OPERATING SYSTEMS LAB - PRACTICAL 6 - MESSAGE QUEUE

Name - Sakshi Soni
Roll No - 13

AIM -
Develop an application for Inter-Process Communication using message queues

PROGRAM AND OUTPUT -

Program- 1

A message queue program that shows a client-server implementation

this is the receiver program using Message Queues

```
#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>


struct my_msg_st {

        long int my_msg_type;
```

```
        char some_text[BUFSIZ]; };

int main(void)
{
        int running = 1;
        int msgid;
        struct my_msg_st some_data;
        long int msg_to_recieve = 0;

        /* Let us set up the message queue */
        msgid = msgget((key_t)1234, 0666 | IPC_CREAT);

        if (msgid == -1) {
          perror("msgget failed with error");
          exit(EXIT_FAILURE);
        }

        /* Then the messages are retrieved from the queue, until an end message is
         * encountered. lastly the message queue is deleted
         */

        while(running) {
          if (msgrcv(msgid, (void *)&some_data, BUFSIZ,
```

```
                    msg_to_recieve, 0) == -1) {
      perror("msgcrv failed with error");
      exit(EXIT_FAILURE);
    }
    printf("You wrote: %s", some_data.some_text);
    if (strncmp(some_data.some_text, "end", 3) == 0) {
            running = 0;
    }
  }

  if (msgctl(msgid, IPC_RMID, 0) == -1) {
    perror("msgctl(IPC_RMID) failed");
    exit(EXIT_FAILURE);
  }

  exit(EXIT_SUCCESS);
}
```

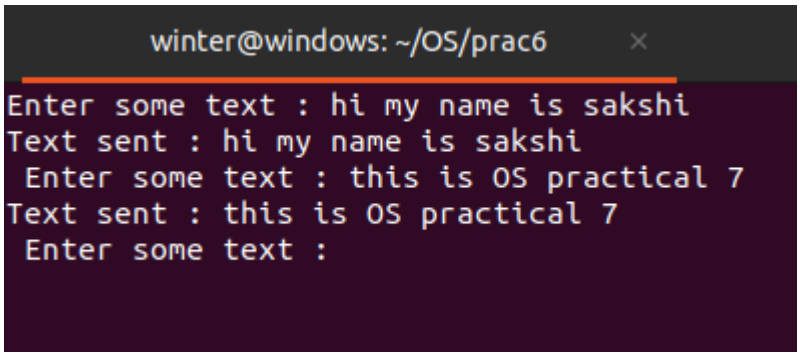
This is the sender program using Message Queues

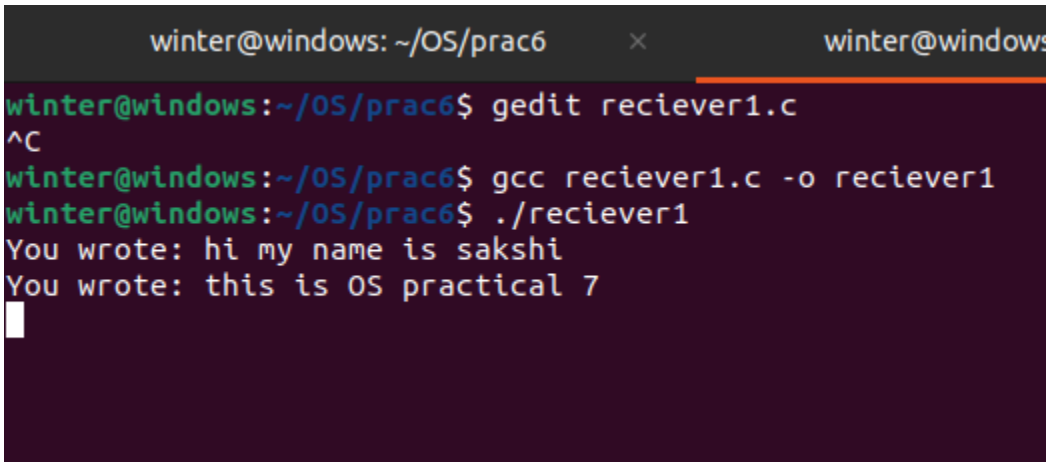
```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
```

```
#include <errno.h>
#include <string.h>

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MAX_TEXT 512

struct my_msg_st
{
	long int my_msg_type;
	char some_text[MAX_TEXT];
};

int main()
{
	int running = 1;
	int msgid;
	char sender[3] = "end";
	struct my_msg_st some_data;
	char buffer[BUFSIZ];
	system("clear");
	msgid = msgget((key_t)1234, 0666 | IPC_CREAT);
```

```
        if (msgid == -1)
        {
          fprintf(stderr,"msgget failed with error : %d
 ", errno);
          exit(EXIT_FAILURE);
        }

 while(running)
 {
printf("Enter some text : ");
fgets(buffer, BUFSIZ, stdin);
        if (strncmp(buffer, ender,3) == 0 )
        { running = 0; }
printf("Text sent : %s ", buffer);
some_data.my_msg_type = 1;
strcpy(some_data.some_text, buffer);
        if (msgsnd(msgid, (void *)&some_data, MAX_TEXT, 0) == -1)
        {
         // perror("msgsnd error");
        fprintf(stderr,"msgsnd failed ");
          exit(EXIT_FAILURE);
        }
```

```
}
        exit(EXIT_SUCCESS);
}
```





Program 2:

A simple implementation of IPC Message Queues.
IPC_msgq_send.c adds the message to the message queue.
IPC_msgq_rcv.c removes the message from the message queue.

```
//IPC_msgq_send.c

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAXSIZE     128

void die(char *s){
  perror(s);
  exit(1);
}

struct msgbuf{
    long  mtype;
    char  mtext[MAXSIZE];
};
```

```
main(){

    int msqid;

    int msgflg = IPC_CREAT | 0666;

    key_t key;

    struct msgbuf sbuf;

    size_t buflen;


    key = 1234;


    if ((msqid = msgget(key, msgflg )) < 0) //Get the message
queue ID for the given key

      die("msgget");


    //Message Type

    sbuf.mtype = 1;


    printf("Enter a message to add to message queue : ");

    scanf("%[^\n]",sbuf.mtext);

    getchar();


    buflen = strlen(sbuf.mtext) + 1 ;
```

```
    if (msgsnd(msqid, &sbuf, buflen, IPC_NOWAIT) < 0)
    {
        printf ("%d, %d, %s, %d\n", msqid, sbuf.mtype,
sbuf.mtext, buflen);
        die("msgsnd");
    }

    else
        printf("Message Sent\n");

    exit(0);
}
```

```
winter@windows:~/OS/prac6$ gedit IPC_msgq_send.c
^C
winter@windows:~/OS/prac6$ gcc IPC_msgq_send.c -o sender1
IPC_msgq_send.c:24:1: warning: return type defaults to ‘int’ [-
   24 | main(){
      | ^~~~
IPC_msgq_send.c: In function ‘main’:
IPC_msgq_send.c:47:23: warning: format ‘%d’ expects argument of
 argument 3 has type ‘long int’ [-Wformat=]
   47 |         printf ("%d, %d, %s, %d\n", msqid, sbuf.mtype, 
en);
      |                       ~^                    ~~~~~~~~~~
      |                        |                        |
      |                        int                      long int
      |                       %ld
IPC_msgq_send.c:47:31: warning: format ‘%d’ expects argument of
 argument 5 has type ‘size_t’ {aka ‘long unsigned int’} [-Wforma
   47 |         printf ("%d, %d, %s, %d\n", msqid, sbuf.mtype, 
en);
      |                               ~^
~~
      |                                |
      |                                int
_t {aka long unsigned int}
      |                               %ld
winter@windows:~/OS/prac6$ ./sender1
Enter a message to add to message queue : this is program 2
Message Sent
winter@windows:~/OS/prac6$
```

```
//IPC_msgq_rcv.c


#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <stdlib.h>

#define MAXSIZE     128


void die(char *s){

  perror(s);

  exit(1);

}

typedef struct msgbuf{

    long  mtype;

    char  mtext[MAXSIZE];

};

main()

{

    int msqid;

    key_t key;
```

```
    struct msgbuf rcvbuffer;

    key = 1234;

    if ((msqid = msgget(key, 0666)) < 0)
      die("msgget()");



     //Receive an answer of message type 1.
    if (msgrcv(msqid, &rcvbuffer, MAXSIZE, 1, 0)< 0)
      die("msgrcv");

    printf("%s\n", rcvbuffer.mtext);
    exit(0);
}
```

```
winter@windows:~/OS/prac6$ ./sender1
Enter a message to add to message queue : this is program 2
Message Sent
winter@windows:~/OS/prac6$ gedit IPC_msgq_rvc.c
^C
winter@windows:~/OS/prac6$ gcc IPC_msgq_rvc.c -o reciever1
IPC_msgq_rvc.c:19:1: warning: useless storage class specifier i
ion
   19 | };
      | ^
IPC_msgq_rvc.c:20:1: warning: return type defaults to 'int' [-W
   20 | main()
      | ^~~~
winter@windows:~/OS/prac6$ ./reciever1
this is program 2
winter@windows:~/OS/prac6$
```

Program 3:

Chat application program

client.c

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

#include <sys/ipc.h>

#include <sys/types.h>

#include <string.h>

#include <stdlib.h>

#include <stdio.h>



struct mesgq

{

long type;

char text[200];
```

```
} mq;



main()
{
int msqid, len;
key_t key = 2016;
if ((msqid = msgget(key, 0644)) == -1)
{
printf("Server not active\n");
exit(1);
}
printf("Client ready :\n");
while (msgrcv(msqid, &mq, sizeof(mq.text), 1, 0) != -1)
{
printf("From Server: \"%s\"\n", mq.text);
fgets(mq.text, sizeof(mq.text),stdin);
len = strlen(mq.text);
if (mq.text[len-1] == '\n')
mq.text[len-1] = '\0';
msgsnd(msqid, &mq, len+1, 0);
}
printf("Server Disconnected\n");
```

}

```
winter@windows:~/OS/prac6$ gedit client.c
^C
winter@windows:~/OS/prac6$ gcc client.c -o client
client.c:16:1: warning: return type defaults to ‘in
   16 | main()
      | ^~~~
winter@windows:~/OS/prac6$ ./client
Server not active
winter@windows:~/OS/prac6$
```

derver.c

```
#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>



struct mesgq

{

long type;

char text[200];

} mq;
```

```
main()
{
int msqid, len;
key_t key = 2016;
if((msqid = msgget(key, 0644|IPC_CREAT)) == -1)
{
perror("msgget error");
exit(1);
}
printf("Server ready :\n");
printf("Enter text, ^D to quit:\n");
mq.type = 1;
while(fgets(mq.text, sizeof(mq.text), stdin) != NULL)
{
len = strlen(mq.text);
if (mq.text[len-1] == '\n')
mq.text[len-1] = '\0';
msgsnd(msqid, &mq, len+1, 0);
msgrcv(msqid, &mq, sizeof(mq.text), 1, 0);
printf("From Client: \"%s\"\n", mq.text);
}
```

msgctl(msqid, IPC_RMID, NULL);

}

```
inter@windows:~/OS/prac6$ ./server
erver ready :
nter text, ^D to quit:
eyyy
rom Client: "heyyy"
C
inter@windows:~/OS/prac6$ ./client
lient ready :
elllo
C
inter@windows:~/OS/prac6$ ./server
erver ready :
nter text, ^D to quit:
```

Program 4:

Write a C program in Linux to sort the array in the sender process and pass that sorted array to the receiver process using the message queue. The receiving process should calculate the square of all received numbers.

Sender4.c

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/ipc.h>

#include <sys/msg.h>



// Define message structure

struct message {

    long mtype;
```

```
    int data;
};

// Function to sort the array
void bubbleSort(int arr[], int n) {
    int i, j;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

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

    // Generate a unique key
```

```
key = ftok(".", 'a');
if (key == -1) {
    perror("ftok");
    exit(1);
}

// Create a message queue
msgid = msgget(key, IPC_CREAT | 0666);
if (msgid == -1) {
    perror("msgget");
    exit(1);
}

// Input array size
int n;
printf("Enter the number of elements in the array: ");
scanf("%d", &n);

// Input array elements
int arr[n];
printf("Enter the elements of the array:\n");
for (int i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
```

```
}

// Sort the array
bubbleSort(arr, n);

// Send sorted array to receiver process
msg.mtype = 1;
for (int i = 0; i < n; i++) {
    msg.data = arr[i];
    if (msgsnd(msgid, &msg, sizeof(msg.data), 0) == -1) {
        perror("msgsnd");
        exit(1);
    }
}

printf("Sorted array sent to receiver process.\n");

// Remove the message queue
if (msgctl(msgid, IPC_RMID, NULL) == -1) {
    perror("msgctl");
    exit(1);
}
```

```
    return 0;
}
```

Reciever4.c

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

// Define message structure
struct message {
    long mtype;
    int data;
};

int main() {
    key_t key;
    int msgid;
    struct message msg;
```

```
// Generate the same unique key
key = ftok(".", 'a');
if (key == -1) {
    perror("ftok");
    exit(1);
}

// Get the message queue
msgid = msgget(key, 0666);
if (msgid == -1) {
    perror("msgget");
    exit(1);
}

// Receive and process the sorted array
printf("Received array from sender process:\n");
while (msgrcv(msgid, &msg, sizeof(msg.data), 1, IPC_NOWAIT) != -1) {
    printf("%d ", msg.data);
    int square = msg.data * msg.data;
    printf("Square: %d\n", square);
}

// Remove the message queue
```

```
    if (msgctl(msgid, IPC_RMID, NULL) == -1) {
        perror("msgctl");
        exit(1);
    }

    return 0;
}
```

```
winter@windows:~/OS/prac6$ gcc sender4.c -o server
winter@windows:~/OS/prac6$ gcc reciever4.c -o client
/usr/bin/ld: /usr/lib/gcc/x86_64-linux-gnu/11/../../../x8
: in function `_start':
(.text+0x1b): undefined reference to `main'
collect2: error: ld returned 1 exit status
winter@windows:~/OS/prac6$ ./server
Enter the number of elements in the array: 3
Enter the elements of the array:
3 4 1
Sorted array sent to receiver process.
```

RESULT -

Linux C programs for Inter-Process Communication using message queues have been implemented