02 Block Driver

What to Expect

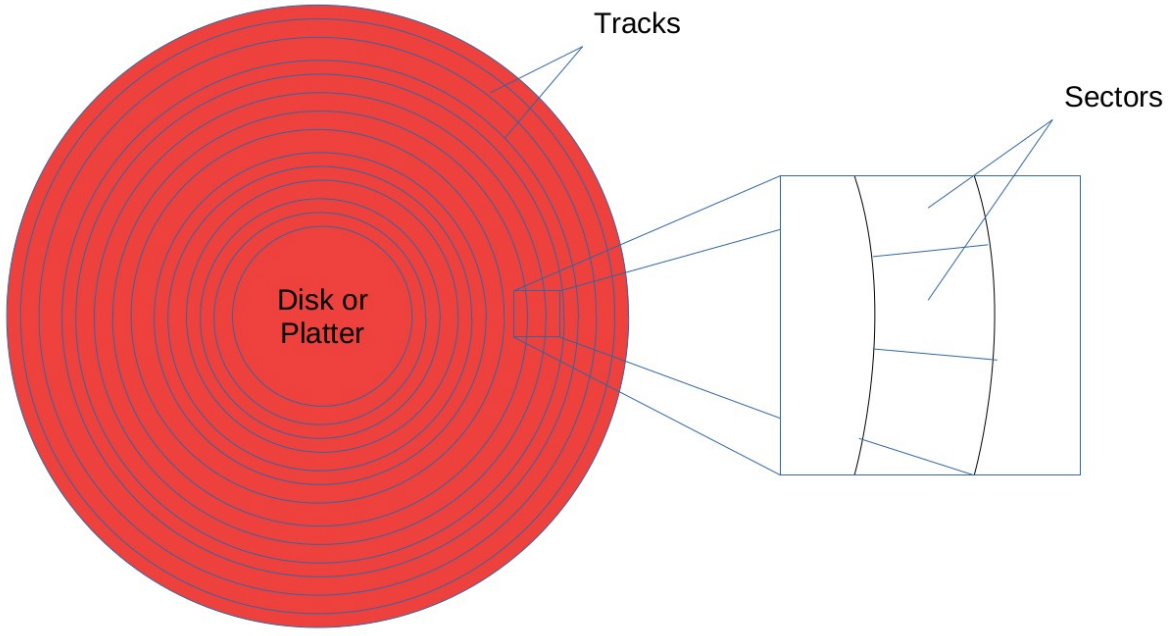
- Why the need for the Block Layer?
- Decoding a Block Device in Linux
- Role of Block Drivers
- Writing a Block Driver

```
[root@BeagleBone ~]# fdisk -l /dev/mmcblk1
Disk /dev/mmcblk1: 3648 MB, 3825205248 bytes, 7471104 sectors
116736 cylinders, 4 heads, 16 sectors/track
Units: sectors of 1 * 512 = 512 bytes

Device       Boot StartCHS    EndCHS        StartLBA     EndLBA    Sectors  Size Id Type
/dev/mmcblk1p1 *  128,0,1     1023,3,16         8192    7471103    7462912 3644M 83 Linux
[root@BeagleBone ~]# fdisk -l /dev/mmcblk0
Disk /dev/mmcblk0: 15 GB, 16106127360 bytes, 31457280 sectors
1958 cylinders, 255 heads, 63 sectors/track
Units: sectors of 1 * 512 = 512 bytes

Device       Boot StartCHS    EndCHS        StartLBA     EndLBA    Sectors  Size Id Type
/dev/mmcblk0p1 *  0,32,33     1023,254,63       2048   31457279   31455232 14.9G  c Win95 FAT32 (LBA)
[root@BeagleBone ~]#
```

The Generic Hard Disk

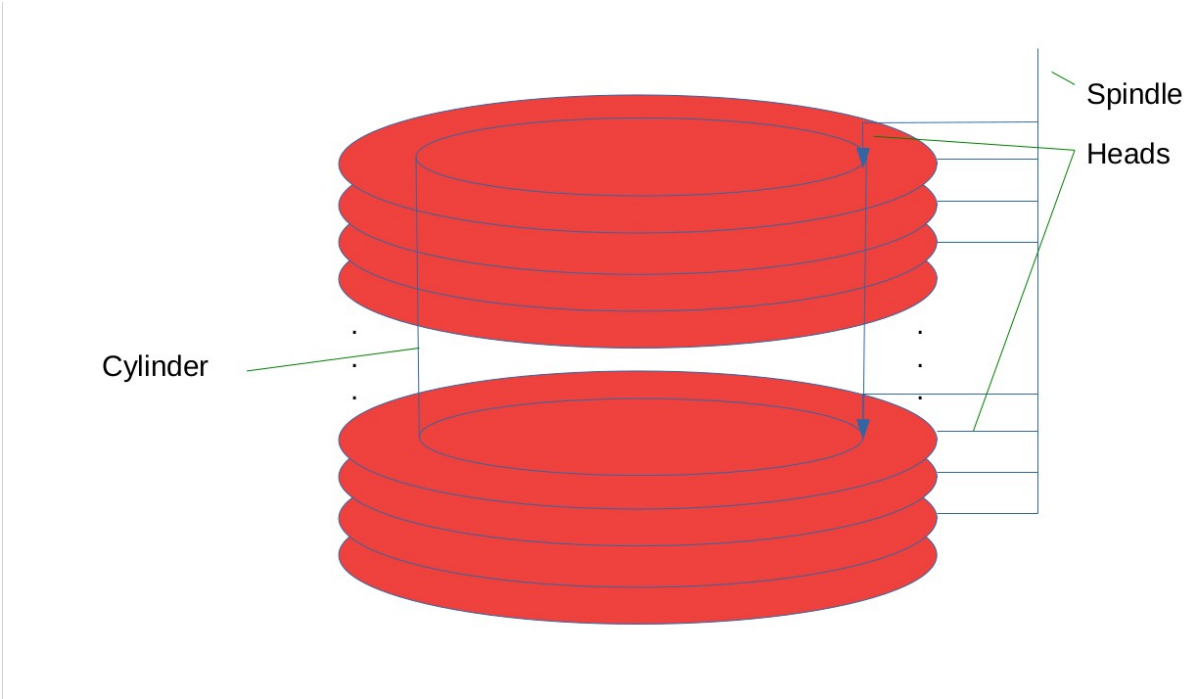


Consenctic rings they are basically track. These circular tracts basically devide into sectors.

Every track having something sectors.
Size of sector -: 512 bytes

You will find upto 63 sectors.

The General Hard Disc



Computing a Generic Hard Disk

- **Example (Hard Disk)**
- **Heads (or Platters): 0 – 9**
- **Tracks (or Cylinders): 0 – 24**
- **Sectors: 1 – 64**

- **Size of the Hard Disk**
- **10 x 25 x 64 x 512 bytes =81920000 8000KiB**

- **Device independent numbering**
- **(h, t, s) → 64 * (10 * t + h) + s → (1 – 16000)**

How many byte per cylinder?

```
[root@BeagleBone ~]# fdisk -l /dev/mmcblk0
Disk /dev/mmcblk0: 15 GB, 16106127360 bytes, 31457280 sectors
1958 cylinders, 255 heads, 63 sectors/track
Units: sectors of 1 * 512 = 512 bytes

Device       Boot StartCHS    EndCHS        StartLBA     EndLBA    Sectors  Size Id Type
/dev/mmcblk0p1 *  0,32,33     1023,254,63       2048   31457279   31455232 14.9G  c Win95 FAT32 (LBA)
[root@BeagleBone ~]# fdisk -l /dev/mmcblk1
Disk /dev/mmcblk1: 3648 MB, 3825205248 bytes, 7471104 sectors
116736 cylinders, 4 heads, 16 sectors/track
Units: sectors of 1 * 512 = 512 bytes

Device       Boot StartCHS    EndCHS        StartLBA     EndLBA    Sectors  Size Id Type
/dev/mmcblk1p1 *  128,0,1     1023,3,16         8192    7471103    7462912 3644M 83 Linux
[root@BeagleBone ~]#
```

1 sector = 512 bytes

Number of sector	**x**	**Size of sector**
63	**x**	**512**

63 x 512 x 255(disc) = 8225280

Partition & Partion Table
==========

- Divides the hdd into one or more logical disks
- called partitions
- Helpful in organizing different types of data
  – Different operating systems data
  – User data
  – Temporary data
  – ...
- Logical division & so need to be maintained by
- metadata – Partition table

Excersise-:
=======

Assignment #01

a. Use dd to create a 1MiB file named disk.
(Hint: You may use /dev/zero or /dev/urandom)
b. Use fdisk to set the heads to 4, sectors to 32, and cylinders to the appropriate number, for disk.
c. Use fdisk to create more than 4 primary partitions, or more than 2 extended partitions.
d. Use fdisk to create the following partitions using the complete disk without gaps:
+ 1 P (FAT32 (LBA)), 1 E, 1 P (Linux)
+ 2 L (Linux swap, Linux)
e. Display the disk partitions using p of "fdisk disk" in both sectors & cylinders (unit) display mode (use u).

Heads Up: Avoid using sudo in any of the above experiments, as you really do not need it.

1. **dd if=/dev/zero of=rd.img bs=512 count=2048**

#P|L | L|P

2. fdisk rd.img

```
infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$ fdisk rd.img

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x73912db8.

Command (m for help): p
Disk rd.img: 1 MiB, 1048576 bytes, 2048 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x73912db8

Command (m for help):
```

disk_size = nhds * secs * cyl * size of sector
1Mib = 4 *32 (512)* cycl
1Mib = 65536 * cyc
1024/65536 = cyl
cyl =16

1. Use dd to create a 1MiB file named disk.
(Hint: You may use /dev/zero or /dev/urandom)

infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$ **dd if=/dev/zero of=rd.img bs=512 count=2048**
2048+0 records in
2048+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00472885 s, 222 MB/s

```
infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$ dd if=/dev/zero of=rd.img bs=512 count=2048
2048+0 records in
2048+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00472885 s, 222 MB/s
infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$ ls -l rd.img
-rw-rw-r-- 1 infinite infinite 1048576 Jan 31 17:49 rd.img
infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$ fdisk rd.img

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x73912db8.

Command (m for help): p
Disk rd.img: 1 MiB, 1048576 bytes, 2048 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x73912db8

Command (m for help): x

Expert command (m for help): h
Number of heads (1-255, default 255): 4

Expert command (m for help): s
Number of sectors (1-63, default 63): 32

Expert command (m for help): c
Number of cylinders (1-1048576, default 16): 16

Expert command (m for help): w
w: unknown command

Expert command (m for help): r

Command (m for help): w

The partition table has been altered.
Syncing disks.

infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$
```

Create 2nd Partition Extended

```
Device     Boot Start End Cylinders   Size Id Type
rd.img1             1   5         5 319.5K 83 Linux

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): e
Partition number (2-4, default 2):
First cylinder (6-16, default 6):
Last cylinder, +/-cylinders or +/-size{K,M,G,T,P} (6-16, default 16): +512
Value out of range.
Last cylinder, +/-cylinders or +/-size{K,M,G,T,P} (6-16, default 16): +512K

Created a new partition 2 of type 'Extended' and of size 576 KiB.

Command (m for help): p
Disk rd.img: 1 MiB, 1048576 bytes, 2048 sectors
Geometry: 4 heads, 32 sectors/track, 16 cylinders
Units: cylinders of 128 * 512 = 65536 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x73912db8

Device     Boot Start End Cylinders   Size Id Type
rd.img1             1   5         5 319.5K 83 Linux
rd.img2             6  14        10   576K  5 Extended

Command (m for help): u
Changing display/entry units to sectors.
```

3. Logical Partition (2 Logical Partition)

```
Device     Boot Start End Cylinders   Size Id Type
rd.img1             1   5         5 319.5K 83 Linux
rd.img2             6  14        10   576K  5 Extended

Command (m for help): n
Partition type
   p   primary (1 primary, 1 extended, 2 free)
   l   logical (numbered from 5)
Select (default p): l

Adding logical partition 5
First cylinder (6-14, default 6):
Last cylinder, +/-cylinders or +/-size{K,M,G,T,P} (6-14, default 14): +256K

Created a new partition 5 of type 'Linux' and of size 319.5 KiB.

Command (m for help): n
Partition type
   p   primary (1 primary, 1 extended, 2 free)
   l   logical (numbered from 5)
Select (default p): l

Adding logical partition 6
First cylinder (11-14, default 11):
Last cylinder, +/-cylinders or +/-size{K,M,G,T,P} (11-14, default 14):

Created a new partition 6 of type 'Linux' and of size 255.5 KiB.

Command (m for help): p
Disk rd.img: 1 MiB, 1048576 bytes, 2048 sectors
Geometry: 4 heads, 32 sectors/track, 16 cylinders
Units: cylinders of 128 * 512 = 65536 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x73912db8

Device     Boot Start End Cylinders   Size Id Type
rd.img1             1   5         5 319.5K 83 Linux
rd.img2             6  14        10   576K  5 Extended
rd.img5             6  10         5 319.5K 83 Linux
rd.img6            11  14         4 255.5K 83 Linux
```

4. Now Primary Partiotion

```
Command (m for help): p
Disk rd.img: 1 MiB, 1048576 bytes, 2048 sectors
Geometry: 4 heads, 32 sectors/track, 16 cylinders
Units: cylinders of 128 * 512 = 65536 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x73912db8

Device     Boot Start End Cylinders   Size Id Type
rd.img1             1   5         5 319.5K 83 Linux
rd.img2             6  14        10   576K  5 Extended
rd.img5             6  10         5 319.5K 83 Linux
rd.img6            11  14         4 255.5K 83 Linux

Command (m for help): n
Partition type
   p   primary (1 primary, 1 extended, 2 free)
   l   logical (numbered from 5)
Select (default p): p
Partition number (3,4, default 3):
First cylinder (15-16, default 15):
Last cylinder, +/-cylinders or +/-size{K,M,G,T,P} (15-16, default 16):

Created a new partition 3 of type 'Linux' and of size 128 KiB.

Command (m for help): p
Disk rd.img: 1 MiB, 1048576 bytes, 2048 sectors
Geometry: 4 heads, 32 sectors/track, 16 cylinders
Units: cylinders of 128 * 512 = 65536 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x73912db8

Device     Boot Start End Cylinders   Size Id Type
rd.img1             1   5         5 319.5K 83 Linux
rd.img2             6  14        10   576K  5 Extended
rd.img3            15  16         3   128K 83 Linux
rd.img5             6  10         5 319.5K 83 Linux
rd.img6            11  14         4 255.5K 83 Linux
```

Last Primary Partition -: w for save and sync

```
Disk rd.img: 1 MiB, 1048576 bytes, 2048 sectors
Geometry: 4 heads, 32 sectors/track, 16 cylinders
Units: cylinders of 128 * 512 = 65536 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x73912db8

Device     Boot Start End Cylinders   Size Id Type
rd.img1             1   5         5 319.5K 83 Linux
rd.img2             6  14        10   576K  5 Extended
rd.img5             6  10         5 319.5K 83 Linux
rd.img6            11  14         4 255.5K 83 Linux

Command (m for help): n
Partition type
   p   primary (1 primary, 1 extended, 2 free)
   l   logical (numbered from 5)
Select (default p): p
Partition number (3,4, default 3):
First cylinder (15-16, default 15):
Last cylinder, +/-cylinders or +/-size{K,M,G,T,P} (15-16, default 16):

Created a new partition 3 of type 'Linux' and of size 128 KiB.

Command (m for help): p
Disk rd.img: 1 MiB, 1048576 bytes, 2048 sectors
Geometry: 4 heads, 32 sectors/track, 16 cylinders
Units: cylinders of 128 * 512 = 65536 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x73912db8

Device     Boot Start End Cylinders   Size Id Type
rd.img1             1   5         5 319.5K 83 Linux
rd.img2             6  14        10   576K  5 Extended
rd.img3            15  16         3   128K 83 Linux
rd.img5             6  10         5 319.5K 83 Linux
rd.img6            11  14         4 255.5K 83 Linux

Partition table entries are not in disk order.
```

How to check h,s,c wise like Linux?
Ans -:

make part_info

./part_info rd.img

```
infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$ ls
part_info.c  rd.img  steps.txt
infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$ vi part_info.c
infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$ make part_info
cc     part_info.c   -o part_info
infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$ ./part_info rd.img

DOS type Partition Table of rd.img:
  B Start (H/C/S)   End (H/C/S) Type  StartSec    TotSec
1:0 (  0/   1/ 2) (  3/   5/32)  83          1       639
2:0 (  0/   6/ 1) (  3/  14/32)  05        640      1152
3:0 (  0/  15/ 1) (  3/  16/32)  83       1792       256
4:0 (  0/   1/ 0) (  0/   1/ 0)  00          0         0

Re-computed Partition Table of rd.img:
  B Start (H/C/S)   End (H/C/S) Type  StartSec    TotSec
1:0 (  0/    1/2) ( 10/    1/10) 83          1       639
2:0 ( 10/    1/11) ( 28/    1/28) 05        640      1152
3:0 ( 28/    1/29) ( 32/    1/32) 83       1792       256
4:0 (  0/    1/1) ( 89/267350/4) 00          0         0

infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$
```

we need to creat device file to support like

we dont have here seprate for rd.img

ls /*dev* / rd.img1
ls /*dev* / rd.img2
ls /*dev* / rd.img3
ls /*dev* / rd.img4

fdisk able to read rd.img

Not easy to creat file system. Like to store file in that partition we not able to mount.

fdisk is able to read it but filesystem not able to use it

We wants to use a driver so that this driver can show saperate device file to us so we are able to mount and use it .

```
infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$ sudo mount rd.img /mnt/
[sudo] password for infinite:
mount: /mnt: wrong fs type, bad option, bad superblock on /dev/loop31, missing codepage or helper program, or other error.
infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$
```

Why my partition gone?

```
infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$ ls /mnt
b
infinite@annu:~/Linux_Drivers/Block_Driver/BlockFSDriver/Stage1Fdisk$ fdisk rd.img1

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p
Disk rd.img1: 1 MiB, 1048576 bytes, 2048 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000
```

Check file History

0002_Steps_Why_Partition_Gone_After_fdisk.patch

Partition read it fdisk but when I do mkfs i wants something like rd1.img. So i need saperate file so i can create file syatem. The moment i created

We need basically rd1.img , rd2.img like that. On which i creat file syatem. But it was only one partition

File system create data structure data structure and inode table and all info. Whne i do mkfs so it lost mbr and data structure.

So mkfs overwrote evrything.

Need -:

support for block driver which create one saperate create device file on which i can create file system. So with the help of that we can store the data into partition. Thats why Block IO layer comes into picture.

How we can creat rmdisk ;

perticular portion of RAM will creat as disk. We will operate on it. And block driver we called block driver of ramdisk.

 # 1Mib (disk) – Ramdisk Block Driver

Excersise -: End Goal

==========

Block_Driver/BlockFSDriver/Stage1Fdisk

file – dor.ko

file -: rd.img

Transfer into board

```
[root@BeagleBone ~]# ls
bin     dor.ko  rd.img
[root@BeagleBone ~]# insmod dor.ko
[ 6803.544100] rb: Device is opened
[ 6803.547483] rb: Inode number is 0
[ 6803.551852]  rb: rb1 rb2 < rb5 rb6 rb7 > rb3
[ 6803.557728] rb: Device is closed
[ 6803.561631] rb: Ram Block driver initialised (1024 sectors; 524288 bytes)
[root@BeagleBone ~]# ls /dev/r
ls: /dev/r: No such file or directory
[root@BeagleBone ~]# ls /dev/r
ram0    ram11   ram14   ram3    ram6    ram9    rb1     rb5     rfkill
ram1    ram12   ram15   ram4    ram7    random  rb2     rb6     rtc0
ram10   ram13   ram2    ram5    ram8    rb      rb3     rb7
[root@BeagleBone ~]# ls /dev/r
```

We wants to achieve this type of driver at the end.

Partttion on board -:

```
[root@BeagleBone ~]# fdisk -l /dev/rb
[ 6969.693968] rb: Device is opened
[ 6969.697390] rb: Inode number is 0
Disk /dev/rb: 0 MB, 524288 bytes, 1024 sectors[ 6969.703737] rb: Device is closed

32 cylinders, 1 heads, 32 sectors/track
Units: sectors of 1 * 512 = 512 bytes

Device Boot StartCHS    EndCHS        StartLBA     EndLBA    Sectors  Size Id Type
/dev/rb1    0,0,2       9,0,32               1        319        319  159K 83 Linux
/dev/rb2    10,0,1      19,0,32            320        639        320  160K  5 Extended
/dev/rb3    20,0,1      31,0,32            640       1023        384  192K 83 Linux
/dev/rb5    10,0,2      13,0,32            321        447        127 65024 83 Linux
/dev/rb6    14,0,2      17,0,32            449        575        127 65024 83 Linux
/dev/rb7    18,0,2      19,0,32            577        639         63 32256 83 Linux
[root@BeagleBone ~]#
```

320 its end but why its starting from 321 -: store LBR

```
[root@BeagleBone ~]# fdisk -l /dev/rb
[ 6969.693968] rb: Device is opened
[ 6969.697390] rb: Inode number is 0
Disk /dev/rb: 0 MB, 524288 bytes, 1024 sectors[ 6969.703737] rb: Device is closed

32 cylinders, 1 heads, 32 sectors/track
Units: sectors of 1 * 512 = 512 bytes

Device Boot StartCHS    EndCHS        StartLBA     EndLBA    Sectors  Size Id Type
/dev/rb1    0,0,2       9,0,32               1        319        319  159K 83 Linux
/dev/rb2    10,0,1      19,0,32            320        639        320  160K  5 Extended
/dev/rb3    20,0,1      31,0,32            640       1023        384  192K 83 Linux
/dev/rb5    10,0,2      13,0,32            321        447        127 65024 83 Linux
/dev/rb6    14,0,2      17,0,32            449        575        127 65024 83 Linux
/dev/rb7    18,0,2      19,0,32            577        639         63 32256 83 Linux
[root@BeagleBone ~]# dd if=/dev/rb of=rd.img
[ 7168.421843] rb: Device is opened
[ 7168.425262] rb: Inode number is 0
[ 7168.462842] rb: Device is closed
1024+0 records in
1024+0 records out
524288 bytes (512.0KB) copied, 0.044489 seconds, 11.2MB/s
[root@BeagleBone ~]# fdisk rd.img

Segmentation fault
[root@BeagleBone ~]# fdisk rd.img
```

mkfs.vfat /*dev*/rb1
mount /*dev*/rb1 /dev

```
[root@BeagleBone ~]# ls
bin     dor.ko  rd.img
[root@BeagleBone ~]# mkfs.vfat /dev/rb1
[ 7490.329799] rb: Device is opened
[ 7490.333220] rb: Inode number is 0
[ 7490.339677] rb: Device is closed
[root@BeagleBone ~]# mount /dev/rb1 /mnt/
[ 7498.763595] rb: Device is opened
[ 7498.766979] rb: Inode number is 0
[ 7498.771322] rb: Device is closed
[ 7498.775057] rb: Device is opened
[ 7498.778538] rb: Inode number is 0
[ 7498.782443] rb: Device is closed
[ 7498.785763] rb: Device is opened
[ 7498.789312] rb: Inode number is 0
[ 7498.793050] rb: Device is closed
[ 7498.796755] rb: Device is opened
[ 7498.800219] rb: Inode number is 0
[root@BeagleBone ~]# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                15.5M     12.0M      3.4M  78% /
devtmpfs                214.3M         0    214.3M   0% /dev
tmpfs                   246.8M         0    246.8M   0% /dev/shm
tmpfs                   246.8M         0    246.8M   0% /tmp
tmpfs                   246.8M         0    246.8M   0% /run
/dev/rb1                153.5K       512    153.0K   0% /mnt
[root@BeagleBone ~]#
```

#disk on ram -> loaded driver -> disk created on the ram -> unloaded -> ram was released

#loaded – malloc(512 KB) ------>> read/write on ram

path -: BlockFSDriver/Stage2Ramdisk

apis.txt
Makefile
partition.c -: partition layout
partition.h
ram_block.c -: Block driver (Verticals)
ram_device.c -:
ram_device.h steps.txt

files -:

ram_block.c

Todo 1 -:

in char driver
char = regter char driver + register file operation + create device file
block = allocate gendisk + add disk

Having device file without file opertaion no need

block = **allocate gendisk+ rgister fops + add disk**

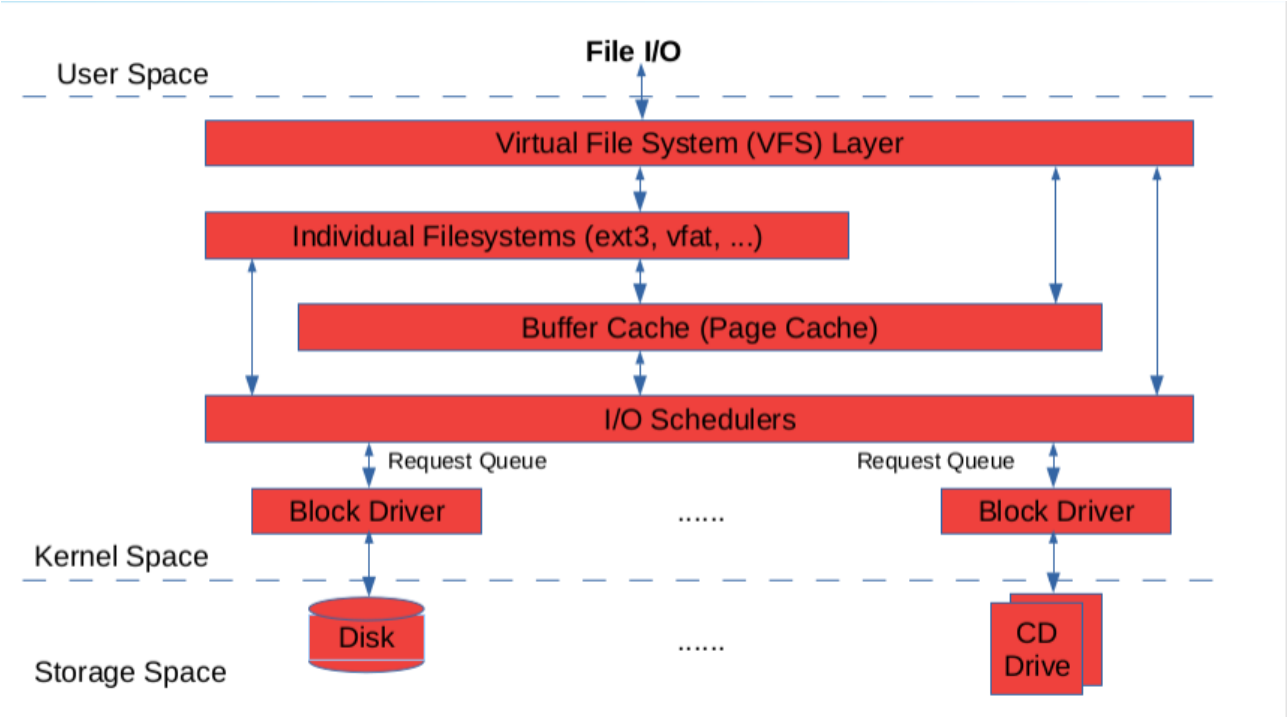
file -: **include/linux/blkdev.h**

```
struct block_device_operations {
        int (*open) (struct block_device *, fmode_t);
        void (*release) (struct gendisk *, fmode_t);
        int (*rw_page)(struct block_device *, sector_t, struct page *, unsigned int);
        int (*ioctl) (struct block_device *, fmode_t, unsigned, unsigned long);
        int (*compat_ioctl) (struct block_device *, fmode_t, unsigned, unsigned long);
        unsigned int (*check_events) (struct gendisk *disk,
                                      unsigned int clearing);
        /* ->media_changed() is DEPRECATED, use ->check_events() instead */
        int (*media_changed) (struct gendisk *);
        void (*unlock_native_capacity) (struct gendisk *);
        int (*revalidate_disk) (struct gendisk *);
        int (*getgeo)(struct block_device *, struct hd_geometry *);
        /* this callback is with swap_lock and sometimes page table lock held */
        void (*swap_slot_free_notify) (struct block_device *, unsigned long);
        struct module *owner;
        const struct pr_ops *pr_ops;
};
```

In files there are not read or write operation.
From userspace we are not exposing read/write system call.

Why?

Bock input /output



input /output

VFS = system driver +block driver
VFS knows which file sytem driver works
when we do read/ write operation it will store in some memory (Buffer cache) for frequesntly data.

When unmount or save file. Like all operation flush on the disk. So Buffer cache send request to Disk. So there are some request.
In char driver like direct read or write.
So i need a queue , so the block driver has queue. So block driver serve keep on processing request que.

bloack driver doesnt have directly interaction with disk(userspace). No direct dealing with read/write operation. W.r.t char driver buffer cache + additional layer not be there

Lets Assume -> request comes

sector 2 to sector N
Sector 10---->>> Sector 5
So 1st serve request no 10 and then request Sector no 5 -: This is inproper
Another senariao
Sector 10 -->> Sector 9-->> Sector 7 -->>Sector 22 -->>Sector 4-->>Sector 8 -->>
So we need like elevator . Select one which you wants.
So that layer we called based on the request come store in I/O Schedulers.

So

block = allocate gendisk + rgister fops + Intialize Queue + add disk

****
****