How to Execute:

1. Compile processor.c with output file name as processor

gcc processor.c -o processor

2. Compile receiver.c with any output file name

gcc receiver.c -o receiver

3. Execute receiver.exe (Secret code – C00L with zero and not capital o in between)

NOTE: Digit count will be +2 because of 00 in C00L

```
10-18-134-74:Assignment_2 rakshitsareen$ ./receiver

Enter your secret line: Hi Sareen is so C00L

Shared memory attached at: 0x105135000

This line was stored in secrets file: Hi Sareen is so C00L

Enter your secret line: _
```

RECEIVER.C

```
if ((sharedmemoryid = shmget(key, MAXSIZE, IPC_CREAT | 0666)) < 0) //create new memory segment using shmget
        die("shmget");
```

We create a new memory segment using the function shmget which passes in arguments like a unique key so that we can actually synchronize the receiver and processor section and it can both access the same shared memory segment. The IPC_CREAT flag makes sure that a new segment is created. A unique id corresponding to the memory segment created is sent back and that gets stored in the variable sharedmemoryid. The if statement checks if the address is less than 0 or not. It throws an error if it is less than 0, meaning address could not be retrieved

```
if ((shm = shmat(sharedmemoryid, NULL, 0)) == (char *) -1) //Using shmat function to make shared memory available and return shared memory address
        die("shmat");
```

To make the shared memory segment available we need to use the shmat function. It takes as argument the id of the shared memory segment that we have created. If we use NULL as second argument the LINUX OS would find an address for this memory segment that we have created. The third argumentsare for flags and since we don't have any specific needs we leave it. If the process is successful this function returns the address of our shared memory segment.

PROCESSOR.C

```
    if ((sharedmemoryid = shmget(key, MAXSIZE, 0666)) < 0) //accessing the same shared memory using key
        die("shmget");
```

We are accessing the same memory segment that was created by the receiver process by passing in the same key value. This function would return the same segment id for the shared memory segment as in the case of the receiver process as we are actually trying to access the same shared memory segment

```
    if ((shm = shmat(sharedmemoryid, NULL, 0)) == (char *) -1) //retrieving the address of shared memory
        die("shmat");
```

Using the shmat function to get the address for the memory segment id that we pass

```
    storage = shmat(sharedmemoryid, NULL, SHM_RDONLY);
```

We store the contents of the shared memory segment into a variable using the shmat function and passing a flag SHM_RDONLY which would read the contents from the shared memory address and copy it to the variable storage