Virtual Development Environment in a Box

Students: Dylan McDougall (dmcdougall2019@my.fit.edu)

and Ian Orzel (iorzel2019@my.fit.edu)

Faculty: Ryan Stansifer (ryan@fit.edu)

Client: Ryan Stansifer (ryan@fit.edu)

Date of meetings:

- August 25 from 2:00 - 2:50 PM
- August 31 from 3:15 - 3:50 PM

Goal and Motivation:

Our goal is to create a platform that provides a virtual environment for students to run software that requires specific system setups (i.e., hardware, architecture, software stack) that is convenient to the user. This platform should be simple to use for both students and faculty, should be scalable, and should support multiple architectures. Specifically, this platform should have support for the *Compiler Theory* class, allowing for students to easily test their compilers on the platform.

Our motivation is to eliminate some of the difficulty that comes with setting up certain development tools that, for instance, require exotic architectures, hardware, or software configurations for use in several classes at Florida Tech, such as *Compiler Theory* and *Operating Systems Concepts*. This creates many inconveniences for both students and faculty alike, as students have to figure out how to get the systems properly set up and faculty have to assist students with this setup process.

Approach:

- Easily download and run containers on your local machine loaded with environments for your classes to easily test and create programs for these courses, using an interface that was created by studying the users of the platform. These containers can have different architectures from the local machine. They will also contain different development tools, such as gcc and make, debugging tools, such as gdb, and specialized tools, such as linkers and assemblers.
- Run your containers on any desktop platform, such as Windows, MacOS, and Linux.
- Import and export files to and from the container as well as provide standard input and output. For instance, in a compiler class, be able to add your compiler and program to the container to be compiled and interactively run it using standard input and output.
- Use pre-built containers for the Compiler Theory, Operating System Concepts, and Programming Language Concepts classes. Also, create containers for new courses or for existing courses without having to create an entire virtual environment from scratch and share them with students.

Novel Features/Functionality:

- The main novel feature of this system is that it will be created to be simple to use in an educational setting. Many other tools like this are general purpose, which causes them to be more difficult to use than is needed for this audience.

Technical Challenges:

- For this project, we plan to use qemu to create and run our containers, but we are not familiar with this program. We need to learn how to create images in qemu and how to interact with these images in a programmatic setting.

- We plan to create virtual environments to be used for the Compiler Theory course, but we have not taken this course. We thus need to determine what the course needs in such an environment.
- In order for students to use our program from any system, we need to learn how to develop cross-platform applications.

Milestone 1:

- Compare and select technical tools for:
    - How the user interacts with the system
    - Emulating virtual environments
    - Distributing containers
- Provide small ("hello world") demo(s) to evaluate the tools for:
    - Containerization/virtualization
    - Container sharing
    - Using developer tools inside a virtual environment
- Resolve technical challenges:
    - Learn how to use qemu
    - Learn about the virtual environment that is needed for a Compiler Theory course
    - Learn how to set up an environment for different platforms
- Compare and select collaboration tools for software development, documents/presentations, communication, task calendar
- Create Requirement Document
- Create Design Document
- Create Test Plan

Milestone 2:

- Implement, test, and demo a basic qemu image for use in Compiler Theory.
- Implement, test, and demo interaction with a qemu image through standard input and output.
- Implement, test, and demo file system interaction with a qemu image (allowing files to be easily transferred between user system and qemu system).

Milestone 3:

- Implement, test, and demo a cross-platform utility to set up the environment to run our program for users.
- Implement, test, and demo an online platform to allow for the qemu images to be extracted from the internet and run on the local machine for the purpose of software development.
- Implement, test, and demo a pipeline that allows all of the implemented features to work together in order to be used in the Compiler Theory course next semester.

Task matrix for Milestone 1:

Task	Ian	Dylan
Compare and Select Technical Tools	User interaction Container Distribution	Emulation
“Hello World” Demos	Container Sharing	Container Creation Container Tools
Resolve Technical Challenges	Compiler Theory Cross-Platform	Learn qemu
Compare and Select Collaboration Tools	Communication Documents/Presentations	Task Calendar Software Development
Requirement Document	Write 49%	Write 51%
Design Document	Write 25%	Write 75%

Test Plan	Write 75%	Write 25%

Approval from Faculty Advisor:

"I have discussed with the team and approved this project plan. I will evaluate the progress and assign a grade for each of the three milestones."

Signature: _____ Date: _____