**Project Report**

**Sleeping Barber Problem using Semaphores**

**1.** **DESCRIPTION:**

The barber is sleeping in his shop. When a customer arrives, the checks if the barber is sleeping if yes then he wakes the barber up. If there are no other customers in the waiting room, the barber cuts his hair, else the customer sit in the waiting room. But if there are no seats available, the customer leaves. 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**.**

**2.** **HOW WE STARTED:**

First of all, we did research on our project and then we started making codes, we took help from our OS lab manuals then we started making kernel configuration with the help of the reference link.

**3.** **PROBLEMS FACED:**

We had a lot of trouble creating the sleeping barber code because we didn't know what the Kernel library we need to used.

**4.** **THE ACTUAL WORKING OF THE PROJECT (Methodology):**

The methodology uses three semaphores: one which counts the number of waiting customers, second the barber is idle or working, and mutex for mutual exclusion which is required for the process to execute.

**5.** **EXPLANATION OF SOURCE CODE:**

The barber is sleeping in his shop. When a customer arrives, the checks if the barber is sleeping if yes then he wakes the barber up. If there are no other customers in the waiting room, the barber cuts his hair, else the customer sits in the waiting room. But if there are no seats available, the customer leaves. 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**.**

**6.** **PROJECT ASPECTS IN CURRENT/FUTURE TECHNOLOGY:**

In future if someone is working with multi threads and he don't know the actual working and syntax then our project can help them with that, and also if they face the deadlock problem or any starvation problem then they can take help from our project about how to deal with deadlocks, starvation and also, they can get the idea how the Interprocess Communication works.

**7.** **Project configurations and Code**

1. **Code**
2. System call file



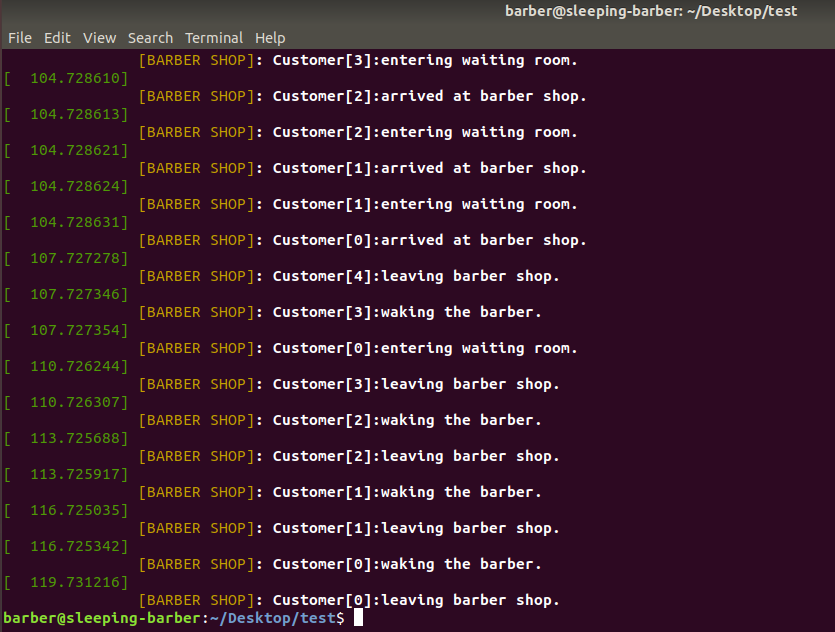
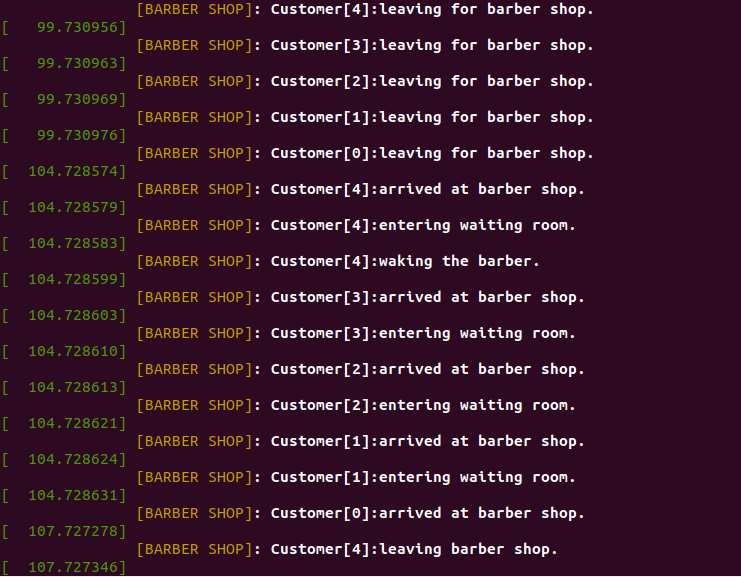
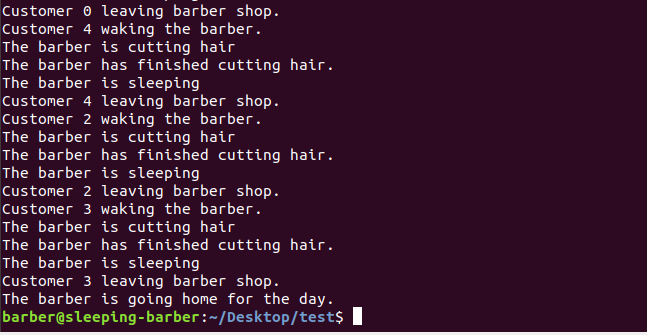
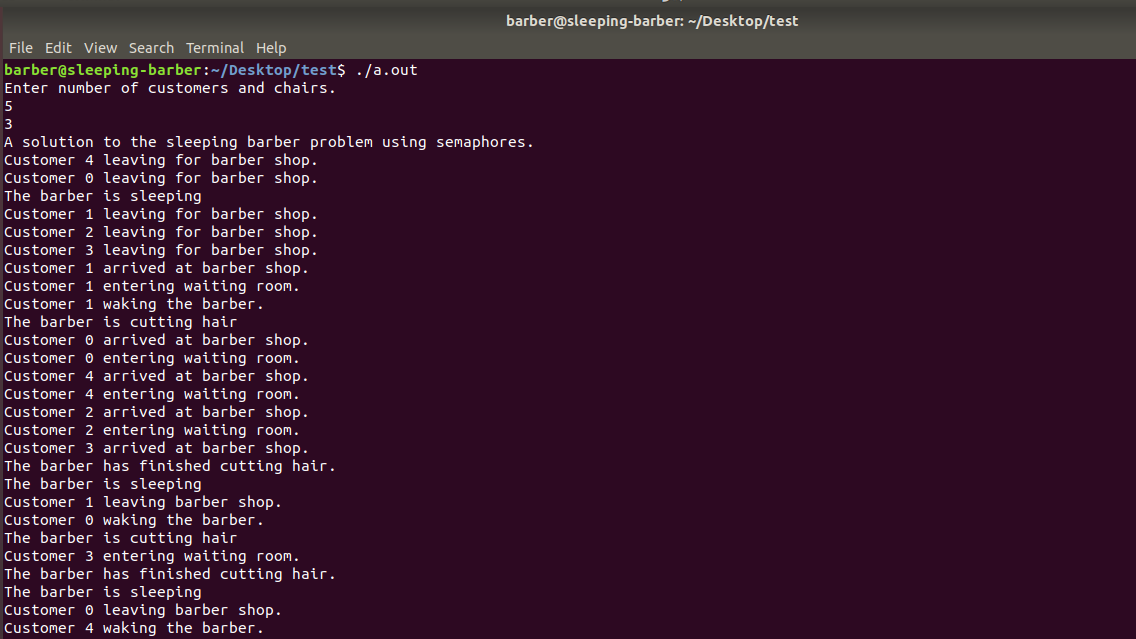
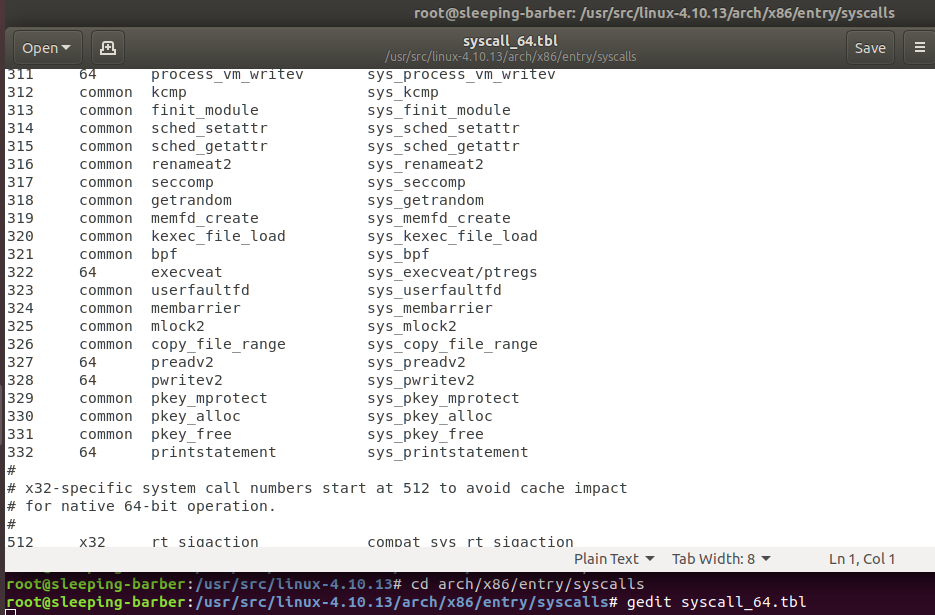
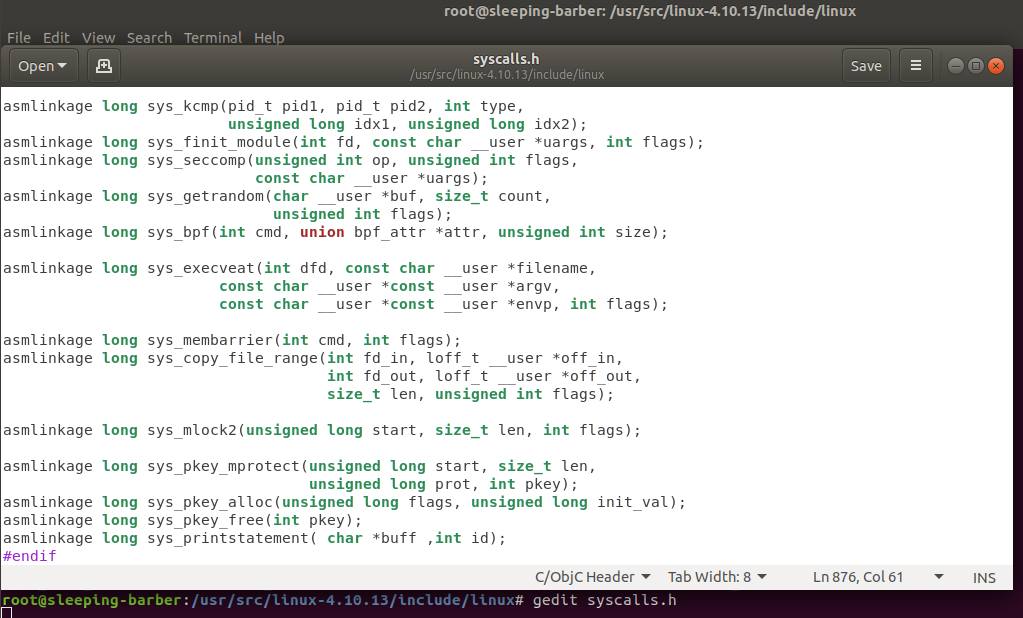
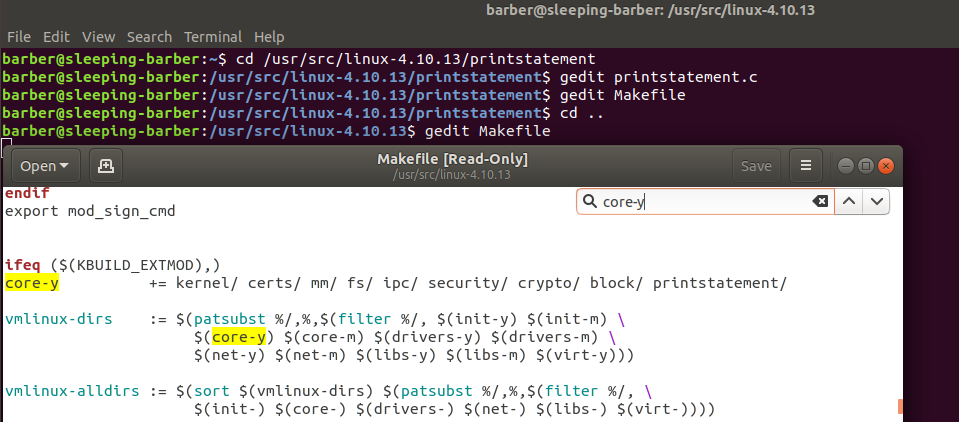
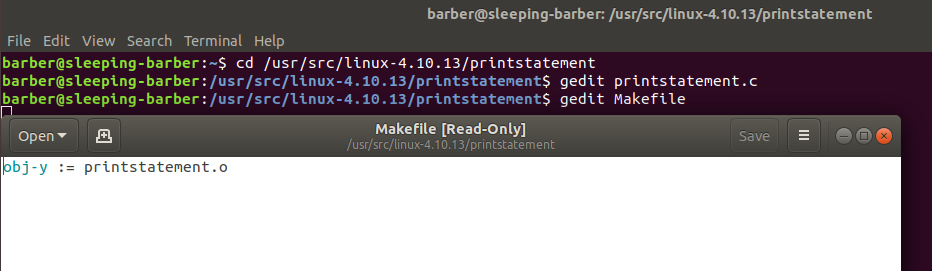
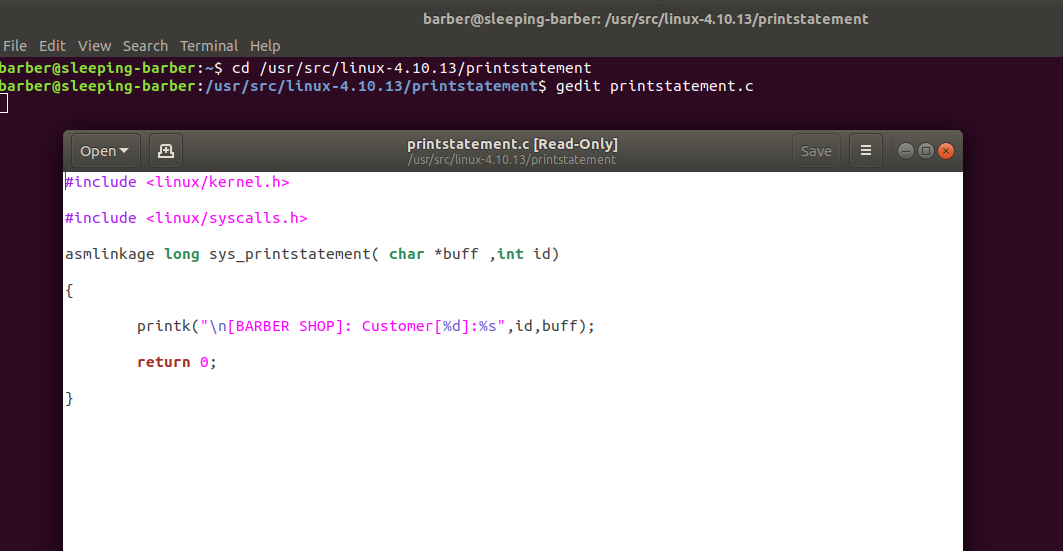
1. Make file



1. User level Sleeping barber file

****

1. **Configuration**

****

**8.** **Conclusion**

In conclusion, we had a problem which can be implement through multithreading but it causes the problem of deadlock which we solved through using three semaphores.

**REFERENCES:**

[System call](https://youtu.be/AP-tBd84vbM)