OS PROJECT REPORT

SYSTEM CALL IMPLEMENTATION OF HOSPITAL SIMULATION USING KTHREADS AND SPIN LOCKS

INSTRUCTOR: MS. ANAUM HAMID
SECTION: BSCS-4E

TEAM MEMBERS:

Fahad Yousuf 21K-4839
Huzaifa Asad 21k-4838
Muhammad Ali 20k-1888

□ INTRODUCTION:

This project is dedicated to creating a system call that deals with the simulation of a hospital. 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:

- Multithreading: The code utilizes kernel threads (kthread_create) to create multiple concurrent threads for doctors and patients. Each doctor and patient runs as a separate thread.

- Resource Synchronization: The code uses spin locks (spin_lock and spin_unlock) to provide mutual exclusion and synchronize access to shared resources such as doctors, patients, and the waiting area.

- Doctor Behavior: Each doctor thread follows a loop where they either treat a waiting patient or rest if no patients are waiting. The treatment and rest times are simulated using msleep to introduce delays.

- Patient Behavior: Each patient thread waits for a doctor to become available. Once a doctor is available, the patient goes through different stages, such as waiting for an operation, being ready for the operation, going to the ultrasound room, returning to the waiting area, having food at the cafe, going to the general practitioner, and completing the visit.

- Thread Management: The code creates and manages the doctor

and patient threads using arrays (doctors and patients) to store the task_struct pointers. The threads are created using kthread_create, and the process is woken up using wake_up_process.

- System Call: The code includes a system call sys_hospital that serves as the entry point for executing the hospital scenario. It initializes the required variables, creates the doctor and patient threads, waits for them to finish, and stops the threads.

TECHNOLOGY USED:

- Programming Language: C language
- VMware Work Station 17 Player
- Platform: Ubuntu 16.04 LTS

Code Snippets for Hospital Simulation:

```c
#include <linux/kernel.h>
#include <linux/kthread.h>
#include <linux/delay.h>

DEFINE_SPINLOCK(mutex);
DEFINE_SPINLOCK(doctor_mutex);
DEFINE_SPINLOCK(patient_mutex);

#define MAX_DOCTORS 10
#define MAX_PATIENTS 20

int num_doctors = 0;
int num_patients = 0;
int num_waiting_patients = 0;

struct patient_info {
    long id;
    const char* status;
};

struct doctor_info {
    int id;
    const char* status;
};

int doctor(void* data)
{
    struct doctor_info* doctor_info = (struct doctor_info*)data;
    spin_lock(&doctor_mutex);
    num_doctors++;
    spin_unlock(&doctor_mutex);

    while (!kthread_should_stop()) {
        // Check if there are waiting patients
        spin_lock(&patient_mutex);
        if (num_waiting_patients > 0) {
            num_waiting_patients--;
            spin_unlock(&patient_mutex);

            // Treat the patient
            printk("Doctor %d is treating a patient.\n", doctor_info->id);
            msleep(2000);  // Simulate treatment time

            spin_lock(&patient_mutex);
            num_patients--;
            spin_unlock(&patient_mutex);
        } else {
            spin_unlock(&patient_mutex);
            // No patients waiting, doctor can rest
```

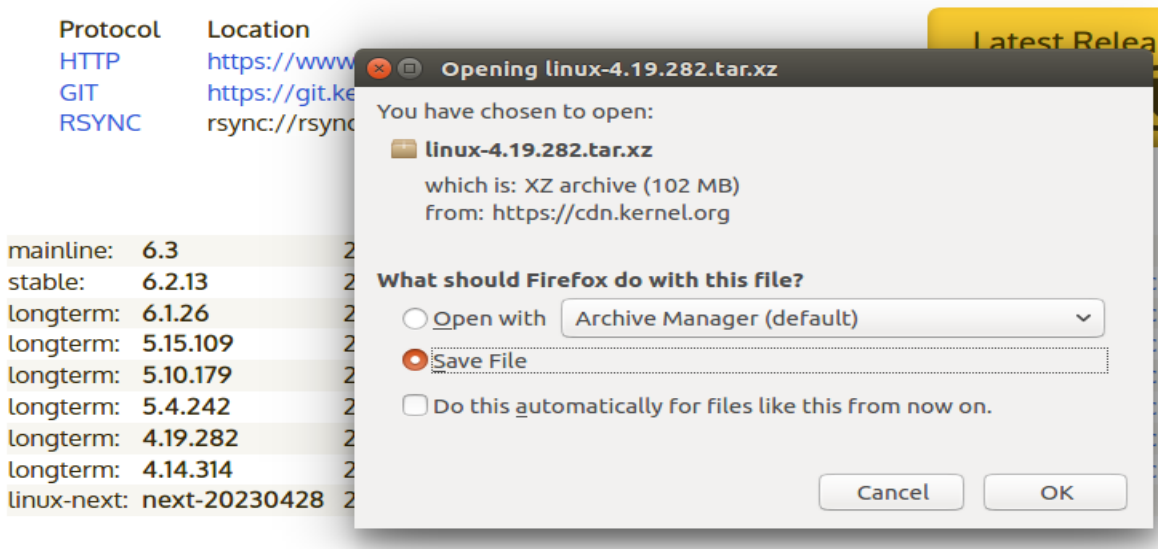
```
            printk("Doctor %d is resting.\n", doctor_info->id);
            msleep(5000);  // Simulate rest time
        }
    }

    spin_lock(&doctor_mutex);
    num_doctors--;
    spin_unlock(&doctor_mutex);

    return 0;
}

int patient(void* data)
{
    struct patient_info* patient_info = (struct patient_info*)data;
    spin_lock(&patient_mutex);
    num_patients++;
    num_waiting_patients++;
    spin_unlock(&patient_mutex);

    // Wait for a doctor
    while (true) {
        spin_lock(&doctor_mutex);
        if (num_doctors > 0) {
            spin_unlock(&doctor_mutex);
            break;
        }
        spin_unlock(&doctor_mutex);
        msleep(1000);  // Wait for a doctor to become available
    }

    // Patient is in the waiting area
    patient_info->status = "waiting for operation";
    printk("Patient %ld is %s.\n", patient_info->id, patient_info->status);
    msleep(1000);

    // Patient is ready for the operation
    patient_info->status = "ready for operation";
    printk("Patient %ld is %s.\n", patient_info->id, patient_info->status);
    msleep(1000);

    // Patient has gone to the ultrasound room
    patient_info->status = "in the ultrasound room";
    printk("Patient %ld is %s.\n", patient_info->id, patient_info->status);
    msleep(2000);

    // Patient is back in the waiting area
    patient_info->status = "waiting for operation";
```

```
    patient_info->status = "waiting for operation";
    printk("Patient %ld is %s.\n", patient_info->id, patient_info->status);
    msleep(1000);

    // Patient is ready for the operation
    patient_info->status = "ready for operation";
    printk("Patient %ld is %s.\n", patient_info->id, patient_info->status);
    msleep(1000);

    // Patient has gone to the ultrasound room
    patient_info->status = "in the ultrasound room";
    printk("Patient %ld is %s.\n", patient_info->id, patient_info->status);
    msleep(2000);

    // Patient is back in the waiting area
    patient_info->status = "waiting for operation";
    printk("Patient %ld is %s.\n", patient_info->id, patient_info->status);
    msleep(1000);

    // Patient is having food at the cafe
    patient_info->status = "having food at the cafe";
    printk("Patient %ld is %s.\n", patient_info->id, patient_info->status);
    msleep(3000);

    // Patient has gone to the general practitioner
    patient_info->status = "with the general practitioner";
    printk("Patient %ld is %s.\n", patient_info->id, patient_info->status);

    msleep(2000);

    // Patient has completed the visit
    patient_info->status = "completed the visit";
    printk("Patient %ld has %s.\n", patient_info->id, patient_info->status);

    spin_lock(&patient_mutex);
    num_patients--;
    spin_unlock(&patient_mutex);

    return 0;
}

asmlinkage long sys_hospital(void)
{
    int i;
    int num_doctors = 3;
    int num_patients = 10;
    struct task_struct* doctors[MAX_DOCTORS];
    struct task_struct* patients[MAX_PATIENTS];
```

```c
asmlinkage long sys_hospital(void)
{
    int i;
    int num_doctors = 3;
    int num_patients = 10;
    struct task_struct* doctors[MAX_DOCTORS];
    struct task_struct* patients[MAX_PATIENTS];
    struct doctor_info doctor_info[MAX_DOCTORS];
    struct patient_info patient_info[MAX_PATIENTS];

    // Create doctor threads
    for (i = 0; i < num_doctors; i++) {
        doctor_info[i].id = i + 1;
        doctor_info[i].status = "resting";
        doctors[i] = kthread_create(doctor, &doctor_info[i], "doctor_thread");
        if (doctors[i]) {
            wake_up_process(doctors[i]);
        } else {
            kthread_stop(doctors[i]);
        }
    }

    // Create patient threads
    for (i = 0; i < num_patients; i++) {
        patient_info[i].id = i + 1;
        patient_info[i].status = "in the waiting area";
        patients[i] = kthread_create(patient, &patient_info[i], "patient_thread");
        if (patients[i]) {
            wake_up_process(patients[i]);
        } else {
            kthread_stop(patients[i]);
        }
    }

    // Wait for all patient threads to finish
    for (i = 0; i < num_patients; i++) {
        kthread_stop(patients[i]);
    }

    // Stop doctor threads
    for (i = 0; i < num_doctors; i++) {
        kthread_stop(doctors[i]);
    }

    return 0;
}
```

STEPS FOR KERNEL MODIFICATION:

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



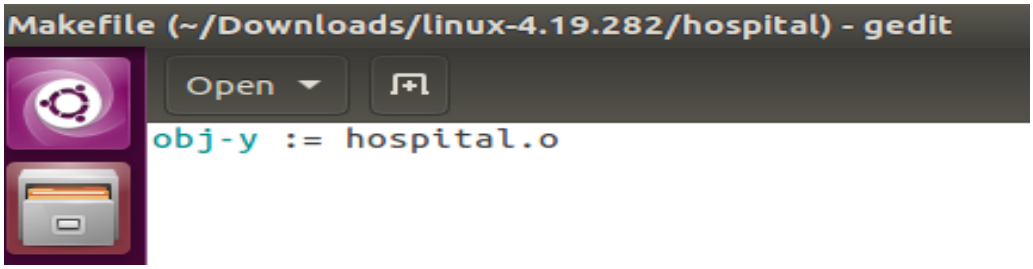
- We have chosen kernel 4.19.282.

Prerequisites:

- sudo apt-get install gcc
- sudo apt-get install libncurses5-dev
- sudo apt-get install bison
- sudo apt-get install flex
- sudo apt install make
- sudo apt-get install libssl-dev
- sudo apt-get install libelf-dev
- sudo add-apt-repository "deb http://archive.ubuntu.com/ubuntu $(lsb_release -sc) main universe"
- sudo apt-get update
- sudo apt-get upgrade

- Creating a Makefile for the C code and put

  ***"obj-y := hospital.o"*.**



- Adding the new code into the system table file.

```
335     64      hospital                sys_hospital
#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation. The __x32_compat_sys stubs are created
# on-the-fly for compat_sys_*() compatibility system calls if X86_X32
# is defined.
#
```

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

```
asmlinkage long sys_hospital(void);

#endif
```

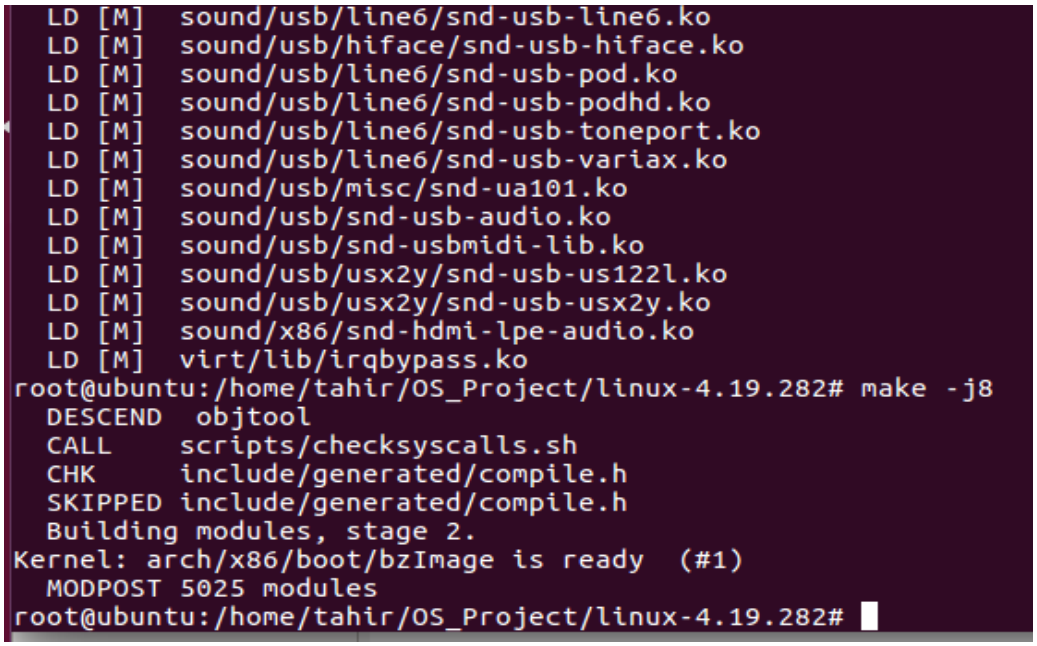
- Changing version and adding the hospital folder in the kernel's Makefile.

```
EXTRAVERSION =-214839
```

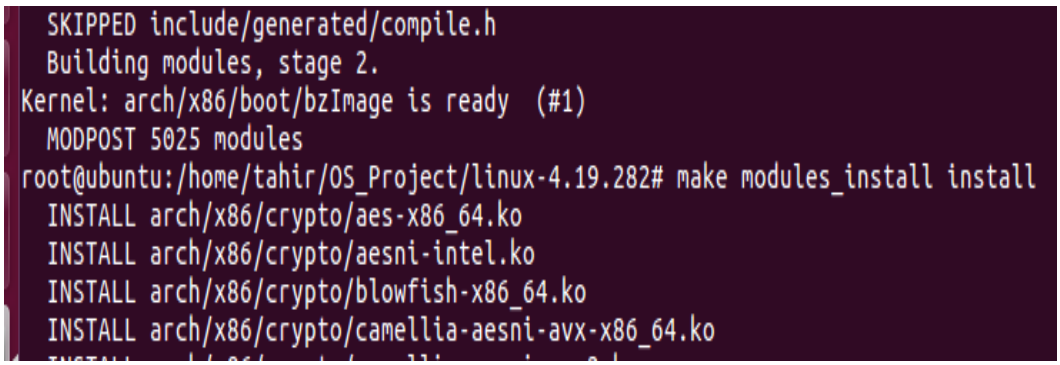
```
core-y          += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ hospital/
```

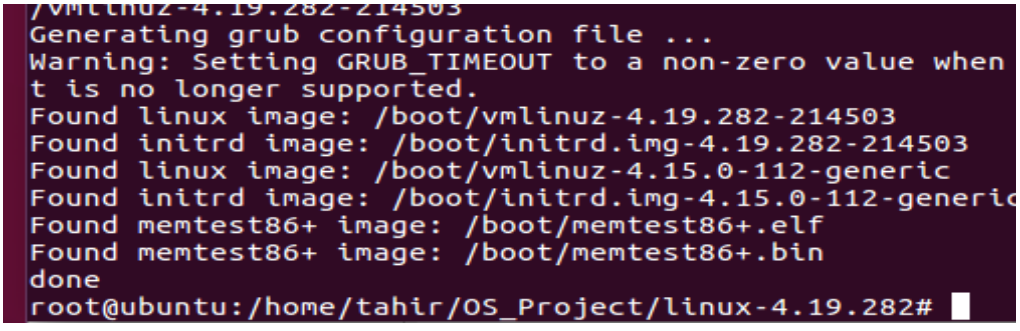
- Creating a config file .
- First typing ***ls /boot | grep config***.
- After that we have to check the present working directory using ***pwd***.
- Now we concatenate the above commands as:
  cp /boot/config-4.15.0-112-generic/home/k214839/Downloads/linux-4.19.282.
- Cleaning and compiling the kernel using ***make clean –j4***.
- Now we build the kernel image using ***make –j4***.

- 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.

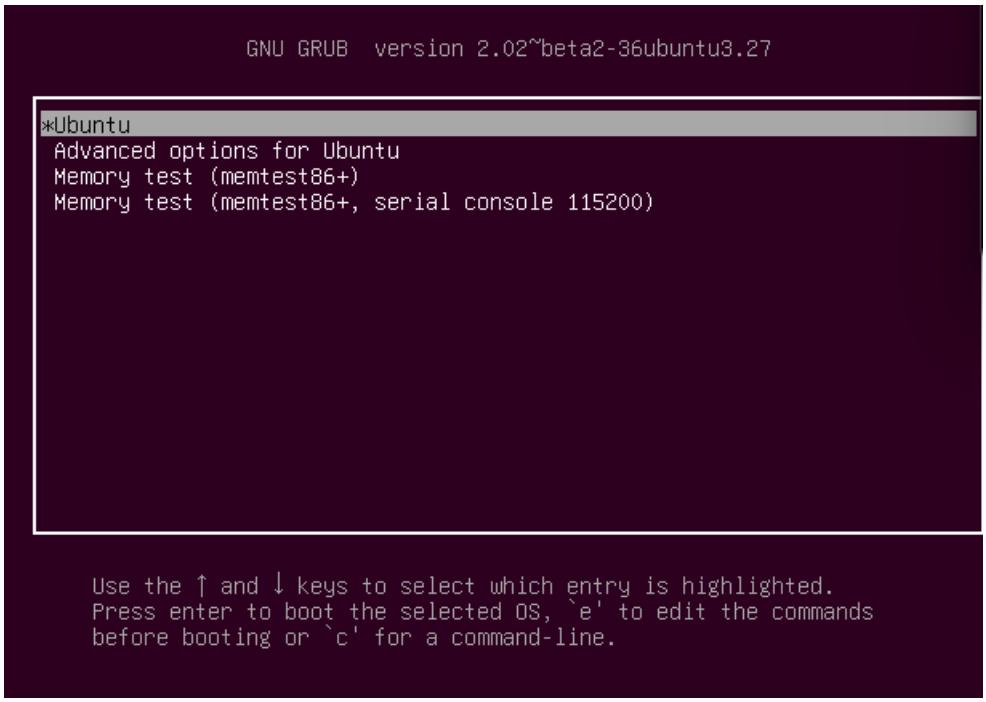


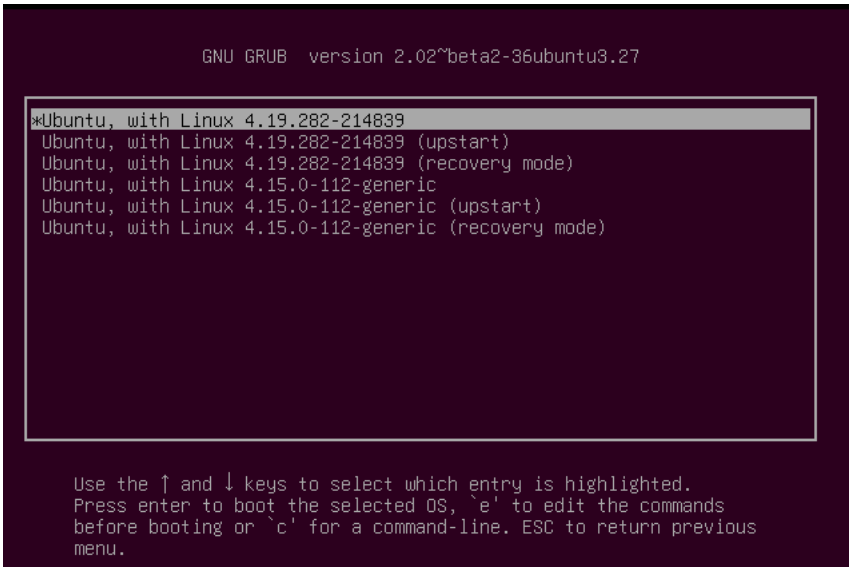
- Installing modules





- Restarting now;





FINALLY EXECUTING THE SMOKER CHAIN CODE:

```c
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>
int main()
{
    long int i = syscall(335);
    printf("System call sys_hospital returned %ld\n", i);
    return 0;
}
```

```
k214839@ubuntu:~$ gedit userspace.c
k214839@ubuntu:~$ gcc userspace.c
k214839@ubuntu:~$ ./a.out
System call sys_hospital returned 0
k214839@ubuntu:~$
```

```
[  151.386432] Doctor 1 is resting.
[  151.386821] Doctor 2 is resting.
[  151.386915] Doctor 3 is resting.
[  151.387032] Patient 1 is waiting for operation.
[  151.387356] Patient 2 is waiting for operation.
[  151.387502] Patient 3 is waiting for operation.
[  151.388173] Patient 4 is waiting for operation.
[  151.388327] Patient 5 is waiting for operation.
[  151.388516] Patient 6 is waiting for operation.
[  151.388685] Patient 7 is waiting for operation.
[  151.388795] Patient 8 is waiting for operation.
[  151.389005] Patient 9 is waiting for operation.
[  151.389136] Patient 10 is waiting for operation.
[  152.414043] Patient 8 is ready for operation.
[  152.414064] Patient 5 is ready for operation.
[  152.414093] Patient 1 is ready for operation.
[  152.414107] Patient 10 is ready for operation.
[  152.414112] Patient 2 is ready for operation.
[  152.414117] Patient 9 is ready for operation.
[  152.414131] Patient 6 is ready for operation.
[  152.414147] Patient 3 is ready for operation.
[  152.414225] Patient 7 is ready for operation.
[  152.414240] Patient 4 is ready for operation.
```

```
[  153.438362] Patient 3 is in the ultrasound room.
[  153.438363] Patient 4 is in the ultrasound room.
[  153.438366] Patient 2 is in the ultrasound room.
[  153.438370] Patient 7 is in the ultrasound room.
[  153.438371] Patient 6 is in the ultrasound room.
[  153.438376] Patient 10 is in the ultrasound room.
[  153.438378] Patient 9 is in the ultrasound room.
[  153.438382] Patient 1 is in the ultrasound room.
[  153.439040] Patient 5 is in the ultrasound room.
[  153.439043] Patient 8 is in the ultrasound room.
[  155.455663] Patient 8 is waiting for operation.
[  155.455679] Patient 5 is waiting for operation.
[  155.455945] Patient 2 is waiting for operation.
[  155.455962] Patient 10 is waiting for operation.
[  155.455971] Patient 1 is waiting for operation.
[  155.455977] Patient 7 is waiting for operation.
[  155.455985] Patient 9 is waiting for operation.
[  155.455990] Patient 4 is waiting for operation.
[  155.456001] Patient 6 is waiting for operation.
[  155.456012] Patient 3 is waiting for operation.
[  156.480663] Patient 5 is having food at the cafe.
[  156.480668] Patient 2 is having food at the cafe.
[  156.480685] Patient 8 is having food at the cafe.
[  156.480720] Patient 3 is having food at the cafe.
```

```
[  159.586943] Patient 4 is with the general practitioner.
[  159.586951] Patient 8 is with the general practitioner.
[  159.586960] Patient 1 is with the general practitioner.
[  159.586968] Patient 5 is with the general practitioner.
[  159.586976] Patient 9 is with the general practitioner.
[  159.586990] Patient 6 is with the general practitioner.
[  159.587006] Patient 3 is with the general practitioner.
[  160.547509] Doctor 2 is treating a patient.
[  160.547511] Doctor 3 is treating a patient.
[  160.547521] Doctor 1 is treating a patient.
[  161.604544] Patient 4 has completed the visit.
[  161.604549] Patient 2 has completed the visit.
[  161.604562] Patient 5 has completed the visit.
[  161.604567] Patient 3 has completed the visit.
[  161.604741] Patient 7 has completed the visit.
[  161.604751] Patient 8 has completed the visit.
[  161.604760] Patient 6 has completed the visit.
[  161.604831] Patient 9 has completed the visit.
[  161.604902] Patient 1 has completed the visit.
[  161.604937] Patient 10 has completed the visit.
[  162.564967] Doctor 3 is treating a patient.
[  162.565161] Doctor 2 is resting.
[  164.582382] Doctor 3 is resting.
k214839@ubuntu:~$
```

LIMITATIONS:

The code assumes a fixed number of doctors (MAX_DOCTORS) and patients (MAX_PATIENTS). This may limit the scalability of the system, as it cannot easily handle dynamic additions or removals of doctors or patients.

CONCLUSION

- In the end our team efforts paid off and we were able to provide a solution to avoid the race condition in the first place that occurs in a typical hospital environment. 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.

REFERENCES

https://github.com/iddemirjs/deu-hospital-simulation
https://stackoverflow.com/questions/25871960/linux-kernel-threads-with-spinlock-freeze
https://www.youtube.com/watch?v=qcsKGWshRZE&t=707s