Fefes Blog

Wer schöne Verschwörungslinks für mich hat: ab an felix-bloginput (at) fefe.de!

Fragen? Antworten! Siehe auch: Alternativlos

Thu Jun 22 2017

- [l] D soll Teil von gcc werden.

  D ist eine Programmiersprache, die sich als Konkurrent zu C++ sieht, und als Nachfolger von C. Die Featureliste klingt auch erst mal nicht schlecht, aber es hat mich persönlich nicht überzeugen können. C++ hat sich ja als Design-Richtlinie entschieden, nur Abstraktionen in der Sprache zu machen, die "nichts kosten", und in der Library im Standard vorzugeben, welche asymptotische Laufzeit die Algorithmen haben sollen. So kann man sich als Programmierer im Wesentlichen darauf verlassen, dass die Laufzeit vorhersagbar und deterministisch bleibt. Überraschungen gibt es jedenfalls nur selbstverschuldete :-)

  Bei D ist genau das über Bord geworfen worden. Das Ziel war anscheinend, ein C++ zu kriegen, das sich mehr wie eine Skriptsprache anfühlt. Problem: An der Stelle kann D nicht mit Go mithalten. Und C++-Programmierer gewinnt es mit dem Ansatz auch nicht viele.

  So hat D Garbage Collection, aber ohne die fanatischen Optimierungen des Go-Teams. D hat dynamische Strings und assoziative arrays, aber als Feature der Sprache, nicht des Runtimes. D nimmt die schlechten Ideen von C++ mit (Template Metaprogramming, Operator Overloading) aber nicht die Vorteile (deterministisches Laufzeitverhalten, zero-cost abstractions).

  Ich hacke auf dem deterministischen Laufzeitverhalten so rum, weil das der Grund ist, wieso heute immer noch viele Realtime-Geschichten in C gemacht werden. Mit striktem Regelkorsett oben drüber, wie "keine Rekursionen" und so. Die Lösung von D? Sie behaupten einfach, man könne in D auch systemnah programmieren. Behaupten reicht nicht, liebe D-Leute.

  Ich will D nicht schlechter machen als es ist. D hat auch gute Ideen. Design by Contract zum Beispiel, das "synchronized"-Keyword, … ist nicht alles schlecht. Und vielleicht wird D-Code ja mit dem Backend von gcc sogar schneller als Go-Code. Auf der anderen Seite hat auch Go ein gcc-Backend. Ich glaube, D ist von Go getötet worden, und gcc macht hier einen Fehler. gcc schleppt schon jetzt mehrere Frontends mit sich herum, die dann nicht ordentlich mitgepflegt werden. java, go, ada, und jetzt dann auch noch D. Und die Probleme im C-Teil bleiben dann liegen (beispielsweise das fehlende Stack Probing, das ich die Tage ansprach).

  Update: Ich sollte das vielleicht knackiger formulieren. Die Vorteile, die D über C++ hat, rechtfertigen dem Umstieg nicht, weil man das zum Großteil auch in C++ haben kann, und dann verliert man nicht seine reingesteckte Erfahrung und den Zugriff auf die ganzen existierenden Libraries. Leute ohne C++-Ballast würden eh lieber gleich zu Go greifen. Und Leute, die die sichere Programmierung haben wollen, mit der D Werbung macht, können das bei Rust haben, ohne die undeterministischen Laufzeiten oder Abstraktionskosten schlucken zu müssen.