Klemens Morgenstern

E-Mail
`contact@klemens.dev`

Github
`klemens-morgenstern`

Klemens D. Morgenstern

Software Developer

About me

I have a Master's Degree in Electrical Engineering, but a strong passion for C++. Since getting my degree I almost exclusively developed software, and became contributor to boost with boost.process. Because of my electrical engineering background, I like my programming environments to be low-level especially bare-metal.

Education

2006 - 2010, HTW Berlin

Bachelor of Engingeering in Electronical Engineering

2010 - 2013, TU Chemnitz

Master of Science in Microsystems and Microelektronics

Experience

2009 - 2013, *Parttime jobs & Thesis Papers*

- C & C++ for Stm32-Controllers
- Gui development (.net & C++)
- Kalman filters

2013-2016, *Software Developer*, EDC Electronic Design Chemnitz GmbH

- ASIC development tool
- STIL Parser (spirit.qi)
- Unit Testing (Tessy)

2017- now, *Independent Contractor*

Skills

Languages

- C++
- C
- Python
- C#
- Typescript
- SQL

Libraries & Tools

- boost
- pybind11
- Qt
- gcc
- clang
- boost.build
- node.js
- PostgreSQL
- GraphQL

Open-Source Contributions

- boost.process
- boost.dll
- metal.test

Interests

- Modern C++
- Embedded Systems
- System Programming
- Template Meta Programming
- Build Systems

Klemens Morgenstern

E-Mail
`contact@klemens.dev`

Github
`klemens-morgenstern`

Project lead

Qyro, ASIC development tool, *closed-source, 2013-2017*

For the ASIC development at EDC, I developed a tool to read and write data from it's registers, using a custom SPI (done through an FTDI chip). Since most of the ASICs were for sensors, it also had to support high data rates and visualization (Qt, qwt). In addition it had a plugin-system to select the proper ASIC-library and a json-rpc interface for script extensions. The latter was delivered with client code in C# (for import in LabView), Python and Java (for easy usage in Matlab).

Tools

- C++
- Qt & qwt
- FTDI/SPI
- Json-Rpc
- Python
- Java & C#

boost.process, *2016-2018*

Boost 1.64 finally contains a process library after over 10 years of development. I took over the development in 2016, redid the interface and a huge part of the implementation, while keeping many design decision of previous versions. The main challenge was to find the concepts shared between the different platforms and implement low-overhead abstractions, in order to create a portable library.
The library can be found **here** and you can listen to me speak about it **here**.

Tools

- C++ 11
- boost.asio
- posix & win-api

metal.test, *2016-2018*

As part of starting my own venture, I am currently creating a toolset for embedded software development and testing. The idea is to as much as possible through the debugger, which is why we provide a debugger-runner, i.e. a tool that executes the debugger automatically and interacts with it. This then loads plugins which add breakpoints to get more diagnostic data. The best example is our test backend, which will give a binray test-result when executed on it's own, but output very detailed information when used with the debugger-runner.
The toolset is currently in beta and can be found **here**

Tools

- C++14
- gdb
- unit-tests
- ld
- pegtl
- python

Experimental Measurement System, *Confidential, 2010*

I wrote an embedded software in C for the development of a experimental measurement process,, which had to comply high requirements like real-time. The code was highly optimized, which included counting the assembly instructions.

Tools

- C
- stm32f4
- Keil μVision

Klemens Morgenstern

E-Mail
`contact@klemens.dev`

Github
`klemens-morgenstern`

Contributions

boost.dll, *2016*

I've written an addition for boost.dll that allows the import of mangled C++ Symbols at runtime.

Tools

- C++14
- MSVC ABI
- Itanium ABI

pybind11, *2016*

I've written and proposed an extension for pybind11 that allows executing Python code from C++, which was merged with some modifications.

Tools

- C++11
- Python

Other C++ Problems, *2012-2018*

- Run-time to compile-time operations (C++ constexpr, templates)
- Fixed-point template class
- Lock- & allocation-free (multithreaded) buffer
- Static polymorphism

Stm32f4, *2010-2018*

- Coroutine Library & a Talk at Embo++ 2018
- Manual RS-485 implementation using the DMA (C)
- Object-based C++ handler for interrupts
- C++ Wrappers for HAL-drivers
- RAII for hardware initialization

Miscellaneous, *2014-2017*

- Code generator in Java with a C++ Parser based on ANTLR
- Boost.Build Extensions