Rust: Programování s nulovými náklady na bezpečnost

Rust je systémový programovací jazyk, který kombinuje výkon nízkoúrovňových jazyků s bezpečností a ergonomií moderních vysokoúrovňových jazyků. Jeho hlavním cílem je eliminovat chyby typu „segmentation fault“ a úniky paměti (memory leaks) už v době kompilace, aniž by k tomu potřeboval Garbage Collector (GC).

1. Revoluční koncept: Vlastnictví (Ownership)

Největší inovací Rustu je systém **Ownership**, který nahrazuje automatickou správu paměti (jako v JS) i manuální správu (jako v C++).

Tři pravidla vlastnictví:

```
1. Každá hodnota v Rustu má proměnnou, která je jejím **vlastníkem**.
2. V jeden okamžik může existovat pouze **jeden vlastník**.
3. Když vlastník odejde z rozsahu platnosti (scope), hodnota je
**automaticky uvolněna** z paměti.
```

Půjčování (Borrowing) a Lifetime

Aby mohl kód efektivně pracovat s daty bez jejich neustálého kopírování, používá Rust reference:

- **Imutabilní reference (&T):** Můžete mít neomezeně mnoho čtenářů.
- **Mutabilní reference (&mut T):** Můžete mít pouze jednoho zapisovatele v daný čas (eliminace Data Race).

2. Architektura a kompilace

Rust využívá infrastrukturu LLVM, což mu dává přístup ke špičkovým optimalizacím a podpoře mnoha architektur (x86, ARM, WebAssembly).

Kompilační proces:

```
1. **Rustc (Kompilátor):** Zpracuje zdrojový kód a provede syntaktickou
analýzu.
2. **Borrow Checker:** Unikátní fáze, která kontroluje pravidla vlastnictví
a životnosti (lifetimes). Pokud kód není bezpečný, kompilace skončí chybou.
3. **MIR (Mid-level IR):** Mezi-reprezentace pro optimalizace specifické pro
Rust.
```

```
4. **LLVM IR:** Převod do mezikódu [[LLVM]], který se následně zkompiluje do strojového kódu.
```

3. Ekosystém a nástroje

Rust je známý svou vynikající sadou nástrojů, která je součástí instalace:

- **Cargo:** Správce balíčků a sestavovací systém (obdoba npm v JS). Stará se o závislosti, kompilaci i testování.
- **Rustup:** Nástroj pro správu verzí jazyka a toolchainů.
- **Crates.io:** Centrální repozitář knihoven (v Rustu se knihovnám říká „crates“).
- **Clippy:** Linter, který upozorňuje na neefektivní nebo neidiomatický kód.

4. Bezpečnost vs. Výkon (Zero-cost Abstractions)

Rust se drží principu, že abstrakce by neměly zpomalovat výsledný program.

- **Nulové náklady na bezpečnost:** Kontrola paměti probíhá při kompilaci, nikoliv za běhu.
- **Pattern Matching:** Mocný nástroj pro větvení logiky, který kompilátor hlídá, zda jste pokryli všechny možnosti.
- **Error Handling:** Rust nepoužívá výjimky (exceptions). Místo toho používá typy `Result<T, E>` a `Option<T>`, které nutí vývojáře explicitně řešit chyby.

5. Rust v moderním IT světě

Díky své spolehlivosti proniká Rust do kritické infrastruktury:

```
1. **Linux Kernel:** Rust se stal druhým oficiálním jazykem pro psaní ovladačů v jádře [[Linux]]u (hned po C).
2. **WebAssembly (Wasm):** Rust je nejoblíbenějším jazykem pro psaní vysoce výkonného kódu, který běží v prohlížeči (např. grafické editory, hry).
3. **Cloud & Networking:** Projekty jako Firecracker (AWS) nebo síťové stacky Cloudflare sází na Rust kvůli bezpečnosti proti útokům na přetečení bufferu.
```

6. Příklad kódu

Ukázka bezpečné práce s polem a pattern matching:

```
fn main() {
    let cisla = vec![1, 2, 3];

    // Bezpečný přístup k prvku, který nemusí existovat
    match cisla.get(5) {
```

```
        Some(hodnota) => println!("Nalezeno číslo: {}", hodnota),
        None => println!("Prvek na této pozici neexistuje."),
    }
}
```

7. Srovnání: Rust vs. C++

Vlastnost	Rust	C++
Správa paměti	Automatická (Ownership), bez GC.	Manuální (RAII / Smart Pointers).
Bezpečnost	Garantovaná kompilátorem (Safe Rust).	Na zodpovědnosti programátora.
Rychlost vývoje	Pomalejší start (boj s Borrow Checkerem).	Rychlejší prototypování, delší ladění chyb.
Ekosystém	Moderní, integrovaný (Cargo).	Fragmentovaný (CMake, Conan, Makefile).

—

> **Víte, že?** Rust obsahuje klíčové slovo `unsafe`. To umožňuje vývojáři „vypnout“ některé kontroly v případech, kdy je nutný přímý přístup k hardwaru, ale tato místa jsou v kódu jasně označená a izolovaná.

Související: C++, LLVM, Linux, WebAssembly