Lab 9

IMPLEMENT IPC MECHANISM USING MESSAGE PASSING

OBJECTIVES:

To implement message passing to read and write data to the message queue without being connected to each other in Linux using c

BACKGROUND THEORY

Inter-process communication (IPC) is a mechanism that allows processes to communicate with each other and synchronize their actions. The communication between these processes can be seen as a method of co-operation between them. Processes can communicate with each other through: In this IPC model, the processes communicate with each other by exchanging messages. For this purpose, a communication link must exist between the processes and it must facilitate at least two operations send (message) and receive (message). Size of messages may be variable or fixed.

Source code

Message Send

// C Program for Message Queue (Writer Process)

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MAX 10
  // structure for message queue
struct mesg_buffer {
    long mesg_type;
```

```
    char mesg_text[100];
} message;

int main()
{
    key_t key;
    int msgid;

    // ftok to generate unique key
    key = ftok("progfile", 65);

    // msgget creates a message queue
    // and returns identifier
    msgid = msgget(key, 0666 | IPC_CREAT);
    message.mesg_type = 1;

    printf("Write Data : ");
    fgets(message.mesg_text,MAX,stdin);

    // msgsnd to send message
    msgsnd(msgid, &message, sizeof(message), 0);

    // display the message
    printf("Data send is : %s \n", message.mesg_text);
```

This variable is typically used to store the unique key value that's generated using the ftok function

65 : Proj_id value used for IPC

0666: octal permission value for read/write

Max: max no of character in msg, stdin: standard msg inupt, keyboard

0 represent: message will be added to end of the queue, funcn will wait if queue is full

```
    return 0;
}
```

Message Receive

```
// C Program for Message Queue (Reader Process)
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>

// structure for message queue
struct mesg_buffer {
    long mesg_type;
    char mesg_text[100];
} message;

int main()
{
    key_t key;
    int msgid;

    // ftok to generate unique key
    key = ftok("progfile", 65);
```

```
    // msgget creates a message queue
    // and returns identifier
    msgid = msgget(key, 0666 | IPC_CREAT);


    // msgrcv to receive message
    msgrcv(msgid, &message, sizeof(message), 1, 0);

    // display the message
    printf("Data Received is : %s \n",
                    message.mesg_text);


    // to destroy the message queue
    msgctl(msgid, IPC_RMID, NULL);

    return 0;
}
```

1 represent type of msg that we want to receive
0 represent: message will be added to end of the queue, funcn will wait if queue is full

IPC_RMID indicates that you want to remove (delete) the message queue associated with the provided msgid

NULL, used to pass additional information or get information about the message queue. Since you're only deleting the message queue and not requiring any additional information, you can pass NULL

OUTPUT: Screenshots of the output

DISCUSSION:

CONCLUSION: