VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY
Permanently Affiliated to JNTU Kakinada, Approved by AICTE
Accredited by NAAC with 'A' Grade, ISO 9001:2008 Certified
Nambur, Pedakakani (M), Guntur (Dt) - 522508
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
B.Tech Program is Accredited by NBA

CERTIFICATE

Name of the Lab : OPERATING SYSTEMS_
Name of the Student : Thota Nagababu
Student Regd. No. : 18BQ1A05K3
CLASS : III B.TECH. I SEM CSE – D
GithubLink : https://github.com/nagababuthota984/5K3-OS-LAB

INDEX

S. No.	Name of the Experiment	LAST DATE	PAGE NO
5D	INTER PROCESS COMMUNICATION USING MESSAGE QUEUES	JAN 12	3-6

AIM:

INTER PROCESS COMMUNICATION USING MESSAGE QUEUES

DESCRIPTION:

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.

LIBRARIES USED:

```
#include<string.h>
#include<sys/msg.h>
```

CODE:

SENDER:

```
#include<string.h>
#include<sys/msg.h>
int main()
{
int msqid=32769;
struct message{
long type;
char text[20];
}msg;
msg.type=1;
strcpy(msg.text,"this is message1");
msgsnd(msqid,(void
*)&msg,sizeof(msg.text),IPC_NOWAIT);
strcpy(msg.text,"this is message2");
msgsnd(msqid,(void
*)&msg,sizeof(msg.text),IPC_NOWAIT);
return 0;
}
```

RECEIVER:

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

int main()
{
```

```
int msqid=32769;
struct message{
long type;
char text[20];
}msg;
msg.type=0;
msgrcv(msqid,(void*)
&msg,sizeof(msg.text),msg.type,MSG_NOERROR|
IPC_NOWAIT);
printf("%s \n ",msg.text);
return 0;

}
```

OUTPUT:

Step1: run sender_ program first

Step2: run receiver

Step3: It prints first message

Step4: It prints second message

Like this if there are ‘n’ messages in sender.

Receiver can be run ‘n’ no .of times

OUTPUT SCREEN SHOTS:

```
File  Edit  View  Search  Terminal  Help
3-cse-d@Lab-04-24:~$ cd 18BQ1A05K3/UNIXLAB/Experiment\ 5/d
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB/Experiment 5/d$ gcc msq_sender.c
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB/Experiment 5/d$ ./a.out
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB/Experiment 5/d$ gcc msq_reciever.c
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB/Experiment 5/d$ ./a.out
this is message 1
 3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB/Experiment 5/d$ gcc msq_reciever.c
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB/Experiment 5/d$ ./a.out
this is message 2
```