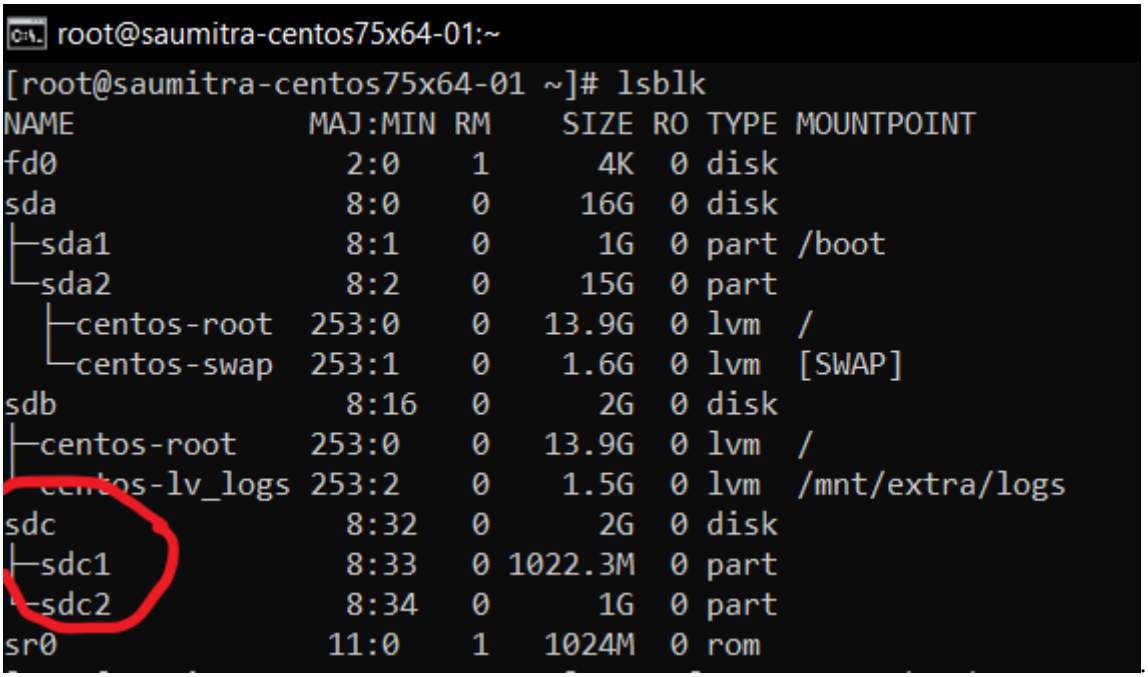
Working with ZFS

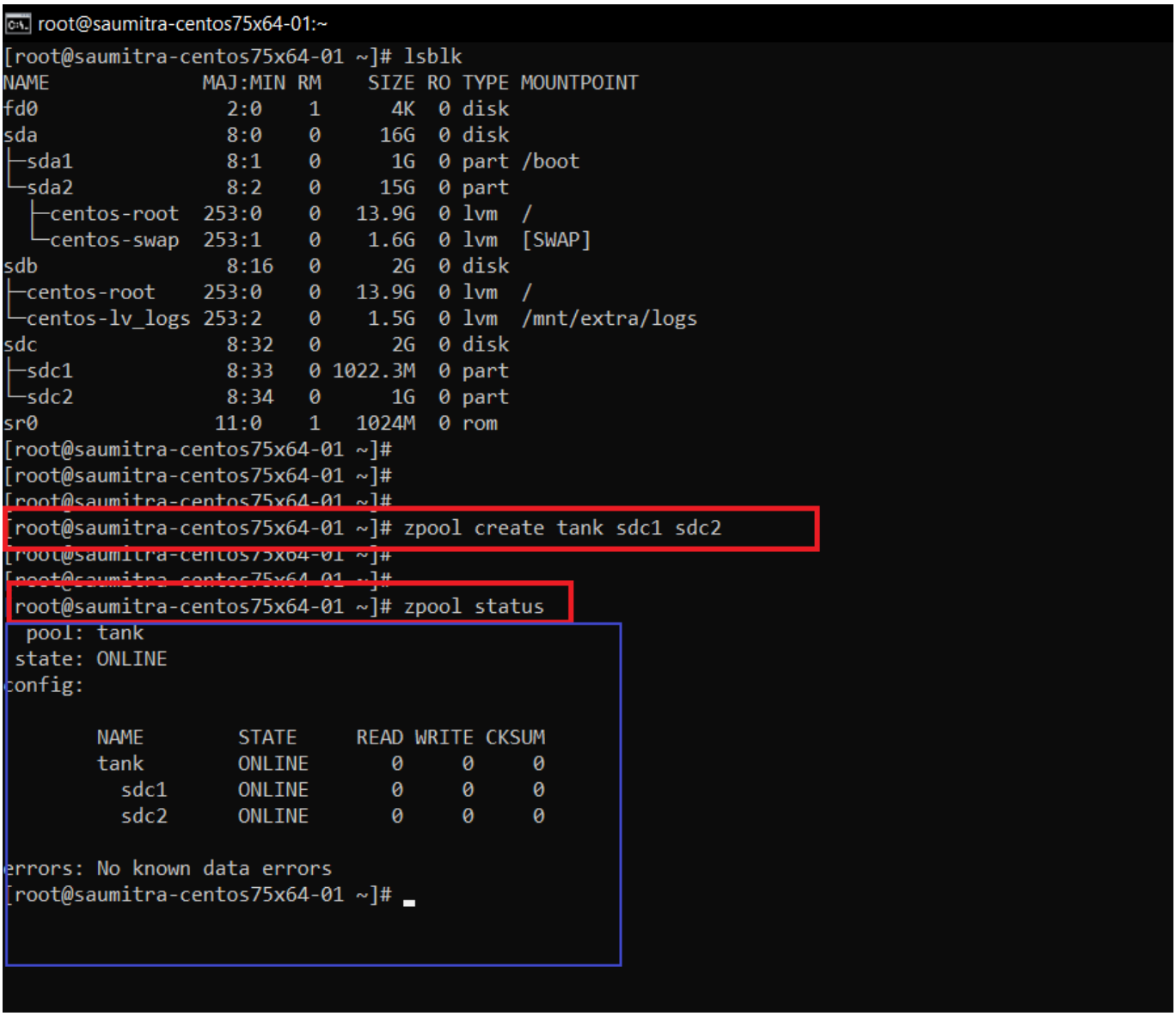
We are tasked with creating a ZFS volume. For that we shall use a blank physical disk `/dev/sdc` with no file system on it. We have made 2 partitions of this disk (`/dev/sdc1` and `/dev/sdc2`).



Creating a Storage Pool

Using ZFS is entirely encompassed in just 2 commands - `zpool` and `zfs`. These two commands manage the entire storage configuration.

We start off by creating a storage pool using the `zpool` command. ZFS can use disks directly, there is no need to create partitions or volumes (unlike LVM). Let's create a storage pool called "tank" from the disks `dev/sdc1` and `/dev/sdc2`. We can query the status of the storage pools using the `zpool status` command.



After creating a storage pool, ZFS will automatically:

- Create a filesystem with the same name (i.e. tank)
- Mount the filesystem under that name (e.g. /tank)

```
[root@saumitra-centos75x64-01 ~]# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime,seclabel)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
devtmpfs on /dev type devtmpfs (rw,nosuid,seclabel,size=929072k,nr_inodes=232268,mode=755)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,seclabel)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,seclabel,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,seclabel,mode=755)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,seclabel,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,xattr,release_agent=/usr/lib/systemd/systemd-cgroup
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,perf_event)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,blkio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,cpuacct,cpu)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,net_prio,net_cls)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,devices)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,memory)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,hugetlb)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,cpuset)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,freezer)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,pids)
configfs on /sys/kernel/config type configfs (rw,relatime)
/dev/mapper/centos-root on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=34,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=12676)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime,seclabel)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,seclabel)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
/dev/mapper/centos-lv_logs on /mnt/extra/logs type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sda1 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,seclabel,size=188240k,mode=700)
tank on /tank type zfs (rw,seclabel,xattr,noacl)
[root@saumitra-centos75x64-01 ~]#
```

Thus, storage is immediately available to us. We are ready to read and write files to it. Its that simple!

Creating a sparse file

A sparse file is a file that has large amounts of space preallocated to it, without occupying that entire space from the filesystem.

For example, lets say that we want to reserve 1GB of space for future use. Now, theoretically we could actually store 1GB worth of empty bytes into that file. Yes, our space is now reserved. But at what cost?

Storing empty bytes is just not efficient. We do know there are many of these bytes in the file, so why store them on the storage device?

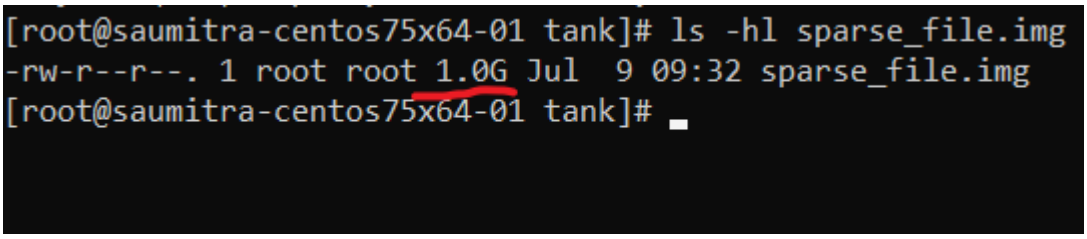
We could instead store metadata describing those zeros.When a process reads the file those zero byte blocks get generated dynamically as opposed to being stored on physical storage.

Thus, our "zero" data is generated *as and when a process reads the file*. Such files are therefore called "Sparse Files".

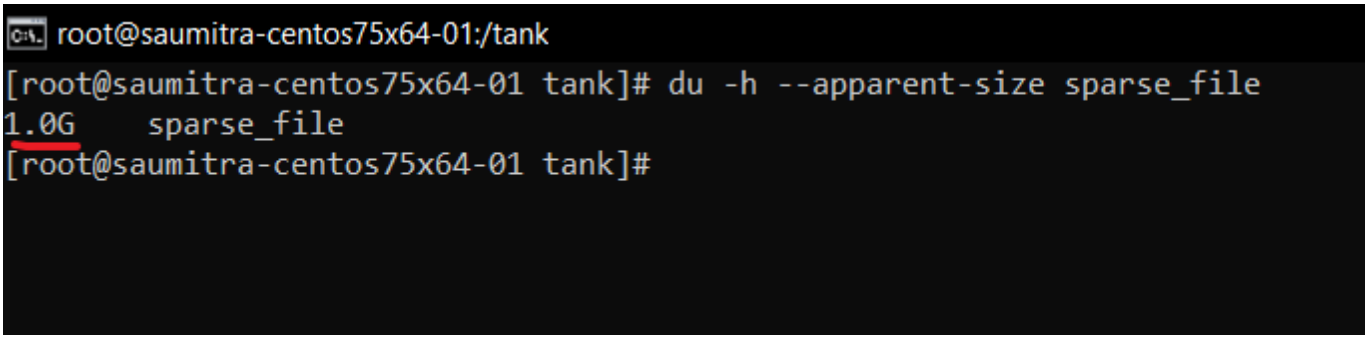
In order to create a sparse file, we use the `dd` command. The `dd` command is used to copy bytes of data from one file to another. We use its `seek` option in order to seek to the amount of space we want (say 1 GB).

```
root@saumitra-centos75x64-01:/tank
[root@saumitra-centos75x64-01 tank]# dd if=/dev/zero of=sparse_file.img bs=1 count=0  seek=1G
0+0 records in
0+0 records out
0 bytes (0 B) copied, 0.000297816 s, 0.0 kB/s
[root@saumitra-centos75x64-01 tank]#
```

In order to check that this file indeed is a sparse file, we can look into the `ls -hl` command.



Another way is to check the `du -h` command with its `apparent-size` option.



Creating a Loop Device from the Sparse file

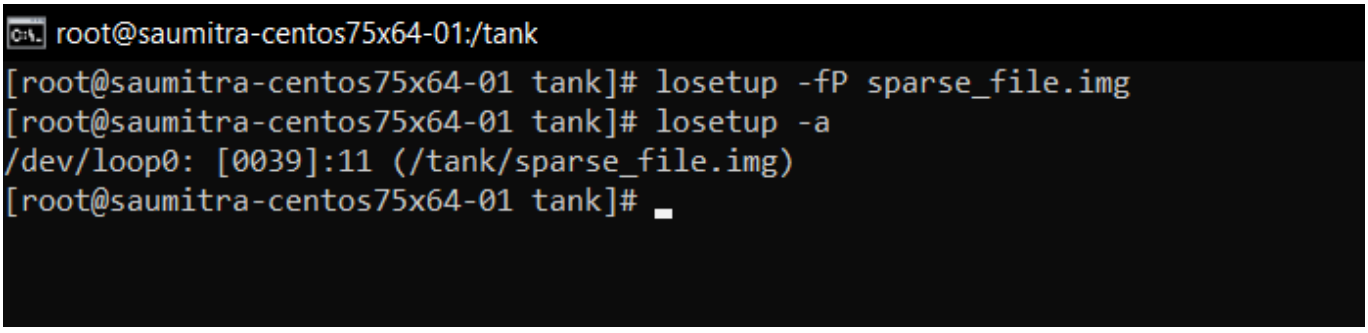
Linux supports a special block device called the loop device, which maps a normal file onto a virtual block device. This allows for the file to be used as a “virtual file system” inside another file. With Linux it’s possible to create a file-system inside a single file.

In order to create a loop device with the sparse file, we use the command `losetup` to create a loop device “loop0”.

Here,

- `-f` implies that linux finds the first unused loop device. If a file argument is present, use this device. Otherwise, print its name.
- `-P` – force kernel to scan partition table on newly created loop device.

To print the loop device generated, we use `losetup -a`.



Creating a filesystem on the loop device

Now lets create a ext4 filesystem on the loopback device. We do this using the command: `mkfs.ext4 /tank/sparse_file.img`

The output looks like:

```
[root@saumitra-centos75x64-01 tank]# mkfs.ext4 /tank/sparse_file.img
mke2fs 1.42.9 (28-Dec-2013)
/tank/sparse_file.img is not a block special device.
Proceed anyway? (y,n) y
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
65536 inodes, 262144 blocks
13107 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

[root@saumitra-centos75x64-01 tank]#
```

Mounting the Loopback Device

We can now mount the loopback filesystem onto a directory. This is done using the `mount` command. The `-o loop` additional option is used to mount loopback filesystems.

```
root@saumitra-centos75x64-01:/tank
[root@saumitra-centos75x64-01 tank]# mkdir /mnt/loopfs
[root@saumitra-centos75x64-01 tank]# mount -o loop /dev/loop0 /mnt/loopfs/
[root@saumitra-centos75x64-01 tank]#
```

We can verify the size of the new mount point and type of filesystem using `df -hP /loopfs/` command.

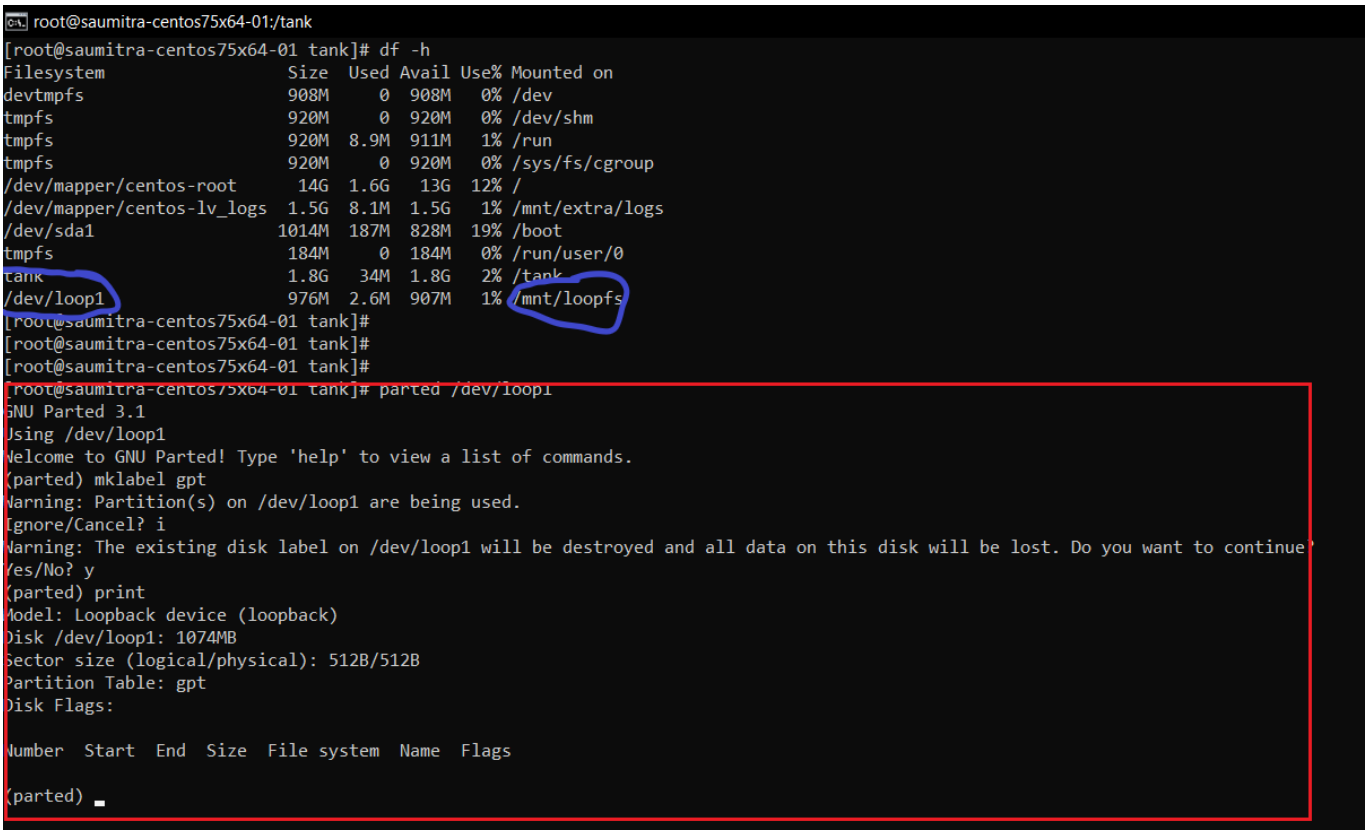
```
[root@saumitra-centos75x64-01 tank]# df -hP /mnt/loopfs/
Filesystem      Size  Used Avail Use% Mounted on
/dev/loop1      976M  2.6M  907M   1% /mnt/loopfs
[root@saumitra-centos75x64-01 tank]#
```

We can also verify that the drive indeed has been mounted using the `mount` command and piping it through the `grep loopfs` command to filter the results.

```
root@saumitra-centos75x64-01:/tank
[root@saumitra-centos75x64-01 tank]# mount |grep loopfs
/dev/loop0 on /mnt/loopfs type ext4 (rw,relatime,seclabel,data=ordered)
```

Partitioning the Mount

In order to partition the mount, we use the `parted` tool in linux. We first specify the volume we wish to partition. Then we make a partition table (generally `gpt`) on the volume.



Let's create 2 partitions of 512 MBs each. This is done using the `mkpart` command:

Partition 1:

```
(parted) mkpart primary ext4 1MB 512MB
(parted) print
Model: Loopback device (loopback)
Disk /dev/loop1: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End    Size   File system  Name     Flags
 1      1049kB  512MB  511MB               primary

(parted)
```

Partition 2:

```
(parted) mkpart primary ext4 512MB 1074MB
(parted) print
Model: Loopback device (loopback)
Disk /dev/loop1: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  512MB   511MB                primary
 2      512MB   1074MB  562MB                primary

(parted)
```

We can see that these partitions are mounted using the `lsblk` command.

```
root@saumitra-centos75x64-01:/tank
[root@saumitra-centos75x64-01 tank]# lsblk
NAME             MAJ:MIN RM    SIZE RO TYPE MOUNTPOINT
fd0                2:0    1      4K  0 disk
sda                8:0    0     16G  0 disk
├─sda1             8:1    0      1G  0 part /boot
└─sda2             8:2    0     15G  0 part
  ├─centos-root  253:0    0   13.9G  0 lvm  /
  └─centos-swap  253:1    0    1.6G  0 lvm  [SWAP]
sdb                8:16   0      2G  0 disk
├─centos-root    253:0    0   13.9G  0 lvm  /
└─centos-lv_logs 253:2    0    1.5G  0 lvm  /mnt/extra/logs
sdc                8:32   0      2G  0 disk
├─sdc1             8:33   0 1022.3M  0 part
└─sdc2             8:34   0      1G  0 part
sr0               11:0    1   1024M  0 rom
loop0              7:0    0      1G  0 loop
loop1              7:1    0      1G  0 loop
├─loop1p1        259:0    0    487M  0 loop
└─loop1p2        259:1    0    536M  0 loop
[root@saumitra-centos75x64-01 tank]#
```

References

1. https://youtu.be/Hjpqa_kjCOI
2. https://www.thegeekdiary.com/how-to-create-virtual-block-device-loop-device-filesystem-in-linux/
3. https://phoenixnap.com/kb/linux-create-partition