Operating System

Lab – 10

Objectives:

1. Understanding the concept of mount point in Linux, and mounting and unmounting file systems
2. Understanding the structure of UNIX file system

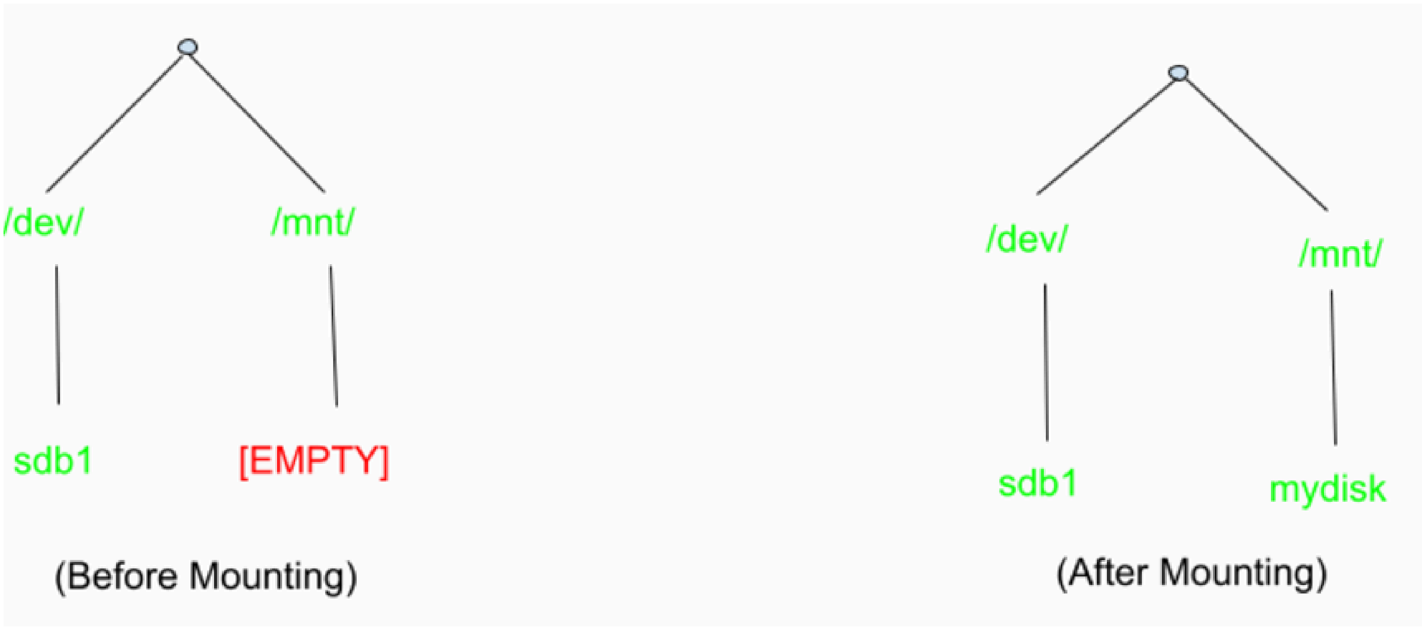
Resources:

- Video Lecture 19: https://youtu.be/HootdM9BgZw?si=5RbMCD1YleYj1rXp
- Video Lecture 20: https://youtu.be/58WJZbcNj2E?si=u1lSvl-0g14vCjxT

Task 1:

- **Describe what do you mean be mount point by drawing a diagram of a tree showing mounted and unmounted file system.**

  Mount POint: It is an empty directory in which we are adding the file system during the process of mounting.

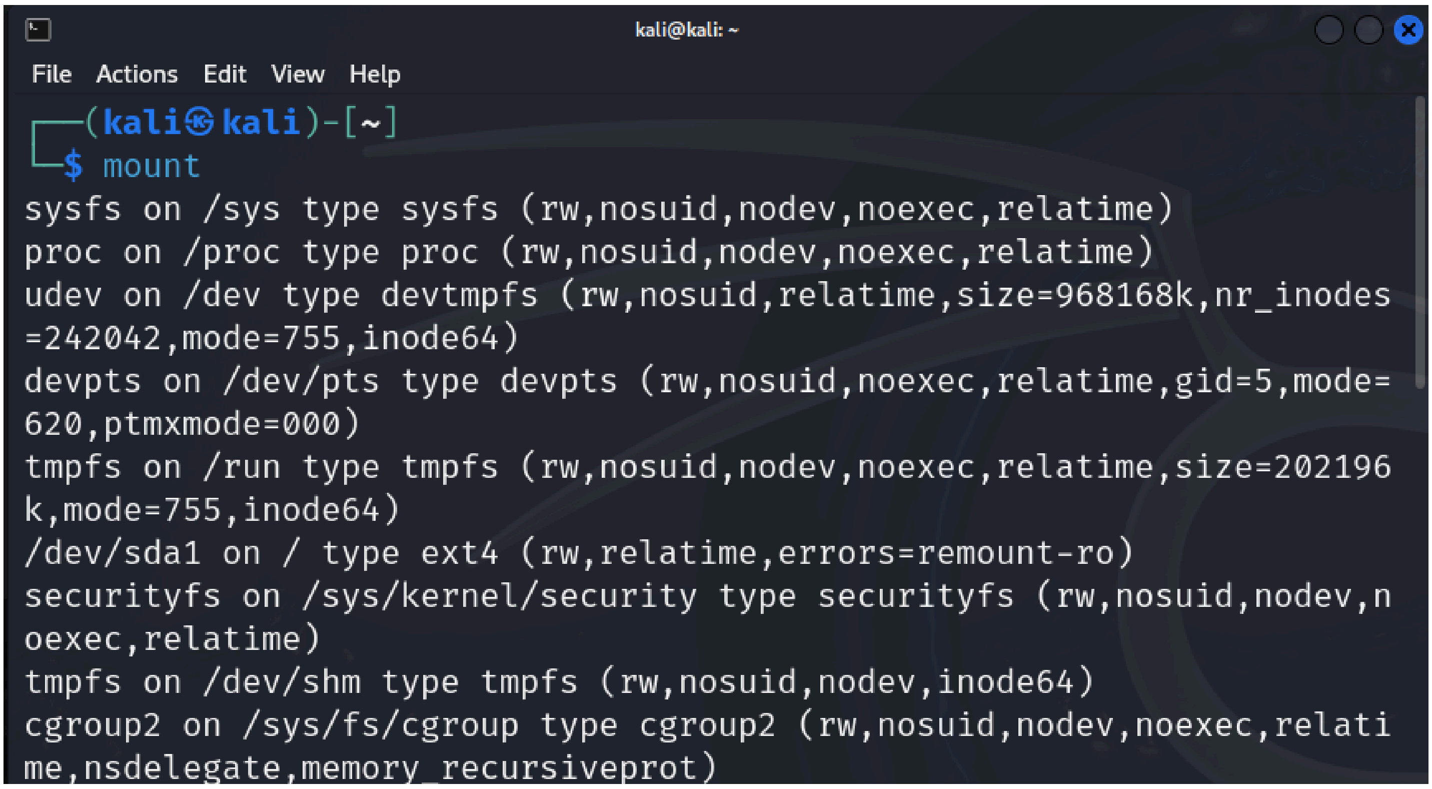


- **Give `mount` command on shell without any argument, what information it displays, use man page to describe each line of information. Mention two files which also contains the same information that mount command displays. Confirm**

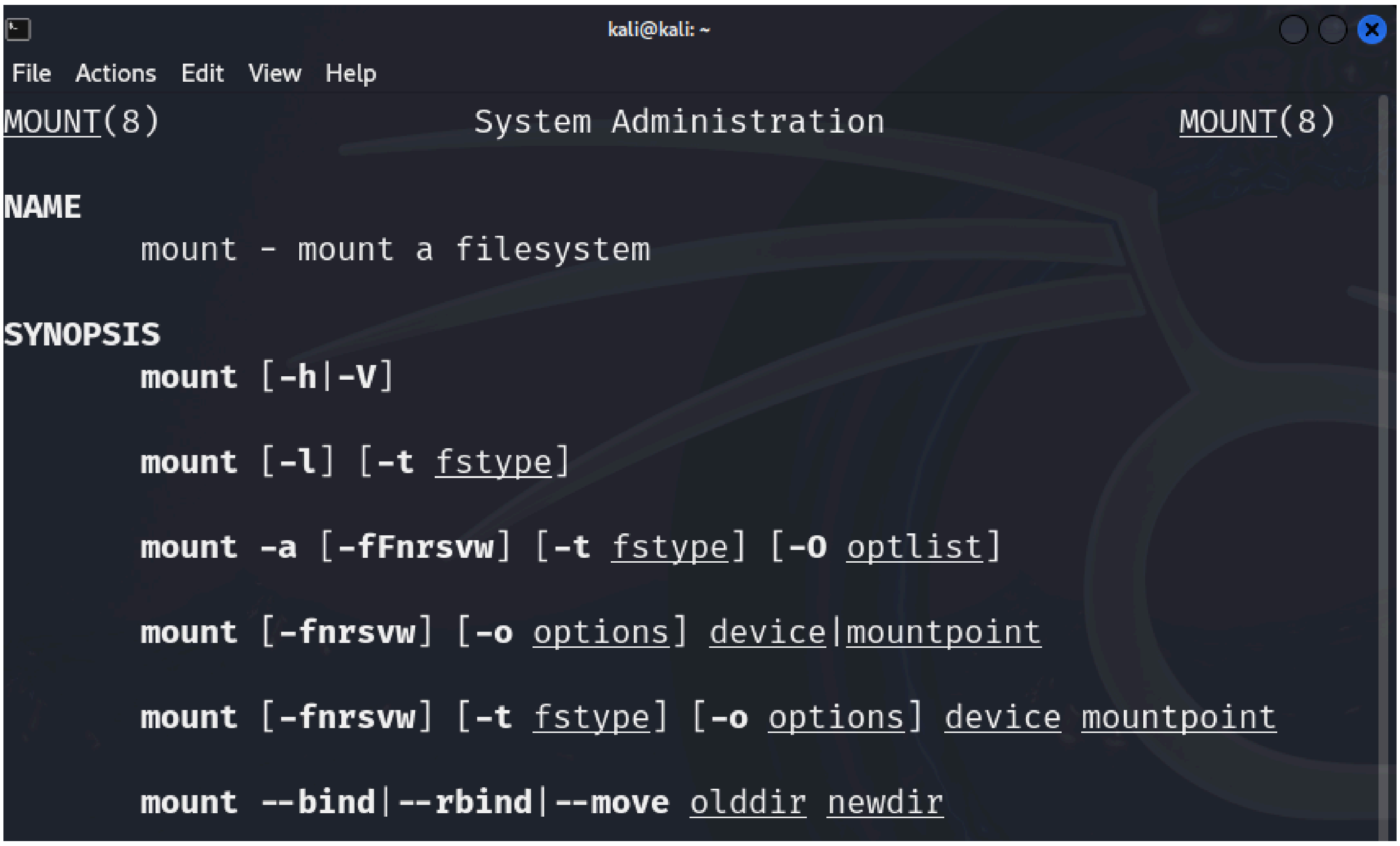
It displays a list of all currently mounted file systems and their associated information.

Two Files Containing Similar Information:

1. **/etc/fstab**:
   - The /etc/fstab file contains static information about the file systems and how they should be mounted during the boot process. It will show file systems, mount points, and options (though without real-time status information).
2. **/proc/mounts**:
   - The /proc/mounts file shows a real-time list of all currently mounted file systems. It's similar to the output of mount, but in a machine-readable format.

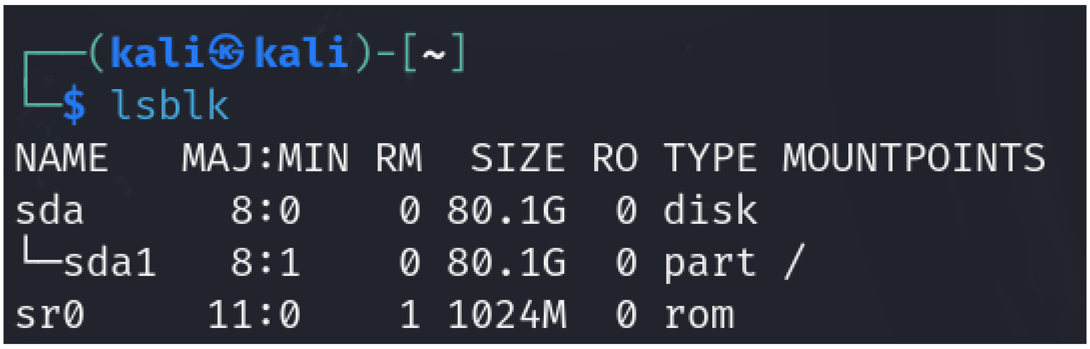


```
┌──(kali㉿kali)-[~]
└─$ mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=968168k,nr_inodes
=242042,mode=755,inode64)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=
620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,noexec,relatime,size=202196
k,mode=755,inode64)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,n
oexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
cgroup2 on /sys/fs/cgroup type cgroup2 (rw,nosuid,nodev,noexec,relati
me,nsdelegate,memory_recursiveprot)
```



```
MOUNT(8)                System Administration               MOUNT(8)

NAME
       mount - mount a filesystem

SYNOPSIS
       mount [-h|-V]

       mount [-l] [-t fstype]

       mount -a [-fFnrsvw] [-t fstype] [-O optlist]

       mount [-fnrsvw] [-o options] device|mountpoint

       mount [-fnrsvw] [-t fstype] [-o options] device mountpoint

       mount --bind|--rbind|--move olddir newdir
```

- **Write down a command that displays a list of all available block devices attached on your system. Use `man` page and write down the sample output on your copies. Do describe each entry.**

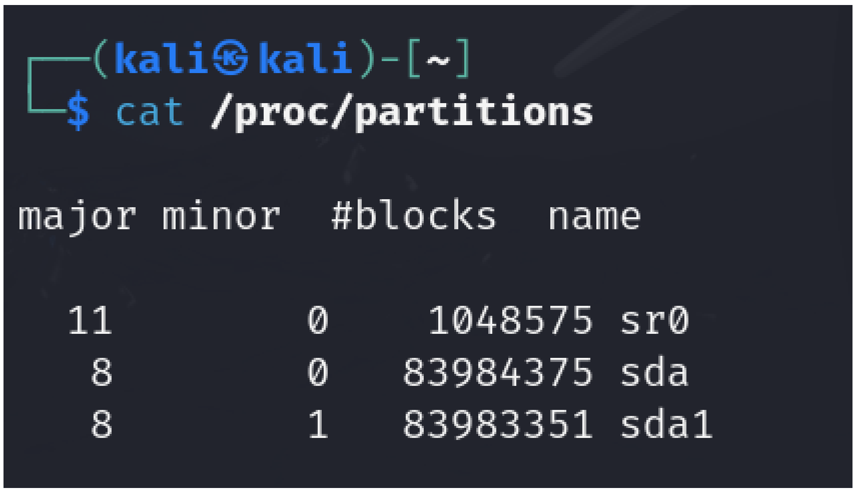


```
┌──(kali㉿kali)-[~]
└─$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0 80.1G  0 disk
└─sda1   8:1    0 80.1G  0 part /
sr0     11:0    1 1024M  0 rom
```

Explanation of Each Column:

1. **NAME:**
- This shows the name of the block device or partition. Block devices typically appear as sda, sdb, etc. Partitions are shown as sub-level devices like sda1, sdb1, sdb2, etc.
2. **MAJ:MIN:**
- This column shows the major and minor device numbers. The major number identifies the driver associated with the device, and the minor number identifies the specific device handled by that driver.
3. **RM:**
- **RM (Removable):** This shows whether the device is removable (e.g., a USB drive).
    - `0` means the device is not removable.
    - 1 means the device is removable.
4. **SIZE:**
- This column shows the size of the block device or partition.
5. **RO:**
- RO (Read-Only): This indicates whether the device is mounted in read-only mode.
    - `0` means the device is not read-only (it is writable).
    - 1 means the device is read-only.
6. **TYPE:**
- This column shows the type of block device.
7. **MOUNTPOINT:**
- This shows the mount points for the devices, if they are mounted. A mount point is the directory in the file system where the device or partition is attached.

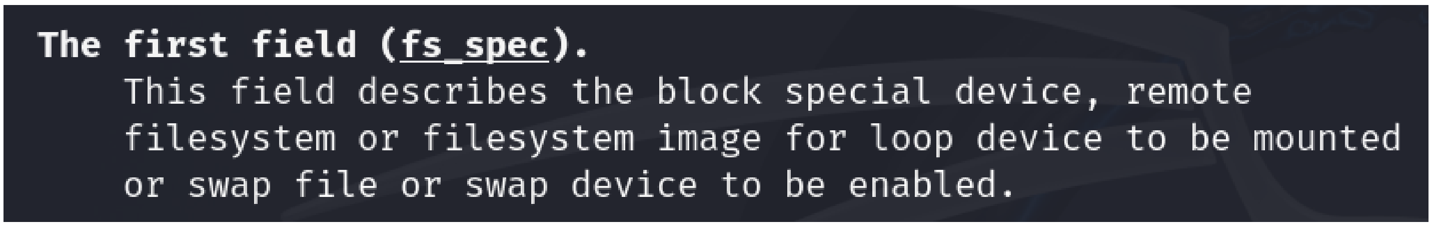
- **Review the contents of the file `/proc/partitions`, use the `man` page of `proc` to understand its contents**



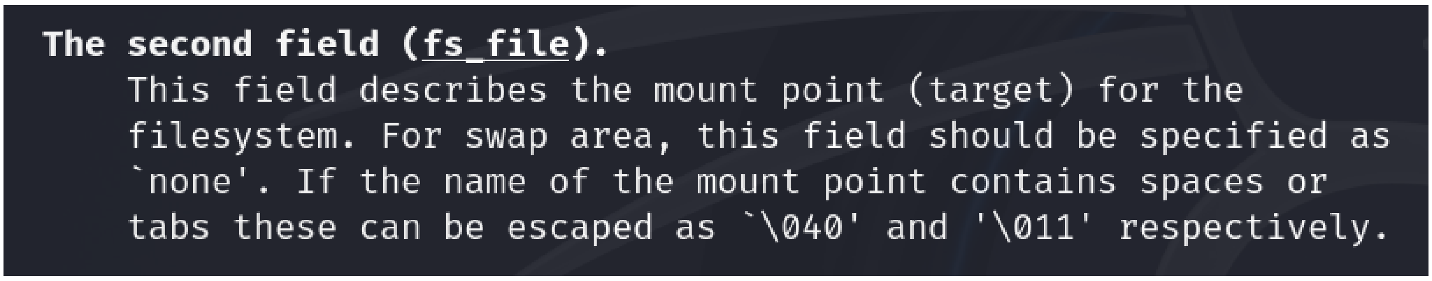
```
┌──(kali㉿kali)-[~]
└─$ cat /proc/partitions

major minor  #blocks  name

  11        0    1048575 sr0
   8        0   83984375 sda
   8        1   83983351 sda1
```

1. **major:**
- This column shows the major device number, which identifies the device driver. This number is associated with the specific device type (e.g., `sda`, `sdb`). The major number helps the kernel determine which driver to use to interact with the device.
2. **minor:**
- The minor device number identifies a particular instance of the device (or partition) handled by the driver.
3. **#blocks:**
- This column shows the size of the partition or disk in 1K blocks. Each block is typically 1024 bytes.
4. **name:**
- It represents the block devices and their partitions in a format understood by the system and users.

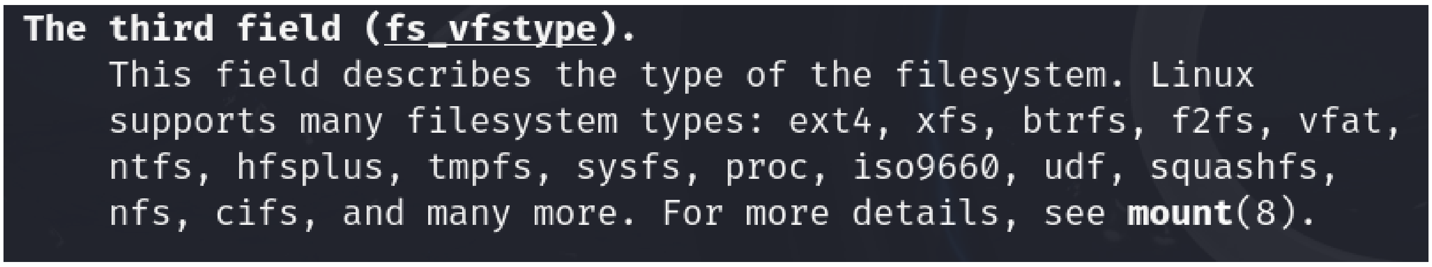
- **Review the man page of `fstab` in section 5 and write at least one line description each of its six fields on your note book. Be ready to answer questions of TAs**



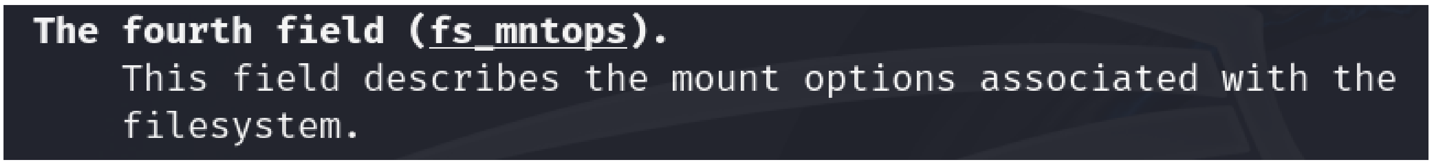
```
The first field (fs_spec).
    This field describes the block special device, remote
    filesystem or filesystem image for loop device to be mounted
    or swap file or swap device to be enabled.
```



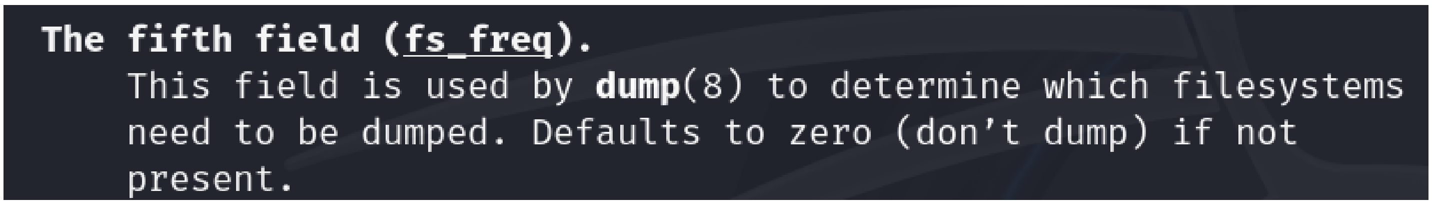
```
The second field (fs_file).
    This field describes the mount point (target) for the
    filesystem. For swap area, this field should be specified as
    `none'. If the name of the mount point contains spaces or
    tabs these can be escaped as `\040' and '\011' respectively.
```



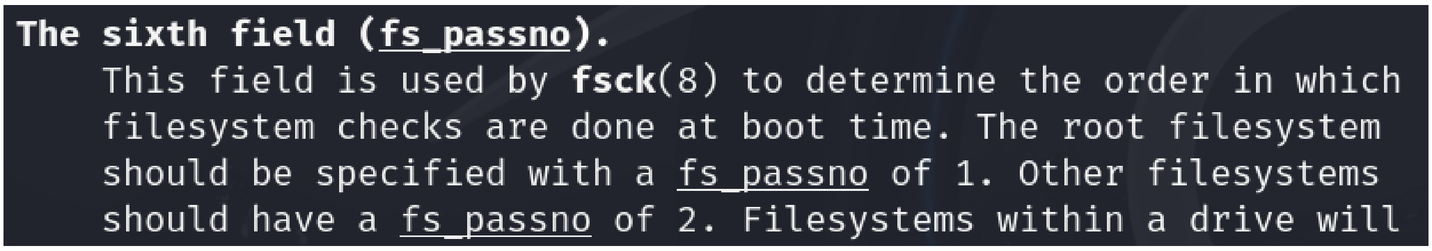
```
The third field (fs_vfstype).
    This field describes the type of the filesystem. Linux
    supports many filesystem types: ext4, xfs, btrfs, f2fs, vfat,
    ntfs, hfsplus, tmpfs, sysfs, proc, iso9660, udf, squashfs,
    nfs, cifs, and many more. For more details, see mount(8).
```



```
The fourth field (fs_mntops).
    This field describes the mount options associated with the
    filesystem.
```

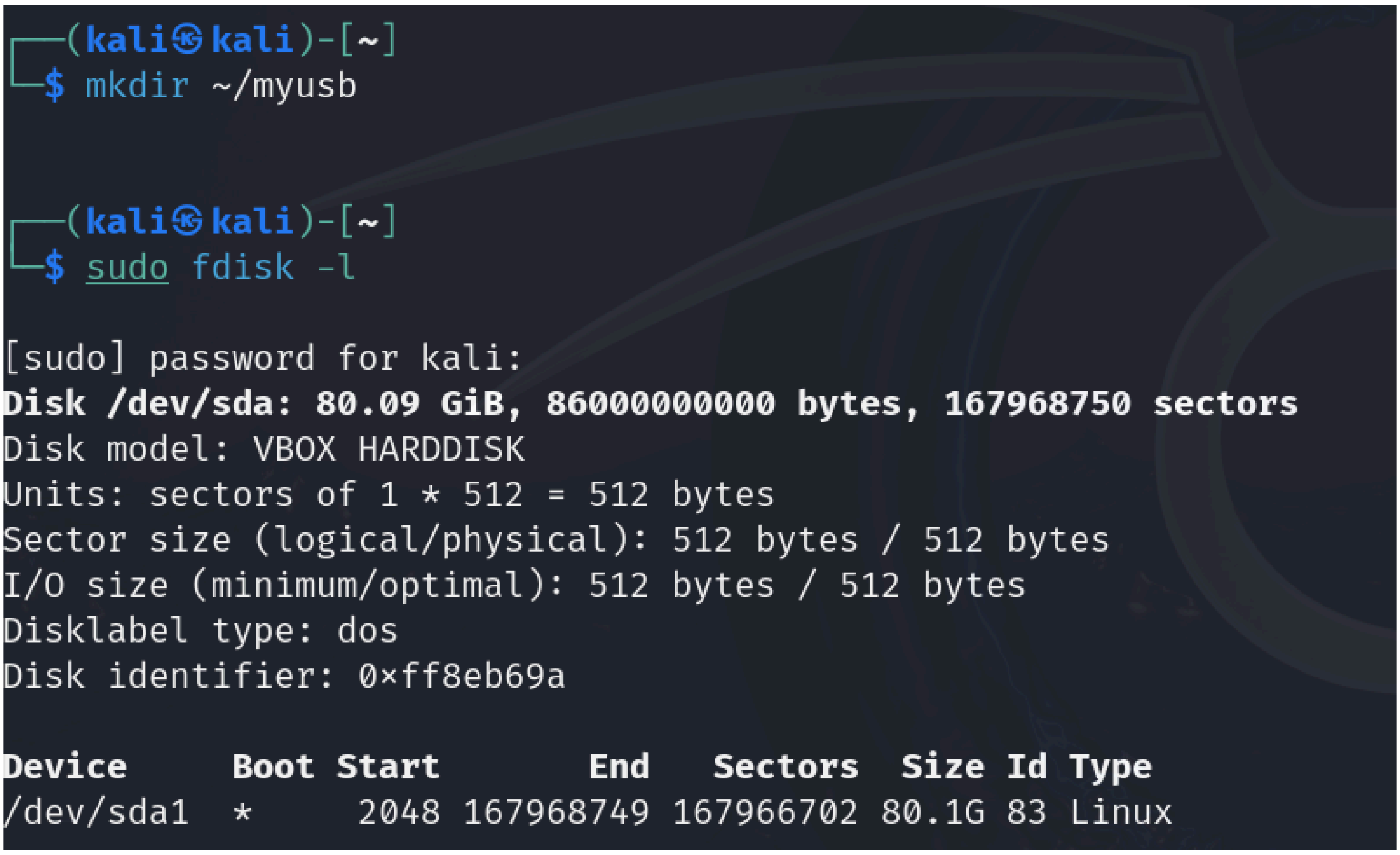


```
The fifth field (fs_freq).
    This field is used by dump(8) to determine which filesystems
    need to be dumped. Defaults to zero (don't dump) if not
    present.
```

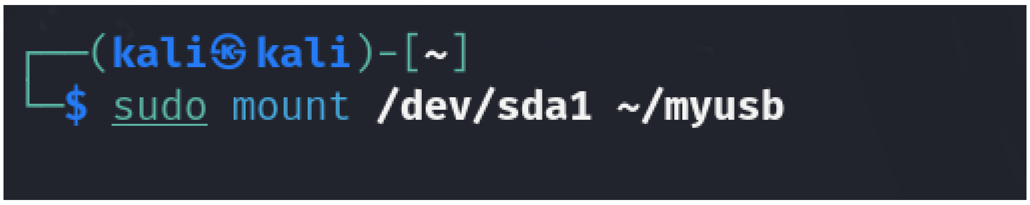


```
The sixth field (fs_passno).
    This field is used by fsck(8) to determine the order in which
    filesystem checks are done at boot time. The root filesystem
    should be specified with a fs_passno of 1. Other filesystems
    should have a fs_passno of 2. Filesystems within a drive will
```

- **Write down a shell command that mounts the USB attached with your system in a directory named `myusb` in your home directory.**

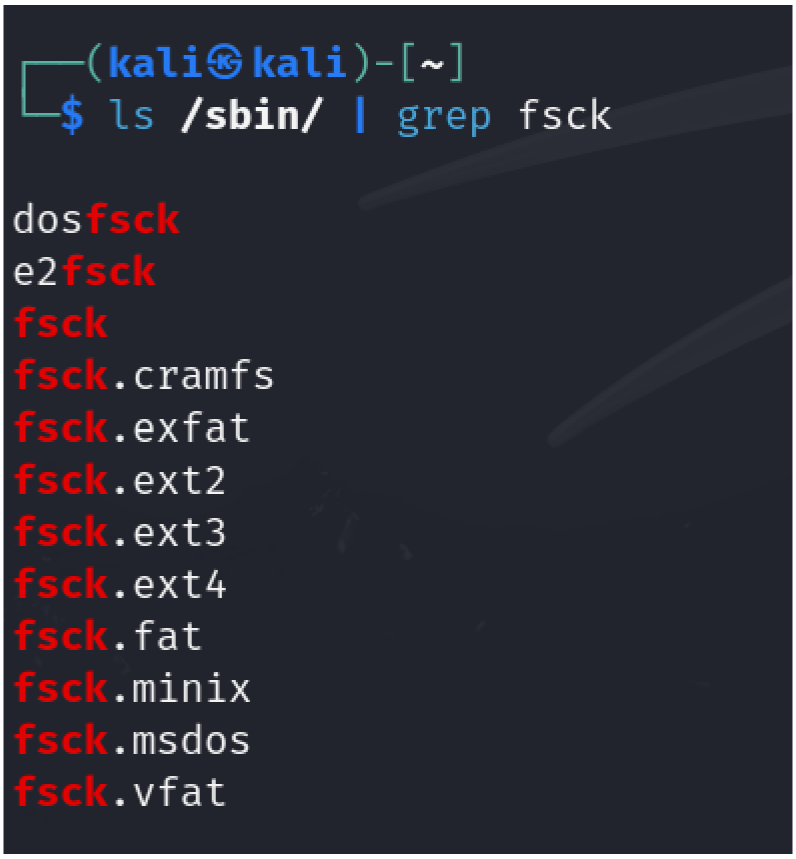


```
┌──(kali㉿kali)-[~]
└─$ mkdir ~/myusb


┌──(kali㉿kali)-[~]
└─$ sudo fdisk -l

[sudo] password for kali:
Disk /dev/sda: 80.09 GiB, 86000000000 bytes, 167968750 sectors
Disk model: VBOX HARDDISK
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: 0×ff8eb69a

Device     Boot Start       End   Sectors  Size Id Type
/dev/sda1  *     2048 167968749 167966702 80.1G 83 Linux
```



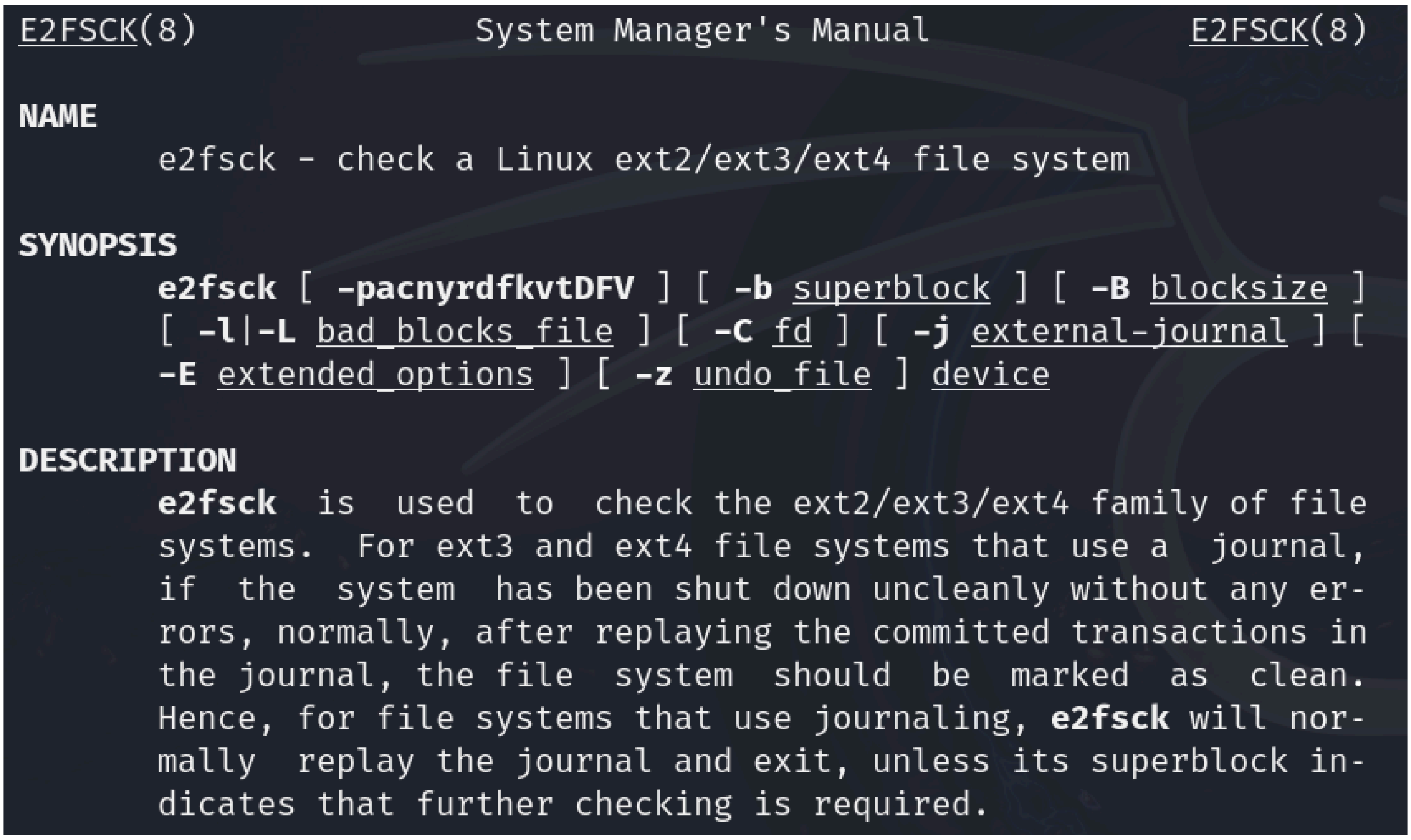
```
┌──(kali㉿kali)-[~]
└─$ sudo mount /dev/sda1 ~/myusb
```

- **Write down a shell command that displays all the commands available on your system in the `/sbin/` directory having string `fsck` in between. After you see these commands use man page to describe difference between `e2fsck` command and `fsck.ext3`**

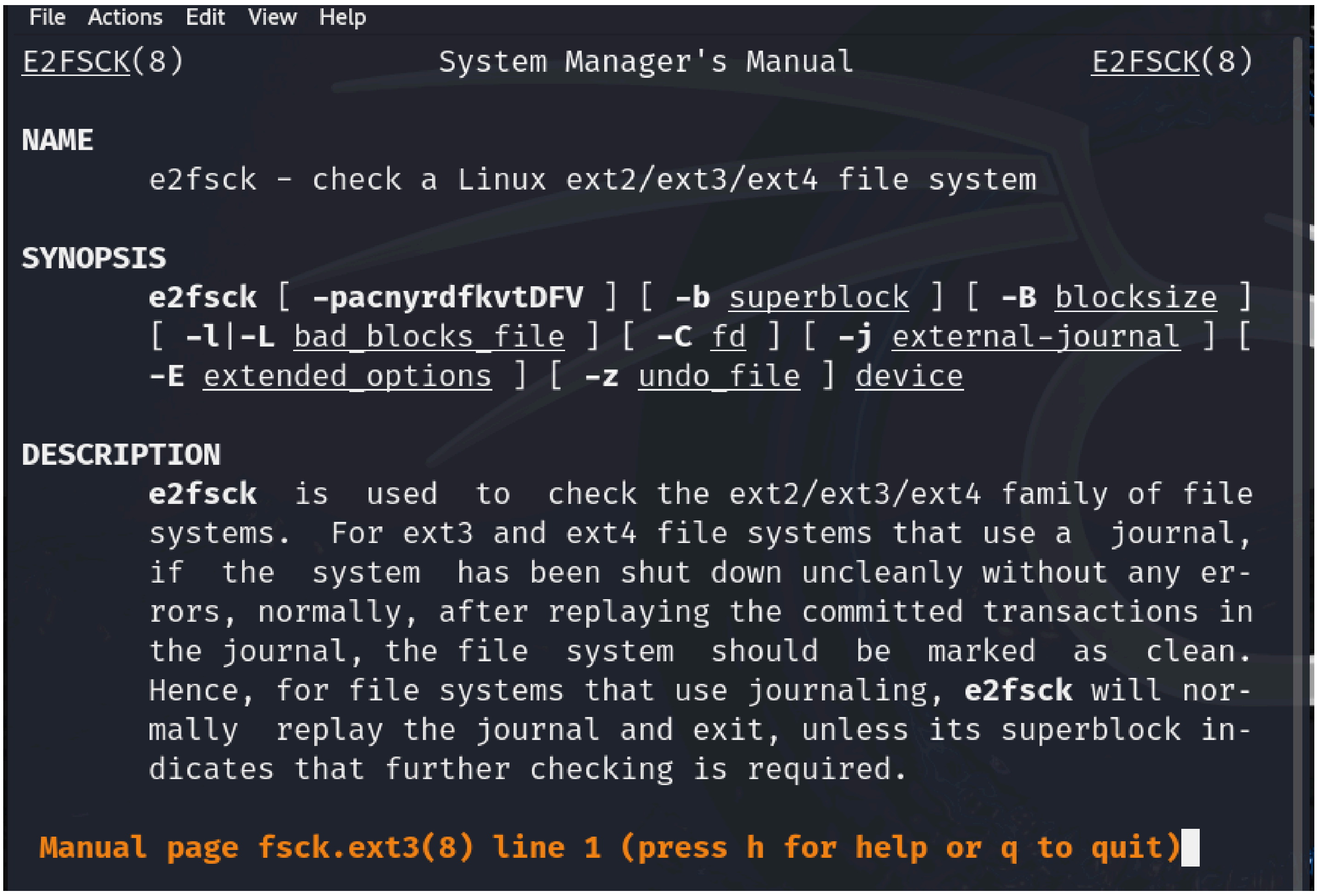


```
┌──(kali㉿kali)-[~]
└─$ ls /sbin/ | grep fsck

dosfsck
e2fsck
fsck
fsck.cramfs
fsck.exfat
fsck.ext2
fsck.ext3
fsck.ext4
fsck.fat
fsck.minix
fsck.msdos
fsck.vfat
```

e2fsck: A more general-purpose tool used to check and repair ext2, ext3, and ext4 file systems. It's the preferred tool for newer ext4 systems as well.
fsck.ext3: A specific version of fsck designed for checking ext3 file systems. It functions similarly to e2fsck but is specific to the ext3 format.



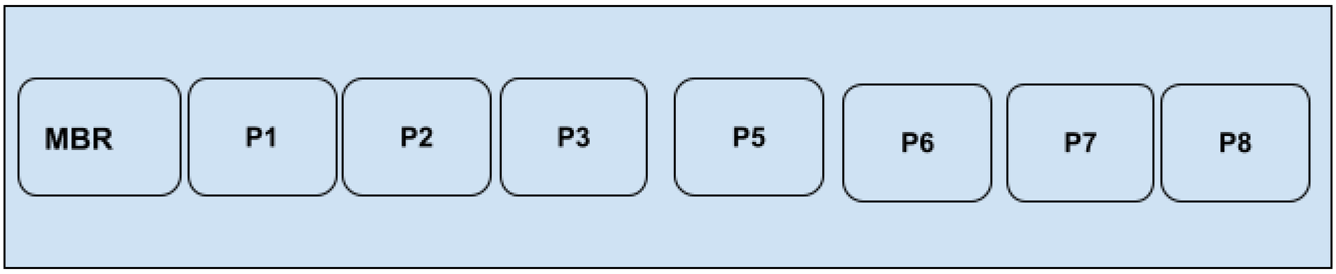
```
E2FSCK(8)                System Manager's Manual               E2FSCK(8)

NAME
       e2fsck - check a Linux ext2/ext3/ext4 file system

SYNOPSIS
       e2fsck [ -pacnyrdfkvtDFV ] [ -b superblock ] [ -B blocksize ]
       [ -l|-L bad_blocks_file ] [ -C fd ] [ -j external-journal ] [
       -E extended_options ] [ -z undo_file ] device

DESCRIPTION
       e2fsck  is  used  to  check the ext2/ext3/ext4 family of file
       systems.  For ext3 and ext4 file systems that use a  journal,
       if  the  system  has been shut down uncleanly without any er‐
       rors, normally, after replaying the committed transactions in
       the journal, the file  system  should  be  marked  as  clean.
       Hence, for file systems that use journaling, e2fsck will nor‐
       mally  replay the journal and exit, unless its superblock in‐
       dicates that further checking is required.
```



```
File  Actions  Edit  View  Help
E2FSCK(8)                   System Manager's Manual                  E2FSCK(8)

NAME
       e2fsck - check a Linux ext2/ext3/ext4 file system

SYNOPSIS
       e2fsck [ -pacnyrdfkvtDFV ] [ -b superblock ] [ -B blocksize ]
       [ -l|-L bad_blocks_file ] [ -C fd ] [ -j external-journal ] [
       -E extended_options ] [ -z undo_file ] device

DESCRIPTION
       e2fsck  is  used  to  check the ext2/ext3/ext4 family of file
       systems.  For ext3 and ext4 file systems that use a  journal,
       if  the  system  has been shut down uncleanly without any er-
       rors, normally, after replaying the committed transactions in
       the journal, the file  system  should  be  marked  as  clean.
       Hence, for file systems that use journaling, e2fsck will nor-
       mally  replay the journal and exit, unless its superblock in-
       dicates that further checking is required.

 Manual page fsck.ext3(8) line 1 (press h for help or q to quit)
```

<u>**Task 2:**</u>

- **Draw the schematic view of UNIX file system with a hard disk showing its linear view and give details of a single partition of your hard disk, showing different data blocks and other related blocks used by operating system for management purposes. Do mention the usage of these blocks.**



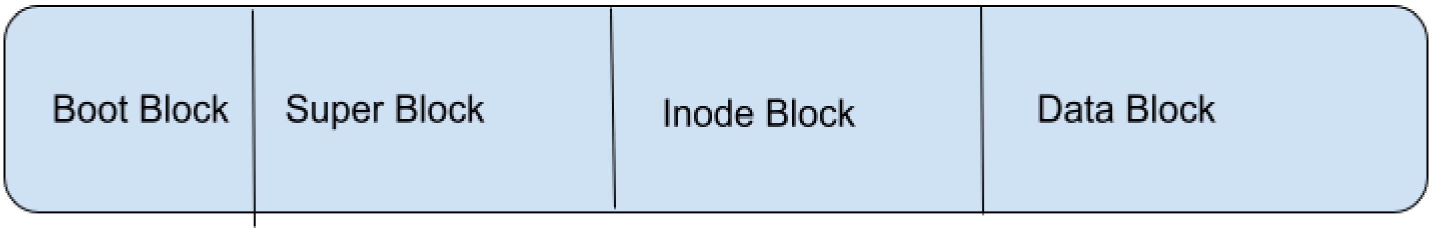
Explanation of Each Block in the Partition:

- **Boot Block (Block 0)**:
    - **Location**: The first block of the partition.

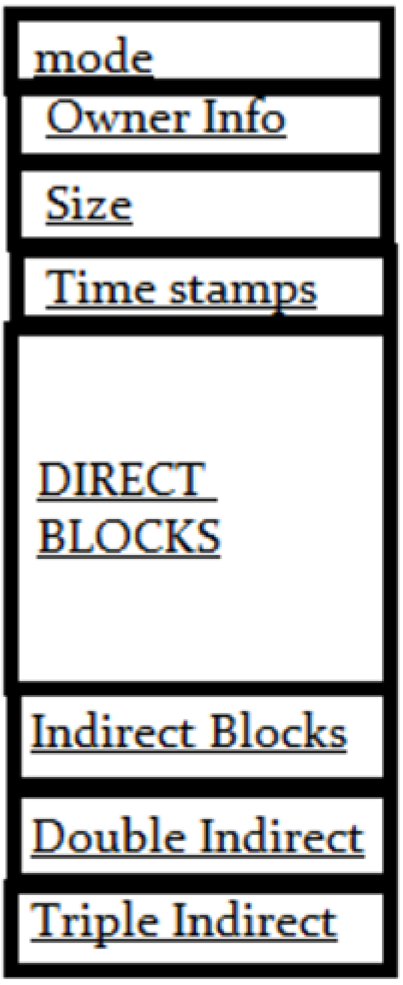
- **Usage**: Contains the **boot loader** or a small program that helps in loading the operating system. The boot block is essential for booting the system from this partition.
    - The boot loader is responsible for loading the kernel into memory during the system startup.

- **Superblock (Block 1)**:
    - **Location**: The second block of the partition.
    - **Usage**: Contains important information about the file system structure, such as:
        - The file system type (e.g., ext4, ext3).
        - The total number of blocks and free blocks.
        - The location of the inode table.
        - The number of inodes and their allocation.
        - File system status and mount information.
        - The block size and other file system parameters.
    - **Importance**: If the superblock is corrupted, the entire file system may become unreadable.
- **Inode Blocks (Block 2 and onwards)**:
    - **Location**: Typically placed after the superblock, starting from block 2.
    - **Usage**: Inodes store metadata about files, such as:
        - File type (regular file, directory, symbolic link, etc.).
        - Ownership (user and group IDs).
        - File size.
        - Access control information (permissions).
        - File's timestamps (creation, modification, and access time).
        - Pointers to data blocks (where the actual data of the file is stored).
    - **Details**:
        - Each inode represents a file or a directory.
        - Inodes do not contain file names; the mapping between the file name and inode is done through directory entries.
- **Data Blocks (Block N onwards)**:
    - **Location**: After the inode blocks, starting at the block number specified by the superblock.
    - **Usage**: These blocks hold the actual content of the files (data), including:
        - File contents (text, binaries, images, etc.).
        - Directory data (list of file names and corresponding inode numbers).
        - Special files like device files and named pipes.
        - Each file in the file system is represented by an inode, and each inode points to a set of data blocks.
    - **Details**: Data blocks are where the file's actual data resides. The file system maintains a list of pointers (addresses) in the inode, and these pointers refer to blocks on the disk that contain the actual file data.

Schematic Representation of the Partition Structure



- **Draw the schematic view of a UNIX inode block showing its contents particularly the thirteen pointers with data blocks in detail. What will be maximum file size support in a UNIX system that has 1KiB block size and each disk pointer occupies 4B of space.**



+----------------- Pointer 1 ----> Data Block 1 (1KiB) —--------------|

+----------------- Pointer 2----> Data Block 2 (1KiB) —--------------|

+----------------- Pointer 3 ----> Data Block 3 (1KiB) —--------------| +----------------- Pointer 4----> Data Block 4(1KiB) —--------------|

+---

—---

—---

—---

—---

—---

—--

+----------------- Pointer12----> Data Block 12 (1KiB) —--------------|

Total Maximum File Size

Now, adding everything together:

- Direct Pointers: 12 KiB

- Single Indirect Pointer: 256 KiB
- Double Indirect Pointer: 64 MiB
- Triple Indirect Pointer: 16 GiB

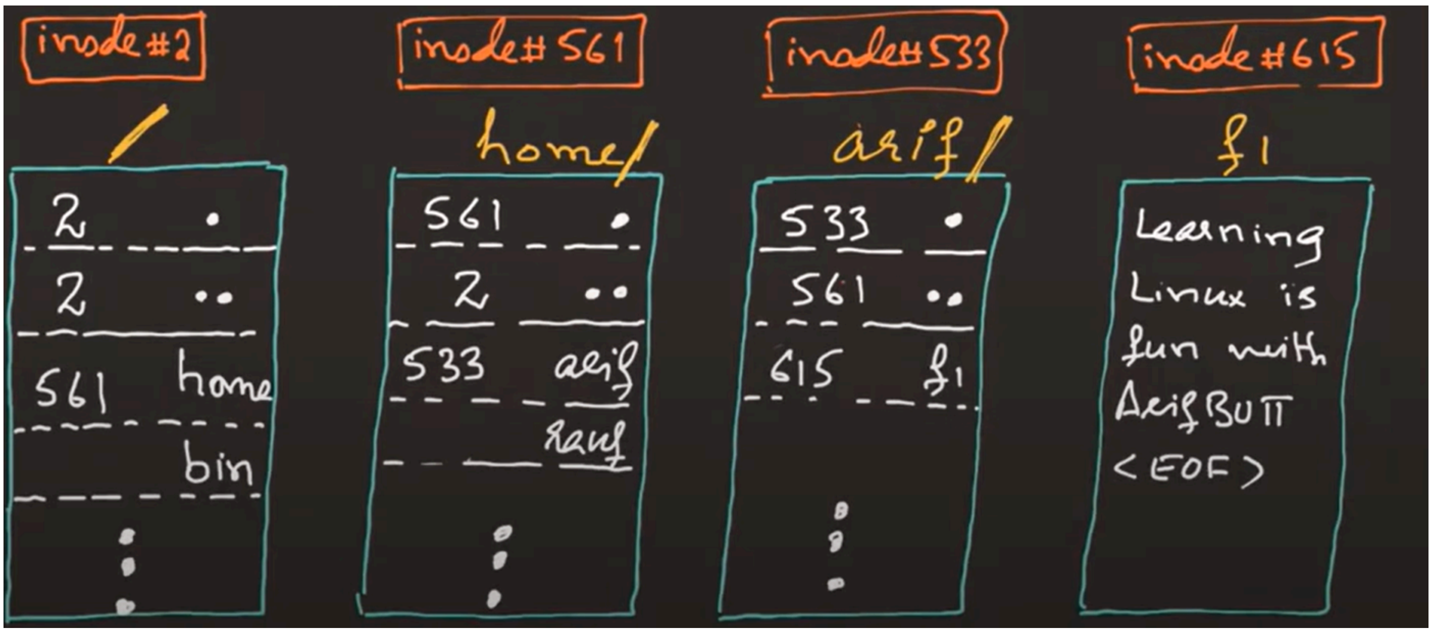
The total maximum file size supported by this inode structure would be:

12 KiB+256 KiB+64 MiB+16 GiB=16.064 GiB
Thus, the maximum file size that can be supported is approximately 16 GiB (gigabytes).

- **When a user creates a file on the disk, what are the steps that are followed by Linux kernel to populate different data structures on the hard disk. Describe by drawing and labeling diagram showing contents of directories involved. Get ready to answer viva questions of TAs as to how a file is searched and read by different programs like `cat`.**

Steps to Create a File:

- **Stores file metadata**
- **Record the address of data blocks**
- **Stores file data in data block**
- **Adds an entry in parent directory**
  - The file needs to be inserted into a directory so that users can access it by its name.
  - A **directory entry** is created for the new file in the specified directory (e.g., `/home/user/`).
  - The directory entry contains the **filename** and a reference to the **inode** of the file.

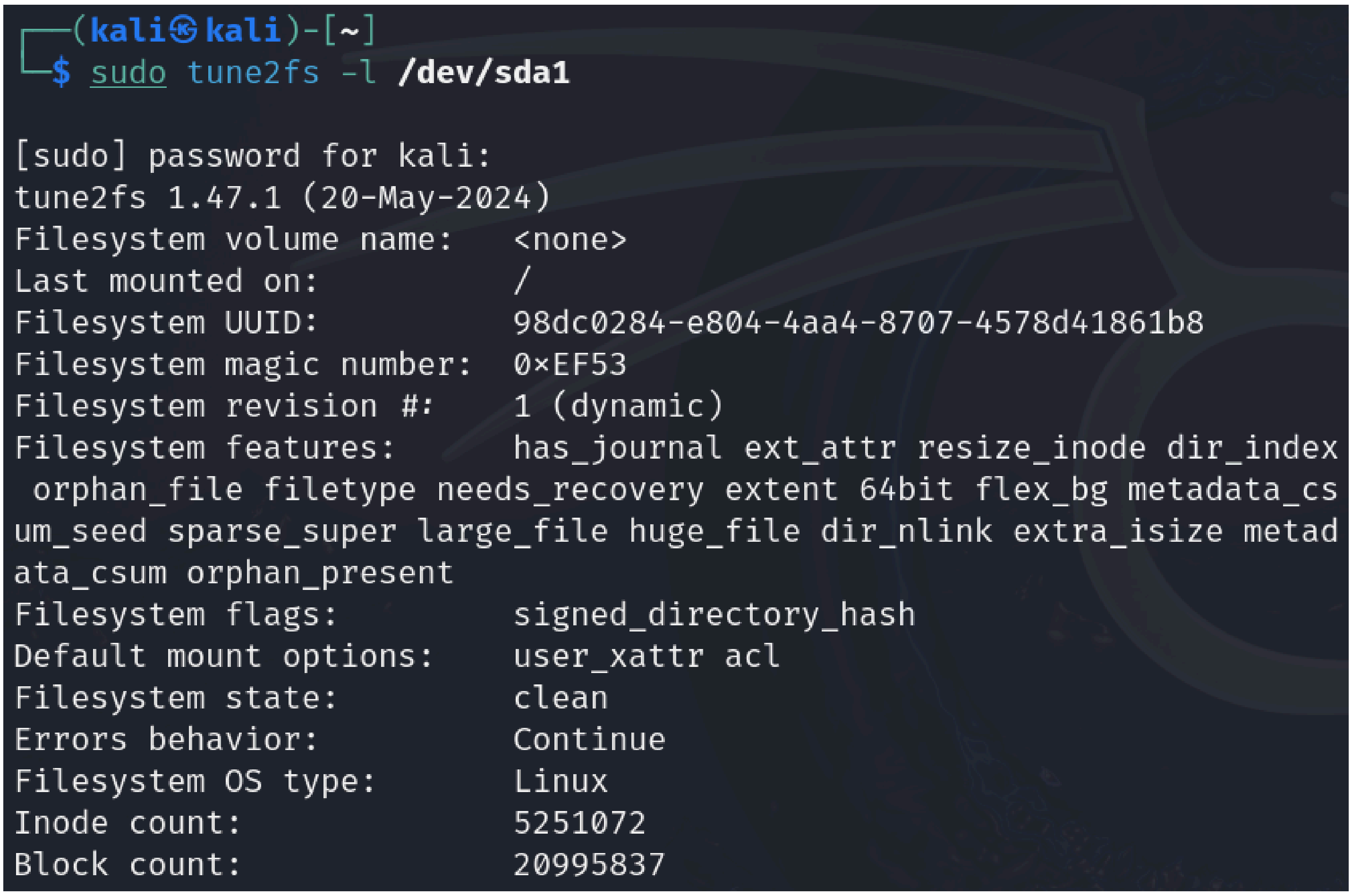


Steps for Searching and Reading a File in Linux:

1. **File Search Process:**
   - Step 1: File Name Lookup:
   - Step 2: Inode Lookup:
   - Step 3: Access Control:
2. **Reading the File with `cat`:**

- Step 1: Open File:
- Step 2: Read Data Blocks:
- Step 3: Output:

- **Write down a shell command that displays a detailed list of file system parameters on your Linux file system, like Mount point, UUID, magic number, free blocks, free inodes, first block address, block size of the first partition of the first scsci hard disk attached with your system.**

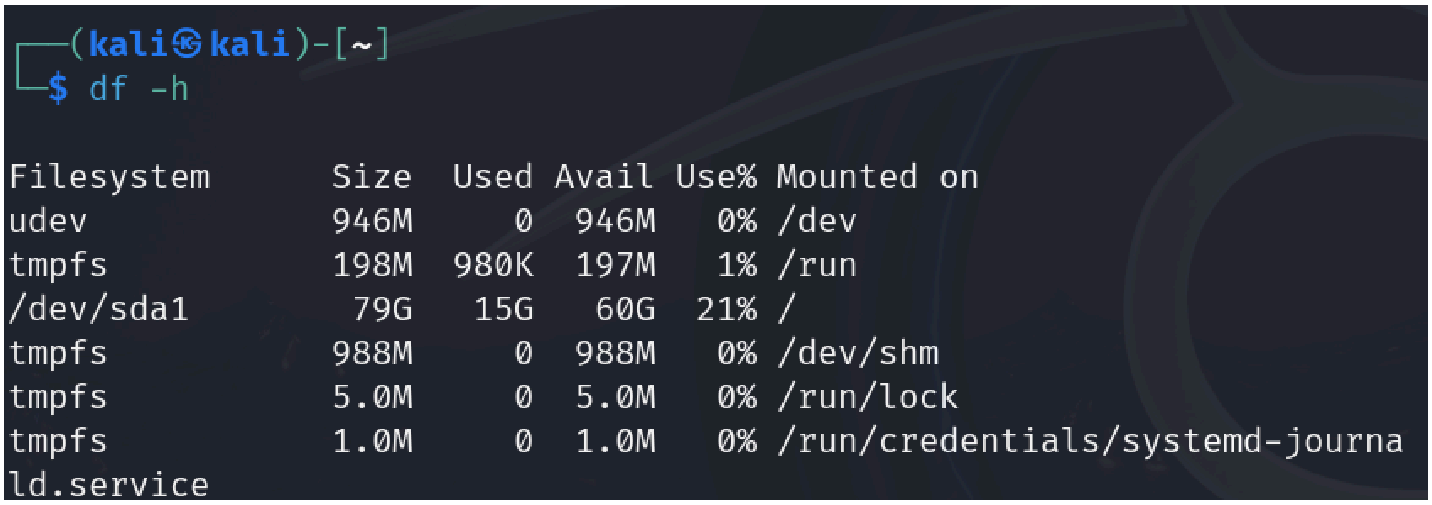


```
┌──(kali㉿kali)-[~]
└─$ sudo tune2fs -l /dev/sda1

[sudo] password for kali:
tune2fs 1.47.1 (20-May-2024)
Filesystem volume name:   <none>
Last mounted on:          /
Filesystem UUID:          98dc0284-e804-4aa4-8707-4578d41861b8
Filesystem magic number:  0×EF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index
 orphan_file filetype needs_recovery extent 64bit flex_bg metadata_cs
um_seed sparse_super large_file huge_file dir_nlink extra_isize metad
ata_csum orphan_present
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              5251072
Block count:              20995837
```

- **Write down a brief usage of `df` and `du` command by giving some sample usage on your note books.**

1. df (Disk Free) Command:

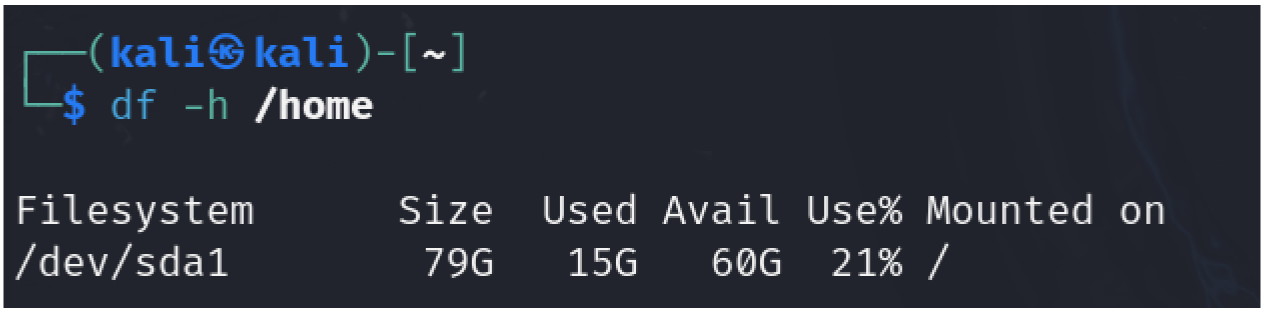
The df command is used to display the amount of free and used disk space on all mounted file systems. It shows the file system's capacity, used space, available space, and the mount points.



```
┌──(kali㉿kali)-[~]
└─$ df -h

Filesystem      Size  Used Avail Use% Mounted on
udev            946M     0  946M   0% /dev
tmpfs           198M  980K  197M   1% /run
/dev/sda1        79G   15G   60G  21% /
tmpfs           988M     0  988M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           1.0M     0  1.0M   0% /run/credentials/systemd-journa
ld.service
```

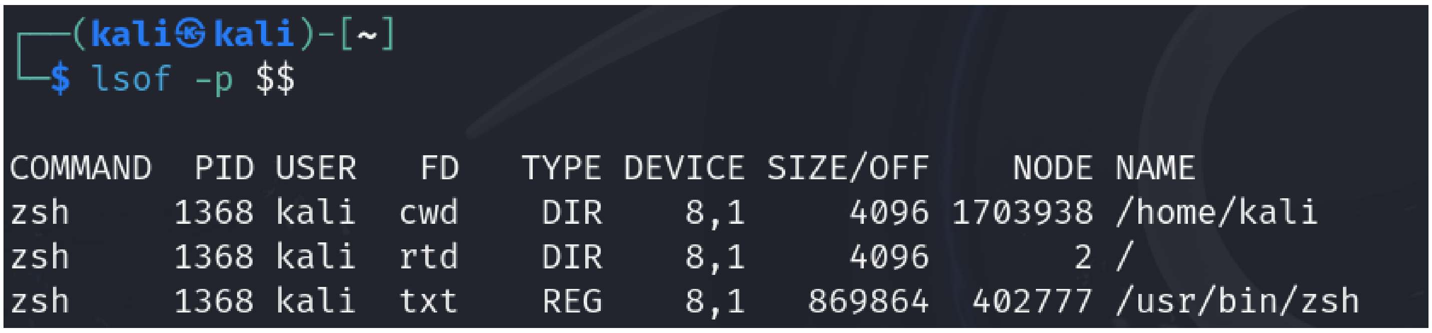
2. du (Disk Usage) Command:

The du command is used to estimate and display the disk space used by files and directories. It provides information about the size of directories and their contents.



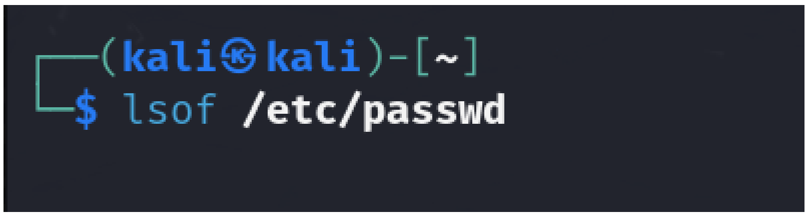
```
┌──(kali㉿kali)-[~]
└─$ df -h /home

Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        79G   15G   60G  21% /
```

- **Write down a shell command that displays the list of files that your terminal has currently opened.**



```
┌──(kali㉿kali)-[~]
└─$ lsof -p $$

COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
zsh     1368 kali  cwd    DIR    8,1     4096 1703938 /home/kali
zsh     1368 kali  rtd    DIR    8,1     4096       2 /
zsh     1368 kali  txt    REG    8,1   869864  402777 /usr/bin/zsh
```

- **Write down a shell command to which you pass a file name e.g, `/etc/passwd` and it tells you which all processes have that file opened.**



```
┌──(kali㉿kali)-[~]
└─$ lsof /etc/passwd
```