

OS REPORT CHARACTER DRIVER

CHARACTER DEVICE DRIVER WITH ENCRYPTION

AND DECRYPTION FUNCTIONALITY

1. Problem Statement

Create a simple device driver (character device) for the current compiled kernel and test it with your sample application.

2. Methodology:

Key Function:

Our character driver takes in an input string. It randomly generates the required key randomly using URandom Module. Then it Encrypts the message stream.

We have written a user program which Decrypts the message stream using the required random key stored.

a) The steps followed are:

1. We have implemented basic I/O operations like read(), write(), open() and close() functions in these character devices, however for open and close have used the kernel callback function to do so.
2. In cryptography, a cipher (or cypher) is an algorithm for performing encryption or decryption—a series of well-defined steps that can be followed as a procedure. An alternative, less common term is encipherment. Our device is capable of encrypting and copies 16 bytes at a time from the user and encrypts it using block ciphers.
3. The key is generated randomly using URandom Module which can be accessed by /dev/urandom and is passed from user space automatically. The encryption device passes the encrypted text back to user space. In user space, this encrypted value is written to a file along with the key.
4. Similarly for decryption, the decryption user space program reads the key and the encrypted text from the file and sends this to the decryption device in blocks of 16 bytes. This is followed by sending back the decrypted text from the decryption device and is then printed on the screen. Frequent updates are printed to the kernel log.

This is the basic functionality of the Character Device we implemented. This can be used especially for masking/ hiding the input stream which the user is inputting.

3. Explanation

The steps followed:

1. We booted the Linux kernel which we had compiled for the previous assignment through GRUB and selected the Kernel we compiled.
To access the GRUB : Long press the SHIFT key while it is booting. This launches the GRUB menu.

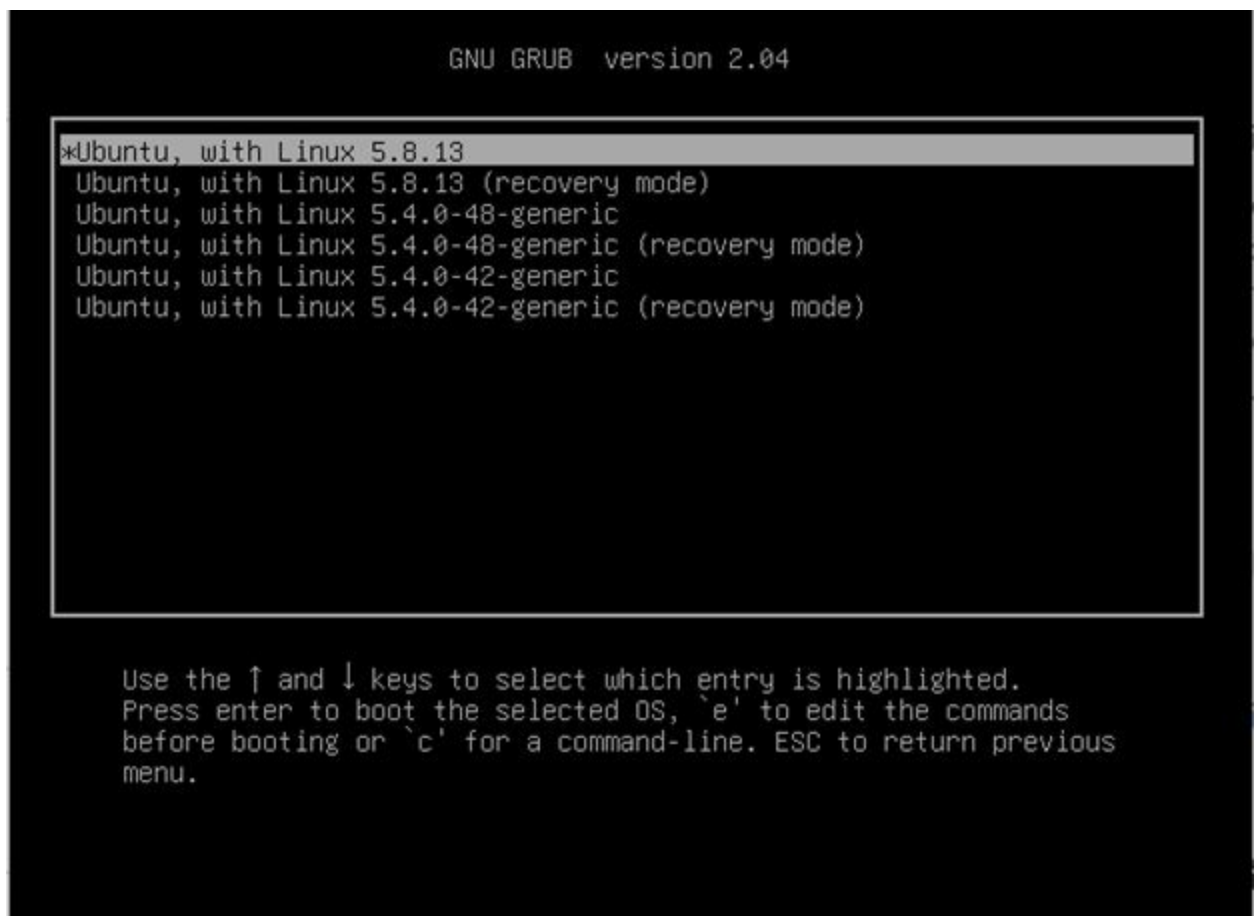


fig (i):Grub menu

2. Then we opened Terminal. And installed the relevant Linux Headers.
Using the command:

```
$ sudo apt-get install linux-headers-$(uname -r)
```

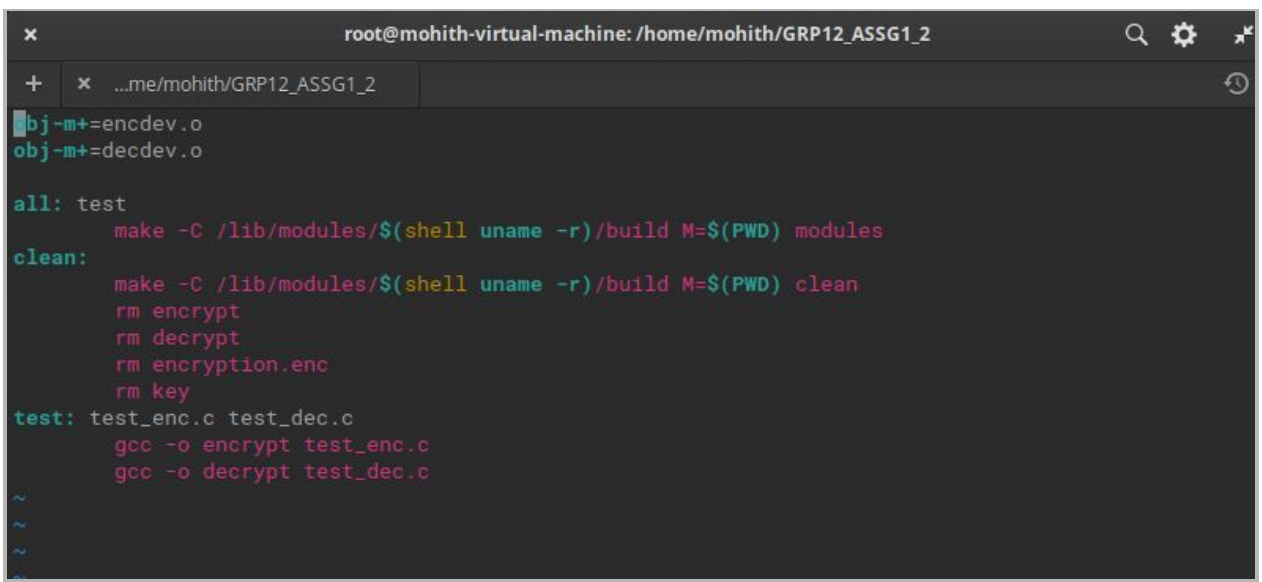
3. Then we wrote the relevant device driver `encdev.c` and `decdev.c` .
`encdev.c` - It will encrypt the given input character stream in blocks of 16 characters when user calls `write()` operation.
`decdev.c` - It will decrypt the code in the file which is being encrypted by the encryption function with the provided cipher using xor operator when the user calls the `read()` operation.

```
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2# ls
decdev.c  encdev.c  Makefile  README.md  run.sh  setup.sh  test_dec.c  test_enc.c  writeup.txt
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2#
```

fig(ii):files list

4. We use the *make* utility.
The purpose of the *make* utility is to determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them.

```
$ make -C /lib/modules/$(uname -r)/build M=$PWD modules
```

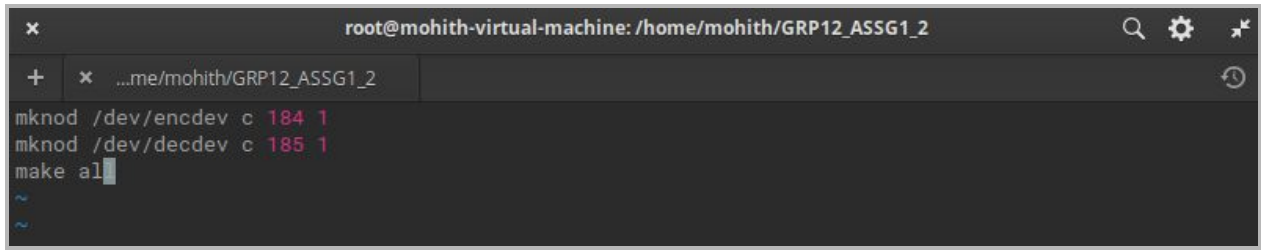


```
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2
+ x ...me/mohith/GRP12_ASSG1_2
obj-m+=encdev.o
obj-m+=decdev.o

all: test
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
    rm encrypt
    rm decrypt
    rm encryption.enc
    rm key
test: test_enc.c test_dec.c
    gcc -o encrypt test_enc.c
    gcc -o decrypt test_dec.c
```

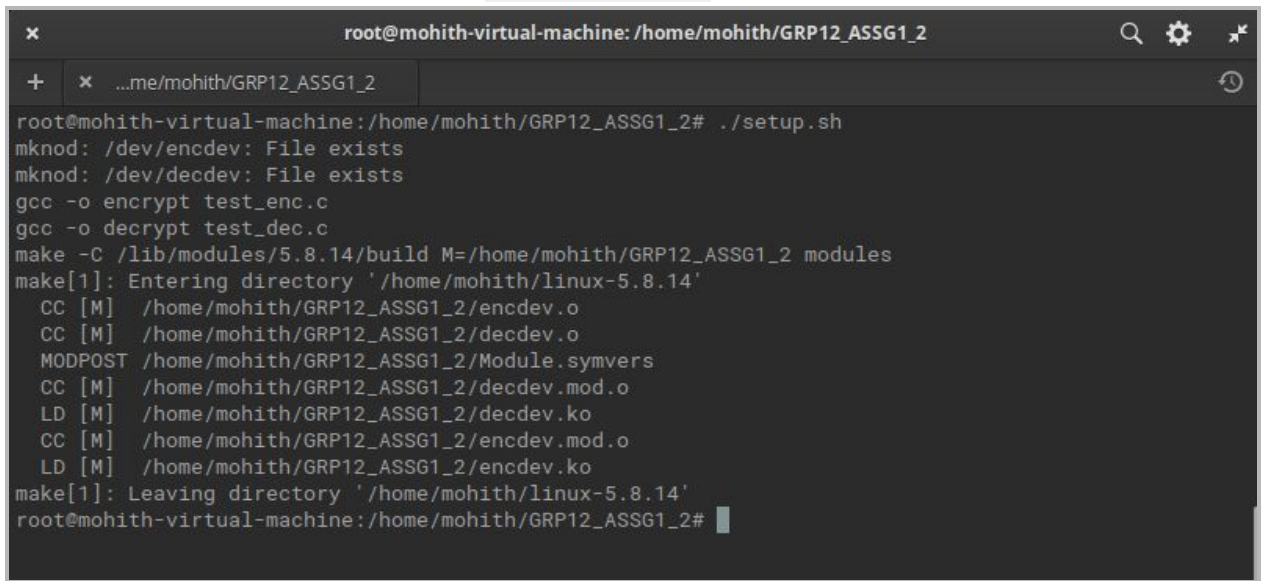
fig(iii):makefile

5. We inserted both modules using the below commands.



```
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2
mknod /dev/encdev c 184 1
mknod /dev/decdev c 185 1
make all
```

fig(iv):setup.sh

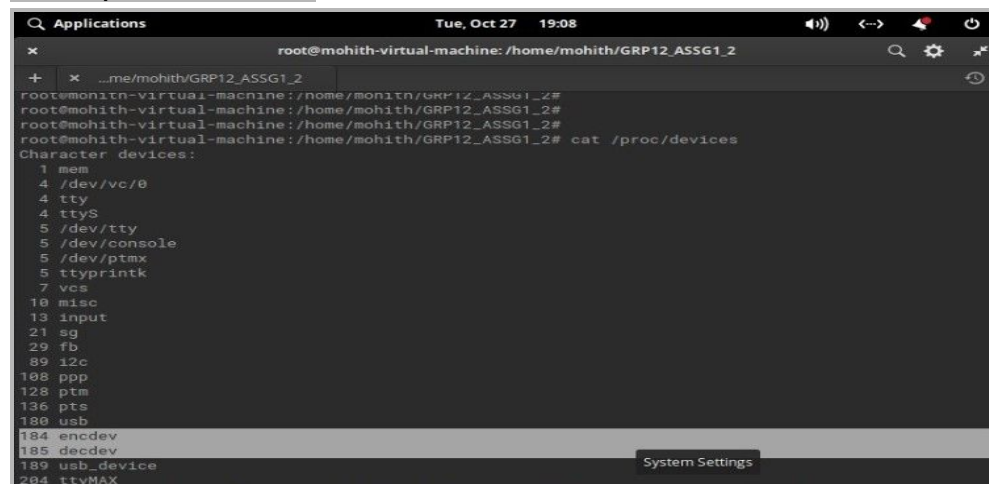


```
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2# ./setup.sh
mknod: /dev/encdev: File exists
mknod: /dev/decdev: File exists
gcc -o encrypt test_enc.c
gcc -o decrypt test_dec.c
make -C /lib/modules/5.8.14/build M=/home/mohith/GRP12_ASSG1_2 modules
make[1]: Entering directory '/home/mohith/linux-5.8.14'
CC [M] /home/mohith/GRP12_ASSG1_2/encdev.o
CC [M] /home/mohith/GRP12_ASSG1_2/decdev.o
MODPOST /home/mohith/GRP12_ASSG1_2/Module.symvers
CC [M] /home/mohith/GRP12_ASSG1_2/decdev.mod.o
LD [M] /home/mohith/GRP12_ASSG1_2/decdev.ko
CC [M] /home/mohith/GRP12_ASSG1_2/encdev.mod.o
LD [M] /home/mohith/GRP12_ASSG1_2/encdev.ko
make[1]: Leaving directory '/home/mohith/linux-5.8.14'
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2#
```

fig(v): ./setup.sh

6. We then checked whether the following drivers got inserted.

\$ cat /proc/devices



```
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2#
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2#
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2# cat /proc/devices
Character devices:
1 mem
4 /dev/vc/0
4 tty
4 ttyS
5 /dev/tty
5 /dev/console
5 /dev/ptmx
5 ttyprintk
7 ves
10 misc
13 input
21 sg
29 fb
89 l2c
108 ppp
128 ptm
136 pts
180 usb
184 encdev
185 decdev
189 usb_device
204 ttyMAX
```

fig(vi): list of character devices

The kernel logs can be checked and verified for proper insertion by using the ***dmesg*** command.

```
$ dmesg
```

```
[14235.994483] encdev: Device Registered
[14236.007025] decdev: Device Registered
[14250.333327] encdev: device was opened
[14273.808316] decdev: Successfully encrypted block 1 of data
[14273.808317] encdev: Read the encrypted text successfully
[14273.808320] encdev: device was closed
[14284.523273] decdev: device was opened
[14284.523278] decdev: Successfully decrypted block 1 of data
[14284.523279] decdev: Successfully read decrypted data
[14284.523280] decdev: device was closed
```

fig(vii): ring buffers

7. Execute the run.sh file

```
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2
+ x ...me/mohith/GRP12_ASSG1_2
mmmod encdev
rmmmod decdev
make all
insmod encdev.ko
insmod decdev.ko
~
~
~
```

fig(viii): run.sh

```
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2# ./run.sh
gcc -o encrypt test_enc.c
gcc -o decrypt test_dec.c
make -C /lib/modules/5.8.14/build M=/home/mohith/GRP12_ASSG1_2 modules
make[1]: Entering directory '/home/mohith/linux-5.8.14'
make[1]: Leaving directory '/home/mohith/linux-5.8.14'
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2#
```

fig(ix): ./run.sh

8. We then wrote the test files to test the functionality of the driver. We wrote user programs *test_enc.c* and *test_dec.c*.

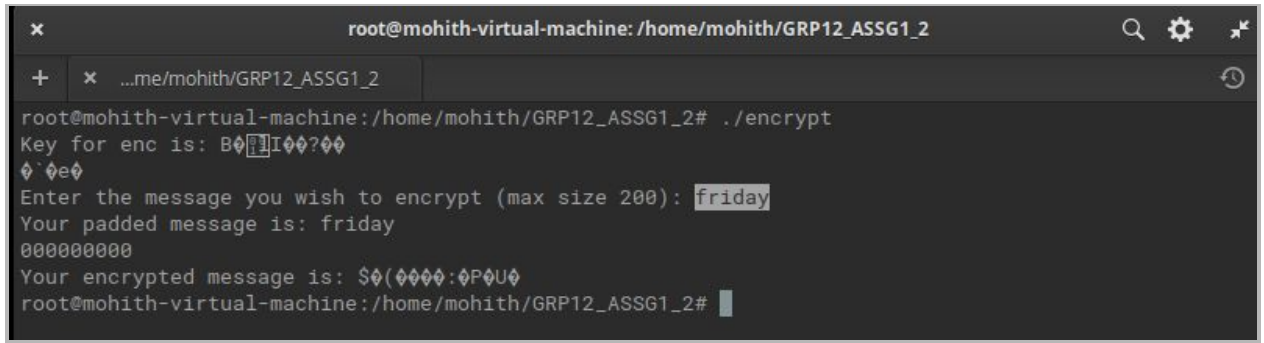
We can see the workflow as shown in the image.

a) For Encryption

Input :

Input that must be given by the user is the message. The maximum length of the message is set to 200. As 0 is used for padding, 0 cannot be part of the message.

`$./encrypt`



```
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2
+ x ...me/mohith/GRP12_ASSG1_2
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2# ./encrypt
Key for enc is: B0I00?00
0`0e0
Enter the message you wish to encrypt (max size 200): friday
Your padded message is: friday
000000000
Your encrypted message is: $0(0000:0P0U0
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2#
```

fig(x): input(encrypted)

Output:

The user will see the following output for the encryption user space program:

- *The Cipher Key*
- *Padded Message*
- *Encrypted Message.*

The kernel log will have:

- Device registration,
- Successful/failed encryption
- Successful/failed reading from user space

b) For decryption

Input : None

`$./decrypt`

```
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2# ./encrypt
Key for enc is: B0I?
Enter the message you wish to encrypt (max size 200): friday
Your padded message is: friday
00000000
Your encrypted message is: S(PP:PU
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2# ./decrypt
Key that was used for decryption is: B0I?
Your Encrypted message is: S(PP:PU
Your Decrypted message is: friday
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2#
```

fig(xi): output(decrypted)

Output:

The user will see the following output for the decryption user space program:

The cipher key, encrypted message, decrypted message.

The Kernel Log now will contain:

- Device Registration
- Successful/failed decryption
- Successful/failed reading from user space

9. We have successfully tested the functionality and correctness of the Character Driver we had written.

4. API's involved:

1. `int __register_chrdev(unsigned int major, unsigned int baseminor, unsigned int count, const char * name, const struct file_operations * fops);`
//Register and create the cdev occupying the region described by *major*, *baseminor* and *count*
2. `void __unregister_chrdev (unsigned int major, unsigned int baseminor, unsigned int count, const char * name);`
//Unregister and destroy the cdev occupying the region described by *major*,

baseminor and count.

3. `unsigned long __copy_from_user(void * to, const void __user * from, unsigned long n);`
//Copy data from user space to kernel space. Caller must check the specified block with access_ok before calling this function.Returns number of bytes that could not be copied. On success, this will be zero.
4. `unsigned long __copy_to_user(void __user * to, const void * from, unsigned long n);`
//Copy data from kernel space to user space. Caller must check the specified block with access_ok before calling this function. Returns number of bytes that could not be copied. On success, this will be zero.

5. Screenshots

```
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2# ls
decdev.c  encdev.c  Makefile  README.md  run.sh  setup.sh  test_dec.c  test_enc.c  writeup.txt
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2#
```

```
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2
+ x ...me/mohith/GRP12_ASSG1_2
obj-m+=encdev.o
obj-m+=decdev.o

all: test
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
    rm encrypt
    rm decrypt
    rm encryption.enc
    rm key
test: test_enc.c test_dec.c
    gcc -o encrypt test_enc.c
    gcc -o decrypt test_dec.c
```

```
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2
+ x ...me/mohith/GRP12_ASSG1_2
mknod /dev/encdev c 184 1
mknod /dev/decdev c 185 1
make all
```



```
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2
+ x ...me/mohith/GRP12_ASSG1_2
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2# ./setup.sh
mknod: /dev/encdev: File exists
mknod: /dev/decdev: File exists
gcc -o encrypt test_enc.c
gcc -o decrypt test_dec.c
make -C /lib/modules/5.8.14/build M=/home/mohith/GRP12_ASSG1_2 modules
make[1]: Entering directory '/home/mohith/linux-5.8.14'
  CC [M] /home/mohith/GRP12_ASSG1_2/encdev.o
  CC [M] /home/mohith/GRP12_ASSG1_2/decdev.o
  MODPOST /home/mohith/GRP12_ASSG1_2/Module.symvers
  CC [M] /home/mohith/GRP12_ASSG1_2/decdev.mod.o
  LD [M] /home/mohith/GRP12_ASSG1_2/decdev.ko
  CC [M] /home/mohith/GRP12_ASSG1_2/encdev.mod.o
  LD [M] /home/mohith/GRP12_ASSG1_2/encdev.ko
make[1]: Leaving directory '/home/mohith/linux-5.8.14'
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2#
```

```
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2
+ x ...me/mohith/GRP12_ASSG1_2
rmmod encdev
rmmod decdev
make all
insmod encdev.ko
insmod decdev.ko
~
~
~
```

```
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2# ./run.sh
gcc -o encrypt test_enc.c
gcc -o decrypt test_dec.c
make -C /lib/modules/5.8.14/build M=/home/mohith/GRP12_ASSG1_2 modules
make[1]: Entering directory '/home/mohith/linux-5.8.14'
make[1]: Leaving directory '/home/mohith/linux-5.8.14'
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2#
```

```
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2
+ x ...me/mohith/GRP12_ASSG1_2
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2# ./encrypt
Key for enc is: B0I?
e
Enter the message you wish to encrypt (max size 200): friday
Your padded message is: friday
00000000
Your encrypted message is: $(P
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2#
```

```
root@mohith-virtual-machine: /home/mohith/GRP12_ASSG1_2
+ x ...me/mohith/GRP12_ASSG1_2
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2# ./encrypt
Key for enc is: B0[?]I??
? ?e?
Enter the message you wish to encrypt (max size 200): friday
Your padded message is: friday
000000000
Your encrypted message is: S0(????:P0U0
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2# ./decrypt
Key that was used for decryption is: B0[?]I??
? ?e?
Your Encrypted message is: S0(????:P0U0
Your Decrypted message is: friday
root@mohith-virtual-machine:/home/mohith/GRP12_ASSG1_2#
```

6. References

1. https://www.youtube.com/channel/UCQ-NwyLyw_-FUQrvXmyW_BA
2. <https://tldp.org/LDP/lkmpg/2.6/html/x569.html>