**EXPERIMENT 7**

**OBJECTIVE**:-

Communication between independent processes using Message Queues.

Write two programs msgq\_server.c and msgq\_client.c.

msgq\_server:

- Creates a message queue

- Reads message from queue for an integer.

- Converts integer to binary format and writes it to the queue.

msgq\_client:

- Attaches itself to the queue

- Prompts user to enter an integer and writes it to the queue.

- Reads the binary format sent from the server on the queue.

**CODE**:-

//------------------------------------------------------------------------------

//

// Desc: msgq\_server.c

//

// Date: 29/3/2018

//

// Author: VU DUY DU - 2k16/CO/364

//

//------------------------------------------------------------------------------

#include<stdio.h>

#include<string.h>

#include<sys/ipc.h>//this header file is for the key\_t and ftok() function

#include<sys/msg.h>//for msgget(), msgsnd(), msgrcv(),etc.

#define CLIENT\_NUMBER 144

#define SERVER\_NUMBER 127

typedef struct msg\_buffer{

long msg\_type;

char msg\_text[100];//100 characters only =((

//you can give many more things here..

} msg\_buffer;

//ultility functions for processing message//////

void reverse(char\*s, int n){

int begin, end = n - 1;

char r[100];

for (begin = 0; begin < n; begin++) {

r[begin] = s[end];

end--;

}

r[begin] = '\0';

strcpy(s,r);

}

void to\_binary(char\*s){

int i = 0;

int number = 0;

while(s[i]!='\0'){

int n = s[i] - '0';

number\*= 10;

number += n;

i++;

}

i = 0;

while(number!=0){

int bin = number%2;

number/=2;

s[i++] = bin+'0';

}

s[i] = '\0';

reverse(s,i);

}

///////////////////////////////////////////////////

int main(){

printf("Hello Message Queue from Msgq\_server\n");

//create a unique key (key\_t) - i think key can be any number

//but by using ftok() we generate a unique key for safer programming.

//after this line is executed, from now on, the key is fixed for this

//pattern of pathname+proj\_id. any other register on this pattern is

//returned same key. the path name to the file must actually exist.

//the key is depending on the current content of the file.

key\_t key = ftok("msgq\_client.c",101);

if(key == -1){

printf("Error: key is not created, check the pathname or proj\_id\n");

return 1;//don't run, it's not safe.

}

printf("I have a key: %d\n",key);

//create a message queue.

//there are 2 possible flags IPC\_CREAT|IPC\_EXCL, but what the hell is 0666???

//again the id returned is an unpredicted number

int msgid = msgget(key,0666|IPC\_CREAT);

if(msgid == -1){

printf("Error: Message Queue is not created, check the provided key\n");

return 1;//don't run, nothing to run

}

printf("Message Queue is created, id: %d\n",msgid);

msg\_buffer message;

//receive message which is an integer number

long type = CLIENT\_NUMBER;//the programmer VU, he told me that, this is the number that

//will send me the message I need.

printf("Waiting for a message of the type: %ld\n", type);

int n = msgrcv(msgid,&message,sizeof(msg\_buffer)-sizeof(long),type,0);

if(n==-1){

printf("Error: Something wrong with receiving message\n");

return 1;

}

printf("Got a message (%ld - %s) (size: %d)\n",message.msg\_type, message.msg\_text,n);;

//proccess message

to\_binary(message.msg\_text);

message.msg\_type = SERVER\_NUMBER;

printf("Message processed! (%ld - %s)\n",message.msg\_type, message.msg\_text);

//send message

printf("Message patched (%ld-%s), ready to send.\n",message.msg\_type, message.msg\_text);

if(msgsnd(msgid,&message,sizeof(msg\_buffer)-sizeof(long),0)==-1){

printf("Error: Something wrong with sending message\n");

return 1;

}

printf("Sent!\n");

return 0;

}

//------------------------------------------------------------------------------

//

// Desc: msgq\_client.c

//

// Date: 29/3/2018

//

// Author: VU DUY DU - 2k16/CO/364

//

//------------------------------------------------------------------------------

#include<stdio.h>

#include<sys/ipc.h>//this header file is for the key\_t and ftok() function

#include<sys/msg.h>//for msgget(), msgsnd(), msgrcv(),etc.

#define CLIENT\_NUMBER 144

#define SERVER\_NUMBER 127

typedef struct msg\_buffer{

long msg\_type;

char msg\_text[100];//100 characters only =((

//you can give many more things here..

} msg\_buffer;

int main(){

printf("Hello Message Queue from Msgq\_server\n");

//create a unique key (key\_t) - i think key can be any number

//but by using ftok() we generate a unique key for safer programming.

//after this line is executed, from now on, the key is fixed for this

//pattern of pathname+proj\_id. any other register on this pattern is

//returned same key. the path name to the file must actually exist.

//the key is depending on the current content of the file.

key\_t key = ftok("msgq\_client.c",101);

if(key == -1){

printf("Error: key is not created, check the pathname or proj\_id\n");

return 1;//don't run, it's not safe.

}

printf("I have a key: %d\n",key);

//create a message queue.

//there are 2 possible flags IPC\_CREAT|IPC\_EXCL, but what the hell is 0666???

//again the id returned is an unpredicted number

int msgid = msgget(key,0666|IPC\_CREAT);

if(msgid == -1){

printf("Error: Message Queue is not created, check the provided key\n");

return 1;//don't run, nothing to run

}

printf("Message Queue is created, id: %d\n",msgid);

//initiate message

msg\_buffer message;

//any number babe. who wants to read this message, use this number

message.msg\_type = CLIENT\_NUMBER;

printf("Enter a integer: ");

int number;

scanf("%d",&number);

sprintf(message.msg\_text,"%d",number);

//send message

printf("Message patched (%ld-%s), ready to send.\n",message.msg\_type, message.msg\_text);

if(msgsnd(msgid,&message,sizeof(msg\_buffer)-sizeof(long),0)==-1){

printf("Error: Something wrong with sending message\n");

return 1;

}

printf("Sent!\n");

//receive message which is in binary format

long type = SERVER\_NUMBER;//the programmer VU, he told me that, this is the number that

//will send me the message I need.

printf("Waiting for a message of the type: %ld\n", type);

int n = msgrcv(msgid,&message,sizeof(msg\_buffer)-sizeof(long),type,0);

if(n==-1){

printf("Error: Something wrong with receiving message\n");

return 1;

}

printf("Got a message (%ld - %s) (size: %d)\n",message.msg\_type, message.msg\_text,n);;

//delete message queue, after this

//what if the msgq is not deleted???

if(msgctl(msgid, IPC\_RMID, NULL)==-1){

printf("Error: Cannot delete the message queue, check msgid\n");

return 1;

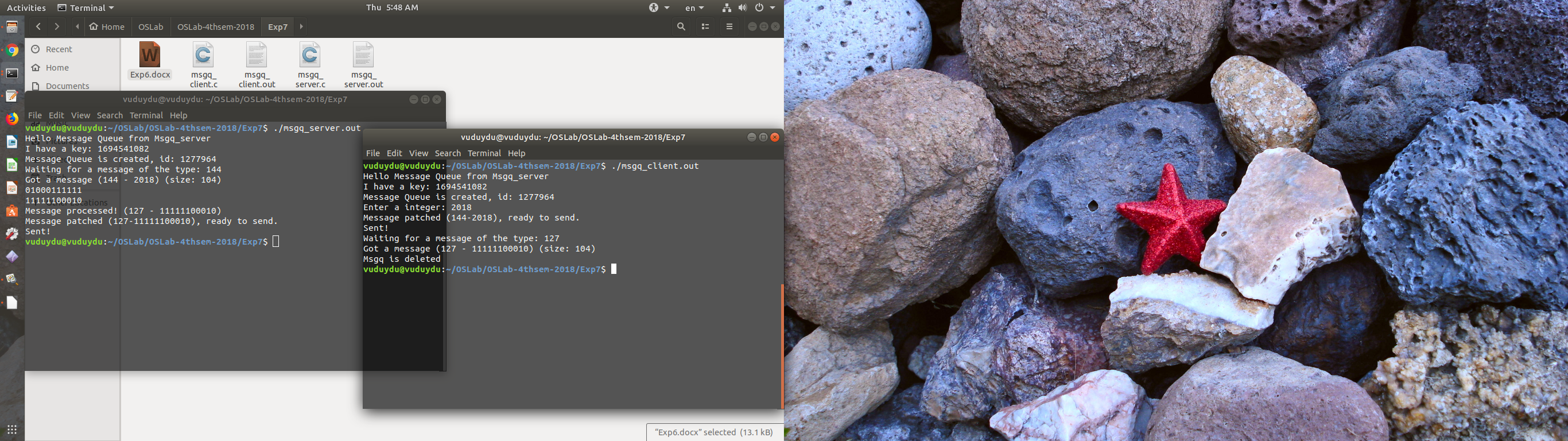
}

printf("Msgq is deleted \n");

return 0;

}

**OUTPUT:-**



**DISCUSSION:-**

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.

*Source code of this experiment can be found here:*

*https://github.com/Sieunguoimay/OSLab-4thsem-2018/tree/master/Exp7*