

OPERATING SYSTEM

PROJECT REPORT

**SYSTEM CALL IMPLEMENTATION**

**of**

**CHAIN SMOKER PROBLEM**

**SEC:** BSCS-4E

**TEACHER:** MS. ANAUM

**TEAM MEMBERS:**

Insha Javed Roll# 21k-3279

Muhammed Tahir Roll# 21k-4503

Sabika Shameel Roll# 21k-4606

# INTRODUCTION

This project is dedicated to creating a system call that deals with the chain smoker problem. A system call is a request for a service that is made by the application programs to the operating system; these can be either user system call (without kernel intervention) or kernel system call (with kernel intervention).

**FEATURES**

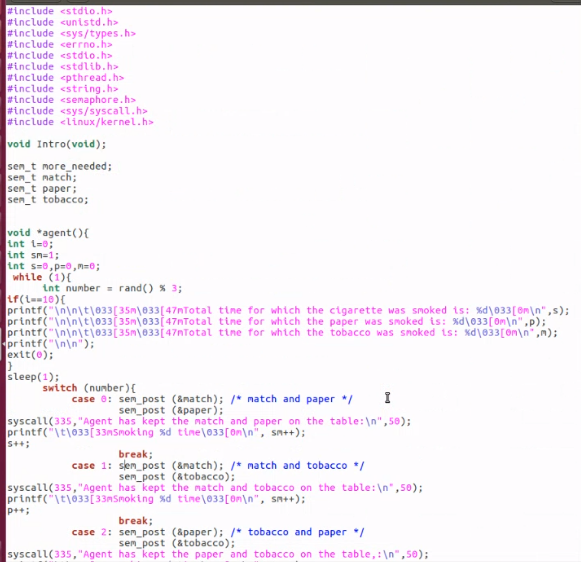
Main function deals with the creation, and deletion of threads, and semaphores. This problem has four processes, three smoker processes, and one agent process. Each of the smoker procedures will create and smoke a cigarette. Tobacco, paper, and matches are needed to produce a cigarette. One of the three 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.

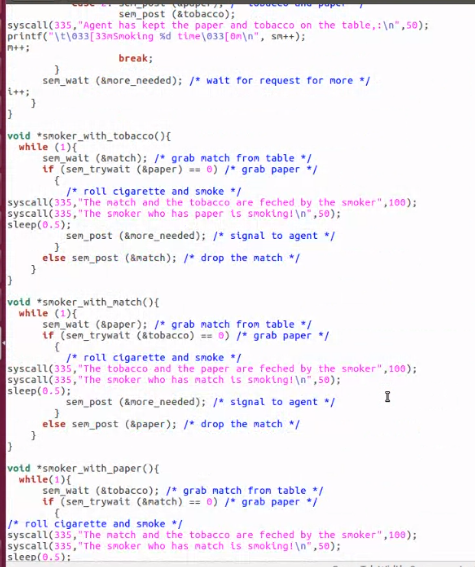
**TECHNOLOGY USED**

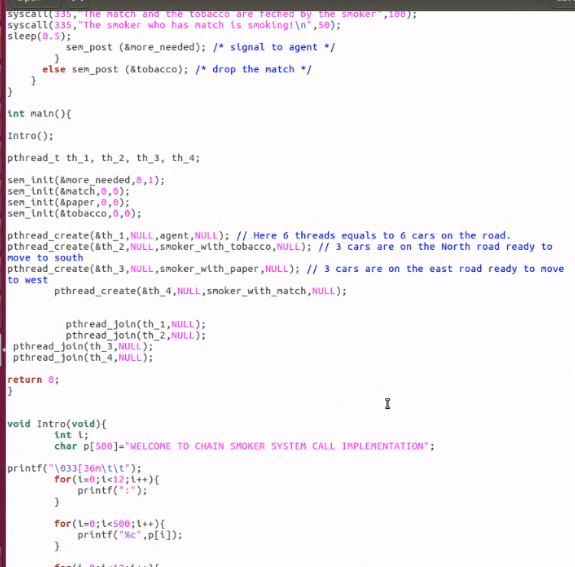
Programming Language: C language

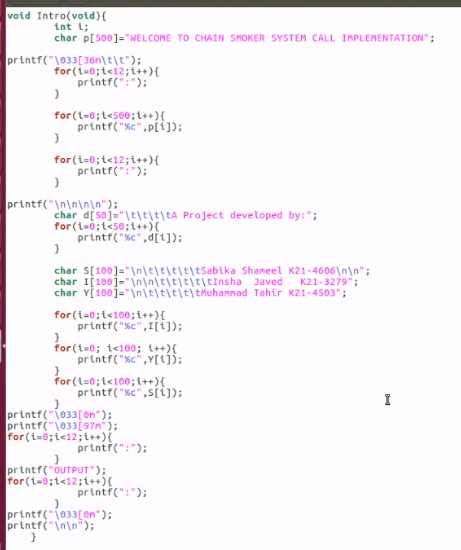
VMware Work Station 17 Player  
Platform: Ubuntu 16.04 LTS

**CODE SNIPPETS**





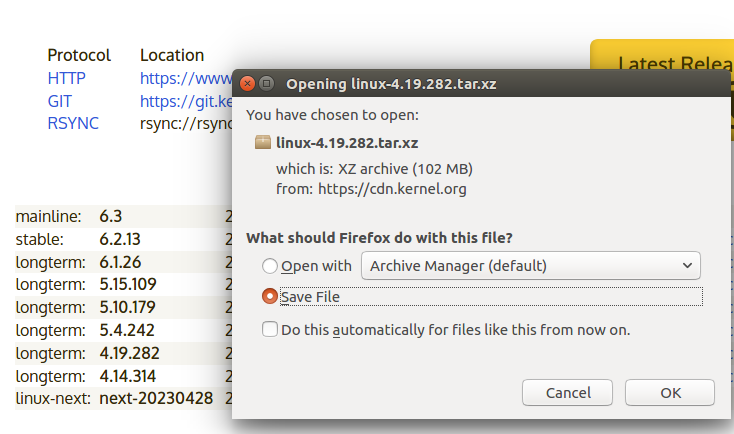




**STEPS FOR KERNEL MODIFICATION**

****

1. Our current kernel is 4.15.0 so we need to upgrade it.

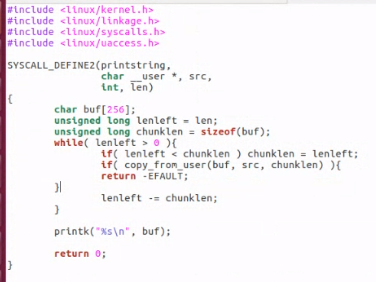


1. We have chosen kernel 4.19.282.

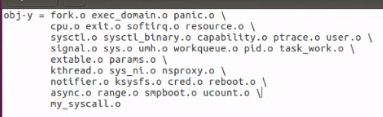


1. **KERNEL LEVEL CODE:**

Making a new folder called hello and Adding a C code for the system call



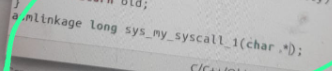
1. Creating a Makefile for the C code and put “obj-y := hello.o”



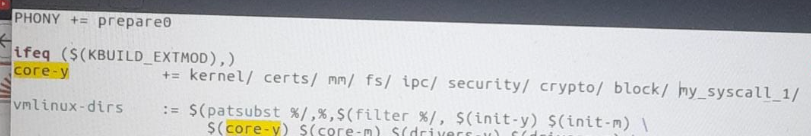
1. Adding the new code into the system table file



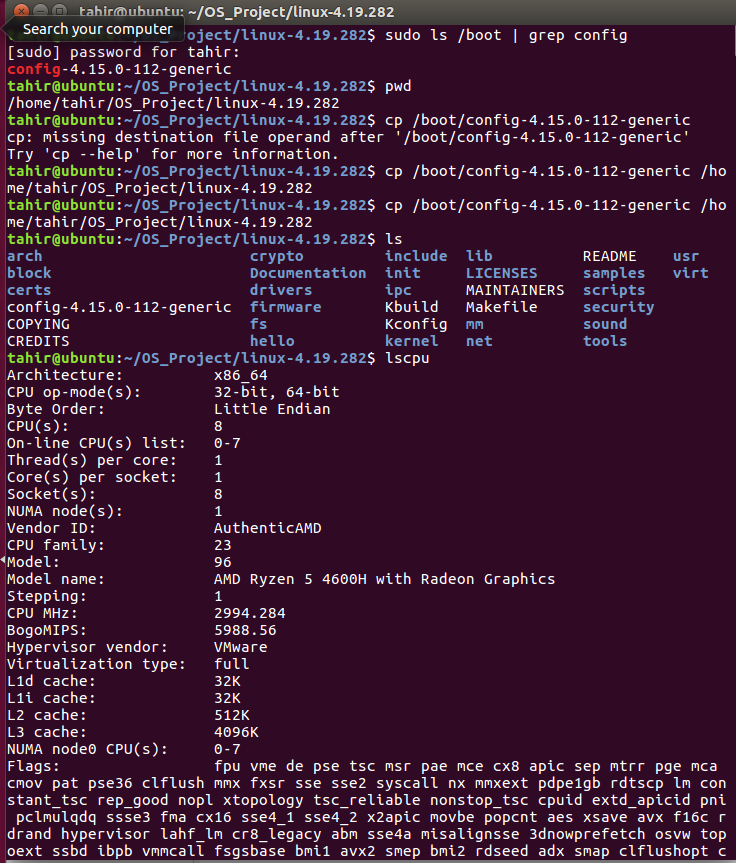
1. Adding the prototype of the new system call into the system calls header file



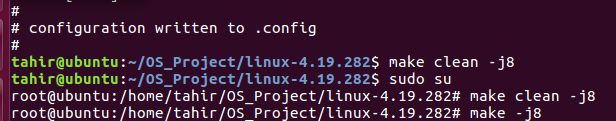
1. Changing version and adding the hello folder in the kernel’s Makefile



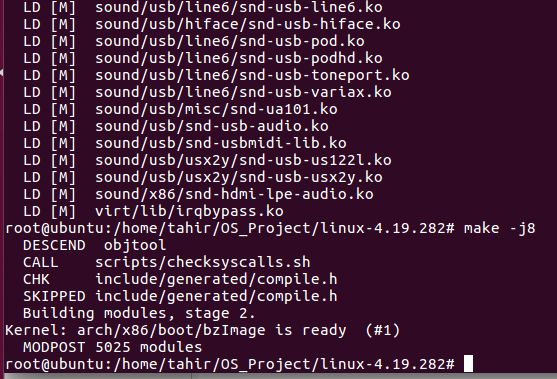
1. Creating a config file



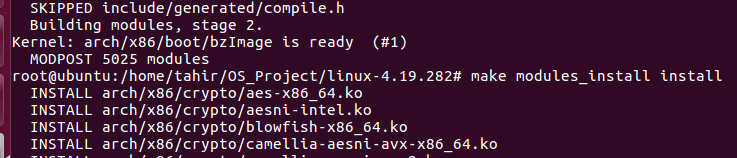
1. Cleaning and Compiling the kernel

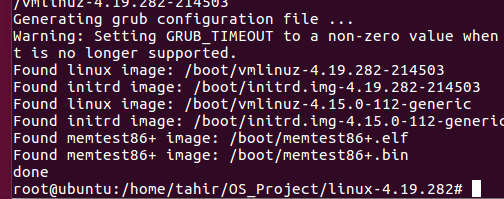


1. Now we have to wait until our Kernel image is built and ready. If we see “Kernel image is ready” when the command is done executing, that means that our kernel image is ready to be installed.

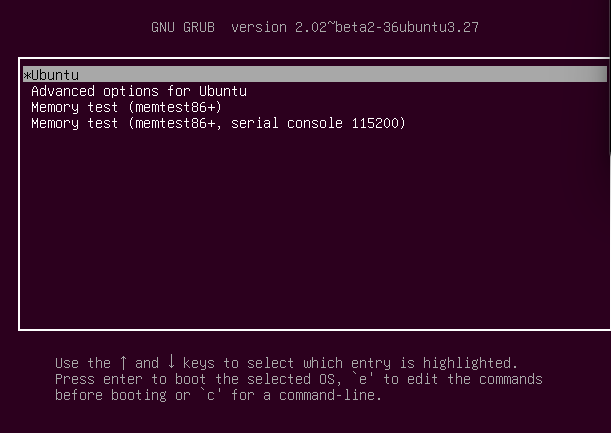


1. Installing modules

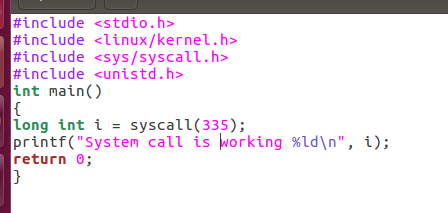




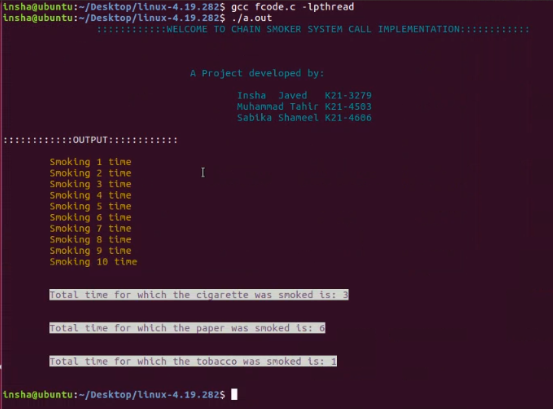
Restarting now;

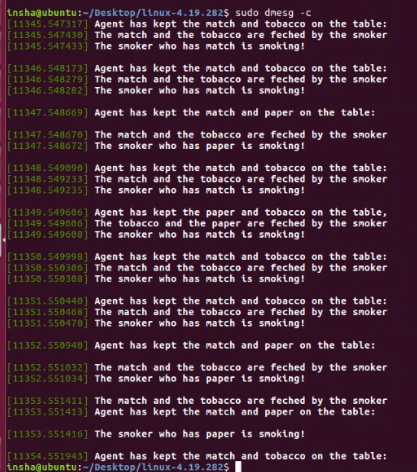


1. Checking if the System call is Working Properly



**EXECUTION STATE:**



****

**LIMITATIONS AND DEADLOCK HANDLING**

The key claim of the cigarette smoker’s problem is that this scenario has no solution for traditional semaphores, as they existed at the time. When this problem was initially proposed, semaphores only provided operations for incrementing or decrementing their internal value by one. The problem proves that, if we are limited to those operations only, there are situations in which avoiding deadlock is provably impossible. Regardless of how the agent and the smoker threads are constructed, once the agent’s structure is fixed, any construction of the smokers will create a possible deadlock situation.

We could generalize the cigarette smoker’s problem to more than three threads. In this generalized form, there would be N smokers and the agent would place only 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. If there are N threads that can all issue concurrent requests, there must be N instances available for the linear ordering to prevent deadlock.

**CONCULSION**

In the end our team efforts paid off and we were able to provide a solution to avoid deadlock in the first place that occurs in the chain smokers problem. This system call is essentially free of race condition, and is a demonstration of how the operating system avoids deadlock in the vast number of processes.

**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>