Helgrind:

- "a thread error detector"

$ valgrind --tool=helgrind

- it is for detecting synchronisation errors in C, C++, and fortran

 for programs using POSIX pthreads

- lock/unlock problems, race conditions, etc

- see <http://valgrind.org/docs/manual/hg-manual.html>

Helgrind is a thread error detector that helps find and resolve thread-related problems, such as race conditions, inconsistent lock orderings, or misuse of POSIX pthreads API.

To effectively use Helgrind:

- write threaded programs from the beginning that allow Helgrind to verify its correctness.

- using a linux distribution

- POSIX threading primitives

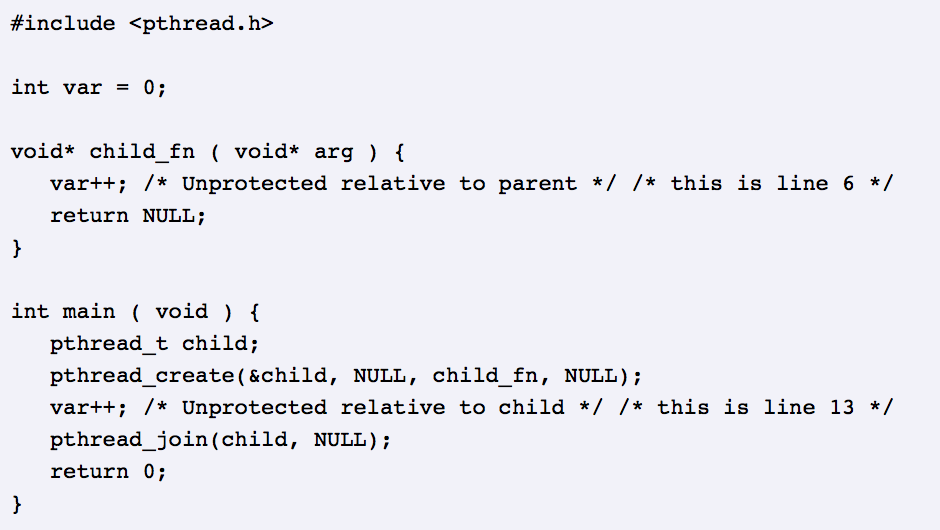
- avoiding memory recycling and POSIX condition variables

- using pthread\_join to gather finished threads

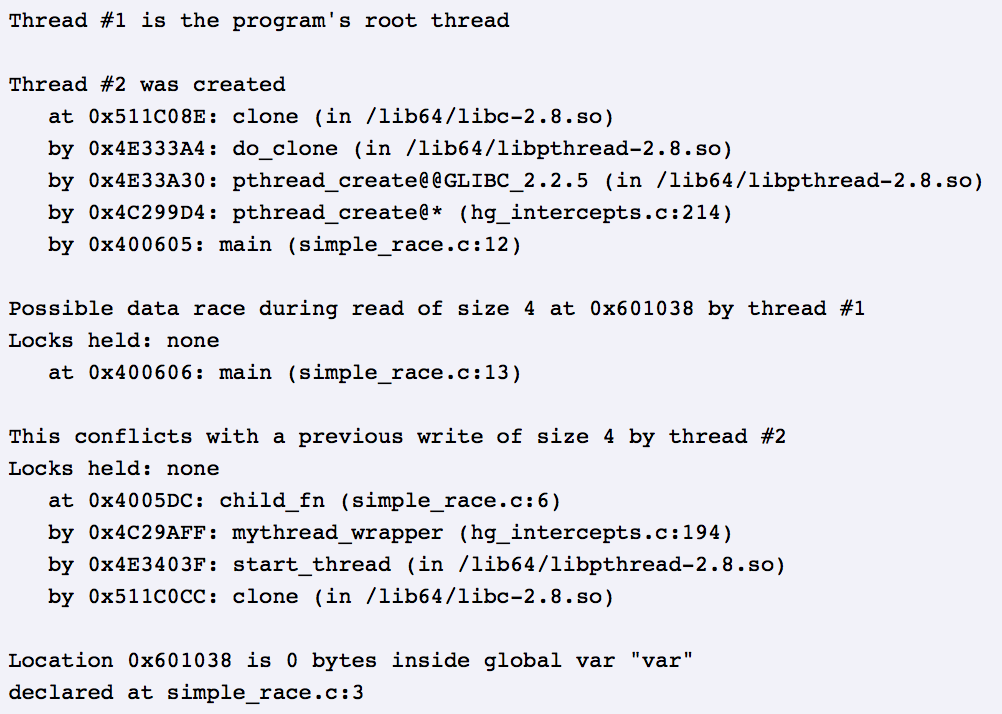
- performing Helgrind, which is thread debugging, and Memcheck, which is memory debugging at the same time

To discuss one thread related problem that Helgrind can help is Data Race/Race Condition. This can occur when there are two or more threads running at the same time that try to access a shared memory location. Due to the thread scheduling algorithm, its difficult to know which thread will attempt to access the shared region, since the threads can swap at any time. For these cases, locking or synchronizing can help prevent these issues.

Example with data race: var being accessed by both threads



Running Helgrind on this program:



A few helpful Helgrind command line options:

--ignore-thread-creation=<yes|no> [default: no]

This will allow control for whether all activities during the thread creation should be ignored or not.

--check-stack-refs=no|yes [default: yes]

Since Helgrind checks all data memory accesses your program makes, this flag allows you to skip the check for accesses to the thread stack.

--free-is-write=no|yes [default: no]

If this is enabled, Helgrind will treat the freeing of heap memory the same as if the memory was written immediately before the free.

--track-lockorders=no|yes [default: yes]

If this is enabled, Helgrind will perform lock order consistency checking.

DRD:

- "a thread error detector"

- for detecting errors in multithreaded C/C++ programs, for POSIX threads

- <http://valgrind.org/docs/manual/drd-manual.html>

DRD is another valgrind tool used to for detecting errors. This one focuses on the errors within multithreaded programs for C/C++ that uses POSIX threading primitives. It is helpful for Data races, lock contention, improper use of POSIX threads API, deadlocks, and false sharing. It will print a message when it finds one of these errors.

To effectively use DRD:

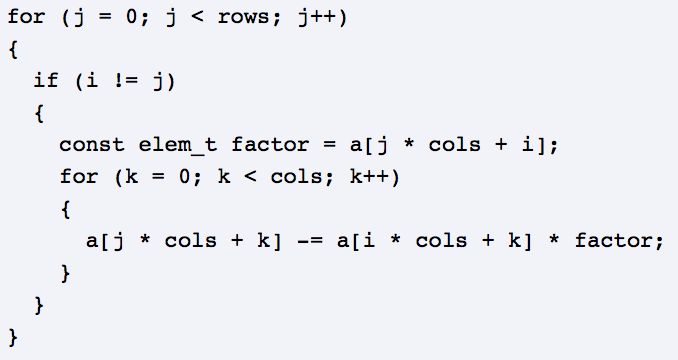
- debug information needs to be present in the executable

- to reduce the amount of generated code, compile with –O1 option.

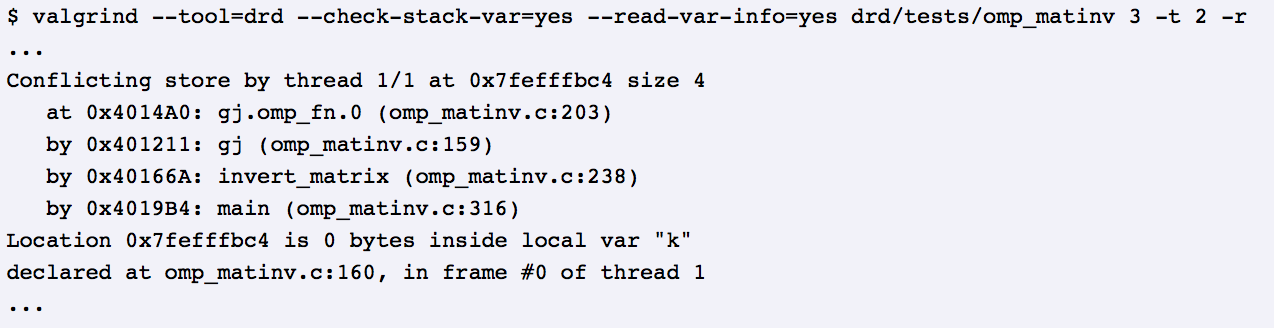
- installing the debug packages for the linux distribution libraries

- Not sending output from more than one thread to std::cout

An example of race condition:



Running DRD on this program:



A few helpful DRD command line options are shown above in Helgrind and below:

--show-stack-usage=<yes|no> [default: no]

This will print the known stack usage at the exit time of the thread.

--show-confl-seg=<yes|no> [default: yes]

This will show the conflicting segments contained in the race reports.

--shared-threshold=<n> [default: off]

It also can print an error message if a reader lock is held longer than the allotted time.

--check-stack-var=<yes|no> [default: no]

Helps controls if DRD detected data races on stack variables.

References:

Wikipedia - https://en.wikipedia.org/wiki/Valgrind

Valgrind Manual - <http://valgrind.org/docs/manual/hg-manual.html>

http://valgrind.org/docs/manual/drd-manual.html