Dev Environment Setup

Samuel Caldwell

Champlain College

CSCI-430: Compilers

Dr. Ken Revett

27 October 27, 2025

Dev Environment Setup

This document describes the development environment setup for this student’s computer. Much of the setup was already completed as part of this student’s professional life, including clang++ and LLVM. The underlying machine is an Apple MacBook Pro with 48GB RAM and an M4 Max CPU.

*(Please note that a .zip file of this student’s work is submitted along with this report to avoid excessive code snippets to explain what has been done.)*

## The Build System

The project uses CMake + Ninja as a build system with a Makefile wrapper for simplicity, all explained in a README.md file at the top-level directory. This Makefile is just sugar to allow ‘make clean build’ instead of more verbose CMake commands. We have docker available should we need to build in Linux, and we can quickly mount an Ubuntu:24.04 build container should this be appropriate.

Our CMakeLists.txt file is kept intentionally thin with modular parts in cmake/\*.cmake as follows:

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a computer program

AI-generated content may be incorrect.

We prefer this approach because the code is easier to read and we can create modular parts which are reusable across many projects.

## Repository Structure

The project is a git repository (<https://github.com/sam-caldwell/csci-430>) which will be maintained over the weeks using branching: week1, week2, week3, and so on.

This student believes in making code repositories intuitive for new people to onboard quickly. One practice he has evolved over the years is the use of a ‘make help’ as the default make target so that simply typing ‘make’ will deliver the helpful information needed:

A screenshot of a computer program

AI-generated content may be incorrect.

This allows the simplicity of ‘make’ to leverage the power of Cmake and Ninja, as should be evident.

## Versions Used

The following output of ‘make version’ summarizes the current environment:

A screenshot of a computer

AI-generated content may be incorrect.

## 

## Evidence of Proper Operation

Finally, we come to the hello world program to prove this development environment works. By executing ‘make clean build’ we get the following:

A screen shot of a computer screen

AI-generated content may be incorrect.

This produces the following artifacts (including the executable hello\_world) and various LLVM outputs:

A screenshot of a computer

AI-generated content may be incorrect.

We then run the binary as follows:

A screenshot of numbers and symbols

AI-generated content may be incorrect.

We can add new projects to the repository alongside this hello world program, keeping it in place as a baseline for our build system.