Bronnen en aanwijzingen bij Hello World sessies

Bash

- man bash
- https://www.dinkum.nl/software/shell/scripting
- https://github.com/marcdinkum/me-bash-scripts

Je kunt BASH-programma's/commands interactief op de command line typen/programmeren maar een text-file met wat instructies is meestal handiger. Een script begint vaak als een sequence van commando's zoals je die op de command line zou typen.

Zet je script in een text-file, uitvoeren met `bash myscript.sh`

of begin met #! /bin/bash en maak het script executable met `chmod 755`. Dan kun je het uitvoeren alsof het een executable is: ./myscript

C

- IDE: CLion, Xcode, Visual Studio, VS Code, Fleet
- intermediate tools: CMake, Make, Conan, Vspkg
- advanced tools: clang-format, clang-tidy, valgrind, lldb, gdb
- libraries & frameworks: Jack, zlib, GTK.

C++

- IDE: CLion, Xcode, Visual Studio, Fleet, VS Code
- tools: CMake, Make, Ninja, Conan, Vspkg
- Advanced tools: clang-format, clang-tidy, valgrind, lldb, gdb
- Frameworks & libraries: Qt, Juce, ImGui, Boost, nholmannjson, {fmt}, Catch2, CTRE, Google Test
- learncpp.com

Compileren en linken op de command line met g++ en make

C en C++ programma's moeten gecompileerd en gelinkt worden om uitgevoerd te kunnen worden omdat ze direct op het OS draaien.

`hello_world.cpp`:

```
#include <iostream>

int main()
{
  std::cout << "Hello world" << std::endl;
}
```

Nu kun je het bouwen (compileren en linken) en uitvoeren:

```
make hello_world
./hello_world

g++ hello_world.cpp
./a.out

g++ -o hello_world hello_world.cpp
./hello_world

g++ -c hello_world.cpp
g++ -o hello_world hello_world.o
./hello_world

g++ -S hello_world.cpp
g++ -c hello_world.s
g++ -o hello_world hello_world.o
./hello_world
```

download/install requirements

- Een C++ compiler (GCC, Clang, MSVC)

Rust

download/install requirements

- Rustup & Cargo, dat bevat alles

Additional Tools

- IDE: elke IDE of code editor die het Language Server Protocol (LSP) ondersteunt. Alle JetBrains IDE's, VS Code, (neo)vim
- Alle tooling die je nodig hebt zit verwerkt in Rust's package manager, Cargo. Testen, linten, externe libraries integreren, formatteren en runnen zit allemaal in dezelfde tool geïntegreerd, wat ervoor zorgt dat iedereen dezelfde tools voor zich heeft wanneer hij bezig is.
- libraries (crates): Serde, Tokio, Yew, Clap, Plotters, Rocket, Tauri, Cpal, bindings voor alles wat ooit heeft bestaan...

cursus en andere info om zelf verder te gaan

- Rust book: doc.rust-lang.org/book/
- Let's Get Rusty op YouTube
- No Boilerplate op YouTube

Bouwen en uitvoeren

Rust programma's moeten gecompileerd en gelinkt worden om uitgevoerd te kunnen worden omdat ze direct op het OS draaien.

Bij Rust gebruik je doorgaans de Cargo package manager om je project op te zetten en te compileren.

Een nieuw project aanmaken doe je zo:

```
cargo new <name>
```

En het project vervolgens bouwen zo:

```
cargo build
```

En het programma uitvoeren zo:

```
cargo run
```

De configuratie van je project staat in `Cargo.toml`. Hier staat ook welke externe libraries je gebruikt bijvoorbeeld.

Java

openjdk - Java Compiler javac - Java Runtime environment java - debugger - development kit (?) JDK

Bouwen en uitvoeren

```
javac hello_world.java
java hello_world.class
```

Javascript

download/install requirements

- elke moderne browser beschikt over een Javascript-interpreter. Invoke with e.g. CTRL-J
- NodeJs, Dino, ..

Additional tools

- IDE: VS Code, WebStorm, Fleet
- Package managers: npm, yarn
- libraries voor web frontend: VueJs, P5Js https://p5js.org, ReactJs, AngularJs, ThreeJs, NextJs, .. +100000
- libraries voor "native" development: Electron, React Native
- other: TypeScript, Jest, PrettierJs, ESLint, Babel, WebPack, SnowPack.

- https://csd.hku.nl/csd1/languages/jsbook

Uitvoeren met Node.js

```
node js_hello.js
```

Python

Download/Install requirements

- Een Python interpreter (CPython of IPython bijvoorbeeld)
- pip

Additional tools

- IDE: PyCharm, VS Code, Spyder, Fleet
- Other: Pip, Jupyter Notebook, Conda
- Libraries & frameworks: Django, Numpy, Scipy, TensorFlow, Keras, PyTorch, matplotlib, PyGame
- Programming with Mosh op YouTube
- https://www.learnpython.org

Bouwen en uitvoeren

```
python hello_world.py
of stukje bij beetje in een interactieve python interpreter
```

Scheme

https://www.scheme.org

- Racket
- GNU guile
- Lilypond
- Clojure
- Emacs

download/install requirements

GNU Guile https://www.gnu.org/software/guile https://racket-lang.org

Bouwen en uitvoeren

Racket heeft diverse programma's om Scheme-code uit te voeren: de Racket IDE, racket interactieve shell en racket op de command line.

```
racket hello_world.rkt
of stukje bij beetje in een interactieve scheme interpreter
```

Compileren tot een executable

- Raco compiler

cursus en andere info om zelf verder te gaan

- https://csd.hku.nl/csd1/languages/schemebook
- https://htdp.org

Pure Data en PlugData

Pure Data

- Purr Data (aka PD-L2ork 2.0) https://github.com/agraef/purr-data
- Vanilla PD https://puredata.info
- https://agraef.github.io/purr-data-intro/Purr-Data-Intro.html

PlugData (aanrader!)

- https://github.com/plugdata-team/plugdata

Bouwen en uitvoeren

Draai je patches binnen pd.

Supercollider

download/install requirements

https://supercollider.github.io

- https://csd.hku.nl/csd1/languages/superbook
- https://ccrma.stanford.edu/~ruviaro/texts/A_Gentle_Introduction_To_SuperCollider.pdf

Spin-offs

- Sonic Pi

PHP

download/install requirements

https://www.php.net

Bouwen en uitvoeren

php hello_world.php

SQL

download/install requirements

- https://www.mysql.com
- https://www.postgresql.org
- SQLite : database zonder server, in een file

Bouwen en uitvoeren

Binnen een sql-host, maar wat ook kan: mysql < hello_world.sql