Practical-5

Aim:- Implement TWO-WAY Inter-process communication using Message Queues. Consider TWO independent processes for communication.
This communication must continue till a specific key is pressed or a STOP message is sent by any one of the processes.

Theory:-
Message queues are a type of inter-process communication (IPC) mechanism used in Unix-like operating systems to allow processes to exchange data in the form of messages. Message queues provide a way for processes to communicate asynchronously, and they are often used in scenarios where processes need to send and receive structured data.
A message queue is essentially a linear list of messages. Each message has a type identifier (mtype) and a message body (the actual data).
Create or access a message queue using ‘msgget’, Send a message to the queue using ‘msgsnd’, Receive a message from the queue using ‘msgrcv’ and remove the message queue using ‘msgctl’ when it’s no longer needed.

Code:-
Sender.c
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct msg_buffer {
   long msg_type;
   char msg_text[100];
};

int main() {
   key_t key;
   struct msg_buffer message;
   int msgid;

   // Generate a unique key
   key = ftok("keyfile", 65);
```

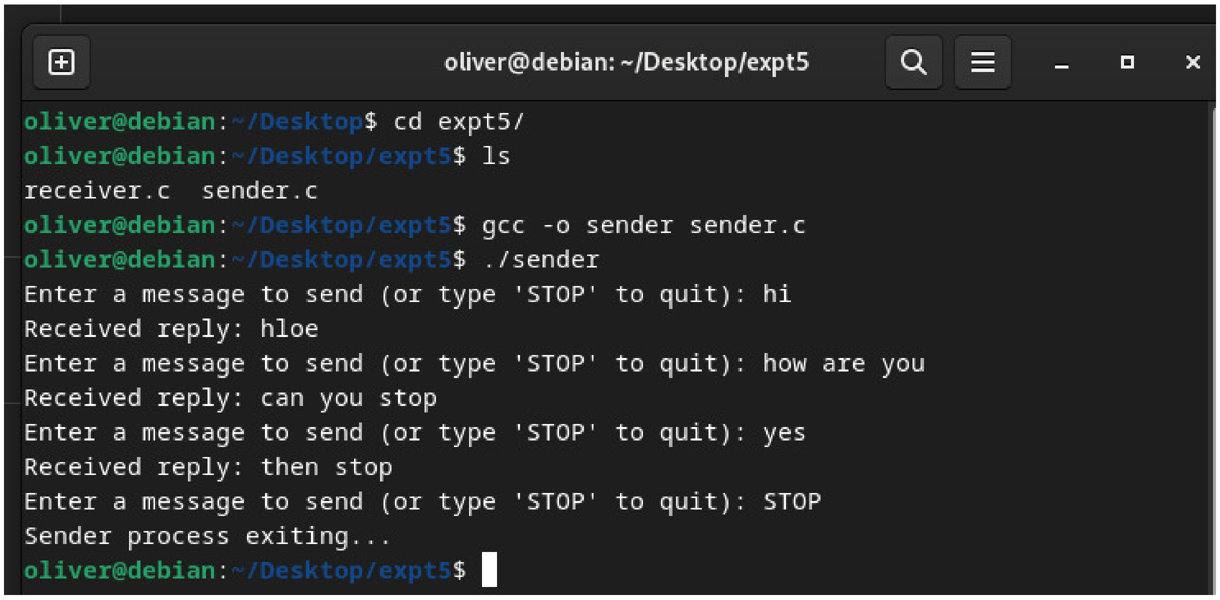
```
    // Create a message queue
    msgid = msgget(key, 0666 | IPC_CREAT);

    if (msgid == -1) {
        perror("msgget");
        exit(1);
    }

    while (1) {
        printf("Enter a message to send (or type 'STOP' to quit): ");
        fgets(message.msg_text, sizeof(message.msg_text), stdin);

        message.msg_type = 1;

        // Send the message to the queue
        msgsnd(msgid, &message, sizeof(message), 0);

        if (strncmp(message.msg_text, "STOP", 4) == 0) {
            printf("Sender process exiting...\n");
            break;
        }

        // Receive replies from the receiver
        msgrcv(msgid, &message, sizeof(message), 2, 0);
        printf("Received reply: %s", message.msg_text);

        if (strncmp(message.msg_text, "STOP", 4) == 0) {
            printf("Sender process exiting...\n");
            break;
        }
    }

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

    return 0;
}
```

Receiver.c

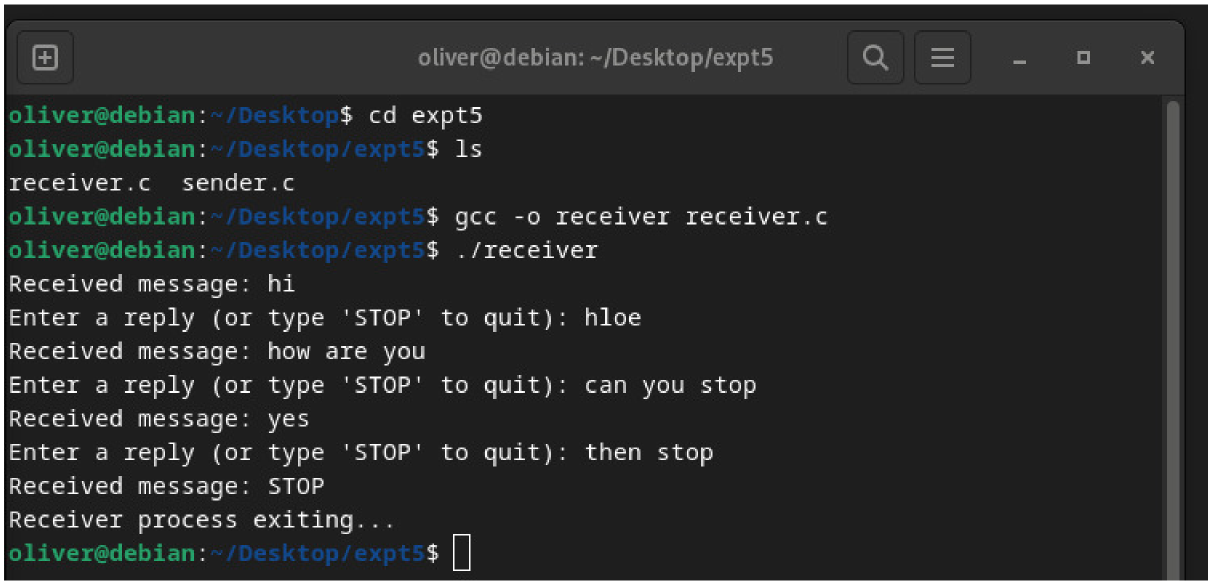
```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct msg_buffer {
    long msg_type;
    char msg_text[100];
};

int main() {
    key_t key;
    struct msg_buffer message;
    int msgid;

    // Generate the same unique key as the sender
    key = ftok("keyfile", 65);

    // Access the message queue
    msgid = msgget(key, 0666 | IPC_CREAT);

    if (msgid == -1) {
        perror("msgget");
        exit(1);
    }

    while (1) {
        // Receive messages of type 1
        msgrcv(msgid, &message, sizeof(message), 1, 0);

        printf("Received message: %s", message.msg_text);

        if (strncmp(message.msg_text, "STOP", 4) == 0) {
            printf("Receiver process exiting...\n");
            break;
        }

        // Process and prepare a reply message
        printf("Enter a reply (or type 'STOP' to quit): ");
        fgets(message.msg_text, sizeof(message.msg_text), stdin);
        message.msg_type = 2;

        // Send the reply message back to the sender
        msgsnd(msgid, &message, sizeof(message), 0);

        if (strncmp(message.msg_text, "STOP", 4) == 0) {
            printf("Receiver process exiting...\n");
            break;
        }
    }

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

    return 0;
}
```

Output:-



```
oliver@debian:~/Desktop$ cd expt5/
oliver@debian:~/Desktop/expt5$ ls
receiver.c  sender.c
oliver@debian:~/Desktop/expt5$ gcc -o sender sender.c
oliver@debian:~/Desktop/expt5$ ./sender
Enter a message to send (or type 'STOP' to quit): hi
Received reply: hloe
Enter a message to send (or type 'STOP' to quit): how are you
Received reply: can you stop
Enter a message to send (or type 'STOP' to quit): yes
Received reply: then stop
Enter a message to send (or type 'STOP' to quit): STOP
Sender process exiting...
oliver@debian:~/Desktop/expt5$
```


oliver@debian: ~/Desktop/expt5

```
oliver@debian:~/Desktop$ cd expt5
oliver@debian:~/Desktop/expt5$ ls
receiver.c  sender.c
oliver@debian:~/Desktop/expt5$ gcc -o receiver receiver.c
oliver@debian:~/Desktop/expt5$ ./receiver
Received message: hi
Enter a reply (or type 'STOP' to quit): hloe
Received message: how are you
Enter a reply (or type 'STOP' to quit): can you stop
Received message: yes
Enter a reply (or type 'STOP' to quit): then stop
Received message: STOP
Receiver process exiting...
oliver@debian:~/Desktop/expt5$ 
```