The following technical document describes multiple ways to go about compiling the Stacer project into an executable form.

Section 1: Absolute Requirements

To compile and run Stacer on Ubuntu the necessary pre-requisites are:

1. Qt5.10.0 (must be this version as it has header definitions that are crucial to the build)
2. Project files for Stacer 1.0.9, available on my branch in the repo labeled “compileRunSuccess” (The most recent version 1.1.0 will compile, but at runtime something is causing it to go out of bounds in the data segment and cause a segmentation fault)
3. CMake
4. GCC

Step 1: One option for obtaining the required Qt version (5.10.0) is to establish an educational license with Qt. It is a simple process and can be completed using your school email address for verification of enrollment in a higher learning environment. Once the profile is set up navigate to the downloads section of the Customer Portal.

A screenshot of a computer

Description automatically generated

Under the products sub-heading, choose Qt and set the version to 5.10.0. For the host operating system choose Linux.

A screenshot of a computer

Description automatically generated

Under the Qt for Linux sub-heading, download the .run file and update the user privileges for the file so that the user has full read, write, and execute privileges. Run the installation choosing all available features and take note of the installation directory.

A screenshot of a computer

Description automatically generated

Step 2: The GitHub repository for our fork of the Stacer project contains a branch called “compileRunSuccess.” This branch has the downgraded version of Stacer (1.0.9) that compiles and runs without crashing on Debian based Linux builds (like Ubuntu/Kubuntu). Assuming this branch has not been merged to the native branch yet do a checkout of the branch onto your local repository.

A screenshot of a computer

Description automatically generated

Once the checkout is complete, there should now be a bash script in the repository called “buildStacer.sh.” This script will allow you to compile and run the project without the need for a specific IDE configuration.

Step 3: In your Linux terminal, ensure that cmake is installed with the following command:

$ cmake –version

If no version of cmake is installed, use the following command to install it:

$ sudo apt-get install cmake

Step 4: Now ensure that the GCC compiler is installed on your system with the following command:

$ gcc –version

If GCC is not installed install it with the following command:

$ sudo apt-get install gcc

Step 5: Now that you have installed the necessary files on your system, you will need to update the “buildStacer.sh” bash script to include your system’s relevant file locations.

* The first section to update is the initial cd command in the script. Set it to the path where your Stacer repository is located, and ensure that it is pointing the directory that holds the Stacer directory without entering the Stacer directory. For example, if your Stacer directory is in

/home/ubuntu/GitHub Repos/Stacer/

then the cd command should point to

/home/ubuntu/GitHub Repos/

A screen shot of a computer

Description automatically generated

Make sure that the path is enclosed in single quotes to avoid confusing bash. If any directories contain spaces and the quotes are not present, then bash will treat them as multiple arguments and the compile will fail.

* The second section to update is the cmake command. This command contains a reference to the Qt cmake dependencies that are necessary for the CMakeLists.txt file to locate all the necessary Qt files. Once you have located the Qt installation directory, update the path in the bash script. For example, if your Qt directory is located in

/home/ubuntu/Qt5.10.0/

Then the -DCMAKE\_PREFIX\_PATH variable should point to

/home/ubuntu/Qt5.10.0/5.10.0/gcc\_64/lib/cmake

A screen shot of a computer

Description automatically generated

Save the updated bash script and update the file so that that the user has full read, write, and execute privileges over the file. Now run the script:

$ ./buildStacer.sh

* Optionally you can save a copy of the bash script with a name that is specific to your system and add it to the .gitignore file so that you don’t have to repeatedly update the script each time you do a remote pull or checkout.

The following section contains some optional configuration settings for IDE’s so that the program can be executed from within your development environment.

Section 2: Optional Setup

1. IDE Configuration with Qt Creator
2. IDE Configuration with CLion

For Qt Creator setup:

1. Open Qt Creator version that was installed with your Qt 5.10.0 installation. Select Open Project from the home screen.

A screenshot of a computer

Description automatically generated

1. Now navigate to the project directory and choose the CMakeLists.txt file.

A screenshot of a computer

Description automatically generated

1. Opening the file should cause the project to attempt to compile. If the compile is unsuccessful, in the bottom left corner of the screen choose the “Release” option, as the other build configurations for this project are not set up.

A screenshot of a computer

Description automatically generated

1. In the event that the file structure does not appear correctly and only shows the CMakeLists.txt file, navigate to projects on the left side option menu

A screenshot of a computer

Description automatically generated

and be sure that the correct Build & Run options are selected

A screenshot of a computer

Description automatically generated

1. You should now be able to execute and edit the project from within the Qt Creator IDE.

For CLion setup:

1. Open CLion (if you have no CLion license, there is an educational license available wth your student email address). On the Welcome screen, choose “Get from VCS”

A screenshot of a computer

Description automatically generated

1. Enter the GitHub space on the left option menu and choose the project and the repo directory on your local device

A screenshot of a computer program

Description automatically generated

1. Once the project is loaded, you can setup the run configuration to work with the bash script in the project repository. Simply navigate to the run menu at the top of the screen and choose “Edit Configurations” from the drop-down menu.

A screenshot of a computer program

Description automatically generated

1. Under “Add New Configuartion” choose the Shell Script option

A screenshot of a computer

Description automatically generated

1. Ensure that the Configuration name is “Release” and update the script path and the working directory to the relevant values.

A screenshot of a computer

Description automatically generated

1. Though the shell script configuration is quick and easy, it does not provide the use of the debugger. There is a way to configure the project so that it compiles from the CMake file and allows GDB disassembly at runtime for debugging purposes. Navigate to the settings page and find the “Build, Execution, Deployment” subheading. Under Toolchains, ensure that the settings match the image below. (You can ignore the version warning for CMake).

A screenshot of a computer

Description automatically generated

1. Now go to the CMake subheading and ensure that the settings match the image below. NOTE: The field for CMake Options should contain the options that were used in the shell script. They are:

-DCMAKE\_BUILD\_TYPE=Release

-DCMAKE\_PREFIX\_PATH=/home/ubuntu/Qt5.10.0/5.10.0/gcc\_64/lib/cmake

Make sure that you update the path for -DCMAKE\_PREFIX\_PATH to match the directory on your local device. Also ensure that the name of the profile and the build type are “Release” or they will fail.

A screenshot of a computer

Description automatically generated

1. Now navigate to the CMake controller on the bottom left of the IDE

A screenshot of a computer

Description automatically generated

1. Click on the option to reload the CMake project

A screenshot of a computer

Description automatically generated

1. Once the CMake has finished rebuilding the project, you can now choose the CMake based run configuration from the run menu

A screenshot of a computer

Description automatically generated

1. This run configuration gives you the option to use the GDB dissembler to debug the program. With GDB enabled, you can pause the program during runtime and view the registry and all assembly instructions that are being executed by the processor.

A screenshot of a computer

Description automatically generated