***Inter Process Communication System***

There are two type of process:

* *Independent Process*
* *Co-operating Process*

***Independent processes*** are those process which are not affected by the execution of other processes if runs simultaneously.

**Co-operating** processes are those process which are affected by the execution of the other process if runs simultaneously.

***Inter Process Communication System*** is such a type of mechanism which allow the process to communicate with each other and synchronize their action. The process can communicate with each other using following two ways:

* *Shared memory.*
* *Message passing.*

***Message Passing***

In message passing method process communicate with each other without using any kind of shared memory. If two process want to communicate with each other they follow the following steps

* *Establish a communication link.*
* *Start exchanging messages by using the basic primitives.*
* *Minimum two primitives are needed:*
* *send (message, destination) or send(message).*
* *receive (message, host) or receive(message).*

A message queue is a linked list of messages stored within the kernel and identified by a message queue identifier. A new queue is created or an existing queue opened by msgget().

New messages are added to the end of a queue by msgsnd(). Every message has a positive long integer type field, a non-negative length, and the actual data bytes (corresponding to the length), all of which are specified to msgsnd() when the message is added to a queue. Messages are fetched from a queue by msgrcv(). We don’t have to fetch the messages in a first-in, first-out order. Instead, we can fetch messages based on their type field.

All processes can exchange information through access to a common system message queue. The sending process places a message (via some (OS) message-passing module) onto a queue which can be read by another process. Each message is given an identification or type so that processes can select the appropriate message. Process must share a common key in order to gain access to the queue in the first place.

***Code (For writer perspective)***

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/msg.h>

struct mesg\_buffer

{

    long mesg\_type;

    char mesg\_text[100];

} message;

int main()

{

    key\_t key;

    int msgid;

    key = ftok("progfile", 65);

    msgid = msgget(key, 0666 | IPC\_CREAT);

    message.mesg\_type = 1;

    printf("Write Data : ");

    gets(message.mesg\_text);

    msgsnd(msgid, &message, sizeof(message), 0);

    printf("Data send is : %s \n", message.mesg\_text);

    return 0;

}

***Code (For Reader Perspective)***

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/msg.h>

struct mesg\_buffer

{

    long mesg\_type;

    char mesg\_text[100];

} message;

int main()

{

    key\_t key;

    int msgid;

    key = ftok("progfile", 65);

    msgid = msgget(key, 0666 | IPC\_CREAT);

    msgrcv(msgid, &message, sizeof(message), 1, 0);

    printf("Data Received is : %s \n",message.mesg\_text);

    msgctl(msgid, IPC\_RMID, NULL);

    return 0;

}