|  |  |
| --- | --- |
| Rust de toekomst van de opleiding? | Abstract  Dit onderzoek gaat over de toepasbaarheid van de taal Rust. Er wordt onderzocht of Rust een goede aanvullende taal is voor de opleiding HBO-ICT met afstudeerrichting Technische Informatica. Eerst zal er gekeken worden naar wat de taal Rust te bieden heeft later wordt er gekeken waar de taal Rust mogelijk een plek zou kunnen hebben binnen de opleiding  Mike Hilhorst  1676029 |

# Beantwoording deelvragen

Hier worden de deelvragen beantwoordt

## Voor welke platformen is/wordt Rust ontwikkeld?

Rust ondersteunt vele platformen. Denk aan bijvoorbeeld windows, MacOS, Linux en embedded microcontrollers. Rust ondersteunt deze platvormen op verschillende niveaus. Er zijn 4 niveaus, deze verschillen per ondersteuning denk hierbij aan ondersteuning voor de standaard bibliotheek, Rustc en cargo.   
Niveau 3 is per ongeluk ontstaan en nieuwe platformen zullen dit niveau nooit krijgen.

**Niveau 1**

Dit niveau is goed ondersteunt en kan gezien worden als “Gegarandeerd werkend”.   
Deze platformen hebben de volgende eigenschappen:

* Officiële binary’s worden geleverd voor deze platformen.
* Automatische testen bestaan voor deze platformen.
* Veranderingen in de taal zijn master worden pas doorgelaten na testen geslaagd zijn
* Documentatie over gebruik is beschikbaar voor deze platformen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Target | std | rustc | cargo | notes |
| i686-apple-darwin | ✓ | ✓ | ✓ | 32-bit OSX (10.7+, Lion+) |
| i686-pc-windows-gnu | ✓ | ✓ | ✓ | 32-bit MinGW (Windows 7+) |
| i686-pc-windows-msvc | ✓ | ✓ | ✓ | 32-bit MSVC (Windows 7+) |
| i686-unknown-linux-gnu | ✓ | ✓ | ✓ | 32-bit Linux (2.6.18+) |
| x86\_64-apple-darwin | ✓ | ✓ | ✓ | 64-bit OSX (10.7+, Lion+) |
| x86\_64-pc-windows-gnu | ✓ | ✓ | ✓ | 64-bit MinGW (Windows 7+) |
| x86\_64-pc-windows-msvc | ✓ | ✓ | ✓ | 64-bit MSVC (Windows 7+) |
| x86\_64-unknown-linux-gnu | ✓ | ✓ | ✓ | 64-bit Linux (2.6.18+) |

**Niveau 2**   
Dit niveau is ondersteunt en kan gezien worden als “Gegarandeerd bouwend”.   
Deze platformen hebben de volgende eigenschappen:

* Officiële binarys worden geleverd voor deze platformen.
* Automatische testen zijn opgezet voor deze platformen, maar deze kunnen misschien niet gerund worden.
* Veranderingen in de taal zijn master worden goedgekeurd door platform bouwen. Voor sommige talen houdt dit in dat alleen de standaard bibliotheek gecompileerd moet worden voor andere Rustc en Cargo ook.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Target | std | rustc | cargo | notes |
| aarch64-apple-ios | ✓ |  |  | ARM64 iOS |
| aarch64-fuchsia | ✓ |  |  | ARM64 Fuchsia |
| aarch64-linux-android | ✓ |  |  | ARM64 Android |
| aarch64-unknown-linux-gnu | ✓ | ✓ | ✓ | ARM64 Linux |
| aarch64-unknown-linux-musl | ✓ |  |  | ARM64 Linux with MUSL |
| arm-linux-androideabi | ✓ |  |  | ARMv7 Android |
| arm-unknown-linux-gnueabi | ✓ | ✓ | ✓ | ARMv6 Linux |
| arm-unknown-linux-gnueabihf | ✓ | ✓ | ✓ | ARMv6 Linux, hardfloat |
| arm-unknown-linux-musleabi | ✓ |  |  | ARMv6 Linux with MUSL |
| arm-unknown-linux-musleabihf | ✓ |  |  | ARMv6 Linux, MUSL, hardfloat |
| armv5te-unknown-linux-gnueabi | ✓ |  |  | ARMv5TE Linux |
| armv7-apple-ios | ✓ |  |  | ARMv7 iOS, Cortex-a8 |
| armv7-linux-androideabi | ✓ |  |  | ARMv7a Android |
| armv7-unknown-linux-gnueabihf | ✓ | ✓ | ✓ | ARMv7 Linux |
| armv7-unknown-linux-musleabihf | ✓ |  |  | ARMv7 Linux with MUSL |
| armv7s-apple-ios | ✓ |  |  | ARMv7 iOS, Cortex-a9 |
| asmjs-unknown-emscripten | ✓ |  |  | asm.js via Emscripten |
| i386-apple-ios | ✓ |  |  | 32-bit x86 iOS |
| i586-pc-windows-msvc | ✓ |  |  | 32-bit Windows w/o SSE |
| i586-unknown-linux-gnu | ✓ |  |  | 32-bit Linux w/o SSE |
| i586-unknown-linux-musl | ✓ |  |  | 32-bit Linux w/o SSE, MUSL |
| i686-linux-android | ✓ |  |  | 32-bit x86 Android |
| i686-unknown-freebsd | ✓ | ✓ | ✓ | 32-bit FreeBSD |
| i686-unknown-linux-musl | ✓ |  |  | 32-bit Linux with MUSL |
| mips-unknown-linux-gnu | ✓ | ✓ | ✓ | MIPS Linux |
| mips-unknown-linux-musl | ✓ |  |  | MIPS Linux with MUSL |
| mips64-unknown-linux-gnuabi64 | ✓ | ✓ | ✓ | MIPS64 Linux, n64 ABI |
| mips64el-unknown-linux-gnuabi64 | ✓ | ✓ | ✓ | MIPS64 (LE) Linux, n64 ABI |
| mipsel-unknown-linux-gnu | ✓ | ✓ | ✓ | MIPS (LE) Linux |
| mipsel-unknown-linux-musl | ✓ |  |  | MIPS (LE) Linux with MUSL |
| powerpc-unknown-linux-gnu | ✓ | ✓ | ✓ | PowerPC Linux |
| powerpc64-unknown-linux-gnu | ✓ | ✓ | ✓ | PPC64 Linux |
| powerpc64le-unknown-linux-gnu | ✓ | ✓ | ✓ | PPC64LE Linux |
| s390x-unknown-linux-gnu | ✓ | ✓ | ✓ | S390x Linux |
| sparc64-unknown-linux-gnu | ✓ |  |  | SPARC Linux |
| sparcv9-sun-solaris | ✓ |  |  | SPARC Solaris 10/11, illumos |
| wasm32-unknown-unknown | ✓ |  |  | WebAssembly |
| wasm32-unknown-emscripten | ✓ |  |  | WebAssembly via Emscripten |
| x86\_64-apple-ios | ✓ |  |  | 64-bit x86 iOS |
| x86\_64-fuchsia | ✓ |  |  | 64-bit Fuchsia |
| x86\_64-linux-android | ✓ |  |  | 64-bit x86 Android |
| x86\_64-rumprun-netbsd | ✓ |  |  | 64-bit NetBSD Rump Kernel |
| x86\_64-sun-solaris | ✓ |  |  | 64-bit Solaris 10/11, illumos |
| x86\_64-unknown-cloudabi | ✓ |  |  | 64-bit CloudABI |
| x86\_64-unknown-freebsd | ✓ | ✓ | ✓ | 64-bit FreeBSD |
| x86\_64-unknown-linux-gnux32 | ✓ |  |  | 64-bit Linux |
| x86\_64-unknown-linux-musl | ✓ |  |  | 64-bit Linux with MUSL |
| x86\_64-unknown-netbsd | ✓ | ✓ | ✓ | NetBSD/amd64 |
| x86\_64-unknown-redox | ✓ |  |  | Redox OS |

**Niveau 3**

Dit niveau is ondersteunt en kan gezien worden als “Gegarandeerd bouwend”, maar dan zonder beschikbare versies via Rustup. Er worden geen automatische testen gerund dus er wordt niet gegarandeerd dat de geproduceerd build werkt, maar meestal werken deze platformen best goed. Deze platformen hebben de volgende eigenschappen:

* Automatische testen zijn opgezet voor deze platformen, maar deze kunnen misschien niet gerund worden.
* Veranderingen in de taal zijn master worden goedgekeurd door platform bouwen. Voor sommige talen houdt dit in dat alleen de standaard bibliotheek gecompileerd moet worden voor andere Rustc en Cargo ook.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Target | std | rustc | cargo | notes |
| aarch64-unknown-cloudabi | ✓ |  |  | ARM64 CloudABI |
| armv7-unknown-cloudabi-eabihf | ✓ |  |  | ARMv7 CloudABI, hardfloat |
| i686-unknown-cloudabi | ✓ |  |  | 32-bit CloudABI |
| powerpc-unknown-linux-gnuspe | ✓ |  |  | PowerPC SPE Linux |
| sparc-unknown-linux-gnu | ✓ |  |  | 32-bit SPARC Linux |

**Niveau 4**

De Rust codebase heft ondersteuning voor deze platformen, maar worden niet gebouwd of getest. Deze platformen kunnen misschien niet werken, verder zijn er geen officiële versies beschikbaar

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Target | std | rustc | cargo | notes |
| i686-pc-windows-msvc (XP) | ✓ |  |  | Windows XP support |
| i686-unknown-haiku | ✓ |  |  | 32-bit Haiku |
| i686-unknown-netbsd | ✓ |  |  | NetBSD/i386 with SSE2 |
| le32-unknown-nacl | ✓ |  |  | PNaCl sandbox |
| mips-unknown-linux-uclibc | ✓ |  |  | MIPS Linux with uClibc |
| mipsel-unknown-linux-uclibc | ✓ |  |  | MIPS (LE) Linux with uClibc |
| msp430-none-elf | \* |  |  | 16-bit MSP430 microcontrollers |
| sparc64-unknown-netbsd | ✓ | ✓ |  | NetBSD/sparc64 |
| thumbv6m-none-eabi | \* |  |  | Bare Cortex-M0, M0+, M1 |
| thumbv7em-none-eabi | \* |  |  | Bare Cortex-M4, M7 |
| thumbv7em-none-eabihf | \* |  |  | Bare Cortex-M4F, M7F, FPU, hardfloat |
| thumbv7m-none-eabi | \* |  |  | Bare Cortex-M3 |
| x86\_64-pc-windows-msvc (XP) | ✓ |  |  | Windows XP support |
| x86\_64-unknown-bitrig | ✓ | ✓ |  | 64-bit Bitrig |
| x86\_64-unknown-dragonfly | ✓ | ✓ |  | 64-bit DragonFlyBSD |
| x86\_64-unknown-haiku | ✓ |  |  | 64-bit Haiku |
| x86\_64-unknown-openbsd | ✓ | ✓ |  | 64-bit OpenBSD |
| [NVPTX](https://github.com/japaric/nvptx#targets) | \*\* |  |  | --emit=asm generates PTX code that runs on NVIDIA GPUs |

\* Dit zijn “bare-metal” microcontrollers doelen, deze hebben alleen toegang tot de kern bibliotheek en niet de standaard bibliotheek.

\*\* Er is back-end ondersteuning voor deze doelen, maar geen doelwit dat (nog) in rustc is ingebouwd. Je moet uw eigen doelspecificatiebestand schrijven. Deze doelen ondersteunen alleen de kernbibliotheek.

Dit zijn niet de enige platforms die Rust kan compileren! Dat zijn degenen met ingebouwde doeldefinities en/of standaard bibliotheekondersteuning. Tijdens het linken van de kernbibliotheek, kan Rust zich richten op "bare metal" in de x86-, ARM-, MIPS- en PowerPC-families, hoewel hiervoor mogelijk aangepaste doelspecificaties moeten worden gedefinieerd. Al dergelijke scenario's zijn niveau 4.

## Hoe werkt Rust intern op low- en highlevel?

## Vanuit welke talen leent rust concepten, wat dat zijn deze concepten?

Rust is een relatieve nieuwe taal en leent concepten uit al bestaande talen.

Waarom …..

Wat..

Sommige geleende concepten zijn verouderd en zijn dus weggevallen. Dit zijn de huidige nog gebruikte geleende concepten.:

|  |  |
| --- | --- |
| Taal | Concepten |
| SML, OCaml | algebraic data types,  pattern matching,  type inference,  semicolon statement separation |
| C++ | references,  Resource Acquisition Is Initialization(RAII),  smart pointers,  move semantics,  monomorphization,  memory model |
| ML Kit, Cyclone | region based memory management |
| Haskell (GHC): | typeclasses, type families |
| Newsqueak, Alef, Limbo | channels, concurrency |
| Erlang | message passing, thread failure |
| Swift | optional bindings |
| Scheme | hygienic macros |
| C# | attributes |
| [Unicode Annex #31](http://www.unicode.org/reports/tr31/) | identifier and pattern syntax |

## Wat zijn de voor- en nadelen van Rust tegenover C++ in de scope van de lesstof van de opleiding HBO-ICT met afstudeerrichting Technische Informatica?

Een van de voordelen kan performance zijn. Als een programma kleiner en sneller is dan kan die op kleinere procesoren.

Source: in welke taal de code is uit gevoerd.

Seconde: uitvoer tijd van de code.

Geheugen: geheugen gebruikt tijdens de uitvoer (in bytes).

Grootte: grootte van de source files (in bytes).

CPU lading: hoeveel elke kern beladen werd tijdens de uitvoer.

|  |  |  |  |
| --- | --- | --- | --- |
| [Reverse-complement](https://benchmarksgame-team.pages.debian.net/benchmarksgame/performance/revcomp.html) | | | |
| Source | Seconde | Geheugen | Grootte | CPU lading | | | |
| C++ | 2,94 | 980.524 | 2.280 | 15% | 50% | 52% | 40% |
| Rust | 1,60 | 995.212 | 1.376 | 24% | 25% | 96% | 30% |
| Python 3 | 16,76 | 1.005.252 | 814 | 65% | 21% | 44% | 17% |
| Haskell | 5,67 | 501.336 | 1.020 | 26% | 12% | 12% | 40% |

|  |  |  |  |
| --- | --- | --- | --- |
| [N-body](https://benchmarksgame-team.pages.debian.net/benchmarksgame/performance/revcomp.html) | | | |
| Source | Seconde | Geheugen | Grootte | CPU lading | | | |
| C++ | 9,42 | 1.712 | 1.763 | 100% | 1% | 2% | 0% |
| Rust | 13,25 | 1.808 | 1.805 | 0% | 0% | 1% | 100% |
| Python 3 | 14min | 8.212 | 1.196 | 91% | 0% | 1% | 9% |
| Haskell | 22,01 | 3.936 | 1.883 | 99% | 99% | 100% | 99% |

|  |  |  |  |
| --- | --- | --- | --- |
| K-nucleotide | | | |
| Source | Seconde | Geheugen | Grootte | CPU lading | | | |
| C++ | 3,83 | 156.104 | 1.624 | 72% | 73% | 98% | 72% |
| Rust | 5,98 | 137.956 | 1.648 | 78% | 49% | 90% | 85% |
| Python 3 | 79,79 | 250.948 | 1.967 | 98% | 96% | 96% | 99% |
| Haskell | 35,51 | 604.996 | 1.486 | 89% | 93% | 90% | 87% |

|  |  |  |  |
| --- | --- | --- | --- |
| Regex-redux | | | |
| Source | Seconde | Geheugen | Grootte | CPU lading | | | |
| C++ | 1,83 | 203.680 | 1.315 | 49% | 87% | 57% | 51% |
| Rust | 2,44 | 194.804 | 765 | 85% | 41% | 20% | 16% |
| Python 3 | 15,56 | 439.964 | 512 | 25% | 92% | 32% | 32% |
| Haskell | Bad output |  |  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Binary-trees | | | |
| Source | Seconde | Geheugen | Grootte | CPU lading | | | |
| C++ | 3,67 | 118.620 | 809 | 75% | 78% | 99% | 76% |
| Rust | 4,14 | 175.692 | 721 | 90% | 90% | 91% | 100% |
| Python 3 | 92,72 | 448.844 | 589 | 87% | 90% | 96% | 87% |
| Haskell | 12,59 | 478.012 | 592 | 95% | 90% | 88% | 94% |

|  |  |  |  |
| --- | --- | --- | --- |
| Fasta | | | |
| Source | Seconde | Geheugen | Grootte | CPU lading | | | |
| C++ | 1,33 | 1.744 | 2.711 | 82% | 82% | 81% | 81% |
| Rust | 1,46 | 3.112 | 1.906 | 84% | 83% | 84% | 89% |
| Python 3 | 62,88 | 680.736 | 1.947 | 60% | 56% | 48% | 62% |
| Haskell | 9,46 | 4.980 | 969 | 100% | 2% | 2% | 3% |

|  |  |  |  |
| --- | --- | --- | --- |
| Spectral-norm | | | |
| Source | Seconde | Geheugen | Grootte | CPU lading | | | |
| C++ | 1,98 | 1.168 | 1.044 | 100% | 99% | 99% | 99% |
| Rust | 1,97 | 2.600 | 1.126 | 100% | 100% | 100% | 99% |
| Python 3 | 193,86 | 50.556 | 443 | 98% | 98% | 99% | 99% |
| Haskell | 4,17 | 3.812 | 987 | 99% | 97% | 99% | 98% |

|  |  |  |  |
| --- | --- | --- | --- |
| Fannkuch-redux | | | |
| Source | Seconde | Geheugen | Grootte | CPU lading | | | |
| C++ | 10,07 | 1.856 | 980 | 99% | 100% | 100% | 95% |
| Rust | 9,87 | 1.848 | 1.020 | 100% | 95% | 100% | 100% |
| Python 3 | 9 min | 48.052 | 950 | 99% | 100% | 97% | 100% |
| Haskell | 18,20 | 4.024 | 842 | 99% | 100% | 95% | 100% |

|  |  |  |  |
| --- | --- | --- | --- |
| Mandelbrot | | | |
| Source | Seconde | Geheugen | Grootte | CPU lading | | | |
| C++ | 1,82 | 29.092 | 1.002 | 98% | 97% | 97% | 100% |
| Rust | 1,74 | 33.712 | 1.332 | 98% | 100% | 98% | 98% |
| Python 3 | 279,86 | 49.334 | 688 | 100% | 100% | 100% | 100% |
| Haskell | 11,60 | 38,744 | 782 | 100% | 100% | 100% | 100% |

|  |  |  |  |
| --- | --- | --- | --- |
| Pidigits | | | |
| Source | Seconde | Geheugen | Grootte | CPU lading | | | |
| C++ | 1,89 | 4,312 | 513 | 1% | 1% | 99% | 1% |
| Rust | 1,74 | 4.520 | 1.366 | 1% | 3% | 0% | 99% |
| Python 3 | 3,51 | 10.500 | 385 | 1% | 1% | 0% | 100% |
| Haskell | 4,20 | 9.724 | 585 | 27% | 8% | 8% | 81% |

Deze testen zijn uitgevoerd op een computer met:

Processor: Intel Q6600 quad-core op 2,4Ghz

RAM: 4GB ram.

OS: Ubuntu™ 18.10 Linux x64 4.18.0-10-generic.

### Hoe is dit gemeten?

Elk programma is uitgevoerd en gemeten bij de kleinste input waarde, de output werd opgeslagen in een bestand om vergeleken te worden met de verwachte output. Zolang de output overeen blijft komen met de verwachte output werd het programma getest met een grotere input. Totdat alle inputs waren doorlopen. Als het programma de verwachte output bleef geven tijdens het testen (binnen een redelijke tijd) dan werd er nog 5 keer gemeten. Tijdens deze testen werd de output geledigd (/dev/null).

Als het programma niet de verwachte output leverde binnen een redelijke tijd dan werd het programma geforceerd gestopt. Als de metingen met kleinere input wel de verwachte uitvoer leverde dan werd deze 5 keer gemeten. Ook tijdens deze testen werd de output geledigd /dev/null).

### Hoe zijn de programma’s getimed?

Elk programma is uitgevoerd als een “Child-process” van een Python script.  
Er werd gebruik gemaakt van “time.time()” voordat het proces werd gestart en na dat deze gesloten was.

### Hoe is het geheugen gebruik gemeten?

Door “GLIBTOP\_PROC\_MEM\_RESIDENT” te samplen voor het hoofd proces en zijn “Child-process” elke 0,2 seconde.

### Hoe is de source file gemeten?

De code werd in een bestand gezet zonder commentaar, extra spaties en extra witregels. Op dit bestand werd een lichte GZip compressie uitgevoerd. De grootte van dit bestand werd gemeten.

### Hoe is de CPU lading gemeten?

De GTop CPU inactief en de GTop CPU totaal werden gemeten voor de opstarten van het “Child-process” en na het afsluiten hiervan. De percentages zijn het totaal min de inactieve waardes.

## Wat zou een student leren van Rust tegenover C++ scope van de opleiding HBO-ICT met afstudeerrichting Technische Informatica?

## Waar zou de taal Rust een plek hebben in de opleiding HBO-ICT met afstudeerrichting Technische Informatica?