# **Team: Naveen Chandra Pinnika (B00869700)**

**Sai Krishna Varakala (B00929496)**

# Assignment 3: Design Document

# Chatroom using a character device

**Problem:** Enable two or more processes to chat with each other using a character device (say /dev/chatroom).

Character devices support byte stream abstraction, they send and receive data in the form of bytes. No distinct message boundary. Ex: serial ports, keyboard, mouse

* In this we need to develop a kernel module and this kernel module should create a Character Device called /dev/chatroom.
* When we insert this character device into the kernel, users from the userspace can access this character device. This user opens this character device using an open system call and it’ll return the file descriptor to the user space program.
* When we call open, the kernel module calls kfifo\_alloc and it creates a kfifo queue in the kernel. We shouldn’t create this before the userspace program actually opens the character device.
* Then we create two threads in the user space program, one is used for read operation and other is used for write operation. This user space program basically does the read and write operations.
* For each user present in the user space we create a kfifo queue in the kernel. We need to write while loops for read and write operations.
* When one user writes into the chatroom, then from the pipe it should write into kfifo queues of other users and read from kfifo itself.
* This Character Device should implement open(), close(), read() and write() functions when inserted into the kernel.
* To create character devices we use structure. Ex: struct miscdevice dev{} we define the name of the character device, file operations in it and we also define structure for file operations to redirect the system calls. Ex: static const struct file\_operations myfops={ .write=kwrite, .read=kread, .open=kopen, .release=kclose}; In this left hand side is of function declaration and right hand side is of user defined function.
* We need to register the character device into the kernel once we’re done using it then we need to unregister it.
* We need to create a P number of FIFO queues in the kernel. These FIFO are just like pipes running inside the kernel.
* These pipes have got read end and write end and they need to have a message boundary in these inorder not to read the entire message at a time. We use some kind of message boundary in this.
* kfifo interface has several functions like kfifo\_alloc, kfifo\_in\_spinlocked, kfifo\_out\_spinlocked
* In this FIFO queue we have an init method, which is called kfifo\_alloc. For this we need to provide three parameters like pointer for the function, size of FIFO and run area.
* In this init method we use spin\_lock\_init and provide the spinlock variable that we have defined.
* We also define a character buffer to store the data. We use kfifo\_in\_spinlocked to put the data into the FIFO queue. For this we provide four parameters: the lock, the buffer we want to put in, the size of FIFO and the lock.
* kfifo\_skip accepts one parameter: the buffer. It directly drops elements present in the buffer.