**NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES**



**OPERATING SYSTEM PROJECT REPORT**

**SLEEPING BARBER(SYSTEM CALL)**

**GROUP MEMBERS:**

Syed Meesum Ali Meer (21K-3595)

Silmi Maknojia (21k-3607)

**SUBMITTED TO:**

Ms Rabia Ansari

**SECTION:**

BCY-4A

**DATE OF SUBMISSION:**

8TH MAY 2023

**Sleeping Barber Problem (system call) Using Semaphores**

**Introduction**

This project is based on the procedure of implementing sleeping barber mechanism as a system call in linux kernel. It can help you in how you can deal with starvation problem and deadlock problem and how at kernel level system calls can be made

**How We Started**

First of all, we did research on our project and then we started writing code at user level for sleeping barber by looking at our OS lab manuals of semaphores and threading and then we searched in Google on how to create a semaphore at kernel level and how to create threads at kernel level and after getting it from the reference link which is attached below then we wrote the code for creating a system call for our sleeping barber problem and finally we did kernel configuration using the link which is attached below

**Problems faced**

The problems arise when we wanted to create a system call for sleeping barber problem firstly since our linux version was greater than 5.0 and asmlinkage support is only available for linux versions below 5.0 and due to that we replaced it with syscall\_define0(‘systemcall name’) . Another problem arise when we declared chairs and customers more than 10 then our Ubuntu operating system would get hanged due to limited kernel space and as a result we changed chairs to 3 and customers to 5 so less kernel space is used.

**Working of source code**

Firstly we have declared semaphores and threads globally using struct and then we moved to syscall\_define0 () function where we have initialized our semaphores which are waitingroom to 3, barber\_chair to 1, barber\_sleeping to 0 and barber\_seatbelt to 0. Then we have created a barber thread using kthread function and after that in a loop we have a created threads for customers using kthread function. Then with the help of two functions which are barber and customer our source code would work as follows:

Firstly the barber is sleeping in his shop. When the customer arrives, customer checks if the barber is sleeping and if barber is sleeping then customer wakes the barber up and if barber is cutting another customer’s hair then customer goes to the waiting room to check if there are seats available and sits on the seat if seat is available but if there are no seats available, then customer leaves barber shop and comes when seats are available in the waiting room. Once the barber finishes cutting hair, he checks if there are any customers in the waiting room, if not than the barber goes back to sleep and after all customers have got their haircut then all the threads would stop using kthread\_stop() function.

**Technology**

Platform : **Ubuntu 16.04**

Coding : **C languag**

**Project Code**

Firstly create a system call code

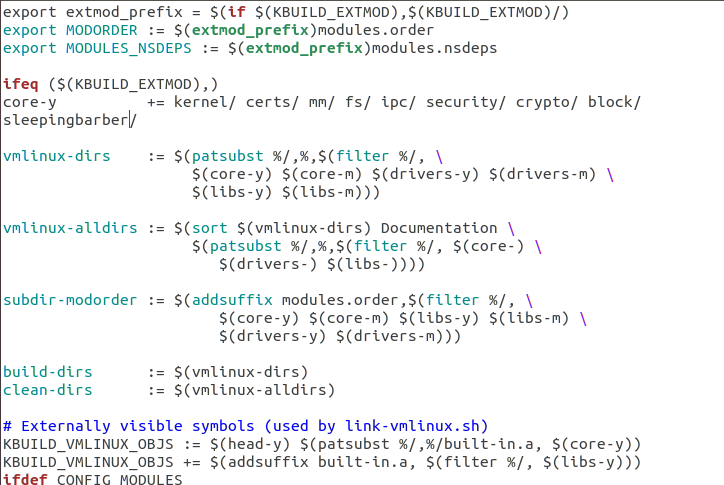
**System Call Code**

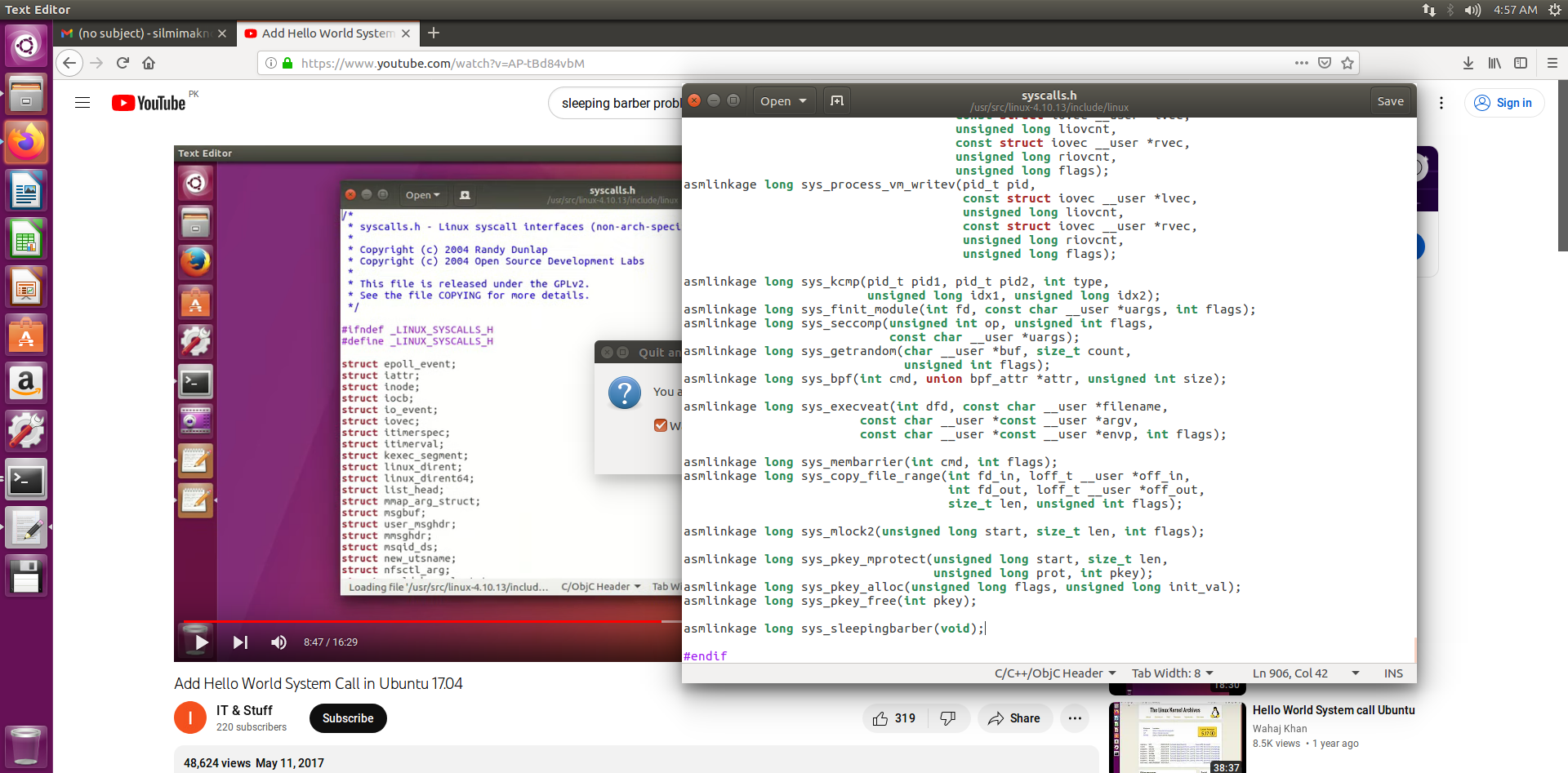


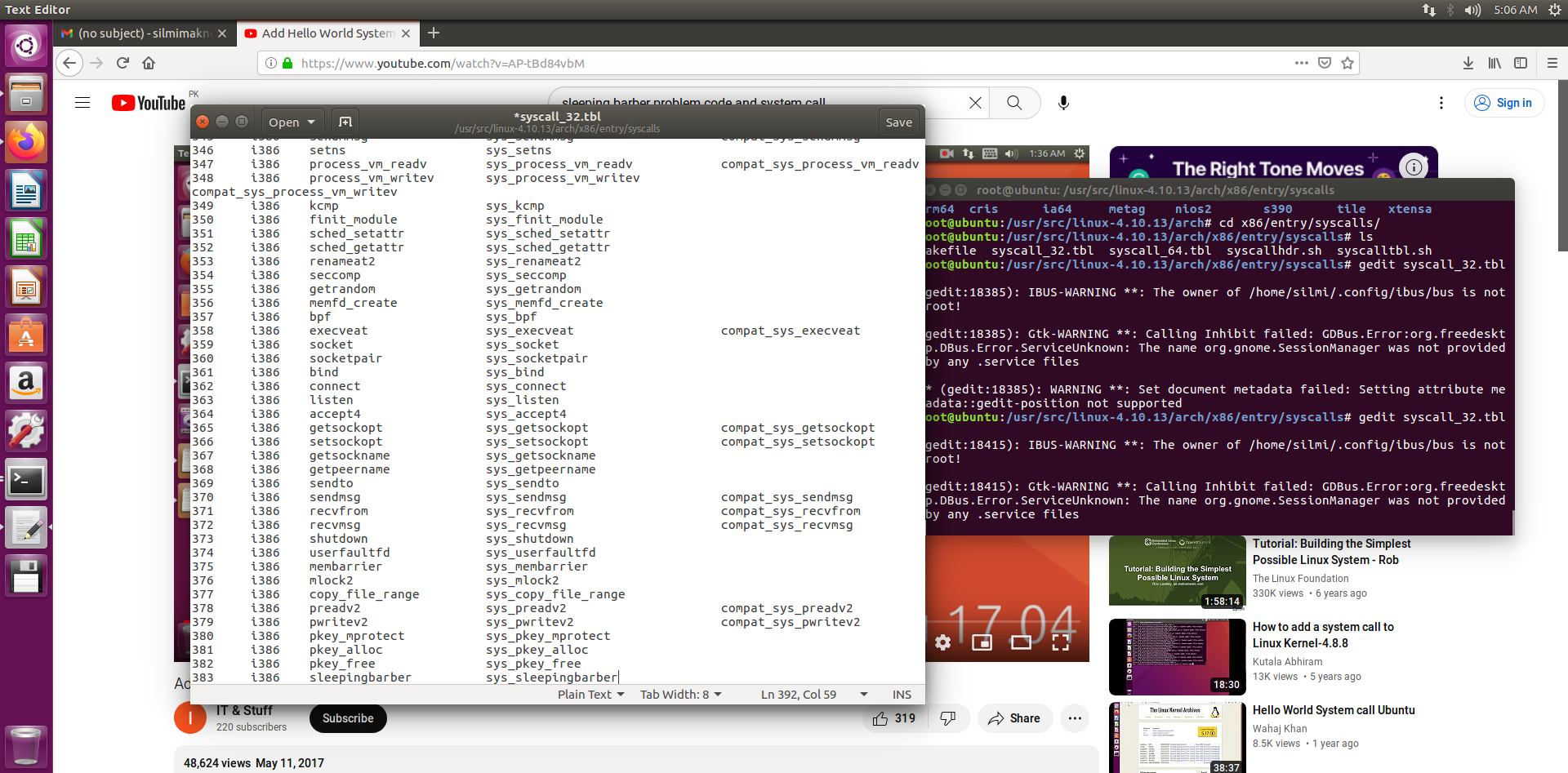
Then create a Makefile for your system code



After that do **kernel configuration**







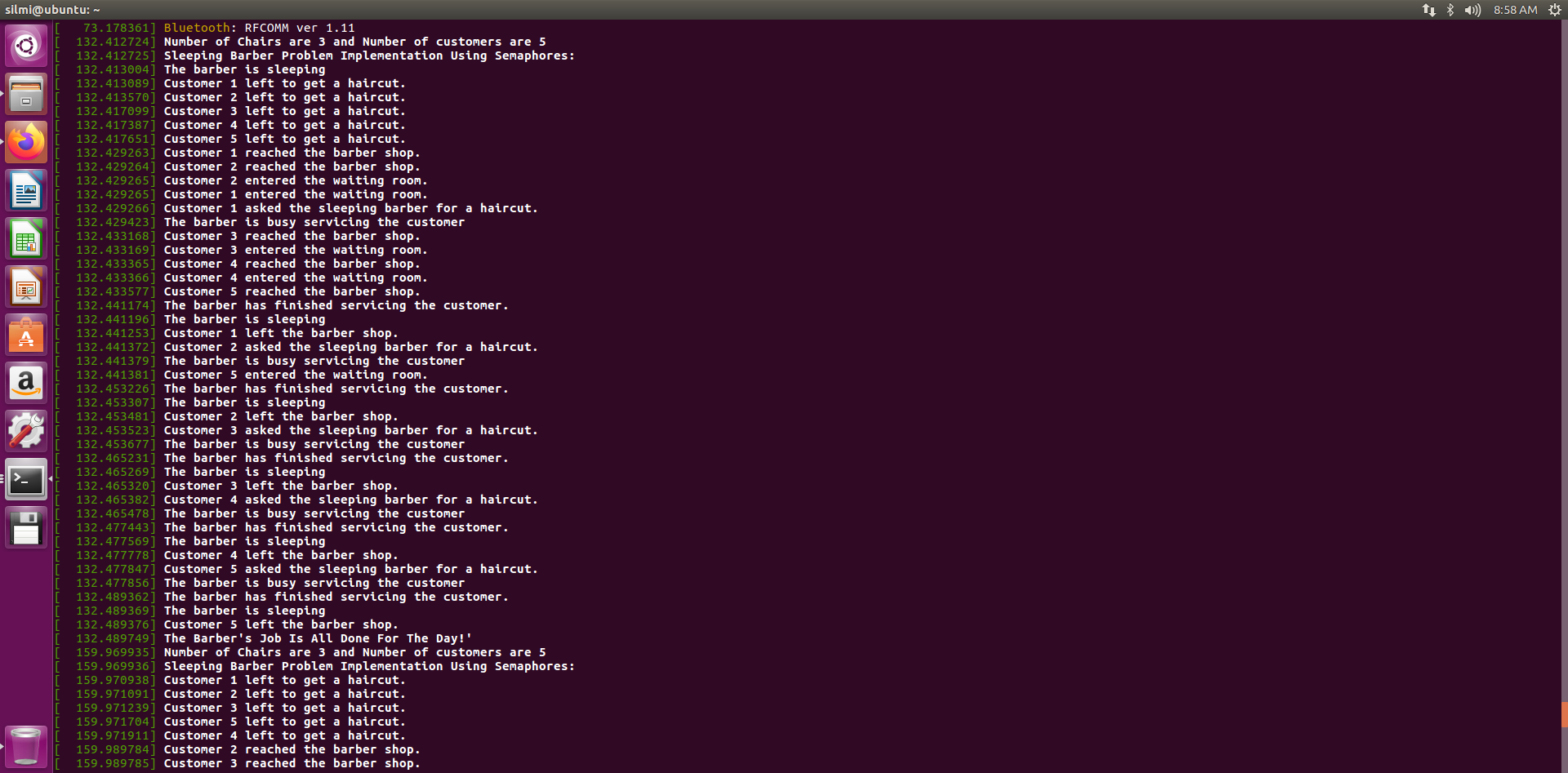
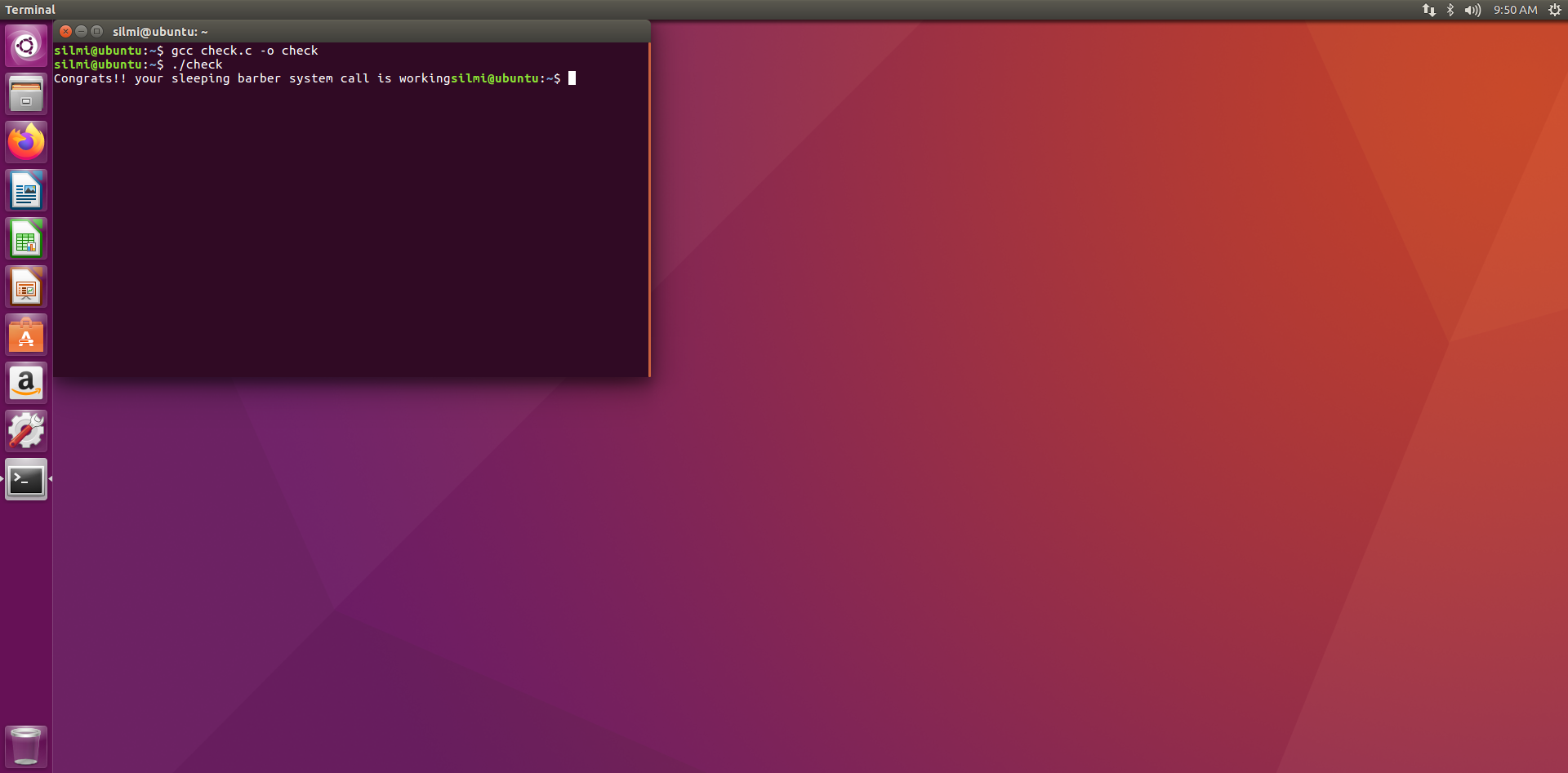
**Output**

To check if your system call is working properly compile the below code and run the below code and then type “**dmesg**” in the command terminal.

**Code**



**Output**



**References**

<https://www.geeksforgeeks.org/sleeping-barber-problem-in-process-synchronization/> (for understanding sleeping barber)

<https://0xax.gitbooks.io/linux-insides/content/SyncPrim/linux-sync-3.html> (for semaphore and process management)

Stackoverflow.com

<https://www.youtube.com/watch?v=AP-tBd84vbM> (for kernel configuration)