**Name:Rohan Nivas Chavan**

**PRN:2019BTEIT00084**

**UOS LAB**

**7.1** Write a program to perform IPC using message and send did u get this? and then reply.

**Theory:**

Msgget() -

The msgget() system call returns the System V message queue identifier associated with the value of the key argument. A new message queue is created if key has the value IPC\_PRIVATE or key isn't IPC\_PRIVATE, no message queue with the given key key exists, and IPC\_CREAT is specified in msgflg. If msgflg specifies both IPC\_CREAT and IPC\_EXCL and a message queue already exists for key, then msgget() fails with errno set to EEXIST. (This is analogous to the effect of the combination O\_CREAT | O\_EXCL for open(2).) Upon creation, the least significant bits of the argument msgflg define the permissions of the message queue.

Msgrcv() -

The msgrcv() function reads a message from the queue associated with the message queue identifier specified by msqid and places it in the user-defined buffer pointed to by msgp.

The argument msgp points to a user-defined buffer that must contain first a field of type long int that will specify the type of the message, and then a data portion that will hold the data bytes of the message. The structure below is an example of what this user-defined buffer might look like:

struct mymsg { long int mtype; /\* message type \*/ char mtext[1]; /\* message text \*/}

The structure member mtype is the received message's type as specified by the sending process.

The structure member mtext is the text of the message.

Msgsnd() -

The msgsnd() and msgrcv() system calls are used, respectively, to send messages to, and receive messages from, a message queue. The calling process must have write permission on the message queue in order to send a message, and read permission to receive a message.

The msgp argument is a pointer to caller-defined structure of the following general form:

164

struct msgbuf { long mtype; /\* message type, must be > 0 \*/ char mtext[1]; /\* message data \*/};

The mtext field is an array (or other structure) whose size is specified by msgsz, a nonnegative integer value. Messages of zero length (i.e., no mtext field) are permitted. The mtype field must have a strictly positive integer value. This value can be used by the receiving process for message selection (see the description of msgrcv() below).

Data/Variables dictionary for sender:

S r. N o .

NAME OF VARIABLE/FUNCTION/ STRUCT

TYPE USE WHERE THE VARIABLE IS CALLED

1 Msqid int FOR SOCKET TUPLE

In if statement for the result of msgget method

2 Msgflg int

FOR SEMAPHORE

As a parameter to msgflg 3 Key key\_t Semaphore id Initialising it with some constant value

4 Sbuf struct msgbuf For composing message 5 buf\_length size\_t As a parameter to msgsnd function

165

Data/Variables dictionary for Receiver:

Sr. No. NAME OF VARIABLE/FUNCT ION/STRUCT

TYPE USE WHERE THE VARIABLE IS CALLED

1 msqid int FOR SOCKET TUPLE

In if statement for the result of msgget method 3 Key key\_t Semaphore id Initialising it with some constant value

4 rbuf struct msgbuf To store the message

For composing message

5 buf\_length size\_t Length of message to be sent

As a parameter to msgrcv function

Algorithm / Flowchart :

Algorithm : send(msqid, msgflg, key, sbuf, buf\_length)

1. Initialise key with 1234

2. Assign result of msgget(key, msgflg) to msqid

3. If msqid is less than 0

a. Print error

b. Exit

4. End if

5. Set message type to in sbuf.mtype

6. Copy string “did you get this?” to sbuf.mtext

7. Assign length of sbuf.mtext to buf\_length

8. Apply msgsnd(msqid, &sbuf, buf\_length)

a. If returned –ve result

i. exit

b. End if

166

c. Else

i. Print “message sent”

9. End send()

Algorithm : receive()

1. Initialise key with 1234

2. Assign result of msgget(key, msgflg) to msqid

3. If msqid is less than 0

a. Print error

b. Exit

4. End if

5. Receive message using msgrcv(msqid, &rbuf, MSGSZ, 1, 0)

a. If returned –ve result

i. Print error

ii. Exit

b. End if

c. Else

i. Print rbuf.mtext

6. End receive()

Programs

Sender :

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <string.h>

167

#define MSGSZ 128

/\* Declare the message structure.\*/

typedef struct msgbuf {

long mtype;

char mtext[MSGSZ];

} message\_buf;

main()

{

int msqid;

int msgflg = IPC\_CREAT | 0666;

key\_t key;

message\_buf sbuf;

size\_t buf\_length;

/\*

\* Get the message queue id for the

\* "name" 1234, which was created by

\* the server.

\*/

key = 1234;

(void) fprintf(stderr, "\nmsgget: Calling msgget(%#lx,\

%#o)\n",

key, msgflg);

if ((msqid = msgget(key, msgflg )) < 0) {

perror("msgget");

exit(1);

}

else

(void) fprintf(stderr,"msgget: msgget succeeded: msqid = %d\n", msqid);

168

/\*

\* We'll send message type 1

\*/

sbuf.mtype = 1;

(void) fprintf(stderr,"msgget: msgget succeeded: msqid = %d\n", msqid);

(void) strcpy(sbuf.mtext, "Did you get this?");

(void) fprintf(stderr,"msgget: msgget succeeded: msqid = %d\n", msqid);

buf\_length = strlen(sbuf.mtext) + 1 ;

/\*

\* Send a message.

\*/

if (msgsnd(msqid, &sbuf, buf\_length, IPC\_NOWAIT) < 0) {

printf ("%d, %d, %s, %d\n", msqid, sbuf.mtype, sbuf.mtext, buf\_length);

perror("msgsnd");

exit(1);

}

else

printf("Message: \"%s\" Sent\n", sbuf.mtext);

exit(0);

}

Receiver :

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#define MSGSZ 128

/\*

\* Declare the message structure.

169

\*/

typedef struct msgbuf {

long mtype;

char mtext[MSGSZ];

} message\_buf;

main()

{

int msqid;

key\_t key;

message\_buf rbuf;

/\*

\* Get the message queue id for the

\* "name" 1234, which was created by

\* the server.

\*/

key = 1234;

if ((msqid = msgget(key, 0666)) < 0) {

perror("msgget");

exit(1);

}

/\*

\* Receive an answer of message type 1.

\*/

if (msgrcv(msqid, &rbuf, MSGSZ, 1, 0) < 0) {

perror("msgrcv");

exit(1);

}

/\* Print the answer.\*/

170

printf("%s\n", rbuf.mtext);

exit(0);

}

Output Sender:

$hp@mousmi:~/Desktop/panu/IPC/message queue$ ./send

msgget: Calling msgget(0x4d2,01666)

msgget: msgget succeeded: msqid = 0

msgget: msgget succeeded: msqid = 0

msgget: msgget succeeded: msqid = 0

Message: "Did you get this?" Sent

Output Reciever:

$hp@mousmi:~/Desktop/panu/IPC/message queue$ ./rec

Did you get this?

Conclusions:

Use of message queue functions like msgget, msgsend, and msgrecv to implement message passing mechanism between server and client studied. References: Dave’s Programming in C Tutorials