OS LAB :IPC : MESSAGE QUEUE

April 13, 2020

OBJECTIVE: TO IMPLEMENT IPC WITH THE HELP OF MESSAGE QUEUE CONCEPT

1. Write a C program to perform the same:

So the theoretical aspect is really simple, what is basic concept of a message queue?

- Where we can write the data or the message, data will be paced in a buffer, if anyone wants to read the message can come and read this. Therefore simply we can say that from one end the message can be written or pushed into the queue and from the other end the data can be read. Please refer to the theory session material.
- So we should have two processes: one is sender and another is receiver.
- The receiver cannot receiver until the sender sends and on the hand the sender cannot send until the receiver is not ready to receive.
- So before sending the data the sender needs to inform the receiver that it is going to send a data of type A, B OR C. For example in my below mentioned example, I have already mentioned that I am sending the mtyepe=1.
- msgsnd – will initialize the queue
  msgrcv – will help to receive the message
  msgctl – will be for administrative dealings , will manage the queue.

Sender process:

```
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/types.h>
#include <stdio.h>
#include <string.h>
// Structure of the message
struct msgAB
{
        long mtype;//this signals like an identification of from which process to which process.
        char msgtxt[150]; // the actual message
};

int main()
{
        struct msgAB msg;
```

```
int msgid;
key_t key; //msgsnd manual has this description. Please go and read.
////////////////// part 1 of the programs//////////
if (_key = ftok("msend.c",'b')) == -1)
// this is a file to key system call will generate the key,there should be an exixting file,it can be a .c file or a .txt file.so we are
//trying to create a key for a particular file msend.c,the same file name should be used at the receiver end .
{
        perror("key"); // if key is not generated perror wull let us know that why it is not created
        exit(1);
}
////////////////// part 2 of the programs: Generation of msg id  //////////
if ((msgid = msgget(key,0644 | IPC_CREAT))== -1)
{
        perror(""msgid");
        exit (1);
}
 printf("\n the msgid is %d", msgid);
 printf("enter the text");
msg.mtype=1;
while (gets(msg.msgtxt), !feof(stdin))
{
////////////////// part 3 of the programs: sending of msg   //////////
        if (msgsnd(msgid, &msg.sizeof(msg),0) == -1)
//sending the msg,as we are not using any flag so the 4th parameter is 0,if any error in send then it will return -1.
        {
                perror("msgsnd");
                exit(1);
        }
}
```

```
////////////////// part 4 of the programs: to delete the msgid when work is done //////////////

        if(msgctl(msgid,IPC_RMID,NULL))== -1)
        {
                perror ("msgctl");
               exit(1);
        }
        return 0;
}
```

Receiver process:

```
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/types.h>
#include <stdio.h>
#include <string.h>
// Structure of the message
Struct msgAB
{
        long mtype;//this signals like an identification of from which process to which process.
        char msgtxt[150]; // the actual message
};

int main()
{
        Struct msgAB msg;
        int msgid;
        key_t key; //msgsnd manual has this description. Please go and read.
        ////////////////// part 1 of the programs//////////
        if (_key = ftok("myfile.c",'b')) == -1)
```

```
        {
        perror("key"); // if key is not generated perror wull let us know that why it is not created
        exit(1);
    }
    // Till this part is same as the sender process, key is used to validate
    ////////////////// part 2 of the programs: need to check that msgid is generated again properly or not ////////////
    If ((msgid = msgget (key,0644)) == -1)
    // if failure will return -1
    {
        perror(""msgid");
        exit (1);
    }
    ////////////////// part 3 of the programs: receive of the message  ////////////
    for( ; ; );
    {
        // the user can keep on typing till 200,as this is the limit
        if (msgrcv(msgid, &msg, sizeof(msg),1,0) == -1)
        {
            perror(""msgrcv");
            exit (1);
        }
        printf("%s\n " , msg, msgtxt);
        // this is where the msg will get printed
    }
    return 0;
}
```

So, this is all about your sending message and receiving the message. Use different terminals. run the receiver first.