Topics To be Covered

Week	Broader Topic	Lecture	Topics	Tools to be covered
1	File System	15-20	1. File System 1.2 Types of file system 1.3 Linux file system features 1.4 File system structure 2. Navigation Commands 2.1 pwd 2.2 cd 2.3 ls (list) 2.4 mkdir 2.5 rmdir 2.6 cp (copy) 2.7 mv (move) 2.8 rm (Remove) 2.9 find 2.10 touch 2.11 cat 3. Absolute and Relative Paths 4. Practice Questions	Linux

1 File system

A file system is a method or structure used to organize and manage files and directories on a storage device, such as a hard disk drive (HDD) or solid-state drive (SSD). It provides a way to store, retrieve, and organize data within the storage media. In Linux, there are various file systems available, each with its own characteristics and features.

Here are some key aspects of file systems:

1. **Data Organization:**
    - File systems organize data into files and directories, allowing users to logically group and access related information.
    - Files are individual units that store data, such as text, images, programs, or configuration files.
    - Directories (also called folders) are containers that hold files and other directories, forming a hierarchical structure.
2. **File Naming and Path:**
    - Files are identified by names, which can consist of alphanumeric characters and special symbols.
    - Paths specify the location of a file or directory within the file system hierarchy.
    - An absolute path starts from the root directory ("/") and provides the full path to the file or directory.
    - A relative path is specified relative to the current working directory.
3. **File System Features:**
    - File systems offer various features to enhance data management, performance, reliability, and security.
    - Examples include journaling (to improve reliability and recovery after system crashes), extended attributes (to store additional metadata), access control

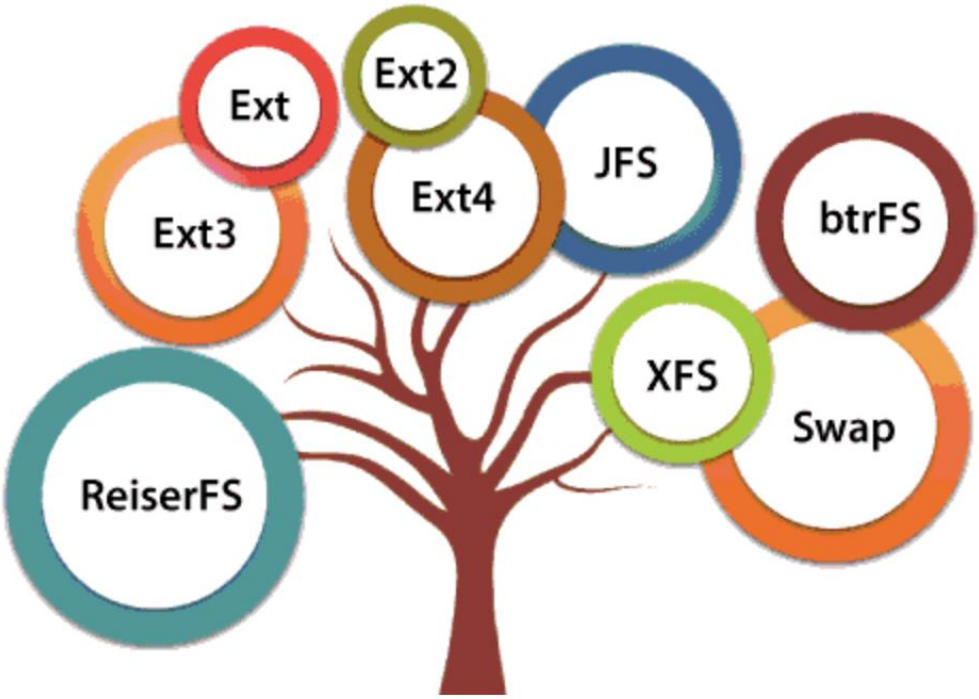
lists (for fine-grained permissions), and file system encryption (to protect data confidentiality).

4. **File System Types:**
    - Linux supports different file systems, such as Ext4, Ext3, Ext2, Btrfs, XFS, F2FS, and ZFS.
    - Each file system has its own optimizations, capabilities, and intended use cases.
    - Factors to consider when choosing a file system include performance requirements, reliability, scalability, compatibility, and specific features needed.
5. **Mounting:**
    - In Linux, file systems need to be mounted before they can be accessed.
    - Mounting is the process of attaching a file system to a specific directory (mount point) in the file system hierarchy.
    - This allows the operating system and applications to interact with the files and directories within the mounted file system.

File systems play a crucial role in managing data on Linux systems. They provide the underlying structure for storing and organizing files, and their choice can impact performance, reliability, and data security. Understanding file systems and their features is important for efficient data management and system administration.

1.2 Types of Linux File Systems

When we install the Linux operating system, Linux offers many file systems such as Ext, Ext2, Ext3, Ext4, JFS, ReiserFS, XFS, btrfs, and swap.



Linux supports multiple file systems, each with its own features, optimizations, and intended use cases. Here are some of the commonly used file systems in Linux:

1. **Ext4 (Fourth Extended File System):**
    - Ext4 is the default and most widely used file system in many Linux distributions.
    - It provides improved performance, reliability, and support for large file systems and file sizes.
    - Features include support for journaling, file system encryption (eCryptfs), extended attributes, and more.
    - Ext4 is backward compatible with its predecessor, Ext3, and can be upgraded from Ext2.

2. **Ext3 (Third Extended File System):**
    - Ext3 is an earlier version of the Ext4 file system with similar features but without some of the optimizations and capabilities.

- It supports journaling, which improves reliability and recovery in case of system crashes or power failures.
- Ext3 is compatible with Ext2, allowing for easy migration from Ext2 to Ext3 by adding journaling support.

3. **Ext2 (Second Extended File System):**
    - Ext2 is the second version of the Extended File System and is the predecessor of Ext3 and Ext4.
    - It does not support journaling, making it less resilient to crashes or unclean system shutdowns.
    - Ext2 is known for its simplicity, reliability, and widespread compatibility with different operating systems.

4. **Btrfs (B-Tree File System):**
    - Btrfs is a modern and feature-rich file system designed for Linux systems.
    - It offers advanced features such as snapshotting, checksums, RAID support, subvolumes, and online file system resizing.
    - Btrfs is known for its flexibility, scalability, and ability to handle large storage arrays efficiently.
    - While Btrfs has many compelling features, it may still be considered relatively new and might not be as widely adopted as Ext4.

5. **XFS (XFS File System):**
    - XFS is a high-performance file system initially developed by Silicon Graphics, Inc. (SGI).
    - It is optimized for scalability, parallelism, and handling large files and file systems.
    - XFS supports features such as journaling, file system snapshots, and online resizing.
    - XFS is commonly used in enterprise environments, especially for storage-intensive workloads.

6. **F2FS (Flash-Friendly File System):**
    - F2FS is a file system specifically designed for flash-based storage devices, such as solid-state drives (SSDs) and eMMC storage.
    - It is optimized for efficient wear leveling, garbage collection, and minimizing write amplification on flash memory.
    - F2FS aims to maximize the lifespan and performance of flash-based storage devices.

7. **ZFS (Zettabyte File System):**
    - ZFS is a robust and feature-rich file system developed by Sun Microsystems (now owned by Oracle).
    - It provides advanced capabilities like data integrity checks, automatic repair (scrubbing), built-in RAID support, and snapshotting.
    - ZFS has native support for managing storage pools and can efficiently handle large amounts of data.

These are just a few examples of file systems available in Linux. Other file systems, such as JFS, ReiserFS, and NILFS, also exist but may be less commonly used or have specific use cases. The choice of file system depends on factors like performance requirements, intended use case, hardware configuration, and personal preference.

1.3 Linux File System Features

Linux file systems offer various features that enhance data management, performance, reliability, and security. Here are some notable features commonly found in Linux file systems:

1. **Journaling:**
    - Journaling ensures file system consistency by maintaining a log (journal) of changes before they are committed to the main file system structures.
    - It helps recover the file system quickly in the event of a crash or power failure, reducing the chances of data corruption.

2. **Extended File Attributes:**
    - Extended attributes allow file systems to store additional metadata about files and directories beyond the standard file attributes (e.g., permissions, timestamps).
    - This feature enables the association of extended metadata, such as file capabilities, security labels, and user-defined tags, with files and directories.

3. **Access Control Lists (ACLs):**
    - ACLs provide fine-grained control over file and directory permissions beyond the traditional user, group, and other permissions.
    - ACLs allow specifying permissions for multiple users and groups, enabling more flexible and granular access control.

4. **File System Encryption:**
    - Some Linux file systems, such as Ext4 and Btrfs, support built-in file system encryption, allowing encryption at the file system level.
    - File system encryption helps protect data confidentiality by encrypting data stored on disk, ensuring its security even if physical access is compromised.

5. **Snapshotting:**

- File systems like Btrfs and ZFS offer snapshot capabilities, allowing users to create point-in-time copies of file systems.
- Snapshots provide efficient and space-saving mechanisms for backups, system rollbacks, and data versioning.

6. **Online Resizing:**
    - Online resizing enables file systems to be resized without unmounting or disrupting system operations.
    - This feature allows for dynamic adjustment of file system sizes to accommodate changing storage needs.

7. **RAID Support:**
    - Some file systems, such as XFS and Btrfs, offer built-in support for Redundant Array of Independent Disks (RAID) configurations.
    - RAID support allows combining multiple physical disks into logical arrays for enhanced performance, data redundancy, and fault tolerance.

8. **File System Checksums:**
    - Certain file systems, including Btrfs and ZFS, utilize checksums to detect and mitigate data corruption issues.
    - Checksums ensure data integrity by verifying the integrity of stored data and identifying potential errors or inconsistencies.

9. **Transparent Compression:**
    - Some file systems, like Btrfs and ZFS, support transparent compression of files and directories.
    - Transparent compression reduces storage space requirements by compressing data on the fly, resulting in improved disk utilization.

10. **Online Defragmentation:**

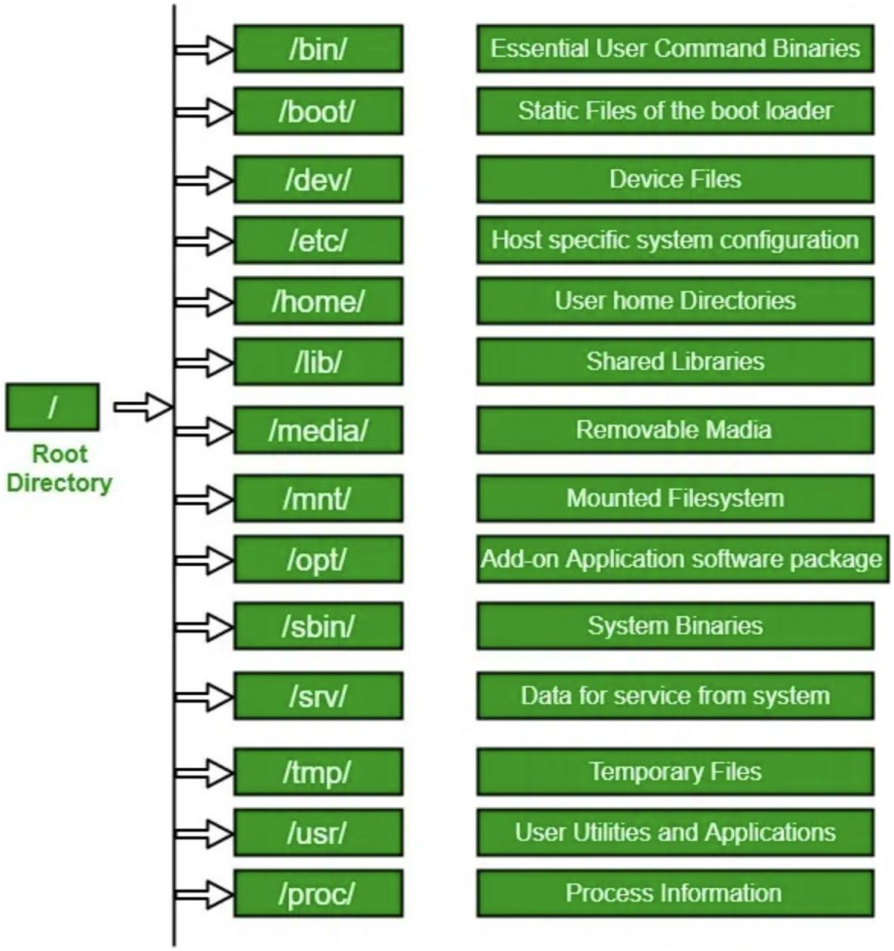
- File systems like Ext4 have built-in online defragmentation tools to optimize disk space usage and improve file system performance.
- Online defragmentation rearranges file data on disk to reduce fragmentation and improve file access speeds.

These features contribute to the overall functionality, performance, and reliability of Linux file systems, allowing efficient data management and protection. The availability of these features may vary depending on the specific file system used and the configuration options chosen during file system creation.

1.4 File system structure

The file system structure in Linux organizes files and directories in a hierarchical manner, starting from the root directory ("/"). The file system structure follows a tree-like structure,

with



directories (folders) containing files and other directories. Here's an overview of the key directories and their purposes in the Linux file system structure:

1. **/ (Root Directory):**
    - The root directory is the top-level directory in the file system hierarchy.
    - It contains all other directories and files in the system.

```
bash-3.2$ cd /
bash-3.2$ ls
Applications    Volumes         etc             sbin
Library         bin             home            tmp
System          cores           opt             usr
Users           dev             private         var
bash-3.2$
```

2. **/bin (Binary Programs):**
    - The /bin directory contains essential executable binaries (programs) required for basic system operation.
    - Common system utilities and commands, such as ls, cp, and rm, are stored here.

```
bash-3.2$ cd bin/
bash-3.2$ ls
[               dash            expr            ln              pwd             stty            zsh
bash            date            hostname        ls              realpath        sync
cat             dd              kill            mkdir           rm              tcsh
chmod           df              ksh             mv              rmdir           test
cp              echo            launchctl       pax             sh              unlink
csh             ed              link            ps              sleep           wait4path
bash-3.2$
```

3. /boot (Boot Files):

- The /boot directory contains files related to the system boot process.
- It includes the Linux kernel, initial RAM disk (initrd), bootloader configuration files (e.g., GRUB), and sometimes kernel modules.

4. /dev (Device Files):

- The /dev directory contains device files that represent hardware devices in the system.
- Each device file provides an interface for interacting with a specific hardware device, such as hard drives, keyboards, and network interfaces.

```
bash-3.2$ cd /dev
bash-3.2$ ls -lrt
total 1
crw-rw-rw-  1 root          wheel      0x3000003 Jun 19 15:10 zero
crw-rw-rw-  1 root          wheel     0x11000001 Jun 19 15:10 urandom
crw-rw-rw-  1 root          wheel     0x1e000000 Jun 19 15:10 uart.wlan-debug
crw-rw-rw-  1 root          wheel      0x400007f Jun 19 15:10 ttywf
crw-rw-rw-  1 root          wheel      0x400007e Jun 19 15:10 ttywe
crw-rw-rw-  1 root          wheel      0x400007d Jun 19 15:10 ttywd
crw-rw-rw-  1 root          wheel      0x400007c Jun 19 15:10 ttywc
crw-rw-rw-  1 root          wheel      0x400007b Jun 19 15:10 ttywb
crw-rw-rw-  1 root          wheel      0x400007a Jun 19 15:10 ttywa
crw-rw-rw-  1 root          wheel      0x4000079 Jun 19 15:10 ttyw9
crw-rw-rw-  1 root          wheel      0x4000078 Jun 19 15:10 ttyw8
crw-rw-rw-  1 root          wheel      0x4000077 Jun 19 15:10 ttyw7
crw-rw-rw-  1 root          wheel      0x4000076 Jun 19 15:10 ttyw6
crw-rw-rw-  1 root          wheel      0x4000075 Jun 19 15:10 ttyw5
crw-rw-rw-  1 root          wheel      0x4000074 Jun 19 15:10 ttyw4
```

5. /etc (Configuration Files):

- The /etc directory contains system-wide configuration files.

- Various configuration files for system services, network settings, user

```
bash-3.2$ cd /etc
bash-3.2$ ls -lrt
total 856
-rw-r--r--@  1 root  wheel      27 Feb  1  2022 ntp.conf
drwxr-xr-x   4 root  wheel     128 Feb  1  2022 dotnet
-rw-r--r--   1 root  wheel      66 Apr 19  2022 paths
-rw-r--r--   1 root  wheel     365 May 19  2022 hosts
-rw-r--r--   1 root  wheel    9335 Apr  1 22:16 zshrc_Apple_Terminal
-r--r--r--   1 root  wheel    3094 Apr  1 22:16 zshrc
-r--r--r--   1 root  wheel     255 Apr  1 22:16 zprofile
-rw-r--r--   1 root  wheel       0 Apr  1 22:16 xtab
drwxr-xr-x   6 root  wheel     192 Apr  1 22:16 wfs
drwxr-xr-x   5 root  wheel     160 Apr  1 22:16 uucp
-rw-r--r--   1 root  wheel    1316 Apr  1 22:16 ttys
-rw-r--r--   1 root  wheel      96 Apr  1 22:16 syslog.conf
```

accounts, and more are stored here

6. **/home (User Home Directories):**

- The /home directory contains home directories for individual users.
- Each user typically has a subdirectory within /home where their personal files and configurations are stored.

```
bash-3.2$ cd /home
bash-3.2$ ls -lrt
total 0
bash-3.2$
bash-3.2$
```

7. **/lib (Shared Libraries):**
    - The /lib directory contains shared libraries required by the system and other programs.
    - Shared libraries are collections of pre-compiled code that multiple programs can use, helping to reduce redundancy and improve efficiency.

8. **/mnt (Mount Points):**
    - The /mnt directory is used as a temporary mount point for manually mounting external or temporary file systems.
    - It provides a location to access files on external storage devices, such as USB drives or network shares.
9. **/opt (Optional Software):**
    - The /opt directory is used for installing optional, third-party software packages.
    - Software packages installed in this directory typically have their own subdirectories to keep their files organized.

```
bash-3.2$ cd /opt/
bash-3.2$ ls -lrt
total 0
drwxr-xr-x   3 root          wheel    96 Nov 16  2021 R
drwxr-xr-x  33 nitikaraghwa  admin  1056 Feb 21 15:09 homebrew
bash-3.2$
bash-3.2$
```

10. /sbin (System Binaries):

- The /sbin directory contains essential system administration binaries (programs) primarily used by the system administrator.
- These binaries are typically executed with administrative privileges and perform critical system management tasks.

```
bash-3.2$ cd /sbin/
bash-3.2$ ls -lrt
total 3848
-r-xr-xr-x  1 root  wheel   135824 Apr  1 22:16 umount
-rwxr-xr-x  1 root  wheel   170384 Apr  1 22:16 shutdown
-r-xr-xr-x  1 root  wheel   186224 Apr  1 22:16 route
-rwxr-xr-x  2 root  wheel   135552 Apr  1 22:16 reboot
-r-xr-xr-x  1 root  wheel   168592 Apr  1 22:16 quotacheck
-r-xr-xr-x  1 root  wheel   253264 Apr  1 22:16 ping6
-r-xr-xr-x  1 root  wheel   203120 Apr  1 22:16 ping
-r-xr-xr-x  1 root  wheel   501040 Apr  1 22:16 pfctl
-rwxr-xr-x  1 root  wheel   133760 Apr  1 22:16 nologin
-rwxr-xr-x  1 root  wheel   134032 Apr  1 22:16 nfsiod
-rwxr-xr-x  1 root  wheel   341680 Apr  1 22:16 nfsd
lrwxr-xr-x  1 root  wheel       63 Apr  1 22:16 newfs_udf -> /System/Library/Filesystems/udf.fs/Contents/Resources/newf
s_udf
lrwxr-xr-x  1 root  wheel       67 Apr  1 22:16 newfs_msdos -> /System/Library/Filesystems/msdos.fs/Contents/Resources/
newfs_msdos
lrwxr-xr-x  1 root  wheel       63 Apr  1 22:16 newfs_hfs -> /System/Library/Filesystems/hfs.fs/Contents/Resources/newf
```

11. /tmp (Temporary Files):

- The /tmp directory is used for storing temporary files that are created and accessed by various programs.
- Files in /tmp are typically deleted upon system reboot or periodically by the system.

```
bash-3.2$ cd /tmp
bash-3.2$ ls -lrt
total 16
drwxr-xr-x  2 root          wheel   64 Jun 19 15:10 powerlog
-rw-------  1 _mysql        wheel    5 Jun 19 15:10 mysqlx.sock.lock
srwxrwxrwx  1 _mysql        wheel    0 Jun 19 15:10 mysqlx.sock
-rw-------  1 _mysql        wheel    4 Jun 19 15:10 mysql.sock.lock
srwxrwxrwx  1 _mysql        wheel    0 Jun 19 15:10 mysql.sock
drwx------  3 nitikaraghwa  wheel   96 Jun 19 15:36 com.apple.launchd.Rj29yb7G25
drwxr-xr-x@ 4 nitikaraghwa  wheel  128 Jun 19 15:37 com.google.Keystone
bash-3.2$
```

12. **/usr (User Binaries and Libraries):**

- The /usr directory contains user-related binaries, libraries, documentation, and other non-essential files.
- It includes subdirectories such as /usr/bin (user executables), /usr/lib (libraries), /usr/share (shared data), and more.

```
bash-3.2$ cd /usr/
bash-3.2$ ls -lrt
total 0
drwxr-xr-x    5 root  wheel    160 Apr  1 22:16 standalone
drwxr-xr-x   42 root  wheel   1344 Apr  1 22:16 share
drwxr-xr-x  230 root  wheel   7360 Apr  1 22:16 sbin
drwxr-xr-x  347 root  wheel  11104 Apr  1 22:16 libexec
drwxr-xr-x   32 root  wheel   1024 Apr  1 22:16 lib
drwxr-xr-x  936 root  wheel  29952 Apr  1 22:16 bin
lrwxr-xr-x    1 root  wheel     25 Apr  1 22:16 X11R6 -> ../private/var/select/X11
lrwxr-xr-x    1 root  wheel     25 Apr  1 22:16 X11 -> ../private/var/select/X11
drwxr-xr-x    9 root  wheel    288 Apr 12 22:52 local
bash-3.2$
bash-3.2$
```

13. **/var (Variable Data):**

- The /var directory contains variable data files that change during system operation.

```
bash-3.2$ cd /var/
bash-3.2$ ls -lrt
total 0
drwxr-xr-x     3 root          wheel             96 Oct 23  2021 MobileSoftwareUpdate
drwxr-xr-x     4 root          wheel            128 Oct 23  2021 personalized_factory
drwxr-xr-x     3 root          wheel             96 Nov 24  2022 mobile
drwxr-x---    11 root          wheel            352 Mar 10 20:55 root
drwxr-xr-x     3 root          wheel             96 Apr  1 22:16 yp
drwxr-xr-x     6 root          wheel            192 Apr  1 22:16 spool
drwxr-xr-x     3 root          wheel             96 Apr  1 22:16 select
drwxr-xr-x     2 daemon        wheel             64 Apr  1 22:16 rwho
drwxr-xr-x     4 root          wheel            128 Apr  1 22:16 rpc
drwxr-xr-x     2 root          wheel             64 Apr  1 22:16 netboot
drwxr-xr-x     3 root          wheel             96 Apr  1 22:16 msgs
```

- It includes files such as logs (/var/log), spool files (/var/spool), temporary files (/var/tmp), and other data specific to running services.

These are some of the key directories in the Linux file system structure. However, keep in mind that Linux distributions may have additional directories or use slightly different conventions based on their specific configurations and purposes.

2. Navigation Commands

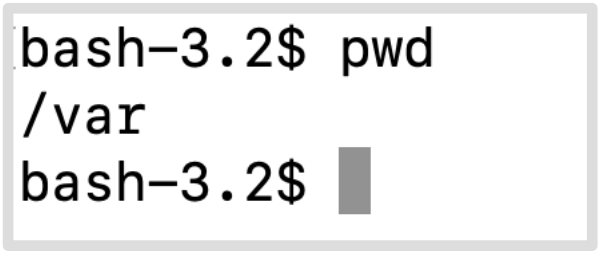
In Linux, navigating through the file system is primarily done using command-line navigation commands. Here are some commonly used navigation commands:

2.1 pwd (Print Working Directory):

The pwd command in Linux stands for "Print Working Directory." It is used to display the current working directory, which represents your current location within the file system.

When you run the pwd command, it will print the absolute path of the current working directory to the terminal.

- Displays the current working directory, which represents your current location in the file system.



The pwd command is handy when you need to know your current location in the file system while working in the command line interface. It can help you keep track of the directory you're in and provide you with the necessary information to navigate to other directories or perform specific operations on files and directories.

2.2 cd (Change Directory):

The cd command in Linux stands for "Change Directory." It is used to navigate between different directories within the file system.

Here's how you can use the cd command:

- **Change to a specific directory:**

  cd /path/to/directory

  This command changes the current working directory the actual path of the directory you want to navigate to.

- **Change to the home directory:**

  cd

  Running cd without any arguments will take you to your home directory.

- **Change to the parent directory:**

  cd ..

  This command moves you one level up in the directory hierarchy to the parent directory of the current working directory.

```
bash-3.2$ cd log/apache2/
bash-3.2$
bash-3.2$ ls
bash-3.2$
bash-3.2$
```

```
bash-3.2$ cd ..
bash-3.2$
bash-3.2$ cd ~
bash-3.2$
bash-3.2$ ls
Applications                            docker-machine-driver-vmware_darwin_amd64
Applications (Parallels)                eclipse
Desktop                                 eclipse-workspace
Documents                               encryptedsecret.txt
Downloads                               iCloud Drive (Archive)
IdeaProjects                            key.txt
Library                                 list.py
```

- **Change to the previous directory:**

  cd -

  This command takes you back to the previous directory you were in before the current one.

- **Use a relative path:**

  cd directory_name

  If you want to navigate to a directory that is within the current directory, you can use the directory name directly without specifying the complete path.

- **Use tab completion:**

Typing a partial directory name and pressing the Tab key can auto-complete the directory name if there is a unique match, or show you available options if there are multiple matches.

- Use environment variables:

cd $VARIABLE_NAME

You can use environment variables to specify a directory. Replace "VARIABLE_NAME" with the name of the environment variable holding the desired directory path.

The cd command is a fundamental command for navigating the file system in Linux. It allows you to move to different directories, explore the file hierarchy, and perform operations on files and directories within the current working directory.

2.3 ls (List):

The ls command in Linux is used to list the files and directories in the current working directory or a specified directory.

Basic usage:

bash

ls

Running ls without any arguments lists the files and directories in the current working directory.

Common options:

- -l: Long format listing. Displays detailed information about files and directories, including permissions, ownership, size, and modification timestamp.
- -a: Include hidden files. Shows all files and directories, including those with names starting with a dot (hidden files).

- -h: Human-readable file sizes. Displays file sizes in a more readable format, such as "1K", "2M", "3G", etc.
- -t: Sort by modification time. Lists files and directories in descending order based on the modification timestamp.
- -r: Reverse order. Reverses the order of listing, showing files and directories in reverse order.

Examples:

ls -l

Displays a long format listing of files and directories in the current working directory.

```
bash-3.2$ ls -l
total 144184
drwx------@    4 nitikaraghwa  staff       128 Sep 28  2022 Applications
drwxr-xr-x@    4 nitikaraghwa  staff       128 Feb 10  2022 Applications (Parallels)
drwx------+   33 nitikaraghwa  staff      1056 Jun 20 11:43 Desktop
drwx------+    4 nitikaraghwa  staff       128 Jun  6 18:33 Documents
drwx------+ 1513 nitikaraghwa  staff     48416 Jun 20 11:14 Downloads
drwxr-xr-x    18 nitikaraghwa  staff       576 Apr  4  2022 IdeaProjects
```

ls -a

Lists all files and directories, including hidden files, in the current working directory.

```
bash-3.2$ ls -a
.                                       .zsh_sessions
..                                      .zshenv
.CFUserTextEncoding                     .zshenv.save
.DDLocalBackups                         Applications
.DDPreview                              Applications (Parallels)
.DS_Store                               Desktop
.IdentityService                        Documents
.PenTablet                              Downloads
```

ls /path/to/directory

Lists the files and directories in the specified directory ("/path/to/directory") rather than the current working directory.

The ls command provides a quick way to view the contents of a directory and get basic information about files and directories. It is a versatile command with various options to customize the output based on your requirements.

2.4 mkdir (Make Directory):

The mkdir command in Linux is used to create new directories (folders) within the file system. It allows you to create one or multiple directories at once, with options to set permissions and specify the directory path.

Here's the basic usage of the mkdir command:

- Create a single directory:

  mkdir directory_name

  This command creates a new directory with the specified directory_name in the current working directory.

- Create multiple directories:

  mkdir directory1 directory2 directory3

You can create multiple directories by specifying their names as separate arguments. This will create multiple directories in the current working directory.

- Create a directory with specific permissions:

  mkdir -m permissions directory_name

  The -m option is used to set specific permissions for the created directory. Replace permissions with the desired permission value, such as "755" or “drwxr-xr-x".

- Create directories with nested structure:

  mkdir -p path/to/directory

  The -p option allows you to create directories with a nested structure. If any intermediate directories in the path don't exist, they will be created along with the final directory.

- Create directories with parent directory creation:

  mkdir -p parent_directory/new_directory

  This command creates a new directory new_directory within the parent_directory. If the parent_directory doesn't exist, it will be created.

The mkdir command provides a convenient way to create directories in Linux. It is commonly used to organize files, create directory structures, and prepare directories for file

operations. Exploring different options and scenarios with mkdir will help you become more proficient in managing directories in Linux.

2.5 rmdir (Remove Directory):`

- Deletes an empty directory.

```
sh-3.2# rmdir demo
sh-3.2#
```

• Example: rmdir directory_to_delete removes the empty directory named "directory_to_delete".

2.6 cp (Copy):

The cp **command** in Linux is used to copy files and directories from one location to another. It allows you to duplicate files, create backups, and transfer data between directories or devices.

Here's the basic usage of the cp command:

```
sh-3.2# cd work/
sh-3.2#
sh-3.2# pwd
/Users/nitikaraghwa/work
sh-3.2#
sh-3.2# cp first /Users/nitikaraghwa/demob
sh-3.2# cd /Users/nitikaraghwa/demob
sh-3.2# ls
first
sh-3.2#
```

- Copy a file to a specific location:

  cp source_file destination_directory

  This command copies the source_file to the specified destination_directory.

- Copy multiple files to a directory:

  cp file1 file2 file3 destination_directory

  You can copy multiple files by listing them as separate arguments followed by the destination_directory.

- Copy a directory and its contents:

  cp -r source_directory destination_directory

  The -r option is used to copy directories and their contents recursively. It ensures that

all files and subdirectories within the source_directory are copied to the destination_directory.

- Preserve file attributes and timestamps:

cp -a source_file destination_file

The -a option (or --archive) preserves the file attributes, permissions, timestamps, and symbolic links when copying files.

- Prompt for confirmation:

cp -i source_file destination_directory

The -i option (or --interactive) prompts for confirmation before overwriting an existing file in the destination_directory.

- Copy with a different filename:

cp source_file new_file_name

You can specify a different filename for the copied file in the destination_directory.

The cp command is a versatile tool for copying files and directories in Linux. It provides several options to control the copying process, such as preserving attributes, prompting for confirmation, and recursively copying directories. Experimenting with different options and scenarios will help you become more proficient in using the cp command.

2.7 mv (Move/Rename):

- Moves or renames files and directories.
    - Example:
    - mv old_name new_name: Renames a file or directory from "old_name" to "new_name".
    - mv source_file destination_directory: Moves "source_file" to “destination_directory".

```
sh-3.2# echo "bye" > second
sh-3.2#
sh-3.2# pwd
/Users/nitikaraghwa/demob
sh-3.2#
sh-3.2#
sh-3.2# mv second /Users/nitikaraghwa/work/
sh-3.2# cd /Users/nitikaraghwa/work/
sh-3.2# ls
first   second
sh-3.2#
sh-3.2#
```

2.8 rm (Remove):

- Deletes files and directories.
    - Common options:
    - rm file: Deletes a file.
    - rm -r directory: Deletes a directory and its contents recursively.
    - rm -f file: Forces the removal of a file without prompting for confirmation.

```
sh-3.2# cd /Users/nitikaraghwa/work/
sh-3.2# ls
first   second
sh-3.2#
sh-3.2# rm second
sh-3.2# cd ..
sh-3.2# rm -r work/
sh-3.2# ls
```

2.9 find:

- Searches for files and directories in a directory hierarchy based on various criteria.
- Example: find /path/to/directory -name "filename" searches for files with the specified name in the given directory and its subdirectories.

```
sh-3.2# find . -name first
./demob/first
```

10. touch:

The touch command in Linux is used to create new empty files or update the access and modification timestamps of existing files. It is a versatile command that allows you to modify file timestamps or create new files.

Here's the basic usage of the **touch** command:

- Create a new file:

  touch filename

  This command creates a new empty file with the specified filename. If the file already exists, the command updates its access and modification timestamps to the current time without modifying the file's content.

- Create multiple files:

  touch file1 file2 file3

You can

```
sh-3.2# touch second
sh-3.2#
sh-3.2# touch -t 202106191230.30 third
sh-3.2# ls -lrt
total 8
-rw-r--r--  1 root  staff  0 Jun 19  2021 third
-rw-r--r--  1 root  staff  6 Jun 20 11:50 first
-rw-r--r--  1 root  staff  0 Jun 20 12:00 second
sh-3.2#
sh-3.2# touch abc1 abc2 abc3
sh-3.2# ls -lrt
total 8
-rw-r--r--  1 root  staff  0 Jun 19  2021 third
-rw-r--r--  1 root  staff  6 Jun 20 11:50 first
-rw-r--r--  1 root  staff  0 Jun 20 12:00 second
-rw-r--r--  1 root  staff  0 Jun 20 12:00 abc1
-rw-r--r--  1 root  staff  0 Jun 20 12:00 abc2
-rw-r--r--  1 root  staff  0 Jun 20 12:00 abc3
sh-3.2#
```

create multiple files by specifying their names as separate arguments. This will create empty files with the specified names.

- Update timestamps:

touch -c filename

The -c option is used to update the access and modification timestamps of an existing file without creating a new file. If the file doesn't exist, it will not be created.

- Set specific timestamps:

  touch -t YYYYMMDDHHMM.SS filename

  The -t option allows you to set specific timestamps for a file. Replace YYYYMMDDHHMM.SS with the desired timestamp in the format “YearMonthDayHourMinute.Second".

- Use wildcards:

  touch *.txt

  You can use wildcards to create multiple files based on a pattern. In this example, the command creates empty files with the ".txt" extension in the current directory.

```
sh-3.2# touch demo{1..5}.txt
sh-3.2# ls -lrt
total 8
-rw-r--r--  1 root  staff  0 Jun 19  2021 third
-rw-r--r--  1 root  staff  6 Jun 20 11:50 first
-rw-r--r--  1 root  staff  0 Jun 20 12:00 second
-rw-r--r--  1 root  staff  0 Jun 20 12:00 abc1
-rw-r--r--  1 root  staff  0 Jun 20 12:00 abc2
-rw-r--r--  1 root  staff  0 Jun 20 12:00 abc3
-rw-r--r--  1 root  staff  0 Jun 20 12:01 demo1.txt
-rw-r--r--  1 root  staff  0 Jun 20 12:01 demo2.txt
-rw-r--r--  1 root  staff  0 Jun 20 12:01 demo3.txt
-rw-r--r--  1 root  staff  0 Jun 20 12:01 demo4.txt
-rw-r--r--  1 root  staff  0 Jun 20 12:01 demo5.txt
sh-3.2#
```

The touch command is a handy tool for creating and modifying file timestamps in Linux. It is commonly used to update timestamps for backup purposes, force a rebuild of files, or create placeholder files for scripting purposes.

2.11 Cat command

The cat command in Linux and Unix-like operating systems is used to concatenate and display the contents of files. It is a versatile command that can be used for various purposes. Here are some common use cases of the cat command:

- Display File Contents:
- bash
- Copy code

cat filename

- This command displays the contents of the specified file (filename) on the terminal. If multiple file names are provided, cat will display the contents of all the files sequentially.
- Concatenate Files:
- bash
- Copy code

cat file1 file2 > outputfile

- The cat command can be used to concatenate the contents of multiple files and save the combined output into a new file (outputfile in this example). The > symbol is used for output redirection.
- Append to a File:
- bash
- Copy code

cat file1 >> file2

- This command appends the contents of file1 to the end of file2. The >> symbol is used for appending output.
- Display Line Numbers:
- bash

- Copy code

cat -n filename

- The -n option adds line numbers to the output, showing the line number before each line of the specified file.
- Create a New File:
- bash
- Copy code

cat > filename

- This command allows you to create a new file (filename) and start entering text. Press Ctrl + D to save the file.
- Display Non-Printable Characters:
- bash
- Copy code

cat -v filename

- The -v option displays non-printable characters in a visible format, making them visible as escape sequences.
- Display Tab Characters:
- bash
- Copy code

cat -T filename

- The -T option displays tab characters as ^I, making them visible and distinguishing them from spaces.

These are just a few examples of the cat command's usage. The cat command is simple but powerful, and it can be combined with other commands and options to perform more complex operations on files. You can explore additional options and combinations by referring to the command's manual page (man cat).

3. Absolute and Relative Paths

An absolute path in Linux refers to the complete and exact location of a file or directory in the file system, starting from the root directory ("/").

The root directory is the top-level directory in the file system hierarchy, denoted by "/". From the root directory, each directory in the path is specified, separated by forward slashes ("/"), until reaching the target file or directory.

Here's an example of an absolute path:

/home/user/Documents/file.txt

In this example, "/home/user/Documents" represents the path to a directory, and "file.txt" is the name of the file within that directory. The absolute path specifies the exact location of the file "file.txt" in the file system.

Absolute paths are independent of the current working directory. They provide an unambiguous way to refer to a file or directory from any location in the file system.

Absolute paths are typically used when you need to directly reference a file or directory and want to ensure the precise location is specified. They are useful when performing operations on files or directories across different locations in the file system or when writing scripts that need to access specific files or directories.

It's important to note that absolute paths start from the root directory ("/") and remain the same regardless of the current working directory.

A relative path in Linux refers to the location of a file or directory relative to the current working directory. Unlike absolute paths that start from the root directory ("/"), relative paths are specified based on the current location in the file system.

Here's an example to illustrate relative paths:

Assume the current working directory is "/home/user/" and there are two directories within it: "Documents" and "Pictures." The "Documents" directory contains a file named "report.txt."

- To refer to the "Documents" directory from the current working directory, you can use a relative path:

  Documents/

  This path specifies that the "Documents" directory is located within the current working directory.

- To refer to the "report.txt" file within the "Documents" directory using a relative path:

  Documents/report.txt

  This path indicates that the "report.txt" file is located within the "Documents" directory, which is in the current working directory.

Relative paths are context-dependent and rely on the current working directory to determine the target file or directory. They are useful when navigating and working within a specific directory hierarchy.

Relative paths are often used when you want to refer to files or directories that are within the same directory or its subdirectories. They simplify referencing files or directories without needing to provide the complete path starting from the root directory.

It's important to note that relative paths may change depending on the current working directory. Therefore, it's crucial to be aware of your current location when using relative paths.

4 . Practice Questions

1. touch:
   - How can you create a new file named "my_file.txt" using the touch command?
   - How can you update the access and modification timestamps of an existing file named "existing.txt" without modifying its content?
   - Can you specify a specific timestamp using the touch command? If yes, how?

2. cp:

- How can you copy a file named "file1.txt" to a directory named "destination" using the cp command?
- How can you copy an entire directory named "my_directory" to another location while preserving its contents and structure?
- What flag can you use with cp to prompt for confirmation before overwriting an existing file?

3. mkdir:
   - How can you create a new directory named "my_folder" in the current working directory using the mkdir command?
   - Can you create multiple directories at once with the mkdir command? If yes, how?
   - How can you create a directory with specific permissions using the mkdir command?
4. cd:
   - How can you change the current working directory to your home directory using the cd command?
   - How can you navigate one level up from the current directory using the cd command?
   - Can you use a relative path with the cd command? If yes, how?
5. ls:
   - How can you list all files and directories in the current working directory using the ls command?
   - How can you list the files and directories in a specific directory, such as "/home/user/documents," using the ls command?
   - How can you use the ls command to display file sizes in human-readable format?
6. pwd:

- How can you display the current working directory using the pwd command?
- Is the output of the pwd command an absolute path or a relative path?
- Can you use the pwd command to change the current working directory?

These practice questions should help you reinforce your understanding of the touch, cp, mkdir, cd, ls, and pwd commands in Linux. Feel free to explore and experiment with these commands to deepen your knowledge and familiarity with them.