HThread

A THREAD ERROR DETECTOR

INDEX

- Overview
- Detected Errors
- Error Report Example
- Enabling HThread
- License
- Contact
- Q&A

Overview

- Is a lightweight tool for detecting synchronization errors in C/C++ applications that use the Pthreads API, specifically designed for embedded systems.
- Has a negligible effect on run-time speed.
- Does not require any source code change.
- Operating system and architecture independent.
- Easy to use.

Detected Errors

- Misuses of PThreads API
- Lock Order Violation
- Lock Contention

Detected Errors

Misuses of PThreads API

Although some of them are too obvious, early detection is much better than to deal with hard-to-find bugs.

- destroying an invalid mutex
- locking an invalid mutex
- unlocking an invalid mutex
- locking an already locked mutex
- unlocking an unheld mutex
- destroying a locked mutex
- destroying an invalid condition
- signaling an invalid condition
- broadcasting an invalid condition
- [timed]waiting on an invalid condition
- [timed]waiting on an invalid mutex
- [timed]waiting on an unheld mutex
- join invalid thread
- detach invalid thread
- unlocking mutex that was held by other thread

Detected Errors

Lock Order Violation

Useful because they usually result to deadlocks. They may never be discovered during testing and lead to hard-to-find bugs.

- Lock order violation in same thread.
- Lock order violation while [timed]waiting on condition.
- Lock order violation between threads.

Detected Errors

Lock Contention

Monitoring lock contentions is handy because, they usually cause unwanted delays or they may point to an undetected potential deadlock.

- waiting to lock a mutex more than allowed threshold.
- hold a mutex lock more than allowed threshold.

Error Report Example

Lock Order Violation

```
# ./test/fail-43-debug
 new thread created: 'root-process (0x14ae0c0)’
     at: (null) (null):0
 new thread created: 'thread(main fail-43.c:73) (0x14af0f0)’
     at: main fail-43.c:73
 mutex lock order 'mutex(main fail-43.c:53) (0x14ae010)' before 'mutex(main fail-43.c:58) (0x14aea00)' violated
   incorrect order is: acquisition of 'mutex(main fail-43.c:58) (0x14aea00)’
       by: thread(main fail-43.c:73) (0x14af0f0)
       at: worker fail-43.c:23
     followed by a later acquisition of 'mutex(main fail-43.c:53) (0x14ae010)’
       by: thread(main fail-43.c:73) (0x14af0f0)
       at: worker fail-43.c:28
           0x408d24: hthread.c (debug_dump_callstack:829)
           0x40c754: hthread.c (debug_mutex_try_lock:1088)
           0x40206f: fail-43.c (worker:29)
           0x40db5a: hthread.c (thread_run:619)
           0x7f58cbd27e9a: pthread_create.c (start_thread:308)
   required order is: acquisition of 'mutex(main fail-43.c:53) (0x14ae010)’
       by: root-process (0x14ae0c0)
       at: main fail-43.c:63
     followed by a later acquisition of 'mutex(main fail-43.c:58) (0x14aea00)’
       by: root-process (0x14ae0c0)
       at: main fail-43.c:68
   created 'mutex(main fail-43.c:58) (0x14aea00)’
     at: main fail-43.c:58
   created 'mutex(main fail-43.c:53) (0x14ae010)’
     at: main fail-43.c:53
fail-43-debug: hthread.c:1036: debug_mutex_add_lock: Assertion `"lock order violation"' failed.
```

Enabling HThread

- Add to CFLAGS:

```
-include hthread.h -DHTHREAD_DEBUG=1 -g –O1
```

- Add to LDFLAGS if HTHREAD_ENABLE_CALLSTACK is 0

```
-lhthread –lrt
```

- Add to LDFLAGS if HTHREAD_ENABLE_CALLSTACK is 1

```
-lhthread -lrt -ldl –lbfd
```

License

Contact

- http://alperakcan.net/projects/libhthread
- https://github.com/anhanguera/libhthread
- alper.akcan@gmail.com

Q&A