National University
Of Computer and Emerging Sciences

Chain Smoker Problem Project Report

Course Name and Code:

Operating Systems (CS2006)

Under the supervision of:

Dr. Nausheen Shoaib

Team Members:

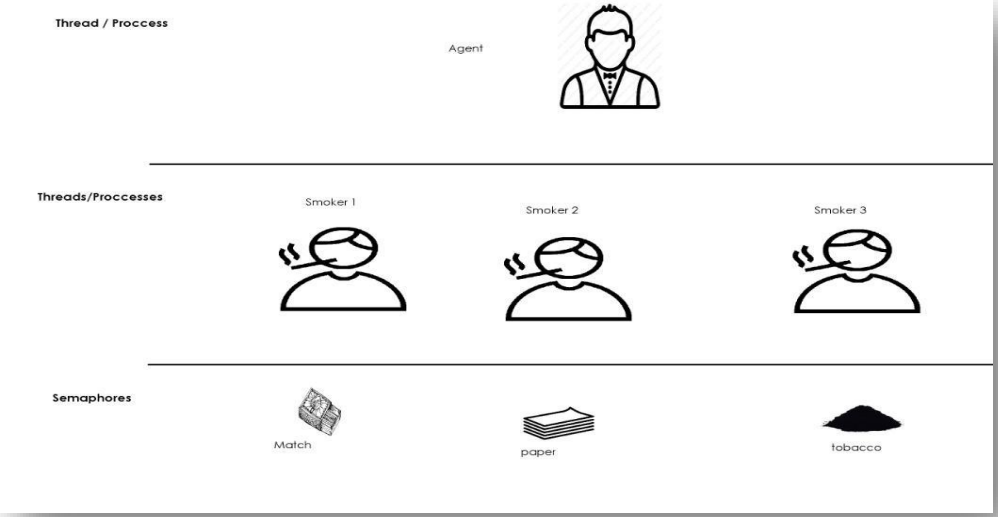
1. Syed Zeeshan Ahmed 21K-4844
2. Aqib Ali Shah 21K-4518
3. Khalid Khurshid Siddiqui 21K-4673

Introduction:

Assume a cigarette requires three ingredients to make and smoke: tobacco, paper, and matches. There are three smokers around a table, each of whom has an infinite supply of one of the three ingredients — one smoker has an infinite supply of tobacco, another has paper, and the third has matched.

There is also a non-smoking agent that enables the smokers to make their cigarettes by arbitrarily (non-deterministically) selecting two of the supplies to place on the table. The smoker with the third supply should remove the two items from the table, using them (along with their own) to make a cigarette, which they smoke for a while. Once the smoker has finished his cigarette, the agent places two new random items on the table. This process continues forever.

Three semaphores are used to represent the items on the table; the agent increases the appropriate semaphore to signal that an item has been placed on the table, and smokers decrement the semaphore when removing items. Also, each smoker has an associated semaphore that they use to signal to the agent that the smoker is done smoking; the agent has a process that waits on each smoker's semaphore to let the agent know that it can place the new items on the table.



Background:

In the background we have 3 threads (smokers) one for each smoker, each smoker has one of the 3 ingredients tobacco, match, and paper, each smoker will require the 2 other ingredients to achieve a critical section (smoke). Here we get the agent; the agent contains all three resources and will randomly pick two ingredients and present them to the smokers. Many unique situations can be derived from it.

For example, the agent presents the smokers with Tobacco and Match and the only way for this scenario to work out is for the smoker with the paper must comfortable and form a cigarette to smoke if any other smoker comes worth with the wrong ingredients, we will enter into a state of Deadlock, to avoid this condition we have put forward our solution and its implementation.

Each Smoker will have an infinite supply of the ingredient (Tobacco, Paper, and Match) it has, and the agent will have an infinite supply of all three ingredients. Still, it has to randomly present the ingredient to the smokers (threads) so that they may function.

Objective:

We, the students of Computer Science, are the future of the next technological revolution. We are living in an era, where Technology is changing so fast that, it is hard to keep track of. However, behind every modern-day tech, there is a major role in algorithm design. No tech can prevail in this era without a solid algorithm design.

The operating system is a very vast sea of algorithms and designs, and in this, there are numerous opportunities to practice and implement algorithms. Keeping all these things in sight, we decided to work on the chain smoker problem because of the beautiful question it possesses for synchronization.

The objective of this project was to understand the synchronization problem proposed by the author to build a solution that can solve the problem, gain

experience from the implementation, and use it as a stepping-stone for future projects.

Result:

The Deadlock condition between the smokers can be avoided and resolved by using three semaphores. To design the complete solution, we use four threads to simulate Agent, Smoker with an infinite supply of Tobacco, Smoker with an infinite supply of matchsticks, and Smoker with an infinite supply of paper.

The agent thread will be used to unlock any two random semaphores (putting items on the table). In that brief time, the smoker thread, which has a third ingredient, will enter the critical, section and will execute (according to the problem he will make and then smoke the cigarette).

This project made us understand the complexity of deadlock and how to avoid them, for in the years to come the world is developing, and in order to avoid differences and achieve maximizing the chances of our success we had to bring forth steps of change, change in syntax, change in

Platform:

While working on a project it is necessary to know what Platform will be most aligned to help out or work out for your project The Platform on which this project was performed was Ubuntu 22.04.1 and the Linux kernel version was Linux-4.19.282. We used A **Virtual Box** to use Ubuntu 22.04.1 and did our project on it.

This project is of such a type that it cannot be Platform independent and we had to decide its parameters and take into consideration carefully its minute requirements to increase the chance of our success.

Languages:

As the project was done on the kernel level the language we used was C, and researched some other functions and libraries of the C-Language to help us in successfully executing our program, for example, in this project pthreads and semaphores were very useful, and used a lot. Slight differences occurred as this project involved a system call and we had to make severe changes to achieve the desired syntax used in a system call, we had to keep in mind that many things that are part of a normal C program will not be used in the same way in the kernel programming because it requires different syntax for it to function properly.

Methodologies:

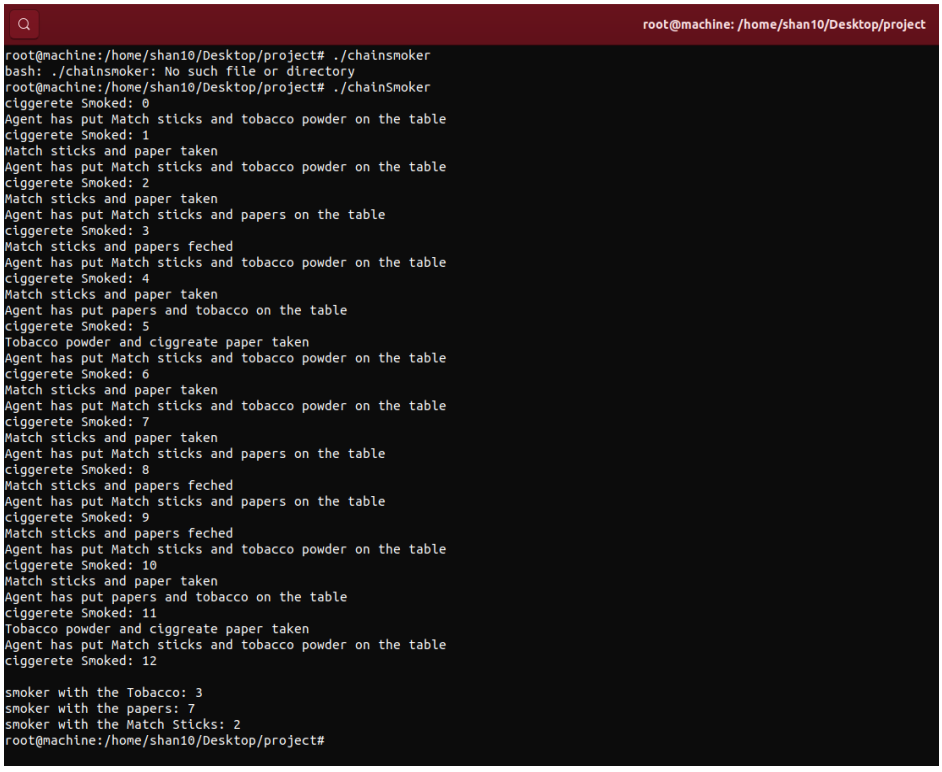
As for the successful implementation of our project, we used Pthreads and Semaphores for the synchronization of the code. We tried to do our code with just one system call, but it was not adding up, so we had to introduce 3 separate system calls to ensure the correct working of the code. For the help on System call, we used this manual available to us online:

https://dev.to/jasper/adding-a-system-call-to-the-linux-kernel-5-8-1-in-ubuntu-20-04-lts-2ga8 .

We had to follow these instructions because we were using the latest version of Ubuntu and there were not so many guidelines for system calls to run on it. These proved very helpful and proved rather useful in our journey to accomplish the task.

Conclusion:

Because of hard work and good coordination between the team members, not only did we manage to solve the chain smoker problem Furthermore, but we also added three system calls (for all three Smokers) in the Linux kernel Linux-4.19.282 to make the solution interactive and easy to understand, the purpose of this build will be to show the interactivity of the semaphores and threads so that to avoid a circumstance which could lead to a deadlock and waste all of our valuable resources., It will professionally guide the code/ program to avoid deadlocks and ensure the smooth running of the code that each section will run smoothly.



Useful Links / Additional Readings

https://dev.to/jasper/adding-a-system-call-to-the-linux-kernel-5-8-1-in-ubuntu-20-04-lts-2ga8

https://www.kernel.org/

https://www.youtube.com/watch?v=qcsKGWshRZE&t=1023s

https://medium.com/anubhav-shrimal/adding-a-hello-world-system-call-to-linux-kernel-dad32875872#:~:text=Add%20new%20system%20call%20to%20the%20system%20call%20header%20file%3A&text=Add%20the%20following%20line%20to%20the%20end%20of%20the%20document,Save%20and%20exit.

https://www.virtualbox.org/