Linux Training

PREPARED BY ENES ERTEN

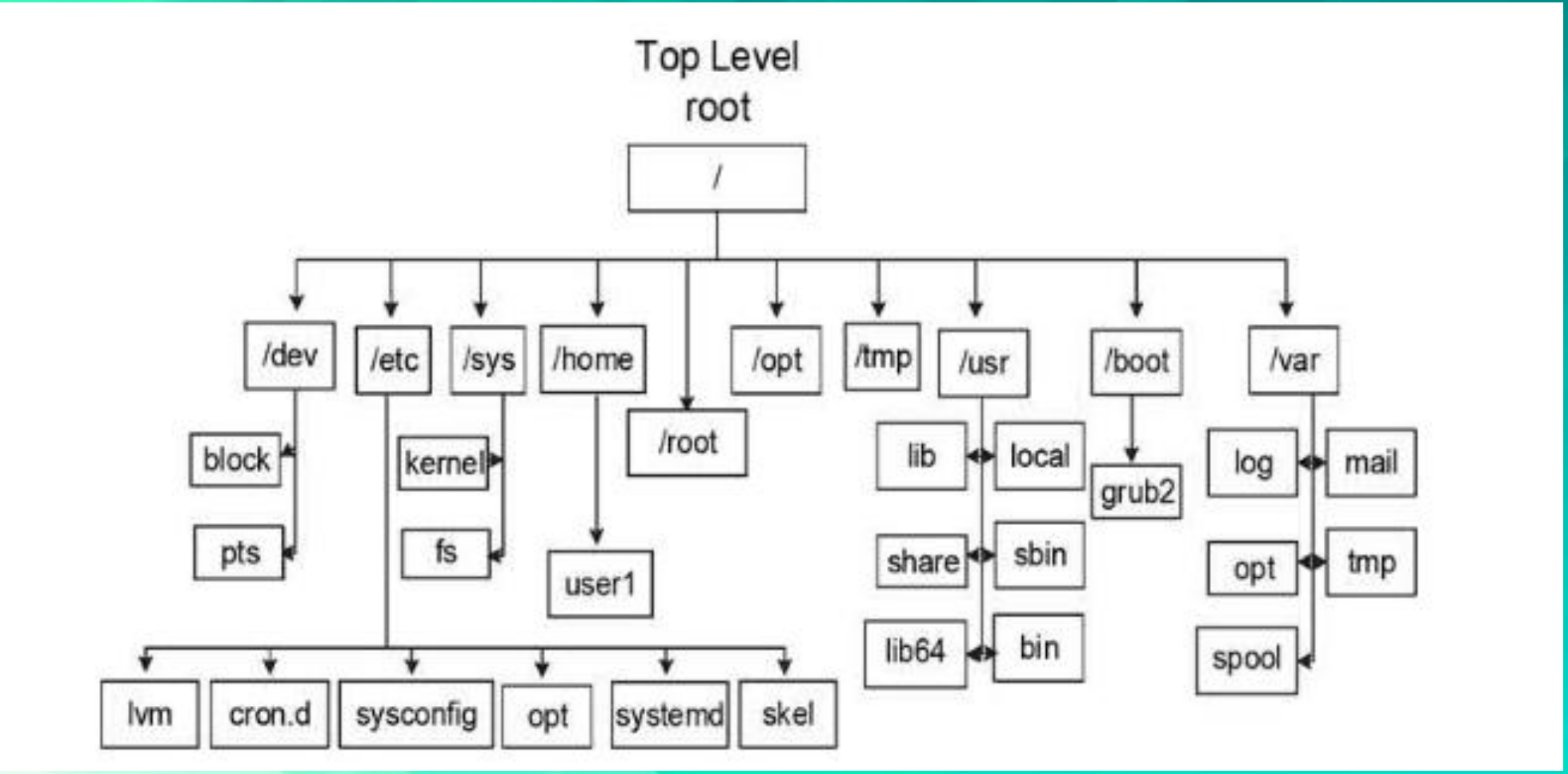
Contents

Linux Directory Structure and File Systems

Linux files are organized logically in a hierarchy for ease of administration and recognition. This organization is maintained in hundreds of directories located in larger containers called file systems.

There is a standard called Filesystem Hierarchy Standard (FHS) for file organization, which describes names, locations, and permissions for many file types and directories.

Linux Directory Structure and File Systems



Linux Directory Structure and File Systems

Linux file systems contain files and subdirectories. A subdirectory, also referred to as a child directory, is located under a parent directory. The parent directory is a subdirectory of a higher-level directory. The Linux directory structure is analogous to an inverted tree, where the top of the tree is the root of the directory, tree branches are subdirectories, and leaves are files. The root of the directory is represented by the forward slash character (/), and this is where the entire directory structure is ultimately connected.

Linux Directory Structure and File Systems

There are variety of file system types. These are can be categorized in three basic groups: disk-based, network-based, and memory-based. Disk-based file systems are typically created on physical media such as a hard drive or a USB flash drive. Network-based file systems are essentially disk-based file systems that are shared over the network for remote access. Memory-based file systems are virtual; they are created automatically at system startup and destroyed when the system goes down. The first two types of file systems store information persistently, while any data saved in virtual file systems is lost at system reboots.

Linux Directory Structure and File Systems

The Root File System (/), Disk-Based
The root directory is the top-level file system in the FHS

/etc: The etcetera (or extended text configuration) directory holds system configuration files. Some common subdirectories are systemd, sysconfig, lvm, and skel, which comprise configuration files for systemd, most system services, the Logical Volume Manager, and per-user shell startup template files, respectively.

/root: This is the default home directory location for the root user.

/**mnt:** This directory is used to mount a file system temporarily.

Linux Directory Structure and File Systems

The Boot File System (/boot) – Disk-Based
The /boot file system contains the Linux kernel, boot support files, and boot configuration files. Just like the root file system,
the size of this file system is also automatically determined by
the installer program based on the available disk space when you
select the default partitioning; however, it may be set to a different size during or after the installation if required.

Linux Directory Structure and File Systems

The Home Directory (/home)

The /home directory is designed to store user home directories and other user contents. Each user is assigned a home directoryto save personal files, and the user can block access to other users.

Linux Directory Structure and File Systems

The Optional Directory (/opt)
This directory can be used to hold additional software that may need to be installed on the system. A subdirectory is created for each installed software.

Linux Directory Structure and File Systems

The UNIX System Resources Directory (/usr)
This directory contains most of the system files.

/usr/bin: The binary directory contains crucial user executable commands.

/usr/sbin: Most commands are required at system boot, and those that require the root user privileges in order to run are located in this system binary directory. In other words, this directory contains crucial system administration commands that are not intended for execution by normal users (although they can still run a few of them). This directory is not included in the default search path for normal users because of the nature of data it holds.

Linux Directory Structure and File Systems

/usr/lib and /usr/lib64: The library directories contain shared library routines required by many commands and programs located in the /usr/bin and /usr/sbin directories, as well as by the kernel and other applications and programs for their successful installation and operation. The /usr/lib directory also stores system initialization and service management programs. The subdirectory /usr/lib64 contains 64-bit shared library routines.

/usr/include: This directory contains header files for C language.

Linux Directory Structure and File Systems

/usr/local: This directory serves as a system administrator repository for storing commands and tools downloaded from the web, developed in-house, or obtained elsewhere. These commands and tools are not generally included with the original Linux distribution.

/usr/share: This is the directory location for manual pages, documentation, sample templates, configuration files, etc., that may be shared with other Linux platforms.

/usr/src: This directory is used to store source code

Linux Directory Structure and File Systems

The Variable Directory (/var)
The /var directory contains data that frequently changes while the system is operational. Files in this directory contain log, status, spool, lock, and other dynamic data.

/var/log: This is the storage for most system log files, such as system logs, boot logs, user logs, failed user logs, installation logs, cron logs, mail logs, etc.

/var/opt: This directory stores log, status, and other variable data files for additional software installed in /opt, t.

Linux Directory Structure and File Systems

/var/spool: Directories that hold print jobs, cron jobs, mail messages, and other queued items before being sent out to their intended destinations are located here.

/var/tmp: Large temporary files or temporary files that need to exist for longer periods of time than what is typically allowed in another temporary directory, /tmp, are stored here. These files survive system reboots and are automatically deleted if they are not accessed or modified for a period of 30 days.

Linux Directory Structure and File Systems

The Temporary Directory (/tmp)
This directory is a repository for temporary files. Many programs create temporary files here during runtime or installation. These files survive system reboots and are automatically removed if they are not accessed or modified for a period of 10 days.

Linux Directory Structure and File Systems

The Devices File System (/dev), Virtual

The Devices (dev file system) file system is accessible via the /dev directory, and it is used to store device nodes for physical hardware and virtual devices. The Linux kernel communicates with these devices through corresponding device nodes located here. These device nodes are automatically created and deleted by the udevd service (a Linux service for dynamic device management) as necessary.

Linux Directory Structure and File Systems

The Procfs File System (/proc), Virtual
The Procfs (process file system) file system is accessible via the /proc directory, and it is used to maintain information about the current state of the running kernel. This includes the details for current hardware configuration and status information on CPU, memory, disks, partitioning, file systems, networking, running processes, and so on. This information is stored in a hierarchy of subdirectories that contain thousands of zero-length pseudo files.

These files point to relevant data maintained by the kernel in the memory. This virtual directory structure simply provides an easy interface to interact with kernel-maintained information. The Procfs file system is dynamically managed by the system.

Linux Directory Structure and File Systems

The Runtime File System (/run), Virtual
This virtual file system is a repository of data for processes running on the system. One of its subdirectories, /run/media, is also used to automatically mount external file systems such as those that are on optical (CD and DVD) and flash USB.

Linux Directory Structure and File Systems

The System File System (/sys), Virtual
Information about hardware devices, drivers, and some kernel features is stored and maintained in the /sys file system. This information is used by the kernel to load necessary support for the devices, create device nodes in /dev, and configure the devices. This file system is auto-maintained as well.

Basic System Commands

Command Mechanics:

command option(s) argument(s)

Options (a.k.a. a switch of flag) are optional. You can specify zero or more options with a command. Arguments, in contrast, may be optional or mandatory depending on the command and its usage. Many commands have preconfigured default options and arguments.

Basic System Commands

- **tree command:** The tree command lists a hierarchy of directories and files. There are a number of options with this command that can be specified to include additional information.

  # tree [options] [file path]

- **ls command:** Listing files and Directories.

  # ls [options] [file path]

Basic System Commands

- **pwd command:**Printing Working Directory
  #pwd

- **cd command:** Change Directory
  # cd /filepath

- **uptime command:** display the system's current time, length of time it has been up for, number of users currently logged in, and the average CPU (processing) load over the past 1, 5, and 15 minutes.
  #uptime

Basic System Commands

- **clear command:** clear terminals
  # clear

- **which | whereis | type command:** determines the command path of argument
  # which | wheris | type [argument]

- **uname comand:** information about the OS, hardware, kernel.
  # uname [option]

- **lscpu command:** print out all about cpu information of system
  # lscpu

Getting Help

- **man command:**

  # man [command]
- Searching keyword:

  # man –k [keyword]

  # apropos [keyword]

Compression and Archiving

- gzip and gunzip

  # gzip [files]

  # gunzip [zippedfile]

- bzip2 and bunzip2

  # bzip2 [files]

  # bunzip2 [zippedfile]

- zip and unzip

  # zip [files]

  # unzip [zippedfile]

Compression and Archiving

- Tar (type archive) command:

  Compressed files by gzip format

  # tar –czf /compressedfile.tar.gz /filespath

  Compressed files by bzip format

  # tar –cjf /compressedfile.tar.bz2 /filespath

  List compressed files

  # tar –tf /compressedfile.tar.gz (bz2)

  Extract files from into current directory

  # tar –xf /compressedfile.tar.gz (bz2)

  Extract files from into specified directory

  # tar –xf /compressedfile.tar.gz (bz2) -C /filepath

File Editing

- Most commonly text editor is vi (vim) editor. Vi editor ensures use keyboard %100.
- Has three modes: command mode, input mode, last line mode.
- Common mode default mode it is used for entering vi commands.
- Input mode can be entered by pressing 'i' (insert) keyword

File Editing

Entering input mode

'i' Inserts text before the current cursor position

'I' Inserts text at the beginning of the current line

'a' Appends text after the current cursor position

'A' Appends text to the end of the current line

'o' Opens a new line below the current line

'O' Opens a new line above the current line

File Editing

Navigating

'h' Moves backward one character

'j' Moves downward one line

'k' Moves upward one line

'l' Moves forward one character

'w' Moves to the start of the next word

'b' Moves backward to the start of the preceding word

'e' Moves to the ending character of the next word

'$' Moves to the end of the current line Enter Moves to the beginning of the next line

'Ctrl+f' Scrolls down to the next page

'Ctrl+b' Scrolls up to the previous page

File Editing

Deleting

'x'	Deletes the character at the cursor position
'X'	Deletes the character before the cursor location
'dw'	Deletes the word or part of the word to the right of the cursor location
'dd'	Deletes the current line
'D'	Deletes at the cursor position to the end of the current line
':6,12d'	Deletes lines 6 through 12
':1,$d'	Deletes all lines

File Editing

Undoing and Repeating

'u' Undoes the previous command.

'U' Undoes all the changes done on the current line.

':u' Undoes the previous last line mode command.

'. (dot)' Repeats the last command run.

File Editing

Searching Text

/string	Searches forward for a string
?string	Searches backward for a string
n	Finds the next occurrence of a string
N	Finds the previous occurrence of a string

Replacing Text

':%s/old/new'	Replaces the first occurrence of old with new in a file.
':%s/old/new/g'	Replaces all occurrences of old with new in a file.

File Editing

Copying, Moving, and Pasting Text

'yl'	Yanks the current letter into buffer
'yw'	Yanks the current word into buffer
'yy'	Yanks the current line into buffer
'p'	Pastes yanked data below the current line
'P'	Pastes yanked data above the current line
':1,3co6'	Copies lines 1 through 3 and pastes them after line 6
':4,6m9'	Moves lines 4 through 6 after line 9

File Editing

Changing Text

'cl' Changes the letter at the cursor location

'cw' Changes the word (or part of the word) at the cursor location to the end of the word

'cc' Changes the entire line

'C' Changes text at the cursor position to the end of the line

'r' Replaces the character at the cursor location with the character entered following this command

'R' Overwrites or replaces the text on the current line

'J' Joins the next line with the current line

'xp' Switches the position of the character at the cursor position with the character to the right of it

'~' Changes the letter case (uppercase to lowercase, and vice versa) at the cursor location

File Editing

':w' Writes changes into the file without quitting vim

':w!' Writes changes to the file even if the file owner does not have write permission on the file

':wq' Writes changes to the file and quits vim

':wq!' Writes changes to the file and quits vim even if the file owner does not have write permission on the file

':q' Quits vim if no modifications were made

':q!' Quits vim if modifications were made, but we do not wish to save them

File and Directory Operations

- Creating File

  # touch [file]
- Creating Directory

  # mkdir [directory]
- Displaying File Content

  # cat [file]

  # less [file]

  # more [file]

  # head [file]

  # tail [file]

  # head -[number of lines] [file]

  # tail -[number of lines] [file]

File and Directory Operations

- Counting words, lines, and Characters in text

  # wc [file]

  # wc –l [file] -> line

  # wc –w [file] -> word

  # wc –c [file] -> bytes

  # wc –m [file] -> chararcters

File and Directory Operations

- Copying Files

  # cp [sourcefile] [destinationfile]

  # cp –r [sourcedirectory] [destinationdirectory]
- Moving and Renaming Files

  # mv [sourcefile] [destinationfile]
- Removing Files
- # rm [file]
- # rm –rf [files | directories]

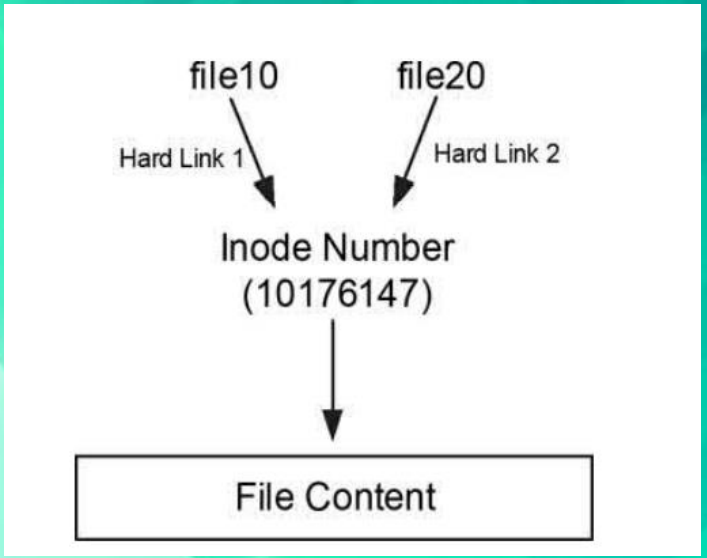
File Linking

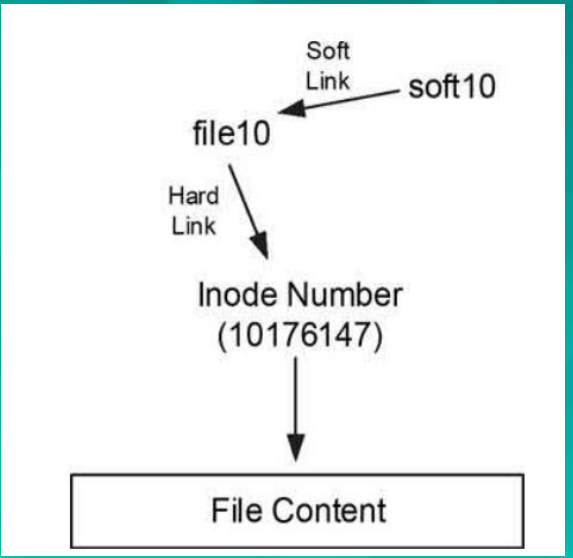
- An inode is assigned a unique numeric identifier that is used by the kernel for accessing, tracking, and managing the file. In order to access the inode and the data it points to, a filename is assigned to recognize it and access it.
- Linking files or directories creates additional instances of them, but all of them eventually point to the same physical data location in the directory tree. Linked files may or may not have identical inode numbers and metadata depending on how they are linked.

File Linking

- Hard Link:A hard link is a mapping between one or more filenames and an inode number, making all hard-linked files indistinguishable from one another. This implies that all hardlinked files will have identical metadata. Changes to the file metadata and content can be made by accessing any of the filenames.
- Soft Link (Symbolic Link): Makes it possible to associate one file with another. The concept is analogous to that of a shortcut in Microsoft Windows where the actual file is resident somewhere in the directory structure, but there can be one or more shortcuts with different names pointing to it.

File Linking





- Creating a soft link (symbolic link)

  # ln –s [referencefile] [linkedfile]

  # unlink [linkedfile]

Advanced File Management

- Permissions: set on files and directories to prevent access from unauthorized users.
- Access permissions on files and directories allow administrative control over which users (permission classes) can access them and to what level (permission types). File and directory permissions discussed in this section are referred to as standard ugo/rwx permissions.
- Users are categorized into three unique classes for maintaining file security through access rights. These classes are user (u), group (g), and other (o, also referred to as public).
- Permissions control what actions can be performed on a file or directory and by whom. There are three types of permissions bits —read (r), write (w), and execute (x)—and they behave differently for files and directories. For files, the permissions allow viewing and copying (read), modifying (write), and running (execute).

Advanced File Management 5

Octal Value	Binary Notation	Symbolic Notation	Explanation
0	000	---	No permissions
1	001	--x	Execute permission only
2	010	-w-	Write permission only
3	011	-wx	Write and execute permissions
4	100	r--	Read permission only
5	101	r-x	Read and execute permissions
6	110	rw-	Read and write permissions
7	111	rwx	Read, write, and execute permissions

Advanced File Management

- chmod command: determines the permission for the user

  # chmod [permissions in octal] file

  # chmod u+rwx … file

- chown command: determines the users

  # chown [user]:[group] file

- Umask:Assigns default permissions to a file or directory at the time of its creation. Default permissions are calculated based on the umask (user mask) permission value subtracted from a preset initial permissions value.

  # umask –S

  # umask [octalnumber]

Initial Permissions	666	
umask	– 002	(subtract)
==========		
Default Permissions	664	