**CIS-Training**

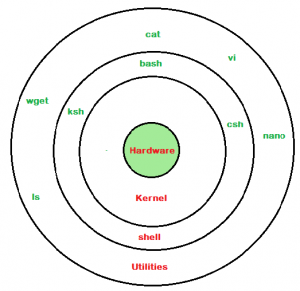
**Day 1:08/09/2025 Topic – Linux**

**Linux vs Unix vs Windows**

* Linux -open source
* Unix and Windows is properatory
* If you are using unix or windows you have no freedom bcoz it is owned by the other companies along with the hardware.
* Before unix there is multics.
* Portable means you can use it in any device with proper kernel.
* FSF-Free software foundation.
* Kernel is the core of the OS
* Distributions of LINUX:
  + RedHat (Major Enterprises)
  + SUSE
  + CENT
  + DEBIAN
  + CANONICA
  + UBUNTU
  + ORACLE
* Linux is robust and stable ,Multitasking
* Linux is less prone to virus attack and Secure OS
* LINUX is mainly command Line CLI
* LINUX is a multi user operating system.
* HCL is available in Redhat website.
* Linux version (Red HAT 3-9)

**RHEL-9**

* USERS -Redhat ,Pseudo
* Shell -Converts high level language to machine language
* Shell is a command interpreter.
* Line of interaction:
  + User->Application or program ->Shell ->OS->Kernel ->Hardware



* All the user root directories are /home/username
* Simple modular programs are chained to complete the complex one and it is easy to Trouble shoot in linux.

**NOTE-Shamba is a file transferring service between linux and windows.**

**KERNEL:**

* Kernerl is the core of the Operating system.
* It has complete control over everything.
* It perform all core functions like resource management ,memory ,I/O device , hardware etc

**Process management:**

* Handling Interrupts ,booting etc

**Linux principle:**

* Everything is a file(including hardware)
* Small, single purpose programs
* Ability to chain programs together to perform complex tasks
* Avoid captive user interfaces.
* Configure data stored in text.
* System and server Configurative files are stored in “/etc” directory

**File hierarchy Concepts:**

* Linux is an inverted Tree directory.
* Paths are defines as ‘/’;
* Case sensitive names.

**Some Important points:**

* Home directories are :/root/home/username.
* User Executable:/bin,/usr/bin,/usr/local.bin
* System executables: /sbin,/usr/sbin,/usr/local/bin
* Other mountpoints: /media,/mnt
* Configuration: /etc
* Temporary Files: /tmp
* Kernels and Bootloaders: /boot
* Server Data:/var,/srv
* System information:/proc,/sys
* Shared Libraries:/lib,/usr/lib,/usr/local/lib

**Architecture:**

* Monolithic kernel
  + Contains modular components
* UNIX like or UNIX based OS
* SIX primary sub systems:
  + Process management
  + Inter process Communication
  + Memory management
* File System management
  + VFS; provides a single interface to all the file system
* I/O management
* DEVICE management

**Installation methods:**

* Locally-From DVD,Pen drive or HDD
* Over the network-NFS,FTP,HTTP
* Kickstart Installation(Automated or unattended installation using kickstart configuration file) and it is used to install on multiple systems simultaneously.
* PXE Installation(Pre execution environment done through network booting)
* Can be installed in GUI or text.

PXE

SERVER

PXE

CLIENT

* BOOT ROM ON NW Card is like a motherboard

**Commands:**

* Cat/etc/os-release :
  + Cat-concatenate
  + Can give you details about the OS.
* Cat/etc/redhat-release :
  + Shows the redhat details
* uname
  + Shows the name of the OS
* uname -r
  + Tells us the kernel release details
* uname -p
  + Gives the details of processor
  + #-r=kernel release,-p=processor ,x86\_64=intel 64 bit processor
* uname -m:
  + Shows details of machine
* uname -i:
  + Shows the details of the architecture
  + #-m=machine, -i=architecture
* uname-a:
  + Shows all the details of it
* Free:
  + How much RAM is present.

**Swap -> Virtual memory (Uses ROM to connect RAM memory which is not physical)**

**Part of HDD is used as RAM= Inactive Apllications are moved from RAM to Swap**

* Swapping is the process of moving from RAM to SWAP
* Paging is the process of moving form SWAP to RAM
* cd /
  + Change directory
* ls
  + List of executable and all files in the current dir

Mountpoint – the point where the are mounted. A **mount point** is simply a **directory** (usually empty) where a file system is **mounted**. Once mounted, the contents of the device or partition appear under that directory.

* Proc=Process =Pseudo or virtual filesystem used by kernel for its process
* We refer process by process name refers by process id(PID)
* PPID=Parent process ID
* Human reference for file=filename ,OS reference=inode or index number.
* We refer users by username or uid,we refer gropuname by gid
* Ls -I:
  + Used to display inode number of each file or directory
* ps:
  + Process stat
* mandb
  + To update the whatis database in the system.
* Whatis
  + Gives single line info regarding the command
  + Eg:- whatis whatis:
* Man=manualpages
  + Man=manualpages=Help in linux
  + Basic command Syntax: Command[Options]<arguments>
* Ls-a
  + Show all files including hidden files
  + ‘.’ Indicates current directory
  + ‘..’ indicates parent directory
  + If a file name started with ‘.’ Then it is hidden in linux and unix
* Ls-l: long listing
  + d=directory
* mkdir=make directory
* cd=change directory
  + Absolute path =full path from / to that file or dir,absolute path with /
  + cd/usr/share/doc
  + Relatice path=relative from your position in the dir structure ,does not begin with /
  + Cd zip
* Pwd -print working directory
* Cd= to change dir
* Cd..= changes to parent directory
* Cd -= changes to immediate previous dir
* Cd =changes to logged in users home dir
* Cat>file1.txt
  + It will create a file
  + Write anything you want (Multiple line)
  + Then ctrl+D to save
* Echo “hi”>file1
* Cat >>file2
  + It adds content to the existing file(Appending)
* -l=long listing , -i=inode number or index number ,-R=recursive
* [Pipe=Output of first command is given as input to next command for further process]
* Ls-lR | more, Output of ls-lR is given to the pager more
* #more =Display one page at a time
* **Putty -is a tool to connect windows to linux for remote administration**
* To copy a file
  + Cp “Source\_file” “Target\_file”
  + **With options:**
* -v (verbose): shows what’s being copied
* -i (interactive): asks before overwriting
* -r (recursive): for copying directories
* To remove
  + File-> rm filename
  + Director->rmdir
* To rename
  + mv old\_filename to new filename
  + if you were renaming a directory the file will be moved there
* Recursive deletion
  + Rm-ri “Target”

**Hard Link**

**What it is:**

A **hard link** is another name for the same file. It points directly to the file's data on the disk.

**Key Points:**

* Both the original and hard link share the same **inode** (internal ID).
* If you delete the original file, the hard link still works.
* You **cannot** create hard links for directories (usually) and across partition.
* Works only **within the same filesystem**.

**Example:**

Now both original.txt and hardlink.txt are the same file. Changes to one affect the other.

**Soft Link (Symbolic Link)**

**What it is:**

A **soft link** is like a shortcut. It points to the file name, not the actual data.

**Key Points:**

* It has a **different inode**.
* If the original file is deleted, the soft link breaks (becomes a “dangling link”).
* You **can** create soft links to directories.
* Works **across filesystems**.

**Example:**

Now softlink.txt is just a pointer to original.txt.

**Feature Hard Link Soft Link (Symbolic)**

**Points toFile** data (inode) File name (path)

**Inode**  Same as original Different

**Survives deletion?** Yes No (breaks)

**Can link directories?** No (usually) Yes

**Cross filesystem?** No Yes

**Dangling link** -> Softlink which points to a file that does not exist



-s : for creating soft link

A computer screen shot of a black screen

AI-generated content may be incorrect.

Dangling link

**DISK PARTITION**

* Df=Disk free Space for partition (displays all the active or the mounted partitions)
* Df -Th
  + T=filesystem
  + h=Displays all the sizes in human readable form (GB,MB,KB)
    - Divides the size by 1024 and gives accurate size in GB,MB,KB
    - H=divides the size by 1000 and is not accurate.
* Du -> Disk usage by file or directory

du -h anaconda.config

* du-sh:Summary

**Basic file and dir permissions:**

* user-u
* Group-g
* Other-o
* UserId=GroupId of root=0

=(Assign) r or w or x

+(Add) r or w or x

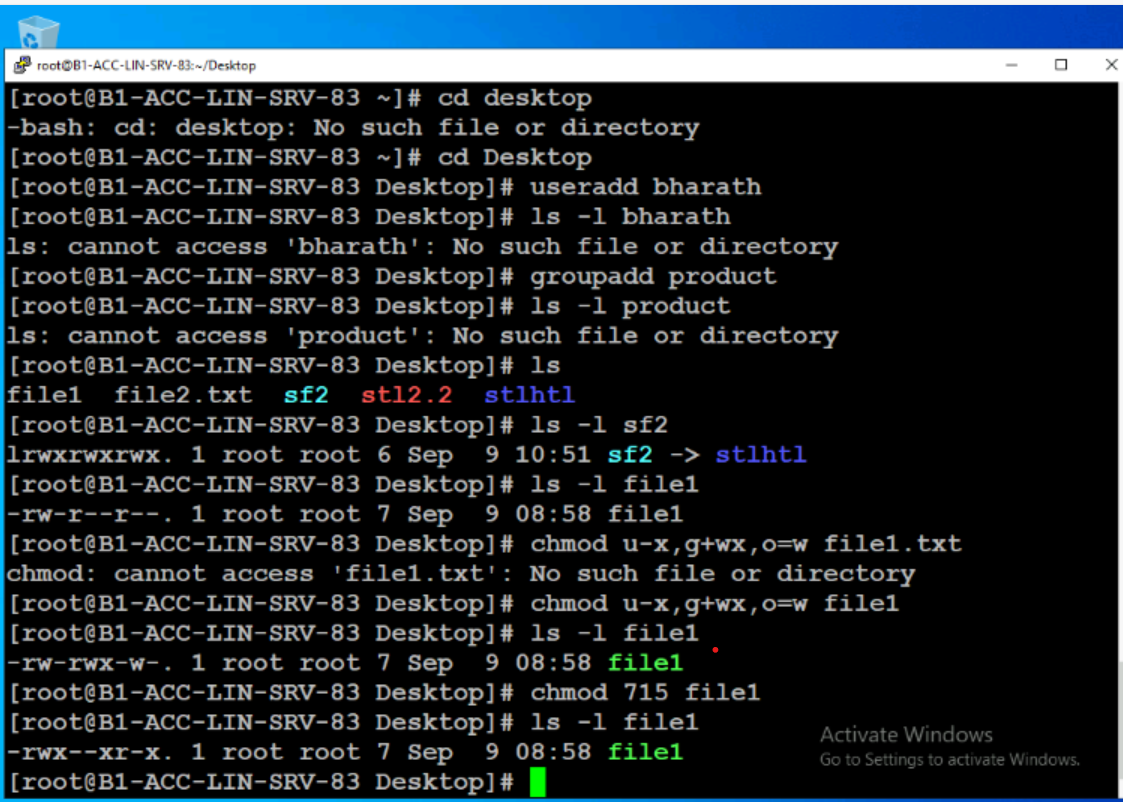
-(remove) r or w or x

* Initial U=rw
* Step 1=U+x=>rwx
* Step 2=U-x=>rw
* Step 3= U=r=>r
* Chown- change owner name(chown <username> <fileordirname>=change owner of file/dir
* Chgrp=change group name(chgrp <grpname> <fileordirname> =change group of file/directory
* Chmod = change the permission.

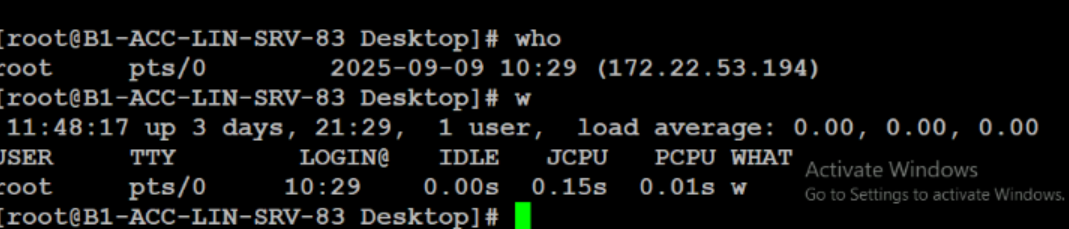
Numerical permissions

* **Read( r) = 4**
* **Write (w)=2**
* **Executable(x)=1**

**Chmod 7(r+w+x) 6(r+w) 5(r+x)**

****

* **Who=All users logged in**
* **W=All users logged in and what are they doing**
* **Whoami=shows the current user**

****

* **All the System variables are in UPPER CASE**

**Day 2:09/09/2025 Topic:Linux**

**Booting**

Bios -> MBR -> Boot Loader -> Kernel -> SystemD(PID 1) ->TARGETS

**RHEL 6< -> Init (slower) rest -> SystemD**

**Bios(Post and RAMTEST)**

* Checks the hardware (RAM, CPU, keyboard, etc.) to ensure everything is functioning.
* Displays error messages if something is wrong.

**BIOS/UEFI Initialization**:

* BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface) is firmware stored on the motherboard.
* It configures hardware settings and determines which device to boot from (hard disk, USB, etc.).

**Boot Device Selection**:

* BIOS/UEFI looks for a bootable device based on the boot order.
* Once found, it hands over control to the **bootloader** on that device.

**🖥️ Why is Preboot Important?**

* It ensures the system is ready and stable before the OS starts.
* It allows users to enter BIOS/UEFI settings (usually by pressing keys like F2, DEL, or ESC during startup).
* It can be used for troubleshooting, changing boot order, or enabling virtualization.

**Bootloader:**

**LILO=Linux loader used in earlier versions.**

**GRUB= GRand Unified Bootloader**

MBR =512 Bytes

|  |  |  |
| --- | --- | --- |
| Bootloader  446 bytes | Partition table  64 Bytes | Magic Number  2 Bytes |

16 Bytes \* Primary partition => 16 \*4=> 64 Bytes

**SystemD is the first Point to run When Boot reloaded**

**What is a Runlevel?**

A **runlevel** is a mode of operation in Unix-based systems that defines what services and processes should be running. It’s part of the **System V init** system, which was traditionally used to manage system startup and shutdown.

Each runlevel represents a different system state, such as:

* **Single-user mode**
* **Multi-user mode**
* **Graphical interface mode**
* **Shutdown or reboot**

**🔢 Common Runlevels**

Here’s a table of standard runlevels (may vary slightly by distribution):

| **Runlevel** | **Description** |
| --- | --- |
| 0 | Halt (shutdown the system) |
| 1 | Single-user mode (maintenance) |
| 2 | Multi-user mode (no networking) |
| 3 | Multi-user mode with networking |
| 4 | Undefined / Custom/Configurable |
| 5 | Multi-user mode with GUI (X11) |
| 6 | Reboot |

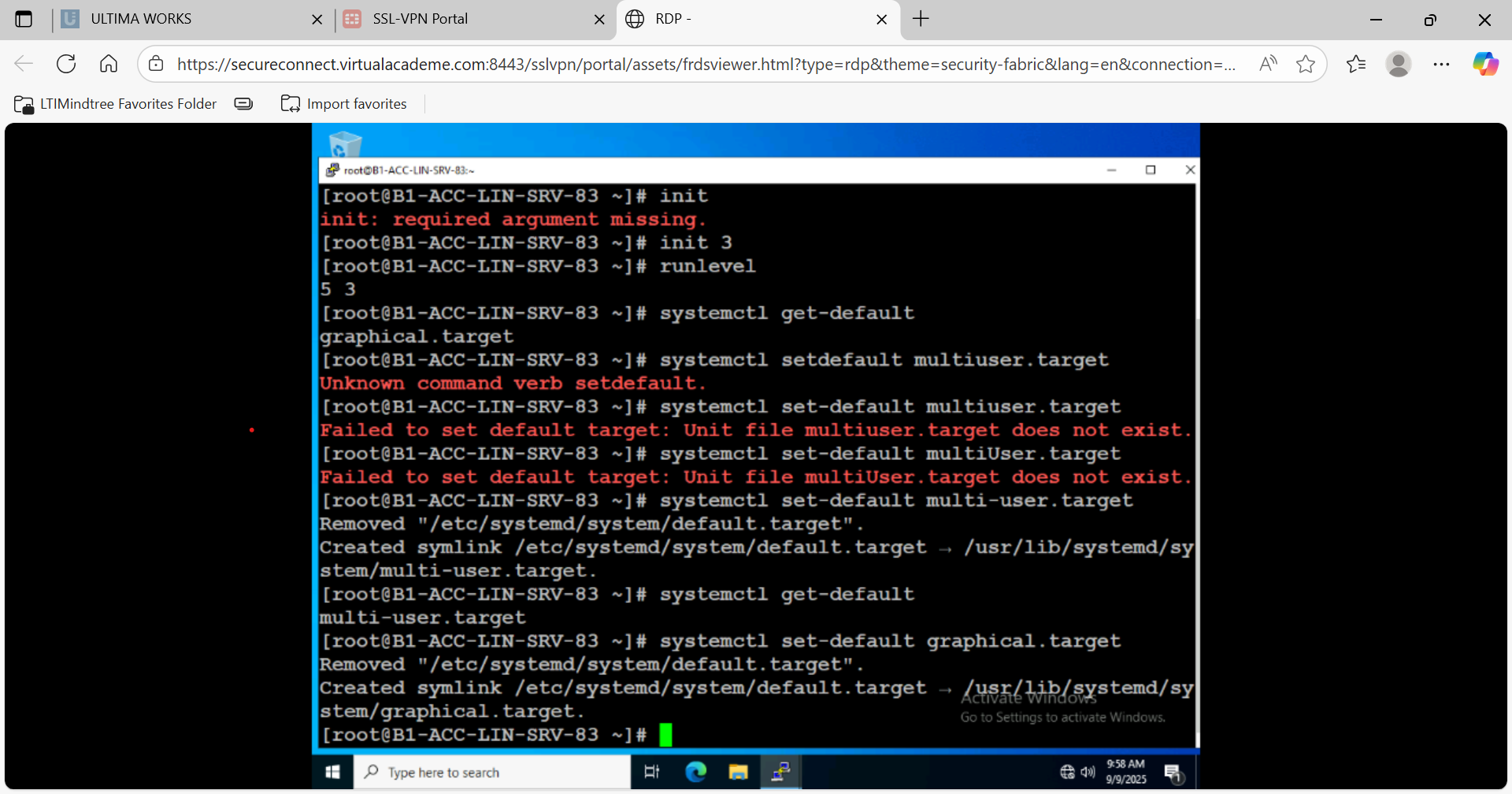
Traget in systemD:

Systemd uses ‘targets’ instead of runlevels.By default there are two main targets

* Multi-user.target:analogous to runlevel 3
* Graphical.target: analogous to runlevel 5

**Partition Table:**

For practice: SDA For exam:SDB And SDC



|  |  |  |  |
| --- | --- | --- | --- |
| P | P | P | E |

|  |  |  |  |
| --- | --- | --- | --- |
| L1 | L2 | L3 | L4 |

3 Primary partition and one extendable partition

E -cannot be configured and mounted only used for logical partitioning

**Commands:**

* Runlevel
  + Shows the run level partition
* Init
  + To initialize the runlevel
* Telinit
  + Same as init
* Systemctl get-default:
  + To get the default target
* System set-default target\_name:
  + To set the default target
  + A computer screen with text on it

    AI-generated content may be incorrect.
* Lsblk:
  + Shows the default Disk partitions
* Fdisk -l
  + To know the disk details
* Fdisk -l | more
  + **fdisk -l**
  + Lists **all available disks and their partitions**.
  + Shows details like:
  + Disk size
  + Partition type
  + Boot flag
  + Filesystem type
    - **🔹 | more**
      * This is a **pipe** (|) that sends the output of fdisk -l to the more command.
      * more lets you **view the output page by page**, which is useful if the list is long.
* Here are the **commands to create an extended partition and logical partitions** using fdisk in Linux:
  + **✅ Step 1: List Disks**
  + This shows all available disks (e.g., /dev/sda, /dev/sdb).
  + **✅ Step 2: Start Partitioning**
  + Choose the disk (example: /dev/sdb):
  + **✅ Step 3: Inside fdisk**
  + **Create Extended Partition**:
  + Press n → New partition
  + Choose e → Extended
  + Accept defaults or specify size (e.g., +2G)
  + **Create Logical Partitions inside Extended**:
  + Press n → New partition
  + Choose l → Logical
  + Repeat for as many logical partitions as needed.
  + **Write Changes**:
  + **✅ Step 4: Format Partitions**
  + Example for ext4: mkfs.ext4 /dev/sdb5 or mkfs -t xfs/dev/sdb5
  + **✅ Step 5: Mount Partition**
  + mkdir /mnt/test
  + mount /dev/sdb5 /mnt/test
* Free -h:
  + Shows the available memory
* Stat tf
  + Shows the stat data
* Umount /data:
  + To unmount the directory from the partition
* tail /etc/fstab
  + This command shows the **last few lines** of the /etc/fstab file, which contains filesystem mount information.

**File Systems in LINUX:**

Ext2:

* 2nd extended fs
* Default ih Redhat 9

Ext 3:

* Compatible with ext 2
* Ext2=extt3+journaling (storing the metadata)
* Meta data =Info about the files

Ext 4:

* Used before RHEL 6
* Compatible with ext 3
* More scalable than ext 3

**XFS file system is supported in RHEL 7,RHEL 8:**

**FSCK**

The fsck (File System Consistency Check) program in Linux is used to check and repair inconsistencies in file systems. It's especially useful when you're facing issues like:

* Mounting errors
* Unexpected shutdowns
* Disk corruption
* "bad superblock" or "wrong fs type" errors

🛠️ Basic Usage

Replace /dev/sdXn with your actual device name (e.g., /dev/sda1).

⚠️ Important Notes

* Unmount the filesystem first before running fsck:
* Never run fsck on a mounted filesystem, especially root (/) unless you're in recovery mode.
* umount /dev/sdXn

🔧 Common Options

| Option | Description |
| --- | --- |
| -y | Automatically answer "yes" to all prompts |
| -n | Do not make any changes; just report issues |
| -f | Force check even if filesystem seems clean |
| -V | Verbose output |

Example:

📂 Filesystem-Specific Tools

Depending on the filesystem, fsck calls different tools:

| Filesystem | Tool |
| --- | --- |
| ext2/ext3/ext4 | e2fsck |
| xfs | xfs\_repair (Note: fsck doesn't work on XFS) |
| btrfs | btrfs check |
| vfat | fsck.vfat |
| ntfs | ntfsfix (limited functionality) |

**RAID**

[**RAID Management System for Linux**](https://www.bing.com/ck/a?!&&p=9e3708d0e03e9247f3fe00b2a9efe841e4c04d8db9798b7bda669c57b88d4d80JmltdHM9MTc1NzM3NjAwMA&ptn=3&ver=2&hsh=4&fclid=39c80044-ddc3-6919-18b7-161ad9c36794&psq=raid+in+linux&u=a1aHR0cHM6Ly9kb2NzLnJlZGhhdC5jb20vZW4vZG9jdW1lbnRhdGlvbi9yZWRfaGF0X2VudGVycHJpc2VfbGludXgvMTAvaHRtbC9tYW5hZ2luZ19zdG9yYWdlX2RldmljZXMvbWFuYWdpbmctcmFpZA&ntb=1)

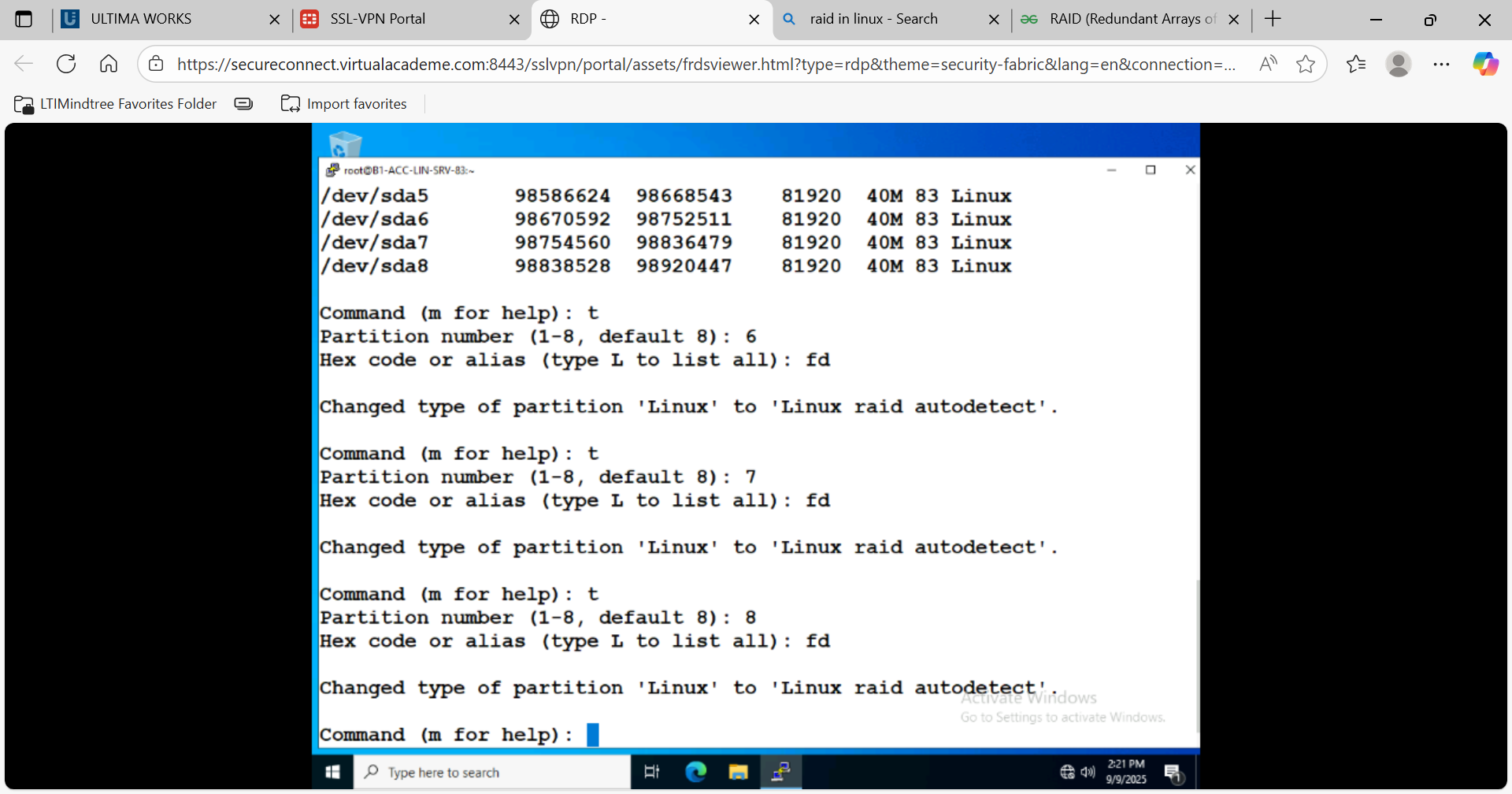
RAID (Redundant Array of Independent Disks) is a technology that combines multiple physical disks into a single logical unit to improve performance, redundancy,fault tolerance or both. Linux provides robust tools and utilities for managing RAID systems, including software-based RAID configurations.

**Key RAID Types in Linux**

1. **RAID 0 (Striping)**: Splits data across multiple disks for improved performance but offers no redundancy. Requires at least two disks.
2. **RAID 1 (Mirroring)**: Duplicates data across disks for redundancy. Requires at least two disks and halves usable storage.
3. **RAID 5 (Striping with Parity)**: Distributes parity across disks, allowing recovery from a single disk failure. Requires at least three disks.
4. **RAID 6 (Double Parity)**: Similar to RAID 5 but can tolerate two simultaneous disk failures. Requires at least four disks.
5. **RAID 10 (Combination of RAID 1 and 0)**: Combines mirroring and striping for both performance and redundancy. Requires at least four disks.

**Parity :- Data is stored in complex form You can recover data using this.**

**Stripe= write info in different hard disk**



**Why Use Type fd?**

This is typically done when you're setting up **software RAID** using tools like mdadm. The fd type helps the system recognize that these partitions are intended to be part of a RAID array.

t = is used to select a particular partition Number

* mdadm -C /dev/md1 -a yes -l 5 -n 3 /dev/sda{6,7,8}
  + is used to **create a RAID 5 array** using mdadm.
* -C=> is to create the raid
* -a => yes to create the RAID device file
* -l => Raid level
* -n=> no of RAID components
* -x => spare hdd
* mkfs -t xfs /dev/md1
  + mkfs: Stands for **make filesystem**.
  + -t xfs: Specifies the **XFS** filesystem type.
  + dev/md1: The RAID device you created with mdadm.
  + **Ensure the array is not mounted** before formatting.
  + **This will erase all data** on /dev/md1.
* Cat /proc/mdstat
  + This command displays the current status of all active RAID arrays managed by the md driver. It shows:
  + RAID devices (e.g., md0, md1)
  + RAID level (e.g., RAID1, RAID5)
  + Active disks and their sync status
  + Whether the array is rebuilding or degraded
* Mdadm –detail /dev/md1

This gives a detailed report about the RAID array /dev/md1, including:

* + RAID level
  + UUID
  + Number of active, working, and failed devices
  + State of the array (e.g., clean, degraded, rebuilding)
  + List of member devices and their roles
* Mdadm -f /dev/md1 /dev/sda7
  + This **marks** the device /dev/sda7 as **failed** in the RAID array /dev/md1.
  + Use this when a disk is behaving abnormally or you want to simulate a failure.
  + After this, the array will likely go into a **degraded** state if it was previously healthy.
* Mdadm -r /dev/md1 /dev/sda7
  + This **removes** the failed device /dev/sda7 from the array /dev/md1.
  + This step is only valid **after** the device has been marked as failed.
  + It prepares the array to accept a **replacement** device.
* Mdadm -a /dev/md1 /dev/sda 7
  + This will re-add the device to the array, and if it's a mirror (like RAID1), it will start **resyncing** the data.
* Man mdadm
* Umount /raiddata
* Mdadm –stop /dev/md1
* Lsblk

Shell is configured using variables ,aliases and functions

Eg: a=10 b=20 c=hello

Echo a=$a b=$b c=$c

Output = a=10 b=20 c=hello

**Shell Variables**

* **Defined in the shell or script globally**
* **Accessible throughout the shell session or script**
* Can be exported to child processes using export
* **Example**:
  + NAME="Bharath"
  + echo $NAME
  + export NAME

**Local Variables**

* **Defined inside a function**
* **Accessible only within that function**
* Prevents variable name conflicts in larger scripts
* Declared using the local keyword
  + greet() {
  + local NAME="Kumar"
  + echo "Hello, $NAME"
  + }
  + greet
  + echo $NAME  # This will be empty or undefined
* **1. vi script1**
  + **Purpose**: Opens the file script1 in the vi editor (or vim, depending on your system).
  + **Use case**: To **edit** or **view** the contents of the script.
  + **Example**: You might use this to write or modify shell commands inside script1.
* **2. sh script1**
  + **Purpose**: Executes the script using the **sh shell**, which is typically the Bourne shell or a compatible shell.
  + **Use case**: To run the script in a **POSIX-compliant** shell environment.
  + **Note**: If the script uses syntax specific to another shell (like Bash), this might cause errors.
* **3. bash script1**
  + **Purpose**: Executes the script using the **bash shell** (Bourne Again SHell).
  + **Use case**: To run scripts that use **Bash-specific syntax** or features.
  + **Note**: This is generally safer for modern scripts that rely on Bash extensions.
* **Summary**

| * **Command** | * **Action** | * **Shell Used** |
| --- | --- | --- |
| * vi script1 | * Opens script for editing | * N/A |
| * sh script1 | * Executes script with sh | * sh |
| * bash script1 | * Executes script with bash | * bash |

**What is a Shell?**

A shell is a program that takes your commands, interprets them, and instructs the operating system to execute them. It acts as a bridge between you and the system.

Types of Shell:

1. **Bourne Shell (sh)** :— One of the earliest UNIX shells, used mainly for scripting.
2. **Bash (Bourne Again Shell — bash) :** — The most popular shell, an improved version of Bourne Shell, used in most Linux distributions.
3. **C Shell (csh) and Tcsh :** — Inspired by the C programming language, mainly used for scripting.
4. **Korn Shell (ksh)** :— A mix of Bourne and C shell with additional features.
5. **Z Shell (zsh) :** — An advanced shell with better auto-completion and customization.

**🐚 1. Bourne Shell (sh)**

* **Original Unix shell** developed by Stephen Bourne.
* **Simple and fast**, but lacks many modern features.
* **Default shell** on many older Unix systems.
* **Script extension**: .sh

**🐚 2. Bourne Again Shell (bash)**

* **Most popular shell** on Linux systems.
* Backward-compatible with sh, but adds:
  + Command history
  + Tab completion
  + Arithmetic operations
  + Arrays
  + Improved scripting features
* **Script extension**: .sh or .bash

**🐚 3. C Shell (csh)**

* Syntax similar to the **C programming language**.
* Introduced features like:
  + Aliases
  + Job control
* Less commonly used today due to scripting limitations.

**🐚 4. TENEX C Shell (tcsh)**

* An **enhanced version of csh**.
* Adds:
  + Command-line editing
  + Filename completion
  + Better scripting support

**🐚 5. Korn Shell (ksh)**

* Developed by David Korn at AT&T.
* Combines features of sh and csh.
* Supports:
  + Arrays
  + Functions
  + Arithmetic
  + Job control
* Used in many commercial Unix systems.

**🐚 6. Z Shell (zsh)**

* A **powerful and flexible shell**.
* Combines features of bash, ksh, and tcsh.
* Popular for:
  + Customization
  + Plugins (e.g., Oh My Zsh)
  + Advanced tab completion
* Often used by developers and power users.

**🐚 7. Fish Shell (Friendly Interactive SHell)**

* Focuses on **user-friendliness** and **modern features**.
* Features:
  + Syntax highlighting
  + Autosuggestions
  + Web-based configuration
* Not POSIX-compliant, so scripts may not be portable.
* Echo $BASH\_VERSION
  + Tell me the version of the bash

**Shell startup or Shell login Scripts**

1. /ETC/PROFILe
2. ~/.bash\_profile
3. /etc/bashrc
4. ~/.bashrc
5. /etc/profile.d
6. /etc/skel

