IT doupě

# Rust – Rychlost céček a syntaxe podobná pythonu

Čím déle se zajímám o programování, tím častěji narážím na Rust – jazyk, který se prezentuje jako hybrid mezi „low levelem“ (vysoký výkon, těžká syntax a vlastní správa paměti) a „high levelem“ (paměť programu skenuje garbage collector, z toho vyplývá nižší výkon) a mimo to patří podle mého oblíbeného webu, kde kradu zdrojáky, mezi nejmilovanější jazyky. Dá se v něm psát vše, kernely a microcontrollery počínaje a aplikacemi a webem konče a i to je zřejmě důvod, proč se říká, že v něm bude jednou přepsáno úplně vše. Proto jsem se ho rozhodl během prázdnin prozkoumat a tady vám předkládám výpis toho, co jsem o něm zjistil a proč byste mu možná také měli dát šanci.

Jednou z jasných výhod je bezpochyby rychlost. Zatímco většina vyšších jazyků používí pro uvolňování paměti *garbage collector*, který kód konstantně skenuje v pozadí a vše brzdí, Rust má díky vlastnictví a referencím přehled o všech proměnných a pokud se nějaká přestane využívat, uvolní ji. Sice je s tímto postupem malinko obtížnější kód psát, protože vám jazyk nedovolí bez přemýšlení přesouvat proměnné z jednoho místa na druhé, stále je to ale lepší, než paměť uvolňovat manuálně nebo nějak omezovat výkon programu.

Dalším důležitým faktorem je bezpečnost. Jazyk je staticky typovaný, takže jsou typy proměnných kontrolovány už při kompilaci kódu a nestane se vám, že se pokusíte sečíst string s číslem (mluvím o tobě, javascripte). Když už jsme u té kompilace, kompilátor bude váš nejlepší kamarád. Nejen že kód nezkompiluje, pokud v něm uděláte jakoukoliv chybu, ale dokonce vám i napíše, jak ji spravit. V jazyku neexistuje nic jako null neboli prázdná hodnota a místo try/catch bloků můžete použít Result a chytat chyby daleko efektivněji. Někdy je ale potřeba udělat něco nebezpečného, třeba pracovat s pamětí na přímo nebo s jejími adresami. Aby se kód zkompiloval, stačí kód zabalit do unsafe {} bloku a kompilátor vás nechá na pokoji.

I přesto, že je základní *std* knihovna docela ořezaná a spousta věcí v ní chybí, můžete si potřebné moduly (v Rustu se jim říká *crates*) přidat přes <https://crates.io/>. Je jich tu kolem 80ti tisíc a denně přibývají nové.

Pokud jste to dočetli až sem, jistě vás zajímá, jak se k jazyku dostat. Je to jednoduché (možná až moc) – stáhněte si rustup z této stránky <https://rustup.rs/> a po otevření terminálu stiskněte jedničku. Otevřete si VSCodium (nebo jiné preferované IDE) a do terminálu napište „*cargo new jmeno\_programu“*. Někde v systému (patrně v domovské složce) se vám vytvoří nový rust projekt s předvygenerovanou šablonou. Nás bude zajímat složka *src* (místo pro zdrojáky) a soubor cargo.toml, který obsahuje informace o vašem projektu a kde pod *[dependencies]* můžete přidávat nové knihovny ve formátu ***název = "verze"***. Kód spustíte pomocí příkazu *cargo run*.

No, a to by bylo k Rustu vše*.* Mezi jeho největší problémy patří těžší syntax, která jazyk znepřístupňuje nováčkům, pouze částečná podpora globálních proměnných (koho by napadlo, že můžou být nebezpečné…) a taky posouvání a půjčování proměnných, kvůli čemu musíte změnit styl, kterým kód píšete. Nic z toho ale není tak strašné a jeho výhody, jako rychlost, bezpečnost, univerzálnost a použití na všech platformách tyto problémy zcela převažuje. V dalším čísle vám ukážu, jak v něm napsat něco smysluplného a pokud už se nemůžete dočkat, čekání vám ukrátí maskot jazyka, kterého náš dvorní grafik Jerry nakreslil.

//krab, měl by přijít v neděli