# Simple Caesar Cipher Driver

Github repo: https://github.com/ftp24/Char-Driver

**Group No: 03**
**Group Members**

| | |
|---|---|
| 1. Jesvin Sebastian Madona | B190700CS |
| 2. Naveen S D | B190707CS |
| 3. Joseph Mani Jacob Mani | B190529CS |
| 4. Lenoah Chacko | B190657CS |
| 5. Varun Chittezhath Anilkumar | B190621CS |

## Application Programming Interface

| Library Function | Function Code | Argument 1 | Argument 2 | Argument 3 | Argument 4 | Return Value |
|---|---|---|---|---|---|---|
| init_module | - | - | - | - | - | 0 for Success |
| | | | | | | <0 for Failure |
| device_open | open | inode value | file descriptor | - | - | 0 for Success |
| | | | | | | -EBUSY for Failure |
| device_write | write | file descriptor | buffer | buffer size | File offset | length of string for Success |
| | | | | | | - |
| device_read | read | file descriptor | buffer | buffer size | File offset | bytes of string for Success |
| | | | | | | 0 for failure |
| device_release | close | file descriptor | buffer | buffer size | File offset | 0 for Success |
| cleanup_module | - | - | - | - | - | - |

# Simple Caesar Cipher Driver

## Problem Statement

Create a simple device driver for a character device for the compiled kernel and test it with a sample application program.

## Methodology

### Char driver

A character device driver is used for character devices which generally transfers data in a stream of bytes.Unlike block devices, character devices do not allow seeks back and forth through the data. In linux character devices are accessed as files through /dev/<filename>

### Caesar Cipher

A Caesar cipher is a basic traditional shift or additive cipher where each alphabet is mapped to its index and added with a particular key to encrypt and subtracted with the same key to decrypt. The key we have chosen is 1.
For example => encryption of ABCZ would be BCDA (A->B, B->C, C->D, Z->A)
decryption of BCDA would be ABCZ

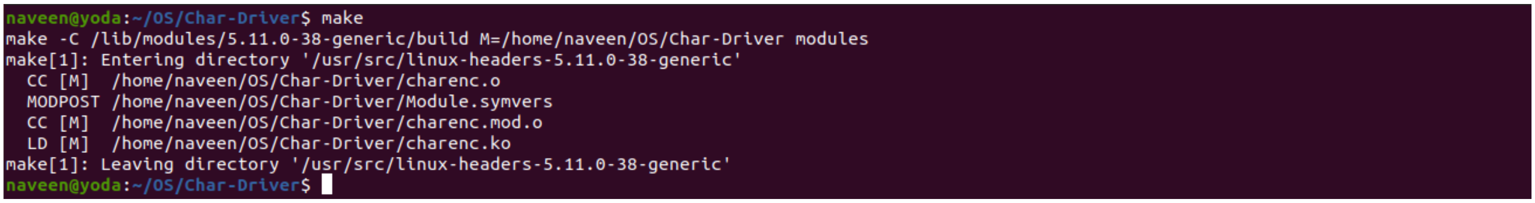
## Explanation

- A Caesar cipher character device driver and a device to work with the linux kernel 5.11.0
- We have written a C program to make a driver with our specification and to obtain a free major number so that we could assign it to a new character device to work with.
- The **init module** will be called when the driver is loaded.
- On loading the driver to the linux kernel a message containing the **major number** will be printed to the log.
- This major number is used to create the device.
- All the commands required to run the driver are mentioned below in the doc. They are also attached in the README.md file.
- Once the device is created we can access the device using the **API specifications** mentioned in the above table.
- The API calls include

    - device_open , open
    - device_write , write
    - device_read , read
    - device_release , close
- **open** - Opens the character device. this will give the access of that particular device to the user program that called it.
- **write** - Obtains the text that is to be entered into the device as an argument and inserts it. We also encrypt the text with caesar cypher before writing into the device.
- **read** - Reads the contents of the device. This will be the encrypted text.
- **close** - This will decrypt the text and close the character device.
- **init_module** - This function is called when we load the driver into the kernel.
- **cleanup_module** - This function is called when the driver is removed from the kernel.

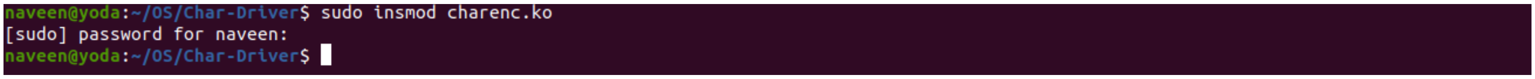
## To run our Driver use the following commands

- make
  - To compile the driver code



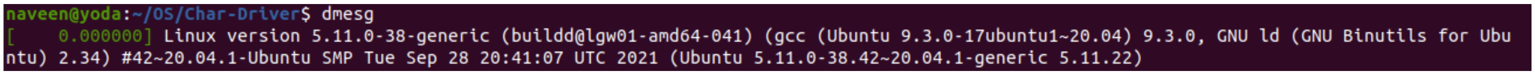
```
naveen@yoda:~/OS/Char-Driver$ make
make -C /lib/modules/5.11.0-38-generic/build M=/home/naveen/OS/Char-Driver modules
make[1]: Entering directory '/usr/src/linux-headers-5.11.0-38-generic'
  CC [M]  /home/naveen/OS/Char-Driver/charenc.o
  MODPOST /home/naveen/OS/Char-Driver/Module.symvers
  CC [M]  /home/naveen/OS/Char-Driver/charenc.mod.o
  LD [M]  /home/naveen/OS/Char-Driver/charenc.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.11.0-38-generic'
naveen@yoda:~/OS/Char-Driver$
```

- insmod charenc.ko
  - To load the driver to the kernel



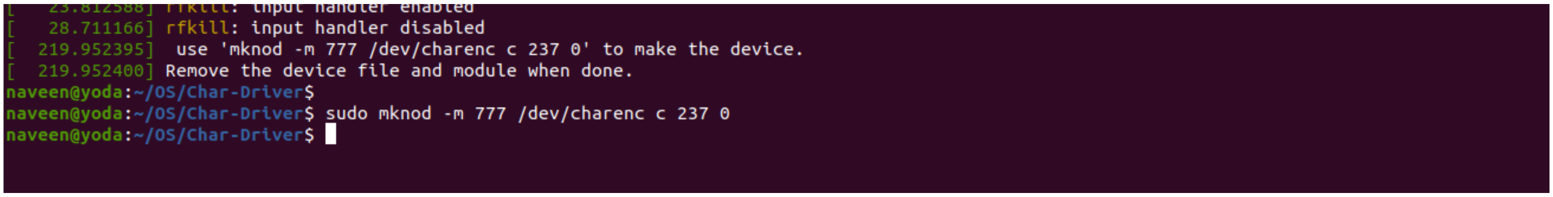
```
naveen@yoda:~/OS/Char-Driver$ sudo insmod charenc.ko
[sudo] password for naveen:
naveen@yoda:~/OS/Char-Driver$
```

- dmesg
  - print or control the kernel message buffer



```
naveen@yoda:~/OS/Char-Driver$ dmesg
[    0.000000] Linux version 5.11.0-38-generic (buildd@lgw01-amd64-041) (gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0, GNU ld (GNU Binutils for Ubu
ntu) 2.34) #42~20.04.1-Ubuntu SMP Tue Sep 28 20:41:07 UTC 2021 (Ubuntu 5.11.0-38.42~20.04.1-generic 5.11.22)
```

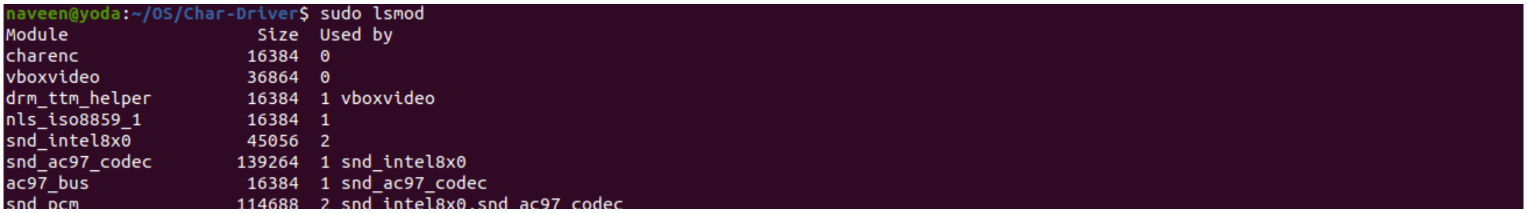
- mknod -m 777 /dev/charenc c <major-number> <minor-number>
  - This creates a device file with read write and execute permission



```
[   23.812588] rfkill: input handler enabled
[   28.711166] rfkill: input handler disabled
[  219.952395]  use 'mknod -m 777 /dev/charenc c 237 0' to make the device.
[  219.952400] Remove the device file and module when done.
naveen@yoda:~/OS/Char-Driver$
naveen@yoda:~/OS/Char-Driver$ sudo mknod -m 777 /dev/charenc c 237 0
naveen@yoda:~/OS/Char-Driver$
```

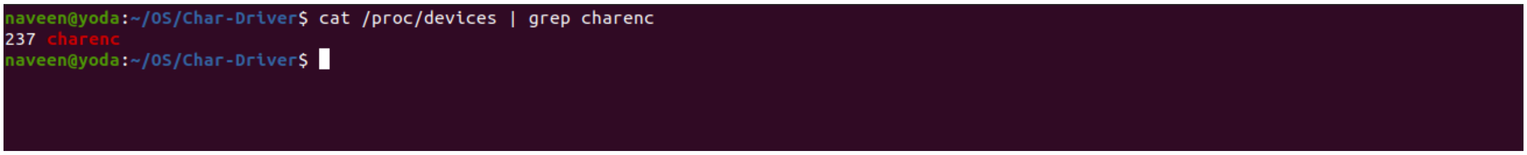
## Existency check for driver and device

- lsmod
    - Displays all the drivers loaded to the kernel



```
naveen@yoda:~/OS/Char-Driver$ sudo lsmod
Module                  Size  Used by
charenc                16384  0
vboxvideo              36864  0
drm_ttm_helper         16384  1 vboxvideo
nls_iso8859_1          16384  1
snd_intel8x0           45056  2
snd_ac97_codec        139264  1 snd_intel8x0
ac97_bus               16384  1 snd_ac97_codec
snd_pcm               114688  2 snd_intel8x0,snd_ac97_codec
```

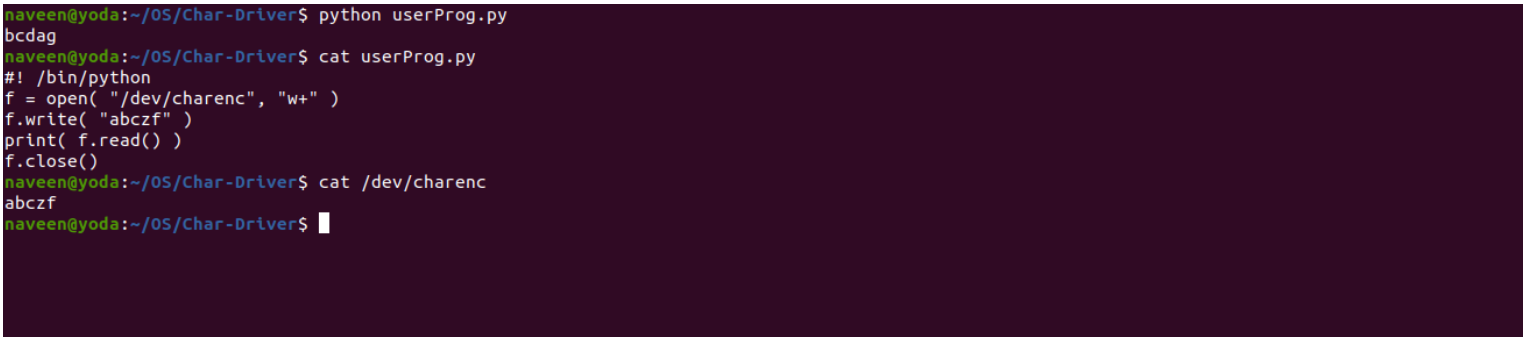
- cat /proc/devices
    - Displays all the connected devices



```
naveen@yoda:~/OS/Char-Driver$ cat /proc/devices | grep charenc
237 charenc
naveen@yoda:~/OS/Char-Driver$ 
```

## User process

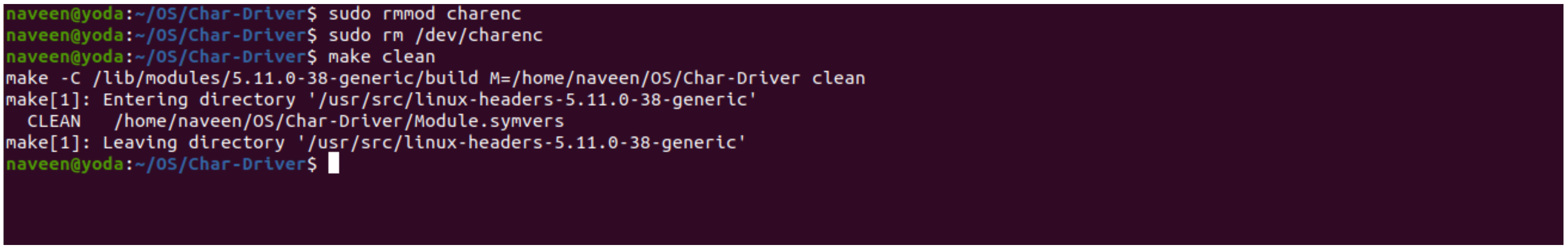
- python userProg.py
    - Opens the device file and writes "abczf" into the device file
    - Prints the encrypted text
    - Closes the file
- cat userProg.py
    - Displays the source code
- cat /dev/charenc
    - Shows decrypted text (ie. after closing the file)



```
naveen@yoda:~/OS/Char-Driver$ python userProg.py
bcdag
naveen@yoda:~/OS/Char-Driver$ cat userProg.py
#! /bin/python
f = open( "/dev/charenc", "w+" )
f.write( "abczf" )
print( f.read() )
f.close()
naveen@yoda:~/OS/Char-Driver$ cat /dev/charenc
abczf
naveen@yoda:~/OS/Char-Driver$ 
```

## Cleanup

- rmmod charenc
  - Unloads the driver
- rm /dev/charenc
  - Remove the device



```
naveen@yoda:~/OS/Char-Driver$ sudo rmmod charenc
naveen@yoda:~/OS/Char-Driver$ sudo rm /dev/charenc
naveen@yoda:~/OS/Char-Driver$ make clean
make -C /lib/modules/5.11.0-38-generic/build M=/home/naveen/OS/Char-Driver clean
make[1]: Entering directory '/usr/src/linux-headers-5.11.0-38-generic'
  CLEAN   /home/naveen/OS/Char-Driver/Module.symvers
make[1]: Leaving directory '/usr/src/linux-headers-5.11.0-38-generic'
naveen@yoda:~/OS/Char-Driver$ 
```