OPERATING SYSTEMS LAB

Practical 6 (Part 3)

NAME: VEDANT BHUTADA
ROLL: 69
BATCH: A4

Aim: Develop an application for Inter-Process Communication using message queues.

Program- 1

A message queue program that shows a client server implementation

this is the reciever 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;
```

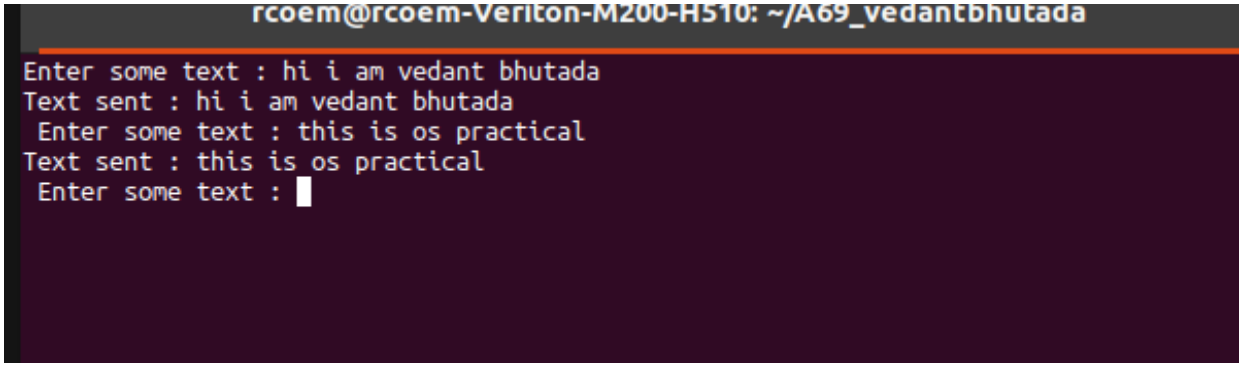
```
long int msg_to_recieve = 0;

/* 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);
}
```

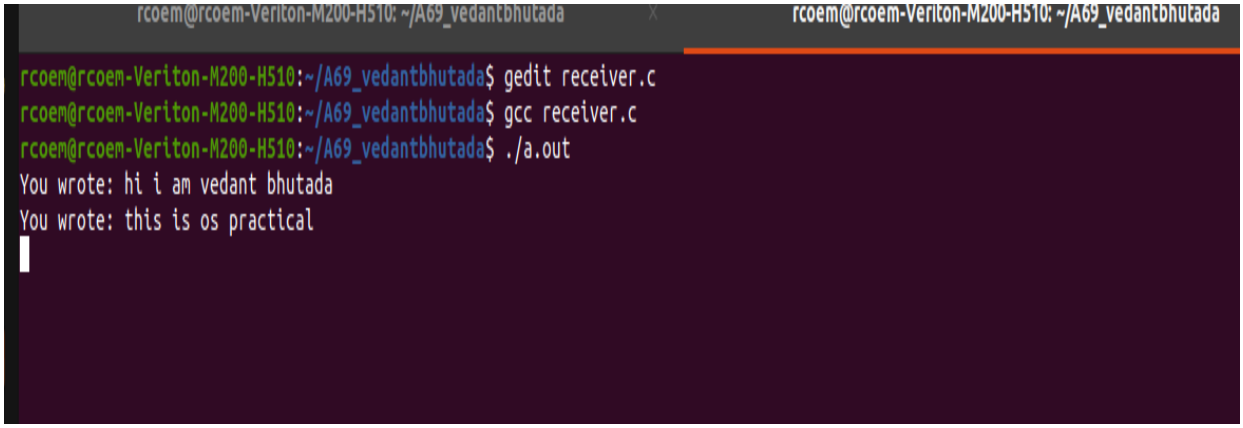
```
        exit(EXIT_SUCCESS);
}
```

This is the sender program using Message Queues

```
/*
 * A message queue program that shows a client server implementation
 * 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);
}
```





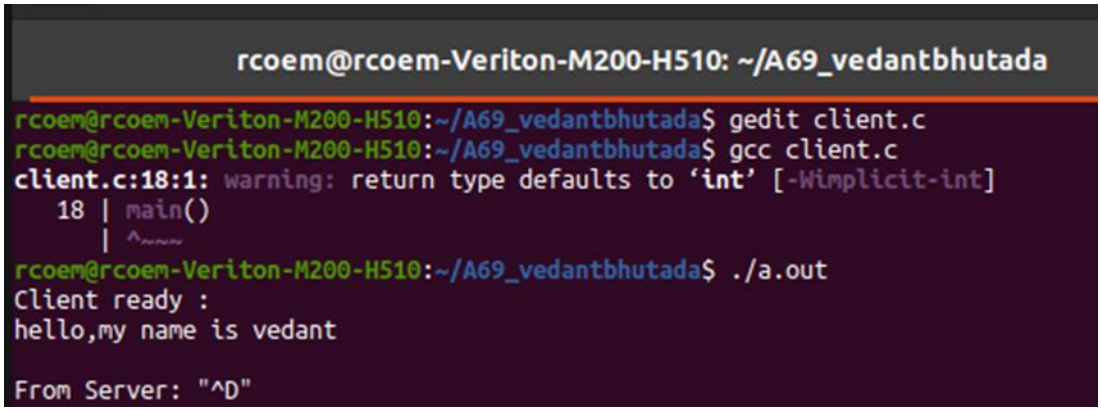
Program 3: chat application program (attached as client.c and server.c)

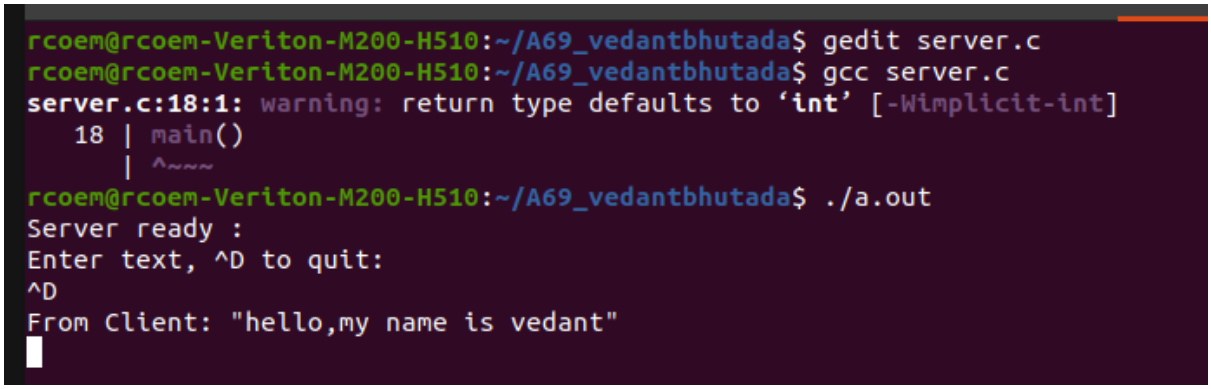
```
/*Program: Write a C program in Linux to implement Chat application
using message queue.
Code for Client
Client chat process - climsg.c */

#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");
}
```

```
/*Code for server
 Server chat process - srvmsg.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);
}
```





Program 4: 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.

Sender.c

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

#define MAX_SIZE 100

struct msg_buffer {
   long msg_type;
   int numbers[MAX_SIZE];
};

int compare(const void *a, const void *b) {
   return (*(int *)a - *(int *)b);
}

int main() {
   key_t key;
   int msg_id;
   struct msg_buffer msg;
```

```
    // Generate a unique key
    key = ftok("sender.c", 'A');

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

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

    if (size > MAX_SIZE) {
        printf("Size exceeds maximum limit. Exiting.\n");
        exit(1);
    }

    printf("Enter the array elements:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &msg.numbers[i]);
    }

    // Sort the array
    qsort(msg.numbers, size, sizeof(int), compare);

    // Set the message type
    msg.msg_type = 1;

    // Send the sorted array to the receiver process
    if (msgsnd(msg_id, &msg, size * sizeof(int), 0) == -1) {
        perror("msgsnd");
        exit(1);
    }

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

    // Remove the message queue
    msgctl(msg_id, IPC_RMID, NULL);

    return 0;
}
```

Receiver.c

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

#define MAX_SIZE 100
```

```
struct msg_buffer {
    long msg_type;
    int numbers[MAX_SIZE];
};

int main() {
    key_t key;
    int msg_id;
    struct msg_buffer msg;

    // Generate the same key as the sender process
    key = ftok("sender.c", 'A');

    // Access the message queue
    msg_id = msgget(key, 0666 | IPC_CREAT);
    if (msg_id == -1) {
        perror("msgget");
        exit(1);
    }

    // Receive the sorted array from the sender process
    if (msgrcv(msg_id, &msg, MAX_SIZE * sizeof(int), 1, 0) == -1) {
        perror("msgrcv");
        exit(1);
    }

    // Calculate the square of each number and print the result
    printf("Squared numbers: ");
    for (int i = 0; i < MAX_SIZE && msg.numbers[i] != '\0'; i++) {
        int square = msg.numbers[i] * msg.numbers[i];
        printf("%d ", square);
    }
    printf("\n");

    return 0;
}
```

```
rcoem@rcoem-Veriton-M200-H510:~/A69_vedantbhutada$ gcc sender1.c
rcoem@rcoem-Veriton-M200-H510:~/A69_vedantbhutada$ ./a.out
Enter the size of the array: 5
Enter the array elements:
1
4
6
2
3
Array sent to receiver process.
rcoem@rcoem-Veriton-M200-H510:~/A69_vedantbhutada$
```

```
rcoem@rcoem-Veriton-M200-H510:~/A69_vedantbhutada$ gcc receiver1.c
rcoem@rcoem-Veriton-M200-H510:~/A69_vedantbhutada$ ./a.out
Squared numbers: 1 4 9 16 36
rcoem@rcoem-Veriton-M200-H510:~/A69_vedantbhutada$
```

Result: Linux C programs **for Inter-Process Communication using message queues** has been implemented.

.