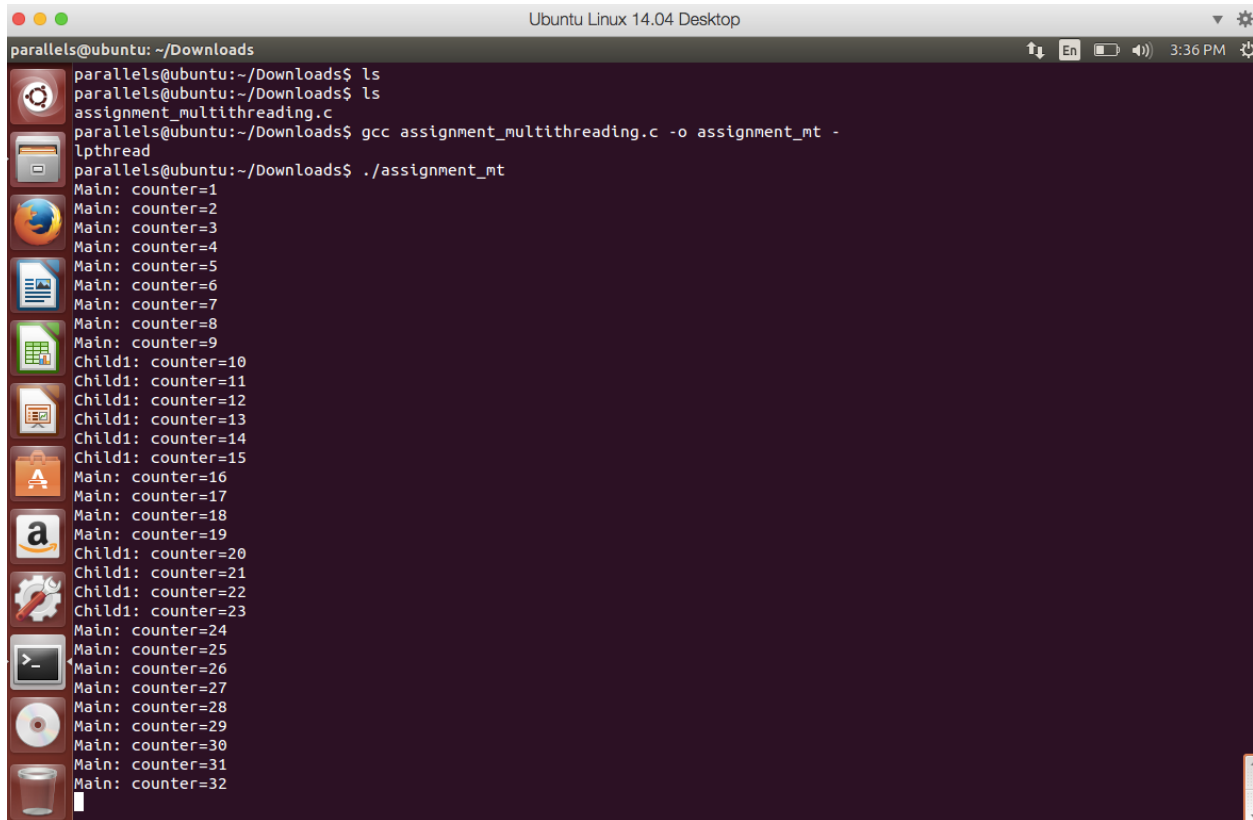


Report on Assignment 4

1. a) Run the program and explain the output.



```
parallels@ubuntu: ~/Downloads
parallels@ubuntu:~/Downloads$ ls
assignment_multithreading.c
parallels@ubuntu:~/Downloads$ gcc assignment_multithreading.c -o assignment_mt -lpthread
parallels@ubuntu:~/Downloads$ ./assignment_mt
Main: counter=1
Main: counter=2
Main: counter=3
Main: counter=4
Main: counter=5
Main: counter=6
Main: counter=7
Main: counter=8
Main: counter=9
Child1: counter=10
Child1: counter=11
Child1: counter=12
Child1: counter=13
Child1: counter=14
Child1: counter=15
Main: counter=16
Main: counter=17
Main: counter=18
Main: counter=19
Child1: counter=20
Child1: counter=21
Child1: counter=22
Child1: counter=23
Main: counter=24
Main: counter=25
Main: counter=26
Main: counter=27
Main: counter=28
Main: counter=29
Main: counter=30
Main: counter=31
Main: counter=32
```

The main function initializes a mutex and creates a thread, in which child1 function runs. Then, it locks the mutex, increase the counter by 1, print its value and then it unlocks the mutex. The program runs in above sequence in infinite loop.

Meanwhile, the child1 function starts running. It locks the same mutex, which, if already locked, will put the child1 function in blocked state, until the mutex is unlocked by the function that locked it at the first place, which, in this case is the main function. Then, the child1 function locks the mutex and goes to the ready queue, to be run by the processor. Then, it increases the counter by 1, prints its value and unlocks the mutex. All of this also happens in an infinite loop.

The same procedure happens in the main function if its mutex locks while child1 has it locked already, then the former has to wait for the latter to unlock the mutex, going through the steps explained above.

Therefore, the output is the counter number, which is printed one time by the main function, and the other time by the child1 function, alternating between them in each of their loops. This happens until the main function prints the number 32, when the program reaches a deadlock.

b) Why do the print statements stop appearing after a certain point in the program? Explain.

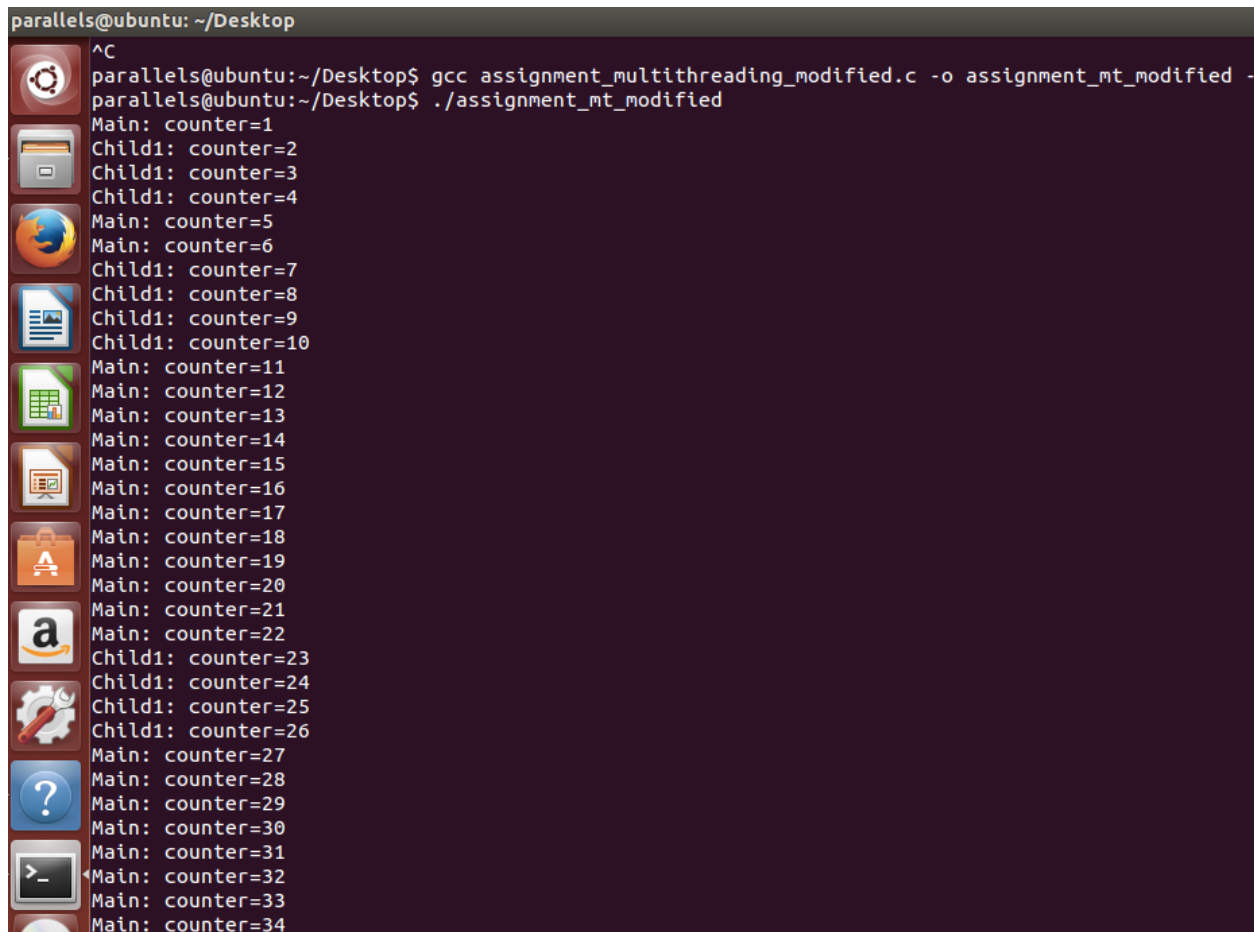
Since the thread that runs child1 is exited before unlocking the mutex. Then, as the mutex will be locked forever, the main function cannot run, and the program reaches a deadlock.

When the counter is more than 25, the next time child1 runs (when the counter is at 32), there is a condition that tells the thread to exit. However, this condition is after the mutex lock and before the mutex unlock. And, when a thread is exited, it doesn't automatically unlocks the mutex. A way to fix that will be presented below.

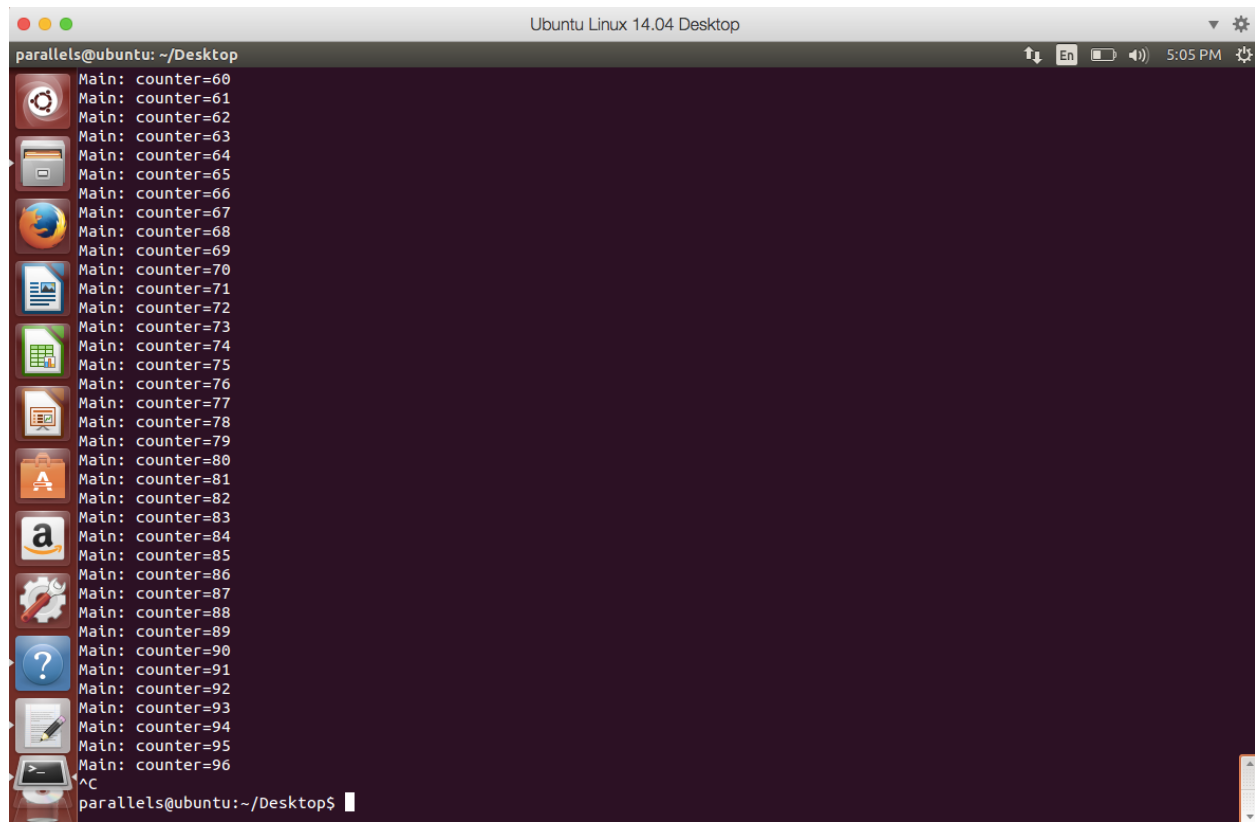
2. Modify the program and write a correct version that fixes the problem that you just discovered. Explain how you fixed the program.

My way to fix this problem is to exit the thread after unlocking the mutex. This has been done in the file `assignment_multithreading_modified.c`.

In this file, the main thread keeps counting and printing after the second thread is exited. The running output is showed as followed.

A terminal window titled 'parallels@ubuntu: ~/Desktop' showing the execution of a C program. The user enters the command 'gcc assignment_multithreading_modified.c -o assignment_mt_modified' followed by './assignment_mt_modified'. The output shows a sequence of 'Main: counter=' and 'Child1: counter=' messages. The 'Main' thread prints from counter 1 to 34. The 'Child1' thread prints from counter 2 to 26, then stops. The terminal has a dark background with a sidebar of application icons on the left.

```
parallels@ubuntu: ~/Desktop
^C
parallels@ubuntu:~/Desktop$ gcc assignment_multithreading_modified.c -o assignment_mt_modified
parallels@ubuntu:~/Desktop$ ./assignment_mt_modified
Main: counter=1
Child1: counter=2
Child1: counter=3
Child1: counter=4
Main: counter=5
Main: counter=6
Child1: counter=7
Child1: counter=8
Child1: counter=9
Child1: counter=10
Main: counter=11
Main: counter=12
Main: counter=13
Main: counter=14
Main: counter=15
Main: counter=16
Main: counter=17
Main: counter=18
Main: counter=19
Main: counter=20
Main: counter=21
Main: counter=22
Child1: counter=23
Child1: counter=24
Child1: counter=25
Child1: counter=26
Main: counter=27
Main: counter=28
Main: counter=29
Main: counter=30
Main: counter=31
Main: counter=32
Main: counter=33
Main: counter=34
```



```
parallels@ubuntu: ~/Desktop
Main: counter=60
Main: counter=61
Main: counter=62
Main: counter=63
Main: counter=64
Main: counter=65
Main: counter=66
Main: counter=67
Main: counter=68
Main: counter=69
Main: counter=70
Main: counter=71
Main: counter=72
Main: counter=73
Main: counter=74
Main: counter=75
Main: counter=76
Main: counter=77
Main: counter=78
Main: counter=79
Main: counter=80
Main: counter=81
Main: counter=82
Main: counter=83
Main: counter=84
Main: counter=85
Main: counter=86
Main: counter=87
Main: counter=88
Main: counter=89
Main: counter=90
Main: counter=91
Main: counter=92
Main: counter=93
Main: counter=94
Main: counter=95
Main: counter=96
^C
parallels@ubuntu:~/Desktop$
```