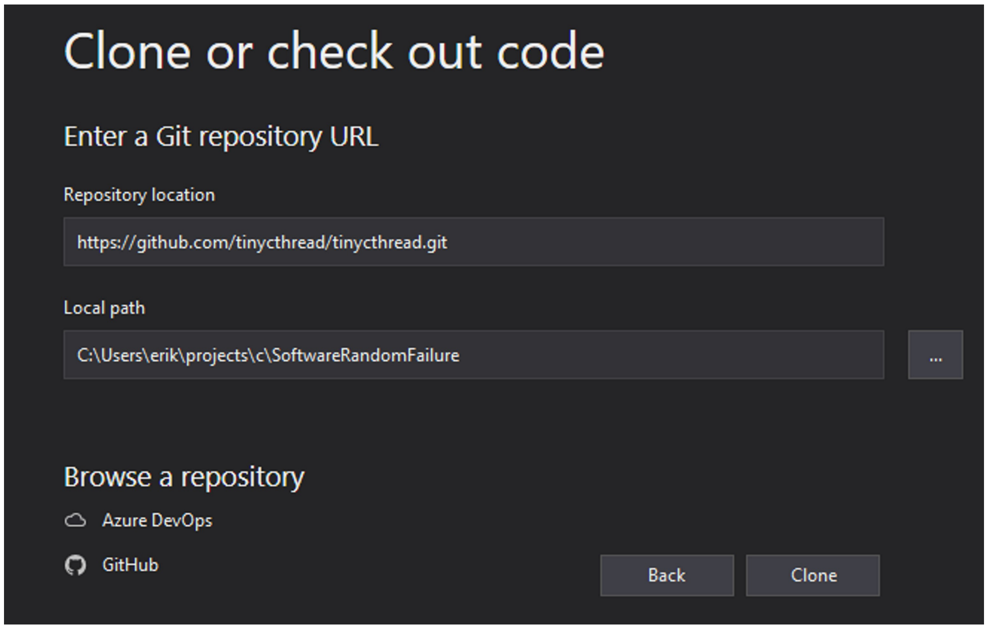
Demonstrate Random SW Failure

Friday, January 24, 2020 12:30 PM

Creating Demo of Random SW Failure and volatile variable

1. See url -> https://tinycthread.github.io/

2. Visual Studio 2019 - Clone or Check out Code using git clone https://github.com/tinycthread/tinycthread.git



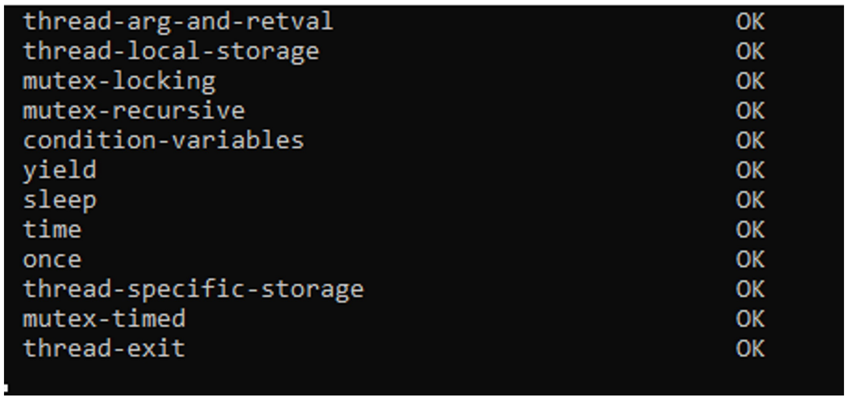
Clone or check out code

Enter a Git repository URL

Repository location: https://github.com/tinycthread/tinycthread.git

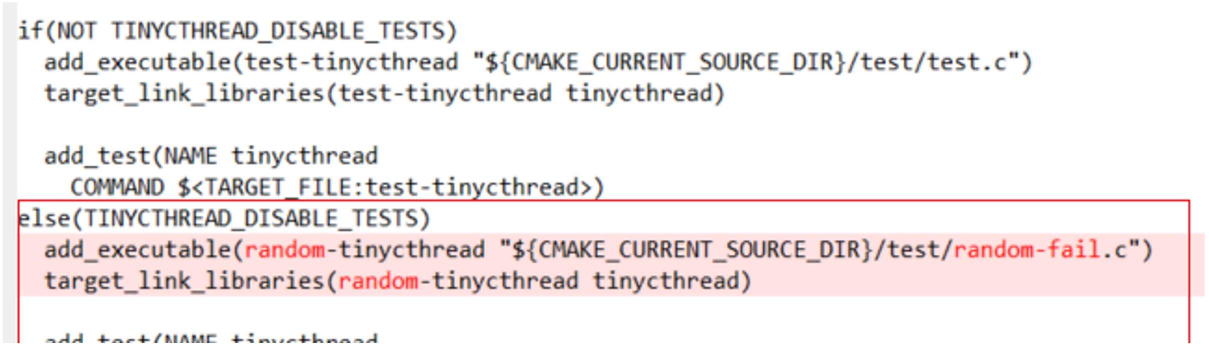
Local path: C:\Users\erik\projects\c\SoftwareRandomFailure

3. Build and test you can run the test (output below). (Build F7, Run F5)

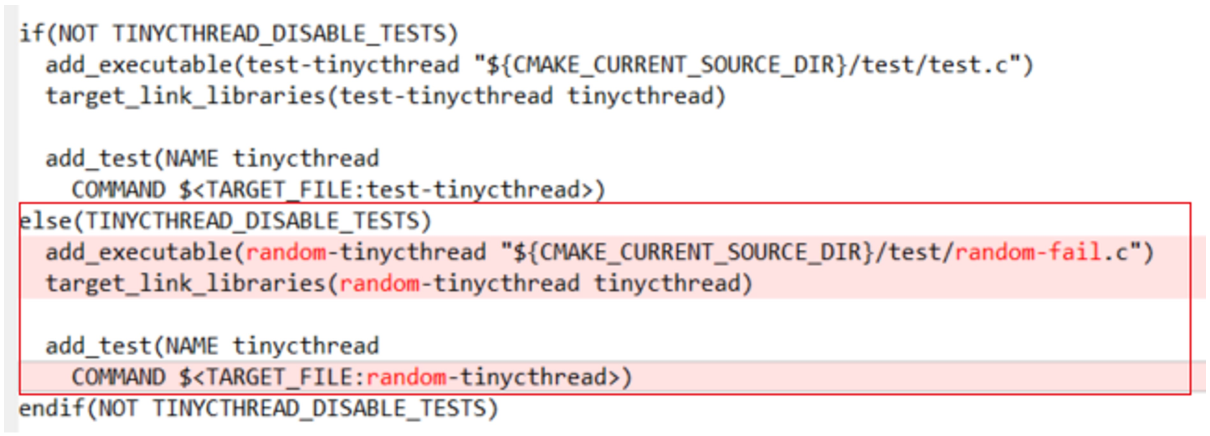


```
thread-arg-and-retval                               OK
thread-local-storage                                OK
mutex-locking                                       OK
mutex-recursive                                     OK
condition-variables                                 OK
yield                                               OK
sleep                                               OK
time                                                OK
once                                                OK
thread-specific-storage                             OK
mutex-timed                                         OK
thread-exit                                         OK
```

4. Update CMakeList.txt to include make for the /test/random-fail.c

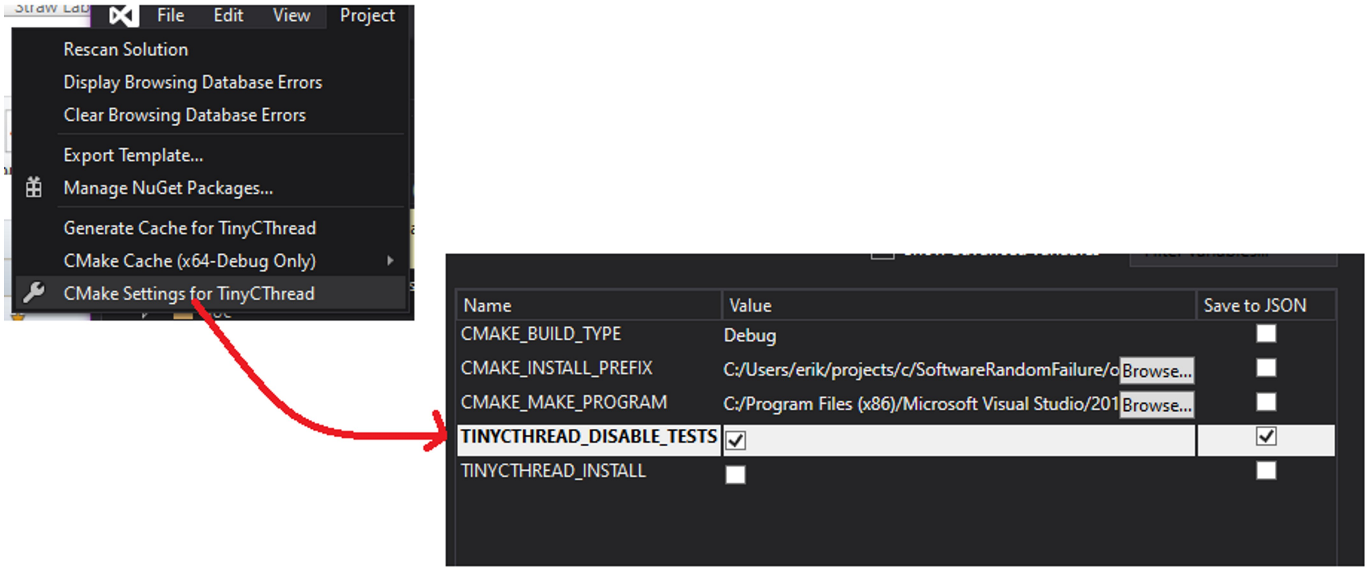


```
if(NOT TINYCTHREAD_DISABLE_TESTS)
  add_executable(test-tinycthread "${CMAKE_CURRENT_SOURCE_DIR}/test/test.c")
  target_link_libraries(test-tinycthread tinycthread)

  add_test(NAME tinycthread
    COMMAND $<TARGET_FILE:test-tinycthread>)
else(TINYCTHREAD_DISABLE_TESTS)
  add_executable(random-tinycthread "${CMAKE_CURRENT_SOURCE_DIR}/test/random-fail.c")
  target_link_libraries(random-tinycthread tinycthread)
```



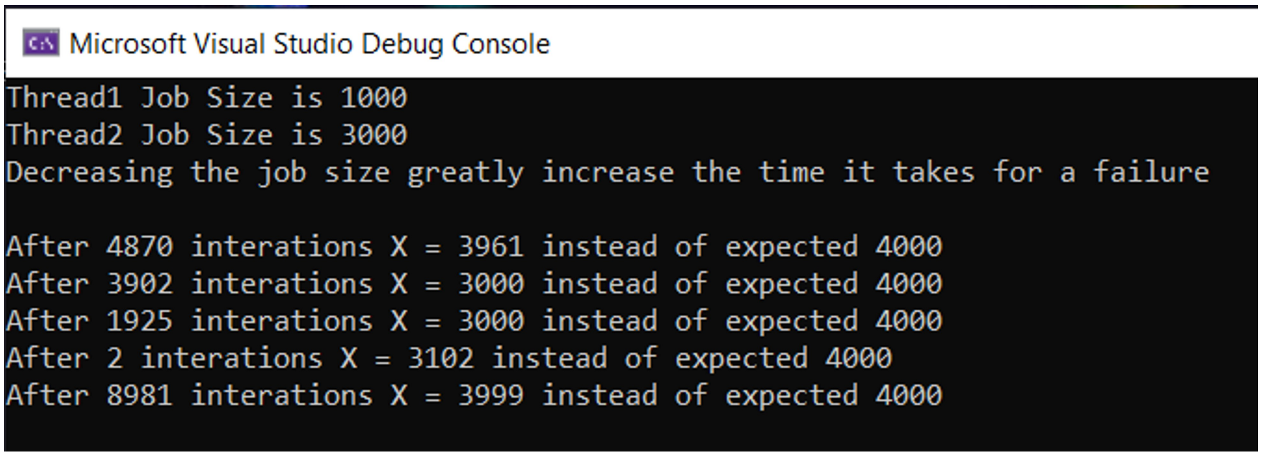
```
if(NOT TINYCTHREAD_DISABLE_TESTS)
  add_executable(test-tinycthread "${CMAKE_CURRENT_SOURCE_DIR}/test/test.c")
  target_link_libraries(test-tinycthread tinycthread)

  add_test(NAME tinycthread
    COMMAND $<TARGET_FILE:test-tinycthread>)
else(TINYCTHREAD_DISABLE_TESTS)
  add_executable(random-tinycthread "${CMAKE_CURRENT_SOURCE_DIR}/test/random-fail.c")
  target_link_libraries(random-tinycthread tinycthread)

  add_test(NAME tinycthread
    COMMAND $<TARGET_FILE:random-tinycthread>)
endif(NOT TINYCTHREAD_DISABLE_TESTS)
```

5. Turn off the build of the test software (test/test.c).


File
Edit
View
Project
Rescan Solution
Display Browsing Database Errors
Clear Browsing Database Errors
Export Template...
Manage NuGet Packages...
Generate Cache for TinyCThread
CMake Cache (x64-Debug Only)
CMake Settings for TinyCThread

Name	Value	Save to JSON
CMAKE_BUILD_TYPE	Debug	☐
CMAKE_INSTALL_PREFIX	C:/Users/erik/projects/c/SoftwareRandomFailure/o Browse...	☐
CMAKE_MAKE_PROGRAM	C:/Program Files (x86)/Microsoft Visual Studio/201 Browse...	☐
TINYCTHREAD_DISABLE_TESTS	✓	✓
TINYCTHREAD_INSTALL	☐	☐

6. Build All the Software and run random fail experiment. You can change the T1_JOB and T2_JOB size in test/random-fail.c.



Microsoft Visual Studio Debug Console

```
Thread1 Job Size is 1000
Thread2 Job Size is 3000
Decreasing the job size greatly increase the time it takes for a failure

After 4870 interations X = 3961 instead of expected 4000
After 3902 interations X = 3000 instead of expected 4000
After 1925 interations X = 3000 instead of expected 4000
After 2 interations X = 3102 instead of expected 4000
After 8981 interations X = 3999 instead of expected 4000
```

Fallacy 3 *It is often said (and repeated in many standards) that software does not fail randomly — all software failure is systematic and anyone analyzing software failure statistically falls into a state of sin. There are many academic papers explaining why this is a fallacy, but one simple program should be enough: see Figure 5.4.*