LDD Lab Exam Questions

Note: **Time: 3 hrs**

Attempt any 2 programs. Mandatory to attempt 1st program and attempt any one form 2nd and 3rd program. Once done upload programs on GIT with output snapshots.

1. **Kernel Synchronization Mechanisms for Multiple devices.**
   - **a)** Write a character driver for Multiple devices and register 2 device numbers in the kernel space and implements Read and Write Functionality in the kernel space.
   - **b)** Maintain a global Kernel buffer of 50 bytes which will be used by the Read and Write methods of the driver. Use a semaphore to protect the buffer which represents a Critical section.
   - **c)** Also, use wait-queues to avoid consecutive write operations on the 2 devices that are created to prevent the overwrite of the previous data.
   - **d)** Write 2 separate user space programs that operate on the 2 devices, individually. The write function in the user application should be followed by sleep and then a read. However, the write should put 80 bytes of data, while the read should read it from the kernel.
   - **e)** Since the write function is trying to write 80 bytes, the write function should be made to sleep using wait-queues until read is completed. The read function should Wake-up the sleeping write function to copy remaining data.

2. **Verify User Read/Write operations using IOCTL functionality.**
   - **a)** Write a Simple character driver with Open , Read ,Write , Close and Ioctl Functionality.
   - **b)** Initialize Structure in Kernel space which records kernel buffer data , size of the buffer and Recent activity(Read = 0 /Write = 1).

```
struct stats
        {
                int size;
                char buff[];
                int r_w;
                        };
```

   - **c)** Write 3 separate user programs implementing Read function, Write function and Ioctl function individually, in 3 User applications.
   - **d)** Check the status of driver using Ioctl program with command GETSTATS, which should return structure (struct stats) from kernel space.
   - **e)** User should run write application first with any string and check stats using Ioctl program and then run Read application and check stats using Ioctl program and verify the string and recent activity(r_w).

3. **LED Operation Using GPIOs and Kernel Timers.**
   - **a)** Write a character driver using ioctl functionality, kernel timer.
   - **b)** Use GPIO_23 to interface led on raspberry pi and blink the led periodically using kernel timer.
   - **c)** Use ioctl functionality to change the periodicity of led blinking using a command UPDATE_PERIODICITY
   - **d)** Use another ioctl command GET_PERIODICITY to read the timer periodicity from the kernel and update the user application.
   - **e)** The IOCTL function in the user space should be implemented in a loop, taking periodicity from the user and retrieving the updated periodicity from the kernel.