PROGRAM – 8

AIM: Communication between independent processes using Message Queues.
Write two prorams msgq_server.c and msgq_client.c.
msgq_server.c : - creates a message queue – reads from queue for an integer. - converts integer to binary format and writes it to the queue.
msgq_client.c : -attaches itself to the queue . -prompts user to enter an integer and writes it to the queue. -reads the binary equivalent format sent from the server on the queue.

INTRODUCTION: 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. In the following C program we have implemented the concept of message queues for the purpose of communication between a server program and a client program.

C PROGRAM :
Server source code (os8s.c) :

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct msg_buff{
        long type;
        long int msg;
}msgout,msgin;
```

```
long int inttobin(long int num){
	if (num == 0){
		return 0;
	}else{
		return (num%2 + 10*inttobin(num/2));
	}
}
int main(){
	key_t key;
	int msgid;

	key = ftok("os8",1);

	msgid = msgget(key,0660 | IPC_CREAT) ;

	printf("\nServer running....\n");

	while(msgrcv(msgid,&msgin,sizeof(msgin),1,0) == -1);

	msgout.msg = inttobin(msgin.msg);
	msgout.type = 2 ;

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

	printf("\nTask completed...Shutting down Server....bye!\n");

	return 0;
}
```

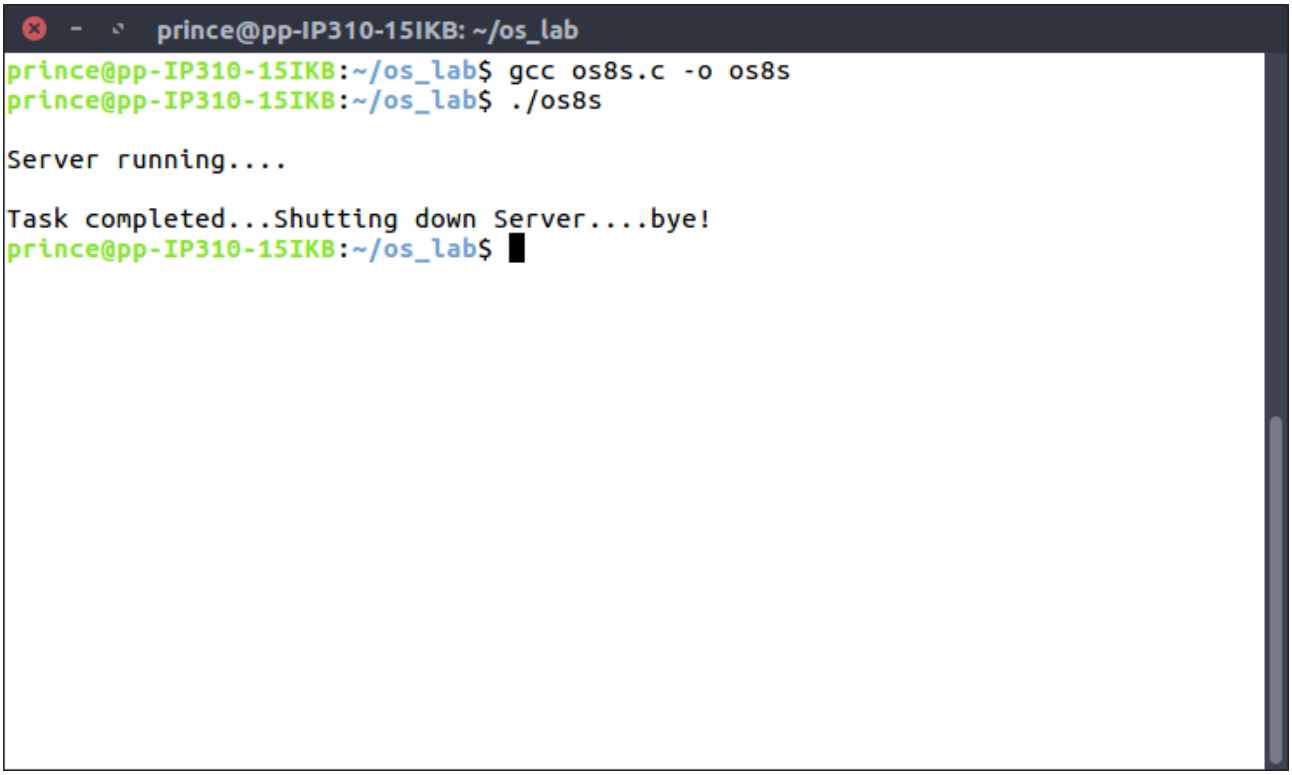
Client source code (os8c.c):

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct msg_buff{
	long type;
	long int msg;
}msgout,msgin;
```

```
int main(){
        key_t key;
        int msgid;
        msgout.type = 1;
        printf("\nEnter an integer no. : ");
        scanf("%ld",&msgout.msg);

        key = ftok("os8",1);

        msgid = msgget(key,0660 | IPC_CREAT) ;
        msgsnd(msgid,&msgout ,sizeof(msgout),0);

        while(msgrcv(msgid , &msgin , sizeof(msgin),2,0) == -1);

        printf("\nBinary equivalent of integer entered : %ld\n\n",msgin.msg);

        msgctl(msgid,IPC_RMID,NULL);
        return 0;
}
```

OUTPUT:

Server Program :



Client Program:

LEARNING OUTCOMES: We learnt about the concept of Message Queues and how they can be used to achieve Inter Process Communication. We also learnt about the several system calls that we make while making use of message queue.