Testing your Code Generator with `runtests.bash`

By Sylvie, for CS241

1 Introduction

The `runtests.bash` script takes as arguments a series of names of WLP4 files, and will automatically run them and compare your results with the results from `wlp4c`, the reference implementation of the WLP4 compiler.

The `runtests.bash` script can be downloaded from here:

`https://www.student.cs.uwaterloo.ca/~cs241/questions/p5-starter/runtests.bash`

You can use `wget` to download it from the terminal.

2 Setup

Copy the `runtests.bash` script to the directory containing your compiler. Make it executable with `chmod u+x runtests.bash`.

Open the `runtests.bash` script in a text editor and confirm that the `MY_COMPILER` variable is set correctly, depending on the language you are using and the input format your compiler expects.

- If you are expecting Standard Format input and using `C++`:
  `MY_COMPILER="./wlp4gen"`
- If you are expecting Prescanned Format input and using `C++`:
  `MY_COMPILER="wlp4scan | ./wlp4gen"`
- If you are expecting Preanalyzed Format input and using `C++`:
  `MY_COMPILER="wlp4scan | wlp4parse | wlp4type | ./wlp4gen"`
- If you want to run with Valgrind, replace `./wlp4gen` with `valgrind ./wlp4gen`
- If you are using Racket, replace `./wlp4gen` with `racket wlp4gen.rkt`

There are other variables in the script you may wish to change, like the directory where test results are stored (default is `runtests_dir`) and the timeout for tests that run too long (default is 5 seconds).

3 Basic Usage

Create a file called `test.wlp4` containing a WLP4 program. Then create a file called `test.in` containing the input for `mips.twoints` or `mips.array`. For `mips.twoints`, the input should consist of two lines, each with one integer. For `mips.array`, the input should consist of a line containing an integer `n` representing the size of the array, followed by `n` lines containing integers for the array contents.

After setting up the test program and input, run the test with the following command:

```
./runtests.bash test.wlp4
```

The results of the test, or any errors that occured, will be printed to standard output.

You can also skip the step of setting up the input file if you wish, as `runtests.bash` will prompt you to enter input if it cannot find an input file, and then it will create the input file for future test runs.

Of course, you can use names other than `test.wlp4` and `test.in` with this command. But the name of the `.wlp4` file and the name of the `.in` file should match.

4 Viewing Results

Running the script will tell you whether the test passed or failed, and if it failed it will show the difference between the expected output and your output using `diff`.

If you want to view more detailed information, there are two options. One is to look in the `runtests_dir` directory. Since this is a bit inconvenient, the `runtests.bash` script also creates symbolic links to the files in the current directory for quick access. (A symbolic link is like an alias for a file.)

The list below shows the symbolic link name, followed by the corresponding full filename from `runtests_dir`, followed by a description of the file.

- `.xout` or `runtests_dir/test.wlp4.x.out`: The expected standard output for the test, as produced by running the MIPS simulator on the reference compiler's output.
- `.xerr` or `runtests_dir/test.wlp4.x.err`: The full results of the MIPS simulation for the reference compiler's output, showing the values in all registers and whether the program terminated normally or crashed. The `err` name is because the MIPS simulator outputs this information on standard error.
- `.xreg3` or `runtests_dir/test.wlp4.x.reg3`: The final value of register 3 when running the MIPS simulator on the reference compiler's output.
- `.uout` or `runtests_dir/test.wlp4.u.out`: The user's standard output for the test, as produced by running the MIPS simulator on the user's compiler's output.
- `.uerr` or `runtests_dir/test.wlp4.u.err`: The full results of the MIPS simulation for the user's compiler's output, showing the values in all registers and whether the program terminated normally or crashed.
- `.ureg3` or `runtests_dir/test.wlp4.u.reg3`: The final value of register 3 when running the MIPS simulator on the user's compiler's output.
- `.uasm` or `runtests_dir/test.wlp4.u.asm`: The assembly code produced by the user's compiler.

Additionally, a symbolic link `.in` is created to the test input file so that you can easily change the test input if desired.

For example, suppose you fail a test due to timeout (in this case `runtests.bash` doesn't show you standard output, because it could be extremely long). You can view the standard output produced by your compiler's results by typing `cat .uout` or `less .uout`. You can view the expected output from the reference compiler with `cat .xout` or `less .xout`. You can view the assembly file your compiler produced with `cat .uasm` or `less .uasm`.

5 Running Multiple Tests

You can run multiple tests by just passing multiple arguments to `runtests.bash`. For example, to run `test1.wlp4` and `test2.wlp4` in sequence:

```
./runtests.bash test1.wlp4 test2.wlp4
```

You can also use shell globbing patterns. To run all `.wlp4` files in the directory `mytests`:

```
./runtests.bash mytests/*.wlp4
```

Two notes about running test suites in this way:

- Don't simply do `./runtests.bash mytests/*` because this will try to run any non-WLP4 files in the `mytests` directory (such as input files) as if they were tests.
- The symbolic links for quickly viewing results will refer to the last test that gets executed only.
- Suppose you have two directories containing tests, called `problem1` and `problem2`, and both directories contain a file called `test1.wlp4`. This shouldn't cause issues, but in the `runtests_dir` directory, the test results for both tests will not be available. There will just be one set of results for `test1.wlp4` and it will be whichever test ran last.

6 Reporting Bugs

If you are getting strange or suspicious output from `runtests.bash`, there may be a bug. This script was originally written in 2020 and I had to make a bunch of quick fixes to get it running again. Please make a post on Piazza if you find a bug or don't understand something about the results from the script.