

Engineer

Rust

et kompileret sprog, der tjekker hukommelsen for dig



Hvad det er

Rust er et programmeringssprog til det arbejde, der traditionelt laves i C og C++, hvor et program selv styrer sin hukommelse og ofte linkes ind i software skrevet i andre sprog. Compileren laver maskinkode på forhånd. Version 1.0 blev annonceret den 15. maj 2015 som et sprog, der forener kontrol på lavt niveau over ydelsen med sikkerhedsgarantier, opnået uden en garbage collector eller et runtime, og det er netop det, der lader et Rust-bibliotek træde i stedet for et bibliotek i C.

Hukommelsen styres gennem ejerskab, et sæt regler, som compileren tjekker. Hver værdi har én ejer ad gangen, og når ejeren går ud af scope, droppes værdien, og dens hukommelse frigives på et tidspunkt, der ligger fast, når programmet oversættes, og ikke når en collector næste gang kigger forbi. Kode, der har brug for en værdi uden at overtage den, låner en reference, og lånereglerne tillader på ethvert tidspunkt enten én foranderlig reference eller et vilkårligt antal uforanderlige og kræver, at hver reference er gyldig. Et program, der bryder en regel, oversættes ikke. De samme regler udelukker datakapløb, så to tråde, der skriver til de samme data uden synkronisering, er en kompileringsfejl og ikke en fejl, der kun viser sig af og til.

Rust har ingen undtagelser. En funktion, der kan fejle på en forventet måde, returnerer et `Result`, der rummer enten en værdi eller en fejl, og compileren advarer som standard, hvis kalderen lader resultatet være ubrugt; en fejl, som programmet ikke kan komme sig over, som at læse forbi enden af et array, går i stedet gennem makroen `panic!`.

Statisk analyse er konservativ, så compileren afviser nogle programmer, der ville have været i orden, og noget arbejde, som at styre hardware eller kalde et andet sprog, kan slet ikke tjekkes. Til de tilfælde findes `unsafe`, der låser fem ekstra

evner op, blandt dem at dereferere en rå pointer og at kalde en usikker funktion, hvilket som standard omfatter enhver funktion, der er erklæret fra C. Det slår ikke borrow checkeren fra; det markerer de linjer, hvor programmøren og ikke compileren står inde for hukommelsessikkerheden, så sikkerheden ved disse operationer afhænger af, at programmøren opretholder deres kontrakter, også i den omgivende sikre kode. Rust-bogens råd er at holde de blokke små og pakke dem ind i en sikker grænseflade, og sådan er dele af standardbiblioteket bygget.

Der kommer en ny compiler hver sjette uge. Ændringer, der ville bryde eksisterende kode, venter på en udgave, en såkaldt edition, som kommer cirka hvert tredje år. Der er fire, 2015, 2018, 2021 og 2024, og den nyeste blev stabil den 20. februar 2025 med Rust 1.85.0. Hver pakke angiver sin udgave i `Cargo.toml`, hver compiler bygger alle de udgaver, der kom før den, og pakker i forskellige udgaver linkes sammen, så et projekt rykker op, når det selv vælger. Cargo, pakkehåndteringen og buildværktøjet, følger med sproget, rustfmt holder kodestilen ensartet, og tredjepartsbiblioteker kommer som pakker, der kaldes crates.

Brug begrebet i en beslutning

Brug »Hvad er Rust?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Den del, brugerne aldrig ser: hvor data ligger, hvilke regler der gælder, og hvordan andre systemer må tale med den.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har designet og drevet 6 ETL/ELT-pipelines — 6 ETL-pipelines på tre databasemotorer

Har designet og drevet 6 ETL/ELT-pipelines Kontakt os →

Hvorfor det betyder noget

I C og C++ bliver en pointer, der bruges, efter at dens hukommelse er frigivet, eller to tråde, der skriver de samme data, oversat uden indvendinger og fejler senere, af og til og ofte langt fra årsagen. Rust gør de programmer til kompileringsfejl, og fordi tjekkene kører, når programmet bygges, betaler det færdige program intet for dem.

Uden en garbage collector og uden et runtime at slæbe rundt på kan Rust-kode ligge under software i andre sprog. Alt, der kan kalde C, kan kalde en Rust-kerne gennem en C-grænseflade, så én gennemtestet implementering kan betjene

klienter på flere platforme. Rust-bogen nævner kommandolinjeværktøjer, webtjenester, indlejrede enheder og store dele af webbrowseren Firefox blandt sprogets anvendelser i produktion.

Prisen betales på forhånd. Koden skal formes, så compileren kan se, at den er sikker, og det betyder ofte at beslutte, hvem der ejer en værdi, i stedet for at sende referencer frit rundt, og bogen siger selv, at ejerskab tager tid at vænne sig til. Når compileren afviser et program, den ikke kan verificere, er svaret som regel en anden struktur og ikke en vej uden om tjekket.

Stabilitet er den anden halvdel af handlen. Siden 1.0 har ændringer, der bryder eksisterende kode, stort set været udelukket, så en sådan ændring kun kommer gennem en udgave, som en pakke selv vælger.

Sådan ser det ud i praksis

I porteføljen er Rust kernen under en native beskedklient til macOS: protokollens velafprøvede kerne er skrevet i Rust og stiller en C-grænseflade til rådighed, og klienten, 11.141 linjer Swift fordelt på 53 filer, linker den ind som et statisk arkiv, kompileret fra en fastlåst upstream-revision. Fastlåsningen betyder noget, fordi C-grænsefladen er kontrakten: en kerne i bevægelse ændrer kontrakten uden at ændre en eneste linje Swift. Alt, hvad C-grænsefladen returnerer, har en ejerskabsregel, og tager man fejl af én, får man en lækage eller et nedbrud uden nogen advarsel fra compileren imellem, så en tynd indpakning ejer hver pointer og hver streng, der krydser grænsen, og intet lag over den ser en rå pointer.

At linke den kerne ind i et program betyder at levere alt, hvad den afhænger af, og det er emnet for en licensrevision af 644 Rust-crates. Hver crate erklærer sin licens som et udtryk, og revisionen tjekker hvert udtryk i hele afhængighedsgrafen mod en tilladelsesliste ved hvert build og læser OR som et valg mellem licenser og AND som en pligt til at overholde begge. Alt, der ikke matcher, får buildet til at fejle. Fordi et tjek, der aldrig har fejlet, ser ud præcis som et, der slet ikke kan fejle, blev det bragt til at fejle to gange med vilje: én gang ved at omskrive en crates licens til et vilkår, listen afviser, og én gang ved at flytte den fastlåste kernerevision uden at generere revisionen igen. Den læser kun deklarerede metadata og er ikke juridisk rådgivning.

Dette site har ingen Rust-kildekode selv; det møder sproget på afstand gennem Ruff, der er skrevet i Rust og tjekker hvert Python-script, et commit rører ved.

Hvor det går galt

Det meste besvær med Rust ligger i kanterne af det, compileren tjekker: aritmetik i et optimeret build, grænsen til et andet sprog og kode, der følger med en afhængighed.

Aritmetik løber rundt i et release-build. En tæller, der gik i panik med et overløb under test, bliver i produktion stille til et lille tal. I debug-builds tjekker Rust heltalsaritmetik for overløb og går i panik; et build med `--release` udelader tjekket og løber rundt, så 256 i en `u8` bliver til 0, og bogen betragter det som en fejl at stole på den omløbning. Hvor en værdi kan løbe over, så vælg opførslen udtrykkeligt: `checked_*` returnerer `None`, `saturating_*` stopper ved grænsen, og `wrapping_*` løber rundt med vilje.

```
fn add_stock(on_hand: u8, delivered: u8) -> Option<u8> {
    // None when the sum does not fit, in debug and release alike
    on_hand.checked_add(delivered)
}
```

Hukommelsesfejl vender tilbage ved C-grænsen. Et program, hvis Rust-side bygger uden en eneste advarsel, lækker hukommelse eller går ned inde i et fremmed kald. Funktioner, der er erklæret i en `extern`-blok, er usikre at kalde, fordi det andet sprog ikke håndhæver nogen af Rusts regler, så hvem der frigiver hver pointer, er en konvention, som compileren ikke kan se. Sæt én indpakning til at styre hver pointer og streng, der krydser grænsen, skriv hver ejerskabsregel ned ved siden af kaldet, og udstil kun en sikker grænseflade over den.

En panik når C, og processen afbrydes. En Rust-funktion, der kaldes fra C, går i panik, og i stedet for en fejlværdi stopper hele programmet. Siden Rust 1.81, der udkom den 5. september 2024, afbryder en panik, der forsøger at forlade en funktion erklæret med den almindelige `"C"`-grænseflade, processen. Gør fejl til returværdier, før de når grænsen, og erklær kun en funktion med `"C-unwind"`, hvor koden på den anden side forventer, at stakken afvikles.

En rutineopdatering ændrer licensen for hele buildet. Et versionsløft tre niveauer nede trækker en crate ind, hvis licens produktet ikke kan acceptere, og intet i buildet siger det. Hver crate erklærer sin egen licens, og forpligtelserne

lægges sammen på tværs af hele grafen. Tjek hver crates udtryk mod en tilladelsesliste ved hvert build, lad alt, der ikke matcher, fejle, og bryd tjekket med vilje én gang for at bevise, at det slår til.

Læs mere

- The Rust Programming Language — den officielle bog, fra installation over ejerskab og fejlhåndtering til usikker kode.
- What Is Ownership? — de tre ejerskabsregler, og hvordan hukommelse frigives uden en garbage collector.
- Unsafe Rust — de fem usikre evner, rå pointere og kald til C gennem `extern`.
- The Manifest Format — felterne `edition`, `rust-version` og `license` i `Cargo.toml`.
- Announcing Rust 1.0 — de oprindelige mål og det stabilitetsløfte, der begyndte med 1.0.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er Rust?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/rust/