OS - Practical – 6 Part 3

Name: Rishabh Jain

Branch & Sem: CSE - IV

Sec: A

Roll no.:54  
  
Aim: Develop an application for Inter-Process Communication using message queues.  
  
Program1   
A message queue program that shows a client server implementation this is the receiver program using Message Queues.

#include <errno.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <sys/types.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 ender[4] = "end"; // Corrected variable name from "sender" to "ender"

    struct my\_msg\_st some\_data;

    char buffer[BUFSIZ];

    system("clear");

    // Create or get the message queue with key 1234

    msgid = msgget((key\_t)1234, 0666 | IPC\_CREAT);

    if (msgid == -1)

    {

        fprintf(stderr, "msgget failed with error: %d\n", 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);

        // Send the message to the message queue

        if (msgsnd(msgid, (void \*)&some\_data, MAX\_TEXT, 0) == -1)

        {

            fprintf(stderr, "msgsnd failed\n");

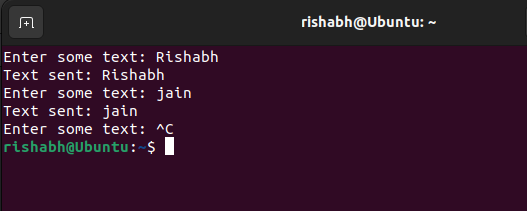
            exit(EXIT\_FAILURE);

        }

    }

    exit(EXIT\_SUCCESS);

}

OUTPUT :  
  


Program2

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <sys/types.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, %ld, %s, %ld\n", msqid, sbuf.mtype, sbuf.mtext, buflen);

        die("msgsnd");

    }

    else

        printf("Message Sent\n");

    exit(0);

}

// receiver

#include <stdio.h>

#include <stdlib.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <sys/types.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)

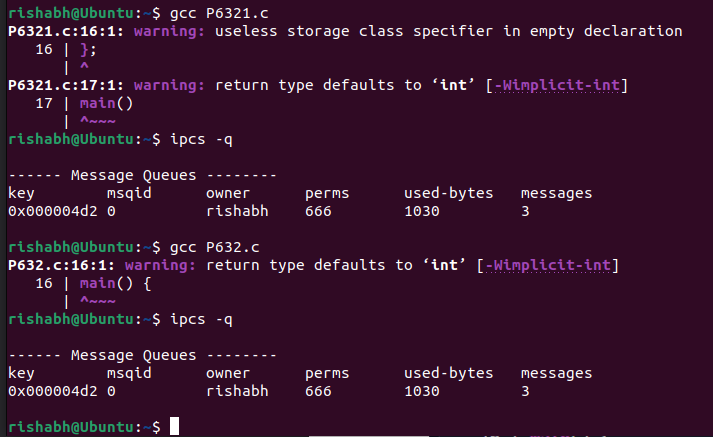
        die("msgrcv");

    printf("%s\n", rcvbuffer.mtext);

    exit(0);

}

OUTPUT:



**Program3**

chat application program (attached as client.c and server.c)

server.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define BUFFER\_SIZE 1024

int main()

{

    char message[BUFFER\_SIZE];

    while (1)

    {

        printf("Server: ");

        fgets(message, BUFFER\_SIZE, stdin);

        if (strcmp(message, "exit\n") == 0)

            break;

        memset(message, 0, sizeof(message));

    }

    printf("Chat ended\n");

    return 0;

}

Client.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define BUFFER\_SIZE 1024

int main()

{

    char message[BUFFER\_SIZE];

    while (1)

    {

        printf("You: ");

        fgets(message, BUFFER\_SIZE, stdin);

        if (strcmp(message, "exit\n") == 0)

            break;

        memset(message, 0, sizeof(message));

    }

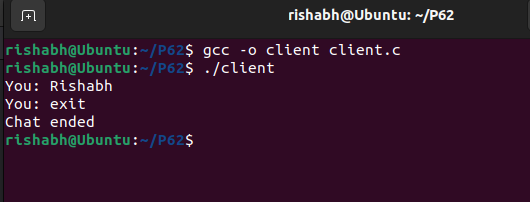
    printf("Chat ended\n");

    return 0;

}

OUTPUT:





**Program4**

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

s.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <sys/types.h>

#include <unistd.h>

#define MAX\_ARRAY\_SIZE 10

#define MESSAGE\_SIZE sizeof(struct message) - sizeof(long)

#define MSG\_TYPE 1

struct message

{

    long mtype;

    int array[MAX\_ARRAY\_SIZE];

};

void die(const char \*s)

{

    perror(s);

    exit(1);

}

void bubbleSort(int array[], int size)

{

    int i, j;

    for (i = 0; i < size - 1; i++)

    {

        for (j = 0; j < size - i - 1; j++)

        {

            if (array[j] > array[j + 1])

            {

                int temp = array[j];

                array[j] = array[j + 1];

                array[j + 1] = temp;

            }

        }

    }

}

int main()

{

    int i;

    int array[MAX\_ARRAY\_SIZE];

    struct message msg;

    key\_t key;

    int msqid;

    if ((key = ftok(".", 'q')) == -1)

        die("ftok");

    if ((msqid = msgget(key, 0666 | IPC\_CREAT)) == -1)

        die("msgget");

    printf("Enter %d numbers: ", MAX\_ARRAY\_SIZE);

    for (i = 0; i < MAX\_ARRAY\_SIZE; i++)

        scanf("%d", &array[i]);

    // Sort the array

    bubbleSort(array, MAX\_ARRAY\_SIZE);

    msg.mtype = MSG\_TYPE;

    memcpy(msg.array, array, MAX\_ARRAY\_SIZE \* sizeof(int));

    if (msgsnd(msqid, &msg, MESSAGE\_SIZE, 0) == -1)

        die("msgsnd");

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

    return 0;

}

r.c

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <sys/types.h>

#include <unistd.h>

#define MAX\_ARRAY\_SIZE 10

#define MESSAGE\_SIZE sizeof(struct message) - sizeof(long)

#define MSG\_TYPE 1

struct message

{

    long mtype;

    int array[MAX\_ARRAY\_SIZE];

};

void die(const char \*s)

{

    perror(s);

    exit(1);

}

int main()

{

    struct message msg;

    key\_t key;

    int msqid;

    int i;

    if ((key = ftok(".", 'q')) == -1)

        die("ftok");

    if ((msqid = msgget(key, 0666)) == -1)

        die("msgget");

    if (msgrcv(msqid, &msg, MESSAGE\_SIZE, MSG\_TYPE, 0) == -1)

        die("msgrcv");

    for (i = 0; i < MAX\_ARRAY\_SIZE; i++)

        msg.array[i] = pow(msg.array[i], 2);

    printf("Squared numbers received from sender process:\n");

    for (i = 0; i < MAX\_ARRAY\_SIZE; i++)

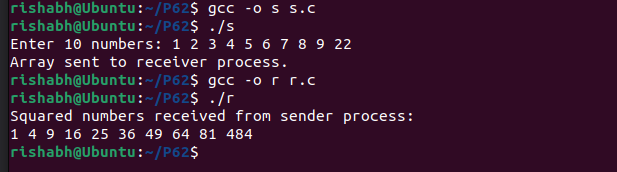
        printf("%d ", msg.array[i]);

    printf("\n");

    return 0;

}

OUTPUT:



**Result**

In this practical we learned about C programs for Inter-Process Communication using message queues .