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)

Progress of current Milestone:

Task	Completion	Ian	Dylan	Todo
Implement, demo, and test command-line interface support in Python	100%	80%	20%	
Implement, demo, and test command-line interface as shell and batch scripts to send information to Python	100%	50%	50%	
Implement, demo, and test installer or installation guide for installing system on Windows, MacOS, and Debian	100%	10%	90%	

Discussion:

- **Implement, demo, and test command-line interface support in Python**:

  In order to implement a command-line interface, we had to create a module that would take strings written by the user as input, validate them, and then dispatch to the correct modules of the container manager. This process requires a lot of exception handling in order to prevent users from inputting something invalid. It is difficult to ensure that all cases have been handled, so we have spent a lot of time testing it with strange inputs. Also, during this process, we realized that we needed to partially change how the program was set up, as certain parts of the program needed to be continually running even when the user was not directly interacting with it. Thus, we split our application into a client and a server to facilitate this.

- **Implement, demo, and test command-line interface as shell and batch scripts to send information to Python**:
  Creating the batch script proved more trivial than expected. The batch script simply calls a python program and sends all arguments given to it as input to the python program. After some more work on this, we determined that we did not need to create custom scripts and instead used PyInstaller to create executable files for us.
- **Implement, demo, and test installer or installation guide for installing system on Windows, MacOS, and Debian**:
  There are a few dependency challenges we have when distributing our program. The program depends on QEMU and the Python interpreter in order to function properly, as well as all of its Python module dependencies. To solve the Python issue, we're using a program called PyInstaller in order to distribute our Python program(s) in executable wrappers, which come bundled with all the necessary dependencies for running the specific version of Python with the exact modules we want. For QEMU, we are just going to have the installer prompt the user to install it on their computer.
  With regard to the installer itself, it will be distributed as a single executable file. The end user will download the release for their platform of choice, run the installer file to launch the installer, and follow the installer's instructions. The installer will copy the program files to the installation directory, create the .containers directory, and add the relevant folder containing the run program to the user's PATH.

Member Discussion:

- **Ian Orzel**: My main focus for this milestone was creating the command-line interface and a batch script that would interact with the Python program. I implemented the new module of the program that takes user input and dispatches it to the proper locations of the program. I also created the basic batch script to interact with it. After these two things were completed, the focus turned to testing. I have spent the rest of the time testing the system working together to try to run into problems, so that I could make the program deal with those situations when they arose.
- **Dylan McDougall**:

  There was a lot of general code maintenance we had to do this milestone just to get it to a state where we feel comfortable distributing the code to other students. One big thing that had to be done was to allow the container manager to run in the background and have a command-line interface interact with it. Ian worked on the command line interface. What I did was split the functionality of the program into a kind of client-server relationship. Since we're going to be transmitting a lot of text between processes, we figured that the best way to do this was by using sockets on localhost to facilitate that communication. This required rewriting a lot of the existing codebase. This task occupied a majority of my time. Once I got that to a point where I felt comfortable with it, I moved onto the installer task.

  A lot of the challenge was finding the best way to distribute Python. I considered shipping an embedded Python environment with the program, but it wasn't really feasible because the Python project does not provide an embedded version for macOS and Linux, so that was a dead-end. PyInstaller was really the only practical option after this. The other alternative was using the system Python, but that would have been problematic because the version of Python a given system has might not be compatible with our program, which was written with a very recent version of Python in mind. So with that, PyInstaller. This also allowed me to write the installer program itself in Python, which is nice and allows consistency with the rest of the project. Right now there's no way to cross-compile, but that's really a non-issue since we both use Windows and it's trivial to build for Linux using WSL, and we wouldn't be able to build for macOS without using macOS anyway.

Next Milestone Matrix:

Task	Ian	Dylan
Work with Compiler Theory students to fix bugs in the system and improve its usability	50%	50%
Implement, demo, and test interaction between server-side repositories and our program	100%	0%
Implement, demo, and test wizard for creating new containers	0%	100%

Discussion of Planned Tasks for Next Milestone:

- **Work with Compiler Theory students to fix bugs in the system and improve its usability**: During the next milestone, the Compiler Theory course will begin, and Dr. Stansifer will be using the first generation of the tool that we have been developing. During this time, we need to teach students how to use the tool in order to complete their coursework, and we will be taking feedback from them in order to determine ways to improve the tool and fix bugs that they find. During this milestone, we will work to quickly fix bugs and add basic features as quickly as possible as students report them.

- **Implement, demo, and test interaction between server-side repositories and our program**: One of the key functionalities of our program is the ability to upload containers onto servers that students can then easily download from. For this milestone, we would like to create this basic functionality. The two main pieces of this is to create the code for the server that takes in requests and sends back containers based on the requests and to upgrade the code of the students so that their program can interact with the servers.
- **Implement, demo, and test wizard for creating new containers**: Another key functionality of our program is a wizard that allows new containers to be built from scratch easily. We plan to upgrade the command-line interface to allow users to specify various options that allow them to create containers that will be configured to their liking.

Dates of Meeting with Client: (See Faculty Meeting Times)

Client Feedback for Milestone: (See Faculty Feedback)

Dates of Meeting with Faculty:

- November 2
- November 16

FacultyFeedback for Milestone:

- **Implement, demo, and test command-line interface support in Python**:

- **Implement, demo, and test command-line interface as shell and batch scripts to send information to Python**:

- **Implement, demo, and test installer or installation guide for installing system on Windows, MacOS, and Debian**:

Faculty Advisor Signature: ____ Date: _________

Score for each member:

Ian	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Dylan	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

Faculty Advisor Signature: ____ Date: _________