L3: Building, Testing and Debugging Scientific Software

P. de Oliveira Castro, M. Jam

September 1, 2025

Master Calcul Haute Performance et Simulation - GLHPC | UVSQ

Objectives

- Build systems: Advanced Makefiles, introduction to CMake for managing multi-file and multi-platform projects.
- Debugging: GDB, Valgrind for detecting memory errors and leaks.
- Software testing:
  - Principles: Unit testing, integration testing.
  - Test frameworks in C (e.g., Unity).
  - Importance of testing for regression prevention and validation.
- Code documentation: Doxygen.

Makefiles

Dependency Management

- How to determine which files have changed?

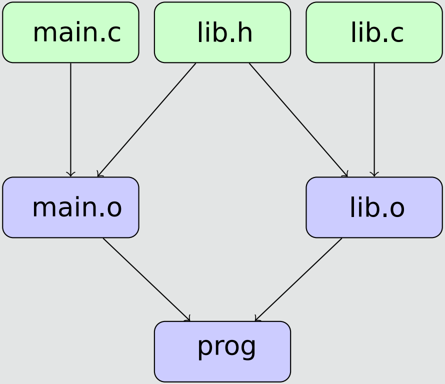


Figure 1: makefile-dependencies

- **dependencies:** `main.o` depends on changes in `lib.h`

Makefile

- A `Makefile` uses a declarative language to describe targets

CMake

Why CMake?

- **Advantages of Makefiles:**
  - Simplicity and transparency.
  - No additional tools required.
  - Direct control over the build process.
- **Advantages of CMake:**
  - Cross-platform support (Linux, Windows, macOS).
  - Generates build files for multiple build systems (Make, Ninja, etc.).
  - Modular and target-based design.
  - Built-in support for testing, installation, and packaging.

General Design of CMake

- **CMake as a Meta-Build System:**
  - Generates build files for different generators (e.g., Make, Ninja).
  - Abstracts platform-specific details.
- **Workflow:**
  1. Write `CMakeLists.txt` to define the project.
  2. Configure the project:

     ```
     cmake -B build
     ```

  3. Build the project:

     ```
     cmake --build build
     ```

Debugging Tools

GDB: GNU Debugger

Valgrind: memory debugging and leak detection

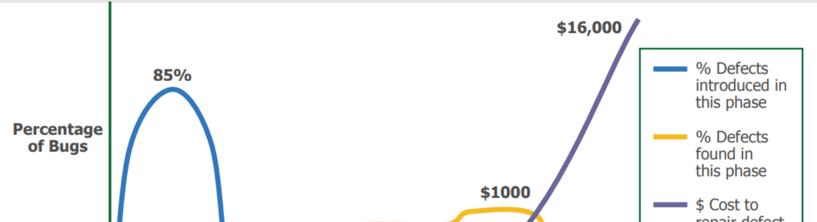
Other tools: ASAN, UBSAN

Software Testing

Importance of Software Testing

- 1996: Ariane-5 self-destructed due to an unhandled floating-point exception, resulting in a $500M loss.
- 1998: Mars Climate Orbiter lost due to navigation data expressed in imperial units, resulting in a $327.6M loss.
- 1988-1994: FAA Advanced Automation System project abandoned due to management issues and overly ambitious specifications, resulting in a $2.6B loss.
- 1985-1987: Therac-25 medical accelerator malfunctioned due to a thread concurrency issue, causing five deaths and numerous injuries.

Technical Debt



Unity Test Framework

Introduction to Unity

Figure 6: Unity Logo

Unity Test Framework

- Lightweight and simple unit testing framework for C.
- Designed for embedded systems but can be used in any C project.
- Provides a set of macros and functions to define and run tests.

Setting Up Unity

- Separate Unity tests into a separate directory, e.g., `tests/`.

Credits and Bibliography

- Course “Automated Software Testing,” Sébastien Bardin.
- CMake Tutorial: https://cmake.org/cmake/help/latest/guide/tutorial/index.html
- CMake Best Practices: https://cliutils.gitlab.io/modern-cmake/
- Unity Test Framework: http://www.throwtheswitch.org/unity
- Valgrind: http://valgrind.org/
- GDB: https://www.gnu.org/software/gdb/
- ASAN/UBSAN: https://clang.llvm.org/docs/AddressSanitizer.html
- Doxygen: https://www.doxygen.nl/index.html