EXPERIMENT - 6.

AIM :- Implement IPC using message queue

Sender.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define BUF_SIZE 30
    struct smsg
    {
      long int msg_type;
      char data [BUF_SIZE];
    } send, receiver;
       int main()
    {
      int msgid = msgget (1234, 0666|IPC_CREAT);
      if (msgid == -1)
    {
    fprintf (stderr," Cannot create");
     return EXIT_FAILURE;
    }
    fprintf(stdout," Enter some data to be
                          sent to the receiver");
    fgets(send.data, BUF_SIZE, stdin);
     send.msg_type = 1;
     if (msgsnd (msgid, (void *) &send,
                    BUF_SIZE, 0) == -1)
```

```
{
fprintf (stderr, "Error writing to message
                        queue \n");
return EXIT_FAILURE;
}
  sleep (2);
  if (msgrcv (msgid, (void*) &receive, BUF_SIZE+2,
                                    == -1)
  {
  fprintf (stderr, "Error reading data from
                        MSG Q \n");
  }
    fprintf (stdout, "Reciver got : %s \n",
                        receive.data);
    msgctl (msgid, IPC_RMID, 0);
    return EXIT_SUCCESS;
  }
```

receiver.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/msg.h>
#define BUF_SIZE 30
  struct smsg
  {
   long int msg_type;
   char data [BUF_SIZE];
  } receive, send;
```

```
int main()
{
 int msgid = msgget (1234, 0666 | IPC_CREAT);
 if (msgid == -1)
 {
  fprintf (stderr, "Cannot create");
  return EXIT_FAILURE;
 }
 if (msgrcv (msgid, (void*) &receive, BUF_SIZE, 0)
                    == -1)
 {
  fprintf (stderr, "Error reading data from
                    msg Q \n");
 }
  fprintf (stdout, "Receiver got %s \n",
                    receive.data);
  fprint (stdout, " Enter some data to
                    be sent to the sender");
  fgets (send.data, BUF_SIZE, stdin);
  send.msg_type = 2;
  if (msgsnd (msgid (void*) &send,
                    BUF_SIZE, 0) == -1)
  {
   fprintf (stderr, " Error writing to
                    message queue");
   return EXIT_FAILURE;
  }
  sleep (2);
  msgctl (msgid, IPC_RMID, 0);
  return EXIT_SUCCESS;
}
```

Algorithm

Sender

1. Define a structure with an integer variable and a character array.
2. Create a message queue using msgget()
3. Set message type as 1.
4. Accept the message.
5. Send the message using msgsnd()
6. Receive the reply message using msgrcv()
7. Print the message.
8. Remove the message id using msgctl()
9. Stop

Receiver

1. Define a structure with an integer variable and a character array.
2. Create a message queue using msgget()
3. Set message type as 1
4. Receive the message using msgrcv()
5. Print the message
6. Set message type as 1
7. Send back a message using msgsnd()
8. Remove the message id using msgctl()
9. Stop.

Output

Sender

Enter some data to be sent to the receiver. Hello are you there?.
Receiver got: Yes, I am here.

Receiver

Receiver got: Hello are you there?.
Enter some data to be sent to the sender: Yes, I am here.