Lab7 – Message Queue and Shared Memory

Name: Rhea Adhikari
Reg No: 190905156
Roll No: 23
Batch: D-1

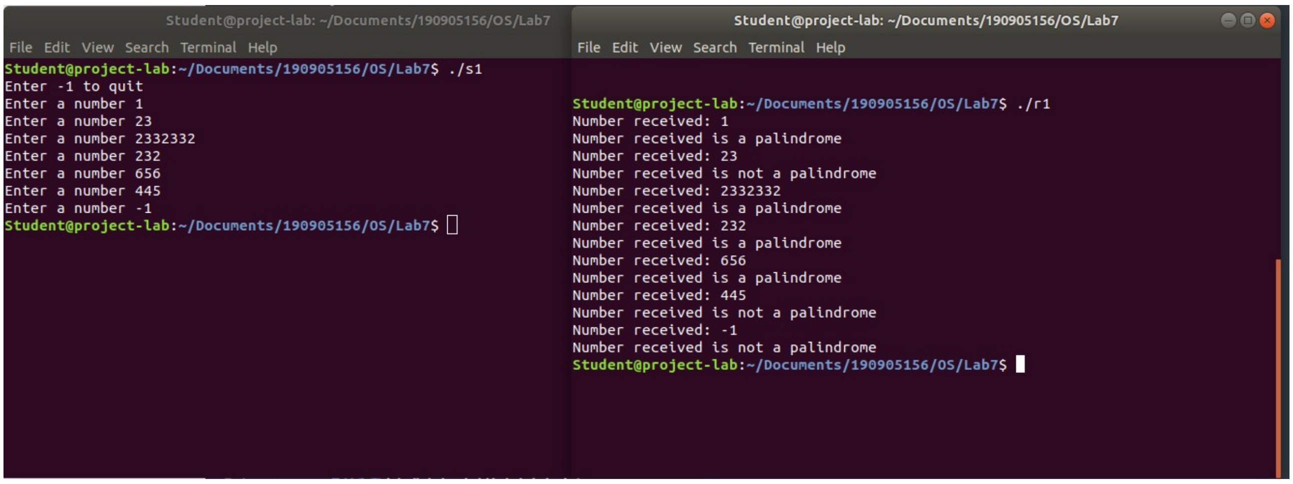
Q) Process A wants to send a number to Process B. Once received, Process B has to check whether the number is palindrome or not. Write a C program to implement this interprocess communication using a message queue.

1a.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<limits.h>
#include<fcntl.h>
#include<sys/msg.h>
#include<sys/stat.h>
#include<string.h>
#include<sys/msg.h>
#include<sys/ipc.h>
#include<errno.h>

#define MX_LEN 512

struct myMessage
{
    long int my_msg_type;
    int msg;
};

int reverse(int x)
{
    int y = 0;
    while(x > 0)
    {
        y *= 10;
        y += x % 10;
        x /= 10;
    }
    return y;
}

int main(int argc, char const *argv[])
{
    int running=1;
    struct myMessage some_data;
    long int msg_to_receive=0;
    int msgid;
    int num;
    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)
    {
        if (msgrcv(msgid,(void*)&some_data,BUFSIZ,msg_to_receive,0)==-1)
        {
            fprintf(stderr, "msgrc failedwith error %d\n",errno );
            exit(EXIT_FAILURE);
        }

        printf("Number received: %d\n",some_data.msg);

        if(some_data.msg == reverse(some_data.msg))
            printf("Number received is a palindrome\n");
        else
            printf("Number received is not a palindrome\n");
        if(some_data.msg==-1)
            running=0;
    }
    if(msgctl(msgid,IPC_RMID,0)==-1){
        fprintf(stderr, "msgctl(IPC_RMID) failed\n");
        exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
```

1b.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<limits.h>
#include<fcntl.h>
#include<sys/msg.h>
#include<sys/stat.h>
#include<string.h>
#include<sys/msg.h>
#include<sys/ipc.h>
#include<errno.h>
#define MAX_TEXT 512

struct my_msg_st
{
    long int my_msg_type;
    int msg;
};

int main(int argc, char const *argv[])
{
    int running=1;
    struct my_msg_st some_data;
    int msgid;
    int num;
    msgid=msgget((key_t)1234,0666|IPC_CREAT);

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

    printf("Enter -1 to quit\n");

    while(running)
    {
        printf("Enter a number ");
        scanf("%d",&num);
        some_data.my_msg_type=1;
        some_data.msg=num;
        if (msgsnd(msgid,(void*)&some_data,MAX_TEXT,0)==-1){
            fprintf(stderr, "msgsnd failed\n" );
            exit(EXIT_FAILURE);
        }
        if(num==-1)
            running=0;
    }

    exit(EXIT_SUCCESS);
}
```


Student@project-lab: ~/Documents/190905156/OS/Lab7
File Edit View Search Terminal Help
Student@project-lab: ~/Documents/190905156/OS/Lab7
File Edit View Search Terminal Help

```
Student@project-lab:~/Documents/190905156/OS/Lab7$ ./s1
Enter -1 to quit
Enter a number 1
Enter a number 23
Enter a number 2332332
Enter a number 232
Enter a number 656
Enter a number 445
Enter a number -1
Student@project-lab:~/Documents/190905156/OS/Lab7$
```

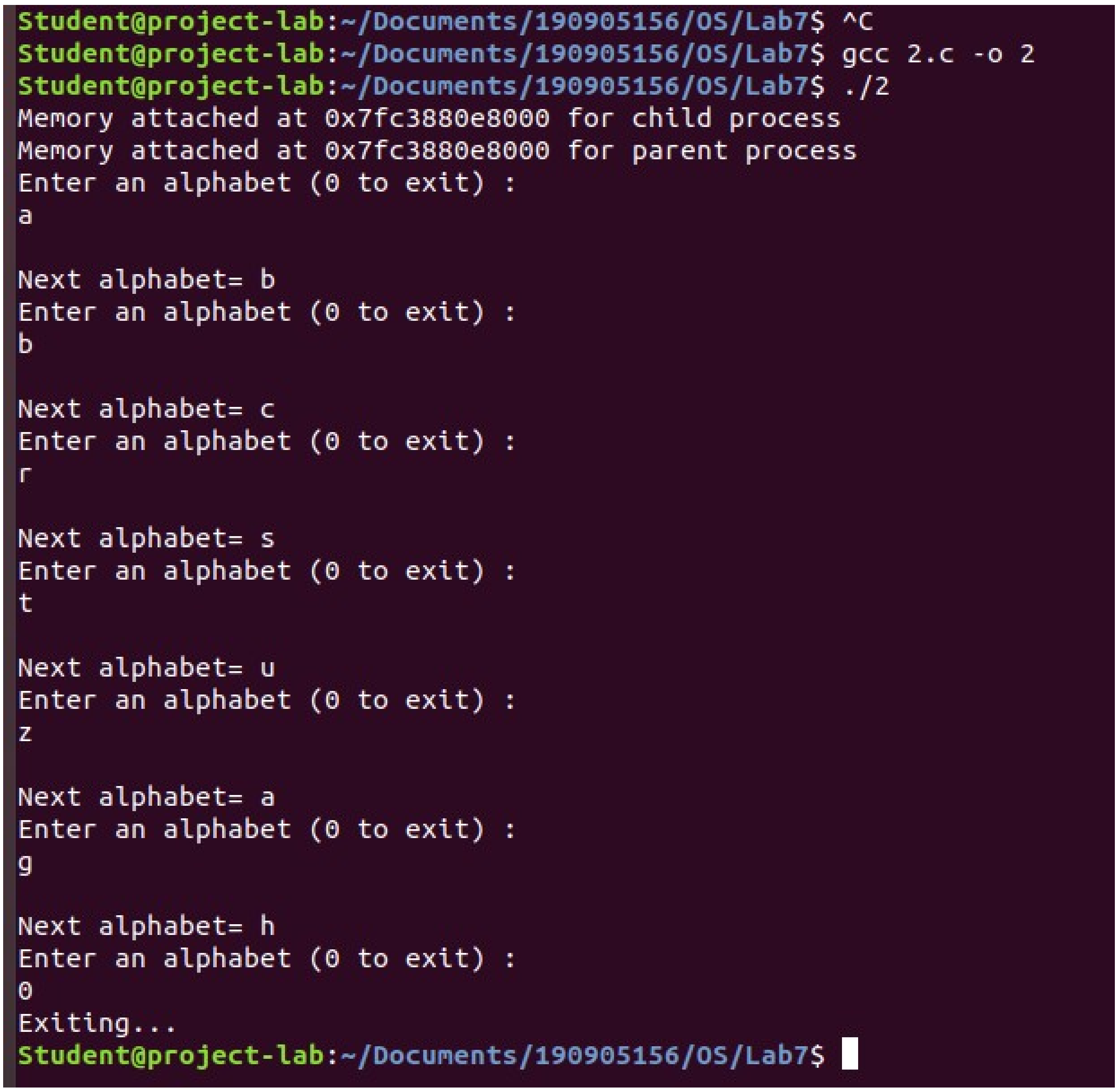
```
Student@project-lab:~/Documents/190905156/OS/Lab7$ ./r1
Number received: 1
Number received is a palindrome
Number received: 23
Number received is not a palindrome
Number received: 2332332
Number received is a palindrome
Number received: 232
Number received is a palindrome
Number received: 656
Number received is a palindrome
Number received: 445
Number received is not a palindrome
Number received: -1
Number received is not a palindrome
Student@project-lab:~/Documents/190905156/OS/Lab7$
```

Q) Implement a parent process, which sends an English alphabet to a child process using shared memory. The child process responds with the next English alphabet to the parent. The parent displays the reply from the Child.

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

struct sharedStr
{
    int status;
    char alphabet;
};

int main(int argc, char const *argv[])
{
    /*
    Status codes
```

```
0 -> nothing written yet by parent process
1 -> alphabet written by parent process
2 -> answer written by child process
-1 -> exit
*/
int shmid = shmget((key_t)1234,sizeof(struct sharedStr),0666|IPC_CREAT);
pid_t pid = fork();

if(pid < 0)
{
    printf("Error in fork()\n");
    exit(-1);
}

else if(pid == 0)
{ //child process
    struct sharedStr* shared_memory = shmat(shmid,(void*)0,0);

    if(shared_memory == (void*)-1)
    {
        printf("shmat() failed\n");
        exit(-1);
    }

    printf("Memory attached at %p for child process\n",shared_memory);

    while(1)
    {
        if(shared_memory->status < 0)
        {
            // printf("Exit code received %d\n",shared_memory->status);
            if(shmdt(shared_memory) == -1)
            {
                printf("shmdt failed\n");
                exit(-1);
            }

            break;
        }

        if(shared_memory->status == 1)
        {
            char c = shared_memory->alphabet;
            printf("\n");

            if((int)c >= 65 && (int)c <= 90)
            { //uppercase
                c = ((c - 'A' + 1)%26) + 'A';
            }

            else if((int)c >= 97 && (int)c <= 122)
            { //lowecase
                c = ((c - 'a' + 1)%26) + 'a';
            }

            else
            {
                printf("Non-alphabetic character received\n");
                //do nothing
            }
            shared_memory->alphabet = c; //write to shared memory
            shared_memory->status = 2;
        }
    }
}

else
{ //parent process
    sleep(1);
    struct sharedStr* shared_memory = shmat(shmid,(void*)0,0);

    if(shared_memory == (void*)-1)
    {
        printf("shmat() failed\n");
        exit(-1);
    }

    printf("Memory attached at %p for parent process\n",shared_memory);
    shared_memory->status = 0;

    while(1)
    {
        if(shared_memory->status == 1)
        {
            // printf("Waiting for child process\n");
            continue;
        }

        if(shared_memory->status == 2)
        {
            printf("%c\n",shared_memory->alphabet);
        }

        shared_memory->status = 0;
        char c,nl;
        printf("Enter an alphabet (0 to exit) : \n");
        scanf("%c",&c);
        if(c != '0')
        printf("Next alphabet= ");
        scanf("%c",&nl);

        if(c == '0')
        {
            shared_memory->status = -1;
            printf("Exiting...\n");

            if(shmdt(shared_memory) == -1)
            {
                printf("shmdt failed\n");
                exit(-1);
            }

            if(shmctl(shmid,IPC_RMID,0) == -1)
            {
                printf("shmctl failed\n");
                exit(-1);
            }
            break;
```

```
          }

          shared_memory->alphabet = c;
          shared_memory->status = 1;
       }
    }
    return 0;
}
```



```
Student@project-lab:~/Documents/190905156/OS/Lab7$ ^C
Student@project-lab:~/Documents/190905156/OS/Lab7$ gcc 2.c -o 2
Student@project-lab:~/Documents/190905156/OS/Lab7$ ./2
Memory attached at 0x7fc3880e8000 for child process
Memory attached at 0x7fc3880e8000 for parent process
Enter an alphabet (0 to exit) :
a

Next alphabet= b
Enter an alphabet (0 to exit) :
b

Next alphabet= c
Enter an alphabet (0 to exit) :
r

Next alphabet= s
Enter an alphabet (0 to exit) :
t

Next alphabet= u
Enter an alphabet (0 to exit) :
z

Next alphabet= a
Enter an alphabet (0 to exit) :
g

Next alphabet= h
Enter an alphabet (0 to exit) :
0
Exiting...
Student@project-lab:~/Documents/190905156/OS/Lab7$
```