This tutorial is to show how to use xSlice to help you locate the bug in a program.

Please read the following descriptions, and finish each step:

Preparation

1. Compile the correct version of the wordcount program without instrumentation

   ```
   > cc -o wordcount main.c wc.c
   ```

   ….. (ignore the warnings) …..

2. Compile the faulty version of the wordcount program with instrumentation

   ```
   > atac cc -g -o wc_err main_err.c wc.c
   ```

   ….. (ignore the warnings) …..

3. Clean the ATAC_TRACE environmental variable

   ```
   > export ATAC_TRACE=
   ```

Find out which test cases invokes the failure

4. Run correct version and faulty version with the 1st test case

   ```
   > ./wordcount input1
   ```

   ….. (some output) …..

   ```
   > ./wc_err input1
   ```

   ….. (some output) …..

   The output generated by wordcount and wc_err should be the same, which means this test case did not invoke the error

5. Run the 2nd test case

   ```
   > ./wordcount -w input1
   ```

   ….. (some output) …..

   ```
   > ./wc_err -w input1
   ```

   ….. (some output) …..

   The output generated by wordcount and wc_err should be the same, which means this test case did not invoke the error neither

6. Run the 3rd test case

   ```
   > ./wordcount -w < input1
   ```

   ….. (some output) …..

   ```
   > ./wc_err -w < input1
   ```

   ….. (some output) …..

   The output of correct version and faulty version should be different. This means the 3rd test case invokes the failure

Use xSlice to help you identify the location of the bug

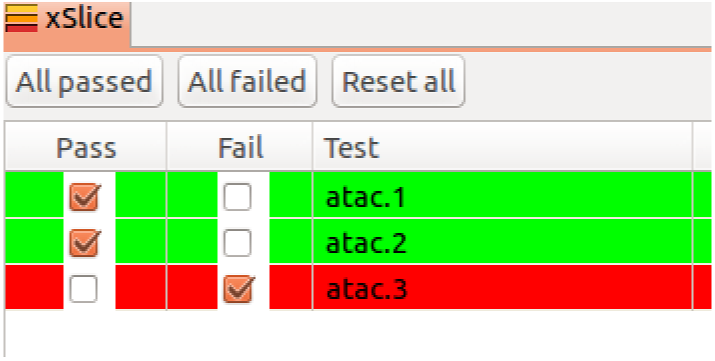
7. Run the tool using

   ```
   > xsuite wc_err.atacp
   ```

   then open the trace file "atac.trace" to load the coverage information to the tool

8. Switch the tool to xSlice mode by clicking menu "Coverage" -> "xSlice"

9. Specify which tests are the correct test cases, and which are the failed test cases.



Pass	Fail	Test
☑	□	atac.1
☑	□	atac.2
□	☑	atac.3

10. Check the source code, the bug location should be highlighted by RED already.