Motivation
Measurement of code coverage is a key tool to determine which parts of a program are actually executed during unit testing. Ideally code coverage should be high, to ensure that each source line has been tested at least once. In practice, corner-case error conditions like out of memory errors, or unexpected failure of library functions complicate testing, and prevent us from reaching high code coverage.

In this project we want to design, and implement flexible a compiler extension for the Low-Level Virtual Machine (LLVM) framework that allow us to insert instrumentation code for simulating corner-case errors into existing programs. The extensions should be agnostic with respect to the underlying hardware, and operating system.

Project Description
Goals
• Investigate the feasibility of source-to-source transformation on top of LLVM’s clang compiler
• Design and implement a prototype of the LLVM-based instrumentation tool (either as bit-code or source-to-source transformation)

Background
• Good C and C++ programming skills
• Interest in compiler internals

Deliverables
• Tool prototype (source code in GIT repository)
• Documentation/Tech Report
• Short presentation (15-20 min)

Scope and Credits
Studies: INF SEW TEL MATH
The scope, effort, and credits of this project are scalable for 1-2 participants.

Advisor/Contact
Johannes.Winter@iaik.tugraz.at

Graz University of Technology
Institute for Applied Information Processing and Communications
http://www.iaik.tugraz.at/