UPES

Xebia

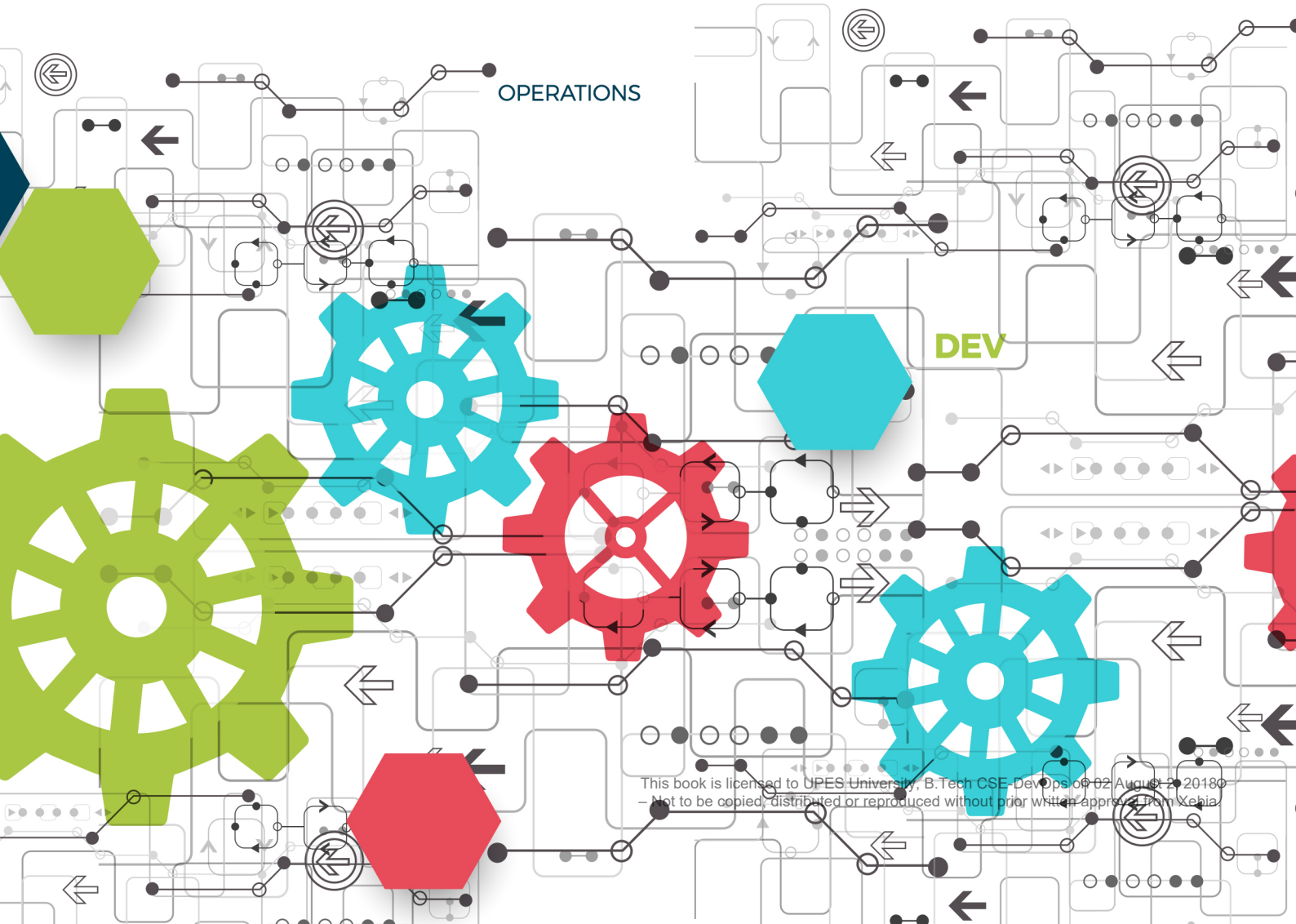
2 YEAR

B.Tech Computer Science and Engineering in DevOps

DEVOPS AUTOMATION

Semester 03 | Facilitator Handbook

Release 1.0.0



Copyright & Disclaimer

B. TECH CSE with Specialization in DevOps

Version 1.0.0

Acknowledgements

We would like to sincerely thank the experts who have contributed to and shaped B. TECH CSE with Specialization in DevOps. Version 1.0.0

SME

Rajagopalan Varadan
A tech enthusiast who loves learning and working with cutting-edge technologies like DevOps, Big Data, Data science, Machine Learning, AWS & Open stack

Course Reviewers.

Aditya Kalia | Xebia
Maneet Kaur | Xebia
Sandeep Singh Rawat | Xebia
Abhishek Srivastava | Xebia
Rohit Sharma | Xebia

Review Board Members.

Anand Sahay | Xebia

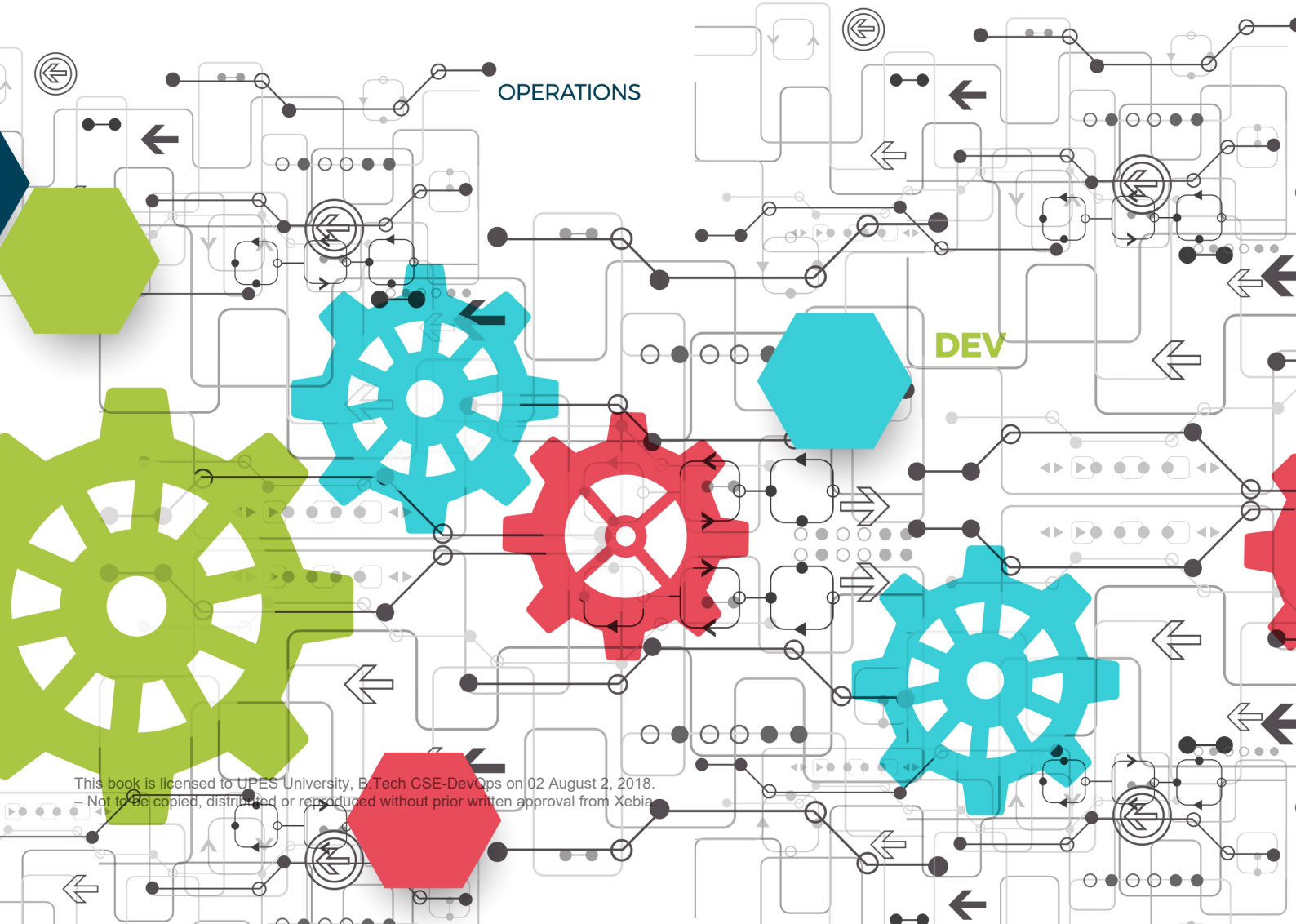
Xebia Group consists of seven specialized, interlinked companies: Xebia, Xebia Academy, XebiaLabs, StackState, GoDataDriven, Xpirit and Binx.io. With offices in Amsterdam and Hilversum (Netherlands), Paris, Delhi, Bangalore and Boston, we employ over 700 people worldwide. Our solutions address digital strategy; agile transformations; DevOps and continuous delivery; big data and data science; cloud infrastructures; agile software development; quality and test automation; and agile software security.

ODW is dedicated to provide innovative and creative solutions that contribute in growth of emerging technologies. As a learning experience provider, ODW strengths include providing unique, up to date content by combining industry best practices with leading edge technology. ODW delivers high quality solutions and services which focus on digital learning transformation.

B.Tech Computer Science and Engineering in DevOps

DEVOPS AUTOMATION

MODULE 3
Interacting with Linux Environment



Contents

Module Learning Objectives	**1**
Module Topics	**1**
1. Understanding Linux File System	**2**
2. User Groups and Permissions	**14**
3. File Permissions	**19**
4. Working with Bash	**23**
In a nutshell, we learnt:	**26**

MODULE 3

Interacting with Linux Environment

This module is the third module of the course and talks about Interacting with Linux Environment.

Module Learning Objectives

At the end of the Module you would be able to learn the following

- Explain Linux file system concepts.
- Describe how to create users and groups.
- Describe how to file permissions.
- Explain the basics of Bash.

Module Topics

Let us take a quick look at the topics we will cover in this module:

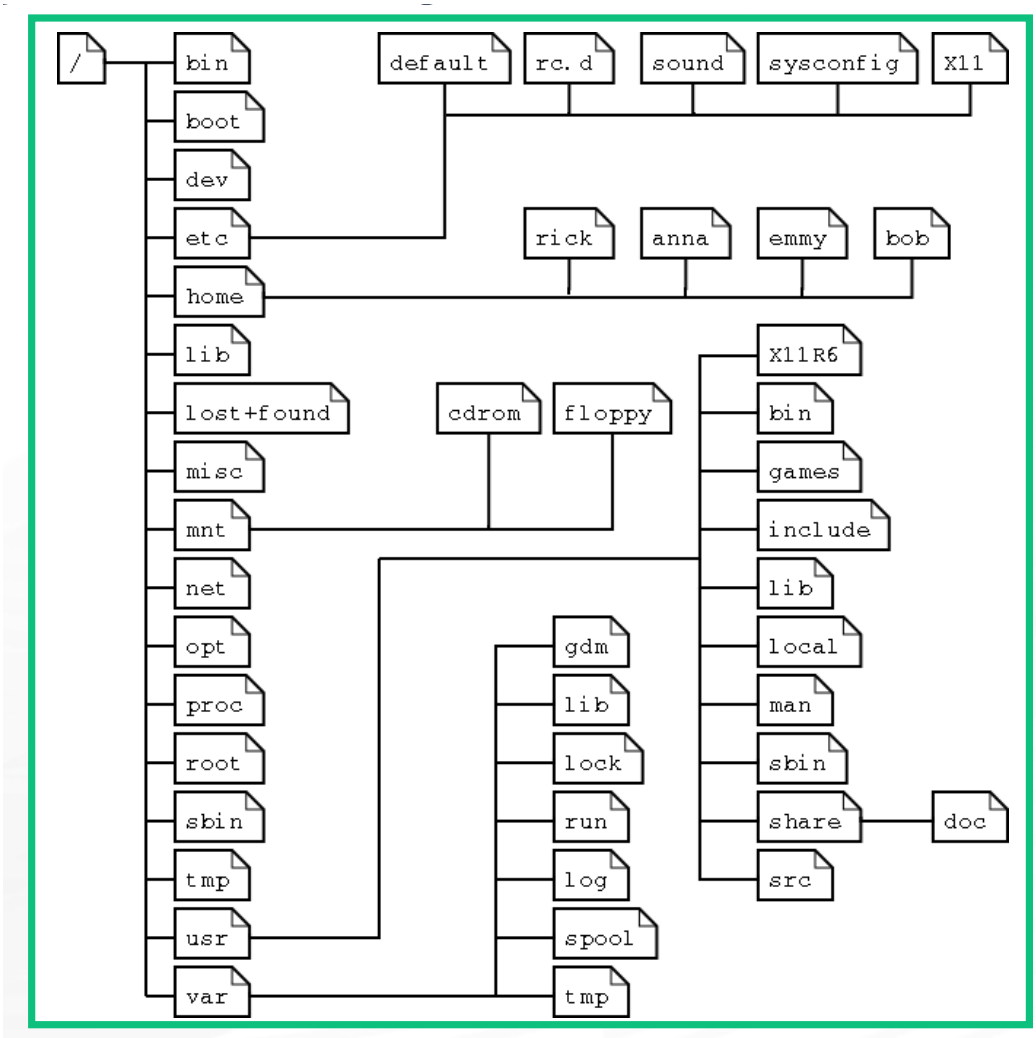
1. Understanding Linux File System.
2. Creating Users and Groups.
3. Creating File Permissions.
4. Working with Bash.

You will learn about the following topics in this module:

- Development Delivery Pipeline Overview
- Automating the Build Pipeline

- RAD (Rapid Application Development)
- Code Generation
- MDA/MDD (Model-Driven Architecture/Development)

1. Understanding Linux File System



You will now learn about the next topic in this module. This topic discusses the Linux File System.

1.1 Everything is a File

Consider you have a Linux Operating System. Then which among the following are files?

⇥ Scanners and Printers.	⇥ HDD and System Partitions.
⇥ Kernel parameters.	⇥ File directories.
⇥ Sockets & Ports.	⇥ Office documents.
⇥ Logical Volumes.	⇥ Bash scripts.

- The result can be surprising. Every single thing mentioned above is a file in the Linux Operating System.
- That gives a lot of opportunities to manipulate the operating system and perform the operations in a far easier way.

All of us know what a file is and in fact, the operating system itself is just a group of files. But the surprising part is even the mouse, keyboard and monitor are all files. But not limited to them. The Linux operating system is designed in a way that everything is stored in a file and this provides a greater robustness to manage and administer the operating system and its functions efficiently.

1.2 The Linux System

A Linux system basically has three major components:

The Linux File System (LFS):

Organizes data in a systematic way.

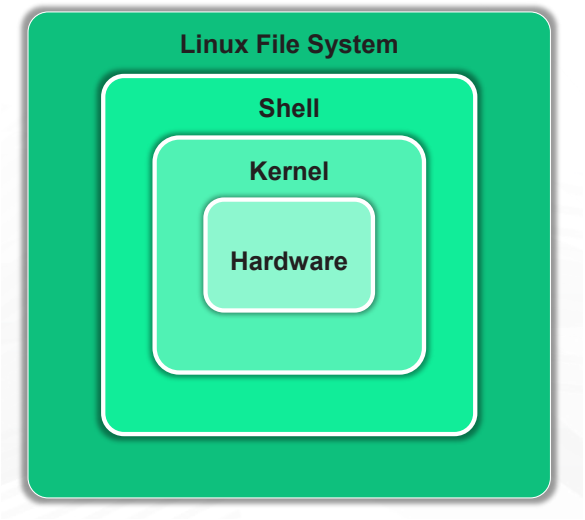
Shell:

Provides user interface to run commands.

Kernel:

The core program that manages system hardware devices.

Overview of the Linux system



You will be learning the components of Linux system in this slide.

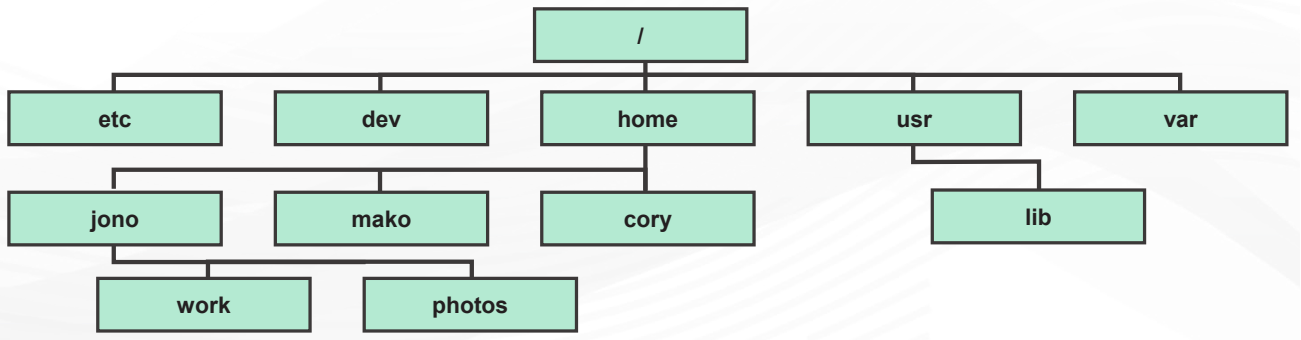
A Linux system basically has three major components:

- **The Linux File System (LFS)** - Organizes the data in a systematic way.
- **Shell** - Provides user interface to run commands.
- **Kernel** - The core program that manages the system hardware devices.

Collectively LFS, Shell and Kernel provide a way to interact with system and an environment to run commands and manage the data.

1.3 Introduction to Linux File System

- The Linux directory starts at the root directory (/). This is the top of the file system tree.
- There are various kinds of data storage formats like EXT3, EXT4, BTRFS, XFS, NTFS, FAT and NFS.
- All the file systems have their own structure that defines how the file is stored and accessed.
- A mount point is an empty directory on a Linux filesystem where a partition formatted with a specific file system can be mounted over.



You will be learning about the introduction to the Linux File System in this slide. The important points are:

- The file system provides a way to organize the data in a physical hard disk or logical partition by providing an address to the file and associating the file with a metadata.
- This metadata contains information about the file's size, author, timestamp and permissions to other users and groups.
- All the file systems have their own method of allocating the spaces on your disk and the format of the metadata varies for each of them.
- A Linux Operating system can have one or more partitions mounted on it and each partition can be formatted with any type of filesystem.

1.4 Linux File System

- The File system provides space for non-volatile storage of data.
- Provides a namespace, a methodology for naming and organizing; defines the logical structure of data on a disk, such as the use of directories for organizing files instead of placing them all together.
- The Metadata structure for providing a logical foundation for the namespace.
- The Filesystems require an Application Programming Interface (API) for getting access to system function calls, which manipulate file system objects like files and directories.
- The Modern file systems provide a security model, a scheme for defining access rights to files and directories.

Instructor will explain the various features of Linux File System.

The Linux file system provides space for non-volatile storage of data. The namespace is for naming and organizing the files in the system. Like Unix, everything in Linux is a file. If it's not a file, it's a process. Let's look at the most important concepts of File System now.

File system Metadata:

Metadata includes:

- Data structures required to support a hierarchical directory structure
- Structures to determine which blocks of space on the disk are used and which are available
- Structures that allow for maintaining the names of the files and directories
- Information about the files such as their size and times they were created, modified or last accessed
- Location(s) of the data belonging to the file on the disk.

The other metadata is used to store high-level information about the subdivisions of the disk, such as logical volumes and partitions. This higher-level metadata and the structures it represents contain

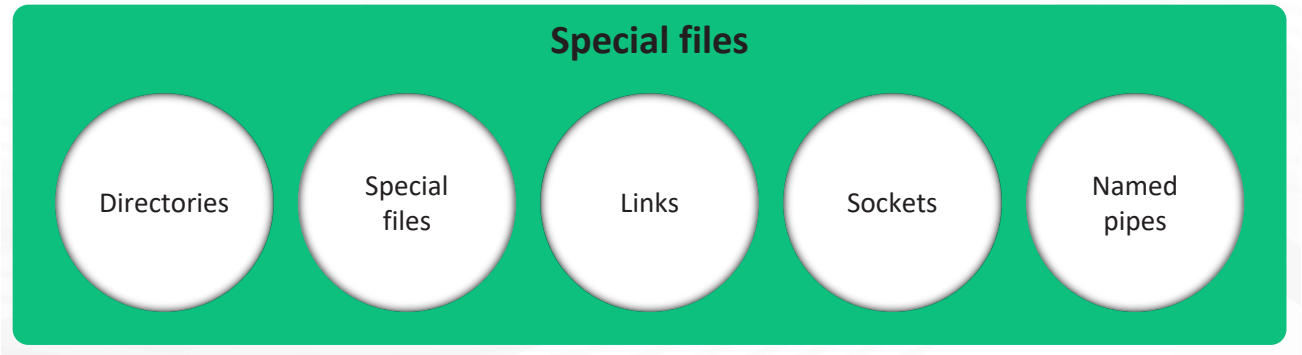
the information describing the filesystem stored on the drive or partition but is separate from and independent of the filesystem metadata.

APIs: APIs provide for tasks such as creating, moving, and deleting files. It also provides algorithms that determine things like where a file is placed on a filesystem. Such algorithms may account for objectives such as speed or minimizing disk fragmentation.

Security model: The Linux filesystem security model helps to ensure that the users only have access to their own files and not those of others or the operating system itself.

1.4.1. Files

- The basic unit of Data Storage which is stored on a physical storage medium such as disk. The OS identifies files by their names.
- Most files are just files containing normal data, i.e., text/images, executable files, input for/output of a program, etc.
- Some of the special files are given in the diagram below.



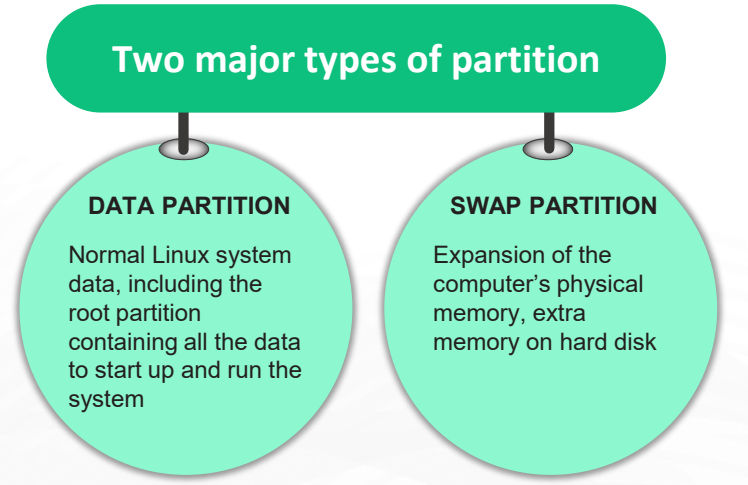
You will be learning about files system in this slide.

- Linux files are the basic unit of data storage, and is stored on a physical storage medium such as disk. The OS identifies files by their names.
- Most files are just files containing normal data, i.e., text/images, executable files, input for/output of a program, etc.
- **Regular files:** denoted by '-'.
- **Directories:** files that are lists of other files. Denoted by 'd'.
- **Special files:** the mechanism used for input and output. Denoted by 'c'.
- **Links:** a system to make a file or directory visible in multiple parts of the system's file tree. Denoted by 'l'.
- **(Domain) sockets:** a special file type, similar to TCP/IP sockets, providing inter-process networking protected by the file system's access control. Denoted by 's'.

- **Named pipes:** act more or less like sockets and form a way for processes to communicate with each other, without using network socket semantics. Denoted by ‘p’.

1.4.2. Partition

- Meant for achieving data security during a disaster. Data will be grouped and separated in each hard disk partition, data in the affected partition will only be lost during the disaster.
- All the partitions are attached to the system by a Mount Point, which defines the location of a particular data set in a file system.



The Instructor will explain the following content about ‘Partition’:

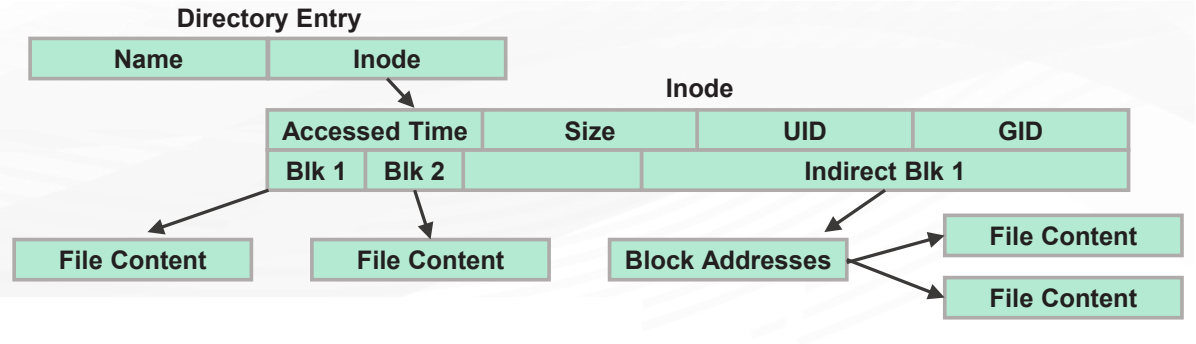
- Simple example for the need of partitioning: a user creates a script, a program or a web application that starts filling up the disk. If the disk contains only one big partition, the entire system will stop functioning if the disk is full. If the user stores the data on a separate partition, then only that (data) partition will be affected, while the system partitions and possible other data partitions keep functioning.
- Most of the Linux systems contain a root partition, one or more data partitions and swap partitions.
- The standard root partition is about 100-500 MB, has the system configuration files, most basic commands and server programs, system libraries, some temporary space and the home directory of the administrative user. A standard installation requires about 250 MB for the root partition.
- Swap space is only accessible for the system itself, and is hidden from view during normal operation. Swap ensures, like on normal UNIX systems, that you can keep on working, whatever happens.
- The kernel is on a separate partition as well in many distributions, because it is the most important file of the system. /boot partition, holds the kernel(s) and accompanying data files.

The rest of the hard disk(s) is generally divided in data partitions. All of the non-system critical data resides on one partition, for example while performing a standard workstation installation. When non-critical data is separated on different partitions, it usually happens following a set pattern:

- a partition for user programs (/usr)
- a partition containing the users' personal data (/home)
- a partition to store temporary data like print- and mail-queues (/var)
- a partition for third party and extra software (/opt)

1.4.3. File System - Important Concepts

- Every partition has its own file system.
- Each file is represented by an i-node: a number containing information about the Data contained in the file, the file owner, and its location on the hard disk.
- Every partition has its own set of i-nodes; throughout a system with multiple partitions, files with the same i-node number can exist.
- At the time of hard disk initialization, a fixed number of i-nodes per partition is created, which is the maximum amount of files, of all types (including directories, special files, links etc.) that can exist at the same time on the partition.
- Usual number - 1, i-node per 2 to 8 KB of storage.



The instructor will explain some important concepts of the file system. These are:

Every partition has its own file system.

- Each file is represented by an i-node: a number containing information about the data contained in the file, file owner, and its location on the hard disk.
- Every partition has its own set of i-nodes; throughout a system with multiple partitions, files with the same i-node number can exist.
- At the time of hard disk initialization, a fixed number of i-nodes per partition is created, which is the maximum amount of files, of all types (including directories, special files, links etc.) that can exist at the same time on the partition.

- Usual number - 1 i-node per 2 to 8 KB of storage.

When a new file is created, it gets a free i-node. In that inode is the following information:

- Owner and group owner of the file.
- File type (regular, directory)
- Permissions on the file
- Date and time of creation, last read and change.
- Date and time this information has been changed in the i-node
- Number of links to this file (see later in this chapter)
- File size
- An address defining the actual location of the file data.

The only information not included in an i-node, is the file name and directory. These are stored in the special directory files. By comparing file names and i-node numbers, the system can make up a tree-structure that the user understands.

1.5 Types of Linux File Systems

Most common types of Linux file systems:

You will learn about the most common types of the Linux file system in this slide. The following are most common types of file system in Linux.

- ext2, ext3, ext4

  The Progressive versions of the Extended File System (ext), primarily was developed for MINIX. The second extended version (ext2) was an improved version. Ext3 added performance improvement. Ext4 was a performance improvement besides additional providing additional features.

- JFS

  The Journaled File System (JFS) was developed by IBM for AIX UNIX which was used as an alternative to system ext. Currently, JFS is used as an alternative to ext4, where stability is required with the use of very few resources, especially when CPU power is limited.

- ReiserFS
  ReiserFS has improved performance and advanced features and was introduced as an alternative to ext3. ReiserFS supports file System Extension dynamically which was relatively an advanced feature but the file system lacked certain area of performance.
- XFS
  XFS was a high-speed JFS meant for parallel I/O processing. NASA uses this file system for their 300+ TB storage server.
- Btrfs
  B-Tree File System (Btrfs) works well for fault tolerance, fun administration, repair system, large storage configuration.

1.6 Common System Directories

The following table describes the common system directories.

Directory	Description
/	First directory in Linux File System. It is also known as root directory or main directory. All files and directories are created and managed under this directory.
/home	Default directory for user data. Whenever we add a new user, Linux automatically creates a home directory matching with the username in this directory. Whenever user login, Linux starts the login session from home directory.
/lib	Shared library files that are required to boot the system.
/root	Home directory for root user. Root user is the super user in Linux. For security reason Linux creates a separate home directory for root user. Root user account is also being created during the installation automatically.
/bin	Standard command files. Commands stored in this directory are available for all users and usually do not require any special permission to run.
/sbin	System administration commands files. Commands stored in this directory are available only for root user and usually requires special privilege to run.

Let us look at the description of the directories listed here:

- / - The First directory in Linux File System. It is also known as root directory or main directory. All files and directories are created and managed under this directory.
- /home - Default directory for the user data. Whenever we add a new user, Linux automatically creates a home directory matching with the username in this directory. Whenever users login, Linux starts the login session from home directory.
- /lib - Shared library files that are required to boot the system.

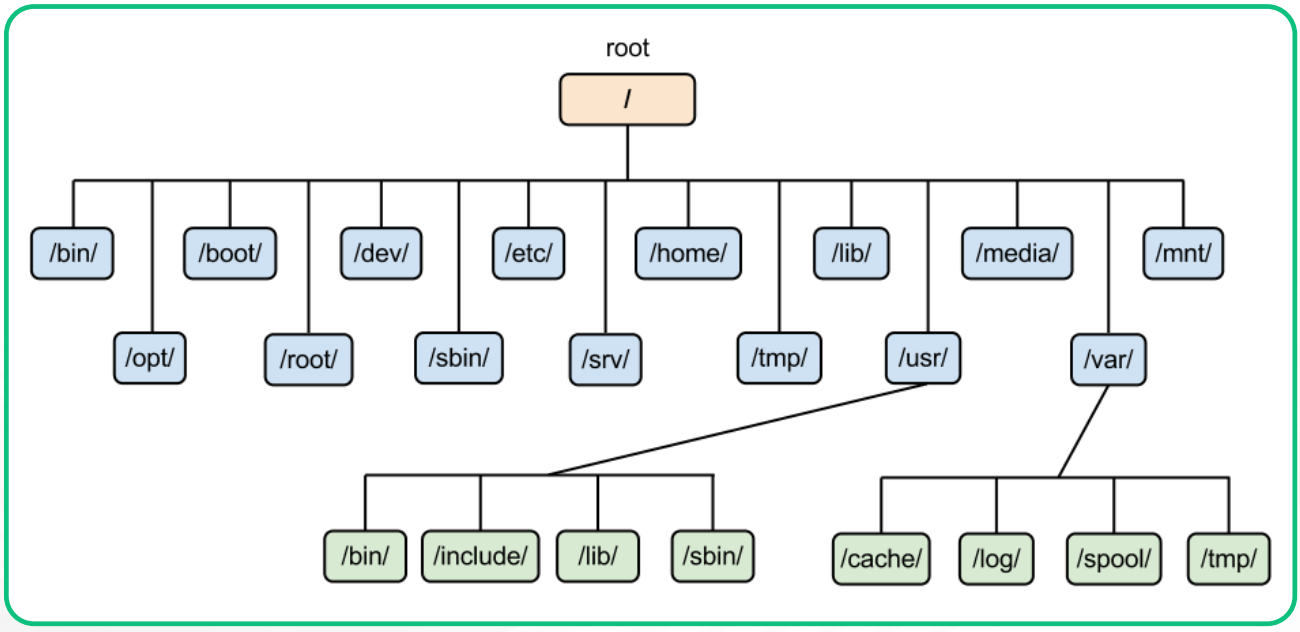
- /root - Home directory for the root user. The Root user is the Super user in Linux. For security reason, Linux creates a separate home directory for the Root user. The Root user account is also created during the installation automatically.
- /bin - Standard command files. The Commands stored in this directory are available for all users and usually do not require any special permission to run.
- /sbin - The System administration commands files. The Commands stored in this directory are available only to the root user and usually require special privilege to run.

Directory	Description
/usr	User application software files, third party software and scripts, document files and libraries for programming languages.
/var	Variable data files such as printing jobs, mail box etc.
/etc	System configuration files.
/boot	Linux boot loader file.
/mnt	To mount remote file system and temporary devices such as CD, DVD and USB.
/dev	Device files. Usually files in this directory are dynamically generated and should be never edited.
/tmp	Provides temporary location for applications.

Let us look at the description of the directories listed here:

- /usr - User application software files, third-party software and scripts, document files and libraries for programming languages.
- /var - Variable data files such as printing jobs, mailbox etc.
- /etc - System configuration files.
- /boot - Linux boot loader file.
- /mnt - To mount the remote file system and temporary devices such as CD, DVD and USB.
- /dev - Device files. Usually, files in this directory are dynamically generated and should be never edited.
- /tmp - Provides a temporary location for applications.

1.7 Linux System Directories - In a Nutshell



Think of the various directories that you saw in the previous slide.

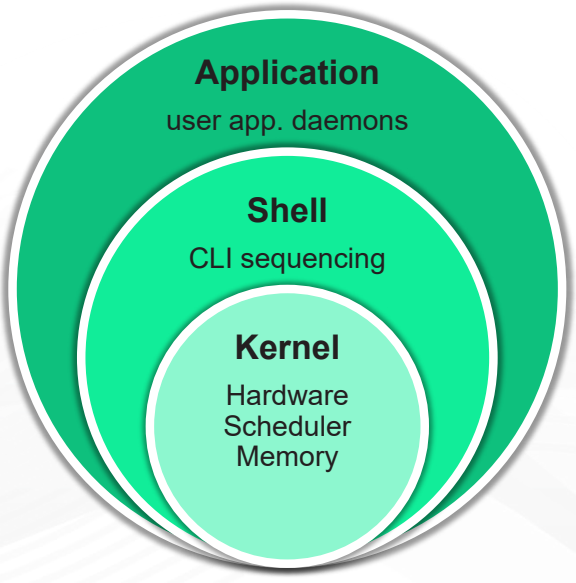
1.8 The Kernel

Kernel:

- The most important component of the system, that manages the communication between the underlying hardware and the peripherals.
- Is important for starting and stopping the processes and daemons at the right times.

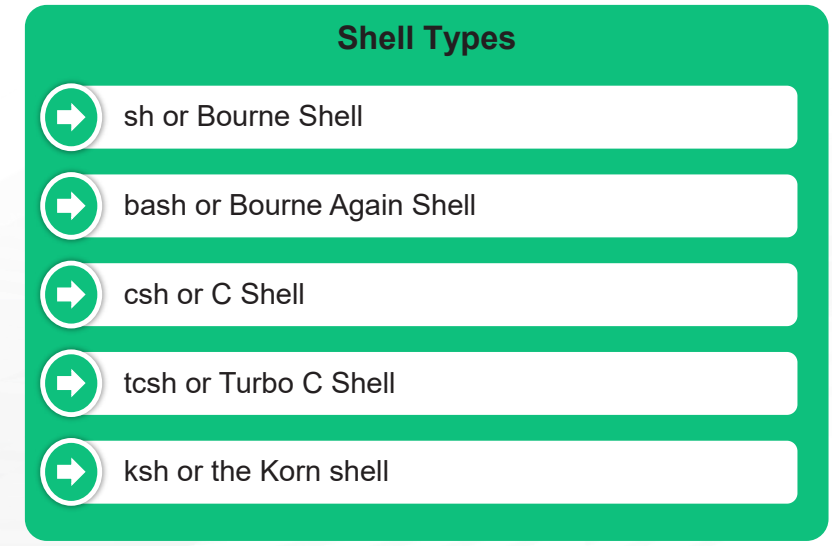
The instructor will explain the Kernel.

The kernel is a most important component of the system that manages the communication between the underlying hardware and the peripherals. It is important for starting and stopping the processes and daemons at the right times.



1.9 Shell

Shell manages the interactions between the user and the system .Allows a two-way conversation and is an advanced way of communicating with the system.



You will learn about the next important component of Unix system that is 'Shell'.

- Shell manages the interactions between the user and the system.
- Allows two-way conversation and is an advanced way of communicating with the system.

There are five types of Shells in Unix system. These are:

- **sh or Bourne Shell:** the original shell still used on UNIX systems and in UNIX related environments. This is the basic shell, a small program with few features. When in POSIX-compatible mode, bash will emulate this shell.
- **bash or Bourne Again Shell:** the standard GNU shell, intuitive and flexible. Probably most advisable for beginning users while being at the same time a powerful tool for the advanced and professional user. On Linux, bash is the standard shell for common users. This shell is a so-called superset of the Bourne shell, a set of add-ons and plug-ins. This means that the Bourne Again Shell is compatible with the Bourne shell: commands that work in sh, also work in bash. However, the reverse is not always the case. All examples and exercises in this book use bash.
- **csh or C Shell:** the syntax of this shell resembles that of the C programming language. Sometimes asked for by programmers.
- **tcsh or Turbo C Shell:** a superset of the common C Shell, enhancing user-friendliness and speed.
- **ksh or the Korn shell:** sometimes appreciated by people with a UNIX background. A superset of the Bourne shell; with standard configuration a nightmare for beginning users.

What did you Grasp?

1. Which of the following statement(s) is/are correct in respect to Linux file system?

   A) The File system provides space for non-volatile storage of data.
   B) All the partitions are attached to the system by a mount point, which defines the location of a particular data set in a file system.
   C) Every partition has its own file system.
   D) All of the above.

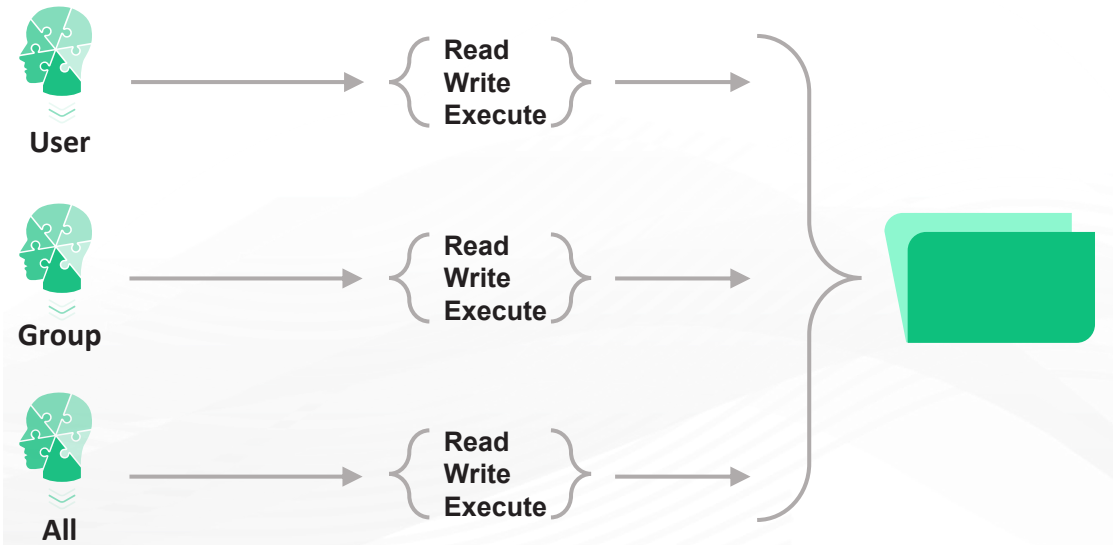
2. /etc directory is used for storing?

   A) System files
   B) User files
   C) Configuration files
   D) Temporary files

3. Which of the following options describes an inode?

   A) A number containing meta information about the file.
   B) A system directory used to store log files.
   C) A type of file system.
   D) A shell command.

2. User Groups and Permissions



You will learn about User Groups and Permissions in this topic.

2.1 User Accounts

- Each user in the system has a personal account with a separate username and password.
- There is an ‘owner’ for each and every file, who can set file access permissions.
- ‘Root’ account is needed for system maintenance, which can be accessed by ‘switch user’ (‘su’) command.
- Other accounts in the system are used by system daemons, which access system files using a specific user ID other than the root or any of the personal accounts.
- The System administrator creates and manages accounts for all real and virtual users in the system.

Instructor will explain the following points about user accounts:

- The User accounts are important to maintain identification and system security.
- The User account is meant for distinguishing between different people who use the system. Each user has a personal account with a separate username and password.
- Each and every file is ‘owned’ by a particular user, who can set access permissions on their files; user accounts are used in user authentication.
- Accounts are helpful in user identification, maintaining system logs, tagging email messages with user names, etc.
- A system administrator uses ‘root’ account to perform maintenance, which can be accessed by ‘switch user’ (‘su’) command.
- Other accounts in the system are used by system daemons, they access the System files using a specific user ID other than the root or any of the personal accounts.
- The System administrator creates and manages accounts for all the real and virtual users in the system.
- Each user will have a home directory usually located at /home/USERNAME
- Generally other programs will store and read the user’s preferences from the configuration file located in the user’s home directory.Example: .bashrc, .vimrc, etc.
- A user can be a member of one or more groups at the same time.
- A mapping of user’s id, their home directory and their groups’ id will be maintained by the Linux operating system
- Linux can also integrated with the other identity providing the mechanisms like LDAP, SAML, etc.

By default, Linux operating systems provide a directory located at /home directory for all the actual users of the system. The virtual users may or may not have a home directory. And Usually, the user's home directory will be named with the user's name itself.

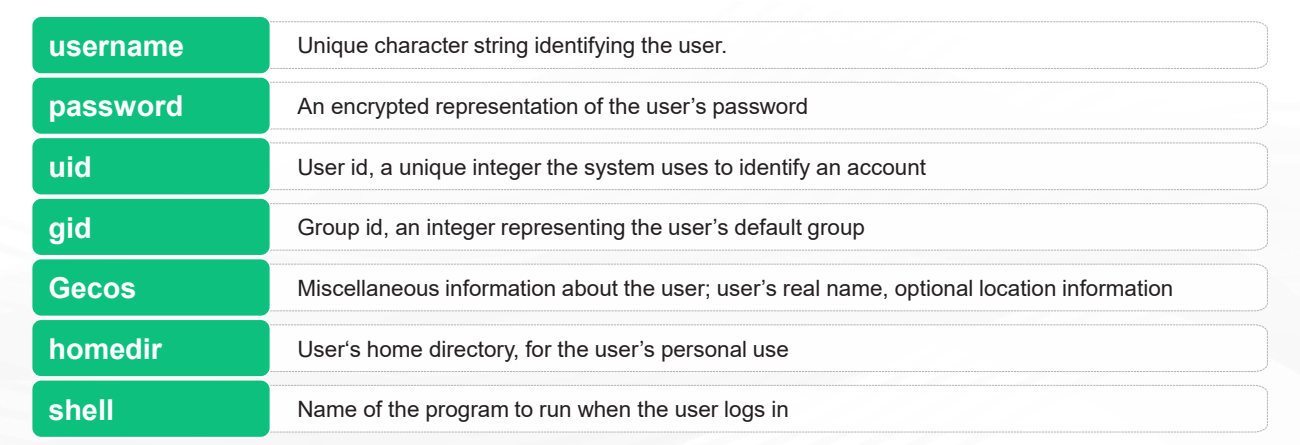
A root user doesn't have its home directory under the /home, Instead, it is located at /root. The root user has the highest privilege among all the users of the system.

Other users (users under sudoers file) can be a root user by using the `su` command and do the system management tasks. It is not at all recommended to do daily tasks as a root user unless it is required to do so.

LDAP (Lightweight Active Directory) and SAML(Security Assertion Markup Language) are different protocols used for basic authentication of the users into the system.

2.2 The passwd File

- Every user account has an entry in the /etc/passwd file, which has the attributes for the account like username, real name, etc.
- Each entry in this file is of the following format:



- Of these, username, uid, gid and homedir are the required fields and others are optional.

The instructor will explain about the passwd file in the Linux system:

- Every user account has an entry in the /etc/passwd file, which has the attributes for the account like username, real name, etc.

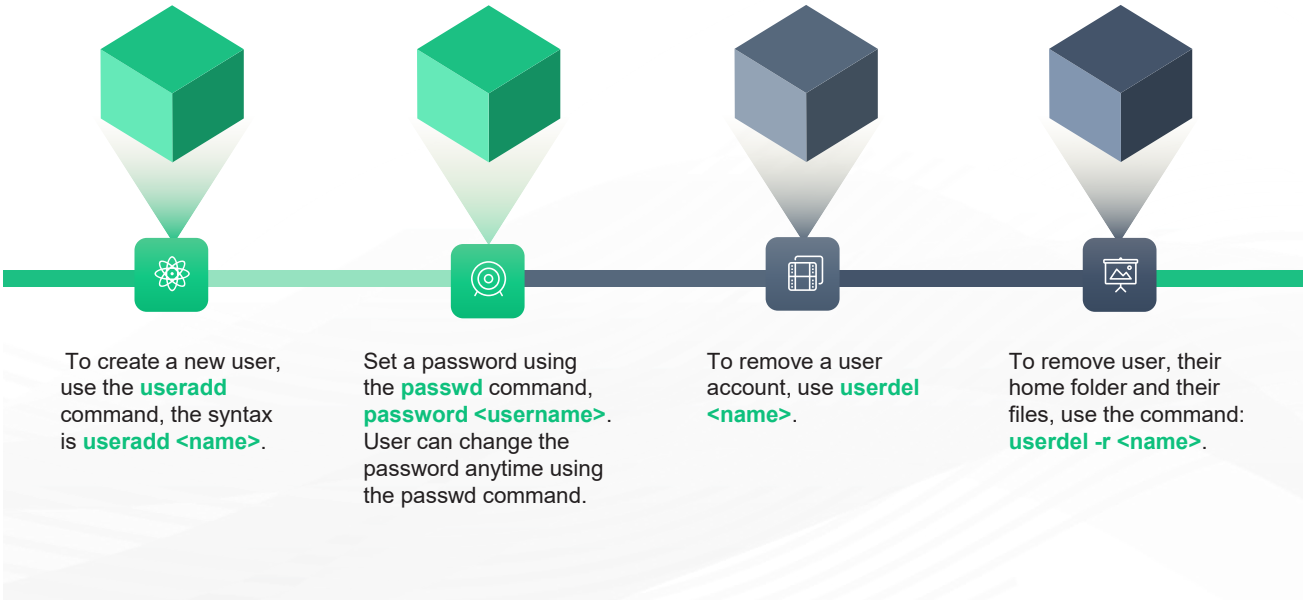
Each entry in this file is of the following format:

- ***username*** - For personal accounts, username is the name the user logs in with. On most systems it is limited to eight alphanumeric characters.
- ***password*** - Account password is set using the passwd program, which uses a one-way encryption scheme that is difficult (but not impossible) to break. However, that if the first character of the password field is * (an asterisk), the account is "disabled"; the system will not allow logins as this user.

- ***uid*** - The system uses the uid field internally when dealing with process and file permissions; both the user ID and the username identify a particular account: the user ID is more important to the system, whereas the username is more convenient for humans.
- ***gid*** - Group ID is found in the file /etc/group.
- ***gecos*** - programs like mail and finger use this information to identify users on the system; gecos stands for General Electric Comprehensive Operating System. This field was originally added to /etc/passwd to provide compatibility with some of its services.
- ***homedir*** - When the user first logs in, the shell finds its current working directory in the named home directory.
- ***shell*** - in most cases, this is the full pathname of a shell, such as /bin/bash or /bin/tcsh.

Of these username, uid, gid and homedir are the required fields and others are optional.

2.3 Creating User Accounts

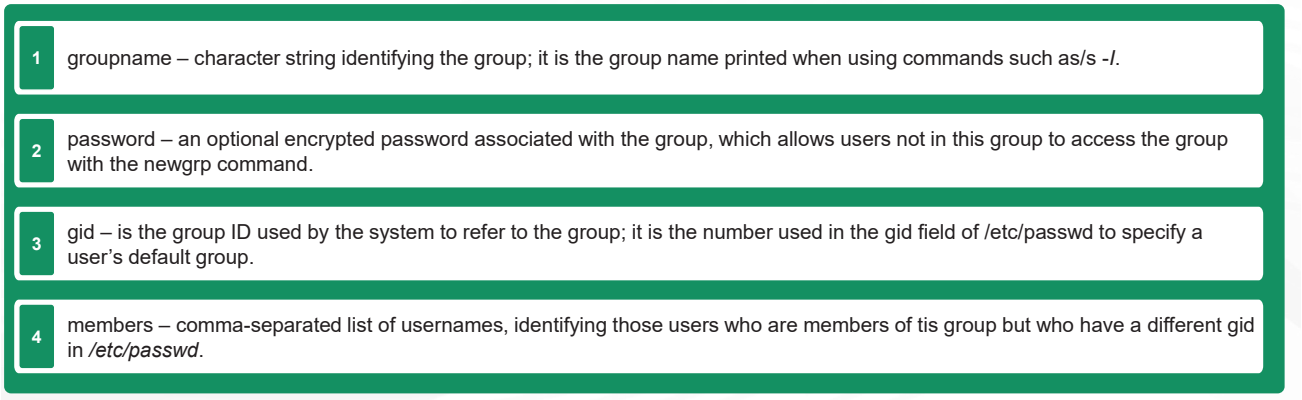


You will learn about the syntax of creating a user account, setting of password and removing a user account.

- To create a new user, use the useradd command, the syntax is useradd <name>.
- Set a password using the passwd command, password <username>. User can change the password anytime using the passwd command.
- To remove a user account, use userdel <name>.
- To remove user, their home folder and their files, use the command: userdel -r <name>.

2.4 Groups

- Groups are a way to organize user accounts logically and allow users to share files within their group or groups.
- Each and every file has both a user and group owner.
- Like /etc/passwd, the file /etc/group has a one-line entry for each group in the system.
- The format is groupname:password:gid:members.



You will be learning about the Groups on this topic/slide.

In Linux, a group is a logical grouping of the users to manage and delegate permissions to them collectively. A user can be a member of multiple groups and a group can have multiple different users at the same time.

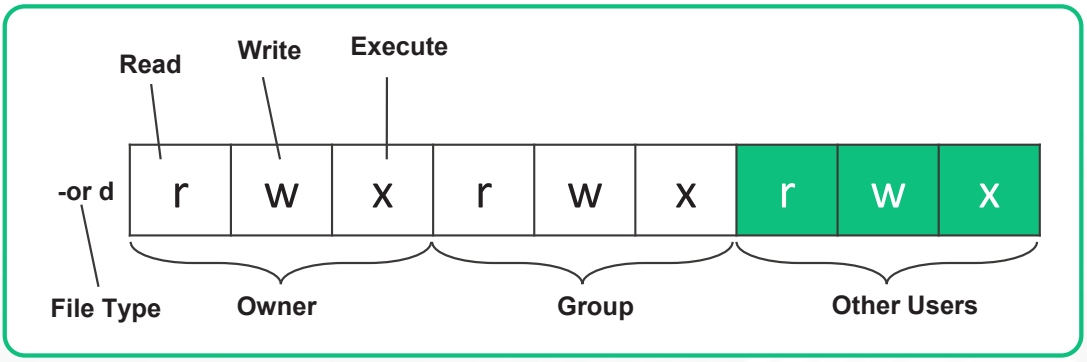
All the group entries can be found on the /etc/group file. Each line of the file have group's name, a group password to restrict open access to the users, a group id and the members of the group as a comma separated list.

- Groups are a way to organize user accounts logically and allow users to share files within their group or groups.
- Each and every file has both a user and group owner.
- Like */etc/passwd,* the file */etc/group* has a one-line entry for each group in the system.
- The format is *groupname:password:gid: members*.
    - **groupname** - character string identifying the group; it is the group name printed when using commands such as ls -l.
    - **password** - an optional encrypted password associated with the group, which allows users not in this group to access the group with the newgrp command.
    - **gid** - is the group ID used by the system to refer to the group; it is the number used in the gid field of /etc/passwd to specify a user's default group.
    - **members** - comma-separated list of usernames, identifying those users who are members of this group but who have a different gid in /etc/passwd.

What did you Grasp?

1. Which of the following is/are correct with respect to Linux user administration?
   - **A) System administrator uses personal account to perform maintenance**
   - **B) 'useradd' command is used to create a new user**
   - **C) 'passwd' program uses a one-way encryption scheme to secure passwords**
   - **D) Shell is the required field in passwd file**

3. File Permissions



You will now learn about the next topic in this module. This topic discusses the 'File Permission in Linux'.

3.1 File Ownership

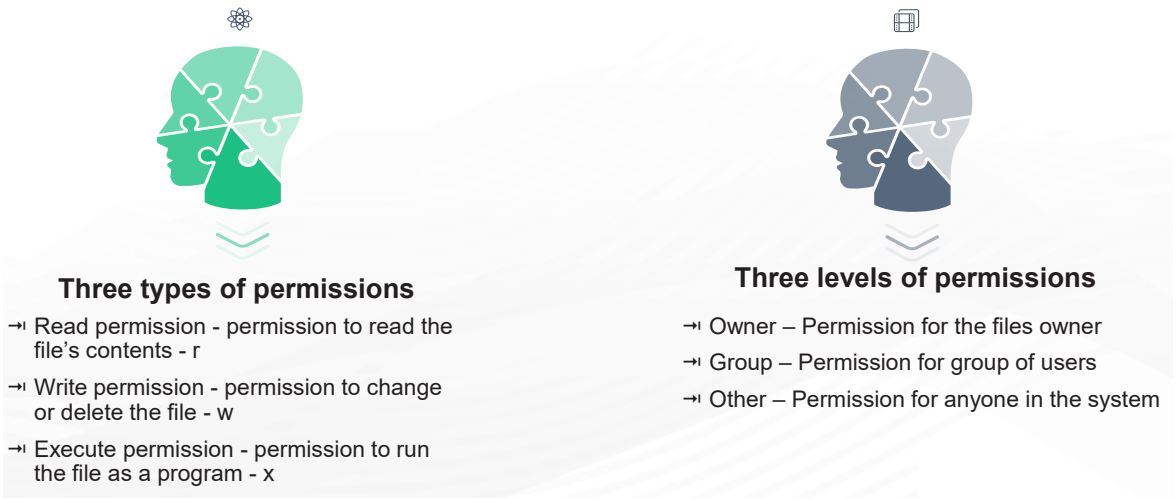
- Each and every file belongs to a user and a group.
- The file's owner and any users under the file's group can change the file's permission even if the file has no permissions.
- The file's user and the group can be simply found by long listing the file or the directory in which the file is located.Example: ls –l will return

```
-rw-rw-r-- 1 user group 150 Mar 19 08:08 somefile.txt
```

All the files in Linux will be owned by a user and a group. The file owners will have complete control to modify the file's permission though they can't read the file's content due to the existing permission of the file.

3.2 File Permissions

In a Multi-user environment, many users run programs and share data. The file permissions are the ones that protect users, their data and the core system files.



File permissions are nothing but the ways in which someone could use a file. Permissions can be viewed by using the "long list" option (-l) with ls.

Three types of permissions:

When each file is created, some default permissions are assigned by the system. For example, both read and write permissions are assigned to the owner, but others will get only read permissions. There are three levels of permissions read, write and execute.

- Read (r) refers to the permission to view the file's contents.
- Write (w) refers to the permission to add or change the content or delete the file.
- Execute (e) refers to the permission to the run the file as a program.

Apart from the default permissions, permissions can be assigned or modified. When an executable program is created by a compiler, execute permission is automatically assigned. Permissions are not assigned by default in some cases. When a shell script or a Perl program is created, we need to assign execute permission ourselves in order to run it.

In case of a Directory, permissions have a different meaning.

- Read permission means we can list the contents of the directory.

- Write permission means we can add or remove files in that directory.
- Execute permission means we can list information about the files in that directory.

Three levels of permissions

Like there are types, there are different levels of permissions: Owner, Group, Other

The owner is the user one who creates the file. Each file has an owner and group. Each user may also belong to a default group, which is assigned to the file the user creates. The reason for having groups is to give a set of users certain permissions.

For example, there's a set of programmers working on a project. The one who created the source code can retain the write permission to himself, and read permission can be assigned to other programmers in the group. By changing the groups assigned to a file, permissions can be given or denied to any set of users.

Changing File Permissions

Use the chmod command to change permissions on a file . This command changes the file permission bits of each file according to a given mode, which can be either a symbolic representation of changes to be made or an octal number representing the bit pattern for the new mode bits.

Chmod in symbolic mode

The syntax of the command in symbolic mode is chmod [references][operator][modes]: references can be "u" for user, "g" for group, "o" for others and "a" for all three types. The operator can be "+" to add and "-" to remove permissions. In the following example, the owner has been given read, write, and execute permissions, the group and everybody else has no permission.

Permissions are arranged as three sets of three characters each:

- Set 1 - User/Owner
- Set 2- Group
- Set 3 - Others/everyone else

User (Owner)	Group	Others
rw-	r--	r--

Each permission can be denoted in octal digits:

- Read – Octal number Four (4)
- Write – Octal number Two (2)
- Execute – Octal number Two (1)

File permission can be set using the above numbers:

- Ex: chmod 754 somefile.txt
- 7 – (4 + 2 + 1) – file's owner has Read, Write and Execute permissions

- 5 – (4 + 0 + 1) – users of file's group have Read and Execute permissions
- 4 – (4 + 0 + 0) – All other users have Read permission to the file

You will be learning about the third level of permissions, that is:

Chmod in numeric mode

The numeric mode is from one to four octal digits (0-7). The rightmost digit selects permissions for the world, the second digit for other users in the group and the third digit for the owner. The fourth digit is rarely used. The value for each digit is derived by adding up the bits with values 4 (rea- only), 2 (write only), and 1 (execute only). The value zero removes all permission for the particular group, whereas the value 7 turns on all permissions (read, write, and execute) for that group.

What did you Grasp?

1. The third digit from right in chmod numeric mode assigns permissions to the world.

   A) True
   B) False

2. The file permission -rwxr--r-x represented in octal expression will be?

   A) 775
   B) 664
   C) 745
   D) 466

3. When a user tries to remove a read-only file, Which of the following will happen?

   A) The rm command prompts for a confirmation, however the operation fails because of insufficient permissions
   B) The rm command prompts for a confirmation, the command is successful upon confirmation
   C) The file is removed successfully (and silently)
   D) The rm command fails because of insufficient permissions

4. Working with Bash

You will learn about Working with Bash in this topic.

4.1 Introduction

- Bash - Bourne Again Shell, command language interpreter, for the GNU operating system, is an sh-compatible shell that incorporates useful features from the Korn shell (ksh) and C shell (csh).
- Bash offers functional improvements over sh for both programming and interactive use. In addition, most sh scripts can be run by Bash without modification.
- Bash is basically a command processor that typically runs in a text window, allowing the user to type commands that cause actions. It can read commands from a file, called a shell script or shell program. Bash supports:
    - File name wildcarding
    - Piping
    - Hear documents
    - Command execution
    - Variables and control structures for condition testing and iteration

The instructor will introduce you to the Bash in Linux in this slide.

- Bash is Bourne Again Shell, command language interpreter, for the GNU operating system, is an sh-compatible shell that incorporates useful features from the Korn shell (ksh) and C shell (csh).
- Bash offers functional improvements over ‘sh’ for both programming and interactive use. In addition, most sh scripts can be run by Bash without modification.
- Bash is basically a command processor that typically runs in a text window, allowing the user to type commands that cause actions. It can read commands from a file, called a script. Bash supports:
    - File name wildcarding
    - Piping

- Hear documents
- Command execution
- Variables and control structures for condition testing and iteration

4.2 Shell Features

The following table explain the Shell features.

Shell Syntax	What your input means to the shell
Shell Commands	The types of commands you can use
Shell Functions	Grouping commands by name
Shell Parameters	How the shell stores values
Shell Expansions	How Bash expands parameters and the various expansions available
Redirections	A way to control where input and output go.
Executing Commands	What happens when you run a command.
Shell Scripts	Executing files of shell commands

- Make note of the different Shell features.
- Understand what each shell feature means.

4.3 Basic Bash Commands

The following table explains the Bash Commands.

1	echo – sends text to stdout
2	ls – lists files and folders in current directory
3	pwd – prints current working directory
4	cd – change directory
5	cat – display whole content of a file
6	head, tail – selectively display the given input
7	vim, nano – edit the file's content
8	cp, mv & rm – copy, move and remove the file
9	clear – clears the terminal
10	mkdir – creates new directory
11	grep & find – search files and folders

You will be learning about Bash Commands in this slide.

- echo – sends text to stdout
- ls – lists files and folders in current directory
- pwd – prints current working directory
- cd – change directory
- cat – display whole content of a file
- head, tail – selectively display the given input
- vim, nano – edit the file's content
- cp, mv & rm – copy, move and remove the file
- clear – clears the terminal
- mkdir – creates new directory
- grep & find – search files and folders

4.4 Bash Operators

The following table explains the Bash Operators..

< or > - Used to redirect files content to a program or redirect stdout and/or stderr to a file.
| - Used to send stdout of a program to the stdin of another program.
& - Runs program in the background.
*, ?, [ ] – Used to match characters in filenames.
( ) – Used to run commands in a subshell.

You will be learning about the Bash Operators in this slide.

The operators available in Bash are:

- **<** or **>** - Used to redirect files content to a program or redirect stdout and/or stderr to a file
- **|** - Used to send stdout of a program to the stdin of another program
- **&** - Runs program in the background
- *****, **?**, **[ ]** – Used to match characters in filenames
- **( )** – Used to run commands in a Subshell

What did you Grasp?

1. Which of the following about bash is true?

   A) Bash supports most of the shell scripts without modification
   B) Shell syntax is the way of storing values
   C) Redirection is the way of giving inputs to the shell
   D) All the above

2. What is the use of the rmdir command?

   A) Removes the specified file.
   B) Removes the specified directory.
   C) Renames the specified directory.
   D) Reruns the previous command.

3. What does the command echo 'abc' >> myfile.txt do?

   A) Replaces the content of the file with the text 'abc'.
   B) Removes the text 'abc' if it exists in the file.
   C) Appends the text 'abc' to the file.
   D) Prints the content of the file with the prefix 'abc

In a nutshell, we learnt:

1. Everything in Linux is a file and is stored somewhere in the root or mounted partition formatted with a specific file system.
2. All the files will have an inode that stores the metadata such as name, size, type, location, timestamp, permissions & ownership of the file.
3. In Linux, a file can belong to a user and the users under a group. Read, Write and Execute permissions are assigned on three levels i.e. user, group and others.
4. Bash is the most commonly used shell among the modern Linux Distributions that provides interface for running shell commands as well as scripting.
5. The I/O in the terminal is provided by stdin, stdout and stderr. This I/O can be manipulated by Redirection or Piping operators in Bash.

Release Notes

B. TECH CSE with Specialization in DevOps

Semester three -Year 02

Release Components.
Facilitator Guide, Facilitator Course Presentations, Student Guide, Mock exams and relevant lab guide.

Current Release Version.
1.0.0

Current Release Date.
2 July 2018

Course Description.
Xebia, has been recognized as a leader in DevOps by Gartner and Forrester and this course is created by Xebia to equip students with set of practices, methodologies and tools that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

The language used in this course is US English. Our sources of reference for grammar, syntax, and mechanics are from The Chicago Manual of Style, The American Heritage Dictionary, and the Microsoft Manual of Style for Technical Publications.

Bugs reported	Not applicable for version 1.0.0
Next planned release	Version 2.0.0 Feb 2019