Basic Coverage Measurement Tool – Gcov

One of the tutorials online:
https://www.tutorialspoint.com/unix_commands/gcov.htm

Compile the source code using gcc with -fprofile-arcs -ftest-coverage options:

prompt:> gcc -fprofile-arcs ftest-coverage foo.c

After executing the program, check the statement coverage using the following command:

prompt:> gcov foo.c

Check branch coverage:

prompt:> gcov -b foo.c

Check the coverage report:

prompt:> vim foo.c.gcov

A More Advanced Tool – xSuite

Read the following link to connect to xSuite server:
http://paris.utdallas.edu/ewong/SE6367/xSuite-Tutorial/TA.html

In this tutorial we illustrate how the basic features of *atac* and *xSuite* can be used in testing by way of a running example. *atac* is used to test wordcount, a program that counts the number of lines, words, and/or characters given to it as input.

Before using ATAC, check that the word counting program compiles and runs on a sample input. To create your executable program, type the appropriate *make* command to build on your system.

prompt:> make CC="atac cc"

Once *wordcount* has been built, run it against a sample input:

prompt:> ./wordcount input1

The file *input1* contains the following line (the first character is a tab):

prompt:>test input file 1

The output of *wordcount* should look like this:

1 4 19 input1
1 4 19 total

Now you are ready to use *atac*. Remove the previously created object files and the executable file. One way to do this is to use the clean command appropriate for your setup. Recompile the wordcount program with *atac*.

prompt:> make clean
prompt:> make CC="atac cc"

Now invoke the graphical browser *xSuite* by entering the following command:

prompt:> xsuite wordcount.atacp

The source window in the middle displays the first source file, *main.c*, with all of its basic blocks highlighted in various colors. Each color represents a certain weight. White represents zero weight and red represents the highest weight among all blocks in the file. If a block is highlighted in white, it means that it has already been covered by a test case and covering it again will not add new coverage. If, on the other extreme, a block is highlighted in red, it means that it has not been covered by any test case so far and covering it first is the most efficient way to add new coverage to the program.

Note that there are no white regions in the scroll bar at this point as we have not run the instrumented program on any inputs, so no blocks in the file have been covered yet.

Click on or near the red spot so that part of the file becomes visible in the source window.

Analysis of the code reveals that the two red blocks are exercised whenever the program reads its input from a file. Let us run wordcount on an input file, *input1*. Open a new command prompt window and type:

prompt:> ./wordcount input1

In addition to the expected output, running this test case has created an execution trace file called *wordcount.trace*. This file contains dynamic coverage information used in test analysis.

Select *wordcount.trace* and click on the "*Open*" button. This will cause *xSuite* to read the trace file and update the source window display.

Note that both the previously red blocks, along with several others, have turned in color to white indicating that they were, indeed, covered by the test case you just ran. Also note that the red block has now shifted to another statement in the file.

Scroll the main window until the new red spot is visible in the source window.

The red block, as you can see by analyzing the code, will be executed only when the program is invoked with an invalid command line option. Let us run *wordcount* with an invalid option, "*-x*":

prompt:> ./wordcount -x input1

It should produce an appropriate error message. Running *wordcount.2* causes its coverage information to be added to the trace file. Open the trace file again to update the coverage information.

Again notice that the block you were trying to cover, as well as some other previously uncovered blocks, have changed in color to white indicating that they were covered by the test you just ran. Also note that the red block has now shifted to yet another part of the program.