

OPERATING SYSTEM

PROJECT REPORT

**“**SYSTEM CALL IMPLEMENTATION

of CHAIN SMOKER

PROBLEM**”**

***SEC:***

*BSE-4A*

***TEACHER:***

*MS. Mubashara Fayyaz*

***TEAM MEMBERS:***

*Khizar Alam 21k3868*

*Taha Ali 21k3867*

# INTRODUCTION

The goal of this project is to develop a system call that addresses the issue of chain smoking. Application programmers can request services from the operating system using system calls, which can either be user system calls (without kernel intervention) or kernel system calls (with kernel intervention). By developing system calls we can enable efficient and effective communication and synchronization between the agent and the smokers. Overall, this project aims to improve the performance and functionality of problems with respect to system call to handle synchronization problems.

**FEATURES**

Main function deals with the creation, and deletion of threads, and semaphores. This problem has four processes, three smoker processes, and one agent process. Each of the smoker procedures will create and smoke a cigarette. Tobacco, paper, and matches are needed to produce a cigarette. One of the three components is present in each smoking procedure. To put it another way, one procedure uses tobacco, another uses paper, and yet another uses matches. All three are infinitely available to the agent. Two of the three objects are placed on the table by the agent, and the smoker with the third item lights the cigarette.

* The highlighted features of­­ this project will be:
  + Effective communication
  + Handle multiple processes.
  + Synchronization between processes
  + Handling deadlock
  + And other management activities.

**TECHNOLOGY USED**

The tools that we will use in our project will be:

- Ubuntu 16.04 LTS desktop version

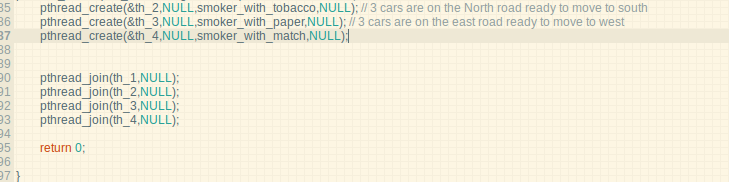
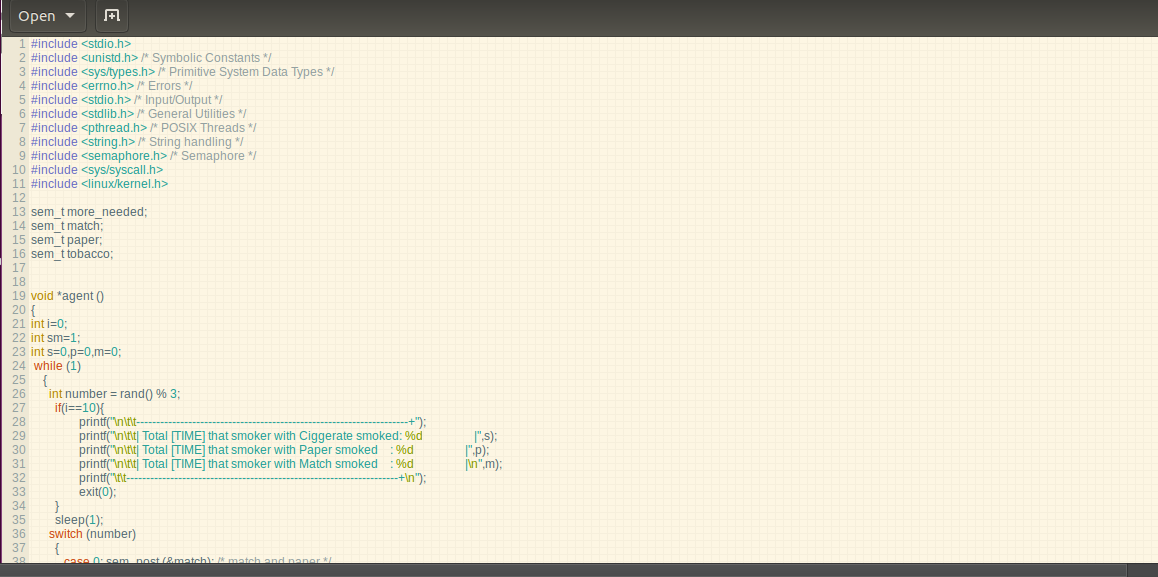
- Kernel for system call

- C/C++ for coding/implementation

- Operating system will be Linux 64 bit.

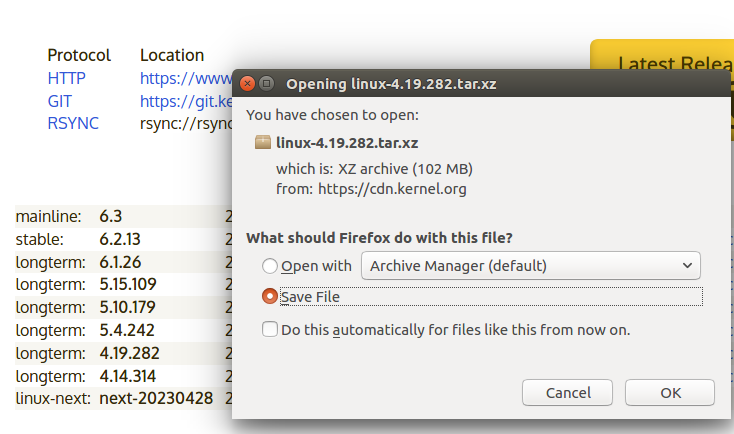
- VBOX virtual machine

**CODE SNIPPETS**



**STEPS FOR KERNEL MODIFICATION**

1. At, first download the latest kernel from kernel.org and extract it into the download folder.

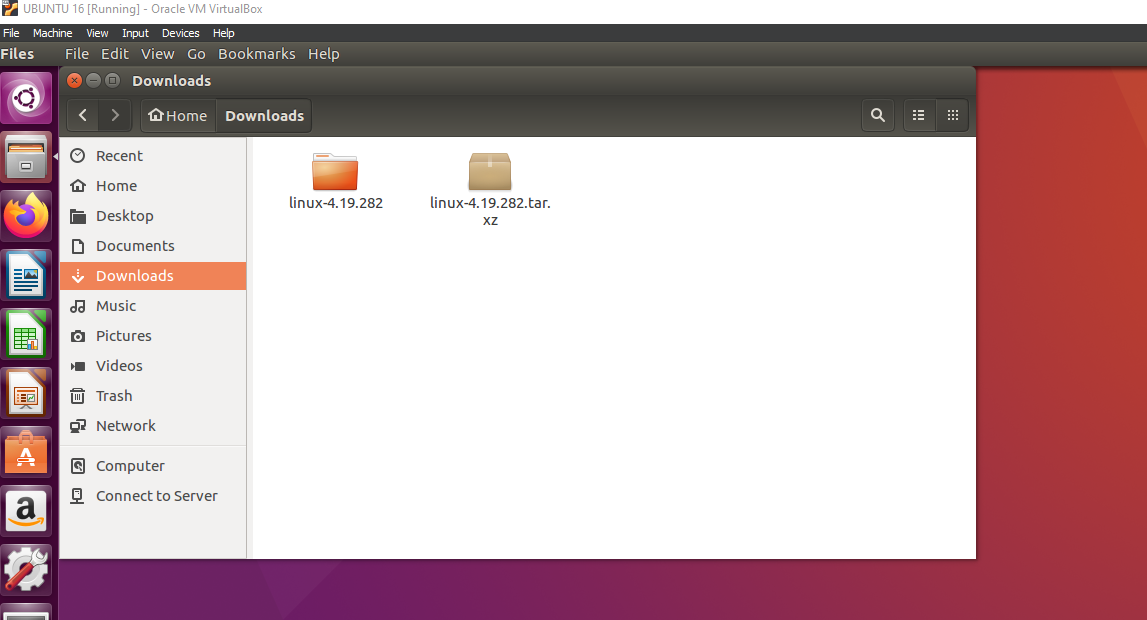


1. After downloading the kernel from website then just download the perquisites required to compile the program.

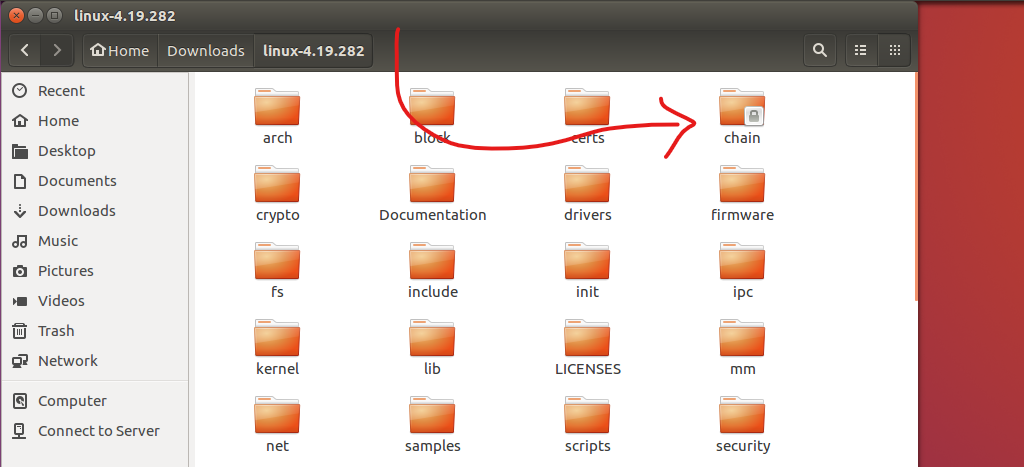


**KERNEL LEVEL CODE:**

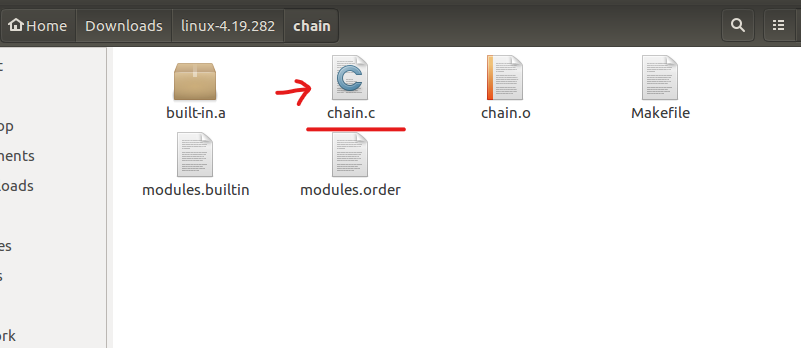
1. Then do into the extracted linux folder.



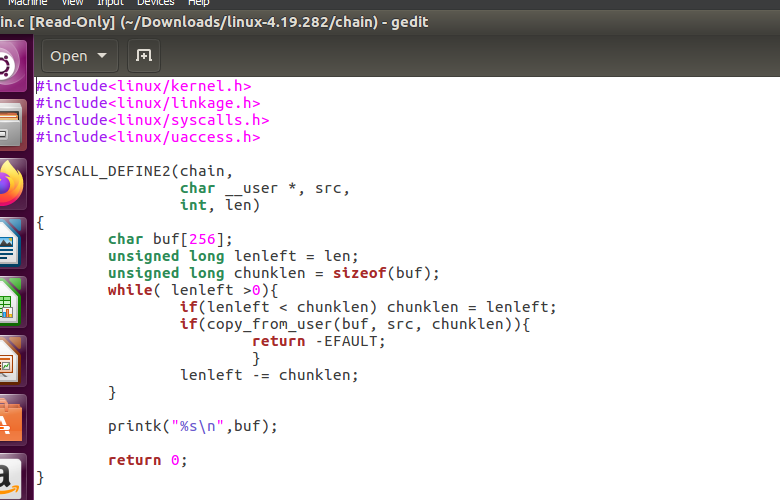
1. Then create the directory name chain using mkdir and in mkdir ccreate a chain.c file and a Makefile.



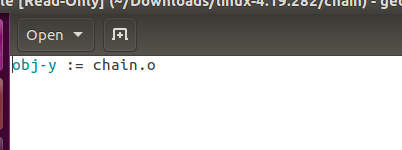
1. Chain and Makefile in chain directory.



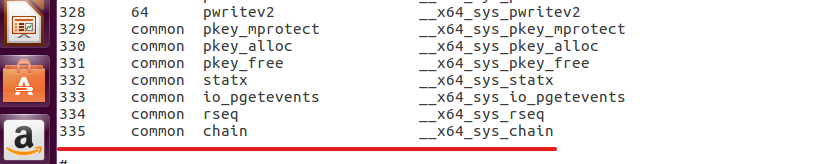
1. Write the system call code in the c file.



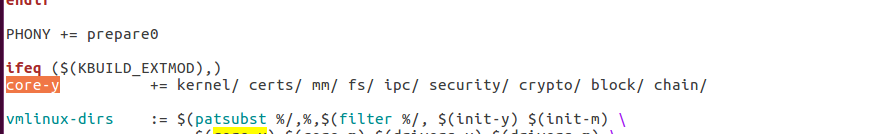
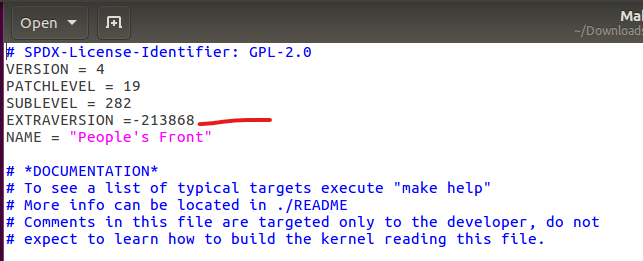
1. In Makefile, write he following and just save and exit.



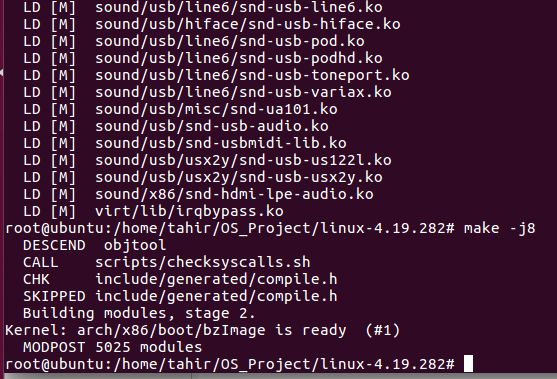
1. Path: gedit arch/x86/entry/syscalls/syscall\_64.tbl, add your config file here.



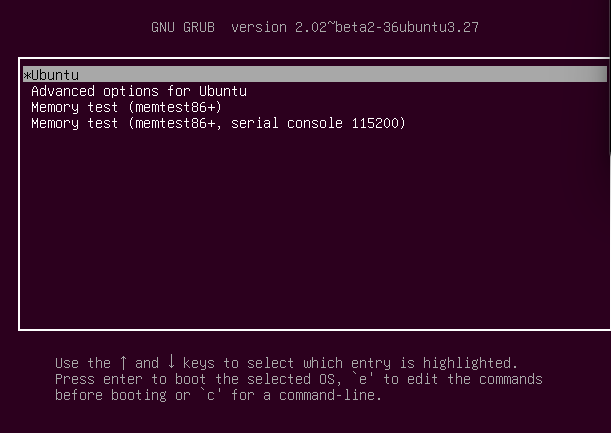
1. In gedit Makefile of linux directory name the kernel your id and and the directory name there after block/ chain/



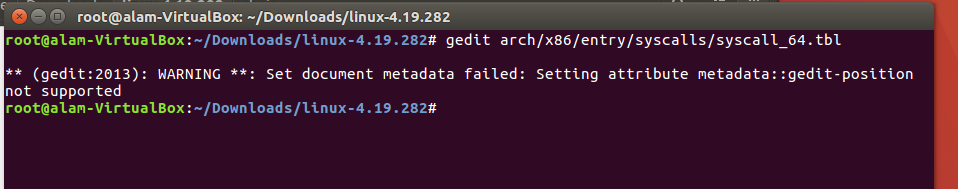
1. Then just run make clean -j4 and make command to compile your kernel.

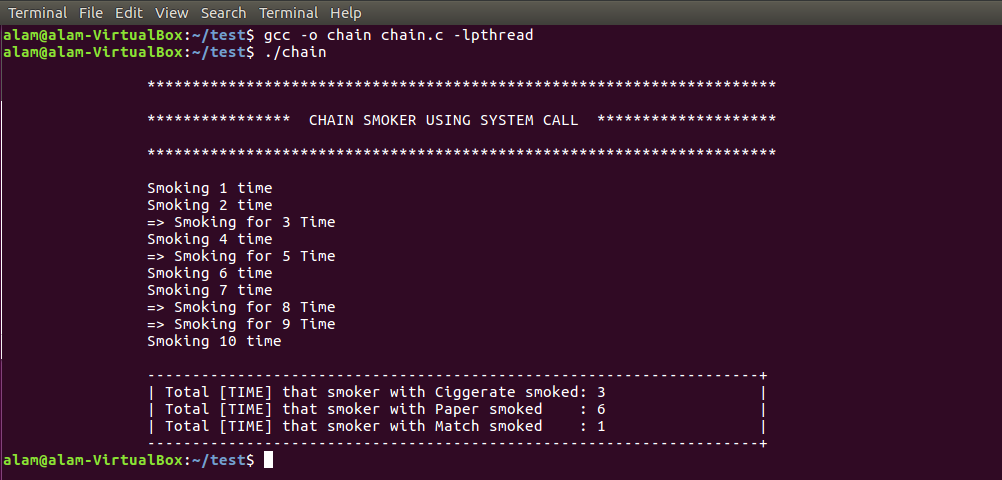


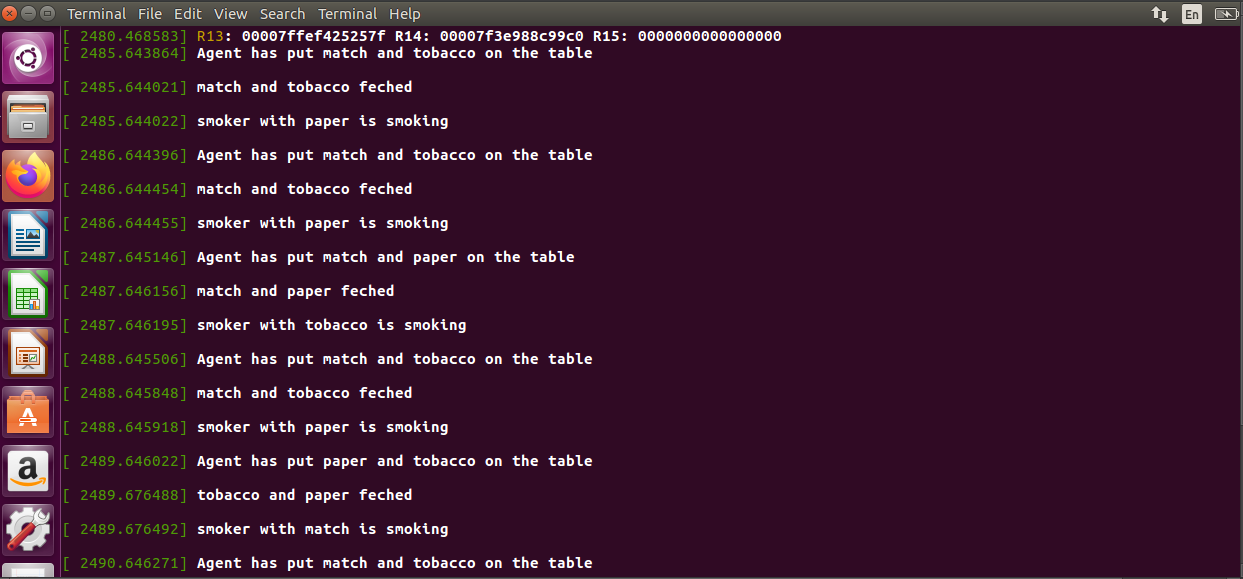
1. After compiling run make modules\_intall install and after that restart your virtual box and select the lastest installed kernel from grub menu.



**EXECUTION STATE:**







**LIMITATIONS AND DEADLOCK HANDLING**

The main argument of the cigarette smoker's difficulty is that there is no way to deal with traditional semaphores as they were at the time in this scenario. Semaphores only offered operations for increasing or decreasing their internal value by one at the time this problem was first put forth. The issue shows that there are instances in which avoiding deadlock is provably impossible if we are restricted to only those operations. Any construction of the smoker threads will likely result in a stalemate once the agent's structure is fixed, regardless of how the threads are built.

The issue with smokers could be applied to more than three threads. In this broader variant, there would be N smokers and the agent would set the table with just N-1 items on the table. If every thread requires two resources (decrementing two semaphores, acquiring two locks, etc.), then a linear ordering will not prevent deadlock. The total number of available resources must be at least the total number of possible requests that can be made There must be N instances accessible for the linear ordering to work to avoid deadlock if there are N threads that can all send out concurrent queries.

**CONCULSION**

In the end, our collective efforts paid off, and we were able to offer a solution to prevent the chain-smoking problem from ever reaching an impasse in the first place. This system call serves as an example of how the operating system prevents deadlock in the numerous processes by being essentially devoid of race conditions.

**REFRENCES**

<https://w3.cs.jmu.edu/kirkpams/OpenCSF/Books/csf/html/CigSmokers.html>

<https://stackoverflow.com/questions/53735886/how-to-pass-parameters-to-linux-system-call>

END OF REPORT