OPERATING SYSTEMS
CCGC-5000

btn

Module - 3

Agenda

Authentic information is available from the given resources in course outline and URL's mentioned from this slides, and this presentation is only supportive document to be read with the given resources and corrected accordingly if required..

- FileSystems, Files, FileSystem types
- UNIX/Linux Files & Directory
- File information, types, listing, wild card search
- File paths - Absolute and Relative
- File management
- File time stamps, stat, file Search
- File System layout, inodes, link files
- Storage device process
- Storage devices, pseudo devices, SWAP partition
- Partition, Storage device tools
- Create filesystem, mounting, /etc/fstab file.
- LVM, Stratis, VDO

Must read

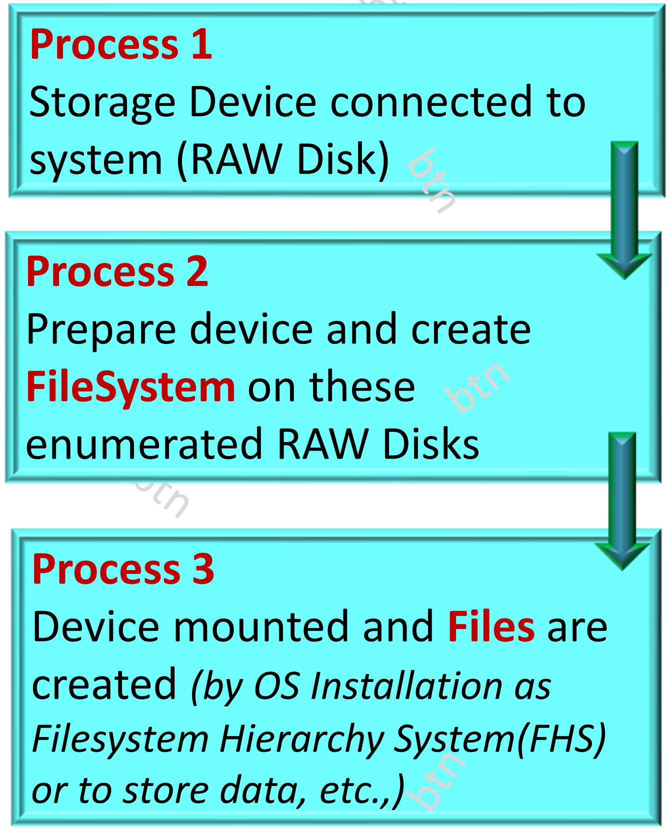
- Chapters 3,4,13,14,15 of RHEL8, 2nd Edition book
- RedHat documentation https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/

btn

FileSystem - Files

- In computing, a **filesystem** is used to control how data is stored and retrieved.
- The structure and logic rules used to manage the groups of information and their names is called a "filesystem"
- Filesystem are of different types and each one has different structure and logic, properties of speed, flexibility, security, size and more.
- Some FileSystems have been designed to be used for specific applications(ex iso9660 for optical disks).
- A **file** is a named collection of related information that is recorded on secondary storage such as disks, tapes, optical disks or any storage devices.
- In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user
- Files are usually created on storage device formatted with a filesystem
- **File Type**
  - File type refers to the ability of the operating system to distinguish different types of file such as text files, source files and binary files, etc.
- **FileSystem Type**
  - FileSystem is used to control how data is stored and retrieved, ex : NTFS, ext4, xfs, etc.,

Simple Data Storage Process with Storage Device



Process 1
Storage Device connected to system (RAW Disk)

Process 2
Prepare device and create **FileSystem** on these enumerated RAW Disks

Process 3
Device mounted and **Files** are created *(by OS Installation as Filesystem Hierarchy System(FHS) or to store data, etc.,)*

Required reading Chapter 13,15, RHEL8 Course book

btn

FileSystem Types

- Windows
  - **FAT 12**, **FAT16** had limit on no of directories in MS DOS and Windows
  - **FAT32** addressed the limitations of FAT12, FAT16 except the file size limit close to 4GB
  - **FAT12**, **FAT16** and **FAT32** also have a limit of eight characters for the file name, and three characters for the extension
  - **NTFS**, introduced with the Windows NT operating system, allowed ACL-based permission control.
  - **ReFS**(Resilient FileSysem), **HPFS** are other Windows FileSystems

https://support.microsoft.com/en-ca/help/100108/overview-of-fat-hpfs-and-ntfs-file-systems

btn

HUMBER
Faculty of Applied Sciences & Technology

UNIX/Linux FileSystem Types

- **UFS** - The basic filesystem for UNIX is UNIX FileSystem, or UFS, or sometimes called the Berkeley Fast FileSystem
- **ext2** - It can support partitions of up to 4 Terabytes in size and single files up to 2 Gigabytes, filename can be up to 255 characters long
- **ext3** is essentially ext2 with journaling support.
- **ext4** was born as a series of backward compatible extensions to ext3, and can support volumes with sizes up to 1 exbibyte (EiB- 2^60 bytes) and files with sizes up to 16 tebibytes (TiB – 2^40 bytes)
- **XFS** is a high-performance 64-bit journaling FileSystem created by Silicon Graphics Inc, it was the default FileSystem in SGI's IRIX operating system starting with its version 5.3. was ported to the Linux kernel in 2001; as of June 2014, XFS is supported by most Linux distributions, some of which use it as the default FileSystem (example RedHat ver 7,8)
- **ReiserFS** was introduced with version 2.4.1 of the Linux kernel and was the first journaling FileSystem available for Linux. ReiserFS is designed to be much faster than ext2, and the later versions allow the FileSystem to be expanded online, and also to shrink offline
- **ZFS** is a combined FileSystem and logical volume manager designed by Sun Microsystems.
- The features of **ZFS** include protection against data corruption, support for high storage capacities, efficient data compression, integration of the concepts of FileSystem and volume management, snapshots and copy-on-write clones, continuous integrity checking and automatic repair, RAID-Z and native NFSv4 ACLs.
- **VFAT** is an extension to the legacy FAT file system type that was used in MSDOS, VFAT support has also been added to Linux

Required reading Chapter 13, 15, RHEL8 Course book

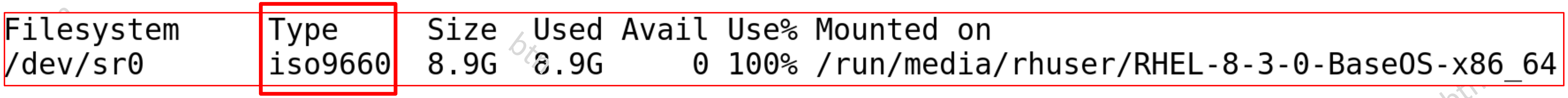
Comparison of filesystems :
https://en.wikipedia.org/wiki/Comparison_of_file_systems

WE ARE HUMBER

btn

FileSystem Types

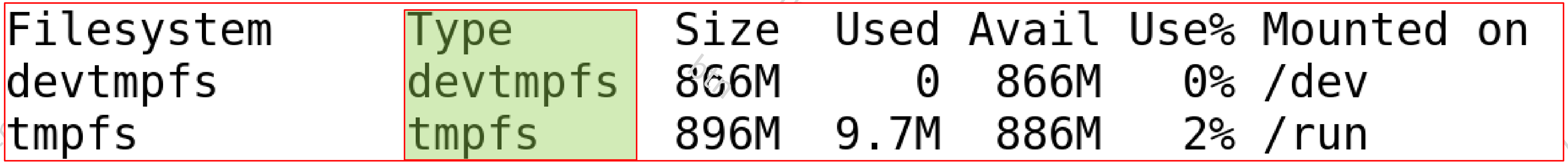
- iso9660 FileSystem is used by CD/DVD ROM enumerated in UNIX/Linux as /dev/sr0 for the first cd/dvd rom.



```
Filesystem     Type     Size  Used Avail Use% Mounted on
/dev/sr0       iso9660  8.9G  8.9G     0 100% /run/media/rhuser/RHEL-8-3-0-BaseOS-x86_64
```

- Apart from these FileSystems, UNIX/Linux have virtual FileSystem also called **pseudo filesystem** doing the same function as other FileSystems like ext3, ext4, xfs etc.,
- Examples of pseudo FileSystem
  - **tmpfs**: for storing temporary files, example /run
  - **squashfs:** used in Ubuntu 18 for loop device files.
  - **devtmpfs**: for device files directory /dev

Refer https://www.linux.org/threads/specfs-devfs-tmpfs-and-others.9382/ for more on pseudo filesystems.



```
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  866M     0  866M   0% /dev
tmpfs          tmpfs     896M  9.7M  886M   2% /run
```

btn

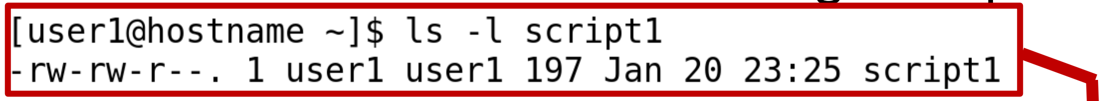
Unix/Linux Files & Directory

- On a UNIX/Linux system, everything is a file; if something is not a file, it is a process
- The Filesystem Hierarchy Standard (FHS) is the official way to organize files in Unix and Linux directories
- The directory structure starts at root directory denoted by **/**
- Standard directories are installed in default installation of Unix / Linux
- Command **ls** helps to list the files in the current directory
- When option -**l** is used with **ls** command **ls -l** display detailed information about the files in the current directory in long listing format.
- If need to list files in specific directory, **ls -l** *directory_name* is used with the path to the directory.

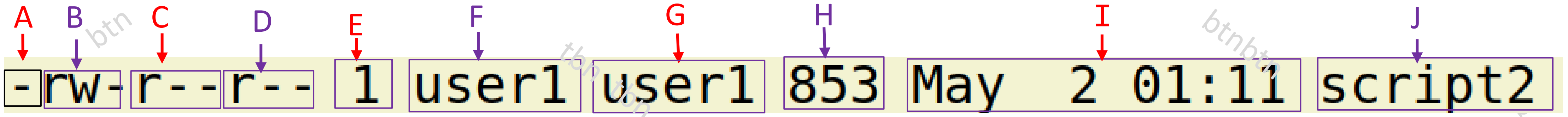
btn

File information

- Command **ls -l** lists file in long format displaying file(s) information as detailed below using example **ls -l script2** *(the detailed information is from A to J)*.



```
[user1@hostname ~]$ ls -l script1
-rw-rw-r--. 1 user1 user1 197 Jan 20 23:25 script1
```



```
-rw-r--r-- 1 user1 user1 853 May  2 01:11 script2
```

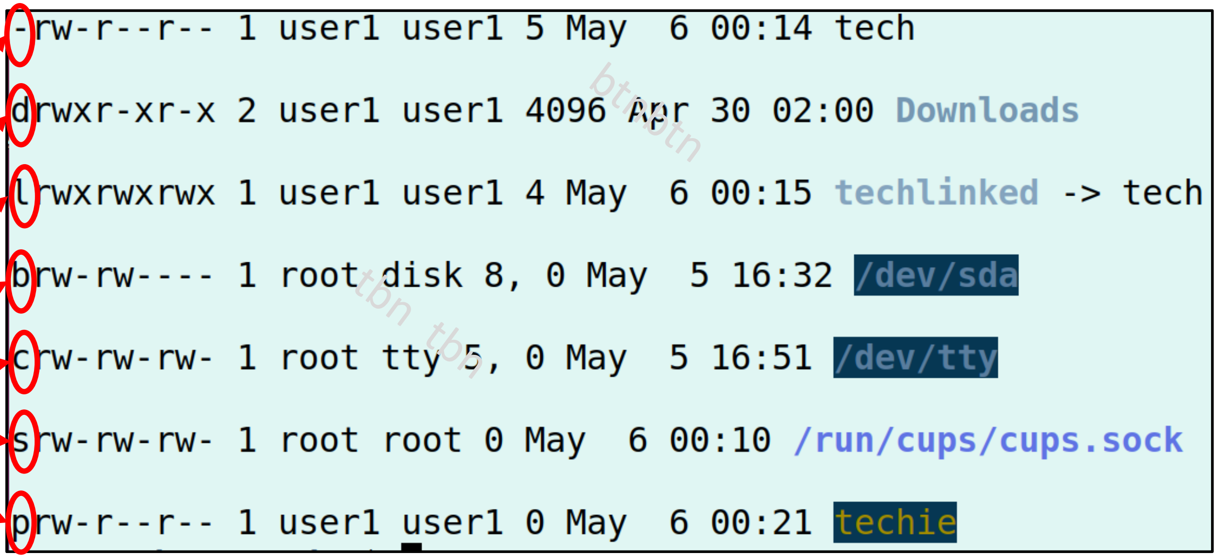
A	File type	1
B	User(Owner) permissions	3 (2-4)
C	Group permissions	3 (5-7)
D	Others permissions	3 (8-10)
E	Link count	1
F	User (Owner) name	
G	Group name	
H	File size	
I	File's Last modified time stamp	
J	File name	

- In the file and directory output, when using **ls -l**, file information is given.
- From your view, starting from left the first character (refer **A**) displays type of file
- From 2nd character to 10 character (refer number B,C,D) these are 9 permission bits of files and direct, where *(more info in next slide)*
  - the first three permission bits (Refer **B**) are User(Owner) permissions
  - the next three permission bits (Refer **C**) are group permissions
  - the last three permission bits (Refer **D**) are other user permissions
- 11th character displays the link count (Refer **E**)
- next displays who is the user (owner) of the file (Refer **F**)
- next displays which group owns the file (Refer **G**)
- next displays size of the file displayed in blocks by default (Refer **H**)
- next displays date when the file was last modified (Refer **I**)
- last one displays the filename (Refer **J**)

File Types

- Command **ls -l** gives the following information after the files in the current directory where the command is run
- The first line of display is a count of the total number of *blocks* (1 block=1,024 bytes) of storage that the listed files use.
- Each successive line displayed by the ls -l command contains detailed information about a file in the directory.
- The first character on each line indicates what type of file it is: **d** for a directory, **-** for a file and other file types as below

File type	**Symbol**
Regular File	-
Directory	d
Symbolic link	l
Device file – block	b
Device file - character	c
Socket	s
Pipe	p



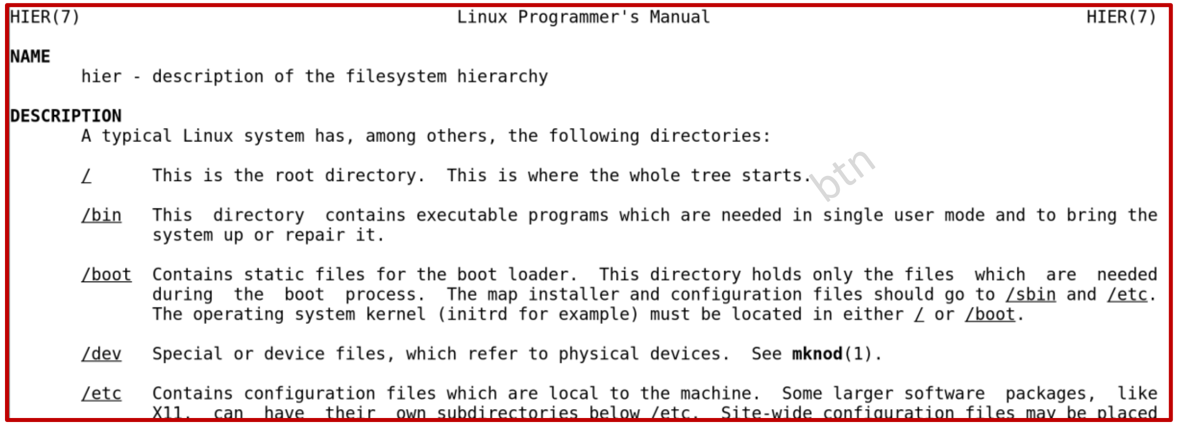
```
-rw-r--r-- 1 user1 user1 5 May  6 00:14 tech

drwxr-xr-x 2 user1 user1 4096 Apr 30 02:00 Downloads

lrwxrwxrwx 1 user1 user1 4 May  6 00:15 techlinked -> tech

brw-rw---- 1 root disk 8, 0 May  5 16:32 /dev/sda

crw-rw-rw- 1 root tty 5, 0 May  5 16:51 /dev/tty

srw-rw-rw- 1 root root 0 May  6 00:10 /run/cups/cups.sock

prw-r--r-- 1 user1 user1 0 May  6 00:21 techie
```

btn

UNIX/Linux Files & Directory

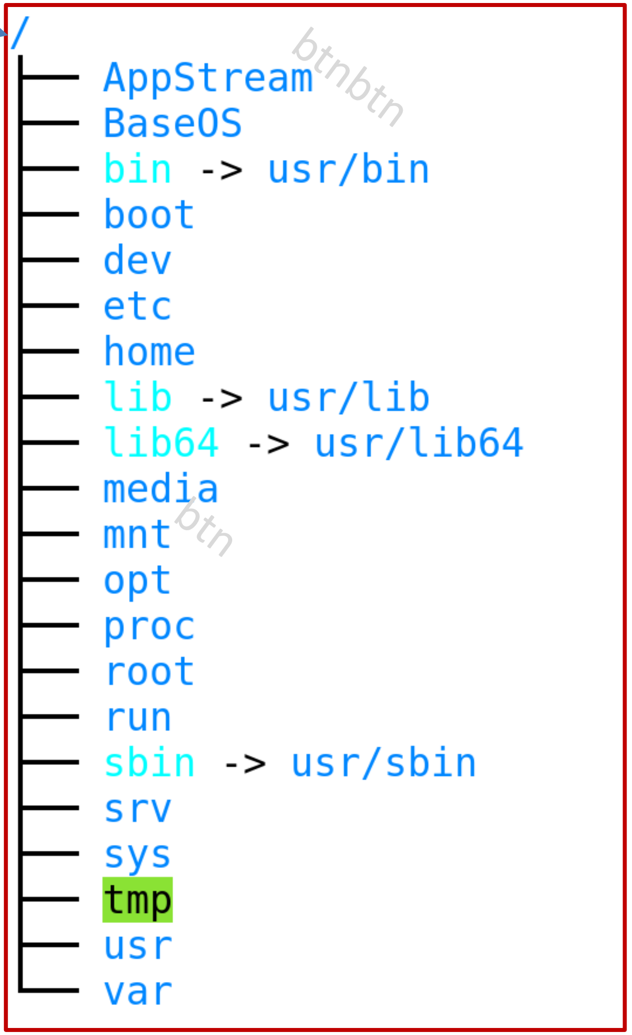
- A typical UNIX/Linux directory tree starts from root denoted by **/**
- Below **/** there are default directories and each directory and its files are for specific purpose to be used in UNIX/Linux
- For example,
  - directory **/home** contains directories of all regular users in the system with their home directory created.

    NOTE: *your home directory is **/home/your_username** and **NOT** /home*
  - directory **/bin** contains binary files used in single user mode
  - directory **/sbin** contains binary files not executed by normal users
  - directory **/etc** contains configuration files
- Command **man hier** helps to display each directory of FHS with description



```
HIER(7)                    Linux Programmer's Manual                   HIER(7)

NAME
       hier - description of the filesystem hierarchy

DESCRIPTION
       A typical Linux system has, among others, the following directories:

       /      This is the root directory.  This is where the whole tree starts.

       /bin   This  directory  contains executable programs which are needed in single user mode and to bring the
              system up or repair it.

       /boot  Contains static files for the boot loader.  This directory holds only the files  which  are  needed
              during  the  boot  process.  The map installer and configuration files should go to /sbin and /etc.
              The operating system kernel (initrd for example) must be located in either / or /boot.

       /dev   Special or device files, which refer to physical devices.  See mknod(1).

       /etc   Contains configuration files which are local to the machine.  Some larger software  packages,  like
              X11,  can  have  their  own subdirectories below /etc.  Site-wide configuration files may be placed
```

The directory structure(FHS)



```
/
├── AppStream
├── BaseOS
├── bin -> usr/bin
├── boot
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lib64 -> usr/lib64
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── srv
├── sys
├── tmp
├── usr
└── var
```

btn

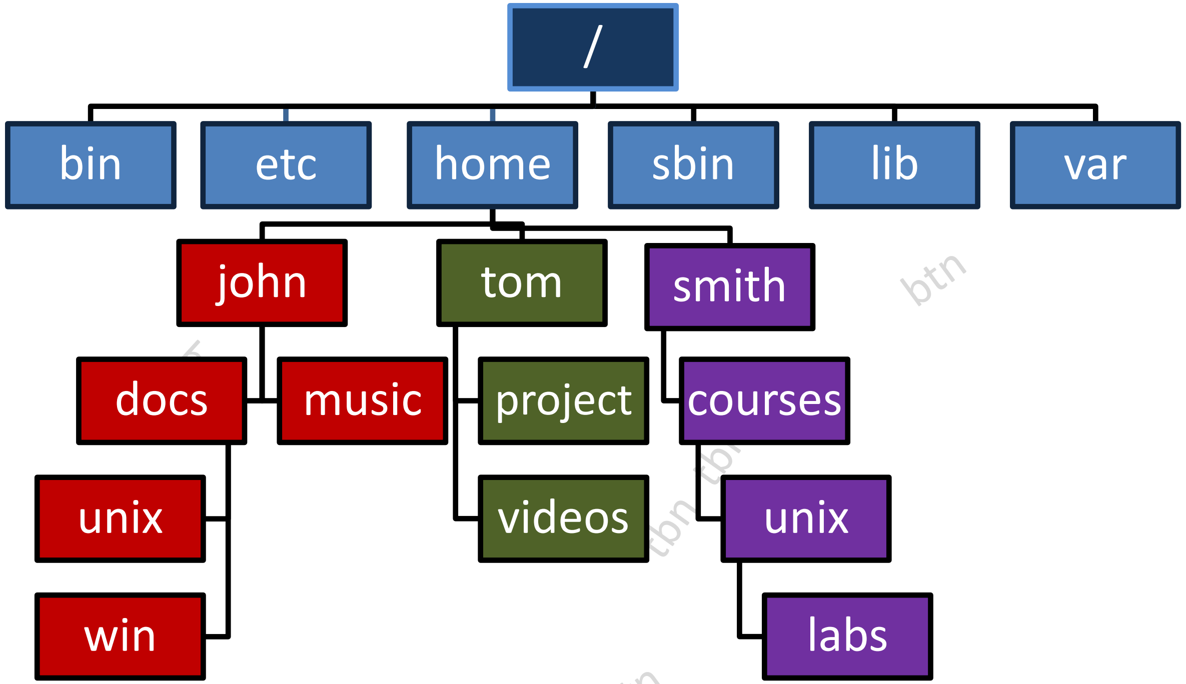
Unix/Linux Files path

- Pathnames enables you to uniquely identify a particular file to the Unix /Linux system.
- In the specification of a pathname, successive directories along the path are separated by the forward slash character **/**.
- To view the files in tree structure, **tree** command can be used. Refer to tree options for various uses the **tree** command. *(try and know the purpose of* **tree -d -L 1***)*
- A pathname that begins with a slash character is known as a **full pathname** or **Absolute path** because it specifies a complete path from the root.
- A users home directory /home/*username* is an example of Absolute path.
- For example, the pathname /home/students/n01010101 identifies the directory n01010101 contained under the directory students which is contained under home; where home is in root directory
- The path relative to your current working directory are known as relative pathnames or **Relative path**

Unix/Linux path

Directory can be Subdirectory, parent, child

- In the directory structure shown aside,
  - John is subdirectory of home.
  - home is parent directory of john and
  - john is child directory of home



- **Scenario – 1; In John's home directory,**

  Absolute path to win : /home/john/docs/win

  Relative path to win : docs/win

  *** alternatively, ~/ could be used for home directory and Absolute path to win is ~/docs/win**

- **Scenario – 2; Being in courses directory in Smith's home directory,**

  Absolute path to labs : /home/smith/courses/unix/labs

  Relative path to labs : unix/labs

btn

Creating Directory

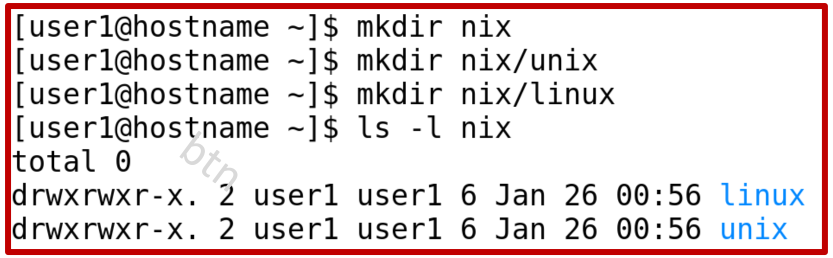
- To create directory, **mkdir** command is used as **mkdir** *dirname*

- Multiple directories also can be created. For example to create 3 directories dir1, dir2 and dir3 we can use **mkdir** *dir1 dir2 dir3*

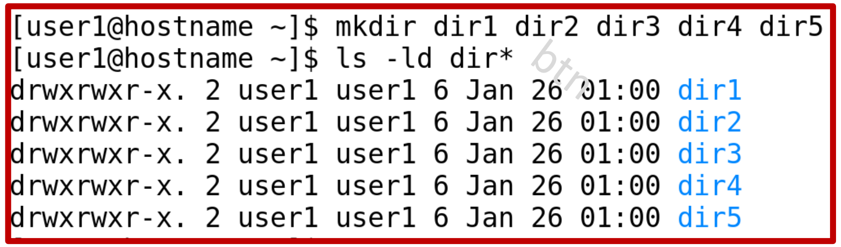
- To list the directory permissions and other information, use **ls -ld** *directory_name* with absolute or relative path

- When need to create a directory and parent directory, we can use **mkdir -p** */parent_directory/directory*

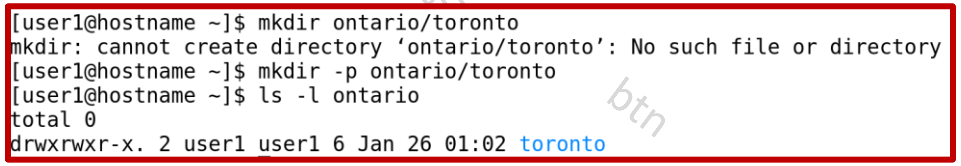
- To list the directory contents use **ls -l** *directory_name* with absolute or relative path.



```
[user1@hostname ~]$ mkdir nix
[user1@hostname ~]$ mkdir nix/unix
[user1@hostname ~]$ mkdir nix/linux
[user1@hostname ~]$ ls -l nix
total 0
drwxrwxr-x. 2 user1 user1 6 Jan 26 00:56 linux
drwxrwxr-x. 2 user1 user1 6 Jan 26 00:56 unix
```



```
[user1@hostname ~]$ mkdir dir1 dir2 dir3 dir4 dir5
[user1@hostname ~]$ ls -ld dir*
drwxrwxr-x. 2 user1 user1 6 Jan 26 01:00 dir1
drwxrwxr-x. 2 user1 user1 6 Jan 26 01:00 dir2
drwxrwxr-x. 2 user1 user1 6 Jan 26 01:00 dir3
drwxrwxr-x. 2 user1 user1 6 Jan 26 01:00 dir4
drwxrwxr-x. 2 user1 user1 6 Jan 26 01:00 dir5
```



```
[user1@hostname ~]$ mkdir ontario/toronto
mkdir: cannot create directory ‘ontario/toronto’: No such file or directory
[user1@hostname ~]$ mkdir -p ontario/toronto
[user1@hostname ~]$ ls -l ontario
total 0
drwxrwxr-x. 2 user1 user1 6 Jan 26 01:02 toronto
```

Copying Files

- Command **cp** is used to copy files in UNIX/Linux
- The syntax for copying file in UNIX/Linux **cp** *source destination*
- The source could be a file or file with path and destination be a directory
- Example: Being in your home directory, to copy file /etc/hosts to backup directory in your home directory

  cp /etc/hosts backup/
- To copy a directory to another directory cp -r *source destination*
- Example: Being in your home directory, to copy a directory /etc and all its contents to a directory backup in your home directory

  cp -r /etc backup or **cp -r** /etc/* backup *(find what is the difference in these two commands)*
- To copy to current directory(pwd) where you are running the command, you can also use period (**.**) as destination instead of destination path.
- Example: Being in your home directory, copying file /etc/ssh/ssh_config to your home directory,

  cp /etc/ssh/ssh_config .
- Refer to **man cp** for more information.

mv, rm, rmdir command

- To **move** a file or directory from one directory location to another, we use **mv** command,
  mv *source_path destination_path*

- To **rename** a file also we use **mv** command
  mv *oldfilename newfilename*

- To **delete** file, we use **rm** command
  rm *filename*

- To **delete** empty directory, we can use **rmdir**
  rmdir *directoryname*

- To **delete** directory with contents, we use
  rm –R *directory name* or **rm –r *directory name***

*When deleting many options can be used and the most serious is **rm -rf filename**, where **r** is recursive and **f** is force.*

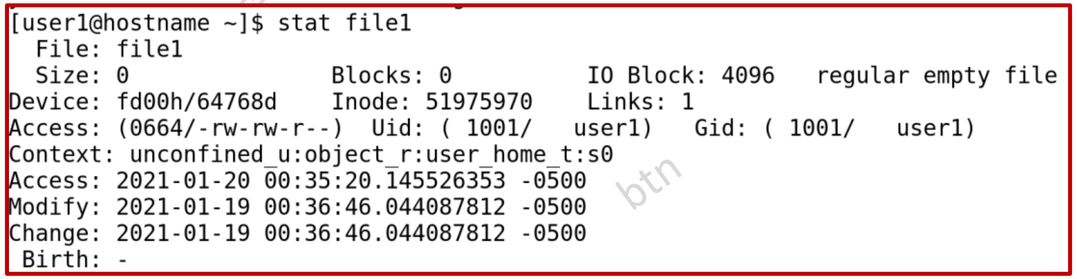
btn

btn

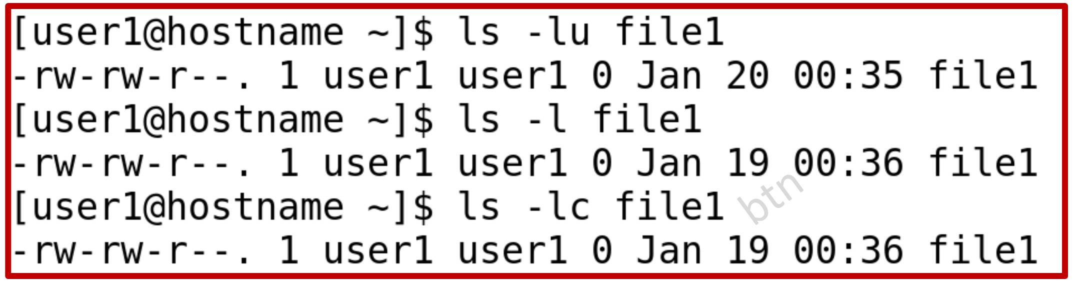
stat

- Command **stat**, display file or FileSystem status and all the three time stamps
- **atime – File Access Time**
  - atime gets updated when you **open a file** but
  - also when a file is used for grep, sort, cat, head, tail, etc.,
  - Command **ls –lu** displays the access time
- **mtime – File Modify Time**
  - The mtime gets updated when you **modify a file**.
  - Whenever you update content of a file or save a file.
  - Command **ls –l** displays the modified time

Command used is **stat *filename***

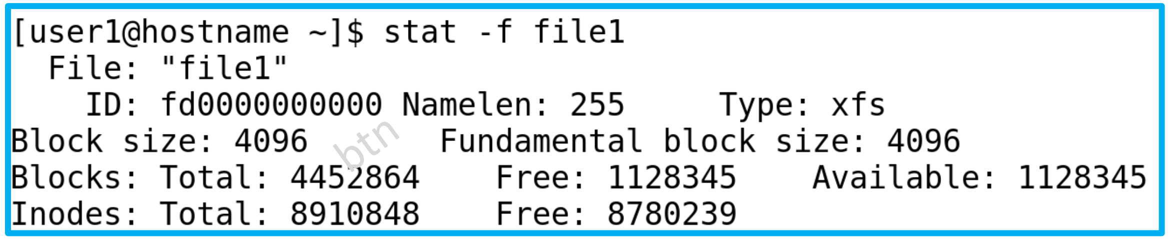


```
[user1@hostname ~]$ stat file1
  File: file1
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: fd00h/64768d    Inode: 51975970    Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1001/   user1)   Gid: ( 1001/   user1)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2021-01-20 00:35:20.145526353 -0500
Modify: 2021-01-19 00:36:46.044087812 -0500
Change: 2021-01-19 00:36:46.044087812 -0500
 Birth: -
```



```
[user1@hostname ~]$ ls -lu file1
-rw-rw-r--. 1 user1 user1 0 Jan 20 00:35 file1
[user1@hostname ~]$ ls -l file1
-rw-rw-r--. 1 user1 user1 0 Jan 19 00:36 file1
[user1@hostname ~]$ ls -lc file1
-rw-rw-r--. 1 user1 user1 0 Jan 19 00:36 file1
```

- **ctime – File Change Time**
  - The ctime gets updated when the **file attributes are changed**, like changing the owner, changing the permission or moving the file to an other filesystem but will also be updated when you modify a file.
  - Command **ls –lc** displays the change time
- For file system status **stat -f *filename***



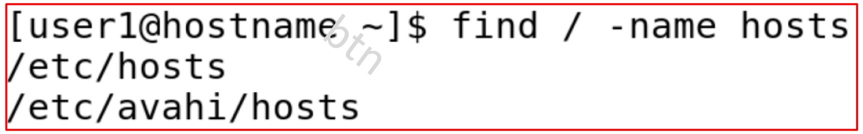
```
[user1@hostname ~]$ stat -f file1
  File: "file1"
    ID: fd0000000000 Namelen: 255     Type: xfs
Block size: 4096       Fundamental block size: 4096
Blocks: Total: 4452864    Free: 1128345    Available: 1128345
Inodes: Total: 8910848    Free: 8780239
```

Search files

- Command **find** is used to search the files in the system
- Command syntax is as below

  find *where_to_search_from search_criteria criteria_info*

- To search file from root (/) named file1: **find** / -name file1
- Other **search criteria** for the files are
  - **iname** search by pattern, file* or file? or *file* and it is case insensitive
  - **group** search by group
  - **perm** search by file permission
  - **size** search by file size
  - **type** search by file type
  - **user** search by user
  - **writable** search by files which are writable by the current user
  - **used** search by last accessed *n* days after its status was last changed
  - **amin** search by last access *n* minutes ago
  - **atime** search by last access *n**24 hours ago
  - *more search criteria can be viewed from manual pages of find command*



```
[user1@hostname ~]$ find / -name hosts
/etc/hosts
/etc/avahi/hosts
```

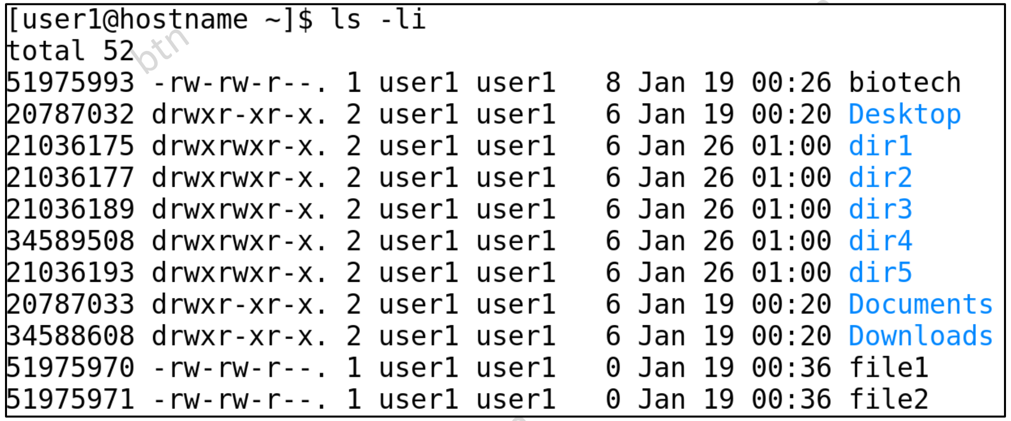
- Command **locate** find files by name
- **locate** reads one or more databases prepared by **updatedb** and writes file names matching at least one of the patterns to standard output, one per line

```
[user1@hostname ~]$ locate file1
[user1@hostname ~]$ sudo updatedb
[user1@hostname ~]$ locate file1
/home/user1/file1
/home/user1/test/file1
/home/user1/test/file100
```

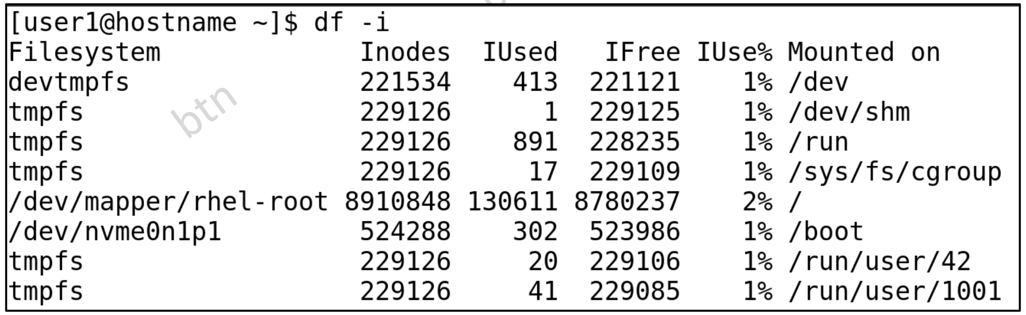
btn btn btn btn btn btn btn btn btn

Inodes

- Inode is an entry in inode table containing metadata information of the regular file, directory with a number associated with it as inode number
- It is a data structure in traditional unix system
- It contains the userid(UID), groupid(GID), permissions, filetype, timestamp, location of file in the HDD, some other metadata of the file
- Using **ls -i** command would display the inode numbers of the file or directory.
- To long list the files with inodes, **ls -li** will be used
- The inodes availability can be found for the mounted partition using **df -i** or **df -hi**



```
[user1@hostname ~]$ ls -li
total 52
51975993 -rw-rw-r--. 1 user1 user1   8 Jan 19 00:26 biotech
20787032 drwxr-xr-x. 2 user1 user1   6 Jan 19 00:20 Desktop
21036175 drwxrwxr-x. 2 user1 user1   6 Jan 26 01:00 dir1
21036177 drwxrwxr-x. 2 user1 user1   6 Jan 26 01:00 dir2
21036189 drwxrwxr-x. 2 user1 user1   6 Jan 26 01:00 dir3
34589508 drwxrwxr-x. 2 user1 user1   6 Jan 26 01:00 dir4
21036193 drwxrwxr-x. 2 user1 user1   6 Jan 26 01:00 dir5
20787033 drwxr-xr-x. 2 user1 user1   6 Jan 19 00:20 Documents
34588608 drwxr-xr-x. 2 user1 user1   6 Jan 19 00:20 Downloads
51975970 -rw-rw-r--. 1 user1 user1   0 Jan 19 00:36 file1
51975971 -rw-rw-r--. 1 user1 user1   0 Jan 19 00:36 file2
```



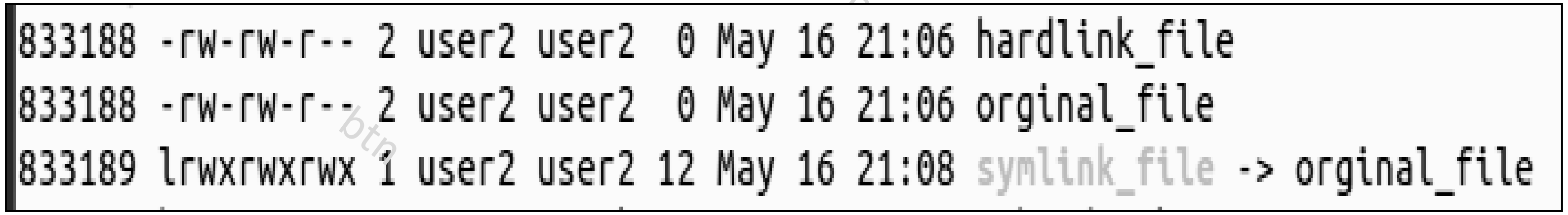
```
[user1@hostname ~]$ df -i
Filesystem              Inodes  IUsed   IFree IUse% Mounted on
devtmpfs                221534    413  221121    1% /dev
tmpfs                   229126      1  229125    1% /dev/shm
tmpfs                   229126    891  228235    1% /run
tmpfs                   229126     17  229109    1% /sys/fs/cgroup
/dev/mapper/rhel-root 8910848 130611 8780237    2% /
/dev/nvme0n1p1          524288    302  523986    1% /boot
tmpfs                   229126     20  229106    1% /run/user/42
tmpfs                   229126     41  229085    1% /run/user/1001
```

Link files

- In linux files can be linked either symbolically or hard link
- Symbolic link is like shortcut in Windows whereas hard link files are the same synchronized replica of the original file
- Command **ln** is used to create hard link files

  ln originalfile hardlinkfile
- Command **ln –s** is used to create soft link or symbolic link or symlink files

  ln –s originalfile symlinkfile
- **Hard link cannot be created for directories**
- **Inodes are equal for hard link files with original file,**
- but for **symlink files inodes are different**.



```
833188 -rw-rw-r-- 2 user2 user2  0 May 16 21:06 hardlink_file
833188 -rw-rw-r-- 2 user2 user2  0 May 16 21:06 orginal_file
833189 lrwxrwxrwx 1 user2 user2 12 May 16 21:08 symlink_file -> orginal_file
```

btn

Unix/Linux Files Management

• Create files	:**touch**, **cat** or using text editors **vi**, **vim**, **nano** **touch** *filename* or **cat** *filename* or **vi** *filename* or **nano** *filename*
• Create directory	:**mkdir** *directoryname*
• Modify/Edit	:text **editors vi**, **vim**, **nano**, … **vi** *filename* or **vim** *filename* or **nano** *filename*
• Copy	:**cp** *source destination* or to copy directory & its contents **cp -r** *source destination*
• Rename	:**mv** *source destination*
• Move	:**mv** *source destination*
• Delete	:**rm** *filename*, **rm -r** *directoryname*, **rmdir** *directoryname* *(only empty directory)*
• List files	:**ls**, **ls –l** *(wild card search with *, ? & [ ])* , **stat**, **find**, **ln**
• Display file content	:**cat** *filename* or **less** *filename* *(less controls display)* or using text editors **vi**, **vim**, **nano**
• List directory	:**ls**, **ls -ld**
• Change directories	:**cd** *path* (**cd** *and enter will return to logged in user's home directory)*
• Find current working directory	:**pwd**

btn

Storage Devices

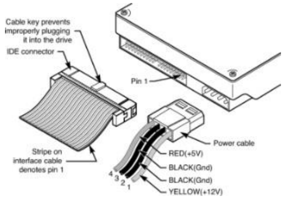
- V-NAND SSD

- SSD
  - Solid State Drives

- SSHD
  - Solid State Hybrid Disk

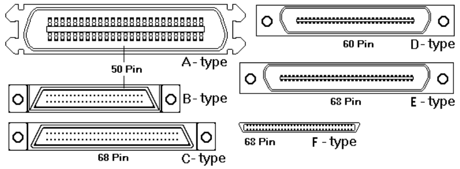
- SATA HDD
  - *Serial ATA*

- PATA HDD *(formerly IDE HDD)*
  - IDE : *Integrated Drive Electronics*
  - PATA : *Parallel ATA or Parallel Advanced Technology Attachment*


Cable key prevents improperly plugging it into the drive
IDE connector
Pin 1
Stripe on interface cable denotes pin 1
Power cable
RED(+5V)
BLACK(Gnd)
BLACK(Gnd)
YELLOW(+12V)
4 3 2 1

- SCSI HDD
  - *small computer system interface*

External SCSI - Cable Connector types (male)



http://ps-2.kev009.com/eprmhtml/eprmb/f1294.htm

- (FLASH) Memory Storage devices
- usb flash memory,
- Memory drives, micro sd **cards**, sd cards, CF cards etc., which can also be used as storage devices
- CD/DVD/Bluray drives

https://www.seagate.com/ca/en/internal-hard-drives/hdd/

https://commons.wikimedia.org/ https://www.samsung.com/in/memory-storage/ en.wikipedia.org

btn

UNIX/Linux Storage Devices

- In Unix-like operating systems, a storage device or special file is an interface for a device driver that appears in a file system as if it were an ordinary file.
- There are also special files in Microsoft Windows.
- **/dev** directory in Unix/Linux holds the device files enumerated
- All devices must be associated with a device file in order to be used by the system, this process is called enumeration.
- **dmesg** has the logs of devices being discovered and enumerated
- UNIX/Linux storage methodology can be termed as **Traditional**, **MultiDisk** and **LVM**

- HDD naming in linux:
  - for NVMe SSD the first disk is **/dev/nvme0n1** and the second disk will be **/dev/nvme0n2** and so on.
    - first partition of first NVMe SSD disk will be /dev/nvme0np1 and second partition fo the first NVMe SSD disk will be /dev/nvme0np2 and so on.
  - for SATA, SSD, SCSI disks, the first disk is **/dev/sda** and second is **/dev/sdb** and so on
    - first partition of first SATA or SSD or SCSI disk will /dev/sda1, second partition of the first SATA or SSD or SCSI disk will be /dev/sda2 and so on ...

- **UUID - Universally unique identifier**
- It is unique identifier for disk partitions, can be used to mount partitions
- UUID can be found
  - using **sudo blkid** or
  - in /dev/disk/by-uuid

btn

Pseudo Devices & Swap partition

- Device nodes on Unix-like systems do not necessarily have to correspond to physical devices.
- Nodes that lack this correspondence form the group of pseudo-devices.
- They provide various functions handled by the operating system.
- Some of the most commonly used (character-based) pseudo-devices include
  - **/dev/null** – accepts and discards all input; produces no output
  - **/dev/zero** – accepts and discards all input; produces a continuous stream of NULL (zero value) bytes
  - **/dev/full** – Linux-specific; produces a continuous stream of NULL (zero value) bytes when read, and returns a "disk full" message when written to
  - **/dev/random** and **/dev/urandom** – they produce a variable-length stream of pseudo-random numbers.

- Swap partition is used in linux system to use more memory when it is required than the available physical memory
- Kernel swaps out less used pages and gives more memory to the current applications
- Similar to pagefile.sys in windows
- Swap can be a partition in a HDD or as a mounted file
- Recommended swap space could be twice the physical memory or minimum of 1GB
- Swap ID is 82 for partitioning

https://en.wikipedia.org/wiki/Device_file

btn

Storage device partition table

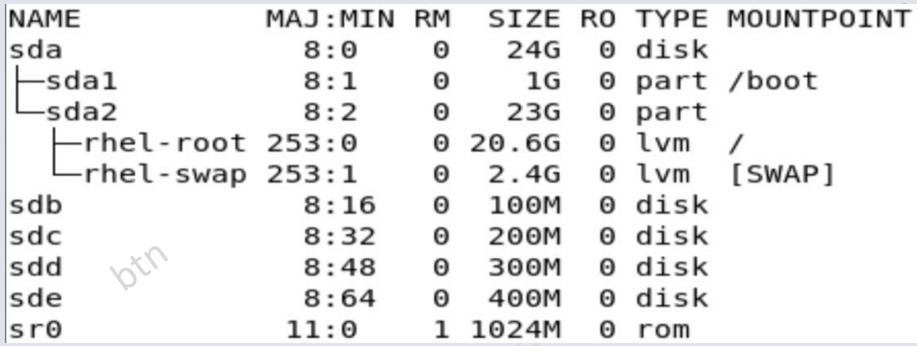
- **MBR (Master Boot Record) Partition Table,**
  - MBR is special sector at beginning of drive and contains the boot loader
  - MBR works with disks upto 2 TB
  - BIOS looks for bootloader in MBR to boot
  - Max 4 Primary Partition Tables
  - Traditionally used in UNIX, Linux and by Microsoft upto Windows 7,
- **GPT (GUID Partition Table)**
  - Globally Unique IDentifiers Partition Table
  - upto 128 primary partitions
  - it is part of UEFI (Unified Extensible Firmware Interface) standard which is faster than BIOS
  - Windows 8,10 required UEFI for secure boot and also can be installed in Linux

btn

Storage Device vs Partition

- Raw storage disk or physical hdd is called as storage device
- These sata/ssd or flash memory storage device are named as /dev/sda or /dev/sdb or /dev/sdc as the case may be based on the type and the number of disk attached.

- When device need to be used, it is good practice to partition the disk as single partition if no other partitions are required.
- If /dev/sda is partitioned into 3 partitions,
  - 1st partition is named as **/dev/sda1**,
  - 2nd partition is named as **/dev/sda2** and
  - 3rd partition is named as **/dev/sdb3**



```
NAME              MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                 8:0    0    24G  0 disk
├─sda1              8:1    0     1G  0 part /boot
└─sda2              8:2    0    23G  0 part
  ├─rhel-root     253:0    0  20.6G  0 lvm  /
  └─rhel-swap     253:1    0   2.4G  0 lvm  [SWAP]
sdb                 8:16   0   100M  0 disk
sdc                 8:32   0   200M  0 disk
sdd                 8:48   0   300M  0 disk
sde                 8:64   0   400M  0 disk
sr0                11:0    1  1024M  0 rom
```

← lsblk

- You need to know if your calling the device or calling the partition. ie., if you are going to use the device or partition.
- A small mistake of identifying either the disk or partition could result in data loss

- HDD device or Partitions has to be mounted in linux /unix to a mountpoint (directory) to join the FHS tree of the UNIX/Linux system.

- Familiar HDD tools or HDD tool used in partitioning are **fdisk**, **gdisk**, **parted**, **cfdisk** and **Disk utility**, a GUI in Ubuntu *(need to run as superuser, i.e., use sudo with these disk tools)*

Create & Mount file systems

- Command **mkfs** is used to create filesystem
  mkfs –t *filesystem_type partition* or **mkfs.***filesystem_type partition*
- Command **blkid** helps to check the filesystem type of the block device and also it gives UUID of the partition or device.

- **Mount**ing is process where the directory or **/** called as **mountpoint** in the File Hierarchy system(FHS) is attached to a filesystem formatted storage device, or storage device's partition or a special directory like /run, /dev/ etc.,
- Command **mount** is used to mount the filesystem(of the partition/device) to the **mountpoint.** (*mountpoint created using **mkdir***)
- To mount partition **/dev/sdc1** to mount point **/finance** : **sudo mount /dev/sdc1 /finance**
- To list ALL mount points commands are **mount**, **findmnt**, **df** or **df** *mountpoint*
- Command **mount** will display mountpoint, filesystem type, partition, etc.,
- To list mountpoint with filesystem type : **df -T** or **df -T** *mountpoint*
- To list mountpoint with sizes human readable & filesystem type : **df -Th**
- Command **lsblk** also will display mountpoints of the block devices.
- To find the size of mountpoint, directory, command **du** is used, try **du -sh** ***directory_name***

Note you can only mount filesystem(of the partition/device) if filesystem is created. Without creating filesystem it cannot be mounted

- To move the mount point : **mount --move** *oldmountpoint newmountpoint*
- The mounted device/partition(s) or mount points can be detached or unmounted from FHS using **umount** command
  sudo umount *device/partition(s)* or **sudo umount** *mountpoint(s)*

- **/etc/fstab** file lists available storage device and partitions indicating their mount points and file system type and **/etc/mtab** contains all the mountpoints with its devices, partitions, pseudo-devices, special directories, etc., to be mounted during boot time.

btn

Storage device Tools

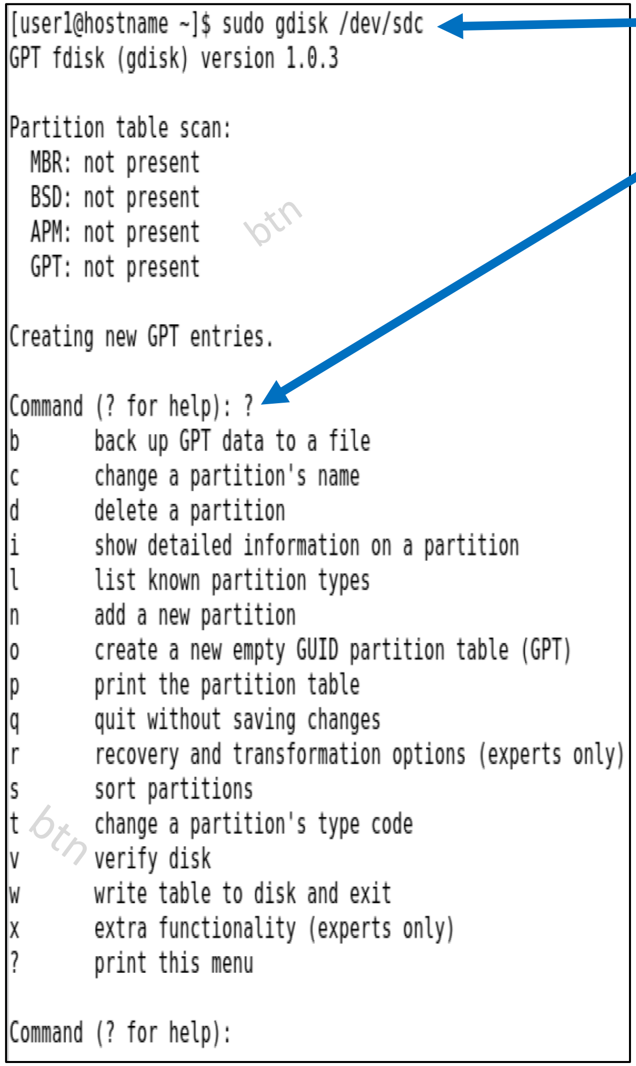
- Managing disks are administrative task and need to be used as root (**sudo**)
- Storage devices can be managed using any one of the tool
  - **fdisk** is a dialog-driven program for creation and manipulation of partition tables. It understands GPT, MBR, Sun, SGI and BSD partition tables
  - GPT fdisk also known as **gdisk** is a text-mode menu-driven program for creation and manipulation of partition tables. It will automatically convert an old-style Master Boot Record (MBR) partition table or BSD disk label stored without an MBR carrier partition to the newer Globally Unique Identifier (GUID) Partition Table (GPT) format, or will load a GUID partition table.
  - **parted** is a program to manipulate disk partitions. It supports multiple partition table formats, including MS-DOS and GPT. It is useful for creating space for new operating systems, reorganising disk usage, and copying data to new storage devices
  - Other GUI tools are **cfdisk**, **disk** utility etc.,
  - **fsck** is used to check and optionally repair one or more Linux filesystems

gdisk

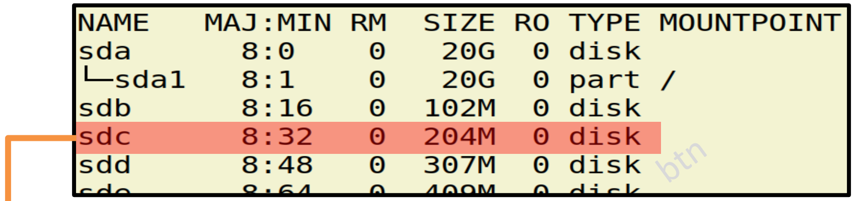
HUMBER
Faculty of Applied Sciences & Technology

- Using gdisk is quite similar to fdisk
- **sudo gdisk** ***device_name*** enters into gdisk command prompt.
- **?** at gdisk command prompt will list all gdisk commands
- Example of creating one single partition using the full disk space

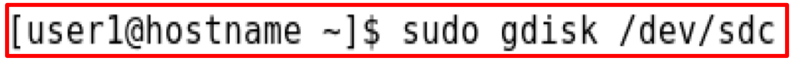
1. **sudo gdisk** ***device_name***
2. New partition is created using **n** command
3. Partition Number - Press Enter to accept default
4. First Sector: Press Enter to accept default
5. Last Sector: If creating one single partition using all the space or if it is the final(last) partition to create then accept default by pressing enter, if not specify the size of partition you need to create as +size{K, M, G, T, P} *(example +100M for 100 MB)*
6. Accept the given hex code for Linux filesystem
7. **p** is type to print the partition created
8. **w** is typed to write to partition table and exit.



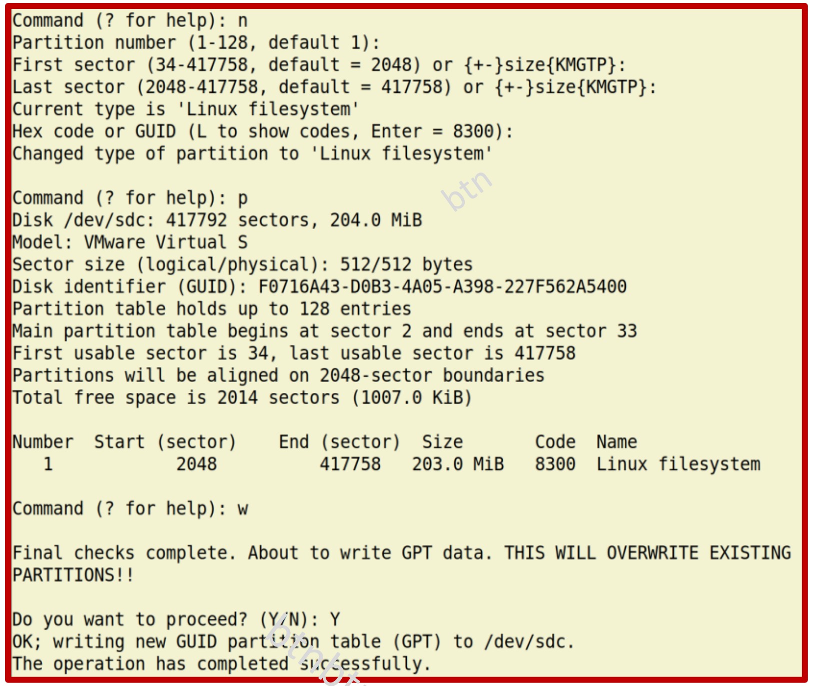
```
[user1@hostname ~]$ sudo gdisk /dev/sdc
GPT fdisk (gdisk) version 1.0.3

Partition table scan:
  MBR: not present
  BSD: not present
  APM: not present
  GPT: not present

Creating new GPT entries.

Command (? for help): ?
b       back up GPT data to a file
c       change a partition's name
d       delete a partition
i       show detailed information on a partition
l       list known partition types
n       add a new partition
o       create a new empty GUID partition table (GPT)
p       print the partition table
q       quit without saving changes
r       recovery and transformation options (experts only)
s       sort partitions
t       change a partition's type code
v       verify disk
w       write table to disk and exit
x       extra functionality (experts only)
?       print this menu

Command (? for help):
```



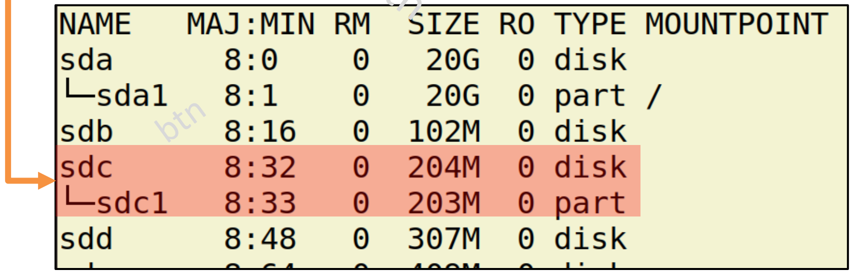
```
NAME     MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda        8:0    0    20G  0 disk
└─sda1     8:1    0    20G  0 part /
sdb        8:16   0   102M  0 disk
sdc        8:32   0   204M  0 disk
sdd        8:48   0   307M  0 disk
sde        8:64   0   409M  0 disk
```



```
[user1@hostname ~]$ sudo gdisk /dev/sdc
```



```
Command (? for help): n
Partition number (1-128, default 1):
First sector (34-417758, default = 2048) or {+-}size{KMGTP}:
Last sector (2048-417758, default = 417758) or {+-}size{KMGTP}:
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'

Command (? for help): p
Disk /dev/sdc: 417792 sectors, 204.0 MiB
Model: VMware Virtual S
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): F0716A43-D0B3-4A05-A398-227F562A5400
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 417758
Partitions will be aligned on 2048-sector boundaries
Total free space is 2014 sectors (1007.0 KiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048          417758   203.0 MiB   8300  Linux filesystem

Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!

Do you want to proceed? (Y/N): Y
OK; writing new GUID partition table (GPT) to /dev/sdc.
The operation has completed successfully.
```



```
NAME     MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda        8:0    0    20G  0 disk
└─sda1     8:1    0    20G  0 part /
sdb        8:16   0   102M  0 disk
sdc        8:32   0   204M  0 disk
└─sdc1     8:33   0   203M  0 part
sdd        8:48   0   307M  0 disk
```

*To delete use **d** and **w** command within gdisk - deletion destroys data.*

WE ARE HUMBER

btn

parted

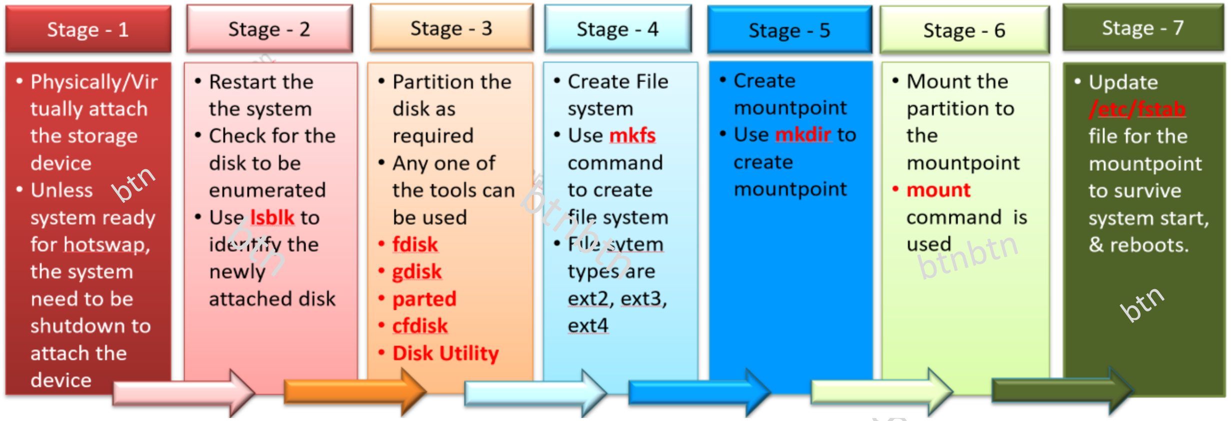
- Command parted*(partition editor)* can also be used to partition disks
- It supports both GPT and MBR
- Need to create label first in parted to start creating partition

  parted *devicename* **mklabel gpt**
- To create partition: **parted** *devicename* **mkpart primary 1** *size*
- To list partitions: **parted** *devicename* **print**
- To delete partition: **parted** *devicename* **rm** *partitionnumber*
- Refer to Chapter 13 of course book

https://access.redhat.com/sites/default/files/attachments/parted_0.pdf

Steps mount partitions/disks

HUMBER
Faculty of Applied Sciences & Technology



Stage - 1
- Physically/Virtually attach the storage device
- Unless system ready for hotswap, the system need to be shutdown to attach the device

Stage - 2
- Restart the the system
- Check for the disk to be enumerated
- Use **lsblk** to identify the newly attached disk

Stage - 3
- Partition the disk as required
- Any one of the tools can be used
- **fdisk**
- **gdisk**
- **parted**
- **cfdisk**
- **Disk Utility**

Stage - 4
- Create File system
- Use **mkfs** command to create file system
- File sytem types are ext2, ext3, ext4

Stage - 5
- Create mountpoint
- Use **mkdir** to create mountpoint

Stage - 6
- Mount the partition to the mountpoint
- **mount** command is used

Stage - 7
- Update **/etc/fstab** file for the mountpoint to survive system start, & reboots.

1. Shut down system , Add the HDD device (physically or virtually) to the system and Restart
2. Check if system is enumerated using **lsblk** command
3. Create partition as required using any hdd utilities like fdisk, gdisk, parted, etc.,
4. Make filesystem using **mkfs.***filesystem_type partition_name* or **mkfs –t** *filesystem_type partition_name*
5. Create mount point to which you want to mount the partition to add to Linux file hierarchy system, **sudo mkdir** *mountpoint_name*
6. Mount the partition using **sudo mount** *partition_name mountpoint_name*

   To check if the partition is mounted, **df -Th** *mountpoint_name* or **findmnt** *mountpoint_name*
7. Add entries in **/etc/fstab** file to make it persistent and

   Check the /etc/fstab entries with **sudo mount --all**

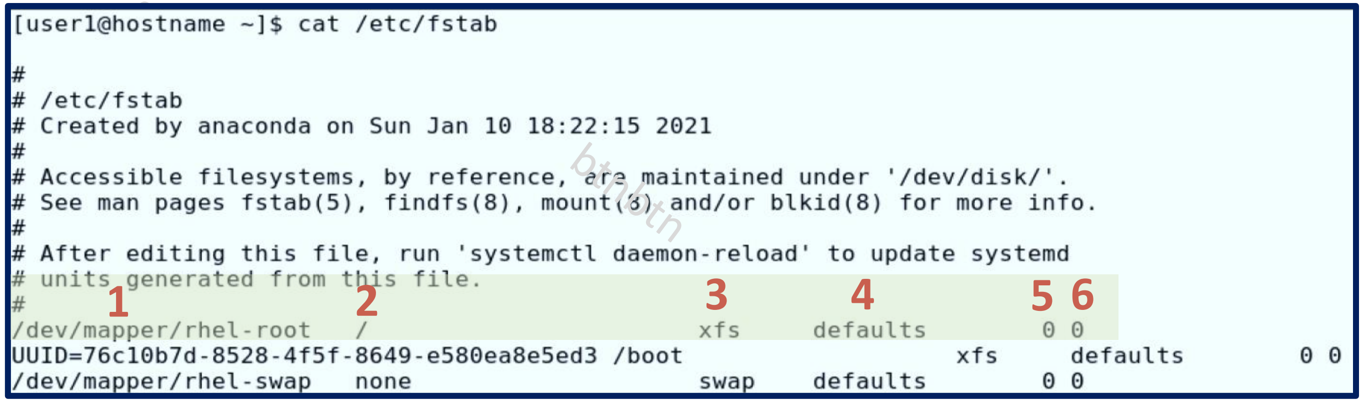
Additional Reading

WE ARE HUMBER

btn

/etc/fstab

- /etc/fstab is a system configuration file and mount point information has to be updated in /etc/fstab file to survive system restart/reboot/poweroff and start
- When system starts up, it reads the /etc/fstab to mount the mount points.
- The /etc/fstab requires 6 information related to the mount point separated by TAB (Recommended to read man fstab)



```
[user1@hostname ~]$ cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Sun Jan 10 18:22:15 2021
#
# 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.
#
/dev/mapper/rhel-root   /                       xfs     defaults        0 0
UUID=76c10b7d-8528-4f5f-8649-e580ea8e5ed3 /boot                   xfs     defaults        0 0
/dev/mapper/rhel-swap   none                    swap    defaults        0 0
```

- TAB/SPACE is the delimiter between the fields
- If the entries are not entered correctly, system could enter maintenance mode while starting up.
- The errored entries needs correction or commenting to startup.

1. **file system** : The partition/device name has to be entered for example /dev/sdb1 *(instead of partition UUID can be used)*
2. **mount point** : The mount point to which the partition/device name has to be mounted, for example /mount1. *This mount point has to be created using sudo mkdir /mount1*
3. **type** : The file system type has to be entered, for example ext4
4. **options** : The options for the mount command to read and apply to the mount point, for example ro for read only or rw or read write . Using defaults option enables **rw, suid, dev, exec, auto, nouser, and async**
5. **dump** : To determine which file systems need to be dumped, for 0 (zero) it is never
6. **pass** : This field is used by fsck to determine the order in which filesystem checks are done at boot time. The root filesystem should be specified with 1. Other filesystems should have 2. Filesystems within a drive will be checked sequentially, but filesystems on different drives will be checked at the same time to utilize parallelism available in the hardware. Defaults to zero (don't fsck) if not present.

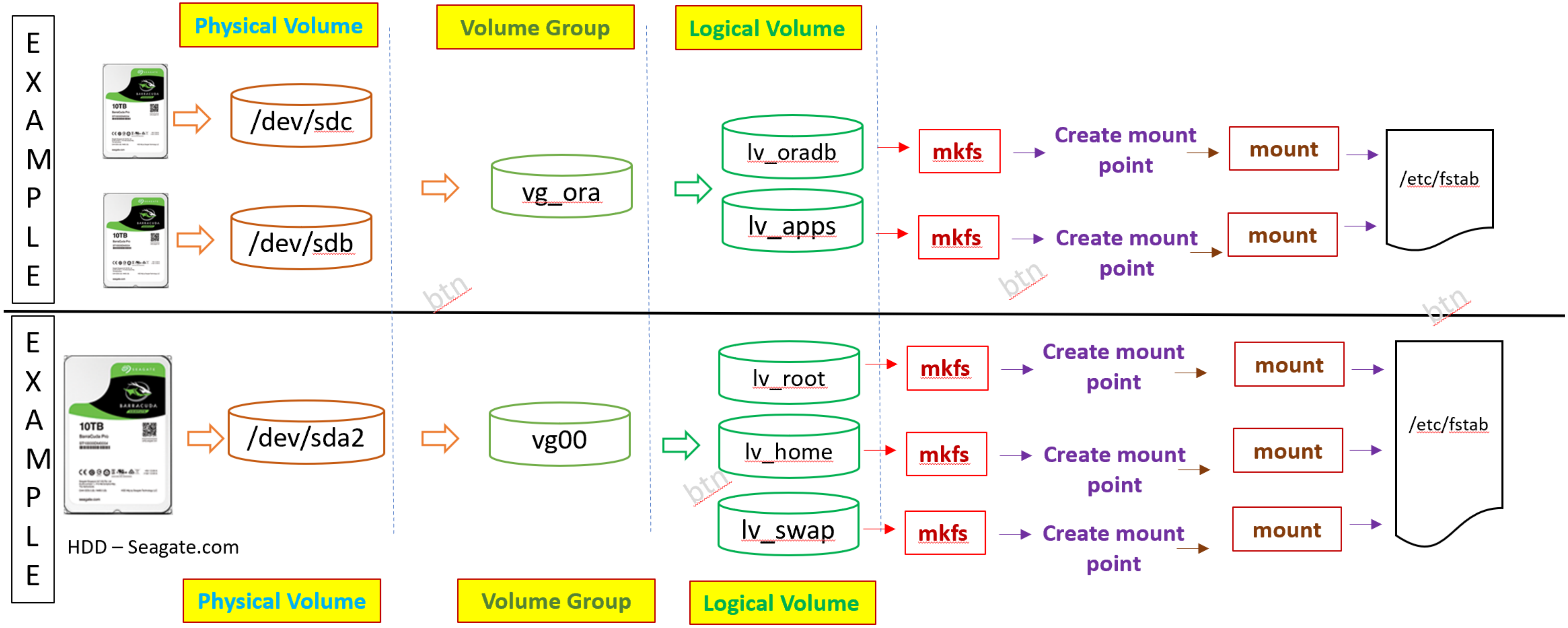
LVM

- LVM provides
  - an abstraction layer between the physical storage and the file system,
  - enabling the file system to be resized,
  - to span across multiple physical disks,
  - use random disk space, etc.
- There are 3 concepts that LVM manages:
  - **Physical Volumes** correspond to disks; they are block devices that provide the space to store logical volumes
  - **Volume Group** is a named collection of physical and logical volumes.
  - **Logical volumes** correspond to partitions: they hold a filesystem. Unlike partitions though, logical volumes get names rather than numbers, they can span across multiple disks, and do not have to be physically contiguous
- LVM allows to accumulate spaces taken from
  - one or several partitions or disks (called **physical volumes** (**pv**))
  - to form a logical a logical container (called **volume group** (**vg**)),
  - which is then divided into logical partitions (called **logical volumes** (**lv**)).

tbn

LVM – Logical Volume Manager

- LVM Examples



EXAMPLE

Physical Volume | Volume Group | Logical Volume

/dev/sdc
/dev/sdb
vg_ora
lv_oradb → mkfs → Create mount point → mount → /etc/fstab
lv_apps → mkfs → Create mount point → mount → /etc/fstab

EXAMPLE

/dev/sda2
vg00
lv_root → mkfs → Create mount point → mount → /etc/fstab
lv_home → mkfs → Create mount point → mount → /etc/fstab
lv_swap → mkfs → Create mount point → mount → /etc/fstab

HDD – Seagate.com

Physical Volume | Volume Group | Logical Volume

tbn
btn
btn
btn
btn

tbn

LVM

Physical Volume

- To get PV info
  - **pvscan**
  - **pvs**
  - **pvdisplay**
- To create PV

  pvcreate *devicename*
- To remove PV

  pvremove *devicename*

Volume Group

- To get VG info
  - **vgscan**
  - **vgs**
  - **vgdisplay**
- To create VG

  vgcreate *vgname pvname*
- To extend VG

  vgextend *vgname pvname*
- To remove PV from VG

  vgreduce *vgname pvname*
- To remove VG

  vgremove *vgname*

Logical Volume

- To get LV info
  - **lvscan**
  - **lvs**
  - **lvdisplay**
- To create LV

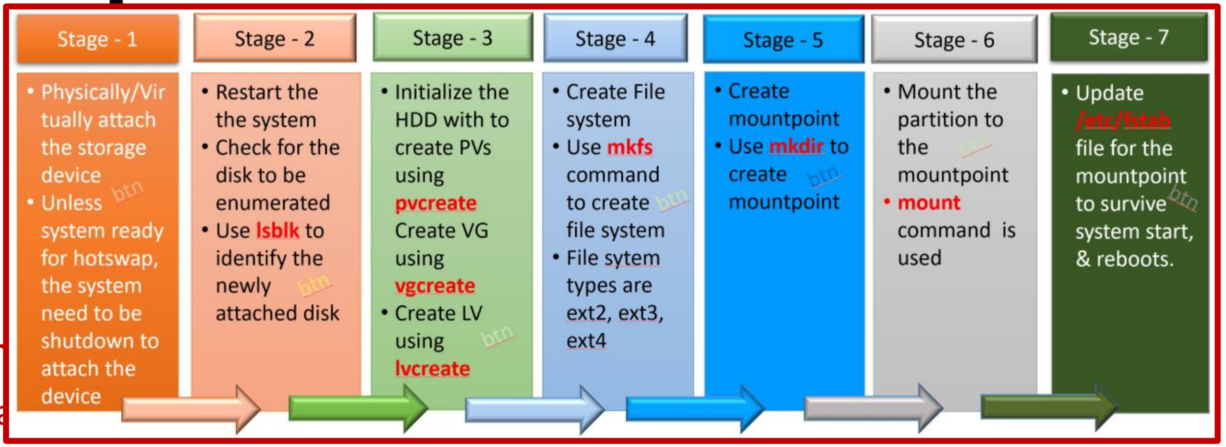
  lvcreate -n *lvname* **-L** *sizewithunits vgname*
- To extend VG

  sudo lvextend –L *+size LVPath* OR
  sudo lvresize -L *+size LVPath*
- LV Path

  /dev/*VGname***/***LVname*
- To remove LV

  lvremove *LVPath*

LVM - steps



Stage - 1
- Physically/Virtually attach the storage device
- Unless system ready for hotswap, the system need to be shutdown to attach the device

Stage - 2
- Restart the the system
- Check for the disk to be enumerated
- Use lsblk to identify the newly attached disk

Stage - 3
- Initialize the HDD with to create PVs using pvcreate Create VG using vgcreate
- Create LV using lvcreate

Stage - 4
- Create File system
- Use mkfs command to create file system
- File sytem types are ext2, ext3, ext4

Stage - 5
- Create mountpoint
- Use mkdir to create mountpoint

Stage - 6
- Mount the partition to the mountpoint
- mount command is used

Stage - 7
- Update /etc/fstab file for the mountpoint to survive system start, & reboots.

Stage-1 - *(this may not be step not required in VCloud labs)*

Stage-2 - Identify required disks using **lsblk** command

Stage-3

1. Create Physical Volume : **sudo pvcreate** physical disk nar
2. Create Volume Group : **sudo vgcreate** VGname physical disk na
3. Check free size : **sudo vgdisplay** VGname_created_as_above or **vgs** command
4. Create Logical Volume : **sudo lvcreate –n** LVname **-L** +Free_size_as_above vgname

Stage-4

Make file system : **sudo mkfs –t** filesystem_type LVPath

Stage-5

Create mount point as required using **mkdir** directory name

Stage-6

1. Mount the partition: **sudo mount** LVPath mount_point
2. Check the partitions using **df -Th** & **lsblk**

Stage-7

Update the required information in /etc/fstab to have mount point mounted on every reboot.

Extending VG, Resizing LV & mount point

- To increase the storage size by adding another storage device, need to **extend VG**

  vgextend *vgname pvname*

- To **extend LV**, check on free available space using **vgdisplay** or **vgs** and then extend LV using **lvextend**

  sudo vgdisplay VGname or **sudo vgs VGName** *to find free space*

  sudo lvextend –L *+size LVPath* OR

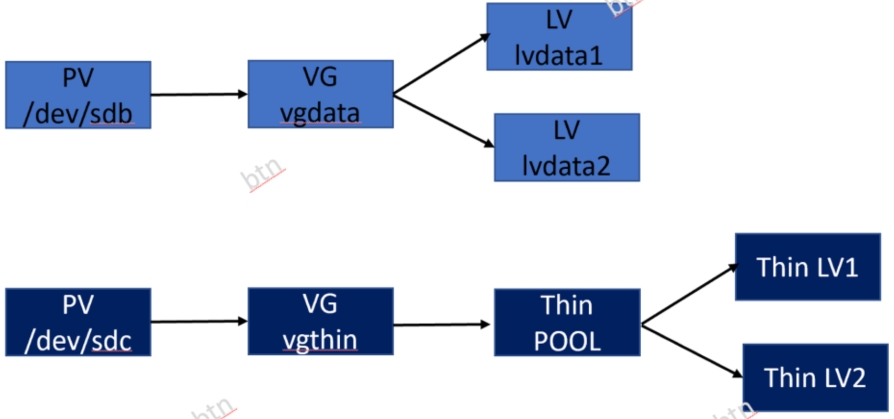
  sudo lvresize -L *+size LVPath*

- Command **lvextend** can increase size, whereas **lvresize** can reduce or increase the size.
- TO RESIZE MOUNT POINT of the LV
- When extending Logical Volume, the mount point of LV is not extended, Commands **xfs_growfs**, **fsadm** can be used to resize the mount point
- To resize : **xfs_growfs** *LVPath* or **fsadm -l resize** *LVPath*

tbn

LVM Thin provisioning

- Logical volumes can be thinly provisioned.
- This allows you to create logical volumes that are larger than the available extents.
- Using thin provisioning, you can manage a storage pool of free space, known as a thin pool, which can be allocated to an arbitrary number of devices when needed by applications.



- Recommended Reading: **man lvmthin**

- You can then create devices that can be bound to the thin pool for later allocation when an application actually writes to the logical volume.
- The thin pool can be expanded dynamically when needed for cost-effective allocation of storage space.
- Logical volumes do not assign the space allotted immediately rather the space used as size increases.

Required Reading
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_logical_volumes/assembly_thinly-provisioned-logical-volumes_configuring-and-managing-logical-volumes

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/logical_volume_manager_administration/lv_overview

Thin Provisioning, Deduplication & Compression

- The Red Hat Virtualization Manager provides provisioning policies to optimize storage usage within the virtualization environment. A **thin provisioning** policy allows you to over-commit storage resources, provisioning storage based on the actual storage usage of your virtualization environment.
- **Deduplication** is a technique for reducing the consumption of storage resources by eliminating multiple copies of duplicate blocks.
- **Compression** is a data-reduction technique that works well with file formats that do not necessarily exhibit block-level redundancy, such as log files and databases.

https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.3/html/technical_reference/over-commitment
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/vdo-integration

tbn

Storage Optimization (rhel9)

- Virtual Data Optimizer (VDO) is feature to optimize storage using thin provisioning, de-duplication and compression technologies

1. Create volume group: **vgcreate** *vgname physicaldiskname*
2. Create Logical Volume: **lvcreate --type vdo -n** *lvname* **-l** *PE* **-V** 50G *vgname*
3. Create file system **mkfs.xfs /dev/***vgname/lvname*
4. Create mount point
5. To mount: **mount /dev/***vgname/lvname /mountpoint_name*
6. Check with *df -Th mountpointname*

tbn

Stratis Volume-Managing File System

Required reading Chapter 14 of course book and exercises 14-7, 14-8, 14-9, 14-10

- Packages required : stratisd and stratis-cli
- Enable and start service stratisd : **systemctl start stratisd**
- Following steps to create pool and mount.

1. Create stratis pool: **stratis pool create** *pool_name hdd_device_name*
2. List pool: **stratis pool list**
3. To display block device used in the pool: **stratis blockdev list** *pool_name*
4. Create filesystem: **stratis filesystem create** *pool_name filesystem_name*
5. Create mount point
6. Mount : for rhel 8.3 **mount** /stratis/*pool_name/filesystem_name mount_point* for rhel 8.4 **mount** /dev/stratis/*pool_name/filesystem_name mount_point*
7. When updating /etc/fstab, in options it should be **x-systemd.requires=stratisd.service** other entries are similar to traditional or multi-disks.

RedHat Stratis Link

tbn

Stratis Volume-Managing File System

Required reading Chapter 14 of course book and exercises 14-7, 14-8, 14-9, 14-10

- The stratis pool can be expanded and renamed
- To expand the pool with an additional hdd,

  stratis pool add-data *pool_name hdd_device_name*
- To rename pool: **stratis pool rename** *old_pool_name new_pool_name*
- To rename filesystem:

  stratis filesystem rename *pool_name old_filesystem_name new_filesystem_name*
- To destroy *(the data will be lost),* unmount the filesystem and then
  1. Remove the filesystem from pool: **stratis filesystem destroy** *pool_name filesystem_name*
  2. Remove the pool: **stratis pool destroy** *pool_name*
- List and check if it is removed.