Revised: 1/20/2020

Filesystem Administration

Michael C. Hackett

Computer Science Department

Community College of Philadelphia

Lecture Topics

- Device Files
- Filesystem Types
- Hard Disk Management
  - Partitioning
  - Formatting
  - Mounting
  - Administration
- Filesystem Maintenance and Monitoring
  - Disk Usage
  - Checking for and Repairing Disk Errors
  - Mounting Filesystems at Boot
  - Disk Quotas

Device Files

- A **device file** is a file that represents a system device and specifies how to transfer data to and from the device.

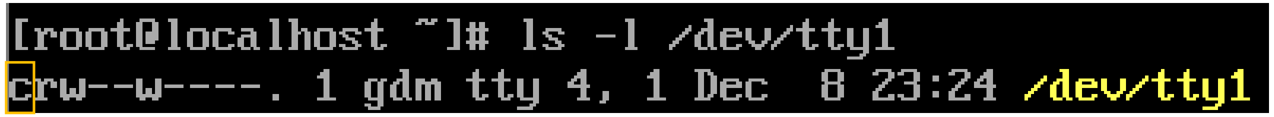
- All device files are typically stored in the /dev directory

Device Files

- System devices are either:
  - **Character Devices**, which transfer data character-by-character to and from the device.
    - Examples: Tape drives, serial ports, parallel ports, keyboards
  - **Block Devices**, which transfer blocks of data using physical memory to buffer the transfer.
    - Examples: Hard disks and CDs/DVDs
- Block devices have faster data transfer than character devices.

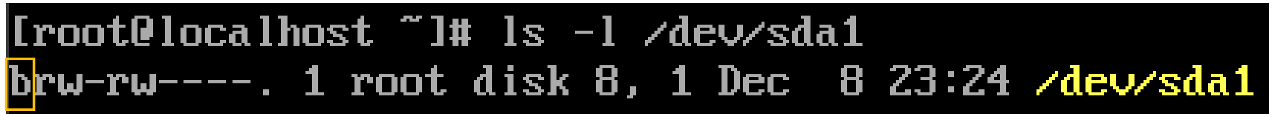
Device Files

- In a long listing, character device files have a “c” before the mode.



```
[root@localhost ~]# ls -l /dev/tty1
crw--w----. 1 gdm tty 4, 1 Dec  8 23:24 /dev/tty1
```

- Block device files have a “b” before the mode



```
[root@localhost ~]# ls -l /dev/sda1
brw-rw----. 1 root disk 8, 1 Dec  8 23:24 /dev/sda1
```

Device Files

- Common device files:
  - /dev/fd0 First floppy disk drive
  - /dev/sr0 First CD/DVD device
    - There is often a /dev/cdrom file linked to /dev/sr0
  - /dev/hda1 First partition on the first PATA hard disk drive
  - /dev/hdb1 First partition on the second PATA hard disk drive
  - /dev/hdc1 First partition on the third PATA hard disk drive
  - /dev/hdd1 First partition on the fourth PATA hard disk drive

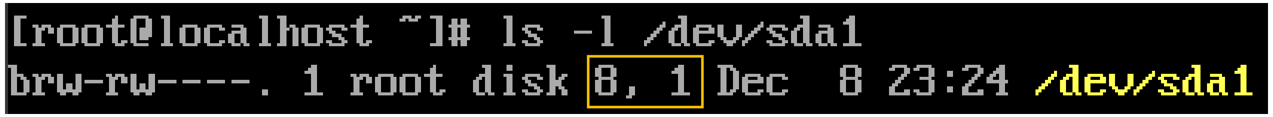
Device Files

- Common device files:

/dev/sda1	First partition on the first SCSI/SATA hard disk drive
/dev/sdb1	First partition on the second SCSI/SATA hard disk drive
/dev/tty1	First local terminal
/dev/tty2	Second local terminal
/dev/ttyS0	First serial port (COM1)
/dev/ttyS1	Second serial port (COM2)
/dev/lp0	First parallel port (LPT1)
/dev/st0	First SCSI tape drive

Device Files

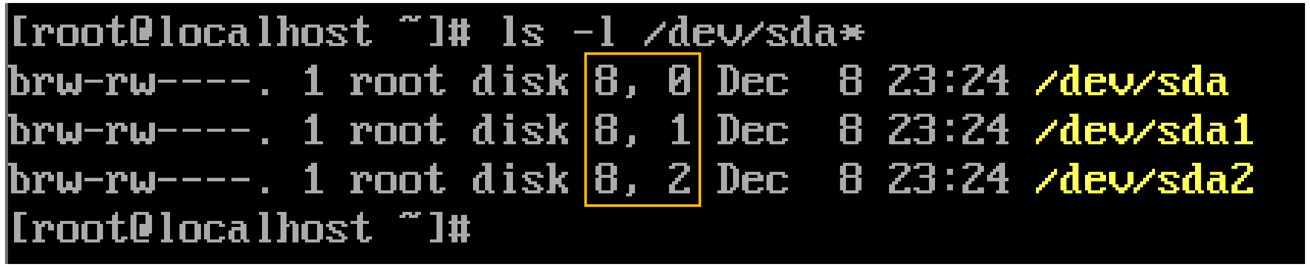
- When performing a long listing of a device file, the file size is replaced with two numbers.
- These indicate the major and minor numbers.
  - **Major number**: Indicates the device driver for it in the kernel
  - **Minor number**: Indicates the device itself



```
[root@localhost ~]# ls -l /dev/sda1
brw-rw----. 1 root disk 8, 1 Dec  8 23:24 /dev/sda1
```

Device Files

- In this example, the system's first SATA hard disk drive (sda) has two partitions (sda1 and sda2)



```
[root@localhost ~]# ls -l /dev/sda*
brw-rw----. 1 root disk 8, 0 Dec  8 23:24 /dev/sda
brw-rw----. 1 root disk 8, 1 Dec  8 23:24 /dev/sda1
brw-rw----. 1 root disk 8, 2 Dec  8 23:24 /dev/sda2
[root@localhost ~]#
```

- All three use the same kernel device driver, indicated by 8 (the major number)
- Each is identified by a unique minor number (0, 1, and 2)

Device Files

- Multiple devices can share the same major number.
  - Though, each device will have a unique minor number.

- If this system had two SATA hard disks, each with two partitions, the major and minor numbers for their device files might be something like:

  sda (8, 0)
  sda1 (8, 1)
  sda2 (8, 2)
  sdb (8, 3)
  sdb1 (8, 4)
  sdb2 (8, 5)

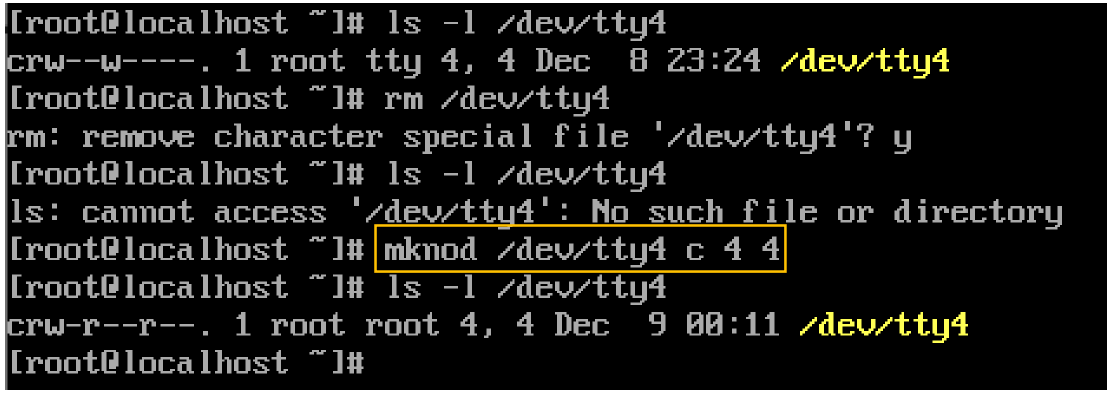
Device Files

- If a device file ever becomes corrupted, it will be listed as a regular file.
  - Can be identified by using **`find /dev –type f`**

- To recreate a device file (if it becomes corrupted or is accidentally deleted), the **`mknod`** command will **m**a**k**e a new device file and with a corresponding i**nod**e

Device Files

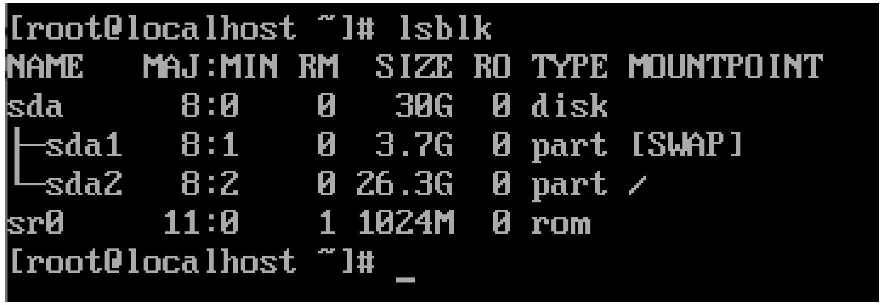
- The syntax of the mknod command is
  mknod *file type major minor*
- If tty4 were accidentally deleted it could be recreated with the following command: **`mknod /dev/tty4 c 4 4`**
  - Note: You need to know the device's major and minor number.



```
[root@localhost ~]# ls -l /dev/tty4
crw--w----. 1 root tty 4, 4 Dec  8 23:24 /dev/tty4
[root@localhost ~]# rm /dev/tty4
rm: remove character special file '/dev/tty4'? y
[root@localhost ~]# ls -l /dev/tty4
ls: cannot access '/dev/tty4': No such file or directory
[root@localhost ~]# mknod /dev/tty4 c 4 4
[root@localhost ~]# ls -l /dev/tty4
crw-r--r--. 1 root root 4, 4 Dec  9 00:11 /dev/tty4
[root@localhost ~]#
```

Device Files

- A listing of all devices (character and block devices) and their major number is in the /proc/devices file.
  - Can be viewed using: **`cat /proc/devices`**
- A listing of all block devices is in the /sys/block directory.
  - Can be viewed using: **`ls /sys/block`**
- To **l**i**s**t detailed information about all **bl**oc**k** devices, use the **`lsblk`** command.



```
[root@localhost ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   30G  0 disk
├─sda1   8:1    0  3.7G  0 part [SWAP]
└─sda2   8:2    0 26.3G  0 part /
sr0     11:0    1 1024M  0 rom
[root@localhost ~]# _
```

Filesystem Types

- There are a number of different filesystem types available to use in Linux.

- Each has their own strengths and weaknesses.

- Luckily, a Linux system can use several devices formatted with different filesystems.

Filesystem Types

- FAT – File Allocation Table
  - Older filesystem compatible with many different operating systems.
  - Used for compatibility reasons; Does not provide the same capabilities as more modern filesystems.

- VFAT – Virtual FAT
  - FAT with long filename support

Filesystem Types

- exFAT – Extended FAT
  - Improved version of FAT with large file support.
  - Most commonly used filesystem for USB flash drives

- NTFS – New Technology File System
  - Proprietary Microsoft filesystem
  - Not natively supported by Linux

Filesystem Types

- ext2 (Second Extended Filesystem)
  - Improved version of the original extended filesystem
  - Traditional filesystem used in Linux
  - Still supported by Linux, but was mostly replaced by ext3

- ext3 (Third Extended Filesystem)
  - Improvement on ext2 and allows for journaling.
  - Faster startup and recovery time.
  - Insures data integrity better than ext2

Filesystem Types

- ext4 (Fourth Extended Filesystem)
  - Improved version of ext3
  - Larger filesystem support and speed enhancements
  - Supports volumes up to 1 exbibyte and files up to 16 tebibyte in size.

- xfs (X File System)
  - High-performance file system
  - Used when there is need to quickly write large numbers of files to the disk.

Filesystem Types

- btrfs (B-tree File System) (“*Butter FS*”)
  - Relatively new; intended to replace ext4
  - Geared toward large-scale storage
  - Has the ability to span multiple devices

Filesystem Types

- Those are a small selection of mostly general-purpose filesystems.

- Some filesystems function as network protocols to share data across a network.
  - Network File Systems

Filesystem Types

- **SMB** (Server Message Block)
    - Primarily used with Windows computers
    - SMB-compatible software called Samba helps interface Linux and Windows network shares
- CIFS (Common Internet File System)
    - Rarely used
    - Was intended to replace Samba 1, but Samba versions 2 and 3 superseded it.
- **NFS** (Network File System)
    - Preferred for network file serving between Linux clients and Linux servers
    - SMB is better for Linux/Windows network file serving

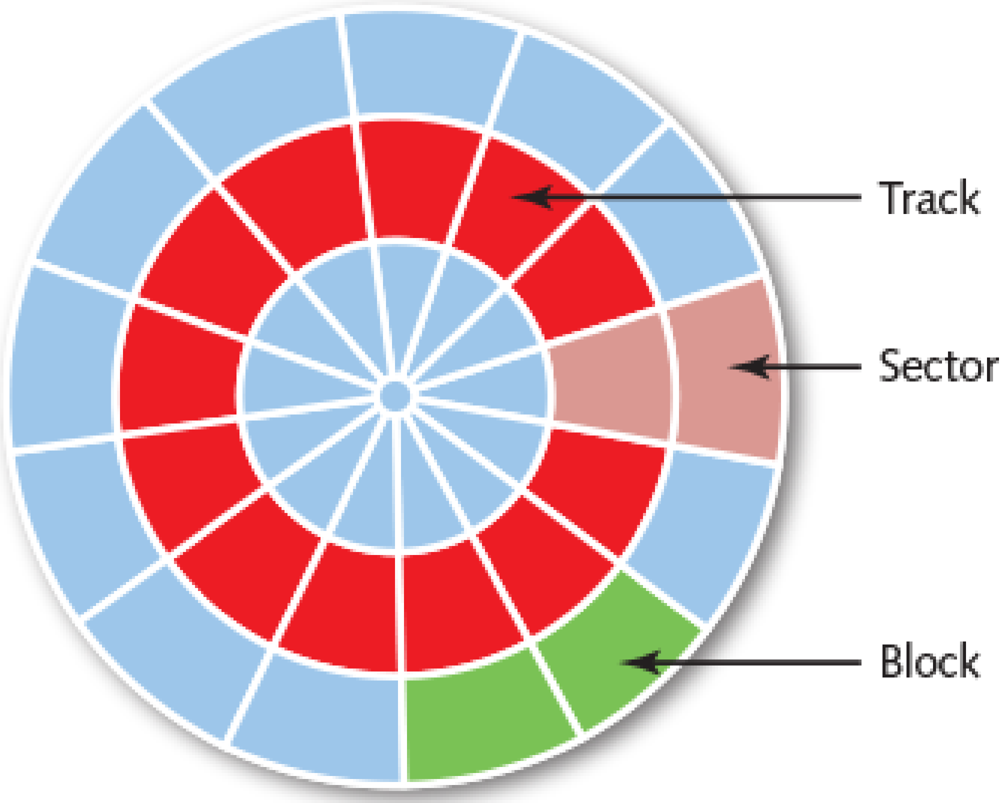
Partitioning

- Physically, hard disks contain one or more circular metal *platters* that spin at high speeds.

- Data is read from concentric circles on the platter called **tracks**

- Each track is divided into **sectors** of data
  - Sectors are combined into one or more **blocks**

Partitioning



Partitioning

- Hard disks are often split up into multiple smaller drives called **partitions**.

- Each partition contains its own filesystem and can each be mounted to different mount point directories.
  - More on this later in the lecture

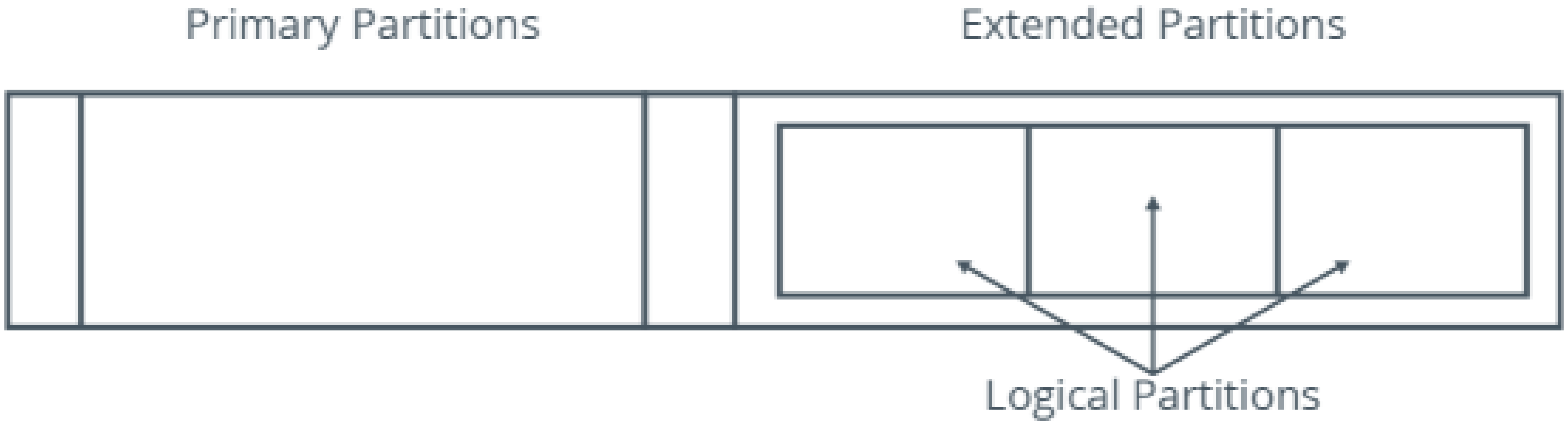
- Segregating data into separate areas of the hard disk is useful from an organizational standpoint.

Partitioning

- Hard disks with a Master Boot Record (MBR) normally contain up to four **primary partitions**
    - Any of the four primary partitions can replaced by an **extended partition**
    - Extended partitions can contain an unlimited number of (but normally up to 12) partitions called **logical partitions**

- Hard disks with a GUID Partition Table (GPT) can have up to 128 primary partitions
    - No need for extended partitions and logical drives

Partitioning



Partitioning

- Some frequently used command line utilities for partitioning hard disks:
  - **`fdisk`** – **F**ixed **Disk**
  - **`cfdisk`** – **C**urses **fdisk**
    - Curses is a terminal control library giving it a different interface than fdisk
  - **`parted`** – GNU **Part**ition **Ed**itor
- We’ll focus on using **`fdisk`**

Partitioning

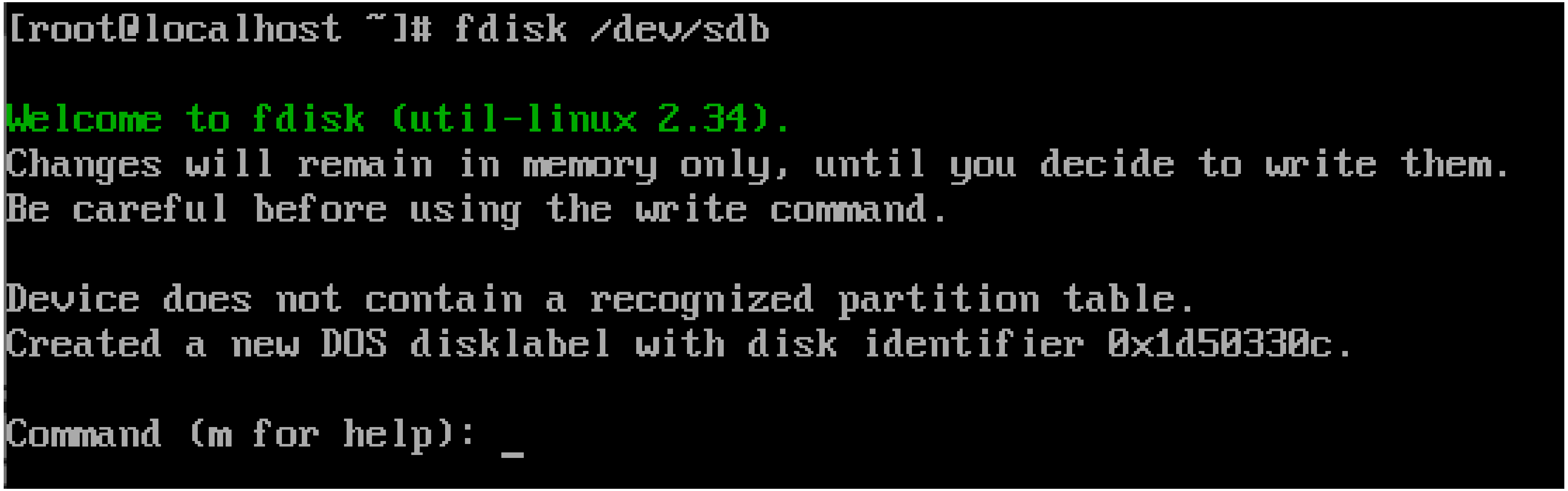
- The **fdisk** utility is a command line, menu-driven program that allows modifying, creating, and deleting partitions on a storage drive.

```
fdisk [options] device_file
```

- For example, to start the utility to edit/create/delete partitions on the second SATA hard drive:

```
fdisk /dev/sdb
```

Partitioning



```
[root@localhost ~]# fdisk /dev/sdb

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 0x1d50330c.

Command (m for help): _
```

This was a brand new virtual disk attached as a second hard drive in the virtual machine

Partitioning

- Common **`fdisk`** options:
    - **`-b`** Specifies the number of drive sectors
    - **`-H`** Specifies the number of drive heads
    - **`-S`** Specifies the number of sectors per track
    - **`-s`** Print the partition size in blocks
    - **`-l`** List the partition table for the device

- We will use the program’s menus instead of using the options.

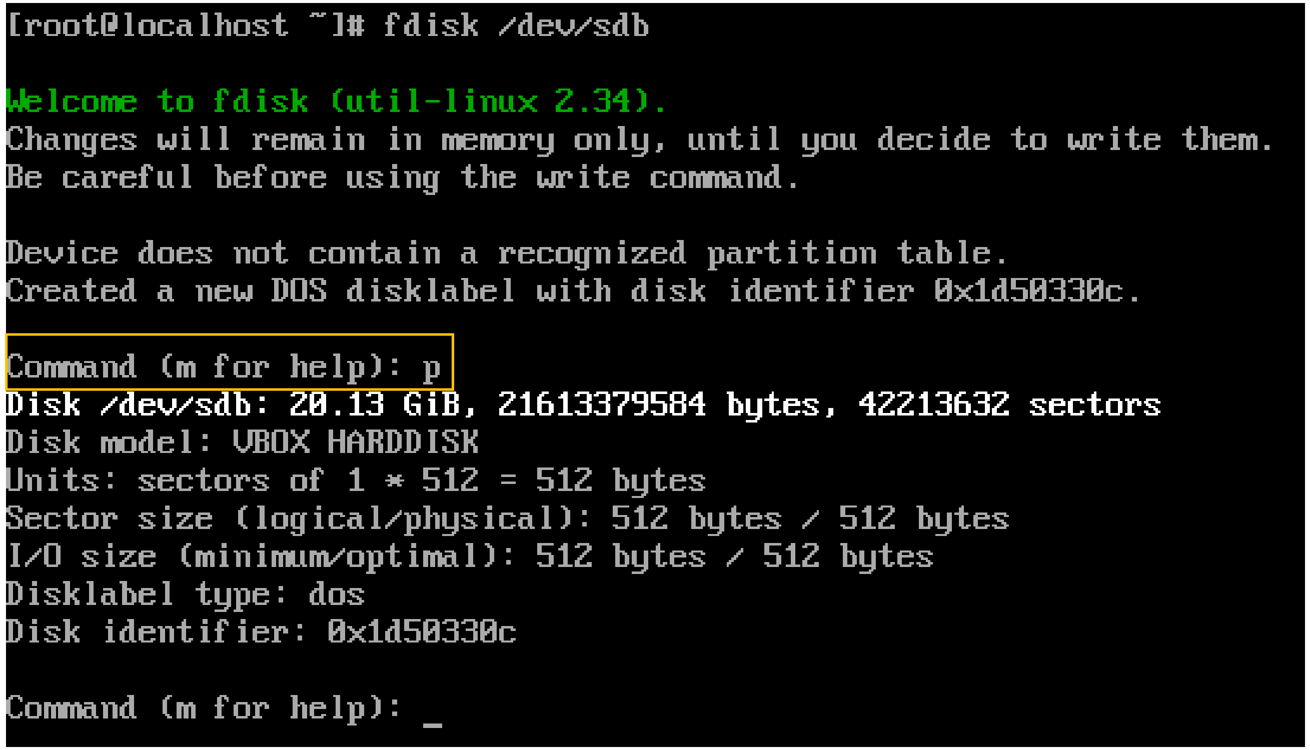
Partitioning

- Common **`fdisk`** menu options:
  - **`n`** Create a **n**ew partition
  - **`d`** **D**elete a partition
  - **`p`** List the existing **p**artitions
  - **`w`** **W**rite the changes to the drive and exit
  - **`m`** Help **m**anual
  - **`q`** **Q**uit/exit fdisk without writing changes to the drive

- This is what we will use in the **`fdisk`** program

Partitioning

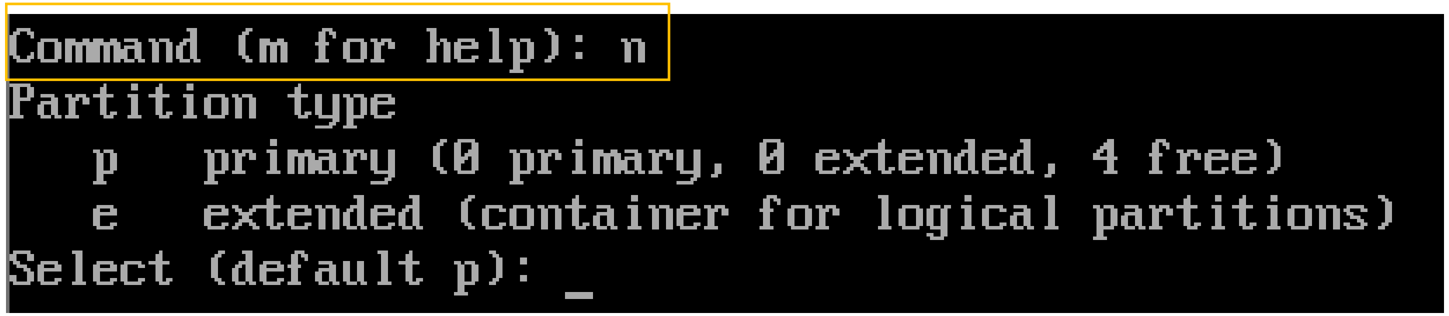
- Using the **p** menu option on a new disk lists no partitions, as none were created yet



```
[root@localhost ~]# fdisk /dev/sdb

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 0x1d50330c.

Command (m for help): p
Disk /dev/sdb: 20.13 GiB, 21613379584 bytes, 42213632 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: 0x1d50330c

Command (m for help): _
```

Partitioning

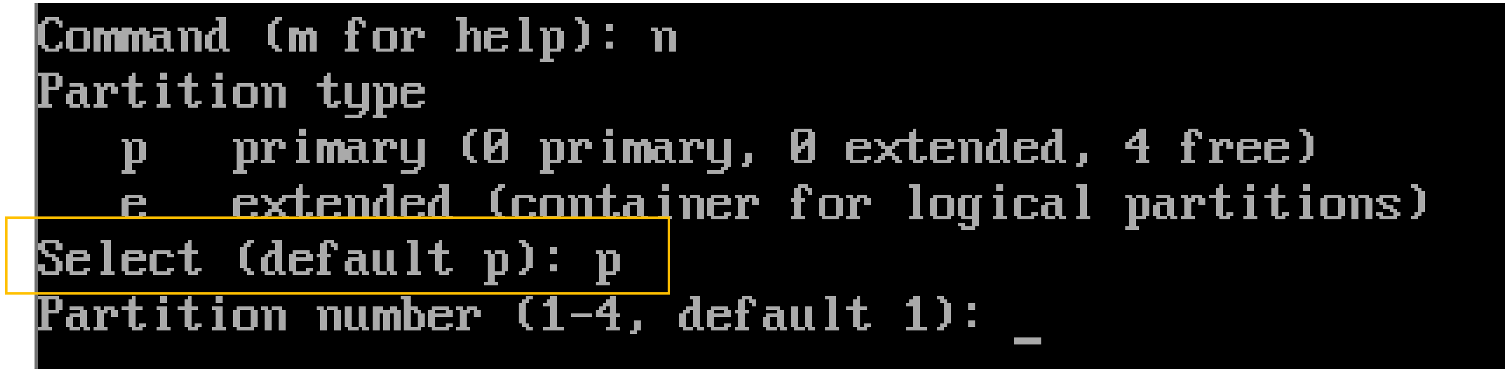
- Using the **n** menu option will begin creating a new partition.
  - We can then choose to make this a primary or extended partition



```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): _
```

Partitioning

- After choosing to make it a primary partition, the option is given to identify the partition number. **n** menu option will begin creating a new partition.
  - Can be given a number 1-4 (All four are available since no others were created yet)



```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): _
```

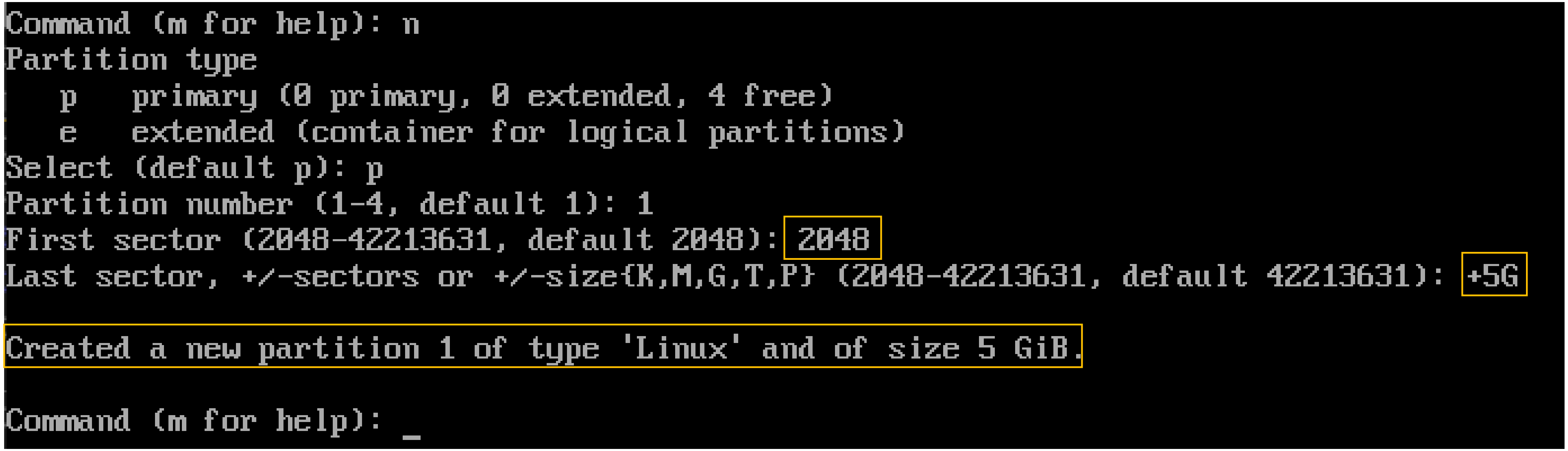
Partitioning

- Then, you choose the first disk sector of the partition.
  - The default will be the first available sector

- Next, you choose the last sector of the partition.
  - By default, it suggests the last available sector which would take up the rest of the disk.
  - Alternatively you can specify the size in kilobytes, megabytes, gigabytes, etc.

- For this example, I will start at the first available sector up through 5 gigabytes worth of sectors.

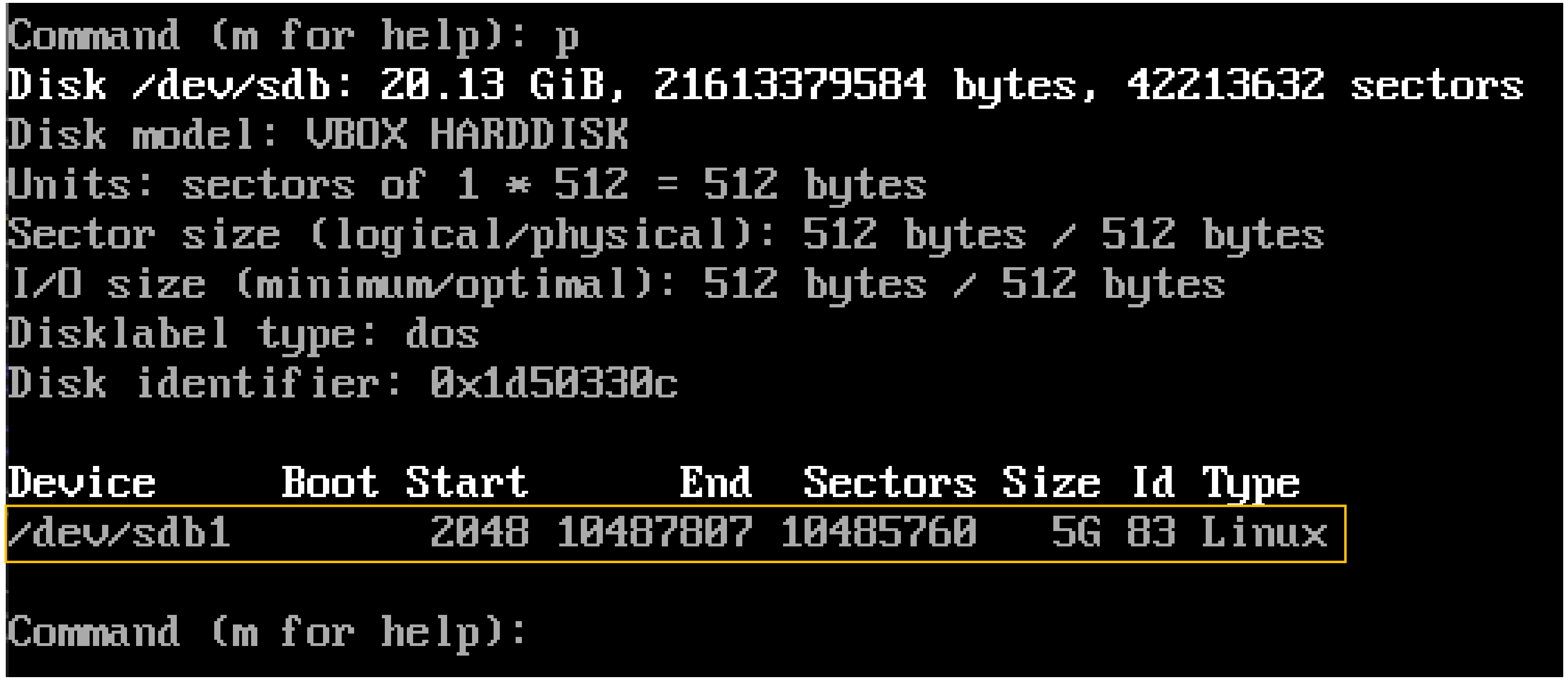
Partitioning



```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-42213631, default 2048): 2048
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-42213631, default 42213631): +5G

Created a new partition 1 of type 'Linux' and of size 5 GiB.

Command (m for help): _
```

Partitioning

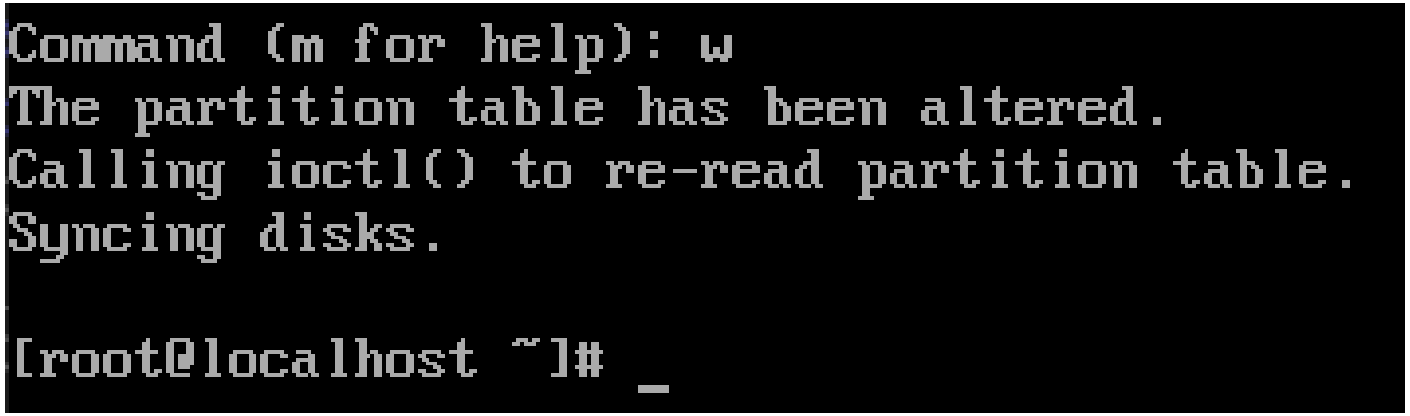
- Using the **p** menu option will display this new partition



```
Command (m for help): p
Disk /dev/sdb: 20.13 GiB, 21613379584 bytes, 42213632 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: 0x1d50330c

Device     Boot Start      End  Sectors Size Id Type
/dev/sdb1        2048 10487807 10485760   5G 83 Linux

Command (m for help):
```

Partitioning

- To write this change to the hard disk, the **w** menu option is used
  - **fdisk** is exited when the partition table is written



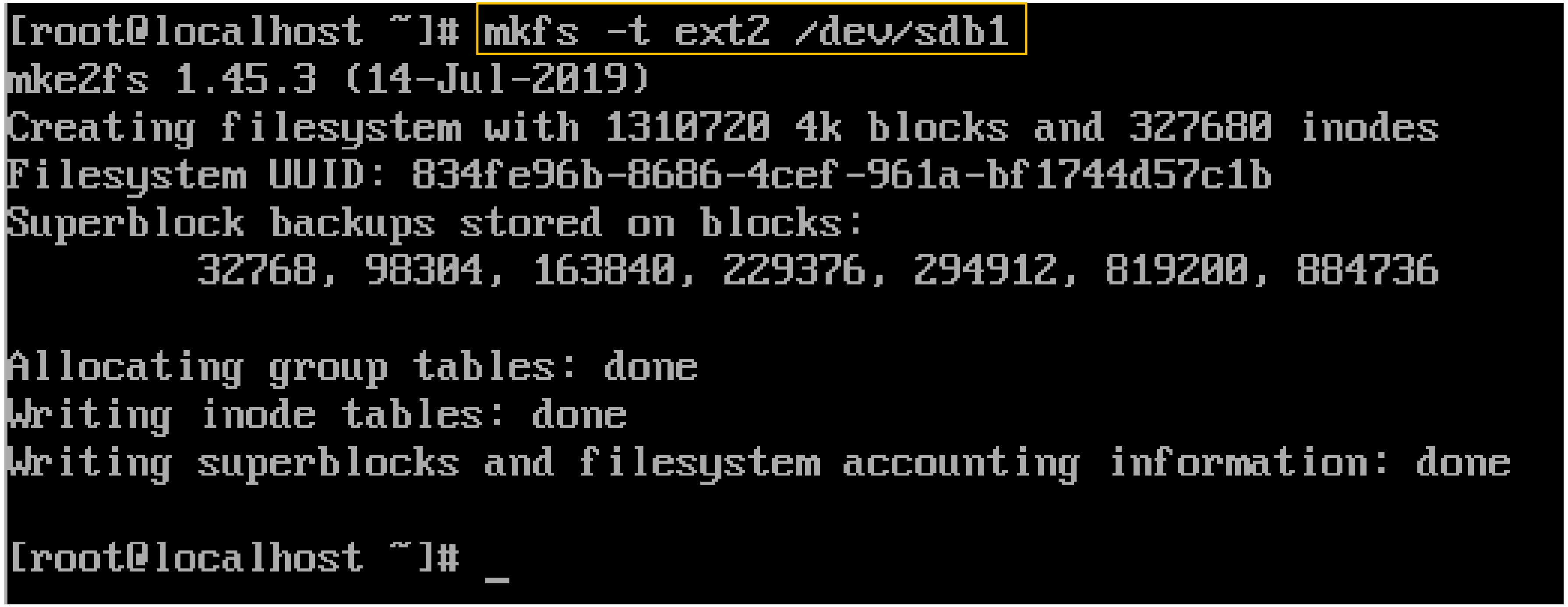
```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

[root@localhost ~]# _
```

Formatting

- Before data can be stored to a disk/partition, it must be formatted with a file system.

- The **`mkfs`** command is used to **m**a**k**e a **f**ile**s**ystem on a partition.
  - The most common option used is the **`-t`** option, which specifies the type of filesystem.

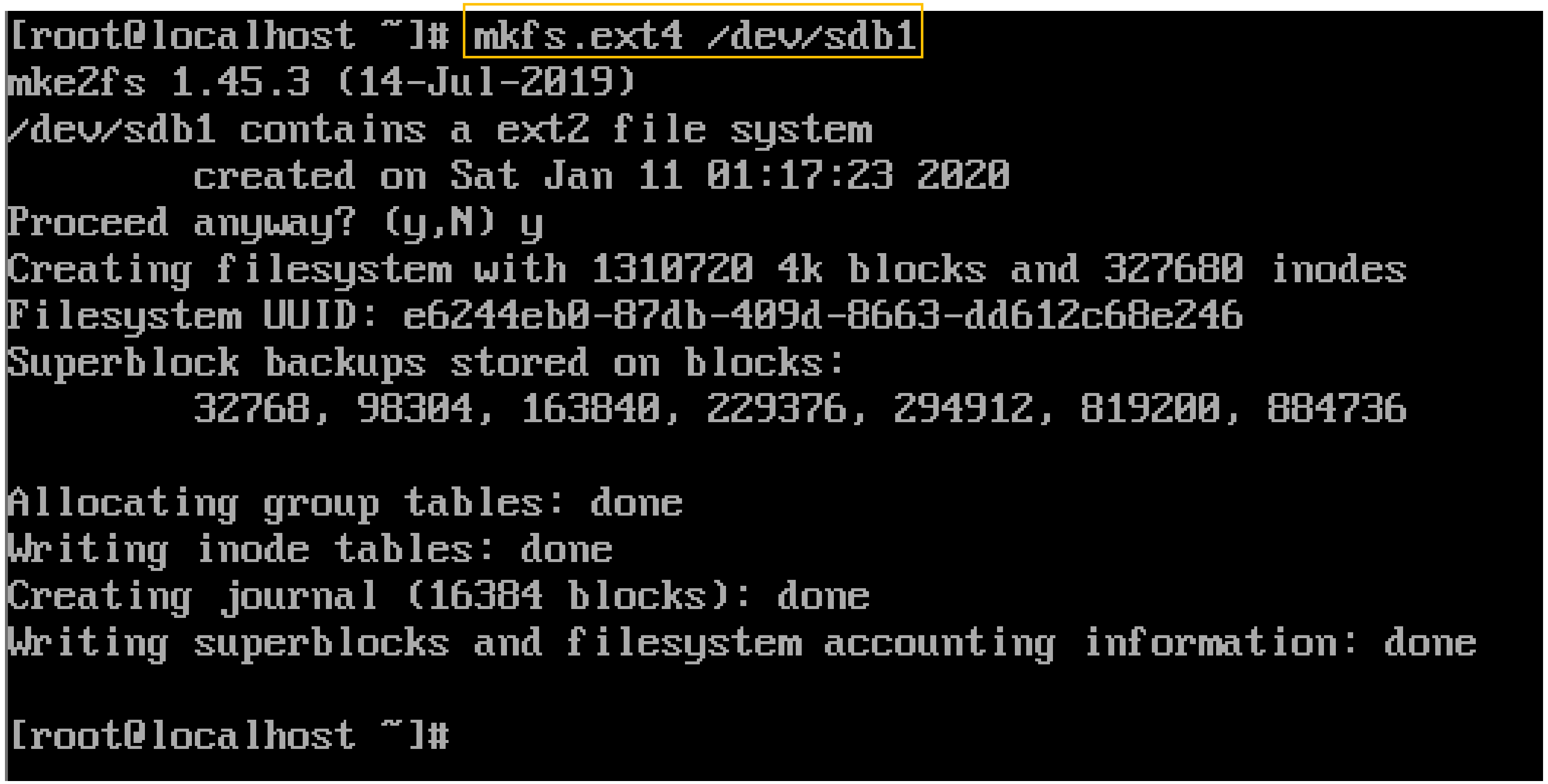
Formatting



```
[root@localhost ~]# mkfs -t ext2 /dev/sdb1
mke2fs 1.45.3 (14-Jul-2019)
Creating filesystem with 1310720 4k blocks and 327680 inodes
Filesystem UUID: 834fe96b-8686-4cef-961a-bf1744d57c1b
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

[root@localhost ~]# _
```

Formatting

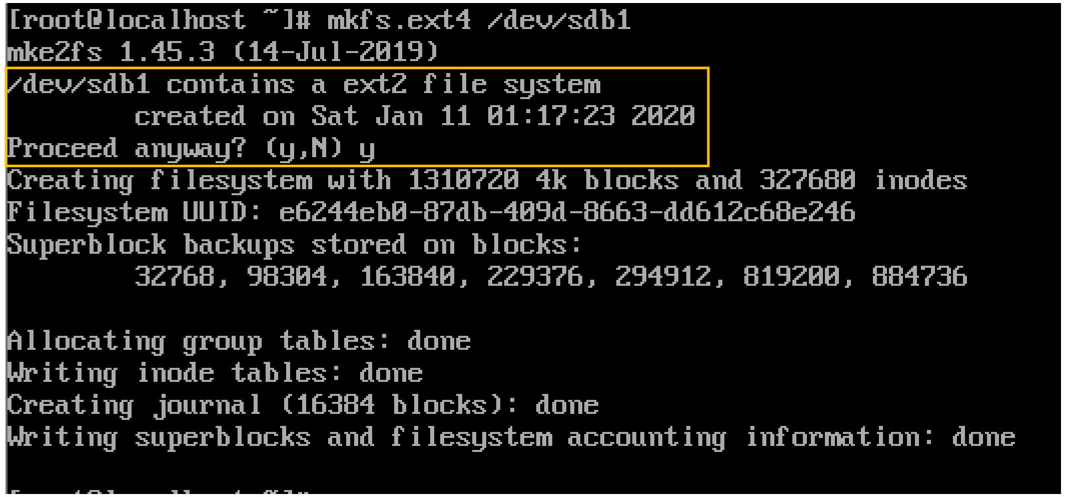
- An alternative is to use **`mkfs.`*`filesystemtype`***



```
[root@localhost ~]# mkfs.ext4 /dev/sdb1
mke2fs 1.45.3 (14-Jul-2019)
/dev/sdb1 contains a ext2 file system
        created on Sat Jan 11 01:17:23 2020
Proceed anyway? (y,N) y
Creating filesystem with 1310720 4k blocks and 327680 inodes
Filesystem UUID: e6244eb0-87db-409d-8663-dd612c68e246
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

[root@localhost ~]#
```

Formatting

- Formatting a partition will cause any existing data on the partition to become inaccessible.
  - Specialized recovery software would be needed to retrieve the lost data



```
[root@localhost ~]# mkfs.ext4 /dev/sdb1
mke2fs 1.45.3 (14-Jul-2019)
/dev/sdb1 contains a ext2 file system
        created on Sat Jan 11 01:17:23 2020
Proceed anyway? (y,N) y
Creating filesystem with 1310720 4k blocks and 327680 inodes
Filesystem UUID: e6244eb0-87db-409d-8663-dd612c68e246
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

Mounting

- Before a partition can be used in a Linux system, the partition must be formatted with a filesystem and then must be ***mounted***
  - The term originated when data was stored on large reels of magnetic tape that needed to be physically lifted/attached onto computers.

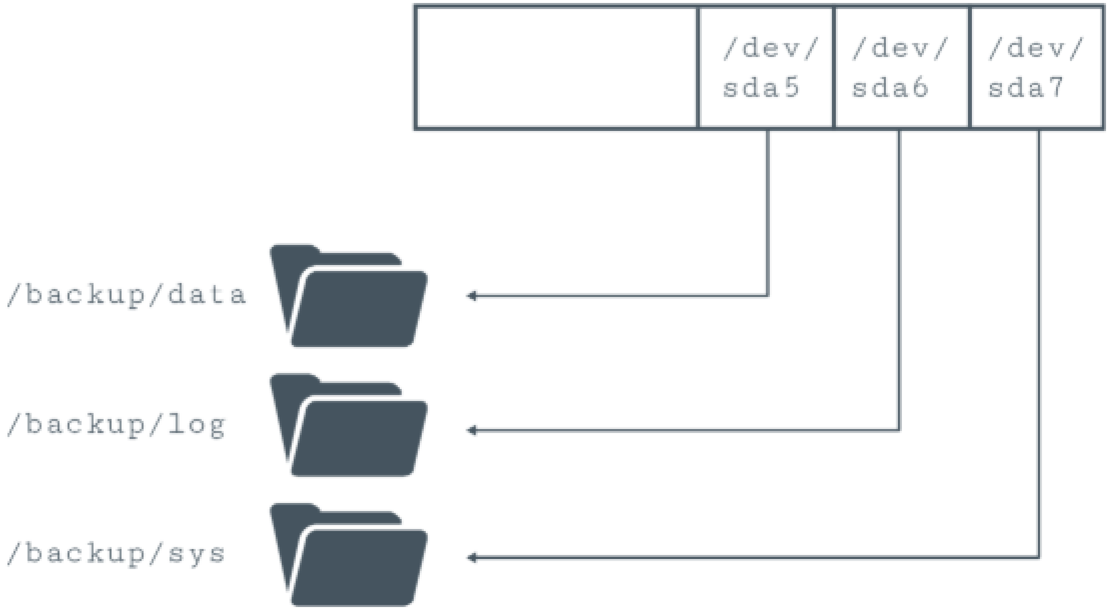
- Today, it refers to when a storage device is made accessible to the system.

Mounting

- Where the storage device is mounted is referred to its **mount point**
    - A directory in the directory tree

- When users create files and subdirectories in the mount point, the files and subdirectories are created *in the filesystem of the mounted device*.

Mounting



Mounting

- When the system is first booted, a filesystem must be mounted to the / directory
    - We know this as the root directory, but the filesystem mounted there is the **root filesystem**

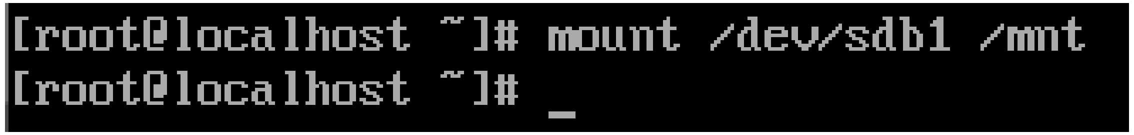
- Other devices can be mounted to other directories at boot time
    - We'll come back to this soon

Mounting

- The **mount** command is used to mount a storage device

  mount *devicefile mountpoint*

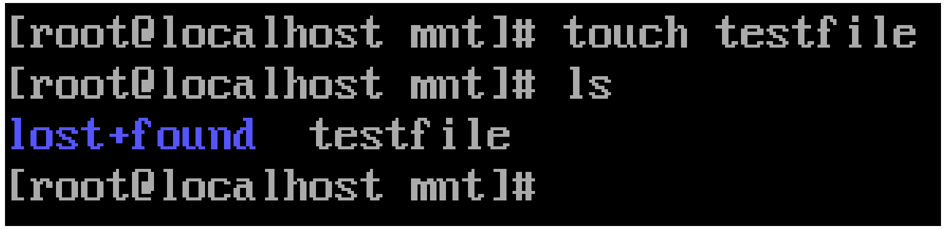
- The command **mount /dev/sdb1 /mnt** would mount the sdb1 partition to the /mnt directory



```
[root@localhost ~]# mount /dev/sdb1 /mnt
[root@localhost ~]# _
```

Mounting

- In this example, any data stored in /mnt will be written to sdb1



```
[root@localhost mnt]# touch testfile
[root@localhost mnt]# ls
lost+found  testfile
[root@localhost mnt]#
```

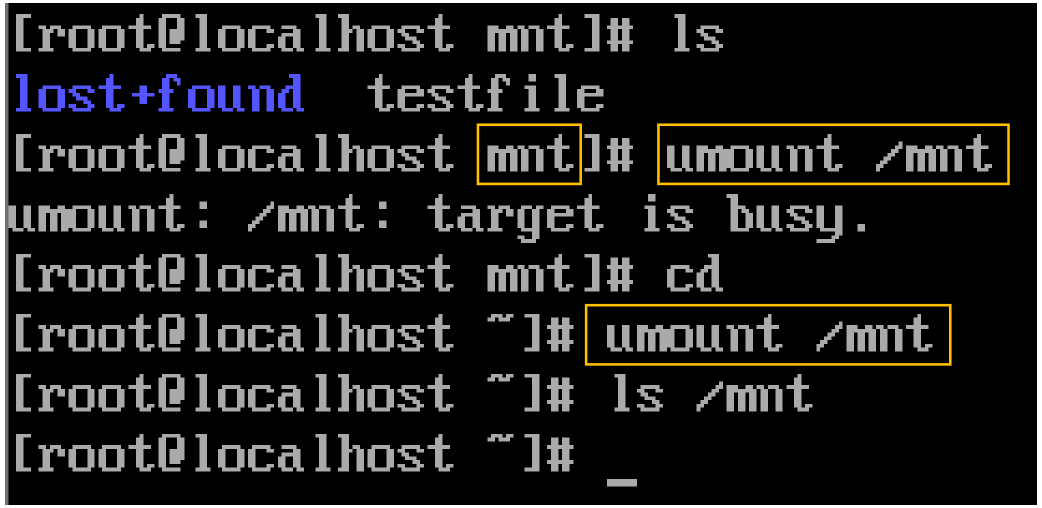
Mounting

- To **unmount** a device, use the **umount** command

```
umount mountpoint
```

- A drive in use will not be able to be unmounted
  - This includes if you are currently in (or in a subdirectory of) the mount point

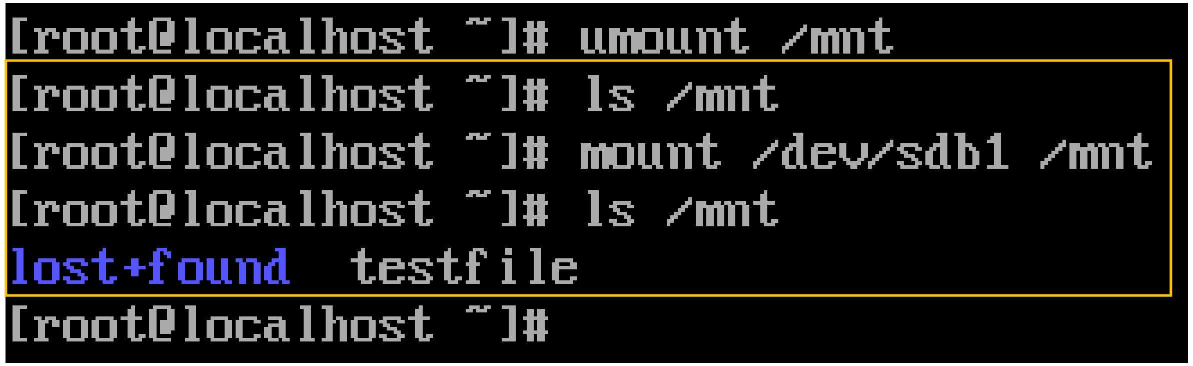
Mounting



```
[root@localhost mnt]# ls
lost+found  testfile
[root@localhost mnt]# umount /mnt
umount: /mnt: target is busy.
[root@localhost mnt]# cd
[root@localhost ~]# umount /mnt
[root@localhost ~]# ls /mnt
[root@localhost ~]# _
```

Mounting

- If I were to remount the disk, the files stored on the disk will be accessible again.



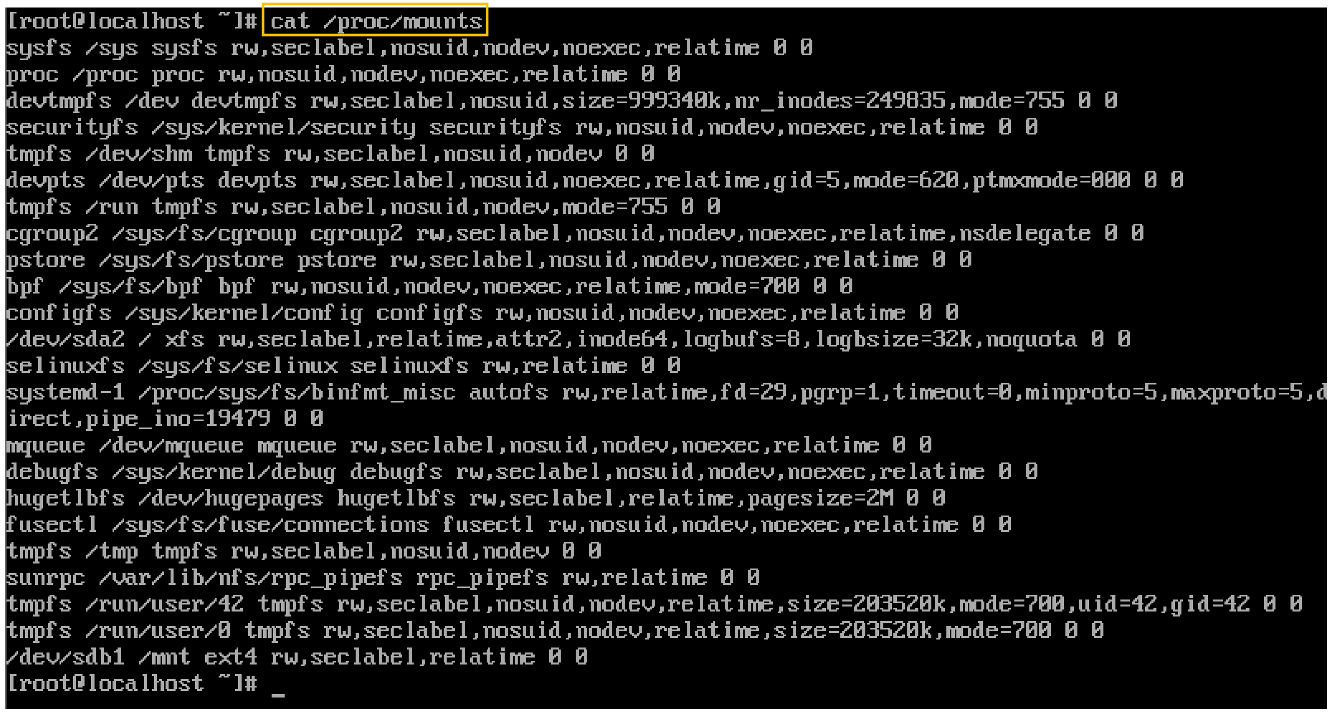
```
[root@localhost ~]# umount /mnt
[root@localhost ~]# ls /mnt
[root@localhost ~]# mount /dev/sdb1 /mnt
[root@localhost ~]# ls /mnt
lost+found  testfile
[root@localhost ~]#
```

Mounting

- Some important notes:
  - Disks that are currently mounted cannot be formatted. The disk must be unmounted when it is to be formatted.
  - Disks that are currently mounted cannot have their partition table changed. The disk must be unmounted when its partitions are to be added/modified/deleted.

Administration (File System Info)

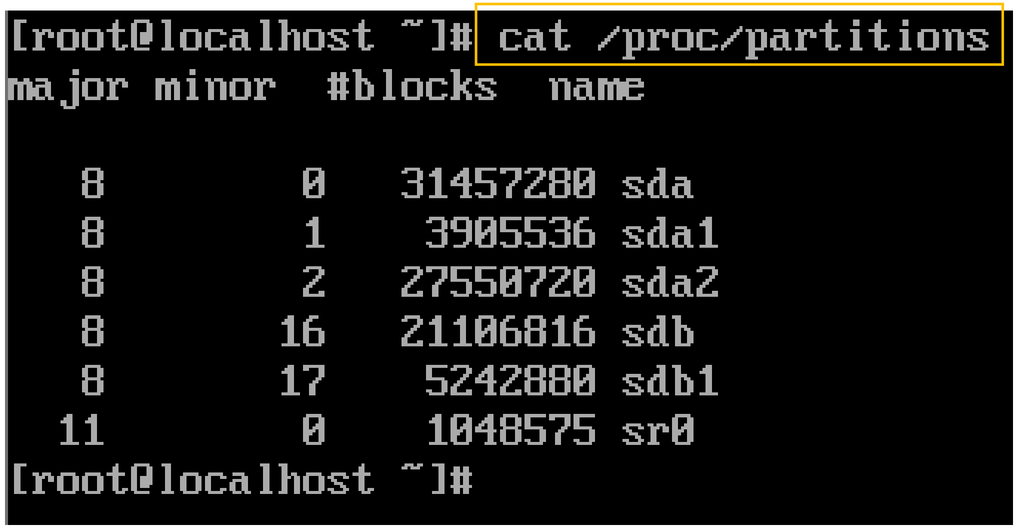
- The **`/proc/mounts`** file contains a list of active mount points



```
[root@localhost ~]# cat /proc/mounts
sysfs /sys sysfs rw,seclabel,nosuid,nodev,noexec,relatime 0 0
proc /proc proc rw,nosuid,nodev,noexec,relatime 0 0
devtmpfs /dev devtmpfs rw,seclabel,nosuid,size=999340k,nr_inodes=249835,mode=755 0 0
securityfs /sys/kernel/security securityfs rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /dev/shm tmpfs rw,seclabel,nosuid,nodev 0 0
devpts /dev/pts devpts rw,seclabel,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
tmpfs /run tmpfs rw,seclabel,nosuid,nodev,mode=755 0 0
cgroup2 /sys/fs/cgroup cgroup2 rw,seclabel,nosuid,nodev,noexec,relatime,nsdelegate 0 0
pstore /sys/fs/pstore pstore rw,seclabel,nosuid,nodev,noexec,relatime 0 0
bpf /sys/fs/bpf bpf rw,nosuid,nodev,noexec,relatime,mode=700 0 0
configfs /sys/kernel/config configfs rw,nosuid,nodev,noexec,relatime 0 0
/dev/sda2 / xfs rw,seclabel,relatime,attr2,inode64,logbufs=8,logbsize=32k,noquota 0 0
selinuxfs /sys/fs/selinux selinuxfs rw,relatime 0 0
systemd-1 /proc/sys/fs/binfmt_misc autofs rw,relatime,fd=29,pgrp=1,timeout=0,minproto=5,maxproto=5,d
irect,pipe_ino=19479 0 0
mqueue /dev/mqueue mqueue rw,seclabel,nosuid,nodev,noexec,relatime 0 0
debugfs /sys/kernel/debug debugfs rw,seclabel,nosuid,nodev,noexec,relatime 0 0
hugetlbfs /dev/hugepages hugetlbfs rw,seclabel,relatime,pagesize=2M 0 0
fusectl /sys/fs/fuse/connections fusectl rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /tmp tmpfs rw,seclabel,nosuid,nodev 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw,relatime 0 0
tmpfs /run/user/42 tmpfs rw,seclabel,nosuid,nodev,relatime,size=203520k,mode=700,uid=42,gid=42 0 0
tmpfs /run/user/0 tmpfs rw,seclabel,nosuid,nodev,relatime,size=203520k,mode=700 0 0
/dev/sdb1 /mnt ext4 rw,seclabel,relatime 0 0
[root@localhost ~]# _
```

Administration (File System Info)

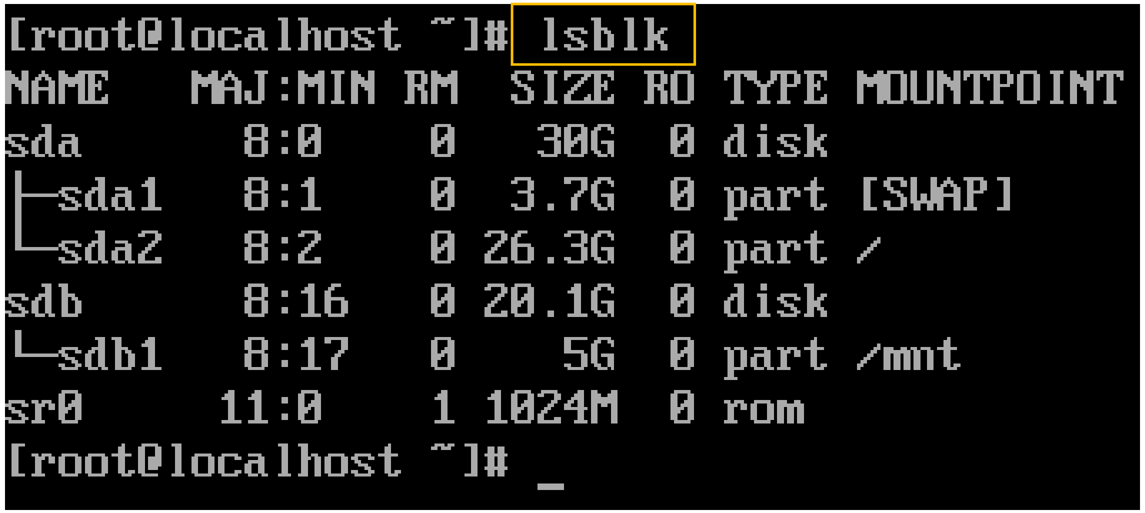
- The **`/proc/partitions`** file contains a list of active partitions



```
[root@localhost ~]# cat /proc/partitions
major minor  #blocks  name

   8        0   31457280 sda
   8        1    3905536 sda1
   8        2   27550720 sda2
   8       16   21106816 sdb
   8       17    5242880 sdb1
  11        0    1048575 sr0
[root@localhost ~]#
```

Administration (File System Info)

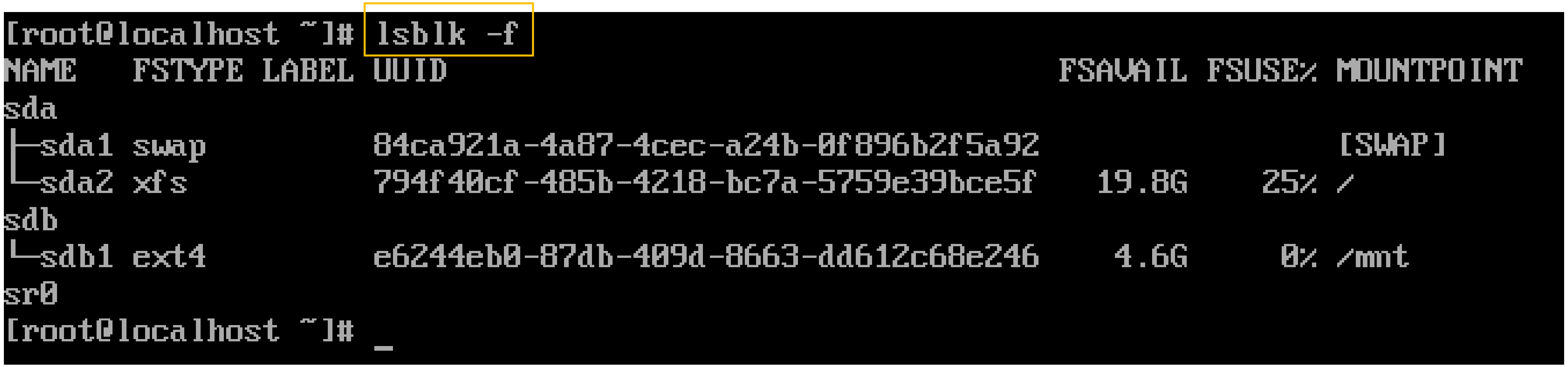
- The **`lsblk`** command **lis**ts all **bl**oc**k** storage devices



```
[root@localhost ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   30G  0 disk
├─sda1   8:1    0  3.7G  0 part [SWAP]
└─sda2   8:2    0 26.3G  0 part /
sdb      8:16   0 20.1G  0 disk
└─sdb1   8:17   0    5G  0 part /mnt
sr0     11:0    1 1024M  0 rom
[root@localhost ~]# _
```

Administration (File System Info)

- The **`-f`** option displays additional information about the block device



```
[root@localhost ~]# lsblk -f
NAME   FSTYPE LABEL UUID                                 FSAVAIL FSUSE% MOUNTPOINT
sda
├─sda1 swap         84ca921a-4a87-4cec-a24b-0f896b2f5a92                [SWAP]
└─sda2 xfs          794f40cf-485b-4218-bc7a-5759e39bce5f   19.8G    25% /
sdb
└─sdb1 ext4         e6244eb0-87db-409d-8663-dd612c68e246    4.6G     0% /mnt
sr0
[root@localhost ~]# _
```

Administration (File System Info)

- The **dumpe2fs** command displays information about an ext 2/3/4 file system.

```
dumpe2fs [options] device_file
```

- Commonly used options:
- **-x** Prints a report about block numbers in the file system
- **-b** Prints the bad blocks in the file system

Administration (File System Info)

- The **`xfs_info`** command displays information about an xfs file system.

```
xfs_info [options] device_file
```

Administration (Resize File System)

- The **resize2fs** command allows for resizing an ext 2/3/4 file system.

```
resize2fs device_file size
resize2fs /dev/sdb1 4G
```

- An ext file system can be enlarged without unmounting it
  - Must be unmounted to shrink the file system
- Does **not** resize partitions, only the file system.

Administration (Resize File System)

- The **xfs_growfs** command allows for resizing an xfs file system.

```
xfs_growfs device_file
xfs_growfs /dev/sdb1
```

- To extend to a specified size:

```
xfs_growfs device_file –D number_of_blocks
```

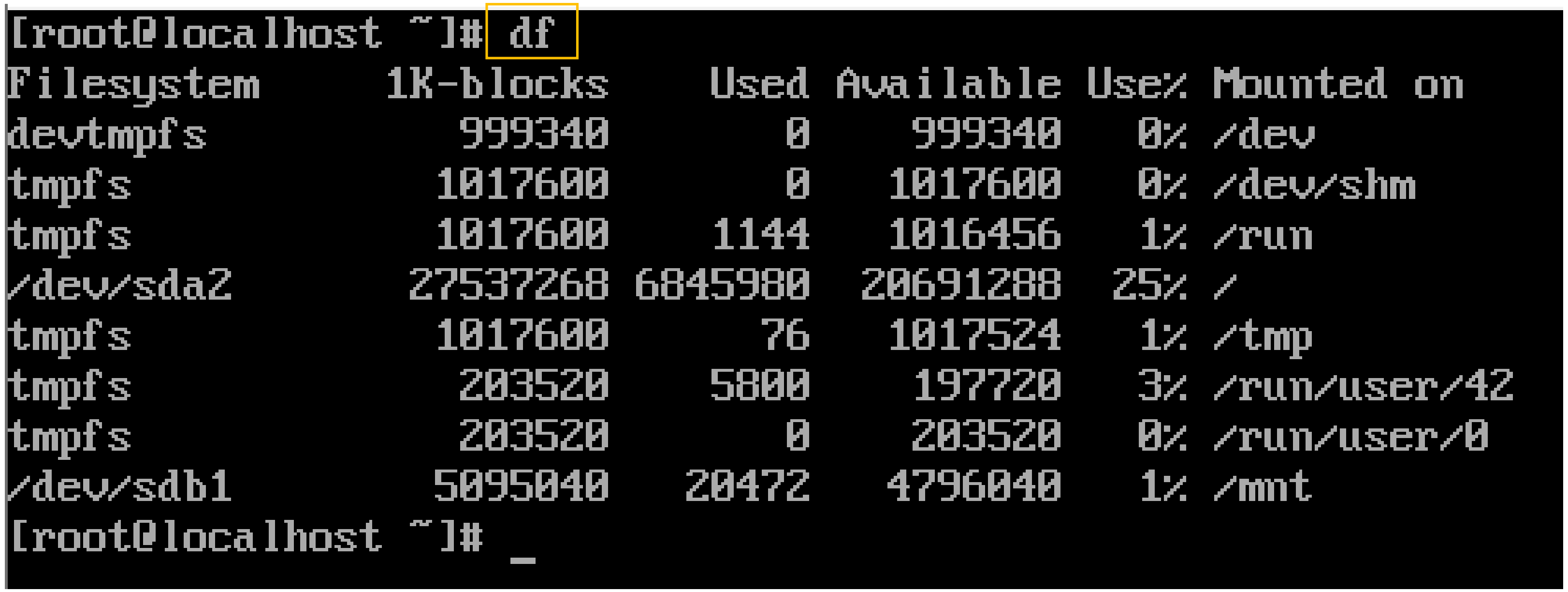
Disk Usage

- The two commands commonly used for displaying disk space utilization are **`df`** and **du**

- The **`df`** command displays the **d**isk’s **f**ree space
- The **du** command displays the **d**isk’s **u**sage

Disk Usage

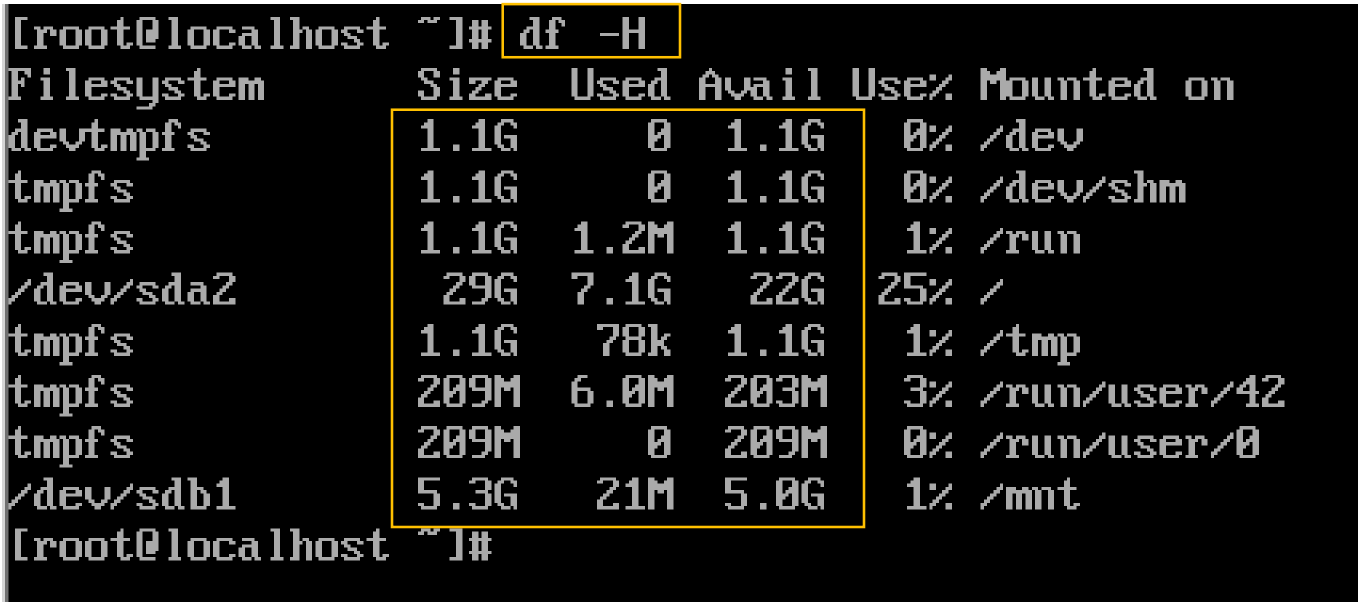
- The **df** command (without any options)



```
[root@localhost ~]# df
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          999340       0    999340   0% /dev
tmpfs            1017600       0   1017600   0% /dev/shm
tmpfs            1017600    1144   1016456   1% /run
/dev/sda2       27537268 6845980  20691288  25% /
tmpfs            1017600      76   1017524   1% /tmp
tmpfs             203520    5800    197720   3% /run/user/42
tmpfs             203520       0    203520   0% /run/user/0
/dev/sdb1        5095040   20472   4796040   1% /mnt
[root@localhost ~]# _
```

Disk Usage

- The **df** command with the **-H** ("human readable") option displays the space using byte units



```
[root@localhost ~]# df -H
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        1.1G     0  1.1G   0% /dev
tmpfs           1.1G     0  1.1G   0% /dev/shm
tmpfs           1.1G  1.2M  1.1G   1% /run
/dev/sda2        29G  7.1G   22G  25% /
tmpfs           1.1G   78k  1.1G   1% /tmp
tmpfs           209M  6.0M  203M   3% /run/user/42
tmpfs           209M     0  209M   0% /run/user/0
/dev/sdb1       5.3G   21M  5.0G   1% /mnt
[root@localhost ~]#
```

Disk Usage

- **df** is good for an overall picture of the drives/partitions

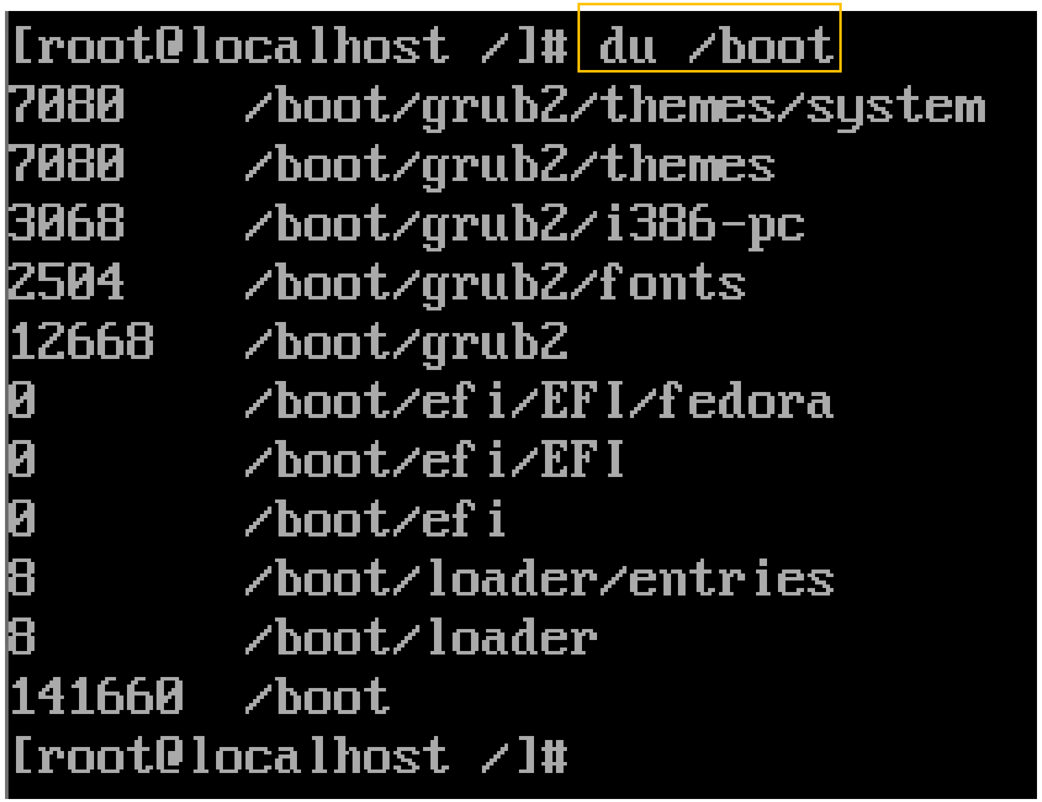
- For more granular disk utilization details, we use the **du** command.

```
du [options] directory
```

- The **du** command recursively traverses the directory
  - If a directory is not specified, the current location is traversed.

Disk Usage

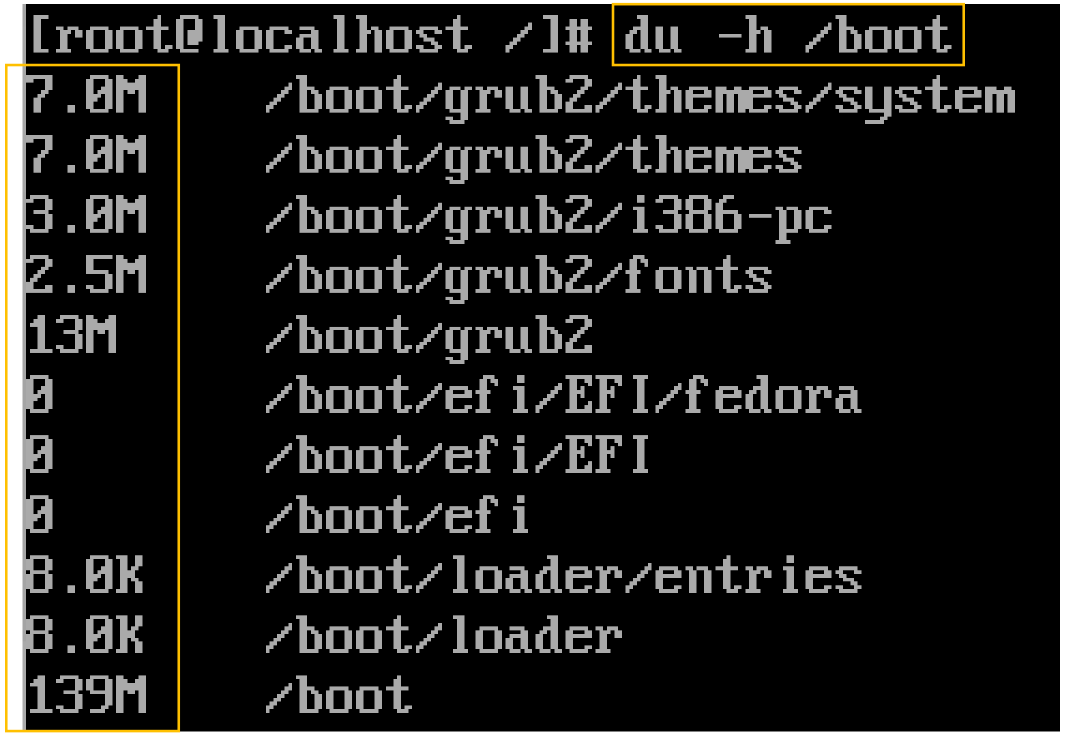
- Utilization of the /boot directory:



```
[root@localhost /]# du /boot
7080    /boot/grub2/themes/system
7080    /boot/grub2/themes
3068    /boot/grub2/i386-pc
2504    /boot/grub2/fonts
12668   /boot/grub2
0       /boot/efi/EFI/fedora
0       /boot/efi/EFI
0       /boot/efi
8       /boot/loader/entries
8       /boot/loader
141660  /boot
[root@localhost /]#
```

Disk Usage

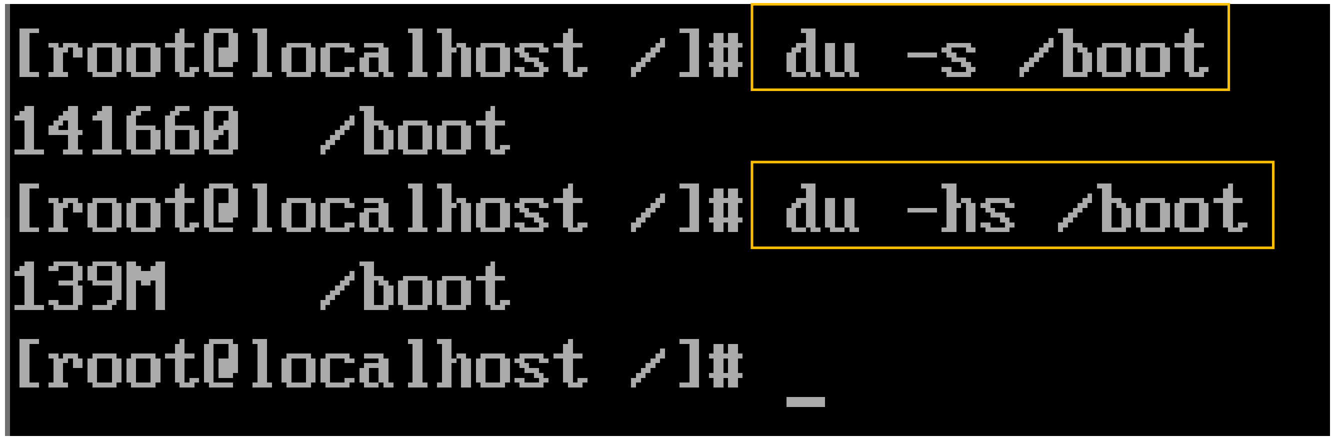
- Use the **-h** option for human readable sizes:



```
[root@localhost /]# du -h /boot
7.0M    /boot/grub2/themes/system
7.0M    /boot/grub2/themes
3.0M    /boot/grub2/i386-pc
2.5M    /boot/grub2/fonts
13M     /boot/grub2
0       /boot/efi/EFI/fedora
0       /boot/efi/EFI
0       /boot/efi
8.0K    /boot/loader/entries
8.0K    /boot/loader
139M    /boot
```

Disk Usage

- Use the **`-s`** option for a summary of the directory:



```
[root@localhost /]# du -s /boot
141660  /boot
[root@localhost /]# du -hs /boot
139M    /boot
[root@localhost /]# _
```

Disk Checking and Repair

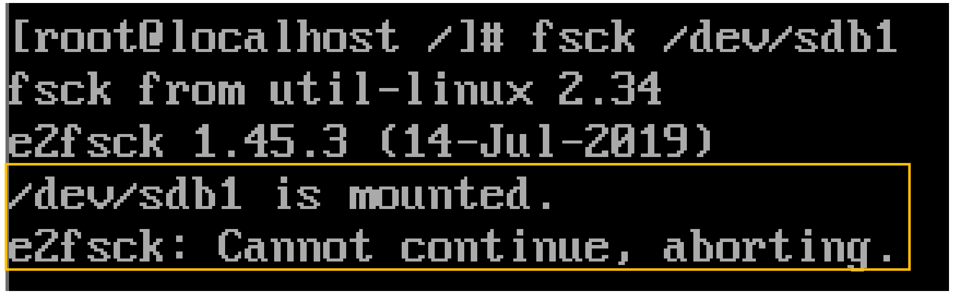
- File system errors are commonly caused by power failures, hardware failures, and the improper shutdown of the system.
- Disk checks can be performed at start-up (every time or every *X* number of days) or performed manually.
- Disk checks are important to maintaining the integrity (correctness and validity) of the file system.

Disk Checking and Repair

- The **`fsck`** command is used to perform a **f**ile **s**ystem **c**hec**k** and can be used to correct any errors.

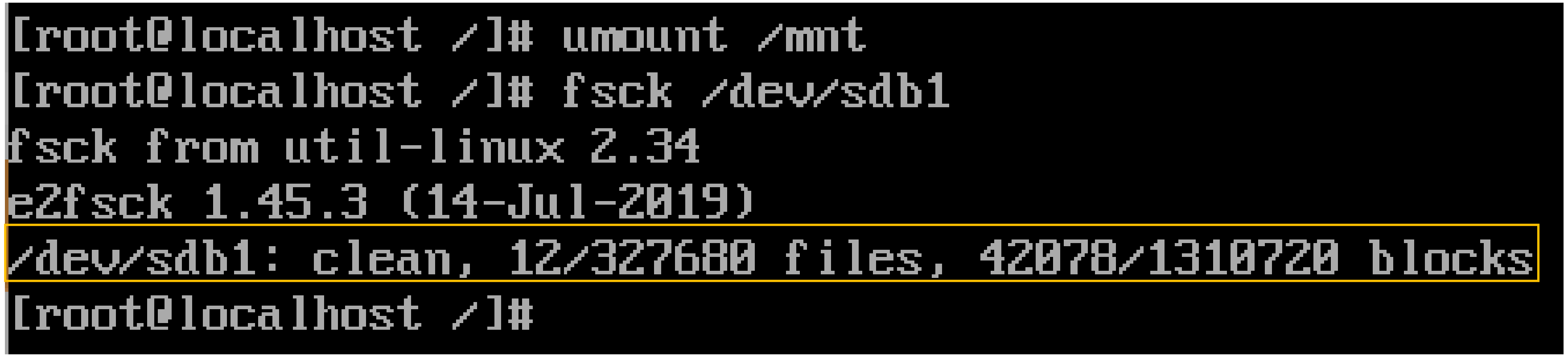
`fsck [options]` *`device_file`*

- The drive must be unmounted before the check can be performed.



```
[root@localhost /]# fsck /dev/sdb1
fsck from util-linux 2.34
e2fsck 1.45.3 (14-Jul-2019)
/dev/sdb1 is mounted.
e2fsck: Cannot continue, aborting.
```

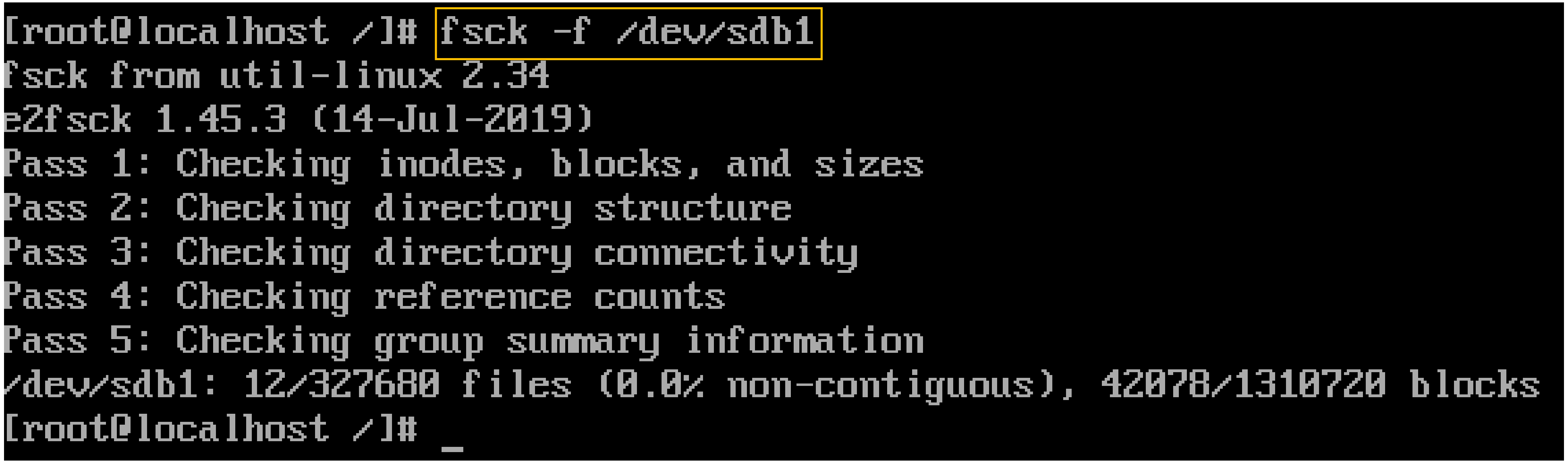
Disk Checking and Repair



```
[root@localhost /]# umount /mnt
[root@localhost /]# fsck /dev/sdb1
fsck from util-linux 2.34
e2fsck 1.45.3 (14-Jul-2019)
/dev/sdb1: clean, 12/327680 files, 42078/1310720 blocks
[root@localhost /]#
```

Disk Checking and Repair

- Use the **`-f`** option to perform a full check:



```
[root@localhost /]# fsck -f /dev/sdb1
fsck from util-linux 2.34
e2fsck 1.45.3 (14-Jul-2019)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sdb1: 12/327680 files (0.0% non-contiguous), 42078/1310720 blocks
[root@localhost /]# _
```

Disk Checking and Repair

- If **`fsck`** finds any problems, it will ask you if it can fix the error.
- To avoid these prompts and allow **`fsck`** to automatically repair any errors, use the **`-y`** option.
- If there are files it cannot repair, they will be placed in the **lost+found** directory of the file system.
  - Only ext filesystems have a lost+found directory

Disk Checking and Repair

- The **`fsck`** command automatically calls the appropriate tool to check that file system type.

- For ext filesystems, it uses **`e2fsck`**

- For xfs filesystems, it uses **`xfs_repair`**

Disk Checking and Repair

- File systems have numerous parameters that can be configured or “tuned”

- For ext filesystems, the **`tune2fs`** command is used.

- For xfs filesystems, the **`xfs_admin`** command is used.

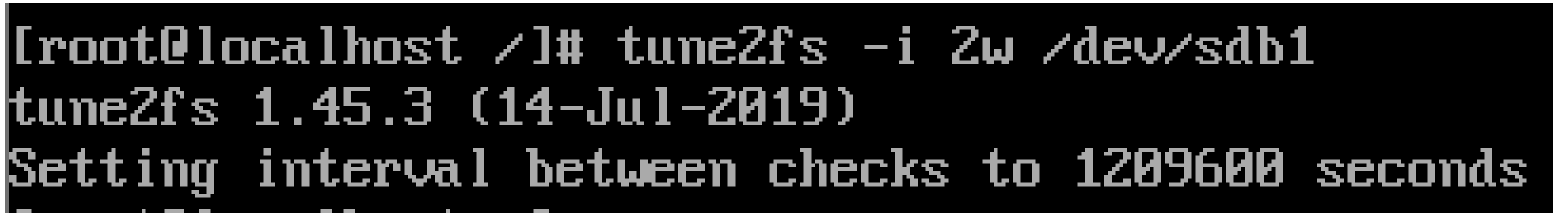
Disk Checking and Repair

- **`tune2fs`** has numerous options, but we'll focus on two:

- **`-i`** Specifies the number of days/months/weeks between automatic filesystem checks
- **`-c`** Specifies the maximum number of times the drive can be mounted between automatic file system checks

Disk Checking and Repair

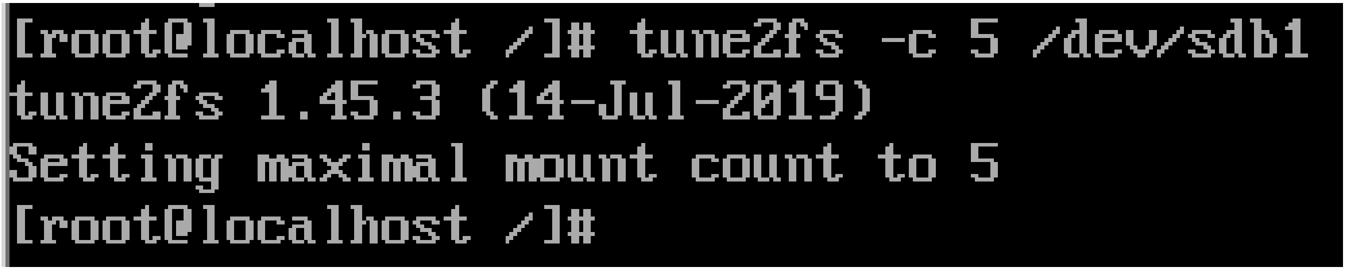
- **tune2fs –i 2w /dev/sdb1** will force an automatic filesystem check on /dev/sdb1 every two weeks.



```
[root@localhost /]# tune2fs -i 2w /dev/sdb1
tune2fs 1.45.3 (14-Jul-2019)
Setting interval between checks to 1209600 seconds
```

Disk Checking and Repair

- **tune2fs –c 5 /dev/sdb1** will force an automatic filesystem check on /dev/sdb1 after every 5 mounts.



```
[root@localhost /]# tune2fs -c 5 /dev/sdb1
tune2fs 1.45.3 (14-Jul-2019)
Setting maximal mount count to 5
[root@localhost /]#
```

Mounting Filesystems at Boot

- The /etc/fstab file contains the **f**ile**s**ystem **tab**le that is used to mount devices at boot time.
- Each entry has six fields:
  - The device to mount
  - The device’s mount point
  - The filesystem type of the device
  - Mounting options
  - Dump number
  - Fsck number

Mounting Filesystems at Boot

- Device to mount:
  - Devices can be identified by one of the following:
    - The path to a device file
    - The filesystem’s UUID
    - The GPT partition UUID (PARTUUID)
    - The filesystem’s label

Mounting Filesystems at Boot

- Device’s mount point:
  - Simply specifies where to mount the filesystem

- The filesystem type of the device
  - For example, ext4 or xfs

Mounting Filesystems at Boot

- Mounting options
  - Options that would be provided if using the mount command manually (See the manual page for the mount command for the full list)
  - Commonly used options:
    - **`defaults`** Mounts the filesystem using default settings
    - **`ro`** Mounts the filesystem in a read only state
    - **`noauto`** Prevents the filesystem from being mounted at boot
  - Multiple options are separated by commas

Mounting Filesystems at Boot

- Dump number (0 or 1):
  - 0 indicates the filesystem should not be backed up
  - 1 indicates the filesystem should be backed up

- Fsck number (0, 1, or 2):
  - 0 indicates the filesystem should not be checked
  - 1 indicates the filesystem to be checked first
    - Usually the root filesystem
  - 2 indicates the filesystem(s) to be checked next
    - Checked in the order they appear in the fstab file

Mounting Filesystems at Boot

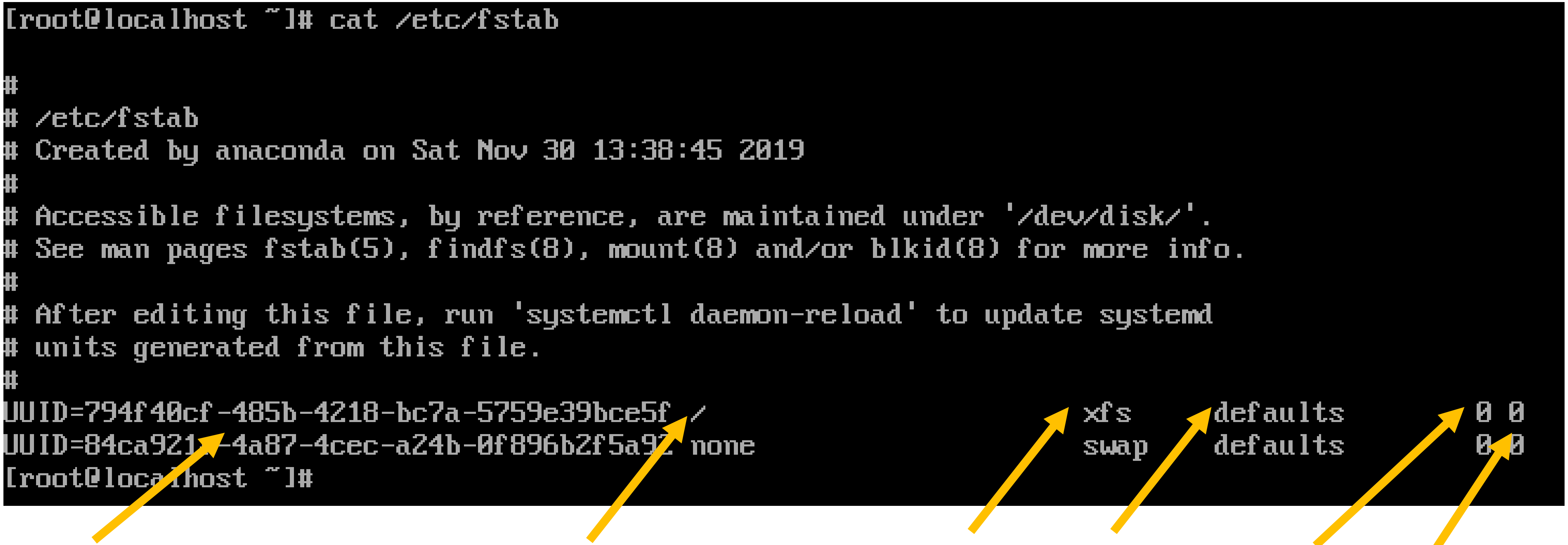
- The /etc/fstab file is also checked when issuing the mount command from the terminal.
- The mount command will use the options specified for that device (if it exists) in the /etc/fstab file.

Mounting Filesystems at Boot

- The /etc/fstab file can be edited using a text editor like vi.
  - Must have administrator privileges to save any changes.

- A command like **`cat`** can be used to display the contents of /etc/fstab

Mounting Filesystems at Boot



```
[root@localhost ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Sat Nov 30 13:38:45 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
UUID=794f40cf-485b-4218-bc7a-5759e39bce5f /                       xfs     defaults        0 0
UUID=84ca9211-4a87-4cec-a24b-0f896b2f5a92 none                    swap    defaults        0 0
[root@localhost ~]#
```

Device

Mount Point

FS Type

Options

Dump #

Fsck #

Mounting Filesystems at Boot

- Adding the following line to /etc/fstab

```
/dev/sdb1  /mnt  ext4  defaults  0  0
```

- Would mount /dev/sdb1 to /mnt at boot time.
  - Filesystem specified as ext4
  - No additional options set
  - No dump or checking set

Disk Quotas

- Quotas allow an administrator to allot and monitor the file system space that a user may use.

- The tools most commonly used are:

quotacheck	Creates the quota database files for a file system
edquota	Edits quotas
setquota	Sets quotas
repquota	Displays a quota report
quota	Allows regular users to check their quotas

Disk Quotas

- Quotas can restrict:
  - How many files and directories a user may create on a filesystem
  - The total size of all files owned by the user on a filesystem

- A **soft limit** is a quota that the user may exceed for some period of time (the default is seven days)

- A **hard limit** is a quota that the user may not exceed.