SE4950 – Fall 2020
Project 5 – Dynamic Analysis Lab

Background

In this project we will use two dynamic analysis tools, valgrind and gcov to find issues in example C programs.

valgrind

To invoke valgrind on a C source file (in this case example.c), looking for memory issues, first you need to build the program using gcc. This can be done by issuing a command line like the following:

```
gcc -Wall -g example.c -o example
```

To invoke valgrind on a C++ program (in this case example.cpp), looking for memory issues, first you need to build the program using gcc. This can be done by issuing a command line like the following:

```
g++ -Wall -g example.cpp -o example
```

Once the C or C++ program is built, it can be executed with valgrind using a command line like the following:

```
valgrind --tool=memcheck --leak-check=yes ./example
```

gcov

To run coverage analysis using gcov, the first thing that needs to be done is to build the application with the `--coverage` option. To do this for an application with one source file, example.c, you would do the following:

```
gcc --coverage example.c -o example
```

To do this for an application with two source files, example2a.c and example2b.c, you would do the following:

```
gcc -o example2 --coverage example2a.c example2b.c
```

Then to collect coverage data on the application you have instrumented with coverage, you simply run the application:

```
./example1
```

This will create a `.gcda` file for each of the source files.

To analyze the coverage, and create a `.gcov` file which shows the coverage for each file, you issue a command like the following:

```
gcov example1
```

The `.gcov` file will contain counters that show how many times each line of the source have been executed.

If you want to collect multiple runs before analyzing, do not delete the `.gcda` files. If you want to clear the coverage counters, then delete these files.

To create HTML reports that are easier to read, the program `lcov` may be used to do generate `.html` files instead of `.gcov` files.

To run the `lcov` analysis on a directory which contains coverage output files (`.gcda`), if you are currently in that directory, you could issue the following commands. The first creates the `lcov` info file, and the second coverts that into web files, in an `out` subdirectory.

```
lcov --capture --directory .  --output-file coverage.info
genhtml coverage.info --output-directory out
```

valgrind Analysis

In the VM, there is a directory named project5Code which contains multiple valtest.* files. For each of the valtest files, build the application and analyze them with valgrind as described in the valgrind section above. Describe the issues the valgrind found in the examples in the Project5Analysis.docx file.

gcov Analysis

In the same directory named project5Code there are three cov*.c files. The covtest1.c file will be used to build a covtest1 application. The covlib2.c and covtest2.c files will be used to build a covtest2 application. Build both of these applications as described in the gcov section above. Then, run the gcov and lcov analysis as described above. For both of the applications, note the lines which were not covered in the application test in the Project5Analysis.docx file.

Grading

The project will be worth 50 points.