Ghasem Ramezani

Software Enginner

g1999ramezani@gmail.com +98 910 294 7375 Iran, Tehran gccore

gccore ghasem_ccore gccore gccore ghasem-ramezani

As a Patient Network and Low-Level programmer, I bring a background in computer science and over 6 years of experience in the software industry. My expertise spans various technology fields and stacks, ranging from embedded Linux to back-end servers/services. I have a comprehensive understanding of the entire software life cycle, from planning to deployment. I am dedicated to designing and implementing high-performance, low-latency applications with SOLID principles in mind.

Professional Experience

Arad Co., *Research Center* Apr 2020 – May 2023
Tehran, Iran

I worked mainly on core libraries, the project's build process, and CMake modules, with minor development focused on a customized GIS 3D Engine mostly based on OpenGL 3.3.

- Design a lock-free and blazingly fast MiniDBus.
- Design an easy-to-use wrapper for gRPC++ that eliminates nearly 70% of manual work.
- Improve QCustomPlot to plot 4x30Hz data series only by CPU
- Enhance QWT for OpenGL-based plotting and customize it for statistical applications using various tools
- Design an easy-to-use Template project for improving CI/CD and project build process.

Baqer al-Uloom, *Research Center* Feb 2018 – Mar 2020
Tehran, Iran

Design and implement telecommunication applications with high-performance computing on a customized RT Linux Kernel, while ensuring a fully functional and high-performance port on other commercial kernels such as WinNT 2000. My responsibilities included maintaining hardware drivers and CR. Our primary technologies were C++11/17 and GNU Radio. I improved the hardware driver, resulting in a 60% increase in network throughput and a reduction in CPU usage.

- Design a real-time scheduler for automating some applications.

- Improve applications for cross-compiling through Windows 7 and GNU/Linux.
- Improve CVS and CI/CD which eliminates 80% of manual work.

Skills

Low Level Programming Languages
- *Proficient in C(11-N1548)*
- *Proficient C++(from 11-N3690 to 17-N4713)*
- *Proficient in Rust(latest release)*
- *Proficient in Go(latest release)*
- *Experienced with x86 Intel ASM syntax*

Mathematics
- *Experienced with Linear Algebra*
- *Familiar with Numeric Theory*
- *Familiar with Set Theory*

Profiling And Monitoring
- *Proficient in perf*
- *Experienced with Hotspot and Orbit*
- *Familiar with Prometheus*

Networking
- *Experienced with TCP and UDP over IP packets*
- *Experienced with Multiplexing and Preforking*
- *Experienced with high-rate streaming channels*

Databases
- *Familiar with MongoDB*
- *Familiar with PostgreSQL*
- *Experienced with Redis*
- *Familiar with SQLite*

Containers
- *Experienced with Docker and familiar with Redis image and advanced futures like Swarm mode*
- *Familiar with LXC and LXD*

Operating Systems
- *Experienced with Linux*
- *Familiar with WinNT*
- *Familiar with OpenBSD*

Distributed Architecture
- *Familiar with HLA IEEE 1516*
- *Experienced with NASA HLA Implementation*

Languages

English	●●●○○	Russian	●●○○○
German	●○○○○	Persian	●●●●●

Projects

Real-time Scheduler, *An automation tool*

We had a real-time application that required a scheduler to perform specific actions in real-time. This project was written in C++11 using Boost and Qt5, designed for Windows, Linux, and Android.

Awesome Plot, *Statistical plot*

A plot library, based on QCustomPlot and QWT, designed to work with large amounts of data on both CPU and OpenGL, featuring highly customized tools for statistical applications. This library is written in C++17 with a port to Rust.

MiniDBus, *Inter-plugin communication*

We had many unknown runtime-loaded plugins that required a fast and easy-to-use data and command communication API. MiniDBus designed a lock-free hash map and an easy-to-use API.

gRPC Without Pain, *gRPC++ Wrapper*

To use gRPC with C++, a considerable amount of manual work and care is needed. I designed and implemented this library to simplify our tasks, improve productivity, and reduce team debugging time.

USRP With USB 3.X, *Driver update*

I did some non-public(unfortunately) changes to the USRP C++ driver to detect USB 3.X.

CMake Common, *CMake utility*

Highly customizable scripts completely integrated with Git to fetch/patch/build/generate QMake targets in CMake.

CMake Template, *CMake root file template*

I carefully designed the root CMake configuration for Shared Libraries and Executable targets. I configured the targets comprehensively, considering their types, and ensured that CMake automatically generates all the configurations related to installation and uninstallation rules.

Explicit QWidget, *A design implementation*

C++ gives the user complete power to create disaster, especially with the Qt framework. I designed an explicit architecture to define the QWidgets classes.