1)

File – receiver.c

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

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

int reverse(int num)
{
   int rev=0;
   while(num>0)
   {
      int digit = num%10;
      rev = rev*10 + digit;
      num = num/10;
   }
   return rev;
}
int isPalindrome(int num)
{
   int rev = reverse(num);
   return (rev==num);
}

int main()
{
   int running=1;
   int msgid;
   struct my_msg_st data;
   long msg_to_receive = 1;
   msgid = msgget((key_t)1234,0666|IPC_CREAT);
   if(msgid==-1)
   {
      fprintf(stderr,"msgget failed with error number %d\n",errno);
      exit(EXIT_FAILURE);
   }
   while(running)
```

```
    {
        if(msgrcv(msgid,(void*)&data,sizeof(data),msg_to_receive,0)==-1)
        {
            fprintf(stderr,"msgrcv failed with error number %d\n",errno);
            exit(EXIT_FAILURE);
        }
        if(isPalindrome(data.num))
        {
            printf("%d is a Palindrome\n",data.num);
        }
        else
        {
            printf("%d is not a Palindrome\n",data.num);
        }

        if(data.num==-1)
        {
            running=0;
        }
    }

    if(msgctl(msgid,IPC_RMID,0)==-1)
    {
        fprintf(stderr,"msgctl(IPC_RMID) failed with error number %d\n",errno);
        exit(EXIT_FAILURE);
    }
    return EXIT_SUCCESS;
}
```

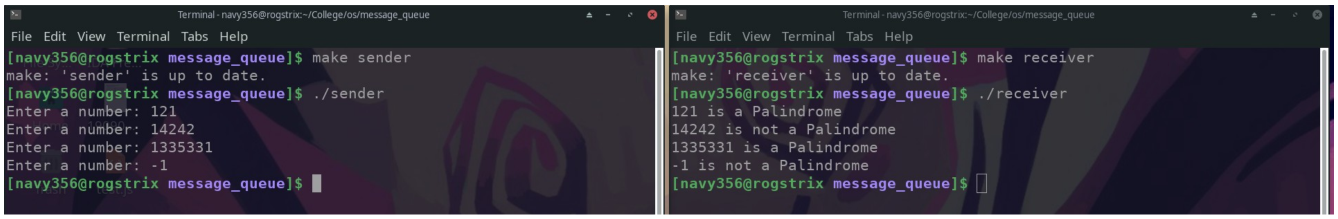
File – sender.c

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

struct my_msg_st
{
    long int my_msg_type;
    int num;
};
```

```
int main()
{
    int running = 1;
    int msgid;
    struct my_msg_st data;
    int num;
    msgid = msgget((key_t)1234,0666|IPC_CREAT);
    if(msgid==-1)
    {
        fprintf(stderr,"msgget failed with error number %d\n",errno);
        exit(EXIT_FAILURE);
    }
    while(running)
    {
        printf("Enter a number: ");
        scanf("%d",&num);
        data.my_msg_type=1;
        data.num = num;

        if(msgsnd(msgid,(void *)&data,sizeof(data),0)==-1)
        {
            fprintf(stderr,"msgsnd failed with error number %d\n",errno);
            exit(EXIT_FAILURE);
        }
        if(data.num==-1)
        {
            running=0;
        }
    }
    return EXIT_SUCCESS;
}
```

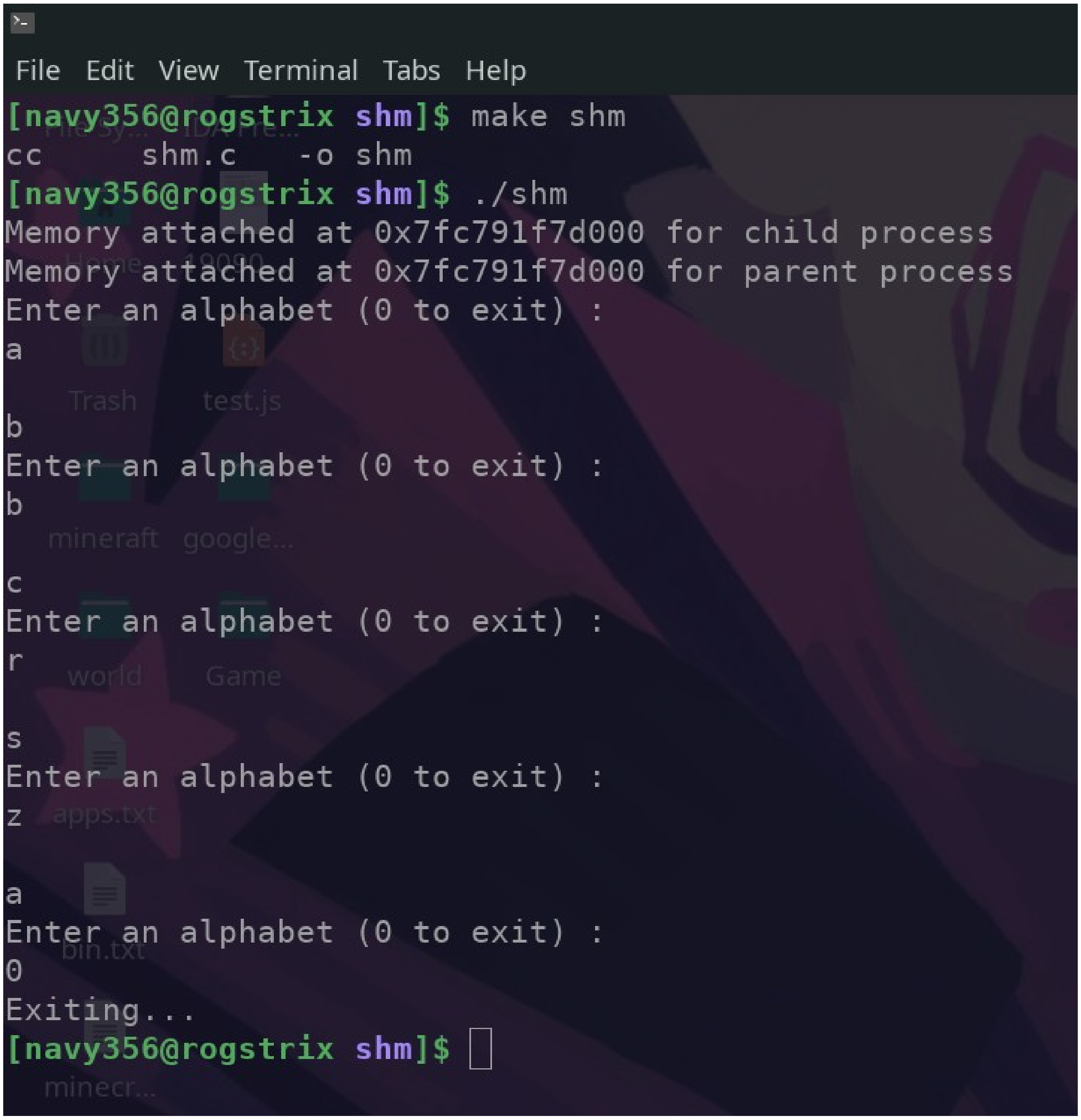

Terminal - navy356@rogstrix:~/College/os/message_queue
File Edit View Terminal Tabs Help
Terminal - navy356@rogstrix:~/College/os/message_queue
File Edit View Terminal Tabs Help

```
[navy356@rogstrix message_queue]$ make sender
make: 'sender' is up to date.
[navy356@rogstrix message_queue]$ ./sender
Enter a number: 121
Enter a number: 14242
Enter a number: 1335331
Enter a number: -1
[navy356@rogstrix message_queue]$
```

```
[navy356@rogstrix message_queue]$ make receiver
make: 'receiver' is up to date.
[navy356@rogstrix message_queue]$ ./receiver
121 is a Palindrome
14242 is not a Palindrome
1335331 is a Palindrome
-1 is not a Palindrome
[navy356@rogstrix message_queue]$
```

2) File – shm.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
struct shared_str
{
    int status;
    char alphabet;
};
void child(int shmid)
{
    struct shared_str *shared_mem = shmat(shmid, (void *)0, 0);
    if (shared_mem == (void *)-1)
    {
        printf("shmat() failed\n");
        exit(-1);
    }
    printf("Memory attached at %p for child process\n", shared_mem);
    while (1)
    {
        if (shared_mem->status < 0)
        {
            if (shmdt(shared_mem) == -1)
            {
                printf("shmdt failed\n");
                exit(-1);
            }
            break;
        }
        if (shared_mem->status == 1)
        {
            char c = shared_mem->alphabet;
            printf("\n");
            if ((int)c >= 65 && (int)c <= 90)
            {
                c = ((c - 'A' + 1) % 26) + 'A';
            }
            else if ((int)c >= 97 && (int)c <= 122)
            {
                c = ((c - 'a' + 1) % 26) + 'a';
            }
            else
            {
                printf("Non-alphabetic character received\n");
                //do nothing
            }
            shared_mem->alphabet = c; //write to shared memory
            shared_mem->status = 2;
        }
    }
}
void parent(int shmid)
{
    sleep(1);
    struct shared_str *shared_mem = shmat(shmid, (void *)0, 0);
    if (shared_mem == (void *)-1)
    {
        printf("shmat() failed\n");
        exit(-1);
    }
    printf("Memory attached at %p for parent process\n", shared_mem);
    shared_mem->status = 0;
    while (1)
    {
        if (shared_mem->status == 1)
        {
            continue;
        }
        if (shared_mem->status == 2)
        {
            printf("%c\n", shared_mem->alphabet);
        }
        shared_mem->status = 0;
        char c, nl;
        printf("Enter an alphabet (0 to exit) : \n");
        scanf("%c", &c);
        scanf("%c", &nl);
        if (c == '0')
        {
            shared_mem->status = -1;
            printf("Exiting...\n");
            if (shmdt(shared_mem) == -1)
            {
                printf("shmdt failed\n");
                exit(-1);
            }
            if (shmctl(shmid, IPC_RMID, 0) == -1)
            {
                printf("shmctl failed\n");
                exit(-1);
            }
            break;
        }
        shared_mem->alphabet = c;
        shared_mem->status = 1;
    }
}
int main(int argc, char const *argv[])
{
    int shmid = shmget((key_t)1234, sizeof(struct shared_str), 0666 | IPC_CREAT);
    pid_t pid = fork();
    if (pid < 0)
    {
        printf("Error in fork()\n");
        exit(-1);
    }
    else if (pid == 0)
    {
        //child process
        child(shmid);
    }
    else
    {
        //parent process
        parent(shmid);
    }
    return 0;
}
```



```
[navy356@rogstrix shm]$ make shm
cc     shm.c   -o shm
[navy356@rogstrix shm]$ ./shm
Memory attached at 0x7fc791f7d000 for child process
Memory attached at 0x7fc791f7d000 for parent process
Enter an alphabet (0 to exit) :
a

b
Enter an alphabet (0 to exit) :
b

c
Enter an alphabet (0 to exit) :
r

s
Enter an alphabet (0 to exit) :
z

a
Enter an alphabet (0 to exit) :
0
Exiting...
[navy356@rogstrix shm]$ 
```