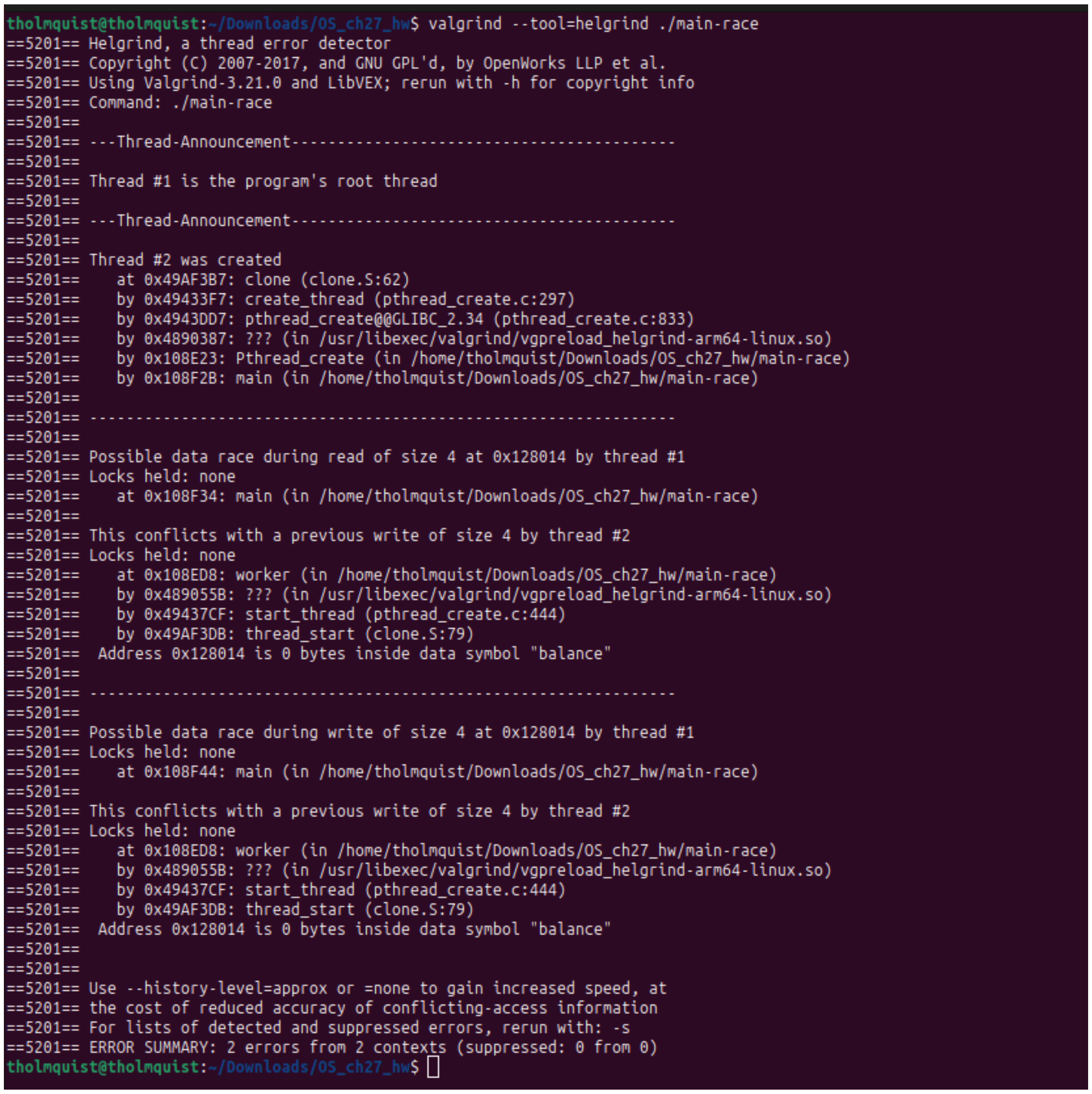
Ch.27 Coding Homework

Tyler Holmquist

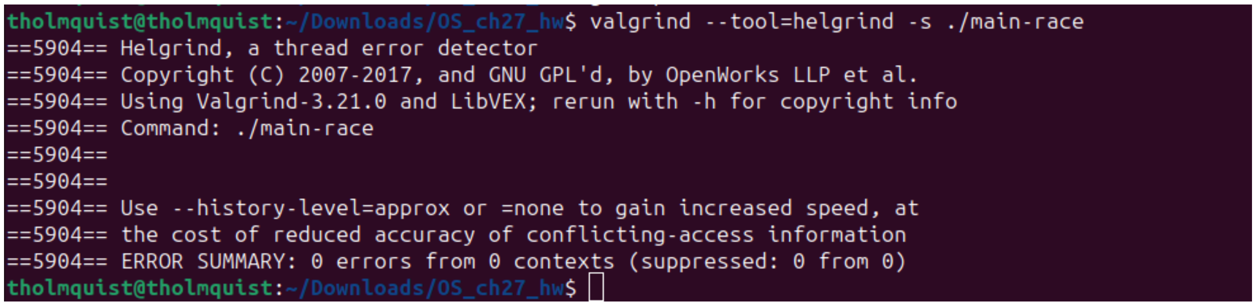
gcc (Ubuntu 13.2.0-4ubuntu3) 13.2.0

1. Yes it points to the right lines of code. It tells us that there is a possible data race during the write of an integer in thread 1 and 2 pointing to the line balance++ in main and in worker, and states the thread ID's and memory location of the offending code. (gcc -pthread -o main-race main-race.c)



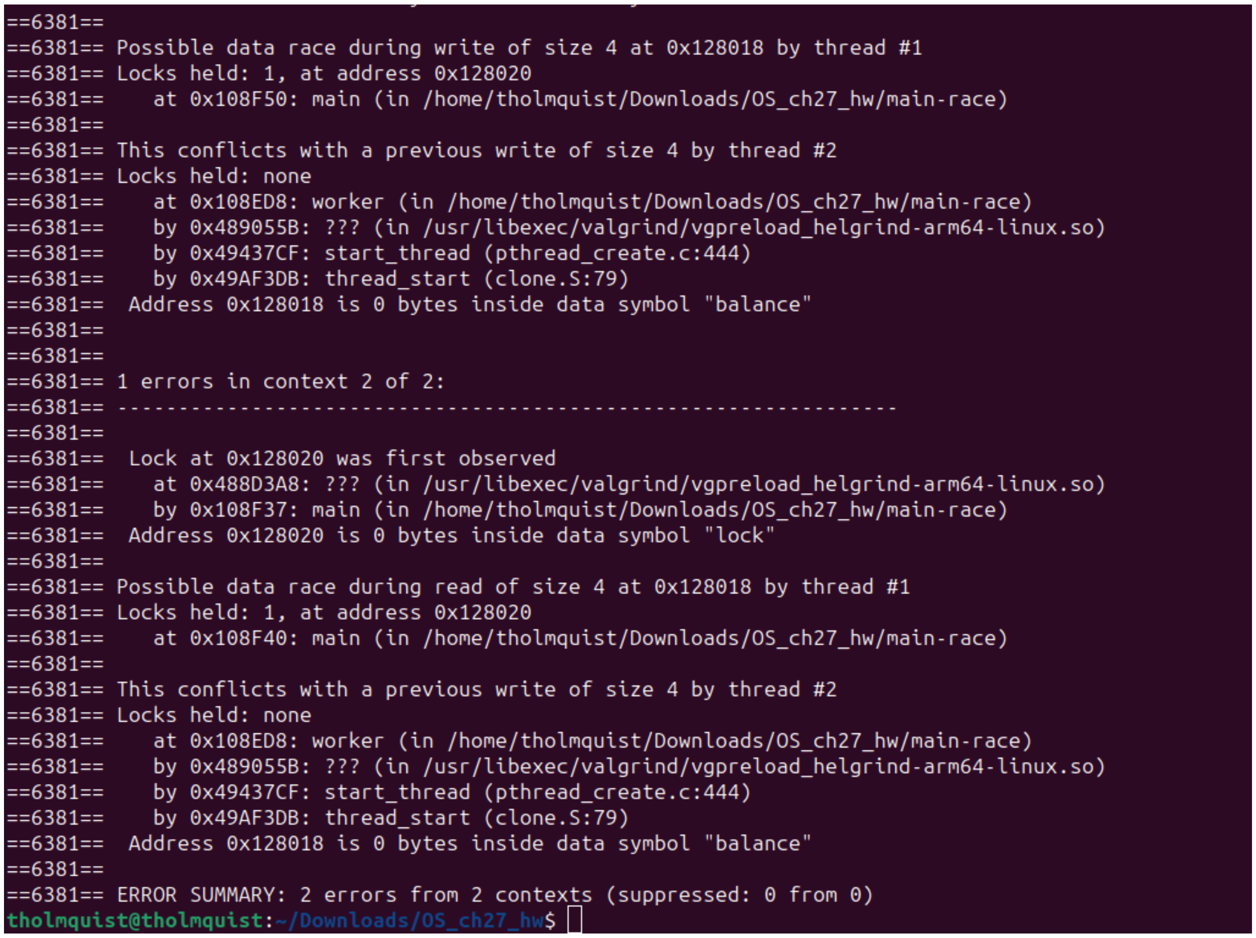
```
tholmquist@tholmquist:~/Downloads/OS_ch27_hw$ valgrind --tool=helgrind ./main-race
==5201== Helgrind, a thread error detector
==5201== Copyright (C) 2007-2017, and GNU GPL'd, by OpenWorks LLP et al.
==5201== Using Valgrind-3.21.0 and LibVEX; rerun with -h for copyright info
==5201== Command: ./main-race
==5201==
==5201== ---Thread-Announcement------------------------------------------
==5201==
==5201== Thread #1 is the program's root thread
==5201==
==5201== ---Thread-Announcement------------------------------------------
==5201==
==5201== Thread #2 was created
==5201==    at 0x49AF3B7: clone (clone.S:62)
==5201==    by 0x49433F7: create_thread (pthread_create.c:297)
==5201==    by 0x4943DD7: pthread_create@@GLIBC_2.34 (pthread_create.c:833)
==5201==    by 0x4890387: ??? (in /usr/libexec/valgrind/vgpreload_helgrind-arm64-linux.so)
==5201==    by 0x108E23: Pthread_create (in /home/tholmquist/Downloads/OS_ch27_hw/main-race)
==5201==    by 0x108F2B: main (in /home/tholmquist/Downloads/OS_ch27_hw/main-race)
==5201==
==5201== ----------------------------------------------------------------
==5201==
==5201== Possible data race during read of size 4 at 0x128014 by thread #1
==5201== Locks held: none
==5201==    at 0x108F34: main (in /home/tholmquist/Downloads/OS_ch27_hw/main-race)
==5201==
==5201== This conflicts with a previous write of size 4 by thread #2
==5201== Locks held: none
==5201==    at 0x108ED8: worker (in /home/tholmquist/Downloads/OS_ch27_hw/main-race)
==5201==    by 0x489055B: ??? (in /usr/libexec/valgrind/vgpreload_helgrind-arm64-linux.so)
==5201==    by 0x49437CF: start_thread (pthread_create.c:444)
==5201==    by 0x49AF3DB: thread_start (clone.S:79)
==5201==  Address 0x128014 is 0 bytes inside data symbol "balance"
==5201==
==5201== ----------------------------------------------------------------
==5201==
==5201== Possible data race during write of size 4 at 0x128014 by thread #1
==5201== Locks held: none
==5201==    at 0x108F44: main (in /home/tholmquist/Downloads/OS_ch27_hw/main-race)
==5201==
==5201== This conflicts with a previous write of size 4 by thread #2
==5201== Locks held: none
==5201==    at 0x108ED8: worker (in /home/tholmquist/Downloads/OS_ch27_hw/main-race)
==5201==    by 0x489055B: ??? (in /usr/libexec/valgrind/vgpreload_helgrind-arm64-linux.so)
==5201==    by 0x49437CF: start_thread (pthread_create.c:444)
==5201==    by 0x49AF3DB: thread_start (clone.S:79)
==5201==  Address 0x128014 is 0 bytes inside data symbol "balance"
==5201==
==5201==
==5201== Use --history-level=approx or =none to gain increased speed, at
==5201== the cost of reduced accuracy of conflicting-access information
==5201== For lists of detected and suppressed errors, rerun with: -s
==5201== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
tholmquist@tholmquist:~/Downloads/OS_ch27_hw$
```

2.



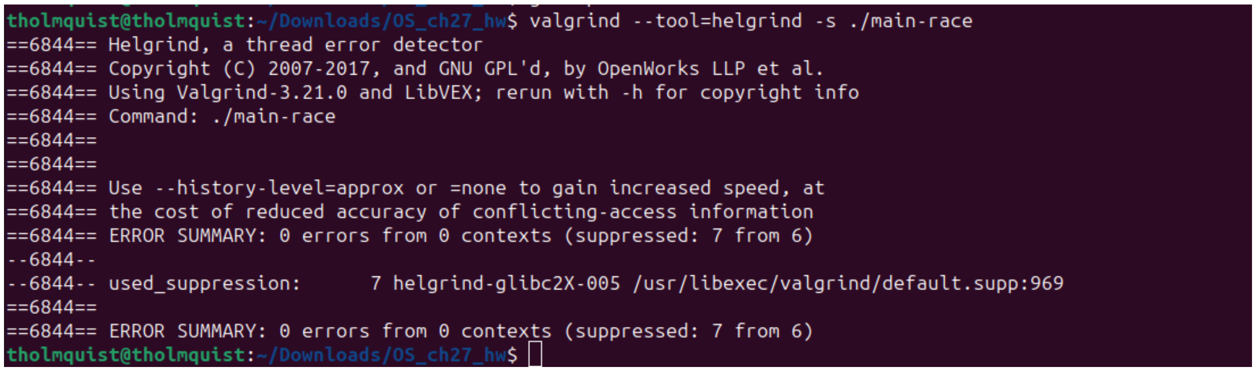
```
tholmquist@tholmquist:~/Downloads/OS_ch27_hw$ valgrind --tool=helgrind -s ./main-race
==5904== Helgrind, a thread error detector
==5904== Copyright (C) 2007-2017, and GNU GPL'd, by OpenWorks LLP et al.
==5904== Using Valgrind-3.21.0 and LibVEX; rerun with -h for copyright info
==5904== Command: ./main-race
==5904==
==5904==
==5904== Use --history-level=approx or =none to gain increased speed, at
==5904== the cost of reduced accuracy of conflicting-access information
==5904== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
tholmquist@tholmquist:~/Downloads/OS_ch27_hw$
```

if you remove one of the balance++ calls then the code does not report any racing risks. (gcc -pthread -o main-race main-race.c)



```
==6381==
==6381== Possible data race during write of size 4 at 0x128018 by thread #1
==6381== Locks held: 1, at address 0x128020
==6381==    at 0x108F50: main (in /home/tholmquist/Downloads/OS_ch27_hw/main-race)
==6381==
==6381== This conflicts with a previous write of size 4 by thread #2
==6381== Locks held: none
==6381==    at 0x108ED8: worker (in /home/tholmquist/Downloads/OS_ch27_hw/main-race)
==6381==    by 0x489055B: ??? (in /usr/libexec/valgrind/vgpreload_helgrind-arm64-linux.so)
==6381==    by 0x49437CF: start_thread (pthread_create.c:444)
==6381==    by 0x49AF3DB: thread_start (clone.S:79)
==6381==  Address 0x128018 is 0 bytes inside data symbol "balance"
==6381==
==6381==
==6381== 1 errors in context 2 of 2:
==6381== ----------------------------------------------------------------
==6381==
==6381==  Lock at 0x128020 was first observed
==6381==    at 0x488D3A8: ??? (in /usr/libexec/valgrind/vgpreload_helgrind-arm64-linux.so)
==6381==    by 0x108F37: main (in /home/tholmquist/Downloads/OS_ch27_hw/main-race)
==6381==  Address 0x128020 is 0 bytes inside data symbol "lock"
==6381==
==6381== Possible data race during read of size 4 at 0x128018 by thread #1
==6381== Locks held: 1, at address 0x128020
==6381==    at 0x108F40: main (in /home/tholmquist/Downloads/OS_ch27_hw/main-race)
==6381==
==6381== This conflicts with a previous write of size 4 by thread #2
==6381== Locks held: none
==6381==    at 0x108ED8: worker (in /home/tholmquist/Downloads/OS_ch27_hw/main-race)
==6381==    by 0x489055B: ??? (in /usr/libexec/valgrind/vgpreload_helgrind-arm64-linux.so)
==6381==    by 0x49437CF: start_thread (pthread_create.c:444)
==6381==    by 0x49AF3DB: thread_start (clone.S:79)
==6381==  Address 0x128018 is 0 bytes inside data symbol "balance"
==6381==
==6381== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
tholmquist@tholmquist:~/Downloads/OS_ch27_hw$
```

If you only lock one instance of balance++ then you still get a potential racing error. (gcc -pthread -o main-race main-race.c)



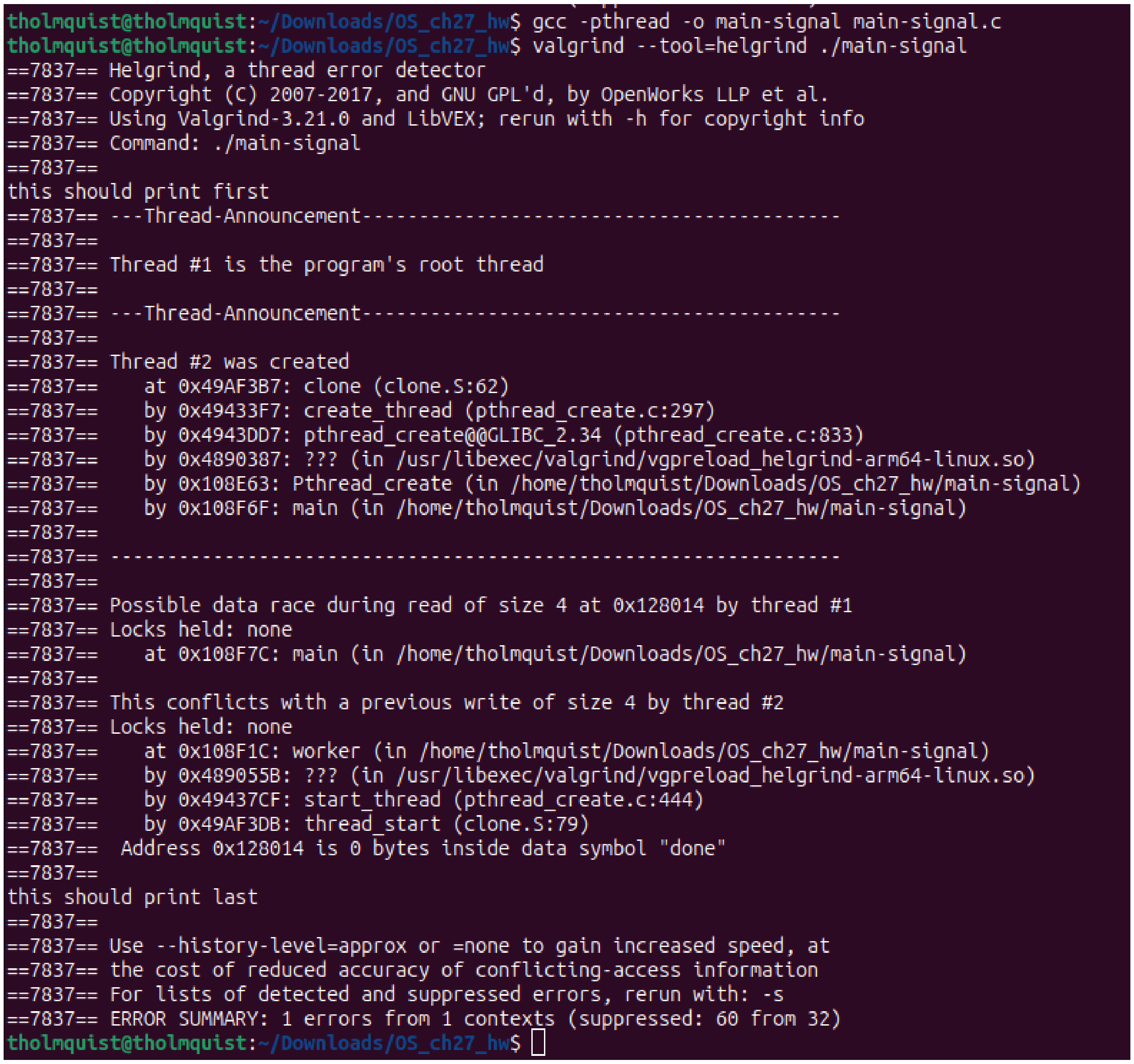
```
tholmquist@tholmquist:~/Downloads/OS_ch27_hw$ valgrind --tool=helgrind -s ./main-race
==6844== Helgrind, a thread error detector
==6844== Copyright (C) 2007-2017, and GNU GPL'd, by OpenWorks LLP et al.
==6844== Using Valgrind-3.21.0 and LibVEX; rerun with -h for copyright info
==6844== Command: ./main-race
==6844==
==6844==
==6844== Use --history-level=approx or =none to gain increased speed, at
==6844== the cost of reduced accuracy of conflicting-access information
==6844== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 7 from 6)
--6844--
--6844-- used_suppression:      7 helgrind-glibc2X-005 /usr/libexec/valgrind/default.supp:969
==6844==
==6844== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 7 from 6)
tholmquist@tholmquist:~/Downloads/OS_ch27_hw$ 
```

if you put a lock on both accesses to balance then you receive no errors from helgrind. (gcc -pthread -o main-race main-race.c)

3. It looks like depending on the argument passed in it locks the mutexes in a different order. I could see this leading to a problem where both mutexes are locked and waiting on each other endlessly. (gcc -pthread -o main-deadlock main-deadlock.c)
4. Helgrind reports that m2 was locked before m1 when the expected order is m1 then m2 by stating "Thread #3: lock order "0x128018 before 0x128048" violated
 Observed (incorrect) order is: acquisition of lock at 0x128048"

5. Helgrind should not be reporting the same error because g should protect the threads from deadlocking on m1 and m2 therefore not giving the code the same problem that main-deadlock.c had. This shows that false positives can still occur with helgrind. (gcc -pthread -o main-deadlock-global main-deadlock-global.c)
6. This code is inefficient because the parent is stuck waiting for the child to change the variable done and is doing nothing productive when it could be implementing synchronization and utilizing the cpu more efficiently. (gcc -pthread -o main-signal main-signal.c)

7.



```
tholmquist@tholmquist:~/Downloads/OS_ch27_hw$ gcc -pthread -o main-signal main-signal.c
tholmquist@tholmquist:~/Downloads/OS_ch27_hw$ valgrind --tool=helgrind ./main-signal
==7837== Helgrind, a thread error detector
==7837== Copyright (C) 2007-2017, and GNU GPL'd, by OpenWorks LLP et al.
==7837== Using Valgrind-3.21.0 and LibVEX; rerun with -h for copyright info
==7837== Command: ./main-signal
==7837==
this should print first
==7837== ---Thread-Announcement------------------------------------------
==7837==
==7837== Thread #1 is the program's root thread
==7837==
==7837== ---Thread-Announcement------------------------------------------
==7837==
==7837== Thread #2 was created
==7837==    at 0x49AF3B7: clone (clone.S:62)
==7837==    by 0x49433F7: create_thread (pthread_create.c:297)
==7837==    by 0x4943DD7: pthread_create@@GLIBC_2.34 (pthread_create.c:833)
==7837==    by 0x4890387: ??? (in /usr/libexec/valgrind/vgpreload_helgrind-arm64-linux.so)
==7837==    by 0x108E63: Pthread_create (in /home/tholmquist/Downloads/OS_ch27_hw/main-signal)
==7837==    by 0x108F6F: main (in /home/tholmquist/Downloads/OS_ch27_hw/main-signal)
==7837==
==7837== ----------------------------------------------------------------
==7837==
==7837== Possible data race during read of size 4 at 0x128014 by thread #1
==7837== Locks held: none
==7837==    at 0x108F7C: main (in /home/tholmquist/Downloads/OS_ch27_hw/main-signal)
==7837==
==7837== This conflicts with a previous write of size 4 by thread #2
==7837== Locks held: none
==7837==    at 0x108F1C: worker (in /home/tholmquist/Downloads/OS_ch27_hw/main-signal)
==7837==    by 0x489055B: ??? (in /usr/libexec/valgrind/vgpreload_helgrind-arm64-linux.so)
==7837==    by 0x49437CF: start_thread (pthread_create.c:444)
==7837==    by 0x49AF3DB: thread_start (clone.S:79)
==7837==  Address 0x128014 is 0 bytes inside data symbol "done"
==7837==
this should print last
==7837==
==7837== Use --history-level=approx or =none to gain increased speed, at
==7837== the cost of reduced accuracy of conflicting-access information
==7837== For lists of detected and suppressed errors, rerun with: -s
==7837== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 60 from 32)
tholmquist@tholmquist:~/Downloads/OS_ch27_hw$
```

This code is not correct due to a potential race condition when done is being changed. Since the main function is constantly accessing done to check it in a while loop it may not check it at the right time causing problems to the timing of the print statements.