**CPE 435 Operating System Lab**

**Lab 5: Message Queue**

There are many applications in which processes need to cooperate with each other. This is always the case; a variety of forms of inter-process communication can be used under **LINUX**. These support resource sharing, synchronization, connectionless and connection-oriented data exchange or combinations of these.

Message queue are a refined method of inter process communication. As an application, we can implement a printer spooler using message queues. Some form of message passing between processes is now part of many modern operating systems. All messages are stored in the kernel, and have an associated message queue identifier. It is this identifier, which we call an **msqid** that identifies a particular queue of messages. Processes read and write messages to arbitrary queues.

Every message on a queue has the following attributes:

* Long integer type;
* Length of the data portion of the message (can be zero);
* Data (if the length is grater than zero).

For every message queue in the system, the kernel maintains the following structure of information:

|  |
| --- |
| # include <sys/types.h>  # include <sys/ipc.h> /\*defines the ipc\_perm structure \*/  stuct msqid\_ds  {  stuct ipc\_perm msg\_perm; /\* opertion permission struct \*/  struct msg \*msg\_first; **/\*** ptr to first message on q \*/  struct msg \*msg \_last; **/\*** ptr to last message on q \*/  ushort msg\_cbytes; /\* current # bytes on q \*/  ushort msg\_qnum; /\* current # of messages on q \*/  ushort msg\_qbytes /\* max # of bytes allowed on q \*/  ushort msg\_lspid; /\* pid of last msgsnd \*/  ushort msg\_lrpid; /\* pid of last msgrcv \*/  time\_t msg\_stime; /\* time of last msgsnd \*/  time\_t msg\_rtime; /\* time of last msgrcv \***/** time\_t msg\_ctime; **/\*** time of last msgct1  (that changed the above) \*/  } |

* **Write two processes (A & B)** .Process A should create a Message Queue. Initially it should wait for the data to be received from the Process B, through the message queue. On reception, it can send data (accepted from user ) to the Process B, process B should open the message queue created by the Process A. then accept the data from the user and send it to the Process A through the queue. It should then wait for the Process A to respond, procedure can continue until the Process A Send “Exit” to the Process B.( Both the programs should be in an infinite loop).
* Hint :

int msgget (key\_t key, int msgflag); // Functions for Creating and Handling Message Queues

int msgsnd (int msqid, struct msgbuf \*ptr, int length, int flag); //function to send a message

int msgrcv (int msqid, struct msgbuf \* ptr, int length ,long msgtype, int flag); //function to Receive Message

int msgctl(int msqid, int cmd, struct msqid\_ds \*buff); // provides a variety of control operations on a message queue.