌وان‌د از ? در مق‌داری از تای‌پ‌ه‌ای Result<U, ErrorInner> ا‌س‌ت‌فاده ک‌ن‌د اگ‌ر ErrorOuter و ErrorInner ی‌ک نوع با‌ش‌ن‌د ی‌ا اگ‌ر ErrorOuter از‘ را پ‌ی‌اده‌س‌ازی ک‌ن‌د.

ی‌ک چای‌گ‌زی‌ن رای‌ج ب‌رای پ‌ی‌اده س‌ازی Fromچ‌ه‌ت Result::map_err ا‌س‌ت، ب‌ه خ‌ص‌وص زمان‌ی ک‌ه ت‌ب‌دی‌ل‌ه‌ا ف‌ق‌ط در ی‌ک م‌ک‌ان ان‌ج‌ام می‌ش‌ود.

ه‌ی‌چ ال‌زام‌ی ب‌رای س‌ازگ‌اری Option وج‌ود ن‌دارد. تا‌ب‌ع‌ی ک‌ه Option<T> را ب‌رمی گ‌ردان‌د می ت‌وان‌د از ع‌مل‌گر ? در Option<U> ب‌رای ان‌واع دل‌خ‌واه T و U ا‌س‌ت‌فاده ک‌ن‌د.

ی‌ک تا‌ب‌ع‌ی ک‌ه Result را ب‌رمی‌گ‌ردان‌د دی‌گ‌ر ن‌می‌ت‌وان‌د از ? در Option ا‌س‌ت‌فاده ک‌ن‌د و ب‌ال‌ع‌ک‌س. ب‌ا ای‌ن-

حال، `Option::ok_or` یک `Option` را به `Result` تبدیل می‌کند در حالی که `Result::ok` «نتیجه» را به `Option` تبدیل می‌کند.

29.5 انواع خطاهای Dynamic

گاهی اوقات می‌خواهیم اجازه دهیم هر نوع خطای بدون نوشتن enum خودمان که تمام احتمالات مختلف را پوشش می‌دهد، برگردانده شود. ویژگی `std::error::Error` یک object مشخص است که می‌تواند حاوی هر خطایی باشد را آسان می‌کند.

```
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") {
        Ok(count) => println!("تعداد: {count}"),
        Err(err) => println!("Error: {err}"),
    }
}
```

This slide should take about 5 minutes.

`read_count` می‌تواند انواع `std::io::Error` (از عملیات فایل) یا `std::num::ParseIntError` (از `String::parse`) را برگرداند.

خطاهای Boxing باعث صرفه‌جویی در کد می شود، اما توانایی رسیدگی به موارد خطای مختلف را به طور متفاوت در برنامه از بین می‌برد. به این ترتیب استفاده از `Box<dyn Error>` در یک public API کتابخانه ایده خوبی نیست، اما می‌توانند گزینه خوبی در برنامه‌ای باشد که فقط می‌خواهید پیام خطا را در جایی نمایش دهید.

هنگام تعریف یک نوع خطای سفارشی، مطمئن شوید که ویژگی `std::error::Error` را اجرا کنید تا بتوان آن را در جعبه قرار داد. اما اگر نیاز به پشتیبانی از ویژگی `no_std` دارید، به خاطر داشته باشید که ویژگی `std::error::Error` در حال حاضر با `no_std` در nightly سازگار است.

29.6 `thiserror` و `anyhow`

این `thiserror` و anyhow crate ها به طور گسترده ای برای ساده کردن رسیدگی به خطا استفاده می‌شوند.

- `thiserror` is often used in libraries to create custom error types that implement `From<T>`.
- اغلب anyhow توسط برنامه‌ها برای کمک به مدیریت خطا در توابع، از جمله افزودن اطلاعات متنی به خطاهای شما، استفاده می‌شود.

```
;{use anyhow::{bail, Context, Result
                                    ;use std::fs
                                ;use std::io::Read
                             ;use thiserror::Error

                  [(derive(Clone, Debug, Eq, Error, PartialEq]#
                     [("دشن تفای {0} در یربارکامن چیه")error]#
                           ;(struct EmptyUsernameError(String

                 } <fn read_username(path: &str) -> Result<String
               ;(let mut username = String::with_capacity(100
                                        (fs::File::open(path
          ?(("دشن زاب {with_context(|| format!("{path.
                              (read_to_string(&mut username.
                              ;?("شکست در خواندن")context.
                                        } ()if username.is_empty
         ;((((bail!(EmptyUsernameError(path.to_string
                                                        {
                                                (Ok(username
                                                            {

                                                    } ()fn main
                                        ;()fs::write("config.dat", "").unwrap//
                                            } ("match read_username("config.dat
    ,("{username} :یربارکامن")!Ok(username) => println
                  ,("{?:Err(err) => println!("Error: {err
                                                        {
                                                            {
```

.This slide should take about 5 minutes

thiserror

- ماکرو استخراج Error توسط thiserror ارائه می‌شود و دارای ویژگی‌های مفید زیادی برای کمک به تعریف انواع خطا به روشی فشرده است.
- ویژگی std::error::Error به طور خودکار مشتق می‌شود.
- پیام #[error] برای استخراج ویژگی Display استفاده می‌شود.

anyhow

- anyhow::Error اساساً پوشش (wrapper) در اطراف Box<dyn Error> است. به این ترتیب معمولاً انتخاب خوبی برای API عمومی یک کتابخانه نیست، اما به طور گسترده در برنامه‌های مختلف استفاده می‌شود.
- anyhow::Result<V> یک type مستعار برای است Result<V, anyhow::Error>.
- در صورت لزوم می‌توان نوع خطای واقعی داخل آن را برای بررسی استخراج کرد.
- anyhow::Result<T> عملکرد ارائه شده توسط ممکن است برای توسعه‌دهندگان Go آشنا باشد، زیرا الگوهای استفاده و ارگونومی مشابهی را با (T, error) از Go ارائه می‌دهد.
- anyhow::Context یک ویژگی است که برای typeهای است‌دارد Result و Option پیاده‌سازی شده است. use anyhow::Context برای فعال کردن .context() و .with_context() در آن typeها ضروری است.

29.7 تمرین: بازنویسی با Result

در زیر یک تجزیه‌کننده ساده برای یک زبان عبارت پیاده‌سازی می‌کند. با این حال، با panic خطاها را کنترل می‌کند. آن را بازنویسی کنید تا به جای آن از مدیریت خطای اصطلاحی استفاده کند و خطاها را به بازگشت از main منتشر کند. با خیال راحت از thiserror و anyhow استفاده کنید.

نکته: با رفع خطا در عملکرد parse شروع کنید. هنگامی که به درستی کار کرد، Tokenizer را برای پیاده‌سازی Iterator<Item=Result<Token, TokenizerError>> به‌روزرسانی کنید و آن را در parser کنترل کنید.

```
use std::iter::Peekable;
use std::str::Chars;

/// An arithmetic operator.
#[derive(Debug, PartialEq, Clone, Copy)]
enum Op {
    Add,
    Sub,
}

/// A token in the expression language.
#[derive(Debug, PartialEq)]
enum Token {
    Number(String),
    Identifier(String),
    Operator(Op),
}

/// An expression in the expression language.
#[derive(Debug, PartialEq)]
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} هرظتنمرغی رتکاراک")!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
                                    ;("پایان غیرمنتظره در ورودی")!panic
                                                                    ;{
                                                    } let expr = match tok
                                                } <= (Token::Number(num
    ;("ربتعمان  bit-32 حیحص ددع")let v = num.parse().expect
                                                (Expression::Number(v
                                                                    {
                ,(Token::Identifier(ident) => Expression::Var(ident
        ,("{?:tok} هرظتنمرغی نکوت")!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} هرظتنمرغی نکوت")!Some(tok) => panic
                                                {
                                                    {

                                        (parse_expr(&mut tokens
                                                            {

                                                } ()fn main
                                    ;("let expr = parse("10+foo+20-30
                                                ;("{?:println!("{expr
                                                            {
```

لحاره 29.7.1

```
                                                ;use thiserror::Error
                                            ;use std::iter::Peekable
                                                ;use std::str::Chars

                                                .An arithmetic operator ///
                                        [(derive(Debug, PartialEq, Clone, Copy]#
                                                        } enum Op
                                                         ,Add
                                                         ,Sub
                                                            {

                                                .A token in the expression language ///
                                                    [(derive(Debug, PartialEq]#
                                                                } enum Token
                                                            ,(Number(String
                                                        ,(Identifier(String
                                                             ,(Operator(Op
                                                            {

                                                .An expression in the expression language ///
                                                    [(derive(Debug, PartialEq]#
                                                                } 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
                                                            {

                                                        [(derive(Debug, Error]#
                                                          } enum TokenizerError
                                                [("کاراک رتغی هرظتنمر "{0}" در ورودی")error]#
```

```
                                                ,(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 <= _
                                                                      {
                                                                          {
                                                                              {

                                                        [(derive(Debug, Error]#
                                                            } enum ParserError
                                                    [("{0} :خطای توکن‌سازی")error]#
                                                ,(TokenizerError(#[from] TokenizerError
                                                    [("پایان غیرمنتظره در ورودی")error]#
                                                                    ,UnexpectedEOF
                                                [("{?:0} توکن (token) غیرمنتظره")error]#
                                                                ,(UnexpectedToken(Token
                                                                [("ربت‌عمان هارمش")error]#
                                            ,(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
                                                                          {
```

فصل 30

Rust نمایان

این بخش دارای حدود ۱ ساعت و ۵ دقیقه طول بکشد.و شامل موارد زیر است:

اسلایدها	مدت زمان
نمایان	۵ دقیقه
عدم ارجاع به اشاره‌گرهای خام	۱۰ دقیقه
متغیرهای ثابت قابل تغییر	۵ دقیقه
توابع دانگدچنده	۵ دقیقه
توابع نامان	۵ دقیقه
صفات (Traits) نامان	۵ دقیقه
تمرین: FFI Wrapper	۳۰ دقیقه

30.1 نمایان Rust

زبان Rust دو بخش دارد:

- در **Safe Rust:** حافظه ایمن یا memory safe، هیچ رفتار تعریف نشده‌ای امکان پذیر نیست.
- در **Unsafe Rust:** در صورت نقض پیش‌شرط‌ها، می‌توانید باعث رفتار نامشخص شود.

ما عمدتاً safe Rust را در این دوره دیدیم، اما مهم است که بدانیم Unsafe Rust چیست.

کد نامان معمولی کوچک و ایزوله است و صحت آن باید به دقت مستند شود. معمولاً در یک لایه انتزاعی ایمن پیچیده می‌شود.

این Unsafe Rust به شما امکان دسترسی به پنج قابلیت جدید را می‌دهد:

- اشاره‌گرهای خام.
- به متغیرهای mutable static variable دسترسی دهید.
- به فیلدهای `union` دسترسی پیدا کنید.
- توابع `unsafe`، از جمله توابع `extern` را فراخوانی کنید.
- ویژگی‌های `unsafe` را اجرا کنید.

در ادامه به طور خلاصه به قابلیت‌های unsafe می‌پردازیم. برای جزئیات کامل، لطفاً به Chapter 19.1 in the Rust Book و Rustonomicon.

This slide should take about 5 minutes.

همین‌طور Unsafe Rust به این معنی نیست که کد نادرست است. این بدان معنی است که توسعه‌دهندگان برخی از ویژگی‌های امنی کامپایلر را خاموش کرده‌اند و باید کد صحیح را خودشان بنویسند. این بدان معناست که کامپایلر دیگر قواعد امنی Rust را اجرا نمی‌کند.

30.2 عدم ارجاع به اشاره‌گرهای خام

ایجاد اشاره‌گر امن است، اما عدم ارجاع به آن‌ها «ناامن» یا unsafe است:

```
fn main() {
    let mut s = String::from("باشاب بقارم!");

    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 برابر است: {}", *r1);
        r1 = String::from("واوه");
        println!("r2 برابر است: {}", *r2);
    }

    /*
    NOT SAFE. DO NOT DO THIS.
    */
    let r3: &String = unsafe { &*r1 };
    drop(s);
    println!("r3 is: {}", *r3);
}
```

This slide should take about 10 minutes.

این تمرین خوبی است (و طبق راهنمای سبک Android Rust لازم است) برای هر بلوک unsafe یک نظر بنویسید و توضیح دهد که چگونه کد داخل آن الزامات امنی عملیات ناامن را که انجام می‌دهد برآورده می‌کند.

در مورد عدم ارجاع اشاره‌گر، این بدان معنی است که نشانگرها باید *valid* باشند، یعنی:

- اشاره‌گر باید غیر تهی یا non-null باشد.
- اشاره‌گر باید *dereferenceable* باشد (در محدوده یک object اختصاص داده شده).
- این object نباید جابجا شده باشد.
- دسترسی همزمان به یک مکان نباید وجود داشته باشد.
- اگر اشاره‌گر با فرستادن یک reference به دست آمده باشد، object زیرین باید live و نمی‌توان از هیچ مرجعی برای دسترسی به حافظه استفاده کرد.

در بیشتر موارد، اشاره‌گر زیر باید به درستی تراز شود.

بخش «NOT SAFE» نمونه‌ای از یک نوع رایج از اشکال UB را ارائه می‌کند: r1* دارای طول عمر 'static است، بنابراین r3 دارای نوع &'static String است و بنابراین عمر s بیشتر می‌شود. ایجاد یک مرجع از یک اشاره گر نیاز به دقت بسیار دارد.

30.3 متغیرهای ثابت قابل تغییر

خواندن یک متغیر استاتیک تغییرناپذیر بی‌خطر است:

```
static HELLO_WORLD: &str = "سلام دنیا!";

fn main() {
    println!("HELLO_WORLD: {HELLO_WORLD}");
}
```

با این‌حال، از آن‌جایی که شرایط قابل تابیدن داده‌ها ممکن است رخ دهد، خواندن و نوشتن متغیرهای mutable static ناامن است:

```
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}");
    }
}
```

This slide should take about 5 minutes.

- برنامه در اینجا امن است زیرا single-thread است. با این حال، کامپایلر Rust محافظه کار است و بدتر‌ین‌ها را در نظر می‌گیرد. unsafe را حذف کنید و ببینید چگونه کامپایلر توضیح می‌دهد که چه‌طور یک static از چندین thread یک رفتار تعریف نشده است.
- استفاده از یک static قابل تغییر (mutable) به طور کلی ایده بدی است، اما مواردی وجود دارد که ممکن است در کدهای سطح پایین no_std منطقی باشد، مانند اجرای یک heap allocator یا کار با برخی از APIهای مربوط به زبان C.

30.4 نوع داده چندگانه

همین‌طور Unionها مانند enumها هستند، اما شما باید خودتان active field را ردیابی کنید:

```
#[repr(C)]
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
                                                                                {
```

.This slide should take about 5 minutes

به طور کلی Unionها در Rust به ندرت مورد نیاز هستند زیرا معمولاً می‌توانید از enum استفاده کنید.
آن‌ها گاهی اوقات برای تعامل با APIهای کتابخانه C مورد نیاز هستند.

اگر فقط می‌خواهید بایت‌ها را به‌عنوان نوع متفاوتی تفسیر کنید، احتمالاً [std::mem::transmute](https://doc.rust-lang.org/stable/std/mem/fn.transmute.html) را می‌خواهید (یا یک safe wrapper مانند چیزی که
[zerocopy](https://crates.io/crates/zerocopy)).

30.5 توابع ناامن

فراخوانی ناامن

یک function یا method می‌تواند علامت‌گذاری unsafe کرد، اگر دارای پیش‌شرط‌های اضافی باشد که برای جلوگیری از رفتار نامشخص رعایت کنید:

```
                                                    } "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
                                                ;((C: {}", abs(-3 طبق -۳ مطلق مقدار")!println
                                                                                {

            !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
                                                                                {
```

نوشتن متن‌های ناامن

اگر عملکرد‌های خود را برای جلوگیری از رفتار نامشخص به شرایط خاصی نیاز دارند، می‌توانید به‌عنوان unsafe علامت‌گذاری کنید.

```
/// 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);
}
```

This slide should take about 5 minutes.

فراخوانی متن‌های ناامن

توابع get_unchecked، مانند توابع _unchecked، ناامن است، زیرا اگر در محدوده نادرست باشد، می‌تواند یک UB ایجاد کند. abs به دلیل دیگری نادرست است: این یک تابع خارجی (FFI) است. فراخوانی توابع خارجی عموماً زمانی مشکل‌ساز است که آن توابع کاره‌ای را با اشاره‌گره‌ای انجام می‌دهند که ممکن است مدل حافظه Rust را نقض کنند، اما به طور کلی هر تابع C ممکن است تحت هر شرایط دلخواه رفتار نامشخصی داشته باشد.

زبان برنامه‌نویسی "C" در این مثال ABI است. ABIهای دیگر نیز در دسترس هستند.

نوشتن متن‌های ناامن

ما در واقع از pointerها برای یک تابع swap استفاده نمی‌کنیم - این کار را می‌توان به‌طور ایمن با referenceها انجام داد.

توجه داشته باشید که کد ناامن در یک تابع بدون بلوک unsafe مجاز است. ما می‌توانیم این کار را با #[deny(unsafe_op_in_unsafe_fn)] غیرمجاز کنیم. سعی کنید آن را اضافه کنید و ببینید چه اتفاقی می افتد. این احتمالاً در نسخه‌های قدیمی Rust خواهد کرد.

30.6 پیاده سازی صفات (Traits) نامن

مانند توابع، اگر پیاده‌سازی بایدشرایط خاصی را تضمین کند تا از رفتار نامشخص جلوگیری شود، می‌توانید یک ویژگی را به‌عنوان unsafe علامت‌گذاری کنید.

برای مثال، zerocopy crate یک ویژگی نامن دارد که چیزی شبیه به این است:

```
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 {}
```

This slide should take about 5 minutes.

بایدیک بخش # Safety در Rustdoc برای این صفت (trait) وجود داشته باشد که شرایط لازم برای اجرای امن این trait را توضیح دهد.

بخش امنی واقعی برای AsBytes نسبتا طولانی‌تر و پیچیده‌تر است.

ویژگی‌های داخلیSend و Sync نامن (unsafe) هستند.

30.7 امن بودن FFI Wrapper

زبان Rust پشتیبانی بسیار خوبی برای فراخوانی توابع از طریق رابط توابع خارجی *foreign function interface* (FFI) دارد. ما از آن برای ساختن یک پوشش امن برای توابع libc استفاده می‌کنیم که از C برای خواندن نام فایل ها در یک فهرست استفاده می کنید.

شما می‌خواهید به صفحات راهنما مراجعه کنید:

- opendir(3)
- readdir(3)
- closedir(3)

همچنین می‌خواهید std::ffi را مرور کنید

انواع	رمزگذاری	استفاده
str و String	UTF-8	پردازش متن در Rust
CStr and CString	NUL-terminated	ارتباط با توابع C

انواع	رمزگذاری	استفاده
OsStr و OsString	مخصوص سیستم‌عامل	برقراری ارتباط با سیستم‌عامل

شما باید انواع type این مانند لیبت‌ها خواهید کرد:

- &str to CString: you need to allocate space for a trailing \0 character&,
- CString به const i8*: برای فراخوانی توابع C به یک اشاره‌گر نیاز دارید،
- از const i8* به CStr&: به چیزی نیاز دارید که بتواند کاراکتر \0 را پیدا کند،
- CStr& به &[u8]: یک slice بایت universal interface برای «برخی داده‌های ناشناخته» است،
- از &[u8] به &OsStr: OsStr گامی به سوی OsString است، از OsStrExt برای ایجاد آن استفاده کنید،
- &OsStr to OsString: you need to clone the data in &OsStr to be able to return it and& call readdir again.

مورد Nomicon همچنین یک فصل بسیار مفید در مورد FFI دارد.

کد زیر را در https://play.rust-lang.org/ کپی کنید و توابع و متدهای از مفقود شده را پر کنید:

```
// TODO: این را زمانی که پیاده‌سازی‌تان تمام شد حذف کنید.
#![allow(unused_imports, unused_variables, dead_code)]

mod ffi {
    use std::os::raw::{c_char, c_int};
    #[cfg(not(target_os = "macos"))]
    use std::os::raw::{c_long, c_uchar, c_ulong, c_ushort};

    // Opaque type. See https://doc.rust-lang.org/nomicon/ffi.html.
    #[repr(C)]
    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}.
    #[cfg(not(target_os = "macos"))]
    #[repr(C)]
    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).
    #[cfg(all(target_os = "macos"))]
    #[repr(C)]
    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

                        [[[["cfg(not(all(target_os = "macos", target_arch = "x86_64]#
                                    ;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 //
                         [[["cfg(all(target_os = "macos", target_arch = "x86_64]#
                                           ["link_name = "readdir$INODE64]#
                                    ;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

                                                                [(derive(Debug]#
                                                      } 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 راجل

```
                                    } mod ffi
                ;{use std::os::raw::{c_char, c_int
                  [(("cfg(not(target_os = "macos]#
        ;{use std::os::raw::{c_long, c_uchar, c_ulong, c_ushort

    .Opaque type. See https://doc.rust-lang.org/nomicon/ffi.html //
                                        [(repr(C]#
                                    } 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/ //
                                    [(("cfg(not(target_os = "macos]#
                                        [(repr(C]#
                                    } 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 //
                                    [(("cfg(all(target_os = "macos]#
                                        [(repr(C]#
                                    } 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

          [((("cfg(not(all(target_os = "macos", target_arch = "x86_64]#
                            ;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 //
                [(("cfg(all(target_os = "macos", target_arch = "x86_64]#
                                        ["link_name = "readdir$INODE64]#
                            ;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

                                                                [(derive(Debug]#
                                                    } 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} :ربتعمان ریسم")!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 ,"درک زاب را {?:} ناوتیمن")!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 ,"ددنبب را {?:} وانتیمن")!panic
                                                        {
                                                            {
                                                                {
                                                                    {

                                        } <fn main() -> Result<(), String
                                    ;?(".")let iter = DirectoryIterator::new
                                ;(()<<_>println!("files: {:#?}", iter.collect::<Vec
                                                        (())Ok
                                                                {

                                                    [(cfg(test]#
                                                     } mod tests
                                                ;*::use super
                                            ;use std::error::Error

                                                    [test]#
                                        } ()fn test_nonexisting_directory
        ;("وریتکرید نی ایه باشم هن")let iter = DirectoryIterator::new
                                            ;(()assert!(iter.is_err
                                                                {

                                                    [test]#
                  } <<fn test_empty_directory() -> Result<(), Box<dyn Error
                                    ;?()let tmp = tempfile::TempDir::new
                                      )let iter = DirectoryIterator::new
      ,?("رییس در UTF-8 رغی هسیون")tmp.path().to_str().ok_or
                                                    ;?(
                            ;()<<_>let mut entries = iter.collect::<Vec
                                                ;()entries.sort
                                    ;(["..", "."]& ,assert_eq!(entries
                                                        (())Ok
```

```
                                                                {

                                                              [test]#
            } <<fn test_nonempty_directory() -> Result<(), Box<dyn Error
                                        ;?()let tmp = tempfile::TempDir::new
;?("n\ است Foo خاطرات نیا" ,("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
          ,?("ریسم در UTF-8 رغی هسیونن")tmp.path().to_str().ok_or
                                                                    ;?(
                                    ;()<<_>let mut entries = iter.collect::<Vec
                                                        ;()entries.sort
  ;(["assert_eq!(entries, &[".", "..", "bar.png", "crab.rs", "foo.txt
                                                                ((())Ok
                                                                {
                                                                    {
```

بخش IX

دیوردان

فصل 31

به Rust در Android خوش‌آمدید

Rust برای system software در اندروید پشتیبانی می‌شود. این بدان معناست که می توانید سرویس‌ها، کتابخانه‌ها، درایورها یا حتی سیستم‌عامل جدید را در Rust بنویسید (یا در صورت نیاز کده‌ای موجود را به‌بود ببخشید).

> ما امروز سعی خواهیم کرد Rust را از یکی از پروژه‌های خودتان فراخوانی کنیم. بنابراین سعی کنید گوشه کوچکی از پایه کد خود را پیدا کنید تا بتوانیم برخی از خطوط کد را به Rust منتقل کنیم. هر چه وابستگی‌ها و انواع "exotic" کمتر باشد برای ما بهتر است. چیزی که برخی از بای‌ته‌های خام را تجزیه کند ایده آل خواهد بود.

باتوجه‌به افزایش استفاده از Rust در اندروید، سخنران ممکن است به یکی از موارد زیر اشاره کنند:

- مثال سرویس: DNS over HTTP
- کتابخانه‌ها: [Rutabaga Virtual Graphics Interface](https://crosvm.dev/book/appendix/rutabaga_gfx.html)
- Kernel Drivers: Binder
- Firmware: pKVM firmware

فصل 32

میظنت

ما از یک دستگاه Cuttlefish Android Virtual برای آزمایش کد خود استفاده خواهیم کرد. مطمئن شوید که به یکی از آن‌ها دسترسی دارید یا یک مورد جدید ایجاد کنید:

```
source build/envsetup.sh
lunch aosp_cf_x86_64_phone-trunk_staging-userdebug
acloud create
```

لطفاً برای جزئیات به [Android Developer Codelab](https://source.android.com/docs/setup/start) مراجعه کنید.

نکات کلیدی:

- Cuttlefish یک Android device مرجع است که برای کار بر روی دسکتاپ های لینوکس عمومی طراحی شده است. پشتیبانی از MacOS نیز برنامه‌ریزی شده است.
- این Cuttlefish system image تعهد بالایی به دستگاه‌های واقعی دارد و شبیه‌ساز ایده آل برای اجرای بسیاری از موارد استفاده از Rust است.

فصل 33

قوانین ساخت

Android build system (Soong) از Rust از طریق تعدادی ماژول پشتیبانی می‌کند:

Module Type	توضیحات
`rust_binary`	یک Rust binary تولید می‌کند.
`rust_library`	یک کتابخانه Rust تولید می‌کند و هر دو نوع `rlib` و `dylib` را ارائه می‌دهد.
`rust_ffi`	یک کتابخانه Rust C قابل استفاده توسط ماژول‌های `cc` تولید می‌کند و انواع متغیره‌ای `static` و `share` را ارائه می‌کند.
`rust_proc_macro`	یک کتابخانه `proc-macro` تولید می‌کند. این‌ها مشابه پلاگین‌های کامپایلر هستند.
`rust_test`	یک بایناری تست Rust تولید می‌کند که از استاندارد Rust test مهار شده استفاده می‌کند.
`rust_fuzz`	یک `libfuzzer` بایناری Rust fuzz تولید می‌کند.
`rust_protobuf`	یک source تولید می‌کند و یک کتابخانه Rust تولید می‌کند که یک interface برای یک protobuf خاص فراهم می‌کند.
`rust_bindgen`	یک source تولید می‌کند و یک کتابخانه Rust حاوی پیوندهای Rust به کتابخانه‌های C تولید می‌کند.

در ادامه به `rust_binary` و `rust_binary` نگاه خواهیم کرد.

موارد دیگری که سخنران ممکن است ذکر کند:

- Cargo برای repoهای چندزبانه به‌نهسازی نشده است و همچنین packageها را از اینترنت دانلود می‌کند.
- برای انطباق و کارایی، اندرویدبایدcrates in-tree داشته باشد. همچنین بایدبا کد C/C++/Java همکاری داشته باشد. Soong این شکاف را پر می کند.

- 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.

33.1 Rust Binaries

اجازه دهید با یک برنامه ساده شروع کنیم. در ریشه یک AOSP، فایل های زیر را ایجاد کنید:

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() {
    println!("Rust از سلام!");
}
```

اکنون می‌توانید باینری را بسازید، push و اجرا کنید:

```
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!
```

33.2 کتابخانه‌های Rust

شما از `rust_library` برای ایجاد یک کتابخانه Rust جدید برای Android استفاده می‌کنید.

در اینجا ما یک وابستگی به دو کتابخانه را اعلام می‌کنیم:

- `libgreeting`, چیزی که در زیر تعریف می‌کنیم,
- `./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
("!م‌الح‌خوش ار‌ی‌س‌ب ام‌ش ا‌ب ی‌ای‌ن‌آش از ،{name} ام‌س")!format
                        {
```

ب‌ای‌ن‌ری را اج‌را می‌ک‌ن‌ی‌د و push، د‌ی‌ازی‌س می ل‌ب‌ق د‌ن‌ان‌م را:

```
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
```

فصل 34

AIDL

(Android Interface Definition Language (AIDL در Rust پشتیبانی می‌شود:

- کد Rust می‌تواند سرویس‌های AIDL موجود را فراخوانی کند،
- می‌توانید سرویس‌های جدید AIDL را در Rust ایجاد کنید.

34.1 آموزش سرویس Birthday

برای نشان‌دادن نحوه استفاده از Rust با Binder، ما می‌خواهیم روند ایجاد رابط Binder را بررسی کنیم. سپس هم سرویس توصیف‌شده را پیاده‌سازی می‌کنیم و هم کد کلاینتی را می‌نویسیم که با آن سرویس صحبت می‌کند.

AIDL Interfaces 34.1.1

شما API سرویس خود را با استفاده از یک AIDL interface اعلام می‌کنید:

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
,{
,{
{
```

- توجه داشته باشید که ساختار دایرکتوری زیر دایرکتوری `/aidl` باید با نام package استفاده شده در فایل AIDL مطابقت داشته باشد، به‌عنوان‌مثال بسته `com.example.birthdayservice` باید و این فایل در `aidl/com/example/IBirthdayService.aidl` است.

34.1.2 Generated Service API

Binder یک trait مطابق با تعریف interface تولید می‌کند. trait برای صحبت کردن با سرویس است.

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
{
```

سرویس شما باید این trait را پیاده‌سازی کند و کلاینت شما از این ویژگی برای صحبت با سرویس‌ها استفاده خواهد کرد.

- پیوندهای تولید شده را می‌توان در `/<out/soong/.intermediates/<path to module` یافت.
- اشاره کنید که چگونه function signature تولید شده، به ویژه typeهای آرگومان و بازگشتی، با تعریف interface مطابقت دارد.
  - `String` برای آرگومان منجر به type متفاوتی Rust نسبت به `String` به عنوان type برگشتی می‌شود.

34.1.3 پیاده‌سازی سرویس‌ها

اکنون می‌توانیم سرویس AIDL را پیاده‌سازی کنیم:

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"
,[
{
```

- به مسیر ایجاد IBirthdayService trait اشاره کنید و توضیح دهید که چرا هر یک از بخش‌ها ضروری است.
- TODO: trait ویژگی binder::Interface چه کاری انجام می‌دهد؟ آیا متدهایی برای override وجود دارد؟ source کجاست؟

AIDL Server 34.1.4

در نهایت، می‌توانیم سرویسی ایجاد کنیم که سرویس را expose می‌کند:

birthday_service/src/server.rs:

```
.Birthday service !///
;use birthdayservice::BirthdayService
irthdayservice::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
{
```

فرآیند اجرای یک سرویس تعریف‌شده توسط کاربر (در این مورد نوع BirthdayService که IBirthdayService را پیاده‌سازی می‌کند) و شروع آن به‌عنوان یک سرویس Binder چند مرحله دارد و ممکن است پیچیده‌تر از آنچه دانش‌آموزان به آن عادت کرده‌اند به نظر برسد. اگر آن‌ها از Binder برای C++ یا زبان دیگری استفاده کردند. به دانش‌آموزان توضیح دهید که چرا هر مرحله لازم است.

1. نمونه‌ای از نوع سرویس خود (BirthdayService) ایجاد کنید.
2. این service object را در Bn* type مربوطه قرار دهید (در این مورد،BnBirthdayService). این نوع توسط Binder تولید می‌شود و عملکرد رایج Binder را ارائه می‌کند که توسط کلاس پایه BnBinder در C++ ارائه می‌شود. ما در Rust ارث‌بری یا inheritance نداریم، بنابراین در عوض از ترکیب composition می‌کنیم و BirthdayService خود را در BnBinderService تولید شده قرار می‌دهیم.
3. add_service را فراخوانی کنید و به آن یک شناسه سرویس و شیء سرویس خود بدهید (شیء «BnBirthdayService» در مثال).
4. join_thread_pool را فراخوانی کنید تا thread فعلی را به Binder thread اضافه کنید و شروع به گوش دادن برای connectionها کنید.

34.1.5 استقرار

اکنون می‌توانیم سرویس را بسازیم، push کنیم و شروع کنیم:

```
m birthday_server
adb push "$ANDROID_PRODUCT_OUT/system/bin/birthday_server" /data/local/tmp
adb root
adb shell /data/local/tmp/birthday_server
```

در ترمینال دیگر، بررسی کنید که آیا سرویس اجرا شود:

```
adb shell service check birthdayservice
```

```
Service birthdayservice: found
```

همچنین می‌توانید با service call با سرویس تماس بگیرید:

```
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.!.....        ')
```

AIDL Client 34.1.6

درنهایت، ما می‌توانیم یک Rust client برای سرویس جدید خود ایجاد کنیم.

birthday_service/src/client.rs:

```
use com_example_birthdayservice::binder;
use com_example_birthdayservice::aidl::com::example::birthdayservice::IBirthdayService::IBirthdayService;

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
                                                                {
```

توجه داشته باشید که client به libbirthdayservice وابسته نیست.

کل‌این‌ت را در دست‌گاه خود بسازید، push کرده و اجرا کنید:

```
                                                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
```

- Strong<dyn IBirthdayService> یک trait object است که نشان‌دهنده سرویسی است که کل‌این‌ت به آن متصل شده است.
  - Strong یک نوع اشاره‌گر هوشمند سفارشی برای Binder است. هم تعداد ref های درون فرآیندی (in-process) را برای سرویس trait object مدیریت می‌کند و هم شمارنده global Binder را که تعداد فرآیندهای را که به object ارجاع دارند را ردیابی می‌کند.
  - توجه داشته باشید که trait object که کل‌این‌ت برای صحبت با سرویس استفاده می‌کند، دقیقاً از همان ویژگی استفاده می‌کند که سرور پیاده‌سازی می‌کند. برای یک Binder interface معین، یک trait یا ویژگی Rust ایجاد شده است که هم کل‌این‌ت و هم سرور از آن استفاده می‌کنند.
- از همان شناسه سرویس استفاده شده در هنگام ثبت سرویس استفاده کنید. این به طور ایده‌آل باید در یک crate مشترک تعریف شود که هم کل‌این‌ت و هم سرور می توانند به آن وابسته باشند.

34.1.7 تغییر دادن API

اجازه دهید API را با عملکرد بیشتری گسترش دهیم: میخواهیم به مشتریان اجازه دهیم لیستی از خطوط را برای کارت تولد مشخص کنند:

```
;package com.example.birthdayservice

/* .Birthday service interface **/
} interface IBirthdayService
/* .Generate a Happy Birthday message **/
;(String wishHappyBirthday(String name, int years, in String[] text
{
```

این منجر به یک تغییر ویژگی به روز شده برای IBirthdayService می‌شود:

```
} trait IBirthdayService
)fn wishHappyBirthday
,self&
,name: &str
,years: i32
,[text: &[String
;<binder::Result<String <- (
{
```

- توجه داشته باشید که چگونه String[] در تعریف AIDL به عنوان &[String] در Rust ترجمه می‌شود، به‌عنوان مثال از idiomatic Rust type در bindingهای تولید شده تا جایی که ممکن است استفاده می‌شود:
  - آرگومان های آرایه in به sliceها ترجمه می‌شوند.
  - آرگومان‌های out و inout به &mut Vec<T> ترجمه می‌شوند.
  - مقادیر بازگشتی به بازگرداندن Vec<T> ترجمه می‌شوند.

34.1.8 به‌روزرسانی کلاینت و سرویس‌ها

کد سرویس کلاینت و سرور را برای حساب کردن API جدید به‌روز کنید.

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 کار با انواع AIDL

انواع AIDL به نوع متناظر Rust ترجمه می‌شوند:

- انواع اولیه یا Primitive types (بیت‌ها) به idiomatic Rust type نگاشت می‌شوند.
- انواع Collection مانند sliceها، Vec و string type پشتیبانی می‌شوند.
- ارجاع به AIDL objects و دسته‌های فایل را می‌توان بین clientها و سرویس‌ها ارسال کرد.
- دسته‌های فایل و بسته‌بندی‌ها به طور کامل پشتیبانی می‌شوند.

34.2.1 انواع اولیه

انواع ابتدایی یا Primitive type (معمولاً) به صورت idiomatically نگاشت می‌شوند:

AIDL Type	Rust Type	نکته
boolean بولین‌ها	bool	
byte	i8	توجه داشته باشید که بایت‌ها امضا شده‌اند.
char	u16	به استفاده از u16 توجه کنید، نه u32.
int	i32	
long	i64	
float	f32	
double	f64	
String	String	

34.2.2 تایپ‌های آرایه‌ای

انواع آرایه (byte[], T[] و List<T>) بسته به نحوه استفاده از آن‌ها در function signature، به Rust array type مناسب ترجمه می‌شوند:

موقعیت	Rust Type
in argument	`&[T]`
out/inout argument	`&mut Vec<T>`
Return	`Vec<T>`

- در اندروید ۱۳ یا بالاتر، آرایه‌های با اندازه ثابت پشت‌یبانی می‌شوند، یعنی `[T; N]` به `T[N]`. آرایه‌های با اندازه ثابت می‌توانند چندین بعد داشته باشند (مانند `int[3][4]`. در Java backend، آرایه‌های با اندازه ثابت به عنوان array type نمایش داده می‌شوند.
- آرایه‌های موجود در فیلدهای parcelable همیشه به `Vec<T>` ترجمه می‌شوند.

Sending Objects 34.2.3

AIDL objects را می‌توان به‌عنوان یک نوع AIDL مشخص به یک interface IBinder پاک‌شده ارسال کرد:

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 {
    String wishWithProvider(IBirthdayInfoProvider provider);
    /* .The same thing, but using a binder object **/

    String wishWithErasedProvider(IBinder provider);
    /* .`The same thing, but using `IBinder **/
}
```

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
                                                                    {
```

- به اس‌تفاده از BnBirthdayInfoProvider توجه کنی‌د. این همان هدف BnBirthdayService است که قبلاً دی‌دیم.

34.2.4 ب‌س‌ته‌بن‌دی‌ها

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
                                                                    {
```

34.2.5 ارسال فایل‌ها

فایل‌ها را می‌توان با استفاده از نوع ParcelFileDescriptor بین کلاینت‌ها/سرورهای Binder ارسال کرد:

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.
                                ;("ربتعمان لیاف لرتنک")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` یک `OwnedFd` را احاطه می‌کند و بنابراین می‌تواند از یک `File` (یا هر نوع دیگری که یک `OwnedFd` را احاطه می‌کند) ایجاد کند و می‌تواند برای ایجاد یک دسته `File` جدید در طرف دیگر استفاده شود.
- انواع دیگر توصیف‌گره‌ای فایل را می توان بسته‌بندی و ارسال کرد، به‌عنوان‌مثال. سوکت های UDP، TCP و UNIX.

فصل 35

تست‌کردن در Android

بر اساس Testing، اکنون به نحوه عمل‌کرد unit testها در AOSP خواهیم پرداخت. از ماژول rust_test برای تست‌های واحد خود استفاده کنید:

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
    {

[(cfg(test]#
    } mod tests
;*::use super

    [test]#
} ()fn short_string
;("assert_eq!(leftpad("foo", 5), "  foo
    {
```

```
[test]#
} ()fn long_string
;("assert_eq!(leftpad("foobar", 6), "foobar
{
{
```

اکنون می توانید تست را با

```
atest --host libleftpad_test
```

خروجی به شکل زیر است:

```
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
```

توجه کنید که چگونه فقط ریشه crate کتابخانه را ذکر می‌کنید. تست‌ها به صورت بازگشتی در ماژول‌های تودرتو یافت می‌شوند.

GoogleTest 35.1

چعبه GoogleTest با استفاده از *matchers* اجازه می‌دهد تا assertهای آزمایشی انعطاف‌پذیر را انجام دهید:

```
;*::use googletest::prelude

[googletest::test]#
} ()fn test_elements_are
;["let value = vec!["foo", "bar", "baz
;((("expect_that!(value, elements_are!(eq(&"foo"), lt(&"xyz"), starts_with("a
{
```

اگر آخرین عنصر را به "!"تغییر دهیم، آزمایش با یک پیغام خطای ساختار یافته که خطا را pin-pointing می‌کند، شکست می خورد:

```
---- 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
```

.This slide should take about 5 minutes

- بخشی از GoogleTest نیست، بنابراین باید این مثال را در یک محیط local اجرا کنید. برای افزودن سریع آن به پروژه Cargo موجود، از `cargo add googletest` استفاده

کنید.

- خط `use googletest::prelude::*;` تعدادی از ماکروها و typeهای پرکاربرد را وارد می‌کند.
- 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.
- یک ویژگی خاص خوب این است که عدم تطابق در stringهای چند خطی به صورت یک تفاوت نشان داده می شود:

```
#[test]
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)
    );
}
```

تفاوت رنگی را نشان می‌دهد (رنگ ها در اینجا نشان داده نمی‌شوند):

```
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
```

- این crate یک پورت [GoogleTest for C++](https://google.github.io/googletest/) در Rust است.

Mocking 35.2

برای mocking از Mockall که یک کتابخانه محبوب بوده استفاده شده است. برای استفاده از traitها، باید کد خود را مجدداً تغییر دهید، سپس می‌توانید به سرعت آن‌ها را mock کنید:

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

#[mockall::automock]
pub trait Pet {
    fn is_hungry(&self, since_last_meal: Duration) -> bool;
}

#[test]
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);
}
```

.This slide should take about 5 minutes

- Mockall کتابخانه mocking توصیه شده در Android (AOSP) است. کتابخانه‌های mocking دیگری در crates.io در دسترس هستند، به‌ویژه در زمینه سرویس‌های HTTP mocking. سایر کتابخانه‌های mocking به روشی مشابه Mockall کار می‌کنند، به این معنی که اجرای ساختگی یا mock یک ویژگی خاص را آسان می‌کنند.
- توجه داشته باشید که mocking تا حدودی جنجال‌برانگیز است: mockها به شما این امکان را می‌دهند که آزمون را کاملاً از وابستگی‌های آن جدا کنید. نتیجه فوری آن، اجرای سریع‌تر و پایدارتر تست است. از طرف دیگر، mockها را می توان به اشتباه پیکربندی کرد و خروجی متفاوتی با آنچه وابستگی های واقعی انجام می دادند را برگرداند.

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

به‌طور مشابه، بسیاری از frameworkهای وب به شما اجازه می‌دهند یک سرور در یک process دیگر راه‌اندازی کنید که به یک پورت تصادفی در `localhost` متصل می‌شود. همین ایده را به mock کردن frameworkترجیح دهید زیرا به شما کمک می‌کند کد خود را در محیط واقعی آزمایش کنید.

- Mockall بخشی از Rust Playground نیست، بنابراین باید این مثال را در یک محیط local اجرا کنید. از `cargo add mockall` برای اضافه کردن سریع Mockall به پروژه Cargo موجود استفاده کنید.
- Mockall عملکرد بسیار بیشتری دارد. به ویژه، می‌توانید انتظاراتی را تنظیم کنید که به استدلال های ارائه شده بستگی دارد. در اینجا ما از این برای mock کردن عملکرد cat استفاده می‌کنیم که 3 ساعت پس از آخرین باری که به آن غذا داده شده گرسنه می‌شود:

```
#[test]
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);
}
```

- می‌توانید از `.times(n)` برای محدودکردن تعداد دفعاتی که یک mock method می‌تواند به n فراخوانی شود استفاده کنید --- در صورت عدم ارضای این روش، زمانی که آن را حذف کنید به‌طور خودکار دچار panic می‌شود.

فصل 36

لاگ

باید از log crate برای ورود خودکار به logcat (روی دستگاه) یا stdout (روی host) استفاده کنید:

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),
    );
    error!("Something went wrong!");
    info!("خوب پیش می‌رود.");
    debug!("عروش به‌نرمی.");
}
```

ساخت، push و اجرای باینری‌ها روی یک ماشین:

```
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
```

لاگ‌ها در adb logcat نشان‌داده می‌شوند:

```
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
```

فصل 37

قابلیت همکاری

Rust از قابلیت همکاری با زبان‌های دیگر پشتیبانی می‌کند. این بدان معنی است که شما می‌توانید:

- توابع Rust را از زبان های دیگر فراخوانی کنید.
- فراخوانی توابع نوشته شده به زبان های دیگر از Rust.

وقتی توابعی را به یک زبان خارجی فراخوانی می‌کنید، می‌گوییم که از یک رابط تابع خارج(*foreign function interface*) که به نام FFI نیز شناخته می‌شود، استفاده می‌کنید.

37.1 قابلیت همکاری با C

Rust پشتیبانی کاملی برای link دادن object fileهایی با یک فراخوانی C دارد. به طور مشابه، می توانید توابع Rust را export کرده و آنها را از C فراخوانی کنید.

در صورت تمایل می‌توانید این کار را دستی انجام دهید:

```rust
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}");
}
```

ما قبلاً این را در تمرین Safe FFI Wrapper دیدیم.

این مستلزم آگاهی کامل از پلت‌فرم هدف است و برای production توصیه نمی‌شود.

در ادامه گزینه‌های بهتر را بررسی خواهیم کرد.

37.1.1 با استفاده از Bindgen

ابزار bindgen می‌تواند اتصالات را از یک فایل هدر C به طور خودکار ایجاد کند.

ابتدا یک کتابخانه کوچک C ایجاد کنید:

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
                                {
```

ای‌ن را ب‌ه فای‌ل Android.bp خود اض‌اف‌ه ک‌ن‌ی‌د:

interoperability/bindgen/Android.bp:

```
} cc_library
  ,"name: "libbirthday
 ,["srcs: ["libbirthday.c
                        {
```

ی‌ک فای‌ل ه‌در wrapper ب‌رای ک‌ت‌اب‌خ‌ان‌ه ای‌ج‌اد ک‌ن‌ی‌د (در ای‌ن م‌ث‌ال ب‌ه ش‌دت م‌ورد ن‌ی‌از ن‌ی‌س‌ت):

interoperability/bindgen/libbirthday_wrapper.h:

```
"include "libbirthday.h#
```

اک‌ن‌ون م‌ی ت‌وان‌ی‌د ات‌ص‌ال‌ات (bindings) را ب‌ه ط‌ور خ‌ودک‌ار ای‌ج‌اد ک‌ن‌ی‌د:

interoperability/bindgen/Android.bp:

```
} rust_bindgen
  ,"name: "libbirthday_bindgen
  ,"crate_name: "birthday_bindgen
,"wrapper_src: "libbirthday_wrapper.h
  ,"source_stem: "bindings
 ,["static_libs: ["libbirthday
                        {
```

در ن‌ه‌ای‌ت، م‌ی‌ت‌وان‌ی‌م از bindingه‌ا در ب‌رن‌ام‌ه Rust خ‌ود اس‌ت‌ف‌اد‌ه ک‌ن‌ی‌م:

interoperability/bindgen/Android.bp:

```
} rust_binary
  ,"چاپ_ک‌ارت_ت‌ول‌د" :name
    ,["srcs: ["main.rs
,["rustlibs: ["libbirthday_bindgen
                        {
```

interoperability/bindgen/main.rs:

```
.Bindgen demo !//

;{use birthday_bindgen::{card, print_card

} ()fn main
;()unwrap.("رتپی")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
                                                        {
                                                            {
```

س اخت، push و اجرای باینری ها روی یک ماشین:

```
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
```

در نهایت، ما می توانیم تست های تولید شده خودکار را برای اطمینان از کارکرد اتصالات (bindings) اجرا کنیم:

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 فراخوانی Rust

صدور یا Exporting توابع و تایپ ها از Rust به C آسان است:

interoperability/rust/libanalyze/analyze.rs

```
.Rust FFI demo !//
[(deny(improper_ctypes_definitions]!#

;use std::os::raw::c_int

.Analyze the numbers ///
                                    [no_mangle]#
} (pub extern "C" fn analyze_numbers(x: c_int, y: c_int
                                        } if x < y
;("!println!("x ({x}) is smallest
```

```
                                                    } else {
        ;("است ({x ({x از رتگرزب المتحا ({y  ({yرادقم ")!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
                                                        {
```

اکنون می توانیم این را از یک باینری C فراخوانی کنیم:

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
                                                        {
```

ساخت، push و اجرای باینری‌ها روی یک ماشین:

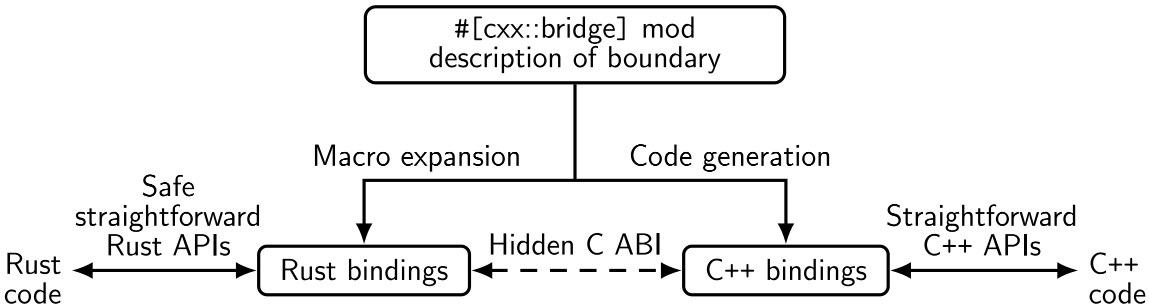
```
                                                        m analyze_numbers
adb push "$ANDROID_PRODUCT_OUT/system/bin/analyze_numbers" /data/local/tmp
                                                adb shell /data/local/tmp/analyze_numbers
```

نام‌گذاری عمومی Rust را غیرفعال می‌کند، بنابراین نماد صادر شده فقط عبات نام خواهد بود. همچنین می‌توانید از #[no_mangle] برای تعیین هر نامی استفاده کنید.

37.2 با ++C

این CXX crate امکان همکاری امن بین Rust و ++C را فراهم می‌کند.

رویکرد کلی به این صورت است:



37.2.1 ماژول پل

CXX متکی به توصیفی از signatureهای تابعات است که از هر زبان به زبان دیگر در معرض دید قرار می‌گیرد. شما این توضیحات را با استفاده از بلوک‌های خارجی در یک ماژول Rust ارائه می‌کنید که با attribute ماکروهای `#[cxx::bridge]` شرح داده شده است.

```rust
#[allow(unsafe_op_in_unsafe_fn)]
#[cxx::bridge(namespace = "org::blobstore")]
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;
    }
}
```

- پل به طور کلی در یک ماژول `ffi` در crate شما اعلام می‌شود.

- از اعلان‌های (declarations) انجام شده در ماژول پل، CXX تعاریف مطابق با type/function در Rust و C++ را ایجاد می‌کند تا آن موارد را در معرض هر دو زبان قرار دهد.
- برای مشاهده کد Rust ایجاد شده از cargo-expand که برای مشاهده ماکرو proc توسعه یافته استفاده کنید. برای بیشتر نمونه‌ها از `cargo expand ::ffi` فقط برای گسترش ماژول `ffi` استفاده کنید (اگرچه این برای پروژه‌های Android کاربرد ندارد).
- برای مشاهده کد C++ تولید شده به `target/cxxbridge` نگاه کنید.

37.2.2 تعریف پل در Rust

```
#[cxx::bridge]
    } 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
                                {
```

- موارد اعلام شده در موارد `extern "Rust"` reference که در محدوده ماژول والد قرار دارند.
- تولیدکننده کد CXX از `extern "Rust"` خارجی شما برای تولید یک فایل هدر C++ حاوی اعلان‌های C++ مربوطه استفاده می‌کند. header تولیدشده همان مسیری را دارد که فایل منبع Rust حاوی پل دارای آن است، به جز استفاده از پسوند فایل rs.h.

37.2.3 Generated C++

```
#[cxx::bridge]
    } mod ffi
.++Rust types and signatures exposed to C //
    } "extern "Rust
    ;type MultiBuf

;[fn next_chunk(buf: &mut MultiBuf) -> &[u8
                                {
                                    {
```

نتیجه (تقریبی) C++ در زیر است:

```
} 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

```
                                                                [cxx::bridge]#
                                                                     } 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
                                                                                          {
                                                                                            {
```

ن‌ت‌یجه (ت‌ب‌ی‌ق‌ی) Rust در زی‌ر اس‌ت:

```
                                                                        [(repr(C]#
                                                                } pub struct BlobstoreClient
                                                        ,private: ::cxx::private::Opaque_
                                                                                          {

                                    } <pub fn new_blobstore_client() -> ::cxx::UniquePtr<BlobstoreClient
                                                                      } "extern "C
                                  ["link_name = "org$blobstore$cxxbridge1$new_blobstore_client]#
                                             ;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
                                ["link_name = "org$blobstore$cxxbridge1$BlobstoreClient$put]#
                                                                   )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__
                                                                {
                                                                    {
                                                                        {

                                                                    ... //
```

- برنامه‌نویسی زبان به تضمینی در مورد درست بودن signatureهایی که تایپ کرده است ندارد. CXX اظهارات ثابتی را انجام می دهد که signatureها دقیقاً با آنچه در C++ اعلام شده مطابقت دارند.
- بلوک‌های unsafe extern به شما امکان می‌دهند توابع C++ را که برای فراخوانی از Rust امن هستند را اعلام کنید.

37.2.5 انواع مشترک

```
                                                                [cxx::bridge]#
                                                                    } mod ffi
                                                            [(derive(Clone, Debug, Hash]#
                                                                    } struct PlayingCard
                                                                    ,suit: Suit
                                                        value: u8,  // A=1, J=11, Q=12, K=13
                                                                    {

                                                                    } enum Suit
                                                                     ,Clubs
                                                                   ,Diamonds
                                                                     ,Hearts
                                                                     ,Spades
                                                                    {
                                                                        {
```

- فقط C-like (unit) enums پشتیبانی می‌شود.
- تعداد محدودی از ویژگی‌ها برای #[derive()]در انواع مشترک پشتیبانی می‌شوند. عملکرد مربوطه نیز برای کد C++ ایجاد می شود، به عنوان مثال. اگر Hash را استخراج کنید، پیاده‌سازی std::hash برای نوع C++ مربوطه نیز ایجاد می‌کند.

37.2.6 Shared Enums

```
                                                                [cxx::bridge]#
                                                                    } mod ffi
                                                                } enum Suit
                                                                 ,Clubs
                                                               ,Diamonds
                                                                 ,Hearts
                                                                 ,Spades
                                                                {
                                                                    {
```

:Generated Rust

```
                                                        [(derive(Copy, Clone, PartialEq, Eq]#
                                                                [(repr(transparent]#
```

```
} pub struct Suit
,pub repr: u8
{

[(allow(non_upper_case_globals]#
} 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
;{
```

- در سمت Rust، کد تولیدشده برای enums مشترک در واقع ساختاری است که یک مقدار عددی را بسته‌بندی می‌کند. به این دلیل که UB در C++ نیست تا یک کلاس enum مقداری متفاوت از همه انواع فهرست شده داشته باشد و نمایش‌دهنده Rust مورد نظر ما باید رفتار مشابه‌ی داشته باشد.

37.2.7 مدیریت خطا Rust

```
[cxx::bridge]#
} mod ffi
} "extern "Rust
;<fn fallible(depth: usize) -> Result<String
{
{

} <fn fallible(depth: usize) -> anyhow::Result<String
} if depth == 0
;(("دارد وجود 0 < عمق به return Err(anyhow::Error::msg("fallible1
{

()Ok("Success!".into
{
```

- توابع Rust که «نتیجه» را برمی‌گردانند به exceptionهای سمت C++ ترجمه می‌شوند.
- این exception به وقوع پیوسته همیشه از نوع rust::Error خواهد بود که در درجه اول راهی برای دریافت string پیام خطا نشان می‌دهد. پیغام خطا از نوع خطای Display می‌آید.
- باز شدن panic از Rust به C++ همیشه باعث می‌شود که فرآیند بلافاصله خاتمه یابد.

37.2.8 مدیریت خطا C++

```
[cxx::bridge]#
} 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
                                                                        {
                                                                            {
```

- توابع C++ اعلام شده (declared) برای برگرداندن `Result`، در صورت گرفته شده در exception در C++ را می‌گیرند و آن را به عنوان مقدار `Err` به توابع فراخوانی Rust برمی‌گردانند.
- اگر یک exception از یک function "C++" extern که توسط پل CXX برای بازگشت "نتیجه" اعلان نشده است، ایجاد شود، برنامه `std::terminate` C++' را فراخوانی می‌کند. این رفتار معادل همان exception است که از طریق یک C++ function `noexcept` فعال می‌شود.

37.2.9 تایپ‌های اضافی

C++ Type	Rust Type
`rust::String`	`String`
`rust::Str`	`str&`
`std::string`	`CxxString`
`rust::Slice`	`[T]/&mut [T]&`
`<rust::Box<T`	`Box<T>`
`<std::unique_ptr<T`	`<UniquePtr<T`
`<rust::Vec<T`	`<Vec<T`
`<std::vector<T`	`<CxxVector<T`

- این typeها را می‌توان در فیلدهای ساختارهای مشترک و آرگومان‌ها و extern functionها استفاده کرد.
- توجه داشته باشید که `String` در Rust مستقیماً به `std::string` نگاشت نمی‌شود. چند دلیل برای این وجود دارد:
  - `std::string` ثابت UTF-8 را که `String` به آن نیاز دارد را پشتیبانی نمی‌کند.
  - این دو نوع طرح‌بندی‌های متفاوتی در حافظه دارند و بنابراین نمی‌توان آن‌ها را مستقیماً بین زبان‌ها منتقل کرد.
  - `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 ساخت در اندروید

یک `cc_library_static` برای ساخت کتابخانه C++ از جمله هدر و فایل منبع تولید شده CXX ایجاد کنید.

```
                                                    } 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
                                                {
```

- به این نکته اشاره کنید که `libcxx_test_bridge_header` و `libcxx_test_bridge_code` وابستگی‌های برای پیوندهای C++ تولید شده توسط CXX هستند. نحوه تنظیم این‌ها را در اسلایدبعدی نشان خواهیم داد.
- توجه داشته باشید که برای ایجاد تعاریف رایج CXX باید به کتابخانه `cxx-bridge-header` وابسته باشید.
- مستندات کامل برای استفاده از CXX در Android را می‌توانید در این آدرس پیدا کنید the Android docs. ممکن است بخواهید آن پیوند را با کلاس به اشتراک بگذارید تا دانش‌آموزان بدانند که در آینده می‌توانند این دستورالعمل‌ها را دوباره پیدا کنند.

37.2.11 ساخت در اندروید

دو نوع ژنر ایجاد کنید: یکی برای تولید هدر CXX و دیگری برای تولید فایل منبع CXX. سپس از این‌ها به عنوان ورودی `cc_library_static` استفاده می‌شود.

```
    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
                                                {
```

- ابزار`cxxbridge` یک ابزار مستقل است که سمت C++ ماژول پل را تولید می‌کند. در Android گنجانده شده و به عنوان ابزار Soong در دسترس است.
- طبق قرارداد، اگر فایل منبع Rust شما `lib.rs` باشد، فایل header شما `lib.rs.h` و فایل منبع شما `lib.rs.cc` نام خواهد داشت. اگرچه این قرارداد نام‌گذاری اجرا نمی‌شود.

37.2.12 ساخت در اندروید

یک`rust_binary` ایجاد کنید که به `libcxx` و `cc_library_static` شما بستگی دارد.

```
                                                } rust_binary
                                  ,"name: "cxx_test
                                 ,["srcs: ["lib.rs
                              ,["rustlibs: ["libcxx
```

```
    ,["static_libs: ["libcxx_test_cpp
                                    {
```

37.3 قابلیت همکاری با جاوا

جاوا می‌تواند objectهای مشترک را از طریق [واسط بومی جاوا (Java Native Interface (JNI باربگیری کند. jni crate به شما امکان می دهد یک کتابخانه سازگار ایجاد کنید.

ابتدا یک تابع Rust برای export به Java ایجاد می‌کنیم:

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 ///
                                [no_mangle]#
)pub extern "system" fn Java_HelloWorld_hello
                                  ,env: JNIEnv
                               ,class: JClass_
                                ,name: JString
                                   } jstring <- (
;()let input: String = env.get_string(name).unwrap().into
                    ;("!{input} ,سلام")!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
                                                    {
```

سپس این تابع را از جاوا فراخوانی می کنیم:

interoperability/java/HelloWorld.java:

```
                                } class HelloWorld
;(private static native String hello(String name

                                                } static
                        ;("System.loadLibrary("hello_jni
                                                      {

                    } (public static void main(String[] args
;("سلی")String output = HelloWorld.hello
                        ;(System.out.println(output
```

```
{
    {
```

interoperability/java/Android.bp:

```
} java_binary
    ,"name": "helloworld_jni
,["HelloWorld.java"این دن مال] :srcs
    ,"main_class": "HelloWorld
,["required": ["libhello_jni
{
```

در نهایت، می‌توانید برنامه را بسازید، همگام‌سازی کنید و اجرا کنید:

```
m helloworld_jni
adb sync  # requires adb root && adb remount
adb shell /system/bin/helloworld_jni
```

فصل 38

ادغام

این یک تمرین گروهی است: ما به یکی از پروژه‌هایی که با آن کار می‌کنید نگاه می‌کنیم و سعی می‌کنیم قدرت Rust را در آن ادغام کنیم. چند پیشنهاد:

- با سرویس AIDL خود با کلاینت که در Rust نوشته شده است تماس بگیرید.
- یک تابع را از پروژه خود به Rust منتقل کنید و آن را فراخوانی کنید.

هیچ راه‌حلی در اینجا ارائه نشده است زیرا این مورد باز است: به فردی در کلاس متکی است که یک قطعه کد دارد که می تواند آن را به Rust on fly تبدیل کند.

X بخش

Chromium

فصل 39

به **Rust** در **Chromium** خوش‌آمدید

Rust برای کتابخانه‌های شخص ثالث در Chromium پشتیبانی می‌شود، با glue code اول شخص برای اتصال بین Rust و کد موجود در Chromium C++.

امروز ما با Rust ارتباط می‌گیریم تا کار احمقانه‌ای با stringها انجام دهد. اگر گوشه‌ای از کد را دارید که در آن رشته UTF8 را به کاربر نشان می دهید، به جای قسمت دقیقی که در مورد آن صحبت می‌کنیم، این دستور العمل را در قسمت خود از پایگاه کد دنبال کنید.

فصل 40

میظنت

مطمئن شوید که می‌توانید Chromium را build و اجرا کنید. هر پلت‌فرم و مجموعه‌ای از build flag ها بدون مشکلات، تا زمانی که کد شما نسبتاً جدید باشد (موقعیت commit 1223636 به عد، مربوط به نوامبر 2023):

```
gn gen out/Debug
autoninja -C out/Debug chrome
out/Debug/chrome # or on Mac, out/Debug/Chromium.app/Contents/MacOS/Chromium
```

(یک component به صورت debug build برای سریعترین زمان تکرار توصیه می‌شود. این یک حالت پیش‌فرض است!)

اگر از قبل در آن محله نیستید، به چگونه Chromium بسازیم مراجعه کنید. هشدار: راه‌اندازی برای build Chromium زمان زیادی می‌برد.

همچنین توصیه می شود که Visual Studio code را نصب کرده باشید.

در مورد تمرین‌ها

این بخش از دوره دارای یک سری تمرین‌ات است که بر روی یکدیگر ساخته می‌شوند. ما آن‌ها را به جای این‌که فقط در انتها انجام دهیم، در طول دوره پخش خواهیم کرد. اگر برای تکمیل قسمت خاصی وقت ندارید، نگران نباشید: می توانید در اسلاید بعدی به عقب برگردید.

فصل 41

مقایسه Chromium و اکوسیستم Cargo

جامعه Rust معمولاً از `cargo` و کتابخانه‌های crates.io استفاده می‌کند. Chromium با استفاده از `gn` و `ninja` و مجموعه‌ای از وابستگی‌ها ساخته شده است.

هنگام نوشتن کد در Rust، انتخاب‌های شما عبارتند از:

- از `gn` و `ninja` با کمک الگوهای `build/rust/*.gni` استفاده کنید (مثلاً `rust_static_library` که بعداً با آن آشنا خواهیم شد). این از toolchain و crateهای بررسی‌شده Chromium استفاده می‌کند.
- از `cargo` استفاده کنید، اما خود را به toolchain و crateهای بررسی‌شده Chromium محدود کنید
- از `cargo` استفاده کنید، به یک toolchain و/یا [crateهای دانلود شده از اینترنت](https://crates.io/) اعتماد کنید.

از اینجا به بعد ما روی `gn` و `ninja` تمرکز خواهیم کرد، زیرا به این ترتیب می‌توان کد Rust را در مرورگر Chromium ایجاد کرد. در عین حال، بخش همه‌ی از اکوسیستم Cargo است و شما باید آن را در جعبه ابزار خود نگه دارید.

خرده تمرین

به گروه‌های کوچک تقسیم شده و:

- سناریوهای طوفان فکری که در آن `cargo` ممکن است مزیتی را ارائه دهد و نمایه ریسک‌های این سناریوها را ارزیابی کنند.
- در هنگام استفاده از `gn` و `ninja`و `cargo` آفلاین و غیره در مورد ابزارها، کتابخانه‌ها و گروه‌های از افراد بحث کنید.

از دانش‌آموزان بخواهید که قبل از اتمام تمرین از نگاه کردن به یادداشت‌های سخنران خودداری کنند. با فرض این‌که افرادی که دوره را می‌گذرانند از نظر فیزیکی با هم هستند، از آن‌ها بخواهید در گروه های کوچک ۳-۴ نفره بحث کنند.

نکته‌ها/تکنیک‌های مربوط به بخش اول تمرین (”سناریوهایی که Cargo ممکن است مزیتی را ارائه دهد”):

- این فوق العاده است که هنگام نوشتن یک ابزار یا نمونه‌سازی بخشی از Chromium به اکوسیستم غنی کتابخانه‌های crates.io دسترسی داشته باشید. تقریباً برای هر چیزی یک crate وجود دارد و معمولاً استفاده از آن‌ها بسیار لذت بخش است. (`clap` برای تجزیه خط

فرمان، `serde` برای سریال‌سازی/جداسازی‌سریال به/از قالب‌های مختلف، `itertools` برای کار با تکرارکننده‌ها (iterators) و غیره).

- `cargo` بررسی کردن یک کتابخانه را آسان می‌کند (فقط یک خط به `Cargo.toml` اضافه کنید و شروع به نوشتن کد کنید)
- شاید ارزش این را داشته باشد که چگونه CPAN به انتخاب `perl` کمک کرد. یا مقایسه با `pip` + `python`.

- 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` افزودن ابزارها را از طریق تست‌های می‌کند.
- ممکن است ارزش مقایسه با برنامه‌های افزودنی کروم یا افزونه‌های VScode را داشته باشد.

- نمونه‌های کلی و عمومی از پروژه‌هایی که `cargo` ممکن است انتخاب مناسبی باشد:

- شاید تعجب‌آور باشد که Rust به طور فزاینده ای در صنعت برای نوشتن ابزارهای خط فرمان محبوب می‌شود. گستردگی و ارگونومی کتابخانه‌ها با پایتون قابل مقایسه است، درحالی‌که قوی‌تر (به لطف تایپ سیستم غنی) است و سریع‌تر کار می‌کند (به عنوان یک زبان کامپایل شده و نه مفسری).
- مشارکت در اکوسیستم Rust مستلزم استفاده از ابزار استاندارد Rust مانند Cargo است. کتابخانه‌هایی که می‌خواهند مشارکته‌ای خارجی دریافت کنند و می‌خواهند خارج از Chromium استفاده شوند (مثلاً در محیط‌های ساخت Bazel یا Android/Soong) احتمالاً باید از Cargo استفاده کنند.

- نمونه‌هایی از پروژه‌های مرتبط با Chromium که مبتنی بر `cargo` هستند:

- `serde_json_lenient` (در قسمت‌های دیگر Google آزمایش شده که منجر به PRهایی با بهبود در عملکرد می‌باشد.)
- کتابخانه‌های فونت مانند `font-types`
- ابزار `gnrt` (ما بعداً در دوره با آن آشنا خواهیم شد) که برای تجزیه خط فرمان به `clap` و برای فایل‌های پیکربندی به `toml` بستگی دارد.
  * 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` از کپی Chromium از `cargo` و `rustc` استفاده می‌کند.`gnrt` به کتابخانه‌های شخص ثالثی بستگی دارد که از اینترنت دانلود شده‌اند، اما `run_gnrt.py` از `cargo` می‌پرسد که فقط محتوای `--locked` از طریق `Cargo.lock` مجاز است.)

دانش‌آموزان ممکن است موارد زیر را به طور ضمنی یا صریح مورد اعتماد تشخیص‌دهند:

- `rustc` (کامپایلر Rust) که به نوبه خود به کتابخانه‌های LLVM، کامپایلر Clang، منابع `rustc` (برگرفته از GitHub، بررسی شده توسط تیم کامپایلر Rust) وابسته است، کامپایلر Rust بازنویسی که برای راه‌اندازی بارگیری (bootstrapping) شده است
- `rustup` (شاید شنیده باشید که `rustup` زیر چتر سازمان https://github.com/rust-lang/ - همانند `rustc` توسعه یافته است)
- `cargo`, `rustfmt`،سایر موارد.
- زیرساخت‌های داخلی مختلف (ربات‌هایی که `rustc` می‌سازند، سیستمی برای توزیع toolchain از پیش‌ساخته شده بین همدستان Chromium و بقیه)
- ابزار Cargo مانند `cargo vet`، `cargo audit` و غیره.

- کتابخانه‌های Rust در `//third_party/rust` عرضه شده است (بازرسی شده توسط security@chromium.org)
- سایر کتابخانه‌های Rust (بعضی خاص، برخی کاملاً محبوب و پرکاربرد)

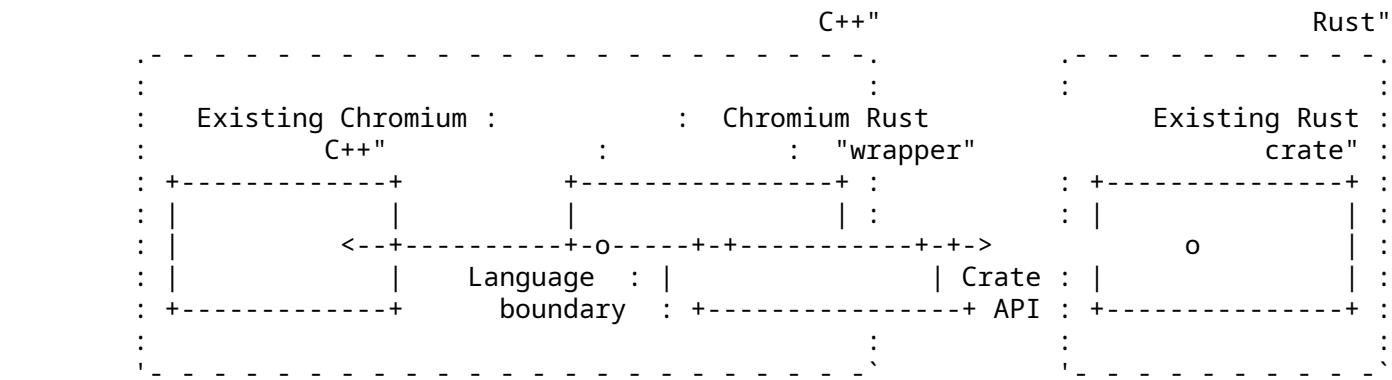
فصل 42

رویکرد Chromium Rust

Chromium هنوز Rust شخص اول را مجاز نمی‌کند، مگر در موارد نادر که توسط Area Tech Leads تأیید شده است.

رویکرده‌ای Chromium در مورد کتابخانه‌های شخص ثالث اینجا مشخص شده است. Rust برای کتابخانه‌های شخص ثالث تحت شرایط مختلف مجاز است، از جمله اینکه آیا آن‌ها به‌ترین گزینه برای کارایی بالا یا موارد امنیتی هستند.

تعداد بسیار کمی از کتابخانه‌های Rust مستقیماً یک C/C++ API را در معرض دید (expose) قرار می‌دهند، به این معنی که تقریباً همه این کتابخانه‌ها به مقدار کمی glue code اول شخص نیاز دارند.



کد Rust glue اول شخص برای یک crate شخص ثالث خاص معمولاً با‌ید در `ty/rust/<crate>/<version>/wrapper` نگه‌داری شود.

به‌همین دلیل، دوره‌ی امروز به شدت بر روی موارد زیر متمرکز خواهد شد:

- آوردن کتابخانه های Rust شخص ثالث ("crates")
- نوشتن glue code برای اینکه بتوانید به کمک آن crateها از Chromium C++ استفاده کنید.

اگر این رویکرد در طول زمان تغییر کند، این دوره به گونه‌ای تکامل می‌یابد که در مسیر مناسب ادامه یابد.

فصل 43

قوانین Build

کد Rust معمولاً با استفاده از `cargo` ساخته می‌شود. Chromium با `gn` و `ninja` جهت کارایی بیشتر ساخته می‌شود --- قواعد استاتیک آن حداکثر موازی‌سازی را امکان‌پذیر می‌سازد. Rust نیز از این قاعده مستثنی نیست.

افزودن کد **Rust** به **Chromium**

در برخی از فایل‌های موجود `BUILD.gn` یک `rust_static_library` را اعلام کنید:

```
("import("//build/rust/rust_static_library.gni

} ("rust_static_library("my_rust_lib
    "crate_root = "lib.rs
  [ "sources = [ "lib.rs
                                {
```

همچنین می‌توانید `deps` را روی سایر اهداف Rust اضافه کنید. بعداً از این برای وابستگی به کد شخص ثالث استفاده خواهیم کرد.

شما باید هر دوی crate root و فهرست کامل منابع را مشخص کنید. `crate_root` فایلی است که به کامپایلر Rust داده می‌شود که نشان‌دهنده فایل ریشه واحد کامپایل است --- معمولاً به‌نام `lib.rs` است. همین‌طور `sources` فهرست کاملی از تمام فایل‌های منبعی است که `ninja` برای تعیین زمان لازم برای بازسازی به آن‌ها نیاز دارد.

(چیزی به نام `source_set` Rust وجود ندارد، زیرا در Rust، تمامی crate یک واحد جمع‌آوری است. `static_library` کوچک‌ترین واحد است.)

دانش‌آموزان ممکن است تعجب کنند که چرا به جای استفاده از پشتیبانی داخلی gn برای کتابخانه‌های استاتیک Rust به یک الگوی gn نیاز داریم. پاسخ این است که این الگو از CXX interop، ویژگی‌های Rust و تست‌های واحد پشتیبانی می‌کند که بعداً از برخی از آن‌ها استفاده خواهیم کرد.

43.1 شامل کد **unsafe Rust**

کد ناامن Rust به طور پیش‌فرض در `rust_static_library` غیرمجاز است --- کامپایل نمی‌شود. اگر به کد unsafe Rust نیاز دارید، `allow_unsafe = true` را به هدف gn اضافه کنید. (بعداً در دوره ما شرایطی را خواهیم دید که در آن لازم است.)

```
("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
                                {
```

43.2 بس‌ته به **Rust Code** از **Chromium C++**

به سادگی هدف بالا را به deps برخی از اهداف Chromium C++ اضافه کنید.

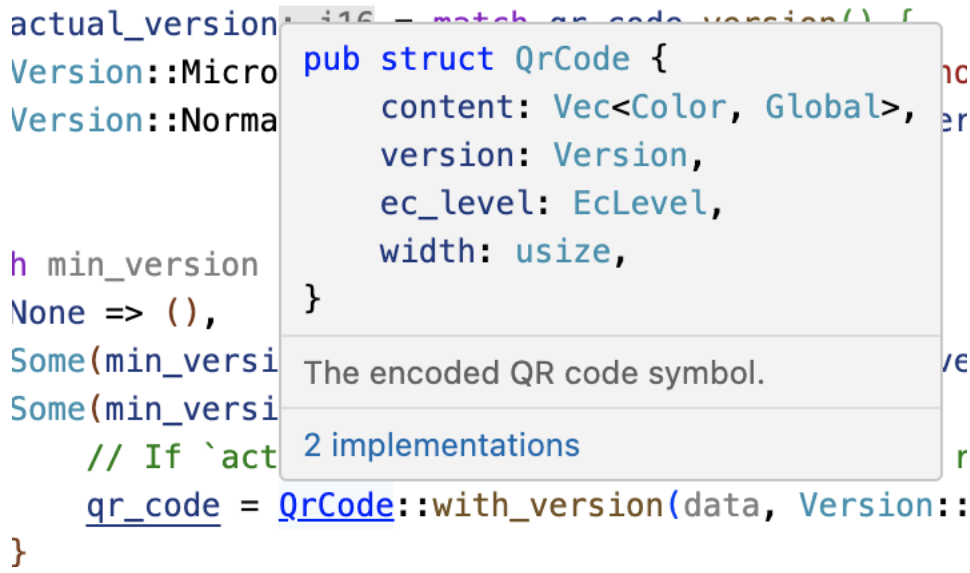
```
("import("//build/rust/rust_static_library.gni

} ("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
                                {
```

We'll see that this relationship only works if the Rust code exposes plain C APIs which can be called from C++, or if we use a C++/Rust interop tool.

43.3 Visual Studio Code

تای‌په‌ا در کد Rust حذف شده اند که باعث می‌شود یک IDE خوب حتی مفیدتر از C++ باشد. کد ویژوال استودیو و برای Rust در Chromium به خوبی کار می کند و برای استفاده از آن،

- اطمینان حاصل کنید که VSCode شما دارای extension `rust-analyzer` است، نه فرمه‌ای قبلی پشتیبانی از Rust
- `gn gen out/Debug --export-rust-project` (یا معادل آن برای دایرکتوری خروجی شما)
- `ln -s out/Debug/rust-project.json rust-project.json`



اگر مخاطب به طور طبیعی نسبت به IDEها علاقه نداشته باشد، نمایش برخی از ویژگی‌های code annotation و کاوش در rust-analyzer می‌توانند مفید باشد.

مراحل زیر ممکن است به نسخه نمایشی کمک کند (اما در عوض از یک قطعه Rust مربوط به Chromium که بیشتر با آن آشنا هستید استفاده کنید):

- components/qr_code_generator/qr_code_generator_ffi_glue.rs را باز کنید
- مکان نما را روی فراخوانی QrCode::new (حدود خط 26) در ‘qr_code_generator_ffi_glue.rs’ قرار دهید
- نسخه‌ی نمایشی **نمایش مستندات** (typical bindings: vscode = ctrl k i; vim/CoC = K).
- نسخه‌ی Demo ای نمایشی g d (typical bindings: vscode = F12; vim/CoC = **go to definition**). (این شما را به //third_party/rust/.../qr_code-.../src/lib.rs می‌رساند.)
- نسخه آزمایشی **outline** و در ادامه به متد QrCode::with_bits بروید (حدود خط 164؛ طرح کلی در پنجره file explorer در vscode است؛ typical vim/CoC bindings = space o)
- نسخه‌ی نمایشی **type annotations** (مثال های بسیار خوبی در متد QrCode::with_bits وجود دارد)

ممکن است هم باشد که gn gen ... --export-rust-project بای د پس از ویرایش فایل‌های BUILD.gn (که در طول تمرین‌های این چاپ‌سه چند بار انجام می‌دهیم) دوباره اجرا شود.

43.4 تمرین: ساخت وقایع

در ساخت Chromium خود، یک Rust target جدید به //ui/base/BUILD.gn اضافه کنید که حاوی:

```
[no_mangle]#
} ()pub extern "C" fn hello_from_rust
    ("!Rust از سلام")!println
{
```

هم: توجه داشته باشید که no_mangle در اینجا توسط کامپایلر Rust نوعی نام (type of) در نظر گرفته می‌شود، بنابراین باید کد unsafe را در target gn خود مجاز کنید.

این هدف جدید Rust را به عنوان وابستگی به //ui/base:base اضافه کنید. این تابع را در بالای ui/base/resource/resource_bundle.cc اعلام کنید (بعداً خواهیم دید که چگونه می‌توان این

کار را با ابزاره‌ای تولید bindings خودکار کرد):

```
extern "C" void hello_from_rust();
```

این تابع را از جایی در `ui/base/resource/resource_bundle.cc` فراخوانی کنید - ما قسمت `ResourceBundle::MaybeMangleLocalizedString` را پیشنهاد می‌کنیم. Chromium را Build و اجرا کنید و مطمئن شوید که "Hello from Rust!" باره‌ا چاپ می‌شود.

اگر از VSCode استفاده می‌کنید، اکنون Rust را تنظیم کنید تا در VSCode به خوبی کار کند. این کار در تمرین‌های بعدی مفید خواهد بود. اگر موفق شده‌اید، می‌توانید از کلیک راست روی "Go to definition" در `println!` استفاده کنید.

کجا می‌توان **help** پیدا کرد

- گزینه‌های موجود برای `rust_static_library` gn template
- اطلاعات درباره `#[no_mangle]`
- اطلاعات درباره `"extern "C`
- اطلاعاتی درباره gnهای `switch--export-rust-project`
- How to install rust-analyzer in VSCode

.It's really important that students get this running, because future exercises will build on it

این مثال غیرعادی است زیرا به زبان متقابل با کمترین مخرج مشترک، C خلاصه می‌شود. بعداً در دوره، C++ آن را مستقیماً به Rust وصل خواهیم کرد.

`allow_unsafe = true` در اینجا مورد نیاز است زیرا `#[no_mangle]` ممکن است به Rust اجازه دهد دو تابع با نام یکسان تولید کند و Rust دیگر نمی‌تواند تضمین کند که تابع مورد نظر فراخوانی شده است.

اگر به یک فایل اجرایی Rust خالص نیاز دارید، می‌توانید این کار را با استفاده از الگوی `rust_executable` gn نیز انجام دهید.

فصل 44

تست‌کردن

چارچوب Rust معمولاً unit testهای را در یک ماژول قرار می‌دهد که در همان فایل منبع کد مورد آزمایش قرار می‌گیرد. این مورد قبلاً در دوره پوشش داده شده بود و به این صورت است:

```
#[cfg(test)]
mod tests {
    #[test]
    fn my_test() {
        todo!()
    }
}
```

در Chromium بایدها unit testها را در یک فایل منبع جداگانه قرار دهیم و همچنان این روش را برای Rust دنبال می‌کنیم --- این باعث می‌شود تست‌ها به طور مداوم قابل کشف باشند و کمک می‌کند از بازسازی فایل‌های `.rs` برای بار دوم (در پیکربندی `test`) جلوگیری شود.

این منجر به گزینه‌های زیر برای تست کد Rust در Chromium می‌شود:

- تست‌های Native Rust (یعنی `#[test]`). خارج از `third_party/rust/` مناسب نیست.
- تست‌های `gtest` که در C++ نوشته شده‌اند و Rust را از طریق تماس‌های FFI انجام می‌دهند. زمانی که کد Rust فقط یک لایه نازکی از FFI است و unitها موجود، پوشش کافی برای ویژگی‌های که ارائه می‌کنند کافی است.
- آزمایش‌های `gtest` که در Rust نوشته شده‌اند و از crate تحت آزمایش از طریق API عمومی آن استفاده می‌کنند (در صورت نیاز از `pub mod for_testing { ... }` » استفاده می‌کنند. این موضوع در چند اسلاید بعد معرفی شده است.

ذکر کنید که تست‌های Rust native در مورد crateهای شخص ثالث باید در نهایت توسط روبات‌های Chromium انجام شود. (چنین آزمایشی به ندرت مورد نیاز است --- فقط پس از افزودن یا به‌روزرسانی crateهای شخص ثالث.)

برخی از مثال‌ها ممکن است به توضیح این‌که چه زمانی باید از `gtest` C++ در مقابل `gtest` Rust استفاده شود کمک کند:

- QR عملکرد بسیار کمی در لایه Rust شخص اول دارد (این فقط یک چسب FFI باریک است) و بنابراین از unitهای C++ موجود برای آزمایش C++ و اجرای Rust استفاده می‌کند (تست‌ها را پارامتر می‌کند تا Rust را با استفاده از یک `ScopedFeatureList` آن را فعال یا غیرفعال کند.
- Hypothetical/WIP PNG ممکن است نیاز به پیاده‌سازی این من از حافظه تبدیل‌های پیکسلی داشته باشد که توسط `libpng` ارائه شده‌اند اما در png crate وجود ندارند - به عنوان مثال.

RGBA => BGRA یا تصحیح‌گر گاما (gamma correction). چنین عملکردی ممکن است از آزمایش‌های جداگانه‌ای که در Rust نوشته شده است به‌رهمند شود.

44.1 `rust_gtest_interop` Library

کتابخانه `rust_gtest_interop` راه‌ی را ارائه می‌دهد:

- از یک تابع Rust به عنوان یک تست `gtest` استفاده کنید (با استفاده از `#[gtest(...)]` (attribute
- از `expect_eq!` و ماکروهای مشابه (شبیه به `assert_eq!` استفاده کنید، اما وقتی assertion ناموفق بود، panic نکنید و تست را خاتمه ندهید).

مثال:

```
use rust_gtest_interop::prelude::*;

#[gtest(MyRustTestSuite, MyAdditionTest)]
fn test_addition() {
    expect_eq!(2 + 2, 4);
}
```

44.2 قواعد GN برای تست‌های Rust

ساده‌ترین راه برای `gtest` build Rust اضافه‌کردن آن‌ها به یک بایناری تست موجود است که از قبل حاوی تست‌هایی است که در C++ نوشته شده‌اند. به عنوان مثال:

```
("test("ui_base_unittests") {
    ...
    "sources += [ "my_rust_lib_unittest.rs" ]
    "deps += [ ":my_rust_lib" ]
}
```

نگارش تست‌های Rust در یک `static_library` جداگانه نیز کار می‌کند، اما نیاز به اعلام دستی وابستگی به کتابخانه‌های پشتیبانی دارد:

```
("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" ]
}
```

`chromium::import!` Macro 44.3

پس از افزودن `my_rust_lib:` به GN deps، همچنان باید نحوه وارد کردن و استفاده از `my_rust_lib` را از `my_rust_lib_unittest.rs` یاد بگیریم. ما یک `crate_name` صریح برای `my_rust_lib` ارائه نکرده‌ایم، بنابراین نام crate آن بر اساس مسیر و نام کامل هدف محاسبه می‌شود. خوشبختانه ما می‌توانیم با استفاده از ماکرو `chromium::import!` از chromium crate که به‌طور خودکار وارد می‌شود، درنتیجه از کار با چنین نام‌های پرهیز کنیم:

```
} !chromium::import
    ;"ui/base:my_rust_lib//"
                            {
```

```
;use my_rust_lib::my_function_under_test
```

در زیر جلد‌ها، ماکرو به چیزی شبیه به زیر گسترش می یابد:

```
;extern crate ui_sbase_cmy_urust_ulib as my_rust_lib
```

```
;use my_rust_lib::my_function_under_test
```

اطلاعات بیشتر را می‌توانید در doc comment پیدا کنید. ماکرو مربوطه `chromium::import`.

`rust_static_library` از تعیین نام صریح از طریق ویژگی`crate_name` پشتیبانی می‌کند، اما انجام این کار ممنوع است و از آن جلوگیری می‌شود زیرا نام crate باید در سطح سراسری crates.io منحصربه‌فرد باشد. منحصربه‌فرد بودن نام crateهای خود را تضمین می‌کند، بنابراین اهداف GN `cargo_crate` (تولید شده توسط ابزار `gnrt` که در بخش بعدی پوشش داده شده است) از نام‌های crate کوتاه استفاده می‌کنند.

44.4 تمرین تستی

وقت یک تمرین دیگر است!

در Chromium build شما باید:

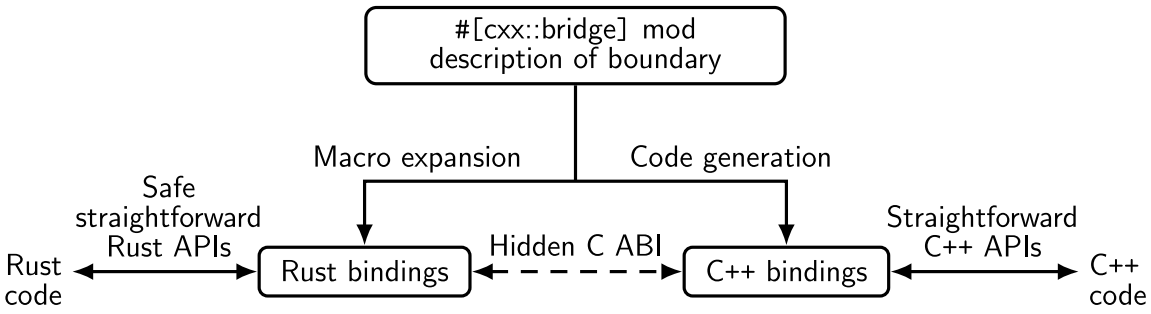
- یک تابع قابل آزمایش در کنار `hello_from_rust` اضافه کنید. چند پیشنهاد: اضافه کردن دو عدد صحیح دریافت شده به عنوان آرگومان، محاسبه عدد فیبوناچی nام، جمع اعداد صحیح در یک برش و غیره.
- یک فایل `unittest.rs_...` جداگانه با یک تست برای تابع جدید اضافه کنید.
- تست‌های جدید را به `BUILD.gn` اضافه کنید.
- تست‌ها را بسازید، اجرا کنید و بررسی کنید که آیا تست جدید به درستی کار می‌کند.

فصل 45

قابلیت همکاری با C++

جامعه Rust گزینه‌های متعددی را برای C++/Rust interop ارائه می‌دهد، با ابزاره‌ای جدیدی که همیشه در حال توسعه هستند. در حال حاضر، Chromium از ابزاری به نام CXX استفاده می‌کند.

شما کل مرز زبان (language boundary) خود را در یک زبان تعریف interface (که بسیار شبیه Rust به نظر می رسد) توصیف می کنید و سپس ابزاره‌ای CXX اعلان‌ه‌ای را برای توابع و تایپ‌ها در Rust و C++ ایجاد می‌کنند.



برای مثال کامل استفاده از این آموزش CXX را ببینید.

از طریق دیاگرام صحبت کنید. توضیح دهید که در پشت صحنه، این دقیقاً همان کاری را انجام می‌دهد که قبلاً انجام می‌دادید. به این نکته اشاره کنید که خودکارسازی فرآیندهای دارای مزایای زیر است:

- این ابزار تضمین می‌کند که مجانب C++ و Rust مطابقت کامل دارند (مثلاً اگر `#[cxx::bridge]` در حالت واقعی C++ یا Rust مطابقت نداشته باشد، درنتیجه با خطاهای کامپایل مواجه می‌شوید اما با اتصال‌های دستی غیرهمگام‌شده (out-of-sync manual bindings) می‌توانید رفتار نامشخص دریافت کنید)
- این ابزار تولید Thunkهای FFI (کارکرده‌ای کوچک، سازگار با C-ABI، و رایگان) را برای ویژگی های زیر C (به عنوان مثال فعال کردن تماس‌های FFI به متدهای Rust) به طور خودکار انجام می دهد؛ اتصال های دستی نیاز به نوشتن چنین عملکرده‌ای سطح بالا و رایگان به صورت دستی دارد.
- ابزارها و کتابخانه می‌توانند مجموعه‌ای از انواع هسته (core types) را مدیریت کنند - به عنوان مثال:
  - `&[T]` را می‌توان از مرز FFI عبور داد، حتی اگر هیچگونه طرح‌بندی ABI یا حافظه خاصی را تضمین نکند. با اتصال‌های (bindings) دستی، `&[T]` / `std::span<T>` بایاب به‌صورت دستی تخریب شود و از یک pointer به صورت مجدد ساخته شود - با توجه به اینکه هر زبان sliceهای خالی را کمی متفاوت نشان می‌دهد، درنتیجه مسعطا است)

- اشاره گره‌ای هوشمند مانند `std::unique_ptr<T>`, `std::shared_ptr<T>` و/یا Box به صورت native پشتیبانی می‌شوند. با کمک اتصال‌های دستی (manual bindings)، باید C-ABI-compatible raw pointers را پاس کنید که خطرات ایمنی و طول عمر را افزایش می‌دهد.
- تایپ‌های `rust::String` و `CxxString` تفاوت‌ها را در نمایش stringها در بین زبان‌ها درک و حفظ می‌کنند (به عنوان مثال، `rust::String::lossy` می‌تواند یک Rust string را از ورودی غیر UTF8 و `rust::String::c_str` بسازد و می‌تواند یک string را با NUL خاتمه دهد).

45.1 نمونه اتصال‌ها (Bindingها)

CXX مستلزم آن است که کل مرز C++/Rust در ماژول‌های `cxx::bridge` در کد منبع `.rs` اعلام شود.

```
#[cxx::bridge]
} 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
```

اشاره کنید:

- اگرچه این شبیه به یک mod معمولی Rust است، اما ماکرو رویه‌ای `#[cxx::bridge]` کاره‌ای پیچیده‌ای برای آن انجام می‌دهد و کد تولید شده کمی پیچیده تر است - اگرچه این کار همچنان منجر به یک mod به نام `ffi` در کد شما می شود.
- پشتیبانی Native در C++ برای `std::unique_ptr`در Rust
- پشتیبانی Native برای Rust Slices در C++
- فراخوانی از C++ به Rust و تایپ‌های Rust (در قسمت بالا)
- فراخوانی از Rust به C++ و تایپ‌های C++ (در قسمت پایین)

تصور نادرست رایج**: به نظر می‌رسد دِیر C++ توسط Rust تجزیه می‌شود، اما این گمراه کننده است. این دِیر هرگز توسط Rust تفسیر نمی‌شود، بلکه به سادگی `#include` در کد C++ تولید شده برای کامپایلر C++ است.

محدودیت‌های CXX

تا حد زیادی مفیدترین صفحه هنگام استفاده از CXX برابر type reference است.

CXX به‌طور کلی مناسب مواردی است که:

- این‌ترفیس Rust-C++ شما به اندازه کافی ساده است که می‌توانید همه آن را اعلام declare کنید.
- شما فقط از تایپ‌هایی استفاده می‌کنید که قبلاً توسط CXX پشتیبانی می‌شوند، برای مثال `&[u8]`, `std::string`, `std::unique_ptr` و غیره.

این مورد محدودیت‌های زیادی دارد --- برای مثال عدم پشتیبانی از تایپ 'Option' Rust.

این محدودیت‌ها ما را محدود می‌کنند تا از Rust در Chromium فقط برای "گره‌های برگ" به خوبی ایزوله شده استفاده کنیم نه برای تعامل دلخواه Rust-C++. هنگام در نظر گرفتن یک مورد استفاده برای Rust در Chromium، یک نقطه شروع خوب این است که پیش نویس پیوندهای CXX برای مرز زبان (language boundary) را پیش‌نویس کنید تا ببینید آیا به اندازه کافی ساده به نظر می رسد یا خیر.

In addition, right now, Rust code in one component cannot depend on Rust code in another, due to linking details in our component build. That's another reason to restrict Rust to use in leaf nodes.

همچنین با برخی از نکات مهم دیگر را با CXX طرح کنید، به عنوان مثال:

- مدیریت خطای آن بر اساس C++ exception است (در اسلایدبعدی ارائه شده است)
- استفاده از Function pointerها دشوار است.

45.2 مدیریت خطا CXX

CXX پشتیبانی از `Result<T,E>` به C++ exception متکی است، بنابراین نمی‌توانیم از آن در Chromium استفاده کنیم. جایگزین‌های آن عبارتند از:

- قسمت T از نتیجه `<T, E>` می تواند باشد:
  - از طریق پارامترهای خارجی (به عنوان مثال از طریق `&mut T`) برگردانده شده است. این مستلزم آن است که T بتواند از مرز FFI عبور کند - برای مثال T باید:
    - یک type اولیه (مانند u32 یا `usize`)
    - تایپی که به طور native توسط cxx پشتیبانی می‌شود (مانند `UniquePtr<T>`) که دارای یک مقدار پیش‌فرض معناداری است که در موارد خرابی است (*unlike* `Box<T>`).
  - در ستم Rust حفظ شده و از طریق مرجع در معرض دید قرار گرفته است. این کار ممکن است زمانی مورد نیاز باشد که T یک تایپ Rust است که نمی‌تواند از مرز FFI عبور کند و نمی توانند در `UniquePtr<T>` ذخیره شوند.
- قسمت E از `Result<T, E>` می‌تواند باشد:
  - به‌عنوان یک boolean برگردانده می‌شود (مثلاً `true` نشان‌دهنده موفقیت و `false` نشان-دهنده یک شکست است)
  - حفظ جزئیات خطا در تئوری امکان پذیر است، اما تاکنون در عمل مورد نیاز نبوده است.

45.2.1 مدیریت خطا CXX: مثال QR

تولید کد QR نمونه‌ای است که در آن بولی برای ارتباط موفقیت در مقابل شکست و جایی که نتیجه موفقیت آمیز را می‌توان از مرز FFI عبور داد استفاده می شود:

```
[("cxx::bridge(namespace = "qr_code_generator]#
                                    } 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 <- (
                                                                                        {
                                                                                            {
```

دانش آموزان ممکن است در مورد معنای خروجی `out_qr_size` کنجکاو باشند. این اندازه بردار نیست بلکه اندازه کد QR است (و مسلماً کمی اضافی است - درواقع این جذر اندازه بردار است).

شاید لازم باشد قبل از فراخوانی تابع Rust به اهمیت مقداردهی اولیه `out_qr_size` اشاره کنیم. ایجاد یک مرجع Rust که به حافظه اولیه اشاره می‌کند، منجر به رفتار نامشخص می‌شود (برخلاف C++، زمانی که تنها عمل عدم ارجاع چنین حافظه‌ای منجر به UB می‌شود).

اگر دانش‌آموزان درباره `Pin` سؤال می‌کنند، توضیح دهید که چرا CXX برای ارجاع‌های قابل تغییر به داده‌های C++ به آن نیاز دارد: پاسخ این است که داده‌های C++ را نمی‌توان مانند داده‌های Rust جابه‌جا کرد، زیرا ممکن است حاوی نشانگرهای خودارجاعی (self-referential pointers) باشد.

45.2.2 مدیریت خطا CXX: مثال PNG

نمونه اولیه PNG decoder نشان می‌دهد که وقتی تجزیه موفقیت‌آمیز نمی‌توانند از مرز FFI عبور کنند و چه کاری می‌توان انجام داد:

```
                                                [("cxx::bridge(namespace = "gfx::rust_bindings]#
                                                                                    } 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
                                                                            {
                                                                                {
```

"PngReader" و "ResultOfPngReader" تایپ‌های Rust هستند --- objectهای از این نوع نمی‌توانند بدون جهت‌گیری غیرمستقیم `Box<T>` از مرز FFI عبور کنند. ما نمی‌توانیم `out_parameter: &mut` را براساس مقدار ذخیره کنند. PngReader داشته باشیم، زیرا CXX به C++ اجازه نمی‌دهد Rust object ها

این مثال نشان می‌دهد که حتی اگر CXX از genericها و templateهای دلخواه پشتیبانی نمی‌کند، ما همچنان می‌توانیم آن‌ها را از مرز FFI عبور دهیم و آن‌ها را به‌صورت دستی تخصیص/تک‌شکلی (specializing / monomorphizing) در یک نوع غیرعمومی تبدیل کنیم. در مثال `ResultOfPngReader` یک نوع non-generic است که به متدهای مناسب `Result<T, E>` (به عنوان مثال به «is_err»، «unwrap»» و/یا «as_mut» ارسال می‌شود.

استفاده از **cxx** در **Chromium**

در Chromium، یک `mod] #[cxx::bridge` مستقل برای هر برگ گره‌ای که می‌خواهیم از Rust استفاده کنیم را تعریف می‌کنیم. شما معمولاً برای هر `rust_static_library` یکی دارید. پس فقط اضافه کنید.

```
                                                [ "cxx_bindings = [ "my_rust_file.rs
      list of files containing #[cxx::bridge], not all source files #
                                                    allow_unsafe = true
```

به هدف `rust_static_library` موجود در کنار `crate_root` و `sources`.

headerهای C++ در یک مکان منطقی تولید می‌شوند، بنابراین شما می‌توانید

```
"include "ui/base/my_rust_file.rs.h#
```

برخی از توابع کاربردی را در `base//` برای تبدیل به/از تایپ‌های C++ Chromium به انواع CXX Rust پیدا خواهید کرد --- برای مثال [SpanToRustSlice](https://source.chromium.org/chromium/chromium/src/+/main:base/containers/span_rust.h;l=21).

دانش‌آموزان ممکن است بپرسند --- چرا هنوز به `allow_unsafe = true` نیاز داریم؟

پاسخ کلی این است که هیچ کد C/C++ با استانداردهای معمول Rust "ایمن" نیست. فراخوانی مجدد و برگشتی به C/C++ از Rust ممکن است کارهای دلخواه را در حافظه انجام دهد و ایمنی طرح‌بندی داده‌های خود Rust را به خطر بیندازد. وجود کلمات کلیدی بسیار زیاد «ناامن» در تعامل C/C++ می‌توانند به نسبت سیگنال به نویز چنین کلمه کلیدی آسیب برسانند و این [چنجال‌برانگیز](https://steveklabnik.com/writing/the-cxx-debate) است، اما به طور دقیق، آوردن هر کد خارجی به یک برنامه Rust می‌تواند باعث رفتار غیرمنتظره از دیدگاه Rust شود.

پاسخ دقیق‌تر دردناگرام بالای این صفحه نهفته است --- در پشت صحنه، CXX توابع Rust «ناامن» و `extern "C"` را درست می‌کند در بخش قبل به صورت دستی انجام دادیم.

45.3 تمرین: قابلیت همکاری با C++

قسمت اول

- در فایل Rust که قبلاً ایجاد کرده‌اید، یک `#[cxx::bridge]` اضافه کنید که یک تابع را مشخص می‌کند که باید از C++ فراخوانی شود که نام `hello_from_rust` دارد، بدون اینکه پارامتر و هیچ مقداری برگرداند.
- تابع `hello_from_rust` قبلی خود را برای حذف `extern "C"` و `#[no_mangle]` تغییر دهید. حالا این فقط یک تابع است‌ندارد Rust است.
- هدف `gn` خود را برای ایجاد این پیوندها (bindings) تغییر دهید.
- در کد C++ خود، forward-declaration برای `hello_from_rust` را حذف کنید. در عوض، فایل هِدِر تولید شده را اضافه کنید.
- Build و !run

قسمت دوم

ایده خوبی است که کمی با CXX بازی کنید. این به شما کمک می کند تا به این فکر کنید که Rust در Chromium واقعا چقدر انعطاف پذیر است.

برخی از چیزهایی که باید امتحان کنید:

- از Rust دوباره به C++ فراخوانی کنید. درنهایت شما نیاز خواهید داشت:

  - یک فایل هِدِر اضافی که می‌توانید از `cxx::bridge` خود `include!` را وارد کنید. شما باید تابع C++ خود را در آن فایل هِدِر جدید اعلام کنید.
  - یک بلوک `unsafe` برای فراخوانی چنین تابعی، یا به طور متناوب کل ماژول را در `#[cxx::bridge]` خود همانطور که در اینجا توضیح داده شده است.
  - همچنین ممکن است لازم باشد `#include "third_party/rust/cxx/v1/crate/include/cxx.h"` را وارد کنید.
- یک رشته C++ را از Rust به C++ منتقل کنید.
- ارسال یک reference از یک object C++ به Rust .
- عمداً امضاهای تابع Rust را که از `#[cxx::bridge]` مطابقت ندارند، دریافت کنید و به خطاهایی که می بینید عادت کنید.
- عمداً امضاهای تابع C++ را که از `#[cxx::bridge]` مطابقت ندارند، دریافت کنید و به خطاهایی که می بینید عادت کنید.
- یک `std::unique_ptr` از نوعی از C++ را به Rust ارسال کنید، به طوری که Rust بتواند دارای یک C++ object باشد.
- یک Rust object ایجاد کنید و آن را به C++ ارسال کنید تا C++ مالک آن باشد. (نکته: شما به یک Box نیاز دارید).
- چند متد را در نوع C++ اعلام کنید. آن‌ها را از Rust فراخوانی کنید.
- چند متد را در Rust type اعلام کنید. از C++ آن‌ها را فراخوانی کنید.

قسمت سوم

اکنون نقاط قوت و محدودیت‌های CXX interop را درک کرده‌اید، به چند مورد استفاده برای Rust در Chromium فکر کنید که در آن رابط به اندازه کافی ساده باشد. نحوه تعریف این رابط را ترسیم کنید.

کجا می‌توان **help** پیدا کرد

- یک cxx binding reference
- یک rust_static_library gn template

As students explore Part Two, they're bound to have lots of questions about how to achieve .these things, and also how CXX works behind the scenes

برخی از سؤالاتی که ممکن است با آن مواجه شوید:برخی از سوالاتی که ممکن است با آن مواجه شوید:

- من مشکلی در مقداردهی اولیه یک متغیر از نوع X با نوع Y می‌بینم، که در آن X و Y هر دو نوع تابع هستند. بخاطر اینکه تابع C++ شما کاملاً با اعلان موجود در `cxx::bridge` شما مطابقت ندارد.
- به نظر می‌رسد می‌توانم آزادانه مراجع C++ را به همان Rust تبدیل کنم. آیا این خطر UB را ندارد؟ برای تایپ‌های CXX *opaque*، خیر، زیرا اندازه آن‌ها صفر است. برای انواع بیه‌مت CXX بله، ممکن است باعث UB شود، اگرچه طراحی CXX ساخت چنین نمونه‌ای را بسیار دشوار می‌کند.

فصل 46

اضافه‌کردن **Crate**های شخص ثالث

کتابخانه‌های "Rust Crates" نام‌ده می‌شوند و در crates.io یافت می‌شوند. وابستگی cratesهای Rust به یکدیگر بسیار آسان است. بنابراین آن‌ها این کار را انجام می‌دهند!

ویژگی	C++ library	Rust crate
Build system	تعداد زیادی	یکپارچگی: `Cargo.toml`
اندازه کتابخانه معمولی	Large-ish	کوچک
وابستگی‌های گذرا	Few	تعداد زیادی

برای یک مهندس Chromium، این مزایا و معایب دارد:

- همه crateها از یک سیستم ساخت مشترک استفاده می‌کنند، بنابراین می‌توانیم گنجاندن آن‌ها در Chromium را خودکار کنیم...
- ... اما، crateها معمولاً وابستگی‌های گذرا دارند، بنابراین احتمالاً مجبور خواهید بود چندین کتابخانه را بیاورید.

بحث خواهیم کرد:

- نحوه قرار دادن یک crate در درخت کد منبع Chromium
- چگونه قوانین ساخت gn برای آن ایجاد کنیم
- نحوه بررسی کد منبع آن برای ایمنی کافی

All of the things in the table on this slide are generalizations, and counter-examples can be found. But in general it's important for students to understand that most Rust code depends on other Rust libraries, because it's easy to do so, and that this has both benefits and costs.

پیکربندی فایل **`Cargo.toml`** برای افزودن **crate**ها

Chromium دارای یک مجموعه واحد از وابستگی‌های crate مستقیم با مدیریت مرکزی است. این‌ها از طریق یک `Cargo.toml` مدیریت می‌شوند:

```
[dependencies]
"bitflags = "1
 "cfg-if = "1
```

```
"cxx = "1
...lots more #
```

مانند در `Cargo.toml` دیگری، می‌توانید جزئیات بیشتر در مورد وابستگی‌ها را مشخص کنید --- معمولاً شما می‌خواهید «ویژگی‌ای» را که می‌خواهید در crate فعال کنید را مشخص کنید.

هنگام افزودن crate به Chromium، اغلب باید اطلاعات اضافی را در یک فایل اضافی، `gnrt_config.toml` ارائه کنید، که در ادامه با آن آشنا خواهید شد.

46.2 تنظیمات `gnrt_config.toml`

در کنار `Cargo.toml` یک `gnrt_config.toml` قرار دارد. این شامل برنامه‌های extension مخصوص Chromium برای مدیریت crate است.

اگر crate جدیدی اضافه کنید، باید حداقل`group` را مشخص کنید. این یکی از:

```
.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'   #
```

به عنوان مثال،

```
[crate.my-new-crate]
group = 'test' # only used in test code
```

بسته به طرح کد منبع crate، ممکن است لازم باشد از این فایل برای تعیین محل فایل (های) مجوز آن استفاده کنید.

بعداً، موارد دیگری را که برای حل مشکلات باید پیکربندی کنید، در این فایل مشاهده خواهید کرد.

46.3 دانلود کردن Crateها

ابزاری به نام `gnrt` می‌داند که چگونه جعبه‌ها را بارگیری کند و چگونه قواعد `BUILD.gn` را ایجاد کند.

برای شروع، crate مورد نظر خود را به صورت زیر دانلود کنید:

```
cd chromium/src
vpython3 tools/crates/run_gnrt.py -- vendor
```

> اگرچه ابزار`gnrt` بخشی از کد منبع Chromium است، با اجرای این دستور، وابستگی‌های آن را از `crates.io` دانلود و اجرا می‌کند. بخش قبلی را در مورد این تصمیم‌گیری ببینید.

این `vendor command` ممکن است بارگیری کند:

- جعبه‌های (crates) کاربردی شما
- وابستگی های مستقیم و گذرا
- نسخه‌های جدید crateهای دیگر، همانطور که `cargo` برای حل کردن مجموعه کامل جعبه‌های مورد نیاز Chromium لازم است.

Chromium وصله‌هایی را برای برخی crateها نگه‌داری می‌کند که در `third_party/rust/chromium_crates_io/patches/` نگه‌داری می‌شوند. این‌ها به‌طور خودکار دوباره اعمال می‌شوند، اما اگر وصله نشد، ممکن است نیاز به اقدام دستی داشته باشید.

46.4 ایجاد قواعد gn Build

هنگامی که crate را دانلود کردید، فایل‌های BUILD.gn را مانند این تولید کنید:

```
vpython3 tools/crates/run_gnrt.py -- gen
```

اکنون git status را اجرا کنید. شما باید این موارد را پیدا کنید:

- حداقل یک کد منبع جدید در third_party/rust/chromium_crates_io/vendor
- حداقل یک BUILD.gn جدید در third_party/rust/<crate name>/v<major semver version>
- یک README.chromium مناسب

لطفا به مرجع Rust مراجعه کنید.

نگاه دقیق بیندازید، به‌خصوص به چیزهایی که در third_party/rust ایجاد می‌شوند.

کمی در مورد semver --- و به‌ویژه روشی که در Chromium اجازه می‌دهد چندین نسخه ناسازگار از crateها را مجاز کند، صحبت کنید. این مورد که در اکوسیستم Cargo منع می‌شود ولی گاهی اوقات ضروری است.

46.5 حل مشکلات

اگر build شما با شکست مواجه شد، ممکن است به دلیل build.rs باشد: برنامه‌هایی که کارهای دلخواه را در زمان build انجام می‌دهند. این مورد به‌طورکلی در تضاد با طراحی gn و ninja است که هدفشان قواعد build ایستا، قطعی برای به حداکثر رساندن موازی‌سازی و تکرارپذیری buildها است.

برخی از اقدامات build.rs به طور خودکار پشتیبانی می‌شوند. دیگران نیاز به اقدام دارند:

اسکریپت ساخت افکت	توسط قالب‌های gn ما پشتیبانی می‌شود	کار مورد نیاز شما
بررسی نسخه rustc برای پیکربندی ویژگی‌ها	بله	None
بررسی پلتفرم یا CPU برای پیکربندی ویژگی‌های روشن و خاموش	بله	None
تولید کردن کد	بله	بله - در gnrt_config.toml مشخص کنید
Building C/C++	خیر	اطراف آن را Patch کنید
سایر اقدامات دلخواه	خیر	اطراف آن را Patch کنید

خوشبختانه، اکثر crateها حاوی build script نیستند و خوشبختانه، اکثر build scriptها تنها دو عمل اصلی را انجام می‌دهند.

46.5.1 ساخت اسکریپت‌هایی که کد را تولید می‌کنند

اگه ninja درباره نبودن فایل‌ها اعتراض کرد، build.rs را بررسی کنید و ببینید که آیا کدهای منبع را می‌نویسد.

در این صورت، gnrt_config.toml را تغییر دهید تا build-script-outputs به crate اضافه شود. اگر این یک وابستگی گذرا است، یعنی وابستگی که کد Chromium نباید مستقیماً به آن وابسته باشد، پس allow-first-party-usage=false را نیز اضافه کنید. چندین نمونه از قبل در آن فایل وجود دارد:

```
[crate.unicode-linebreak]
allow-first-party-usage = false
["build-script-outputs = ["tables.rs
```

اکنون gnrt.py -- gen را مجدداً اجرا کنید تا فایل‌های BUILD.gn را دوباره تولید کند تا به ninja اطلاع دهید که این فایل خروجی خاص ورودی مراحل build بعدی است.

46.5.2 ساخت اسکریپت‌هایی که C++ را Build می‌کنند یا اقدامات دلخواه انجام می‌دهند

برخی از crateها از crate مربوط به cc برای build و link کتابخانه‌های C/C++ استفاده می‌کنند. crateهای دیگر C/C++ را با استفاده از bindgen در اسکریپت‌های build خود تجزیه می‌کنند. این فعالیت‌ها را نمی‌توان در زمینه Chromium پشتیبانی کرد --- سیستم ساخت gn، ninja و LLVM ما در بیان روابط بین build actionsها بسیار خاص است.

بنابراین، گزینه‌های شما عبارتند از:

- از این crateها اجتناب کنید
- یک وصله (patch) روی crate بزنید.

وصله‌ها (Patches) باید در <third_party/rust/chromium_crates_io/patches/<crate نگه‌داری شوند - برای مثال [Patchها در مقابل] (+/main:third_party/rust/chromium_crates_io/patches/cxx - و هر بار که crate را upgrade می‌کنید به‌طور خودکار توسط‘ gnrt’ اعمال می‌شوند.

46.6 وابسته به یک Crate

هنگامی که یک crate شخص ثالث اضافه کردید و قواعد build را ایجاد کردید که با توجه به نوع crate می‌توانند ساده باشد. درنتیجه هدف rust_static_library خود را پیدا کند و یک dep روی هدف lib: در crate خود اضافه کنید.

به‌طورمشخص،

```
       +----------------------+      +------------+
"third_party/rust" | crate name | "/v" | major semver version | ":lib//"
       +----------------------+      +------------+
```

به عنوان مثال،

```
                                    } ("rust_static_library("my_rust_lib
                                        "crate_root = "lib.rs
                                       [ "sources = [ "lib.rs
                [ "deps = [ "//third_party/rust/example_rust_crate/v1:lib
                                                                      {
```

46.7 حسابرسی Crateهای شخص ثالث

افزودن کتابخانه‌های جدید تابع قواعد مربوط به Chromium است، اما البته موضوع بررسی امنیتی نیز وجود دارد. از آنجایی که ممکن است نه تنها یک crate، بلکه وابستگی‌های گذرا را نیز وارد کنید،

ممکن است کده‌ای زیادی برای بررسی وجود داشته باشد. از سوی دیگر، safe Rust code می‌تواند عوارض جانبی مخدودی داشته باشد. پس چگونه باید آن را بررسی کنید؟

با گذشت زمان، Chromium قصد دارد به فرآیندی بر اساس cargo vet حرکت کند.

در همین حال، برای هر crate جدید اضافه شده، موارد زیر را بررسی می کنیم:

- بدانید که چرا هر crate استفاده می‌شود. رابطه بین crateها چیست؟ اگر سیستم ساخت هر جا به حاوی `build.rs` یا ماکروهای رویه‌ای (procedural macros) است، مشخص کنید که آن‌ها برای چه چیزی هستند. آیا آن‌ها با روشی که Chromium به طور معمول ساخته و built می‌شود سازگار هستند؟
- Check each crate seems to be reasonably well maintained
- از `cd third-party/rust/chromium_crates_io; cargo audit` استفاده کنید. بررسی cargo برای بررسی آسیب‌پذیری‌های شناخته‌شده (ابتدا با `cargo install cargo-audit` که از قضا شامل دانلود وابستگی‌های زیادی از اینترنت می‌شود2)
- مطمئن شوید هر کد `unsafe` به اندازه کافی برای قاعده دو خوب است
- هرگونه استفاده از APIهای `fs` یا `net` را بررسی کنید
- تمام کدها را در سطح کافی بخوانید تا به دنبال هر چیزی که ممکن است به طور مخرب وارد شده باشد را بگردید. (در اینجا نمی توانید به طور واقع بینانه به دنبال نتیجه ۱۰۰ درصدی باشید: اغلب کده‌ای زیادی وجود دارد.)

اینها فقط دست‌ورالعمل‌هایی هستند --- با بازبینه‌هایی از `security@chromium.org` کار کنید تا راه درستی برای اطمینان از crate پیدا کنید.

46.8 بررسی Crateها در کد منبع Chromium

`git status` باید نشان دهد:

- کد Crate در `third_party/rust/chromium_crates_io/`
- متادیتا (`BUILD.gn` و `<version>/<crate>/third_party/rust/ in //<README.chromium>`)

لطفاً یک فایل `OWNERS` در مکان دیگر نیز اضافه کنید.

باید همه اینها را به همراه تغییرات `Cargo.toml` و `gnrt_config.toml` خود در مخزن Chromium قرار دهید.

مهم: باید از `git add -f` استفاده کنید زیرا در غیر این صورت فایل‌های `.gitignore` ممکن است مانع از به حذف برخی از فایل‌ها شود.

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
```

46.9 به روز نگه داشتن Crateها

شما به‌عنوان مالک هر وابستگی شخص ثالث Chromium، انتظار می‌رود آن را با هرگونه اصلاحات امنیتی به‌روز نگه دارید. امید است که ما به زودی این را برای crateهای Rust خودکار کنیم، اما در حال حاضر، همچنان مسئولیت شماست، همانطور که برای هر وابستگی به شخص ثالث دیگر این مسئولیت را دارید.

46.10 تمرین‌ها

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

(در تمرین بعدی از uwuify برای Chromium استفاده خواهیم کرد، اما در صورت تمایل می‌توانید این کار را انجام دهید. همین‌طور می‌توانید یک هدف [rust_executable'](https://source.chromium.org/chromium/chromium/src/+/main:build/rust/rust_executable.gni) جدید ایجاد کنید که از `uwuify` استفاده می‌کند).

دانش‌آموزان باید تعداد زیادی وابستگی گذرا را دانلود کنند.

کل crateهای مورد نیاز عبارتند از:

- `instant`,
- `lock_api`,
- `parking_lot`,
- `parking_lot_core`,
- `redox_syscall`,
- `scopeguard`,
- `smallvec` و
- `uwuify`.

اگر دانش‌آموزان حتی بیشتر از آن دانلود می‌کنند، احتمالا فراموش کرده‌اند که ویژگی‌های پیش‌فرض را خاموش کنند.

با تشکر از Daniel Liu برای این crate!

فصل 47

باری جمع‌آوری آن --- تمرینی

در این تمرین، می‌خواهید یک ویژگی کاملاً جدید Chromium را اضافه کنید و همه چیزهایی را که قبلاً یاد گرفته‌اید جمع آوری کنید.

خلاصه‌ای از مصاحبه‌ای

چالش‌هایی از pixyها کشف شده است که در یک جنگل بارانی دور افتاده زندگی می کنند. مهم است که Chromium برای pixyها را در اسرع وقت به آنها تحویل دهیم.

لازمه کار این است که تمام stringهای رابط کاربری Chromium به زبان Pixie ترجمه شوند.

زمانی برای بحث‌مان درباره ترجمه‌های مناسب وجود ندارد، اما خوشبختانه زبان pixie بسیار نزدیک به انگلیسی است و به نظر می‌رسد که Rust crate ای وجود دارد که ترجمه را انجام می‌دهد.

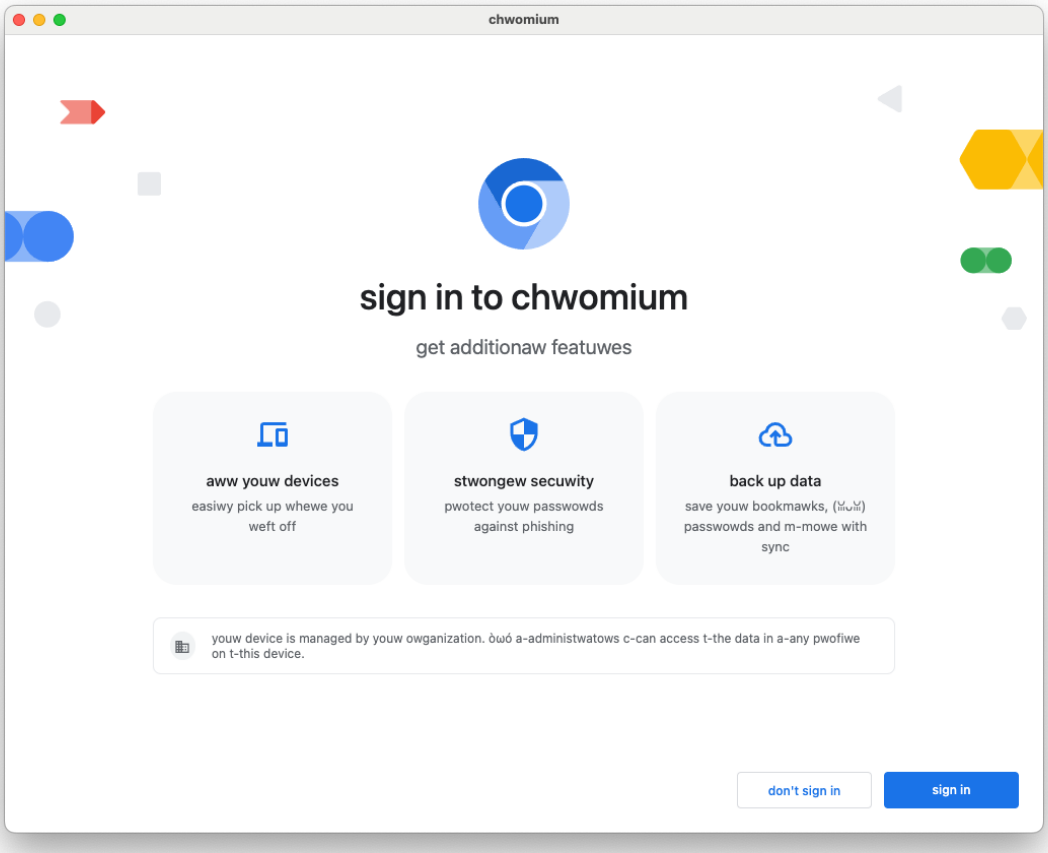
در واقع، شما قبلا آن crate را در تمرین قبلی وارد کردید.

(بدیهی است که ترجمه های واقعی Chrome نیاز به دقت و تلاش باورنکردنی دارند. این مورد را ارسال نکنید!)

گام‌ها

`ResourceBundle::MaybeMangleLocalizedString` را تغییر دهید تا همه stringها را قبل از نمایش یکپارچه کند. در این build خاص Chromium، بدون در نظر گرفتن تنظیمات `_mangle_localized_strings` همیشه باید این کار را انجام دهد.

اگر همه این تمرین‌ها را درست انجام داده‌اید، به شما تبریک می‌گویم، باید Chrome را برای pixies ایجاد می‌کردید!



:Students will likely need some hints here. Hints include

- UTF16 در مقابل UTF8. دانش‌آموزان باید بدانند که stringهای Rust همیشه UTF8 هستند و احتمالاً تصمیم خواهند گرفت که به‌تر است تبدیل را در سمت C++ با استفاده از base::UTF16ToUTF8 انجام دهند و دوباره برگردند.
- اگر دانش‌آموزان تصمیم بگیرند که تبدیل را در سمت Rust انجام دهند، باید [string::from_utf16](https://doc.rust-lang.org/std/string/struct.String.html#method) را در نظر بگیرند. مدیریت خطا را در نظر بگیرید و در نظر داشته باشید که کدام نوع های پشتیبانی شده از CXX می توانند تعداد زیادی از u16 ها را مبادله کنند.
- دانش‌آموزها ممکن است مرز C++/Rust را به روش های مختلف طراحی کنند، به عنوان مثال. گرفتن و برگرداندن stringها بر اساس مقدار، یا گرفتن یک مرجع قابل تغییر به یک string. اگر از یک مرجع قابل تغییر استفاده شود، CXX احتمالاً به دانش‌آموز می‌گوید که باید از Pin استفاده کند. ممکن است لازم باشد توضیح دهید Pin چه می‌کند و سپس توضیح دهید که چرا CXX به آن برای ارجاع‌های قابل تغییر به داده‌های C++ نیاز دارد: پاسخ این است که داده‌های C++ را نمی‌توان مانند داده‌های Rust جابه‌جا کرد، زیرا ممکن است حاوی نشانگرهای خودارجاعی (self-referential pointers) باشد.
- هدف C++ حاوی ResourceBundle::MaybeMangleLocalizedString هدف به باید rust_static_library وابسته باشد. دانش‌آموز احتمالاً از قبل این کار را انجام داده است.
- هدف rust_static_library به باید third_party/rust/uwuify/v0_2:lib// وابسته باشد.

فصل 48

نیرمت ای‌ه‌لحاره

راه‌ل‌ه‌ای ت‌م‌ی‌ر‌ن‌ات Chromium را م‌ی‌وا‌ن‌ت‌ی‌د در ای‌ن س‌ری از CLs پ‌ی‌دا ک‌ن‌ی‌د.

XI بخش

Bare Metal: باب صحرا

فصل 49

به Bare Metal Rust خوش آمدید

این یک دوره مستقل روزه در مورد bare-metal Rust است که با هدف افرادی که با اصول Rust آشنا هستند (شاید از اتمام دوره جامع Rust) و در حالت ایده آل نیز تجربه برنامه‌نویسی bare-metal به زبان دیگری را دارند، مانند C می‌باشد.

امروز ما در مورد'bare-metal' Rust صحبت خواهیم کرد: اجرای کد Rust بدون سیستم‌عامل در ادامه به چند بخش تقسیم خواهد شد:

- این no_std Rust چیست؟
- نوشتن firmware برای میکروکنترلرها.
- نوشتن کد bootloader / kernel برای پردازنده‌های برنامه.
- برخی از crateهای مفید برای توسعه bare-metal Rust.

برای بخش میکروکنترلر دوره ما از BBC micro:bit v2 به عنوان مثال استفاده خواهیم کرد. این یک برد توسعه مبتنی بر میکروکنترلر Nordic nRF52833 با چند LED و دکمه، شتاب‌سنج و قطب‌نما متصل به I2C و یک دیباگر SWD روی برد است.

برای شروع، ابزاره‌ایی را که بعداً به آن‌ها نیاز خواهیم داشت نصب کنید. در لینوکس یا دبیان:

```
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
```

و به کاربران گروه plugdev اجازه دسترسی به برنامه‌نویسی micro:bit را بدهید:

```
BSYSTEM=="hidraw", ATTRS{idVendor}=="0d28", MODE="0660", GROUP="logindev", TAG+="uaccess
                                                  sudo tee /etc/udev/rules.d/50-microbit.rules
                                                            sudo udevadm control --reload-rules
```

در 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
```

فصل 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 و 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

- یک HashMap به RNG وابسته است.
- .std re-exports the contents of both core and alloc

50.1 یک برنامه حداقلی از no_std

```
#![no_std]
#![no_main]

use core::panic::PanicInfo;

#[panic_handler]
fn panic(_panic: &PanicInfo) -> ! {
    loop {}
}
```

- این به یک باینری خالی کامپایل می‌شود.
- std provides a panic handler; without it we must provide our own.
- همچنین توان آن را توسط crate دیگری مانند panic-halt تهیه کرد.
- به‌سته به هدف، ممکن است لازم باشد برای جلوگیری از خطای eh_personality را panic = "abort" کامپایل کنید.
- توجه داشته باشید که main یا هیچ نقطه ورودی دیگری وجود ندارد. این به شما بستگی دارد که نقطه ورود خود را تعریف کنید. این معمولاً شامل یک اسکریپت linker و مقداری کد اسمبلی برای تنظیم موارد آماده برای اجرای کد Rust است.

50.2 alloc

برای استفاده از alloc باید یک global (heap) allocator را پیاده‌سازی کنید.

```
#![no_std]
#![no_main]

extern crate alloc;
extern crate panic_halt as _;

use alloc::string::ToString;
use alloc::vec::Vec;
use buddy_system_allocator::LockedHeap;

#[global_allocator]
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`ی‌ک third-party crate است که ی‌ک تخص‌ی‌ص‌ده‌نده buddy system را پی‌اده‌س‌ازی می‌کن‌د. crateهای دی‌گر در دس‌ت‌رس ه‌س‌ت‌ند ی‌ا می‌توان‌ی‌د ن‌س‌خه مرب‌وط ب‌ه خ‌ود را ب‌ن‌وی‌س‌ی‌د ی‌ا ب‌ه ت‌خ‌ص‌ی‌ص ده‌نده م‌وج‌ود خ‌ود م‌ت‌ص‌ل ک‌ن‌ی‌د.
- پ‌ارام‌ت‌ر const `LockedHeap` ح‌داک‌ث‌ر ت‌رت‌ی‌ب ت‌خ‌ص‌ی‌ص‌ده‌نده (allocator) است. ی‌ع‌نی در ای‌ن م‌ورد می‌توان‌د م‌ن‌اط‌ق‌ی ت‌ا 2**32 ب‌ای‌ت را اخ‌ت‌ص‌اص ده‌د.
- اگ‌ر ه‌ر crate ای در درخ‌ت واب‌س‌ت‌گ‌ی ش‌م‌ا ب‌ه `alloc` ب‌س‌ت‌گ‌ی دارد، ب‌ای‌د دق‌ی‌ق‌اً ی‌ک ت‌خ‌ص‌ی‌ص‌ده‌نده س‌راس‌ری در ب‌ای‌ن‌ری خ‌ود ت‌ع‌ری‌ف ک‌ن‌ی‌د. م‌ع‌م‌ولاً ای‌ن ک‌ار در binary crate س‌ط‌ح ب‌الا ان‌ج‌ام می‌ش‌ود.
- ب‌رای اط‌م‌ی‌ن‌ان از ای‌ن‌ک‌ه `panic_halt` crate ل‌ی‌ن‌ک ش‌ده است، اس‌ت‌ف‌اده `extern crate panic_halt as _` ض‌رور‌ی است، ب‌ن‌اب‌رای‌ن panic handler آن را دری‌اف‌ت می‌ک‌ن‌ی‌م.
- ای‌ن م‌ث‌ال س‌اخ‌ت‌ه می ش‌ود ام‌ا اج‌را ن‌می‌ش‌ود، زی‌را entry point ن‌دارد.

فصل 51

ارهلرتنکورکیم

کی cortex_m_rt crate (در میان چیزهای دیگر) کی reset handler برای میکروکنترلرهای Cortex M فراهم میکنند.

```
#![no_std]
#![no_main]

extern crate panic_halt as _;

mod interrupts

use cortex_m_rt::entry

#[entry]
fn main() -> ! {
    loop {}
}
```

در ادامه نحوه دسترسی به لوازم جانبی (peripherals) را با افزایش سطح انتزاع بررسی خواهیم کرد.

- ماکرو cortex_m_rt::entry مسئلزم این است که تابع دارای نوع cortex_m_rt::entry باشد، زیرا بازگشت به منطقی نیست reset handler.
- مثال را با cargo embed --bin minimal اجرا کنید

خام MMIO 51.1

اکثر میکروکنترلرها از طریق IO دارای memory-map به تجهیزات جانبی (peripherals) دسترسی دارند. بیایید سعی کنیم یک LED را در micro:bit خود روشن کنیم:

```
#![no_std]
#![no_main]

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

                                                    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

                                                                    [entry]#
                                                                } ! <- ()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
}
```

- در GPIO 0 پایه ۲۱ به ستون اول ماتریس LED و پایه ۲۸ به ردیف اول متصل است.

مثال را با:

```
cargo embed --bin mmio
```

51.2 Crateهای دسترسی جانبی

گزینه svd2rust که wrapperهای Rust عمدتاً ایمن را برای تجهیزات جانبی دارای memory-map از فایل‌های CMSIS-SVD تولید می‌کند.

```
#![no_std]
#![no_main]

use panic_halt as _;

use cortex_m_rt::entry;
use nrf52833_pac::Peripherals;

#[entry]
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()
         .input().disconnect()
         .pull().disabled()
         .drive().s0s1()
         .sense().disabled()
    });
    gpio0.pin_cnf[28].write(|w| {
        w.dir().output()
         .input().disconnect()
         .pull().disabled()
         .drive().s0s1()
         .sense().disabled()
    });

    // 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 {
```

- فایل‌های SVD (System View Description) در واقع فایل‌های XML هستند که معمولاً توسط فروشندگان تجهیزات ریزپردازنده ارائه می‌شوند که memory map دستگاه را توصیف می‌کنند.
  - آن‌ها بر اساس peripheral، register، field و value، با نام، توضیحات، آدرس و غیره سازمان‌دهی می‌شوند.
  - فایل‌های SVD اغلب دارای باگ و نقاص هستند، بنابراین پروژه‌های مختلفی وجود دارد که اشتباهات را اصلاح می‌کنند، جزئیات گم‌شده را اضافه می‌کنند و crateهای تولید شده را منتشر می‌کنند.
- cortex-m-rt جدول برداری را از جمله موارد دیگر ارائه می‌دهد.
- اگر cargo install cargo-binutils را انجام دهید، می‌توانید cargo objdump --bin pac -- -d --no-show-raw-insn را اجرا کنید تا لیستی از اپ‌کدهای اصلی را ببینید.

مثال را با:

```
cargo embed --bin pac
```

HAL crates 51.3

این crateهای [HAL](https://github.com/rust-embedded/wesome-embedded-rust#hal-implementation-crates) برای بسیاری از میکروکنترلرها بسته‌بندی‌هایی را در اطراف تجهیزات جانبی مختلف ارائه می‌دهند. این‌ها معمولاً ویژگی‌های embedded-hal را پیاده‌سازی می‌کنند.

```
#![no_std]
#![no_main]

use panic_halt as _;

use cortex_m_rt::entry;
use embedded_hal::digital::OutputPin;
use nrf52833_hal::{gpio::{p0, Level}};
use nrf52833_hal::pac::Peripherals;

#[entry]
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.
- بسیاری از HALها برای انواعی از دستگاه های Cortex-M و RISC-V از جمله میکروکنترلرهای STM32، GD32، nRF، NXP، MSP430، AVR و PIC مختلف وجود دارد.

مثال را با:

```
cargo embed --bin hal
```

Board support crates 51.4

.Board support crates provide a further level of wrapping for a specific board for convenience

```
#![no_std]
#![no_main]

use panic_halt as _;

use microbit::Board;
use embedded_hal::digital::OutputPin;
use cortex_m_rt::entry;

#[entry]
fn main() -> ! {
    let mut board = Board::take().unwrap();

    board.display_pins.row1.set_high().unwrap();
    board.display_pins.col1.set_low().unwrap();

    loop {}
}
```

- در این مورد crate پشتیبانی برد فقط نام‌های مقدار مقداردهی اولیه را ارائه می‌دهد.
- این crate ممکن است شامل درایورهایی برای برخی از دست‌گاه‌های داخلی خارج از خود میکروکنترلرز باشد.
  – ء microbit-v2 شامل یک درایور ساده برای ماتریس LED است.

مثال را با:

```
cargo embed --bin board_support
```

state pattern پیاده‌سازی یک 51.5

```
#[entry]
fn main() -> ! {
    let p = Peripherals::take().unwrap();
    let gpio0 = p0::Parts::new(p.P0);

    let pin: P0_01<Disconnected> = gpio0.p0_01;

    // Error, moved //.let gpio0_01_again = gpio0.p0_01;
    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
                                                                                    {
```

- پین‌های Copy یا Clone را اجرا نمی‌کنند، بنابراین فقط یک نمونه از هر کدام می‌تواند وجود داشته باشد. هنگامی که یک pin از ساختار پورت خارج می شود، هیچ کس دیگری نمی تواند آن را بگیرد.
- تغییر پیکربندی pin، نمونه pin قدیمی را مصرف می‌کند، بنابراین نمی‌توانید پس از آن از pin قدیمی استفاده کنید.
- این type یک مقدار state را نشان می‌دهد که در آن قرار دارد: به عنوان مثال. در این مورد، وضعیت پیکربندی یک پین GPIO. این state machine را در type system رمزگذاری می‌کند و تضمین می‌کند که سیستم از pin به روشی خاص استفاده کند بدون اینکه ابتدا آن را به درستی پیکربندی کند. state transition غیرمجاز در زمان کامپایل شناسایی می‌شود.
- می‌توانید is_high را در یک پین ورودی و set_high را در یک پایه خروجی فراخوانی کنید، اما برعکس امکان پذیر نیست.
- بسیاری از HAL crate‌ها از این الگو پیروی می‌کنند.

embedded-hal 51.6

این 'embedded-hal' crate عادت وظیفه را ارائه می‌دهد که می‌کروکنترلرهای جانبی را جایی پوشش دهد:

- GPIO
- PWM
- تایمرهای تاخیری
- گذرگاه‌ها و دستگاه‌های I2C و SPI

ویژگی‌های مشابه برای جریان‌های بایت (مانند UART)، گذرگاه‌های CAN و RNG و تقسیم شدن به [embedded-can](https ://crates.io/crates/embedded-can)، [embedded-io](https ://crates.io/crates/embedded-io) و rand_core به ترتیب.

سپس crate‌های دیگر درایورها را بر حسب این ویژگی‌ها پیاده‌سازی می‌کنند، به عنوان مثال. یک درایور شتاب سنج ممکن است به نمونه یک I2C یا SPI نیاز داشته باشد.

- این ویژگی‌ها با استفاده از سیستم جانبی(peripherals) پوشش می‌دهند، اما آن‌ها را مقداردهی اولیه پیکربندی نمی‌کنند، زیرا مقداردهی اولیه و پیکربندی معمولاً به پلتفرم خاص بستگی دارد.
- پیاده‌سازی‌های برای بسیاری از میکروکنترلرها و همچنین پلتفرم‌های دیگری مانند لینوکس در Raspberry Pi وجود دارد.
- برای embedded-hal-async نسخه‌های async از trait‌ها را ارائه می دهد.
- مورد embedded-hal-nb روی‌کرد دیگری را برای عدم مسدود کردن I/O ارائه می دهد که بر اساس [nb](https:// crates.io/crates/nb) crate است.

51.7 probe-rs and cargo-embed

probe-rs یک مجموعه ابزار مفید برای اشکال زدایی جاسازی شده است، مانند OpenOCD است، اما به‌تر یکپارچه شده است.

- (SWD (Serial Wire Debug و JTAG از طریق پروب های ST-Link، CMSIS-DAP و J-Link
- GDB stub و Microsoft DAP (Debug Adapter Protocol) server
- ادغام Cargo

buddy_system_allocator یک third-party crate است که یک تخصیص‌دهنده buddy system را پیاده‌سازی می‌کند. crateهای دیگر در دسترس هستند یا می‌توانید نسخه مربوط به خود را بنویسید یا به تخصیص دهنده موجود خود متصل کنید.

- CMSIS-DAP یک پروتکل است‌اندارد ARM از طریق USB است که برای یک دیباگر درون مداری چت دسترسی به پورت CoreSight Debug Access در انواع مختلف پردازنده‌های Arm Cortex مورد استفاده قرار گرفته و این همان چیزی است که دیباگر داخلی در BBC micro:bit از آن استفاده می‌کند.
- ST-Link طیفی از دیباگرهای درون مدار از ST Microelectronics است، J-Link محدوده ای از SEGGER است.
- پورت دسترسی Debug معمولا یا یک رابط JTAG 5 پین یا Serial Wire Debug 2 پین است.
- probe-rs یک کتابخانه است که در صورت تمایل می‌توانید آن را در ابزاره‌ای خود ادغام کنید.
- پروتکل آداپتور Debug مای‌کروسافت به VSCode و سایر IDEها اجازه می‌دهد کده‌ای موجود در هر میکروکنترلر پشتیبانی‌شده را Debug کنند.
- این cargo-embed یک بایناری است که با استفاده از کتابخانه probe-rs ساخته شده است.
- (RTT (Real Time Transfers مکانیزمی برای انتقال داده‌ها بین debug host و target از طریق تعدادی بافر حلقه‌ای (ringbuffers) است.

51.7.1 اشکال یابی (Debugging)

Embed.toml:

```
[default.general]
"chip = "nrf52833_xxAA

[debug.gdb]
enabled = true
```

در یک ترمینال تحت /src/bare-metal/microcontrollers/examples:

```
cargo embed --bin board_support debug
```

در ترمینال دیگری در همان دایرکتوری:

در gLinux یا Debian:

```
rch target/thumbv7em-none-eabihf/debug/board_support --eval-command="target remote :1337
```

در MacOS:

```
gdb target/thumbv7em-none-eabihf/debug/board_support --eval-command="target remote :1337
```

در GDB، اجرا کنید:

```
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 پروژه‌های دیگر

- RTIC
  - "همراه‌ی مبتنی بر وقفه بلادرنگ"
  - مدیریت منابع مشترک، ارسال پیام، زمان‌بندی تسک (task scheduling)، صف تایمر (timer queue)
- Embassy
  - اجرا کننده‌های `async` اولویت‌دار، تایمرها، شبکه، USB
- TockOS
  - RTOS مترکز بر امنیت با برنامه‌نویسی پیشگیرانه و پشتیبانی از واحد حفاظت از حافظه
- Hubris
  - یک Microkernel RTOS از شرکت Oxide Computer با protection از حافظه، درایورهای غیرمجاز و IPC
- Bindings for FreeRTOS
- برخی از پلتفرم‌ها پیاده‌سازی `std` دارند، به عنوان مثال. esp-idf.
- RTIC را می‌توان یک یا چارچوب همزمان (concurrency framework) در نظر گرفت.
  - این شامل هیچ HAL نیست.
  - از Cortex-M NVIC (کنترل‌کننده وقفه مجازی تودرتو-Nested Virtual Interrupt Controller) برای زمان‌بندی به جای یک هسته مناسب استفاده می‌کند.
  - Cortex-M فقط.
- گوگل از TockOS در میکروکنترلر Haven برای کلیدهای امنیتی Titan استفاده می‌کند.
- در واقع FreeRTOS بیشتر به زبان C نوشته شده است، اما رابط‌های Rust برای نوشتن برنامه‌ها در این حالت وجود دارد.

فصل 52

ا ه ن ی ر م ت

ا م چ ه ت را از ق ط ب ن م ا ی I2C م ی خ وا ن م و خ وا ن ه ش ا ه را در ی ک پ ورت س ر ی ال ث ب ت م ی ک ن م.

پ س از د ی د ن ا ه ن ی ر م ت ه ، ا ه ل ح ا ه ر ا] ب ه د ی وا ن ت م ی (solutions-morning.md) ا ر ا ئ ه ش د ه ن گ ا ه ک ن د ی.

52.1 ق ط ب ن م ا

ا م چ ه ت را از ق ط ب ن م ا ی I2C م ی خ وا ن م و خ وا ن ه ش ا ه را در ی ک پ ورت س ر ی ال ث ب ت م ی ک ن م. اگ ر وق ت دا ر ی د، س ع ی ک ن ی د آ ن را ب ه ن ح وی روی LEDا ه ز ی ن م ا ی ش د ه د ی ا ب ه ن و ع ی از د ک م ه ا ه ا س ت ف ا د ه ک ن ی د.

ن ک ت ه ا ه:

- م س ت ن د ا ت lsm303agr و microbit-v2 ب ه م ه ر ا ه crateا ی آ ن را ب ر ر س ر ی ک ن ی د ه م ا ن ط ور ک ه micro:bit hardware را ب ر ر س ر ی م ی ک ن ی د.
- و ا ح د ا ن د ا ز ه گ ی ر ی ا ی ن ر س ر ی د ر ق ط ع ه LSM303AGR ب ه bus د ا خ ل ی I2C م ت ص ل ا س ت.
- TWI ن ا م د ی گ ر ی ب ر ا ی I2C ا س ت، ب ن ا ب ر ا ی ن د س ت گ ا ه چ ا ن ب ی ا ص ل ی I2C TWIM ن ا م ی د ه م ی ش و د.
- د ر ا ی و ر LSM303AGR ب ه چ ی ز ی ن ی ا ز د ا ر د ک ه و ی ژ گ ی embedded_hal::i2c::I2c را ا ج ر ا ک ن د. س ا خ ت ا ر microbit::hal::Twim ا ی ن م و ر د را پ ی ا د ه س ا ز ی م ی ک ن د.
- ش م ا ی ک س ا خ ت ا ر 'microbit::Board' ب ا ف ی ل د ه ا ی ب ر ا ی پ ی ن ه ا و ت ج ه ی ز ا ت چ ا ن ب ی م خ ت ل ف د ا ر ی د.
- د ر ص ور ت ت م ا ی ل ی ا ی ن ت د ی وا ن ت م ی ب ه nRF52833 datasheet ن ی ز ن گ ا ه ک ن ی د، ا م ا ب ر ا ی ا ی ن ت م ر ی ن ل ا ز م ن ی س ت.

ا ی ن ا ل گ و ی ت م ر ی ن را دا ن ل و د ک ن ی د و ف ا ی ل ه ا ی ز ی ر را در د ا ی ر ک ت و ر ی compass ج س ت ج و ک ن ی د.

:src/main.rs

```
#![no_std]
#![no_main]

use panic_halt as _;

use cortex_m_rt::entry;
use core::fmt::Write;
use microbit::{hal::{Delay, uarte::{Baudrate, Parity, Uarte}}, Board};

#[entry]
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 (ازین بجز دادن رییغت):

```
[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 (ازین بجز رییغت این تست):

```
[default.general]
"chip = "nrf52833_xxAA

[debug.gdb]
enabled = true

[debug.reset]
halt_afterwards = true
```

(you shouldn't need to change this. *cargo/config.toml*):

```
[build]
target = "thumbv7em-none-eabihf" # Cortex-M4F

['(("target.'cfg(all(target_arch = "arm", target_os = "none]
["rustflags = ["-C", "link-arg=-Tlink.x
```

مشاهده خروجی سریال در لینوکس با:

```
picocom --baud 115200 --imap lfcrlf /dev/ttyACM0
```

یا در سیستم‌عامل Mac چیزی شبیه به (نام دستگاه ممکن است کمی متفاوت باشد):

```
picocom --baud 115200 --imap lfcrlf /dev/tty.usbmodem14502
```

برای خروج از picocom از Ctrl+A و Ctrl+Q استفاده کنید.

52.2 تمرین صبحگاهی Bare Metal Rust

امنطاق

(back to exercise)

```
#![no_std]
#![no_main]

extern crate panic_halt as _;

use cortex_m_rt::entry;
use core::fmt::Write;
use embedded_hal::digital::InputPin;
use lsm303agr::{
    AccelMode, AccelOutputDataRate, Lsm303agr, MagMode, MagOutputDataRate,
};
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;

#[entry]
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 راه اندازی" ,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.("آماده." ,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
                                                        {
                                                            {
                                                                {

                                           [(derive(Copy, Clone, Debug, Eq, PartialEq]#
                                                        } 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 بخش

Bare Metal: باب رصع

فصل 53

Application processors

تا این‌جا در مورد میکروکنترلرهای مانند سری Arm Cortex-M صحبت کردیم. حالا بیایید سعی کنیم چیزی برای Cortex-A بنویسیم. برای سادگی، ما فقط با برد 'virt' QEMU's aarch64' کار می‌کنیم.

- به طور کلی، میکروکنترلرها دارای MMU یا چندین سطح دسترسی (سطوح استثنا در پردازنده‌های Arm، حلقه‌ها در x86) نیستند، در حالی که پردازنده‌های برنامه دارای دسترسی هستند.
- QEMU از شبیه‌سازی ماشین‌های مختلف یا مدل‌های برد مختلف برای هر معماری پشتیبانی می کند. برد 'virt' با هیچ سخت‌افزار واقعی خاصی مطابقت ندارد، اما صرفا برای ماشین‌های مجازی طراحی شده است.

53.1 آماده شدن برای Rust

قبل از این‌که بتوانیم اجرای کد Rust را شروع کنیم، باید مقداری مقداردهی اولیه را انجام دهیم.

```
"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
```

- این همان چیزی است که برای C وجود دارد: مقداردهی اولیه وضعیت پردازنده، صفر کردن BSS و تنظیم stack pointer.
  - این BSS (نماد شروع بلوک، به دلایل تاریخی) بخشی از object file است که حاوی مقدارهای تخصیص یافته استاتیکی است که مقدار اولیه آن‌ها صفر است. برای جلوگیری از اتلاف

فضا روی صفر، آن‌ها از تصویر حذف شده‌اند. کامپایلر فرض می‌کند که لود از صفر کردن آن‌ها مراقبت می‌کند.
- ممکن است BSS قبلاً صفر شده باشد، بسته به این‌که چگونه حافظه مقداردهی اولیه شده و تصویر بارگذاری شده است، اما برای اطمینان آن را صفر می‌کنیم.
- قبل از خواندن یا نوشتن هر حافظه با MMU و cache را فعال کنیم. اگر این کار را نکنیم:
  - دسترسی‌های بدون تراز خطا خواهند داشت. ما کد Rust را برای هدف `aarch64-unknown-none` می‌سازیم که `+strict-align` را تنظیم می‌کند تا از ایجاد دسترسی‌های بدون تراز توسط کامپایلر جلوگیری کند، بنابراین در این مورد باید خوب باشد، اما لزوماً این‌طور نیست.
  - اگر در VM اجرا می‌شد، این کار می‌تواند منجر به مشکلات زیر شود. مشکل این است که VM مستقیماً با حافظه cache غیرفعال شده به حافظه دسترسی پیدا می‌کند، در حالی که host دارای نام مستعار قابل cache برای همان حافظه است. حتی اگر cache به طور صریح به حافظه دسترسی نداشته باشد، دسترسی‌های گمانه‌زنی می‌توانند منجر به پر شدن حافظه cache شود، و پس از پاک شدن حافظه cache یا فعال کردن حافظه توسط VM، تغییرات از یک یا دیگری از بین می‌رود. (حافظه cache با آدرس فیزیکی کد می‌خورد، نه VA یا IPA.)
- برای سادگی، ما فقط از یک pagetable کدگذاری شده استفاده می کنیم (به `idmap.S` مراجعه کنید) که ۱ گیگابایت اول فضای آدرس را برای دستگاه‌ها، ۱ گیگابایت بعدی را برای DRAM و ۱ گیگابایت دیگر را برای دستگاه‌های بیشتر نگاشت می‌کند. این با چیدمان حافظه‌ای که QEMU استفاده می‌کند مطابقت دارد.
- ما همچنین (exception vector (`vbar_el1`) را تنظیم کردیم که در ادامه بیشتر در مورد آن خواهیم دید.
- همه مثال‌ها امروز بعد از ظهر فرض می‌کنند که ما در سطح استثنا 1 (EL1) اجرا خواهیم کرد. اگر نیاز به اجرا در سطح استثنای متفاوت دارید، باید `entry.S` را بر این اساس تغییر دهید.

Inline assembly 53.2

گاهی اوقات برای انجام کاره‌ایی که با کد Rust امکان پذیر نیست، باید از اسمبلی استفاده کنیم. به عنوان مثال، برای برقراری یک (HVC (hypervisor call نیاز است که به firmware بگوییم سیستم را خاموش کند:

```
[no_main]!#
 [no_std]!#

;use core::arch::asm
;use core::panic::PanicInfo

;mod exceptions

;const PSCI_SYSTEM_OFF: u32 = 0x84000008

[no_mangle]#
} (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
                                                {
```

(اگر واقعاً می‌خواهید این کار را انجام دهید، از crate مربوطه `smccc` استفاده کنید که دارای بسته‌بندی(wrapper) برای همه این عملکردها است.)

- PSCI یک رابط دهی‌تگر Arm Power State است که مجموعه‌ای است‌ندارد از توابع برای مدیریت وضعیت‌های power در سیستم و CPU بوده، از جمله موارد دیگری از این مورد توسط می‌افزار EL3 و hypervisor در بسیاری از سیستم‌ها پیاده سازی شده است.
- یک `0 <= _` syntax به این معنی است که رجیستر را قبل از اجرای کد اسمبلی درون خطی به 0 مقداردهی کنید و پس از آن محتوای آن را نادیده بگیرید. ما باید از `inout` به جای `in` استفاده کنیم زیرا این فراخوانی به طور بالقوه می‌تواند محتویات رجیسترها را مخدوش کند.
- این تابع `main` باید به صورت`#[no_mangle]` و `"extern "C"` باشد زیرا از نقطه ورودی (entry point) ما در `entry.S` فراخوانی می‌شود.
- `x0_x3_` مقادیر رجیسترهای `x0–x3` هستند که به طور معمول توسط bootloader برای ارسال چیزهایی مانند اشاره‌گر به device tree استفاده می‌شود. طبق قرارداد فراخوانی استاندارد aarch64 (که همان چیزی است که `"extern "C"` برای استفاده مشخص می‌کند)، رجیسترهای `x0–x7` برای ۸ آرگومان اول ارسال شده به یک تابع استفاده می‌شوند، بنابراین `entry.S` این کار را انجام نمی‌دهد. لازم نیست کار خاصی انجام دهید، جز این‌که مطمئن شوید که این مورد رجیسترها را تغییر نمی‌دهد.
- مثال را در QEMU با `make qemu_psci` در زیر `src/bare-metal/aps/examples` اجرا کنید.

53.3 دسترسی به حافظه فرار برای MMIO

- از `pointer::read_volatile` و `pointer::write_volatile` استفاده کنید.
- هرگز referenceای را نگه ندارید.
- `addr_of!` به شما امکان می‌دهد بدون ایجاد یک مرجع میانی، فیلدهایی از ساختارها را دریافت کنید.
- دسترسی فرار (Volatile access): عملیات خواندن یا نوشتن ممکن است عوارض جانبی داشته باشد، بنابراین از کامپایلر یا سیستم‌افزار از مرتب‌سازی مجدد، کپی‌کردن یا حذف آن‌ها جلوگیری کنید.
  - معمولاً اگر بنویسید و سپس بخوانید، به عنوان مثال. از طریق یک reference، کامپایلر ممکن است فرض کند که مقدار خوانده شده همان مقداری است که نوشته شده است و در واقع خواندن memory را سخت‌تر نکند.
- برخی از crateهای موجود برای دسترسی فَرار(volatile access) به سخت‌افزار دارای referenceهایی هستند، اما این همیشه درست نیست. هر زمان که یک reference وجود داشته باشد، کامپایلر ممکن است انتخاب کند که reference آن را لغو کند.
- از ماکرو `addr_of!` برای دریافت اشاره‌گرهای struct field از یک اشاره‌گر به ساختار استفاده کنید.

53.4 بی‌ای‌دی یک درایور UART بنویسیم

این ماشین 'QEMU 'virt یک PL011 به‌عنوان UART دارد، پس بی‌ای‌دی یک درایور برای آن بنویسیم.

```
const FLAG_REGISTER_OFFSET: usize = 0x18;
const FR_BUSY: u8 = 1 << 3;
const FR_TXFF: u8 = 1 << 5;

/// Minimal driver for a PL011 UART.
#[derive(Debug)]
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() }
    }
}
```

- توجه داشته باشید که `Uart::new` ناامن است unsafe در حالی که متد‌های دیگر این‌من هستند. این به‌خاطر این است که تا زمانی که تماس گیرنده `Uart::new` تضمین نکند که الزامات این‌منی

آن برآورده شده است (یعنی فقط یک نمونه از درایور برای یک UART مشخص وجود دارد و هیچ چیز دیگری نام مستعار فضای آدرس آن را ندارد)، پس همیشه می‌توان `write_byte` را بعداً فراخوانی کرد زیرا می‌توانیم پیش‌شرط‌های لازم را فرض کنیم.

- ما می‌توانیم این کار را به صورت دیگری انجام دهیم (ساخت `new` را ایمن کنیم، اما `write_byte` را ناامن کنیم)، اما استفاده از آن بسیار راحت‌تر خواهد بود، زیرا هر مکانی که `write_byte` را صدا می‌زند باید در مورد ایمنی یا safety استدلال کند.
- 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.

53.4.1 traitهای بیشتر

ما ویژگی Debug را استخراج کردیم. اجرای چند ویژگی دیگر زیر مفید خواهد بود.

```
;{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
```

- پیاده‌سازی Writeبه ما امکان می‌دهد از ماکروهای write! و writeln! با تایپ Uart خود استفاده کنیم.
- مثال را در QEMU با make qemu_minimal در زیر src/bare-metal/aps/examples اجرا کنید.

53.5 یک درایور UART بهتر

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.

افست	نام رجیستر	عرض
0x00	DR	12
0x04	RSR	4
0x18	FR	9
0x20	ILPR	8
0x24	IBRD	16
0x28	FBRD	6
0x2c	LCR_H	8
0x30	CR	16
0x34	IFLS	6
0x38	IMSC	11

عرض	نام رجیستر	افست
11	RIS	0x3c
11	MIS	0x40
11	ICR	0x44
3	DMACR	0x48

- همچنین برخی از ID register های وجود دارد که برای اختصار حذف شده‌اند.

53.5.1 پرچم‌های بیتی (Bitflags)

این crate برای bitflags چه‌ت کار با bitflags مفید است.

```
use bitflags::bitflags;

bitflags! {
    /// Flags from the UART flag register.
    #[repr(transparent)]
    #[derive(Copy, Clone, Debug, Eq, PartialEq)]
    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;
    }
}
```

- ماکرو `bitflags!` یک نوع جدید چیزی مانند `Flags(u16)` را به همراه تعدادی پیاده‌سازی متد برای دریافت و تنظیم flagها ایجاد می‌کند.

53.5.2 رجیستر چندگانه

ما می‌توانیم از یک ساختار برای نمایش طرح memory layout یک رجیستر UART استفاده کنیم.

```
#[repr(C, align(4))]
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)]` به کامپایلر می گوید که فیلدهای ساختاری را به ترتیب قرار دهد و این پیروی از قوانین مشابه با زبان C است. این کار را برای ساختار ما برای داشتن یک طرح قابل پیش‌بینی ضروری است، زیرا نمایش پیش‌فرض Rust به کامپایلر اجازه می دهد تا (از جمله موارد دیگر) فیلدها را به هر نحوی که صلاح بداند مرتب کند.

53.5.3 درایور

حال بیایید از ساختار جدید `Registers` در درایور خود استفاده کنیم.

```
/// Driver for a PL011 UART.
#[derive(Debug)]
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
                                                                    {
                                                                       {
```

- به استفاده از addr_of! / addr_of_mut! برای دریافت pointerها به فیلدهای جداگانه بدون ایجاد یک reference می‌توانی توجه کنید، که ممکن است نادرست باشد.

53.5.4 با استفاده از آن

یک دیباگر کوچک همراه با کرنل روی QEMU می‌نویسیم و از درایور خود استفاده می‌کنیم تا ورودی را echo کنیم.

```
#![no_main]
#![no_std]
```

```
;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

[no_mangle]#
} (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
{
```

- همان‌طور که در مثال [inline assembly](../inline-assembly.md)، این تابع `main` از کد نظیر ورودی ما در `entry.S` فراخوانی می‌شود. برای جزئیات بیشتر، رشته‌های ساختاریافته را در آنجا ببینید.
- مثال را در QEMU با `make qemu` در زیر `src/bare-metal/aps/examples` اجرا کنید.

53.6 لاگ

خوب است که بتوانید از ماکروهای logging از crate `log` استفاده کنید. ما می‌توانیم این کار را با اجرای ویژگی `Log` انجام دهیم.

```
;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
                                                        {
```

- باز کردن درlog ای‌من است زی‌را LOGGER را قبل از فراخوانی set_logger مقداردهی اولیه می‌کنیم.

53.6.1 با استفاده از آن

قبل از استفاده از لاگر باید مقداردهی اولیه کنیم.

```
[no_main]!#
 [no_std]!#

;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

[no_mangle]#
} (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
{

[panic_handler]#
} ! <- (fn panic(info: &PanicInfo
;("{error!("{info
;()system_off::<Hvc>().unwrap
{} loop
{
```

- توجه داشته باشید که panic handler ما اکنون می تواند جزئیات panic را ثبت کند.
- مثال را در QEMU با `make qemu_logger` در زیر `src/bare-metal/aps/examples` اجرا کنید.

53.7 استثناها

AArch64 یک جدول برداری استثنای با ۱۶ ورودی، برای ۴ نوع استثنا (synchronous، IRQ، FIQ، SError) از ۴ حالت (Current EL with SP0, current EL with SPx, lower EL using AArch64, lower EL using AArch32) تعریف می‌کند. ما این کار را در اسمبلی پیاده‌سازی می‌کنیم تا رجیسترهای فرار (volatile) را قبل از فراخوانی Rust در stack ذخیره کنیم:

```
;use log::error
;use smccc::psci::system_off
;use smccc::Hvc

[no_mangle]#
} (extern "C" fn sync_exception_current(_elr: u64, _spsr: u64
;("error!("sync_exception_current
;()system_off::<Hvc>().unwrap
{

[no_mangle]#
} (extern "C" fn irq_current(_elr: u64, _spsr: u64
;("error!("irq_current
;()system_off::<Hvc>().unwrap
```

```
                                                {

                                            [no_mangle]#
} (extern "C" fn fiq_current(_elr: u64, _spsr: u64
                                    ;("error!("fiq_current
                                ;()system_off::<Hvc>().unwrap
                                                {

                                            [no_mangle]#
} (extern "C" fn serr_current(_elr: u64, _spsr: u64
                                    ;("error!("serr_current
                                ;()system_off::<Hvc>().unwrap
                                                {

                                            [no_mangle]#
  } (extern "C" fn sync_lower(_elr: u64, _spsr: u64
                                    ;("error!("sync_lower
                                ;()system_off::<Hvc>().unwrap
                                                {

                                            [no_mangle]#
   } (extern "C" fn irq_lower(_elr: u64, _spsr: u64
                                    ;("error!("irq_lower
                                ;()system_off::<Hvc>().unwrap
                                                {

                                            [no_mangle]#
   } (extern "C" fn fiq_lower(_elr: u64, _spsr: u64
                                    ;("error!("fiq_lower
                                ;()system_off::<Hvc>().unwrap
                                                {

                                            [no_mangle]#
  } (extern "C" fn serr_lower(_elr: u64, _spsr: u64
                                    ;("error!("serr_lower
                                ;()system_off::<Hvc>().unwrap
                                                {
```

- EL سطح استثنا است. تمام نمونه‌های ما امروز بعدازظهر در EL1 اجرا می‌شوند.
- برای سادگی، ما بین SP0 و SPx برای استثناهای EL فعلی، یا بین AArch32 و AArch64 برای استثناهای پایین EL تمایز قائل نمی‌شویم.
- برای این مثال، ما فقط exception را log کرده و سپس خاموش می‌کنیم، زیرا نداشتن هیچ اطلاعاتی از آنچه واقعاً اتفاق افتاد.
- 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 پروژه‌های دیگر

- oreboot
  - "coreboot without the C"
  - پشتیبانی از x86، aarch64 و RISC-V.
  - به جای این‌که خود درایورهای زیادی داشته باشد، به LinuxBoot متکی است.
- Rust RaspberryPi OS tutorial
  - راه‌اندازی، درایور UART و bootloader ساده، JTAG، سطوح exception، مدیریت exception و page tableها
  - برخی ابهامات در مورد نگه‌داری کَش و راه‌اندازی اولیه در Rust، لزوماً مثال خوبی برای کپی کردن برای کد production نیست.
- `cargo-call-stack`
  - تجزیه‌وتحلیل استاتیک برای تعیین حداکثر استفاده از stack.
- آموزش سیستم عامل RaspberryPi، کد Rust را قبل از فعال شدن MMU و حافظه کَش اجرا می‌کند. این کار memory را می‌خواند و روی آن می‌نویسد (به عنوان مثال stack). بااین‌حال:
  - 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.

فصل 54

جعبه‌های (crates) دربارک

ما به چند crate می‌پردازیم که برخی از مشکلات رایج در برنامه‌نویسی bare-metal را حل می‌کنند.

54.1 zerocopy

این crate zerocopy (از Fuchsia) صفات و ماکروهایی را برای تبدیل ایمن بین بایت‌ها و انواع دیگر فراهم می‌کند.

```
use zerocopy::AsBytes;

#[repr(u32)]
#[derive(AsBytes, Debug, Default)]
enum RequestType {
    #[default]
    In = 0,
    Out = 1,
    Flush = 4,
}

#[repr(C)]
#[derive(AsBytes, Debug, Default)]
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]&
;(
{
```

ای‌ن برای MMIO م‌ناس‌ب نی‌ست (زی‌را از خ‌واندن و ن‌وشت‌ن فرار ی‌ volatile اس‌ت‌فاده ن‌می‌ک‌ند)، اما می‌-ت‌وان‌د برای کار با س‌اخ‌تاره‌ای مش‌ترک با س‌خت افزار مف‌ید باش‌د. ت‌وس‌ط DMA، ی‌ا از طری‌ق برخی از راب‌ط‌ه‌ای خارجی ارس‌ال می‌ش‌ود.

- `FromBytes` را می‌ت‌وان برای ان‌واعی ک‌ه ه‌ر الگ‌وی بای‌تی برای آن‌ها ع‌م‌ل‌ت اس‌ت پی‌اده‌س‌ازی ک‌رد و ب‌ن‌اب‌رای‌ن می‌ت‌وان با خ‌ی‌ال راح‌ت از ی‌ک دن‌ب‌ال‌ه بای‌ت‌ه‌ای نام‌ع‌ت‌ب‌ر ت‌ب‌دی‌ل ک‌رد.
- ت‌اب‌ع `FromBytes` برای ای‌ن ت‌ای‌پ‌ه‌ا نام‌وف‌ق خ‌واه‌د ب‌ود، زی‌را `RequestType` از ه‌م‌ه م‌ق‌دی‌ر م‌م‌ک‌ن `u32` ب‌ه ع‌ن‌وان م‌ت‌ام‌ی‌زک‌ن‌ن‌ده اس‌ت‌فاده ن‌می‌ک‌ند، ب‌ن‌اب‌رای‌ن ه‌م‌ه ال‌گ‌وه‌ای بای‌ت ع‌ت‌ب‌ر ن‌ی‌س‌ت‌ند.
- `zerocopy::byteorder` دارای ت‌ای‌پ‌ه‌ای برای اع‌داد اول‌ی‌ه مط‌ع از byte-order اس‌ت.
- م‌ث‌ال را با `cargo run` در `/src/bare-metal/useful-crates/zerocopy-example` اج‌را ک‌ن‌ی‌د. (ب‌ه دل‌ی‌ل واب‌س‌ت‌گی ب‌ه crate در Playground اج‌را ن‌می‌ش‌ود.)

aarch64-paging 54.2

ای‌ن aarch64-paging crate ب‌ه ش‌م ام‌کان می‌ده‌د page tableه‌ا را مط‌ابق با ع‌م‌ار س‌ی‌س‌ت‌م ح‌اف‌ظه-م‌ج‌ازی AArch64 ای‌ج‌اد ک‌ن‌ی‌د.

```
}::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
```

- در ح‌ال ح‌اض‌ر ف‌ق‌ط از EL1 پ‌ش‌ت‌ی‌ب‌ان‌ی می‌ک‌ند، اما پ‌ش‌ت‌ی‌ب‌ان‌ی از س‌ای‌ر س‌ط‌وح اس‌ت‌ث‌ن‌ا ب‌ای‌د س‌اده ب‌اش‌د.
- ای‌ن م‌ورد در Android ب‌رای [m/android/platform/superproject/+/master:packages/modules/Virtualization/pvmfw] اس‌ت‌فاده می‌ش‌ود.
- ه‌ی‌چ راه آس‌ان‌ی ب‌رای اج‌رای ای‌ن م‌ث‌ال وج‌ود ن‌دارد، زی‌را ب‌ای‌د روی س‌خت‌اف‌زار واق‌ع‌ی ی‌ا ت‌ح‌ت QEMU اج‌را ش‌ود.

buddy_system_allocator 54.3

'buddy_system_allocator' یک third-party crate است که یک buddy system allocator را پیاده‌سازی می‌کند. می‌توان آن را هم برای 'LockedHeap' در پیاده‌سازی [GlobalAlloc](https://doc.rust-lang.org/core/alloc/trait.GlobalAlloc.html) استفاده کرد. بنابراین می‌توانید از crate استاندارد alloc (همان‌طور که قبل از دیدیم) یا برای تخصیص فضای آدرس دیگر استفاده کنید. به عنوان مثال، ممکن است بخواهیم فضای MMIO را برای PCI BARهای اختصاص دهیم:

```
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 BARهای همه‌شان دارای تراز برابر با اندازه خودشان هستند.
- مثال را با `cargo run` در `/src/bare-metal/useful-crates/allocator-example` اجرا کنید. (به دلیل وابستگی به crate در Playground اجرا نمی‌شود.)

tinyvec 54.4

گاهی اوقات شما چیزی را می‌خواهید که بتوان آن را مانند Vec تغییر اندازه داد، اما بدون heap allocation که [tinyvec](https://crates.io/crates/tinyvec) این را فراهم می‌کند: یک برداری که توسط یک آرایه یا برش پشتیبانی می‌شود که می‌توانند به‌صورت ایستا allocate داده شود یا روی stack که تعداد عناصر استفاده شده را ردیابی می‌کند و اگر سعی کنید بیشتر از آنچه که اختصاص‌داده‌شده را استفاده کنید panic می‌کند.

```
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 نیاز دارد که تایپ عنصر را برای مقداردهی اولیه `Default` اجرا کند.
- Rust Playground شامل tinyvec می‌شود، بنابراین این مثال به‌خوبی به صورت داخلی اجرا می‌شود.

spin 54.5

`std::sync::Mutex` و دیگر موارد اولیه همگام‌سازی از `std::sync` در `core` یا `alloc` موجود نیستند. چگونه می‌توانیم همان‌گونه سازی یا تغییرپذیری داخلی، مانند اشتراک‌گذاری وضعیت بین CPUهای مختلف را مدیریت کنیم؟

این crate `spin` عداله‌ای مبتنی بر spinlock، بسیاری از این موارد اولیه را ارائه می‌کند.

```
use spin::mutex::SpinMutex;

static counter: SpinMutex<u32> = SpinMutex::new(0);

fn main() {
    println!("count: {}", counter.lock());
    counter.lock() += 2;
    println!("count: {}", counter.lock());
}
```

- اگر در handlerهای وقفه قفل می‌کنید مراقب باشید که از بن بست(deadlock) جلوگیری کنید.
- `spin` همچنین دارای اجرای ticket lock mutex است. عداله‌ای `RwLock`, `Barrier` و `Once` از `std::sync`. و `Lazy` برای مقداردهی اولیه lazy.
- این crate 'once_cell' همچنین دارای تایپه‌ای مفیدی برای مقداردهی اولیه دیرهنگام با روی‌کرد کمی متفاوت به `spin::once::Once` است.
- Playground Rust شامل `spin` است، بنابراین این مثال به خوبی به صورت داخلی اجرا می‌شود.

فصل 55

دیوردان

برای ساخت‌ن یک bare-metal Rust binary در AOSP، باید از یک Soong `rust_ffi_static` برای ساخت کد Rust خود استفاده کنید، سپس از یک `cc_binary` با یک linker script برای تولید binary استفاده کرده و سپس از یک `raw_binary` برای تبدیل ELF به یک raw binary آماده اجرا استفاده کنید.

```
                                } 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

```
                                                    [no_main]!#
                                                     [no_std]!#

                                                ;{use vmbase::{main, println

                                                        ;(main!(main

} (pub fn main(arg0: u64, arg1: u64, arg2: u64, arg3: u64
                                                ;("println!("Hello world
                                                        {
```

- این ماکرو main!عملکرد اصلی شما را مشخص می‌کند تا از نقطه ورودی vmbase فراخوانی شود.
- نقطه ورودی vmbase مقدار دهی اولیه کنسول را کنترل می‌کند و در صورت بازگشت main function، یک PSCI_SYSTEM_OFF برای خاموش کردن VM صادر می‌کند.

فصل 56

اینرمت

ما یک درایور برای دستگاه PL031 real-time clock خواهیم نوشت.

پس از بررسی تمرین‌ها، می‌توانید به راه‌حل‌ها ارائه شده نگاهی بیندازید.

RTC driver 56.1

The QEMU aarch64 virt machine has a PL031 real-time clock at 0x9010000. For this exercise, you should write a driver for it.

1. از آن برای چاپ زمان جاری در کنسول سریال استفاده کنید. می‌توانید از crate `chrono` برای قالب‌بندی date/time استفاده کنید.
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. افزونه‌ها اگر زمان دارید: وقفه ایجاد شده توسط تطبیق RTC را فعال کرده و آن را مدیریت کنید. می‌توانید از درایور ارائه شده در crate `arm-gic` برای پیکربندی Arm Generic Interrupt Controller استفاده کنید.
   - از وقفه RTC استفاده کنید که به عنوان `IntId::spi(2)` به GIC متصل است.
   - هنگامی که وقفه (interrupt) فعال شد، می‌توانید دستور `arm_gic::wfi()` را از طریق حالت Sleep درآورید، که باعث می‌شود دستور تا زمانی که وقفه دریافت کند به خواب برود.

دانلود از exercise template و فایل‌های زیر را در دایرکتوری `rtc` جستجو کنید.

src/main.rs:

```
#![no_std]
#![no_main]

mod exceptions;
mod logger;
mod pl011;

use arm_gic::gicv3::GicV3;
use core::panic::PanicInfo;
use log::{error, info, trace, LevelFilter};
use smccc::psci::system_off;
use crate::pl011::Uart;
```

```
;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

                                                                          [no_mangle]#
                              } (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
                                                                               {

                                                                  [panic_handler]#
                                                } ! <- (fn panic(info: &PanicInfo
                                                            ;("{error!("{info
                                                   ;()system_off::<Hvc>().unwrap
                                                                   {} loop
                                                                               {
```

src/exceptions.rs (می‌خواهد این بایت‌ها را برای شبیه‌سازی موس تغییر دهید):

```
                                                    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
                                           ;{use log::{error, info, trace
                                             ;use smccc::psci::system_off
                                                          ;use smccc::Hvc

                                                                [no_mangle]#
                        } (extern "C" fn sync_exception_current(_elr: u64, _spsr: u64
                                                ;("error!("sync_exception_current
                                                    ;()system_off::<Hvc>().unwrap
                                                                          {

                                                                [no_mangle]#
                                     } (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
                                                                          {

                                                                [no_mangle]#
                                      } (extern "C" fn fiq_current(_elr: u64, _spsr: u64
                                                           ;("error!("fiq_current
                                                    ;()system_off::<Hvc>().unwrap
                                                                          {

                                                                [no_mangle]#
                                    } (extern "C" fn serr_current(_elr: u64, _spsr: u64
                                                           ;("error!("serr_current
                                                    ;()system_off::<Hvc>().unwrap
                                                                          {

                                                                [no_mangle]#
                                     } (extern "C" fn sync_lower(_elr: u64, _spsr: u64
                                                         ;("error!("sync_lower
                                                    ;()system_off::<Hvc>().unwrap
                                                                          {

                                                                [no_mangle]#
                                      } (extern "C" fn irq_lower(_elr: u64, _spsr: u64
                                                         ;("error!("irq_lower
                                                    ;()system_off::<Hvc>().unwrap
                                                                          {

                                                                [no_mangle]#
                                      } (extern "C" fn fiq_lower(_elr: u64, _spsr: u64
                                                         ;("error!("fiq_lower
                                                    ;()system_off::<Hvc>().unwrap
                                                                          {
```

```
                                                    [no_mangle]#
                                } (extern "C" fn serr_lower(_elr: u64, _spsr: u64
                                                        ;("error!("serr_lower
                                                    ;()system_off::<Hvc>().unwrap
                                                                            {
```

src/logger.rs (دیده ریغت را درم نای تسین یزایز):

```
                                                    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
                                                                        {
```

```rust
/// 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 (نیازی نیست این مورد را تغییر دهید):

```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.

#![allow(unused)]

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.
    #[repr(transparent)]
    #[derive(Copy, Clone, Debug, Eq, PartialEq)]
    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 ///
                                                       [(repr(transparent]#
                                          [(derive(Copy, Clone, Debug, Eq, PartialEq]#
                                                  } 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 //
                                                      [((repr(C, align(4]#
                                                        } 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 ///
                                                        [(derive(Debug]#
                                                      } 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
```

Cargo.toml (دادن رﻳﻐﺗ ﻪﺑ زاﻳن):

```
                                                                [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
```

build.rs (دﻳده رﻳﻐﺗ رو درﻮﻣ ﻦﻳا ﺖﺴﻴﻧ زاﻳن):

```
                                                                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
                                                [("cfg(target_os = "linux]#
                                  ;("env::set_var("CROSS_COMPILE", "aarch64-linux-gnu
                                                [(("cfg(not(target_os = "linux]#
                                   ;("env::set_var("CROSS_COMPILE", "aarch64-none-elf

                                                                ()Build::new
                                                          ("file("entry.S.
                                                       ("file("exceptions.S.
                                                            ("file("idmap.S.
                                                         ("compile("empty.
                                                                          {
```

entry.S (نیازی نیست این مورد را تغییر دهید):

```
                                                                         */
                                                           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 *
mage 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

1:  /* .Prepare the stack */
    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 */
2:  wfi
    b 2b
```

exceptions.S (نازی نیست این مورد را دیده بگیرید):

```
/*
 * 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 (نیازی نیست این مورد را تغییر دهید):

```
*/
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 (نازی‌ن ای تسی‌ن درومه را رغی‌ر ده‌ید):

```
                                                                            */
                                                  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.)*
                                                                   {
                                                                       {
```

Makefile (این تنظیمات را در صورت نیاز تغییر دهید):

```
                                                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
```

.دنک یارجا make qemu اب QEMU رد ار دک

Bare Metal Rust اب‌رهظادعب 56.2

RTC driver

(back to exercise)

:*main.rs*

```
                                                                  [no_main]!#
                                                                   [no_std]!#

                                                                ;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

                                                                    [no_mangle]#
                            } (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
                                                                {

                                                  [panic_handler]#
                                   } ! <- (fn panic(info: &PanicInfo
                                                ;("{error!("{info
                                   ;()system_off::<Hvc>().unwrap
```

```
                                                    {} loop
                                                          {
```

:pl031.rs

```
                                          ;{use core::ptr::{addr_of, addr_of_mut

                                                  [((repr(C, align(4]#
                                                    } 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 ///
                                                      [(derive(Debug]#
                                                      } 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 شخب

حبص :ینامزمه

فصل 57

مبحث به Concurrency در Rust خوش‌آمدید

زبان Rust به طور کامل از concurrency در سطح thread های سیستم‌عامل با استفاده از mutex ها و channel ها پشتیبانی می‌کند.

سیستم Rust راست نقش مهمی در تبدیل بسیاری از باگ‌های concurrency به باگ‌های زمان کامپایل ایفا می‌کند. این مورد اغلب به‌عنوان همزمانی بی‌پروا (*fearless concurrency*) شناخته می‌شود زیرا می‌توانید به کامپایلر برای اطمینان از صحت در زمان اجرا (runtime) اعتماد کنید.

برنامه زمانی

با احتساب استراحت‌های ۱۰ دقیقه‌ای، این جلسه باید حدود ۳ ساعت و ۲۰ دقیقه طول بکشد. شامل موارد زیر است:

بخش	مدت زمان
تردها	۳۰ دقیقه
کانال‌ها	۲۰ دقیقه
Send و Sync	۱۵ دقیقه
ناحیه‌های مشترک	۳۰ دقیقه
تمرین‌ها	۱ ساعت و ۱۰ دقیقه

- Rust به ما اجازه می‌دهد تا به ابزارهای همزمانی سیستم عامل دسترسی داشته باشیم: threadها، سازوکارهای همگام‌سازی و غیره.
- این سیستم تایپ به ما ایمنی لازم برای concurrency بدون هیچ ویژگی خاصی می‌دهد.
- همان ابزارهایی که به ما در دسترسی concurrent در یک thread واحد کمک می‌کنند (مانند یک تابع فراخوانی شده که ممکن است یک آرگومان را تغییر دهد یا مراجعی به آن را برای خواندن بعد ذخیره کند) ما را از مشکلات multi-threading نجات می‌دهند.

فصل 58

تردهها

این بخش بایدحدود ۳۰ دقیقه طول بکشد و شامل موارد زیر است:

اسلایدها	مدت زمان
تردهای ساده	۱۵ دقیقه
محدودهتردها	۱۵ دقیقه

58.1 تردهای ساده

threadهای Rust مانند threadها در زبانهای دیگر کار میکنند:

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

fn main() {
    for i in 0..10 {
        thread::spawn(|| {
            println!("{i} شمارندهthread: {thread}");
            thread::sleep(Duration::from_millis(5));
        });
    }

    thread::sleep(Duration::from_millis(5));
    println!("Main thread: {i}");
    for i in 0..5 {
        thread::sleep(Duration::from_millis(5));
    }
}
```

- ایجاد threadهای جدید به طور خودکار خاتمه برنامه را تا پایان main به تاخیر نمیاندازد.
- Thread panicها مستقل از یکدیگر هستند.
  - Panicها میتوانند payloadای را حمل کنند که میتوان آن را با «downcast_ref» باز کرد.

This slide should take about 15 minutes.

- Rust thread APIها خیلی متفاوت از موارد دیگر به نظر نمیرسند. که C++ یکی از آنها است.

- مثال را اجرا کنید.
  - زمان‌بندی 5 میلی‌ثانیه به اندازه‌ای سست هست که thread اصلی و spawned threadها عمدتاً همگام می‌مانند.
  - توجه داشته باشید که برنامه قبل از این‌که thread spawned به مقدار ۱۰ برسد به پایان می‌رسد!
  - این به خاطر است که انتهای main برنامه است و spawned threadها ایجاد شده باعث تداوم آن نمی‌شوند.
    * در مقابسه با std::thread/boost::thread ++C/pthreads اگر مطلوب باشد.
- چقدر باید صبر کنیم تا یک spawned thread تکمیل شود؟
- `thread::spawn` returns a `JoinHandle`. به سند نگاه کنید.
  - `JoinHandle` دارد `.join()` متد آن بلاکه‌ا.
- از «`let handle = thread::spawn(...)`» و بعد از «`handle.join()`» استفاده کنید تا منتظر بمانید تا thread تمام شود و شمارنده برنامه برابر با مقدار ۱۰ باشد.
- حالا اگر بخواهیم مقداری را برگردانیم چه؟
- دوباره به اسناد نگاه کنید:
  - `thread::spawn`'s closure returns `T`
  - `JoinHandle.join()` returns `thread::Result<T>`
- به کمک `Result` از «`handle.join()`» برای دسترسی به مقدار برگشتی استفاده کنید.
- خوب، مورد دیگر چطور؟
  - فعال‌سازی یک panic در یک thread. توجه شود که این مورد `main` panic نیست.
  - دسترسی به این panic payload. به‌ترین زمان برای پرداخت به این موضوع است `Any`.
- اکنون می‌توانیم مقادیر را از رشته‌ها برگردانیم! در مورد گرفتن ورودی‌ها چطور؟
  - چیزی را از طریق reference در بسته‌بندی thread ثبت کنید.
  - یک پیغام خطا نشان می‌دهد که باید آن را جابجا کنیم.
  - آن را به داخل متقل کنید، درنتیجه ما می‌توانیم محاسبه کنیم و سپس یک مقدار مشتق شده را برگردانیم.
- اگر بخواهیم قرض (borrow) بگیریم چطور؟
  - تابع Main در هنگام بازگشت threadهای فرزند را از بین می‌برد، اما تابع دیگری return می‌شود و آن‌ها را در حال اجرا می‌گذارد.
  - این کار می‌تواند منجر به stack استفاده پس از return شود که memory safety را نقض می‌کند!
  - چگونه از آن جلوگیری کنیم؟ صفحه بعدی را ببینید.

58.2 محدوده تردها

threadهای معمولی نمی‌توانند از محیط خود قرض (borrow) بگیرند:

```
use std::thread;

fn foo() {
    let s = String::from("سلام");

    thread::spawn(|| {
        println!("Length: {}", s.len());
    });
}
```

```
                                                ;({
                                                    {

                                                } ()fn main
                                                 ;()foo
                                                    {
```

به‌حرال, می‌توانید برای این مورد scoped thread ببینید:

```
                                                ;use std::thread

                                                } ()fn main
                                            ;("سلام")let s = String::from

                                                    } |thread::scope(|scope
                                                       } ||)scope.spawn
                                        ;(())println!("Length: {}", s.len
                                                    ;({
                                                        ;({
                                                            {
```

This slide should take about 13 minutes.

- دلیل آن این است که وقتی عبارت «thread::scope» کامل می‌شود، اتصال همه threadها تضمین می‌شود، بنابراین می‌توانند داده‌ای قرض را برگردانند.
- قوانین عادی قرض‌گیری Rust اعمال می‌شود: شما می‌توانید به صورت تغییرپذیر(mutable) یا یک thread ای تغییرناپذیر (immutable) با هر تعداد thread قرض (borrow) بگیرید.

فصل 59

کانال‌ها

این بخش بای حدود ۲۰ دقیقه طول بکشد. این شامل:

اسلاید	مدت زمان
Senders و Receivers	۱۰ دقیقه
کانال‌های نامحدود	۲ دقیقه
کانال‌های محدود	۱۰ دقیقه

Senders و Receivers 59.1

کانال‌های Rust دارای دو بخش هستند: `Sender<T>` و `Receiver<T>`. این دو بخش از طریق channel به هم متصل می‌شوند، اما شما فقط نقاط پایانی (end-points) را می‌بینید.

```
use std::sync::mpsc;

fn main() {
    let (tx, rx) = mpsc::channel();

    tx.send(10).unwrap();
    tx.send(20).unwrap();

    println!("دریافت شد: {:?}", rx.recv());
    println!("دریافت شد: {:?}", rx.recv());

    let tx2 = tx.clone();
    tx2.send(30).unwrap();
    println!("دریافت شد: {:?}", rx.recv());
}
```

This slide should take about 9 minutes.

- `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 کانال‌های دودحمان

ش‌ام یک کانال دودحمان و همزمان‌ها با `mpsc::channel()` درفایل می‌نویسد:

```
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:?}: شد خارج");
    });

    for msg in rx.iter() {
        println!("Main: got {msg}");
    }
}
```

59.3 کانال‌های دودحم

با کانال‌های synchronous (bounded)، فرستنده می‌تواند `send` را فعلی `thread` دودسم کند:

```
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:?}: شد خارج");
    });

    for msg in rx.iter() {
        println!("Main: got {msg}");
    }
}
```

.This slide should take about 8 minutes

- فراخوانی `send` تا زمانی که فضای کافی در کانال برای پیام جدید وجود داشته باشد، thread کنونی را مسدود می‌کند. اگر کسی از کانال چیزی نخواند آنگاه thread را می‌توان به طور نامحدود مسدود کرد.
- اگر کانال بسته شود، تماس برای `send` با یک error قطع می‌شود (به همین دلیل `Result` را برمی‌گردانند. هنگامی که گیرنده از بین می رود یک کانال بسته می‌شود.
- یک کانال محدود (bounded channel) با اندازه صفر را "کانال ملاقات" یا "rendezvous channel" می‌نامند. هر ارسال، thread فعلی را مسدود می‌کند تا زمانی که رشته دیگری `recv` را فراخواند.

فصل 60

Send و Sync

این بخش ۱۵ دقیقه زمان می برد. این بخش شامل:

مدت زمان	اسلایدها
۲ دقیقه	ویژگی‌های نشانگر
۲ دقیقه	Send
۲ دقیقه	Sync
۱۰ دقیقه	مثال‌ها

ویژگی‌های نشانگر 60.1

Rust چگونه می‌داند که دسترسی مشترک در سراسر threadها را مدیریت می‌کند؟ پاسخ در دو trait است:

- Send: در صورتی که جابجایی یک T در مرز thread boundary ایمن باشد، تایپ T از جنس Send است.
- Sync: در صورتی که جابجایی یک &T در سراسر یک thread boundary ایمن باشد، یک تایپ T از جنس Sync است.

Send و Sync [ویژگی‌های نامن] هستند (unsafe-rust/unsafe-traits.md/../). کامپایلر به‌طور خودکار آن‌ها را برای تایپ‌های شما مشتق می‌کند تا زمانی که فقط دارای انواع Send و Sync باشند. شما همچنین می توانید آن‌ها را به صورت دستی پیاده سازی کنید به‌خصوص زمانی که می دانید مقدار آن مربوطه است.

This slide should take about 2 minutes.

- می‌توان این traitها را به عنوان نشانگرهایی در نظر گرفت که نوعی ویژگی thread-safety خاص را دارد.
- آن‌ها را می‌توان در محدودیت‌های generic به عنوان trait عادی استفاده کرد.

Send 60.2

اگر انتقال مقدار T به thread دیگری امن باشد، تایپ T در Send است.

تأثیر انتقال مالکیت (moving ownership) به یک thread دیگر این است که نابودگرها (*destructors*) در آن thread اجرا می شوند. بنابراین سوال این است که چه زمانی می‌توانید یک مقدار را در یک thread تخصیص دهید و آن را در thread دیگر توزیع کنید.

.This slide should take about 2 minutes

به عنوان مثال، اتصال به کتابخانه SQLite فقط بایـد از یک thread قابل دسترسی باشد.

60.3 Sync

یک تایپ T در واقع نوعی Sync است، اگر در دسترسی به یک مقدارT از طریق چندین رشته به طور همزمان امن باشد.

به طور دقیقتر، تعریف این طور است:

T یک نوع Sync فقط و فقط زمانی که &T یک نوع Send باشد

.This slide should take about 2 minutes

این عبارت به طور کلی روشی مختصر برای گفتن این است که اگر یک تایپ برای استفاده مشترک امن باشد، انتقال ارجاعات (pass references) آن به threadها نیز امن است.

این به خاطراست که اگر یک تایپ از جنس Sync باشد، به این معنی است که میتوان آن را در چند thread بدون خطر در مورد وضعیت رقابتی داده ای سایر مشکلات Sync به اشتراک گذاشت، بنابراین انتقال آن به threadای دیگر امن است. ارجاع به تایپ نیز برای انتقال به threadای دیگر ایمن است، زیرا دادههایی که به آن ارجاع میدهد میتوانند از هر threadای با خیال راحت دسترسی داشته باشند.

60.4 مثالها

Send + Sync

اکثر انواعی که با آنها روبرو می شوید Send + Sync هستند:

- ... ,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>:به طور صریح از طریق تعداد شمارش atomic reference با thread-safe.
- <mpsc::Sender<T>: از 1.72.0.
- AtomicBool، AtomicU8، ...: از دسترالعملهای atomic ویژه استفاده میکنند.

در صورت وجود پارامترهای نوع، تایپهای generic معمولاً از نوع Send + Sync هستند.Send + Sync.

Send + !Sync

این تایپها را میتوان به رشتههای دیگر منتقل کرد، اما آنها ایمن نیستند. به طور معمول به دلیل تغییرپذیری داخلی(interior mutability):

- <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!

این تایپ‌ها از نظر thread ایمن نیستند و نمی توان آن‌ها را به رشته های دیگر منتقل کرد:

- "Rc": هر "Rc" دارای یک ارجاع به "RcBox" است که حاوی تعداد مراجع غیر atomic است.
- درمورد `*mut T, *const T`: زبان فرض Rust می کند که اشاره گرهای خام ممکن است ملاحظات همزمانی خاصی داشته باشند.

فصل 61

کرتشم ایهیحان

این بخش دایاب حدود ۳۰ دقیقه طول بکشد و شامل موارد زیر است:

اسلایدِ	مدت زمان
Arc	۵ دقیقه
Mutex	۱۵ دقیقه
مثال	۱۰ دقیقه

Arc 61.1

<Arc<T اجازه می‌دهد تا دسترسی read-only مشترک از طریق Arc::clone صورت پذیرد:

```
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:?}");
}
```

This slide should take about 5 minutes.

- "Arc" مخفف "Atomic Reference Counted" است، یک نسخه ایمن از Rc که از عملیات atomic استفاده می‌کند.

- "Arc" به طور کلی "Clone" را خواه T انجام دهد یا نه، پیاده سازی می کند. Send و Sync را اگر و فقط در صورتی پیاده‌سازی می‌کند که T هر دوی آن‌ها را پیاده‌سازی کند.
- `Arc::clone()` هزینه یک عملیات atomic که اجرا می‌شود را دارد، اما پس از آن استفاده از 'T' آزاد است.
- مراقب reference cycleها باشید، Arc از garbage collector برای شناسایی آن‌ها استفاده نمی‌کند.
  - `std::sync::Weak` می‌تواند مفید باشد.

Mutex 61.2

`Mutex<T>` تضمین می‌کند که حذف متقابل و امکان دسترسی قابل تغییر (mutable) به "T" را در پشت یک read-only interface (شکل دیگری از تغییرپذیری (mutable) داخلی) فراهم می‌کند.

```
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());
}
```

توجه کنید که چگونه یک `impl<T: Send> Sync for Mutex<T>` برای اجرای کامل آن داریم.

This slide should take about 14 minutes.

- «Mutex» در Rust مانند مجموعه‌ای با تنها یک عنصر --- داده‌ای محافظت شده (protected) به نظر می‌رسد.
  - نمی‌توان قبل از دسترسی به داده‌ای محافظت شده با protected، دسترسی mutex را فراموش کرد.
- با گرفتن lock می‌توانید `mut T&` را از `Mutex<T>&` دریافت کنید.MutexGuard تضمین می‌کند که `mut T&` بیشتر از قفل نگه‌داشته شده، موجود نمی‌مانند.
- `Mutex<T>` هر دوی پیاده سازی Send و Sync iff (فقط و فقط) T از Send استفاده می‌کنند.
- همتای قفل خواندن و نوشتن: RwLock.
- چرا `lock()` یک Result برمی‌گرداند؟
  - اگر threadای که Mutex را نگه می‌دارد دچار panic شود، Mutex «مسموم/poisoned» می‌شود تا نشان دهد که داده‌ای که محافظت می‌کند ممکن است در وضعیت ناسازگاری باشند. فراخوانی `lock()` در یک mutex مسموم با یک [PoisonError»](https://doc.rust-lang.org/std/sync/struct.PoisonError.html) انجام نمی‌شود. می‌توانید `into_inner()` را در مورد خطا برای بازیابی داده‌ها بدون توجه به آن فراخوانی کنید.

مثال 61.3

اجازه دهید Arc و Mutex را در عمل ببینیم:

```rust
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:?}");
}
```

This slide should take about 8 minutes.

راه‌حل ممکن:

```rust
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:?}");
}
```

بخش‌های قابل توجه:

- `v` در `Arc` و `Mutex` احاطه می‌شود، زیرا مسائل آن‌ها شبیه به هم است.
  - قرار دادن یک `Mutex` در یک `Arc` یک الگوی رایج برای به اشتراک گذاشتن حالت قابل تغییر (mutable) بین threadها است.
- `v: Arc<_>` باید به عنوان `v2` کلون شود تا بتوان آن را به thread دیگری منتقل کرد. نکته move به lambda signature اضافه شد.
- بلوک‌ها برای محدود کردن دامنه `LockGuard` تا حد امکان معرفی شده‌اند.

فصل 62

امنیت

این بخش بایدحدود ۱ ساعت و ۱۰ دقیقه طول بکشد. این بخش شامل موارد زیر است:

مدت زمان	اسلاید
۲۰ دقیقه	فلسفه Dining
۲۰ دقیقه	جستجوگر پیوند چند تِردی
۳۰ دقیقه	راه حل‌ها

62.1 فلسفه Dining

مسئله ناهار خوردن فیلسوفان، در واقع یک مسئله کلاسیک concurrency است:

> پنج فیلسوف در کنار هم دور یک میز غذا می‌خورند. هر فیلسوف جایگاه خاص خود را در میز دارد. بین هر بشقاب یک چنگال قرار دارد. غذایی که سِرو می‌شود نوعی اسپاگتی است که باید با دو چنگال خورده شود. هر فیلسوف فقط می‌تواند به طور متناوب فکر کند و غذا بخورد. علاوه بر این، یک فیلسوف فقط می‌تواند اسپاگتی خود را زمانی بخورد که هم چنگال چپ و هم چنگال راست را داشته باشد؛ بنابراین دو چنگال فقط زمانی در دسترس خواهد بود که دو همسایه نزدیک آن‌ها در حال فکرکردن باشند و نه در حال غذاخوردن. پس از این‌که یک فیلسوف غذاخوردن را تمام کرد، هر دو چنگال را پایین می‌گذارد.

برای این تمرین به یک Cargo installation محلی نیاز داری. کد زیر را در فایلی به نام `src/main.rs` کپی کنید، جاهای خالی را پر کنید و آزمایش کنید که `cargo run` به بن‌بست (deadlock) نمی‌خورد:

```
use std::sync::{mpsc, Arc, Mutex};
use std::thread;
use std::time::Duration;

struct Fork;

struct Philosopher {
    name: String,
    left_fork: ...,
    right_fork: ...,
    thoughts: ...,
}

{
```

```
                                                    } impl Philosopher
                                                } (fn think(&self
                                                self.thoughts
        ((self.name& ,"!دیدج هدیا کی {} !ولیا")!send(format.
                                                ;()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 //
                                                                {
```

می توانید از Cargo.toml زیر اضافه کنید:

```
                                                        [package]
                                                "name = "dining-philosophers
                                                    "version = "0.1.0
                                                    "edition = "2021
```

62.2 جستجوگر لینک چندنِردی

اجازه دهید از دانش جدید خود برای ایجاد یک جستجوگر لینک multi-thread استفاده کنیم. باید از یک صفحه وب شروع شود و بررسی کند که لینک‌های موجود در صفحه عبارت هستند. باید به‌صورت بازگشتی صفحات دیگر را در همان دامنه بررسی کند و این کار را تا زمانی که همه صفحات تأیید نشده-اند ادامه دهد.

برای این کار به یک کلاینت HTTP مانند reqwest نیاز دارید. شما همچنین به راهی برای افتن لینک-ها نیاز دارید، ما می توانیم از reqwest استفاده کنیم. در نهایت، ما به روشی برای رسیدگی به خطاها نیاز داریم پس درنتیجه از thiserror استفاده خواهیم کرد.

Create a new Cargo project and reqwest it as a dependency with:

```
                                                    cargo new link-checker
                                                        cd link-checker
                                    cargo add --features blocking,rustls-tls reqwest
                                                    cargo add scraper
```

```
cargo add thiserror
```

اگر `cargo add` با `error: no such subcommand` روبه‌رو شد، لطفاً فایل `Cargo.toml` را در ویرایشگر متن دلخواه‌تان باز کنید و وابستگی‌های ذکر شده در زیر را اضافه کنید.

فرامین `cargo add` را در فایل `Cargo.toml` به‌صورت زیر به‌روزرسانی می‌کنند:

```
[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"
```

اکنون می‌توانید صفحه شروع را دانلود کنید. با یک سایت کوچک مانند https://www.google.org امتحان کنید.

فایل `src/main.rs` شما باید چیزی شبیه به این باشد:

```
use reqwest::blocking::Client;
use reqwest::Url;
use scraper::{Html, Selector};
use thiserror::Error;

#[derive(Error, Debug)]
enum Error {
    #[error("request error: {0}")]
    ReqwestError(#[from] reqwest::Error),
    #[error("bad http response: {0}")]
    BadResponse(String),
}

#[derive(Debug)]
struct CrawlCommand {
    url: Url,
    extract_links: bool,
}

fn visit_page(client: &Client, command: &CrawlCommand) -> Result<Vec<Url>, Error> {
    println!("بررسی {:#}", command.url);
    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} :دنک زاب را کنیل دناوتیمن")!Err(err) => println
                                                                          {
                                                                              {
```

ای‌ن ک‌د را در src/main.rs اج‌را ک‌نی‌د

```
cargo run
```

Task

- ب‌رای ب‌ررس‌ی م‌وازی ل‌ی‌نک‌ه‌ا از threadه‌ا اس‌ت‌فاده ک‌نی‌د: URLه‌ای‌ی را ک‌ه ب‌ای‌د ب‌ررس‌ی ش‌ون‌د ب‌ه ی‌ک channel ارس‌ال ک‌نی‌د و اج‌ازه ده‌ی‌د چ‌ن‌د thread ب‌ه‌ص‌ورت م‌وازی URLه‌ا را ب‌ررس‌ی ک‌نن‌د.
- ای‌ن را ب‌ه ص‌ورت ب‌ازگ‌ش‌ت‌ی پ‌ی‌ون‌ده‌ا گس‌ت‌رش ده‌ی‌د ت‌ا از ه‌م‌ه ص‌ف‌ح‌ات را در دام‌ن‌ه «www.google.org» اس‌ت‌خ‌راج ک‌نی‌د. ح‌د ب‌ال‌ا را ح‌دود ۱۰۰ ص‌ف‌ح‌ه ی‌ ب‌ی‌ش‌ت‌ر ق‌رار ده‌ی‌د ت‌ا در ن‌ه‌ای‌ت ت‌وس‌ط س‌ای‌ت م‌س‌دود ن‌ش‌وی‌د.

62.3 راه ح‌ل‌ه‌ا

ف‌ل‌س‌ف‌ه Dining

```
;{use std::sync::{mpsc, Arc, Mutex
                ;use std::thread
            ;use std::time::Duration
```

```rust
;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& ,"!دیدجد هدیا کی {} !ولیا")!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
                                                       {
                                                           {
```

جست‌جوگر Link

```
                                 ;{use std::sync::{mpsc, Arc, Mutex
                                                ;use std::thread

                                    ;use reqwest::blocking::Client
                                                ;use reqwest::Url
                                    ;{use scraper::{Html, Selector
                                             ;use thiserror::Error

                                                   [(derive(Error, Debug]#
                                                       } enum Error
                                          [("{error("request error: {0]#
                                ,(ReqwestError(#[from] reqwest::Error
                                  [("{error("bad http response: {0]#
                                                ,(BadResponse(String
                                                                   {

                                                       [(derive(Debug]#
                                         } struct CrawlCommand
                                                  ,url: Url
                                               ,extract_links: bool
                                                                   {

} <fn visit_page(client: &Client, command: &CrawlCommand) -> Result<Vec<Url>, Error
                                            ;(command.url ,"{#:} بررسی")!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 ,"{#:} :دش تفایرد crawling یاطخ")!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
                                                                          {
```

شخب XIV

همزمان: عرص

فصل 63

خوش آمدید

"Async" یک مدل concurrency است که در آن چندین کار به طور همزمان با اجرای هر کار تا زمانی که مسدود شود، اجرا می‌شود و سپس به کار دیگری که آماده ادامه دادن است سوئیچ می‌شود. این مدل اجازه می‌دهد تا تعداد بیشتری کار را روی تعداد محدودی از رشته‌ها اجرا کنید و به این دلیل است که سربار هر task معمولاً بسیار کم است و سیستم‌عامل‌ها معمولاً مقدماتی را برای شناسایی مؤثر I/O که قادر به ادامه هستند فراهم می‌کنند.

عملیات Rust asynchronous بر اساس "futures" است، که نشان‌دهنده کاری است که ممکن است در آینده تکمیل شود. futureها تا زمانی که عالمت کامل بودنشان را ندهند، «polled» می‌شوند.

Futureها توسط یک زمان‌بندی ناهمزمان نظارت می‌شوند و چندین زمان‌بندی مختلف در دسترس هستند.

مقایسه

- پایتون مدل مشابه‌ی را در «asyncio» خود دارد. بااین‌حال، تایپ «Future» آن مبتنی بر callback است و poll نشده است. برنامه‌های Async Python به یک «حلقه» شبیه به runtime در Rust نیاز دارند.
- این مورد شبیه "Promise" در جاوا اسکریپت است که دوباره مبتنی بر callback است. runtime زبان حلقه رویداد (event loop) را پیاده سازی می‌کند، بنابراین بسیاری از جزئیات واضح در Promise پنهان می‌شوند.

برنامه زمانی

با احتساب استراحت‌های ۱۰ دقیقه‌ای، این جلسه باید حدود ۳ ساعت و ۲۰ دقیقه طول بکشد. شامل موارد زیر است:

بخش	مدت زمان
مبانی Async	۳۰ دقیقه
کانال‌ها و Control Flow	۲۰ دقیقه
Pitfallها	۵۵ دقیقه
تمرین‌ها	۱ ساعت و ۱۰ دقیقه

فصل 64

Async ینابم

ین بخش دیاب حدود ۳۰ هقیقد طول بکشد و شامل موارد زیر است:

اسلایدِ	مدت زمان
async/await	۱۰ هقیقد
Futures	۴ هقیقد
Runtimes	۱۰ هقیقد
Task	۱۰ هقیقد

async/await 64.1

:At a high level, async Rust code looks very much like ”normal” sequential code

```
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));
}
```

.This slide should take about 6 minutes

:نکات کلیدی

- توجه داشته باشید که این یک مثال ساده برای نشان دادن syntax است. هیچ عملیات طولانی مدت یا هیچ همزمانی(concurrency) واقعی در آن وجود ندارد!

- نوع برگشت async call چیست؟
  - برای مشاهده type از `let future: () = async_main(10);` در `main` استفاده کنید.
- کلمه کلیدی "async" شیرینیِ syntax زبان Rust است. کامپایلر نوع بازگشتی را با یک future جایگزین می‌کند.
- شما نمی‌توانید بدون دستورالعمل‌های اضافی به کامپایلر در مورد نحوه استفاده از future بازگشتی، `main` را async کنید.
- برای اجرای کده‌ای همگام به یک اجرا کننده (executor) نیاز دارید. `block_on` که thread رشته فعلی را تا زمانی که future ارائه شده تکمیل شود مسدود می‌کند.
- همه‌شه `await` به طور ناهمزمان (asyn) منتظر تکمیل یک عملیات دیگر است. برخلاف `block_on` یک `await` معمولاً thread فعلی را مسدود نمی‌کند.
- `await` فقط می‌تواند در داخل یک تابع async استفاده شود (یا block؛ این مورد در آینده معرفی می‌شوند).

Futures 64.2

Future یک trait است، اجرا شده توسط objectهایی که نشان دهنده عملیاتی هستند که ممکن است هنوز کامل نشده باشد. می‌توان یک future را poll کرد و poll یک Poll را برمی‌گرداند.

```
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,
}
```

یک تابع async یک `impl Future` را برمی‌گرداند. همچنین امکان (اما غیرمعمول) پیاده‌سازی Future برای تایپ‌های خودتان نیز وجود دارد. برای مثال، `JoinHandle` برگردانده شده از `tokio::spawn` Future را پیاده‌سازی می‌کند تا امکان پیوستن (joining) به آن را فراهم کند.

کلمه کلیدی `await` که برای Future اعمال می‌شود، باعث می‌شود که تابع async فعلی تا زمانی که Future آماده شود متوقف شود و سپس خروجی آن ارزیابی شود.

.This slide should take about 4 minutes

- تایپ‌های Future و Poll دقیقاً همانطور که نشان داده شده است اجرا می شوند. برای نمایش پیاده‌سازی‌ها در اسناد، روی لینک‌ها کلیک کنید.
- ما به Pin و Context نخواهیم رسید، زیرا به جای ساختن کده‌ای اولیه async، بر نوشتن کده‌ای async تمرکز خواهیم کرد. به طور خلاصه:
  - Context به Future اجازه می‌دهد تا زمانی که رویدادی رخ می دهد، خود را برای poll مجدد برنامه‌ریزی کند.

- `Pin` تضمین می‌کند که Future در حافظه جابه‌جا نمی‌شود، بنابراین pointerهای future مرجع باقی می‌مانند. این برای اجازه دادن به referenceها برای مرجع ماندن پس از `await` لازم است.

Runtimes 64.3

یک *runtime* برای انجام عملیات به صورت ناهمزمان از (_a_reactor) پشتیبانی می کند و مسئول اجرای futureها (an *executor*) است. Rust یک runtime داخلی ندارد، اما چندین گزینه دیگر در دسترس است:

- Tokio: کارایی(performant)، با یک اکوسیستم با کارایی بالا به خوبی توسعه یافته مانند Hyper برای HTTP یا [Tonic] (https://github) .com/hyperium/tonic برای gRPC.
- async-std:: هدفش این است که یک "std for async" باشد و شامل یک runtime اولیه در `async::task` است.
- smol: ساده و سبک است

چندین برنامه بزرگتر زمان اجرا (runtime) مخصوص به خود را دارند. برای مثال، Fuchsia اکنون یکی runtime دارد.

.This slide and its sub-slides should take about 10 minutes

- توجه داشته باشید که از میان زمان‌های اجرا ذکر شده، فقط Tokio در playground زبان Rust پشتیبانی می‌شود.playground همچنین اجازه ورود/خروجی (I/O) را نمی دهد، بنابراین بیشتر چیزهای async جالب نمی‌توانند در playground اجرا شوند.
- Futureها از این جهت «بی‌اثر(inert)» هستند که هیچ کاری انجام نمی‌دهند (حتی عملیات I/O شروع نمی‌کنند) مگر اینکه یک مجری(executor) وجود داشته باشد که آن‌ها را polling کند. به عنوان مثال، این با JS Promises متفاوت است که حتی اگر هرگز استفاده نشوند تا پایان کامل شدن برنامه اجرا خواهند شد.

Tokio 64.3.1

:Tokio provides

- یک runtime از نوع multi-thread برای اجرای کدهای ناهمزمان(asynchronous).
- یک asynchronous version کتابخانه‌ای استاندارد است.
- اکوسیستم بزرگی از کتابخانه‌ها.

```
use tokio::time;

async fn count_to(count: i32) {
    for i in 0..count {
        println!("شمارش {i}!");
        time::sleep(time::Duration::from_millis(5)).await;
    }
}

#[tokio::main]
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;
    }
}
```

```
{
    {
```

- با ماکرو `tokio::main` اکنون می‌توانی `main` را async کنی.
- تابع spawn یک "task" جدید و همزمان ایجاد می‌کند.
- توجه: spawn یک Future می‌گیرد، پس `await.` را در `count_to` صدا نمی‌زنید.

Further بررسی:

- چرا `count_to` (مرحله اول) به مقدار ۱۰ نمی‌رسد؟ این نمونه‌ای از غلط async است. `tokio::spawn` یک handle را برمی‌گرداند که می‌توان در انتظار تمام شدن تا تمام شود.
- به جای spawn مورد «`count_to(10).await`» را امتحان کنید.
- منتظر کار برگشتی از `tokio::spawn` باشید.

Task 64.4

Rust یک task system دارد که نوعی thread سبک وزن است.

یک task یک future در سطح بالا دارد که اجراکننده (executor) برای ادامه کار آن را poll می‌کند. آن future ممکن است یک یا چند future تودرتو داشته باشد که متد `poll` آن را poll می‌کند، که به طور ناپایداری با یک stack فراخوانی شده مطابقت دارد. همزمانی در یک task با poll از چندین child future، درون‌رفتن یک تا یم و یک عملیات I/O امکان‌پذیر است.

```
;{use tokio::io::{self, AsyncReadExt, AsyncWriteExt
                ;use tokio::net::TcpListener

                                        [tokio::main]#
                                } <()>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} ،می‌ری‌کشتم ی‌نفل‌ت اس‌امت از")!let reply = format
   ;("socket.write_all(reply.as_bytes()).await.expect("socket error
                                                    ;({
                                                        {
                                                            {
```

This slide should take about 6 minutes.

این مثال را در `src/main.rs` آماده شده خود کپی کنید و آن را از آنجا اجرا کنید.

س‌ع‌ی ک‌ن‌ی‌د ب‌ا ک‌ی ا‌ب‌ز‌ا‌ر ا‌ت‌ص‌ا‌ل TCP م‌ا‌ن‌ن‌د nc ی‌ ا telnet ب‌ه آ‌ن م‌ت‌ص‌ل ش‌وی‌د.

- 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`.
- ب‌ل‌وک async ر‌ا ب‌ه ی‌ک ت‌ع‌ا‌ر‌ی‌ف غ‌ی‌ر‌ه‌د‌ی‌د‌ه و م‌ی‌ر‌ی‌د ت‌ا ا‌ط‌خ ا‌ب‌ا ا‌س‌ت‌ف‌ا‌د‌ه از ? ب‌ه‌ب‌ود ب‌خ‌ش‌ی‌د.

فصل 65

کانال‌ها و Control Flow

این بخش بای حدود ۲۰ دقیقه طول بکشد. این شامل:

اسلاید	مدت زمان
کانال‌های Async	۱۰ دقیقه
Join	۴ دقیقه
Select	۵ دقیقه

65.1 کانال‌های Async

چندین crate از asynchronous channel پشتیبانی می‌کنند. به عنوان مثال tokio:

```
{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;
        println!("تا اکنون {count} عدد ping دریافت شده است.");
    }
}

async fn main() {
    println!("ping_handler کامل شد تست");

    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
{
```

.This slide should take about 8 minutes

- اندازه کانال را به3 تغییر دهید و ببینید که چگونه بر اجرا تأثیر می‌گذارد.
- به‌طورکلی، interface شبیه به channelهای sync است که در کل‌اس صبحگاهی دیده می‌شود.
- تمام std::mem::drop را حذف کنید. چه اتفاقی می‌افتد؟ چرا؟
- این crate مربوط به Flume دارای کانال‌هایی است که sync و async send و recv را اجرا می‌کنند. این کار می‌تواند برای برنامه‌های پیچیده با taskهای پردازشی IO و CPU سنگین مناسب باشد.
- چیزی که کار با کانال‌های async را ترجیح می‌دهد، توانایی ترکیب آن‌ها با دیگر future برای ترکیب آن‌ها و ایجاد جریان کنترل پیچیده است.

Join 65.2

عملیات پیوستن (join) منتظر می‌ماند تا تمام مجموعه‌ای از futureها آماده شوند و مجموعه‌ای (collection) از نتایج آن‌ها را برمی‌گرداند. این شبیه به Promise.all در JavaScript یا asyncio.gather در پایتون است.

```
;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
{

[tokio::main]#
} ()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
{
```

.This slide should take about 4 minutes

این مثال را در src/main.rs آماده شده خود کپی کنید و آن را از آنجا اجرا کنید.

- برای چند future از تابع‌های مختلف، می‌توانید از std::future::join! استفاده کنید، اما باید بدانید که در زمان کامپایل چند future خواهید داشت. این در حال حاضر در جعبه (crate از نوع

futures است که به زودی در std::future تثبیت می‌شود.

- خطر join این است که یکی از futureها ممکن است هرگز resolve نشود، این مسئله باعث می‌شود برنامه متوقف شود.
- همچنین می‌توانید join_all را با join! ترکیب کنید، به‌عنوان مثال برای پیوستن (join!) همه درخواست‌ها به یک سرویس http و همچنین یک کوئری پایگاه داده سعی کنید. این یک tokio::time::sleep را با استفاده از futures::join! به future اضافه کنید. یک timeout نیست (که به select! نیاز دارد و در فصل بعد توضیح داده می‌شود) بلکه join! را نشان می‌دهد.

Select 65.3

یک عملیات انتخابی منتظر می‌ماند تا هر یک از مجموعه‌ای از futureها آماده شود و به نتیجه آن future پاسخ می‌دهد. در JavaScript این مورد شبیه به Promise.race است و در پایتون با (asyncio.wait(task_set, return_when=asyncio.FIRST_COMPLETED قابل مقایسه می‌باشد.

مانند عبارت تطبیق (match statement)، بدنه pattern دارای تعدادی بازو است که هر کدام به شکل عبارت pattern = future => statement هستند. هنگامی که future آماده است، مقدار بازگشتی آن توسط pattern تجزیه می‌شود. سپس statement با متغیرهای حاصل اجرا می‌شود. در نتیجه statement مانند select! می‌شود.

```
use tokio::sync::mpsc;
use tokio::time::{sleep, Duration};

#[tokio::main]
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"),
        }
    });
    tokio::spawn(async move {
        sleep(Duration::from_millis(10)).await;
        tx.send(String::from("سلام!")).expect("Failed to send greeting.");
    });
    listener.await.expect("خورد تست شکست دهنونش");
}
```

This slide should take about 5 minutes.

- بلوک async listener در اینجا یک شکل رایج است: منتظر برخی رویدادهای async یا به‌عنوان مثال برای timeoutها باشید. sleep را به sleep طولانی‌تر تغییر دهید تا شاهد شکست آن باشید. چرا send نیز در این شرایط شکست می‌خورد؟
- 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.

فصل 66

Pitfallها

Async / await انتزاع راحت و کارآمدی را برای برنامه نویسی concurrent asynchronous فراهم می‌کند. با این حال، مدل async/wait در Rust نیز با سهم خود از مشکلات و pitfallها و footgunها همراه است. برخی از آن‌ها را در این فصل توضیح می‌دهیم.

این بخش بایدحدود ۵۵ دقیقه طول بکشد. آن شامل:

اسلایدها	مدت زمان
مسدود کردن Executor	۱۰ دقیقه
Pin	۲۰ دقیقه
صفات Async	۵ دقیقه
لغو	۲۰ دقیقه

66.1 مسدود کردن executor

اکثر async runtimeهای تنها IO task را به صورت همزمان (concurrent) اجرا شوند. این بدان معنی است که تسک‌های block کردن CPU باعث مسدود شدن executor و جلوگیری از اجرای سایر تسک‌ها می‌شود. یک راه حل آسان این است که در صورت امکان از متدهای معادل async استفاده کنید.

```
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()
    );
}

#[tokio::main(flavor = "current_thread")]
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;
}
```

.This slide should take about 10 minutes

- کد را اجرا کنید و ببینید که sleepها به طور متوالی اتفاق می‌افتند و نه به صورت هم‌زمان (concurrent).
- این "current_thread" همه taskها را روی یک thread قرار می‌دهد. این اثرگذاری را آشکارتر می‌کند، اما این اشکال هم‌چنان در طبیعت multi-threaded وجود دارد.
- `std::thread::sleep` را به `tokio::time::sleep` تغییر دهید و منتظر نتیجه باشید.
- راه‌حل دیگر `tokio::task::spawn_blocking` است که یک thread واقعی ایجاد می‌کند و handle آن را بدون مسدود کردن executor به future تبدیل می‌کند.
- شما نباید taskها را به عنوان threadهای سیستم عامل در نظر بگیرید. آن‌ها از نگاشت ۱ به ۱ پشتیبانی نمی‌کنند و اکثر executorها به بسیاری از taskها اجازه می‌دهند روی یک thread سیستم عامل اجرا شوند. این امر به‌ویژه هنگام تعامل با کتابخانه‌های دیگر از طریق FFI مشکل‌ساز است، جایی که آن کتابخانه ممکن است به ذخیره‌سازی محلی thread یا نگاشت (map) به threadهای سیستم‌عامل خاص (مانند CUDA) بستگی داشته باشد. در چنین شرایطی `tokio::task::spawn_blocking` را ترجیح دهید.
- با احتیاط از همگام‌سازی mutexها استفاده کنید. نگه داشتن یک mutex روی یک `await`. ممکن است باعث مسدود شدن task دیگری شود و آن task ممکن است در همان thread در حال اجرا باشد.

Pin 66.2

بلوک‌ها و توابع Async انواعی را برمی‌گردانند که ویژگی Future را پیاده‌سازی می‌کنند. نوع برگشتی تنها چیزی تبدیل کامپایلر است که متغیرهای محلی را به داده های ذخیره شده در future تبدیل می کند.

برخی از این متغیرها می‌توانند اشاره‌گره‌ایی را برای سایر متغیره‌ای محلی نگه دارند. به همین دلیل، future هرگز نباید به مکان حافظه دیگری منتقل شود، زیرا این pointerها را باطل می‌کند.

برای جلوگیری از جابجایی تایپ future در حافظه، فقط از طریق یک pointer پین شده می توان آن را بررسی کرد. Pin یک wrapper در اطراف یک reference است که تمام عملیاتی را که می‌تواند نمونه‌ای را که به آن اشاره می‌کند به یک مکان حافظه متفاوت منتقل کند را ممنوع می‌کند.

```
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.
#[derive(Debug)]
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
                                                                  {

                                                        [tokio::main]#
                                                    } ()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}  رارکت یارب راک هجیتن")!println
                                                                  {
                                                                      {
```

.This slide should take about 20 minutes

- شما ممکن است این را به عنوان نمونه ای از الگوی بازیگر (actor pattern) تشخیص دهید. بازیگران معمولاً `select!` را در یک حلقه صدا می‌زنند.
- این به عنوان یک چمبدی از چند درس قبلی عمل می کند، بنابراین وقت خود را صرف آن کنید.
  - به‌سادگی یک `_ = println { => (..)!sleep(Duration::from_millis(100))` را به `select!` اضافه کنید. این مورد هرگز اجرا نمی شود. چرا؟
  - درعوض، یک `timeout_fut` حاوی future خارج از loop اضافه کنید:

```
                                ;((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& = _
                                                            {
                                                                {
```

– این مورد کام‌پای‌ل می‌شود، اما پس از انقضای timeout و در هر تکرار برابر با `Poll::Ready` است (future ترکیبی به این مسئله کمک می‌کند). به‌روزرسانی برای بازنشانی `timeout_fut` هر بار که منقضی می‌شود:

```
;(((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.
- جای‌گزینی دیگر این است که به هیچ وجه از `pin` استفاده نکنید، بلکه task دیگری ایجاد کنید که هر 100 میلی‌ثانیه به یک کانال `oneshot` ارسال می‌شود.
- داده‌هایی که حاوی اشاره‌گره‌هایی به خود هستند، خود ارجاعی (self-referential) نامیده می شوند. به طور معمول، Rust borrow checker از جابجایی داده‌های خودارجاعی جلوگیری می‌کند، زیرا ممکن است اشاره‌گرها پس از جابجایی داده‌ها به آن‌ها نامعتبر شوند. با این حال، تبدیل کد برای بلوک‌ها و توابع async توسط borrow checker تأیید نمی‌شود.
- `Pin` یک wrapper در اطراف یک reference است. یک object را نمی‌توان با استفاده از یک pointer پین شده از جای خود حرکت داد. با این حال، هنوز هم می‌توان آن را از طریق یک pointer بدون پین جابجا کرد.
- متد `poll` از ویژگی `Future` به جای `Pin<&mut Self>` برای اشاره به نمونه (instance) استفاده می‌کند. به همین دلیل است که فقط می‌توان آن را روی یک اشاره‌گر پین شده فراخوانی کرد.

66.3 صفات Async

متدهای Async در traitها اخیراً در انتشار 1.75 تثبیت شده‌اند. این نیاز به پشتیبانی برای استفاده از موقعیت بازگشتی (RPIT) `impl Trait` در traitها را داشت، زیرا شیرین‌زدایی (desugaring) برای `async fn` شامل `-> impl Future<Output = ...>` است.

با این‌حال، حتی با پشتیبانی native امروز، برخی از مشکلات در مورد `async fn` و RPIT در ویژگی‌ها وجود دارد:

- Return-position impl Trait تمام طول عمر‌های درون محدوده را ثبت می کند (بنابراین برخی از الگوهای قرض کردن (borrowing) نمی توانند بیان شوند)

- ویژگی‌هایی که متدهای آن‌ها از موقعیت بازگشتی `impl trait` یا `async` استفاده می‌کنند با dyn سازگار نیستند.

If we do need dyn support, the crate async_trait provides a workaround through a macro, :with some caveats

```
use async_trait::async_trait;
use std::time::Instant;
use tokio::time::{sleep, Duration};

#[async_trait]
trait Sleeper {
    async fn sleep(&self);
}

struct FixedSleeper {
    sleep_ms: u64,
}

#[async_trait]
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());
        }
    }
}

#[tokio::main]
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;
}
```

.This slide should take about 5 minutes

- استفاده از `async_trait` آسان است، اما توجه داشته باشید که برای رسیدن به این هدف از heap allocation‌ها استفاده می‌کند. این heap allocation دارای سربار عملکرد است.

- چالش‌های پشتیبانی برای `async trait` در زبان Rust بسیار عمیق هستند و احتمالاً ارزش توصیف عمیق در این‌جا را ندارند. Niko Matsakis در این پست آن‌ها را به خوبی توضیح داده است. به خصوص اگر شما به این موضوع علاقه‌مند هستید.
- سعی کنید یک sleep خواب جدید ایجاد کنید که برای مدت زمان تصادفی می‌خوابد و آن را به Vec اضافه کنید.

66.4 لغو

کنار گذاشتن future به این معنی است که دیگر هرگز نمی‌توان آن را poll کرد. به این حالت *cancellation* می‌گویند و می‌توانند در هر نقطه await رخ دهد. برای اطمینان از عملکرد صحیح سیستم حتی در صورت لغو futureها، دقت مناسب لازم است. به‌عنوان مثال، نباید داده‌ها را از دست بدهید یا به بن‌بست (deadlock) برسید.

```
;{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
```

```
{

[tokio::main]#
} <()>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
{
```

.This slide should take about 18 minutes

- کامپایلر در مورد cancellation-safety کمکی نمی‌کند. باید مستندات API را بخوانید و در نظر بگیرید که async fn ها چه وضعیتی دارند.
- برخلاف panic و ?، لغو یا cancellation بخشی از جریان کنترل عادی (و رسیدگی به خطا) است.
- این مثال بخش‌هایی از string را از دست می‌دهد.
  - هر زمان که شاخه tick() اول تمام شود، next() و buf آن حذف می شوند.
  - LinesReader را می‌توانید با دیباگ کردن، از ساختار بخشی از buf به این صورت بسازید:

```
} 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.
... //
{
```

{

- `'Interval::tick'` برای cancellation-safe است زیرا ردیابی می‌کند که آیا یک tick تحویل داده شده است.
- `AsyncReadExt::read` برای cancellation-safe است زیرا داده‌ها را برمی‌گرداند یا نمی‌خواند.
- `AsyncBufReadExt::read_line` مشابه مثال است و شبیه cancellation-safe نیست. برای جزئیات و موارد جایگزین به مستندات آن مراجعه کنید.

فصل 67

اهنیرمت

این بخش بایدحدود ۱ ساعت و ۱۰ دقیقه طول بکشد. این بخش شامل موارد زیر است:

اسلایدم	مدت زمان
فلسفه Dining	۲۰ دقیقه
پخش برنامه چت	۳۰ دقیقه
راه حلها	۲۰ دقیقه

67.1 Dining Philosophers --- Async

برای توضیح مشکل به dining philosophers مراجعه کنید.

مانند قبل، برای این تمرین به cargo/running-locally.md](../../)(Cargo installation) نیاز دارید. کد زیر را در فایلی به نام `src/main.rs` کپی کنید، چاهای خالی را پر کنید و تست کنید که `cargo run` به بستن (`src/main.rs`) نمیخورد:

```
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!("{} یک ایده جدید!", &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"]&

                                                        [tokio::main]#
                                                      } ()async fn main
                                                    Create forks //

                                                Create philosophers //

                                            Make them think and eat //

                                              Output their thoughts //
                                                            {
```

از آن‌جایی که این بار از Async Rust استفاده می‌کنید، به وابستگی tokio نیاز دارید. می‌توانید از Cargo.toml زیر استفاده کنید:

```
                                                        [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
```

همچنین توجه داشته باشید که این بار باید از ماژول Mutex و mpsc از tokio crate استفاده کنید.

.This slide should take about 20 minutes

- آیا می‌توانید پیاده‌سازی خود را تک thread ای کنید؟

67.2 پخش برنامه چت

در این تمرین، ما می‌خواهیم از دانش جدید خود برای پیاده سازی یک برنامه broadcast chat استفاده کنیم. ما یک سرور چت داریم که کاربران به آن متصل می‌شوند و پیام‌های خود را منتشر می‌کنند. کلاینت پیام‌های کاربر را از ورودی استاندارد می‌خوانند و آن‌ها را به سرور ارسال می‌کنند. سرور چت هر پیامی را که دریافت می‌کند برای همه کاربران پخش می‌کند.

برای این کار، از broadcast channel در سمت سرور و tokio_websockets برای ارتباط بین کلاینت و سرور.

یک پروژه Cargo جدید ایجاد کنید و وابستگی‌های زیر را اضافه کنید:

:Cargo.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
websockets = { version = "0.9.0", features = ["client", "fastrand", "server", "sha1_smol
```

APIهای مورد نیاز

شما به توابع زیر از tokio و tokio_websockets نیاز دارید. چند دقیقه را برای آشنایی با API اختصاص دهید.

- StreamExt::next() توسط WebSocketStream: برای خواندن نامزمان پیامها از یک جریان وب سوکت.
- SinkExt::send() پیادهسازی شده توسط WebSocketStream: برای ارسال نامزمان پیامها در یک Websocket Stream.
- Lines::next_line(): برای خواندن نامزمان پیامهای کاربر از ورودی استاندارد.
- Sender::subscribe(): برای اشتراک در یک broadcast channel.

دو باینری

به طور معمول در یک پروژه Cargo، شما می توانید فقط یک فایل باینری و یک فایل src/main.rs داشته باشید. در این پروژه به دو باینری نیاز داریم. یکی برای کلاینت و دیگری برای سرور. شما به طور بالقوه میتوانید آنها را در دو پروژه Cargo جداگانه بسازید، اما ما آنها را در یک پروژه Cargo واحد با دو باینری قرار میدهیم. برای این کار، کلاینت و کد سرور باید زیر src/bin قرار گیرند (به documentation مراجعه کنید).

کد سرور و کلاینت زیر را به ترتیب در src/bin/server.rs و src/bin/client.rs کپی کنید. وظیفه شما این است که این فایلها را همانطور که در زیر توضیح داده شده است تکمیل کنید.

: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>> {

    // TODO: For a hint, see the description of the task below.
```

```
{

[tokio::main]#
} <<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} از دیدی جدید اتصال")!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

[tokio::main]#
} <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 //

{
```

راه‌اندازی برنامه‌ها

سرور را راه‌اندازی کنید با استفاده از:

```
cargo run --bin server
```

و این کلاینت‌ها با:

```
cargo run --bin client
```

Task

- توابع handle_connection را در src/bin/server.rs پیاده‌سازی کنید.
  - نکته: از tokio::select! برای انجام همزمان دو task در یک حلقه پیوسته استفاده کنید. یک task پیام‌های را از کانال دریافت می‌کند و آن‌ها را پخش(broadcast) می‌کند. دیگری پیام‌های دریافت شده توسط سرور را برای کاربر ارسال می‌کند.
- توابع اصلی را در src/bin/client.rs تکمیل کنید.
  - نکته: مانند قبل، از tokio::select! در یک حلقه پیوسته برای انجام همزمان دو task استفاده کنید: (۱) خواندن پیام های کاربر از ورودی استاندارد و ارسال آن‌ها به سرور و (۲) دریافت پیام از سرور و نمایش آن‌ها برای کاربر.
- اختیاری: پس از اتمام کار، کد را تغییر دهید تا پیام‌ها برای همه کلاینت‌ها، به جز فرستنده پیام، منتشر شود.

67.3 راه حل‌ها

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
            .send(format!("{} یک ایده جدید!", &self.name))
            .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"]&

                                                                [tokio::main]#
                                                            } ()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} :در اتاق یک ایده وجود دارد")!println
                                                                {
                                                                    {
```

چت برنامه پخش

: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."دینک پیات امیپ کی !دیدآمد شوخ chat هب")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
                                                    {
                                                        {
                                                            {
                                                                {

                                                [tokio::main]#
                        } <<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} از دیدی جال‌صات")!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

                                                [tokio::main]#
                                } <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
                                                              {
                                                                  {

                                                                      {
                                                                          {
                                                                              {
```

بخش XV

کلمات آخر

فصل 68

سپاس!

ممنون که آموزش‌های را از Comprehensive Rust 🦀! رو گرفت‌ید. امیدواریم لذت برده باشید و برای شما مفید بوده باشد.

ما از برگزاری این دوره بسیار لذت بردیم. این دوره کامل نیست، بنابراین اگر اشتباهی را مشاهده کردید یا ایده‌ای برای به‌بود دارید، لطفاً با GitHub با ما تماس بگیرید وارد شوید. ما دوست داریم از شما بشنویم.

فصل 69

واژه‌نامه

در زیر واژه‌نامه‌ای است که هدف آن ارائه تعریف کوتاه‌ی از بسیاری از اصطلاحات در زبان Rust است. برای ترجمه‌ها، این مورد نیز برای اتصال این اصطلاح به زبان اصلی انگلیسی است.

{ ;h1#glossary ~ ul { list-style: none; padding-inline-start: 0

h1#glossary ~ ul > li { /* Simplify with "text-indent: 2em hanging" when supported: https://caniuse.com/mdn-css_properties_text-indent_hanging */ padding-left: 2em; text- { ;indent: -2em

{ ;h1#glossary ~ ul > li:first-line { font-weight: bold

- :allocate
  تخصیص حافظه پویا در [heap](memory-management/review.md).
- :argument
  اطلاعاتی که به یک تابع در حین فراخوانی آن ارسال می‌شود.
- نوع مرتبط: نوعی که با یک ویژگی خاص مرتبط است. برای تعریف رابطه بین typeها در دفی است.
- :Bare-metal Rust
  توسعه سطح پایین Rust، اغلب در سیستم‌عامل‌های سیستمی که نیازمند استقرار می‌شود. Bare-metal Rust را ببینید.
- :block
  [Blocks](control-flow-basics/blocks-and-scopes.md) و *scope* را ببینید.
- :borrow
  Borrowing را ببینید.
- :borrow checker
  بخشی از کامپایلر Rust که بررسی می‌کند که همه قرض‌ها (borrows) معتبر هستند.
- brace:
  { and }. Also called *curly brace*, they delimit *blocks*.
- :build
  فرآیند تبدیل کد منبع به کد اجرایی یک برنامه قابل استفاده می‌باشد.
- :call
  برای فراخوانی یک تابع، کار درد دارد.

- channel:
  برای ارسال ایمن پیام‌ها [بین threadها](concurrency/channels.md) استفاده می‌شود.
- Comprehensive Rust 🦀:
  دوره‌ای اینجا Comprehensive Rust 🦀 نامیده می‌شوند.
- concurrency:
  اجرای چندین task یا process به طور همزمان.
- Concurrency در Rust را ببینید.
- constant:
  مقداری که در طول اجرای برنامه تغییر نمی‌کند.
- control flow:
  ترتیبی که دستورات یا عملگرها در یک برنامه اجرا می‌شوند.
- crash:
  یک شکست (failure) یا خاتمه غیرمنتظره و کنترل نشده یک برنامه است.
- enumeration:
  یک نوع داده که یکی از چندین ثابت نامگذاری شده را، احتمالاً با یک تاپل یا ساختار مرتبط، نگه می‌دارد.
- error:
  شرایط یا نتیجه غیرمنتظره‌ای که از رفتار مورد انتظار خارج می‌شود.
- error handling:
  فرآیند مدیریت و پاسخگویی به خطاهایی که در حین اجرای برنامه رخ می‌دهد.
- exercise:
  مشکل یا task که برای تمرین و آزمایش مهارت‌های برنامه‌نویسی طراحی شده است.
- function:
  یک بلوک کد قابل استفاده مجدد که وظیفه خاصی را انجام می‌دهد.
- garbage collector:
  مکانیزمی که به طور خودکار حافظه اشغال شده توسط اشیایی که دیگر استفاده نمی‌شوند را آزاد می‌کند.
- generics:
  قابلیتی که امکان نوشتن کد با متغیرهایی برای انواع را فراهم می‌کند و امکان استفاده مجدد از کد با انواع داده‌های مختلف را فراهم می‌کند.
- immutable:
  پس از ایجاد، دیگر قابل تغییر نیست.
- integration test:
  نوعی تست که تعامل بین بخش‌ها یا اجزای مختلف یک سیستم را تأیید می‌کند.
- keyword:
  یک کلمه رزرو شده در یک زبان برنامه‌نویسی که معنای خاصی دارد و نمی‌توان از آن به عنوان شناسه یا سایر نامگذاری‌ها استفاده کرد.
- library:
  مجموعه‌ای از routineها یا کدهای از پیش کامپایل شده که می توانند توسط برنامه‌ها استفاده شود.
- macro:
  ماکروهای Rust را می‌توان با یک ! در نام آن تشخیص داد. ماکروها زمانی استفاده می‌شوند که توابع

عادی کافی نباشد. یک مثال معمولی `format!` است که تعداد متغیری از آرگومان‌ها را می‌گیرد که توسط توابع Rust پشتیبانی نمی‌شوند.

- `main` function:

برنامه‌های Rust با تابع `main` شروع به اجرا می‌کنند.

- match:
یک ساختار جریان کنترلی در Rust که امکان تطبیق الگو بر روی مقدار یک عبارت را فراهم می‌کند.
- memory leak:
وضعیتی که در آن برنامه نمی‌تواند حافظه‌ای را که دیگر مورد نیاز نیست آزاد کند و منجر به افزایش تدریجی استفاده از حافظه می‌شود.
- method:
یک تابع مرتبط با یک object یا یک type در Rust.
- module:
فضای نامی که شامل تعاریف مانند توابع، انواع یا صفات برای سازمان‌دهی کد در Rust است.
- move:
انتقال مالکیت (ownership) یک مقدار از یک متغیر به متغیر دیگر در Rust.
- mutable:
یک ویژگی در Rust که به متغیرها اجازه می‌دهد پس از اعلان، اصلاح شوند.
- ownership:
مفهومی در Rust که مشخص می‌کند کدام قسمت از کد مسئول مدیریت حافظه مرتبط با یک مقدار است.
- panic:
یک وضعیت خطای غیرقابل جبران در Rust که منجر به خاتمه برنامه می‌شود.
- parameter:
مقداری که هنگام فراخوانی به یک تابع یا متد ارسال می‌شود.
- pattern:
ترکیبی از مقادیر، عبارت‌ها یا ساختارهایی که می‌توانند با یک عبارت در Rust مطابقت داده شوند.
- payload:
داده یا اطلاعاتی که توسط یک پیام، رویداد یا ساختار داده حمل می‌شود.
- program:
مجموعه‌ای از دستورالعمل‌هایی که یک کامپیوتر می‌تواند برای انجام یک کار خاص یا حل یک مشکل خاصی اجرا کند.
- مجموعه‌ای از دستورالعمل‌هایی که یک کامپیوتر می‌تواند برای انجام یک کار خاص یا حل یک مشکل خاص اجرا کند.
- receiver:
اولین پارامتر در متد Rust که نمونه‌ای را نشان می‌دهد که متد در آن فراخوانی می‌شود.
- reference counting:
یک تکنیک مدیریت حافظه که در آن تعداد ارجاعات به یک object ردیابی می‌شود و زمانی که شمارش به صفر می‌رسد، object تخصیص داده می‌شود.
- return:
یک کلمه کلیدی در Rust برای نشان دادن مقداری که باید از یک تابع برگردانده شود استفاده می‌شود.

- Rust:
ی ک زبان برنامه‌نویسی سیستمی که بر safety، کارایی و concurrency تمرکز دارد.
- Rust Fundamentals:
روزه‌ای ۱ تا ۴ این دوره.
- Rust در Android: این Rust in Android را ببینید.
- Rust در Chromium:
Rust in Chromium را ببینید.
- safe:
به کدی اشاره دارد که به قوانین مالکیت (ownership) در زبان Rust و قرض‌گرفتن (borrowing) پایبند است و از خطاهای مربوط به حافظه جلوگیری می‌کند.
- scope:
منطقه‌ای از یک برنامه که در آن یک متغیر معتبر است و می‌توان از آن استفاده کرد.
- standard library:
مجموعه‌ای از ماژول‌ها که عملکرده‌ای ضروری را در Rust ارائه می‌دهند.
- static:
یک کلمه کلیدی در Rust برای تعریف متغیره‌ای ثابت یا موارد با طول عمر `'static` استفاده می‌شود.
- string:
نوع داده‌ای که داده‌ای متنی را ذخیره می‌کند. برای اطلاعات بیشتر به Strings مراجعه کنید.
- struct:
یک نوع داده ترکیبی در Rust که متغیره‌ای انواع مختلف را تحت یک نام واحد جمع می‌کند.
- test:
یک ماژول Rust حاوی توابعی که صحت عملکرده‌ای دیگر را آزمایش می‌کند.
- thread:
دنباله‌ای جداگانه از اجرا در یک برنامه که امکان اجرای همزمان را فراهم می‌کند.
- thread safety:
ویژگی برنامه‌ای که رفتار صحیح را در یک محیط multithread تضمین می‌کند.
- trait:
مجموعه‌ای از متدهای تعریف شده برای یک type ناشناخته، راهی برای دستیابی به polymorphism در Rust ارائه می‌دهد.
- trait bound:
.An abstraction where you can require types to implement some traits of your interest
- tuple:
یک data type ترکیبی که شامل متغیره‌ای از انواع مختلف است. فیلدهای Tuple بی‌نام هستند و با شماره‌ترتیبی آنها قابل دسترسی هستند.
- type:
طبقه‌بندی که مشخص می‌کند کدام عملیات را می‌توان بر روی مقادیری از یک تایپ خاص در Rust انجام داد.
- type inference:
توانایی کامپایلر Rust برای شناسایی تایپ یک متغیر یا عبارت.

- undefined behavior:
اقدامات یا شرایطی در Rust که هیچ تضمینی وجود ندارند و اغلب منجر به رفتار غیرقابل پیش‌بینی برنامه می‌شوند.
- union:
یک data type که می‌تواند قادری از انواع مختلف را در خود نگه دارد، اما فقط یکی در یک زمان خاص.
- unit test:
Rust با پشتیبانی داخلی برای اجرای unit testهای کوچک و integration testهای بزرگتر ارائه می‌شود. Unit Tests را ببینید.
- unit type:
نوعی که هیچ داده‌ای را در خود نگه نمی‌دارد و به صورت tuple بدون هیچ عضوی نوشته شده است.
- unsafe:
زیرمجموعه (subset) در Rust که به شما امکان می‌دهد رفتار نامشخصی را فعال‌سازی کنید. [Unsafe Rust (unsafe-rust/unsafe.md)] را ببینید.
- variable:
یک مکان حافظه که داده‌ها را ذخیره می‌کند. متغیرها در یک *scope* معتبر هستند.

فصل 70

سرانجام برای Rust

جامعه Rust منابع بسیار باکیفیت و رایگان را به صورت آنلاین ایجاد کرده است.

مستندات رسمی

پروژه Rust میزبان منابع بسیاری است. این منابع، Rust را به طور کامل پوشش می‌دهند:

- [کتاب رایگان و مرجع در مورد Rust که این زبان را با جزئیات دقیقی پوشش می‌دهد و شامل چند پروژه برای ساخت نرم‌افزار می‌شود.](https://doc.rust-lang.org/book/)
- Rust By Example: در مورد Rust syntax را به کمک یک سری از مثال‌ها پوشش می‌دهد که ساختاره‌ای مختلف را به نمایش می‌گذارد. گاه‌ی اوقات شامل تمرین‌های کوچکی می‌شود که از شما خواسته می‌شود کد را در مثال‌ها گسترش دهید.
- [مستندات کامل کتابخانه استاندارد Rust می‌باشد.](https://doc.rust-lang.org/std/)
- The Rust Reference: کتاب نقاصی که گرامر و مدل حافظه Rust را توصیف می‌کنند.

راهنماهای تخصصی بیشتر میزبانی شده در سایت رسمی Rust:

- Rustonomicon: که unsafe Rust نامن را پوشش می‌دهد، از جمله کار با pointerهای خام و interfaceهای با زبان های دیگر (FFI) را تشریح می‌کنند.
- برنامه نویسی ناهمزمان در Rust: مدل برنامه‌نویسی ناهمزمان (asynchronous programming) جدیدی را پوشش می‌دهد که پس از نگارش کتاب Rust معرفی شده است.
- The Embedded Rust Book: مقدمه‌ای بر استفاده از Rust در embeddedها که بدون سیستم‌عامل هستند را شامل می‌شود.

مطالب آموزشی غیررسمی

مجموعه کوچکی از راهنماها و آموزش‌های دیگر برای Rust:

- Learn Rust the Dangerous Way: درباره Rust را از دیدگاه برنامه‌نویسان سطح پایین C پوشش می‌دهد.
- Rust for Embedded C Programmers: که Rust را از دیدگاه توسعه‌دهندگانی که سیستم‌عامل را به زبان C می‌نویسند پوشش می‌دهد.

- Rust for Professionals: که syntax مورد استفاده Rust را به کمک مقایسه‌های جانبی با زبان‌های دیگر مانند C، C++، Java، JavaScript و Python پوشش می‌دهد.
- Rust on Exercism: بیش از ۱۰۰ تمرین برای کمک به یادگیری Rust را شامل می‌شود.
- Ferrous Teaching Material: مجموعه‌ای از ارائه‌های کوچک که هم بخش پایه و هم پیشرفته زبان Rust را پوشش می‌دهد. موضوعات دیگری مانند WebAssembly و async/wait نیز پوشش داده شده است.
- 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.
- Beginner's Series to Rust: دو راه‌نمای Rust با هدف توسعه‌دهندگان جدید می‌باشد. اولی مجموعه‌ای از ۳۵ ویدیو و دومی مجموعه‌ای از ۱۱ ماژول است که دستور Rust و ساختاره‌ای اولیه را پوشش می‌دهد.
- Learn Rust With Entirely Too Linked Lists](https://rust-unofficial.github.io/too-many-lists/): کاوش عمیق قوانین مدیریت حافظه Rust، از طریق اجرای چند نوع مختلف list structure.

لطفاً [Little Book of Rust Books](https://lborb.github.io/book/) را برای کتابه‌ای بیشتر در مورد Rust ببینید.

فصل 71

اعتبارها

مطالب در این فصل بر روی بسیاری از منابع عالی مستندات Rust ساخته شده است. برای فهرست کامل منابع دیگر به صفحه [منابع](other-resources.md) مراجعه کنید.

محتوای Comprehensive Rust تحت مجوز Apache 2.0 مجوز دارند، لطفاً برای جزئیات بیشتر به 'LICENSE' مراجعه کنید.

مثال‌های Rust

برخی از مثال‌ها و تمرین‌ها از [Rust by Example](https://doc.rust-lang.org/rust-by-example/) کپی و اقتباس شده‌اند. لطفاً برای جزئیات، از جمله شرایط license، به دایرکتوری `third_party/rust-by-example/` مراجعه کنید.

تمرین‌های Rust

برخی تمرین‌ها از [Rust on Exercism](https://exercism.org/tracks/rust) کپی و اقتباس شده‌اند. لطفاً برای جزئیات، از جمله شرایط license، به دایرکتوری `third_party/rust-on-exercism/` مراجعه کنید.

CXX

بخش [Interoperability with C++](https://cxx.rs/) از تصویری از CXX استفاده می‌کند. لطفاً برای جزئیات، از جمله شرایط license، دایرکتوری `third_party/cxx/` را ببینید.