

Pthreads

Lab #2



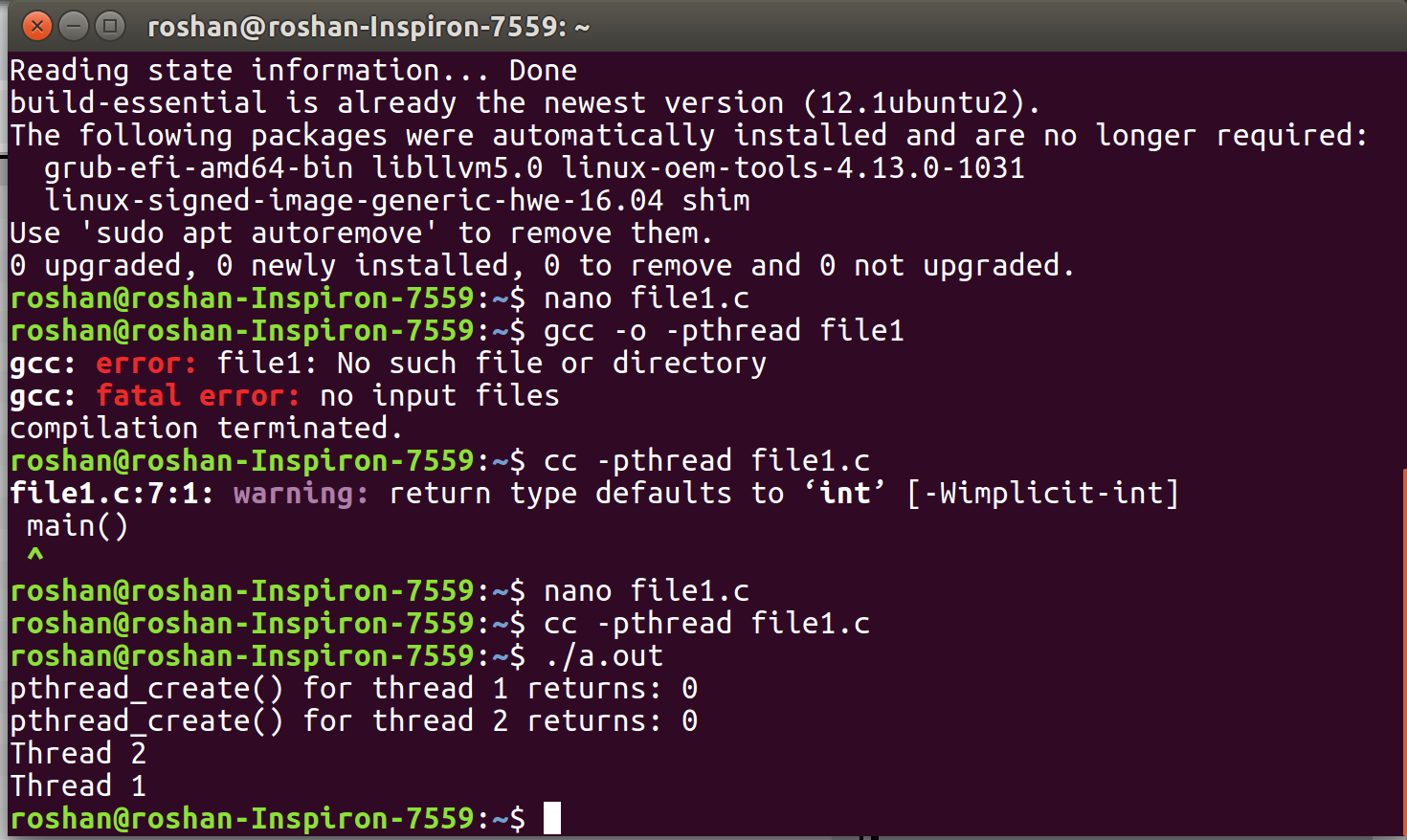
September 19, 2018

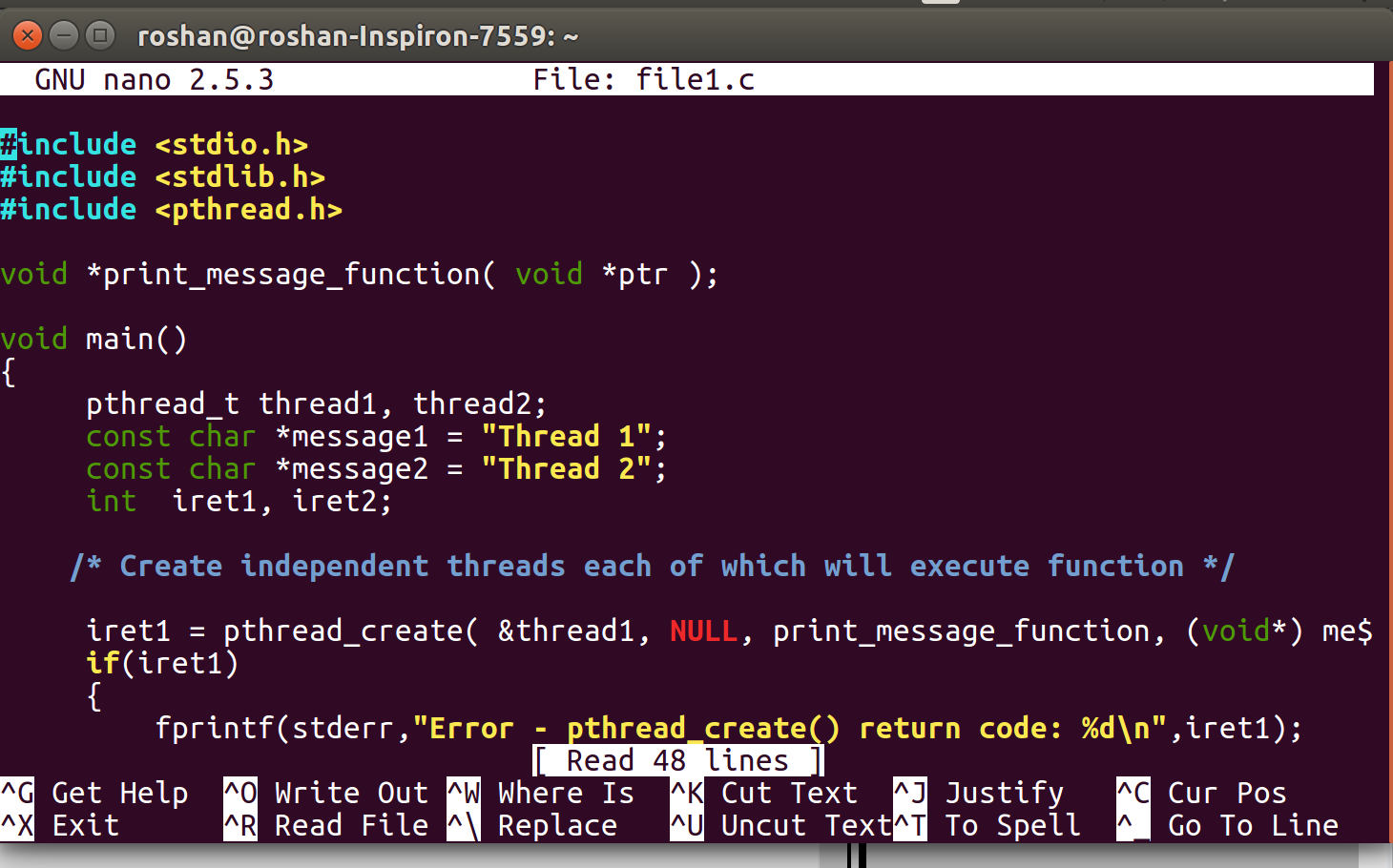
roshan mughees

193590

# Code 1:

* First of all I copied the code from the link given in our lab manual and then in bash I created a file by using this command “nano file1.c” and then past the code inside it and then close the file.
* To compile the file the command which I used is this “cc -pthread file1.c” and then to execute it I used this command “./a.out” .
* In this code we are creating two new thread by using pthread\_create() function. The function will print the message and it returns the thread id.
* If there is a error in creating thread then message will be printed.
* Pthread.exit() is used to terminate the program.

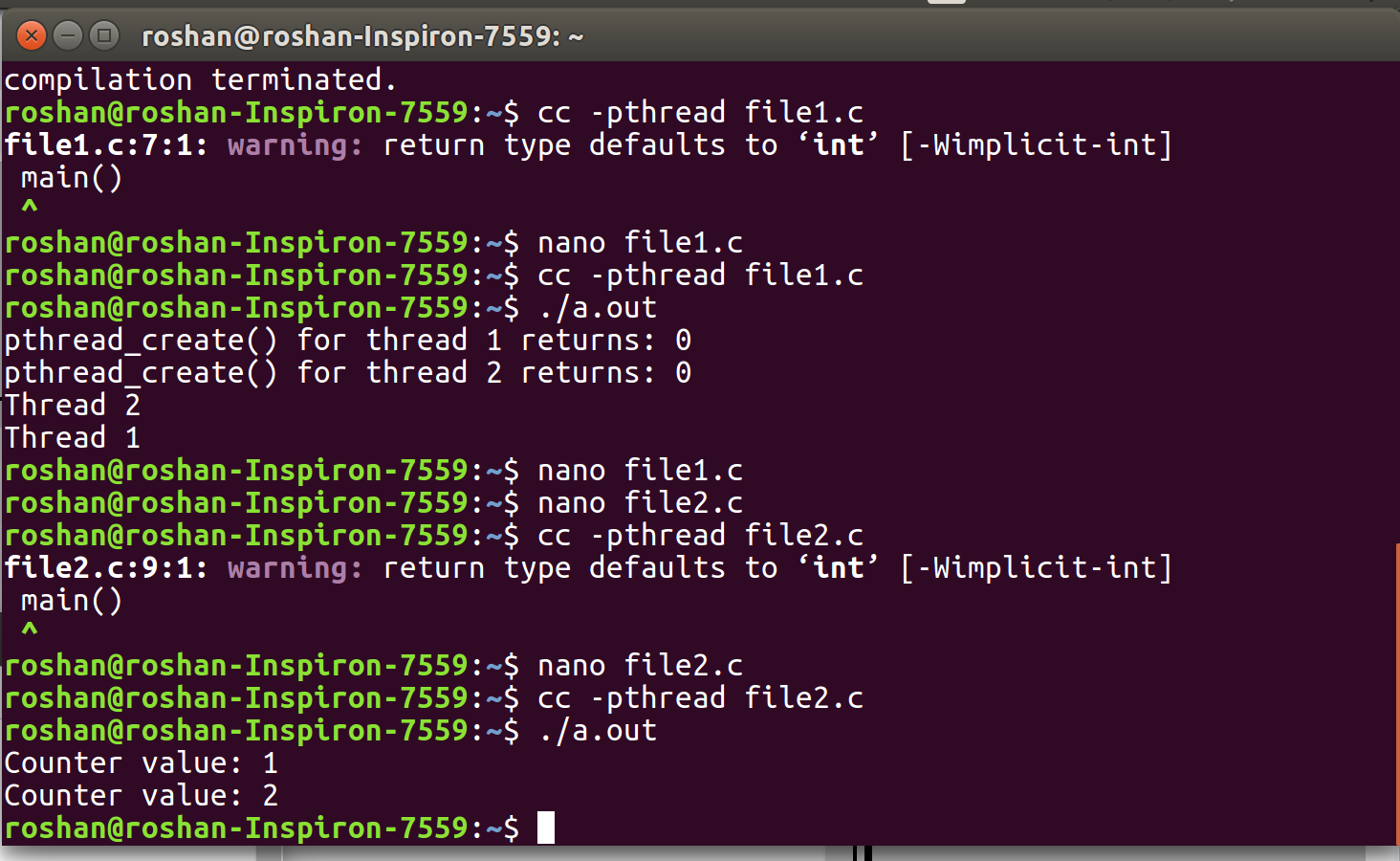


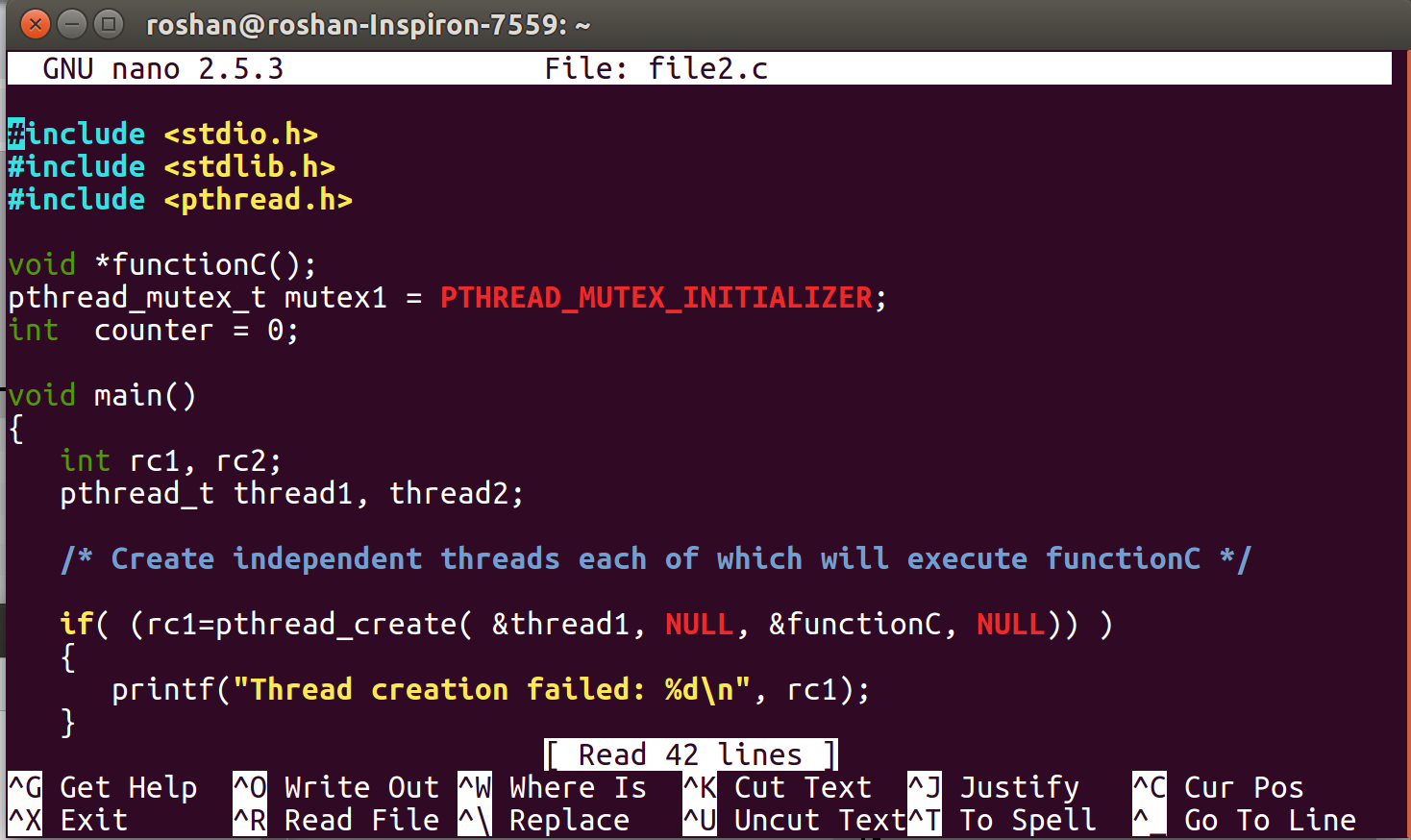


Explanation:

# Code 2:

* First of all I copied the code from the link given in our lab manual and then in bash I created a file by using this command “nano file2.c” and then past the code inside it and then close the file.
* To compile the file the command which I used is this “cc -pthread file2.c” and then to execute it I used this command “./a.out” .
* Here pthread\_function() is used to create two threads.
* Different threads are performing different operation on same memory location so to prevent the data inconsistency here we are using mutexes .
* Just because of this if a thread try to access the location of memory which is currently in used of another thread then it is not allowed to him to do that.
* 1st thread will access the counter variable and increment it, on the other hand 2nd thread will remain block as mutex is locked . When mutex is unlocked 2nd thread access the value of variable.





# Code 3:

* First of all I copied the code from the link given in our lab manual and then in bash I created a file by using this command “nano file3.c” and then past the code inside it and then close the file.
* To compile the file the command which I used is this “cc -pthread file3.c” and then to execute it I used this command “./a.out” .
* In this code the main focus is on thread joins. Once all the threads are created we are calling pthread\_join() function to make sure that next instruction does not start executing before executing all the threads.
* And as a result final counter value is printed on the screen. ­

