Question Bank 23INMCA202 - Linux/ Unix Fundamentals

Course Type	Course Nature	CA Conduct	System	L T P	Credits	CA Total	CA Pass	SEE Total	SEE Pass	Total Pass
Theory	1	End Semester	Mark	3 1 0	4	40	0	60	24	50

Question Bank Summary

Sect. Part A	Sect. Part B	Easy	Med.	Chall.	Th.	Appli.
60	60	16	68	36	22	98

Part A

#	Unit	Question	COS	Categorized
1	1.1	**You are a system administrator in a company and your supervisor has tasked you with creating a presentation for new employees about the Linux Shell, explaining what a shell is, its role in interacting with the operating system, and a comparison of the various shells available in Linux** A. **Define what a shell is in the context of Linux. Explain its role in interacting with the operating system and how it functions as an intermediary between the user and the system. (1 marks)** B. **There are different types of shells available in Linux, each with its unique features. Compare the following types of shells: Bourne Shell (sh): C Shell (csh): Bash Shell (bash): Korn Shell (ksh):** (2 marks) In the context of Linux, a **shell** is a command-line interface (CLI) that allows users to interact with the operating system. It acts as an **intermediary** between the user and the system's core components (the **kernel**). When you type commands into the shell, it interprets them and passes them to the operating system to execute. After execution, it displays the output back to the user. **Key Roles of the Shell:** ● **Command interpreter**: Reads and executes commands entered by the user. ● **Automation tool**: Supports scripting to automate tasks. ● **Environment management**: Manages user environment variables and sessions. ● **Process control**: Allows users to start, stop, and manage system processes. b. **1. Bourne Shell (sh):** ● Developed by Stephen Bourne at AT&T Bell Labs.	CO1,CO1	Easy - Understanding - T

- Basic and straightforward syntax.
- Good for writing simple scripts.
- Highly portable across Unix systems.
- Limited interactive features.
- Commonly used for legacy scripts.

2. C Shell (csh):

- Developed by Bill Joy at the University of California, Berkeley.
- Syntax resembles the C programming language.
- Better interactive use (command history, job control).
- Less portable than Bourne Shell.
- Moderate scripting capabilities.

3. Bash Shell (bash):

- Developed by GNU Project as an improvement of sh.
- Stands for **Bourne Again Shell**.
- Most popular and default shell on Linux systems.
- User-friendly features: tab completion, command history, scripting enhancements.
- Very portable and widely supported.
- Supports advanced scripting and automation.

4. Korn Shell (ksh):

- Developed by David Korn at AT&T Bell Labs.
- Combines features of Bourne Shell and C Shell.
- Powerful scripting capabilities.
- Better performance and scripting than csh.
- Often used in enterprise environments for complex scripts.
- Interactive features and good compatibility.

2	1.1	A new team member, who is relatively new to Linux, needs to understand the concepts of symbolic links and hard links to better organize files and share data between different locations on the system. Your supervisor has asked you to create a guide that explains these concepts clearly and compares the differences between symbolic and hard links. A. Define what symbolic links and hard links are in Linux. Explain how they differ from each other in terms of functionality and use cases (1 marks) B. In a scenario where you need to create a link for a frequently accessed configuration file so that it can be easily referenced from multiple locations without duplicating the file, which type of link would you recommend? Justify your choice with specific advantages. (2 marks) **Symbolic Link (Symlink):** ● A **symbolic link** is like a shortcut or pointer to the original file. ● It contains a path that points to the actual file or directory. ● If the original file is moved or deleted, the symbolic link becomes **broken** (dangling link). **Hard Link:** ● A **hard link** is a direct reference to the data on disk (inode). ● Both the original file and hard link point to the same inode (data blocks). ● If the original file is deleted, the data still exists because the hard link maintains access to it. B. **Symbolic Link (Symlink)** **Justification:** ● **Cross-directory and cross-filesystem support:** Symlinks can point to files in different directories or even different mounted filesystems, making them very flexible. ● **Easier to identify and manage:** It's clear which files are symlinks, making maintenance simpler. ● **No duplication needed:** Symlinks avoid creating extra copies while giving you quick access from multiple locations. ● **Simple to update:** If the original file location changes, you only need to update the symlink. ● **Common practice for config files:** It's a standard Linux practice to use symlinks for frequently accessed configs (e.g., /etc/nginx/sites-enabled/).	CO1,CO1	Easy - Rememberin g - T
3	1.1	Your supervisor has tasked you with creating a detailed guide that explains the various types of files in Linux, their characteristics, and typical use cases. The team will refer to this guide while setting up new systems and performing file management tasks. Analyze different types of files in Linux and their significance in the operating system. Breakdown each type in detail, including their use cases and examples. **Regular Files** **Description:** ● Standard files that store data. ● Can be text files, images, videos, scripts, or executables. **Identifier:** ● Shown as - in ls -l output. **Use Cases:** ● Configuration files (e.g., /etc/hosts) ● Shell scripts (e.g., backup.sh) ● Documents and images (e.g., report.docx, photo.jpg) /home/user/notes.txt **Directory Files** **Description:** ● Special files that contain references to other files and directories. ● Used to organize the filesystem in a hierarchical structure. **Identifier:**	CO1	Medium - Analysing - T

		● Shown as d in ls -l output. **Use Cases:** ● Grouping related files and folders. ● System directories (e.g., /home, /etc, /var) /home/ etc/ var/log/ **Links** **A. Symbolic Link (Symlink)** **Description:** ● A shortcut or reference to another file or directory. ● Can point to files/directories across different file systems. **Identifier:** ● Shown as l in ls -l output. **Use Cases:** ● Simplify access to deeply nested files. ● Manage multiple versions of config files. **Example:** ln -s /var/log/syslog ~/syslog_link **Hard Link** **Description:** ● A direct reference to the same inode as the original file. ● Acts like another name for the same file. **Identifier:** ● Appears as regular file (-) in ls -l. **Use Cases:** ● Duplicate file access without using extra disk space. ● Ensures file access even if the original filename is removed. **Example:** ln original.txt hardlink.txt		
4	1.1	Linux systems support a variety of user types, each with distinct roles and privileges. When configuring a Linux system for specific applications, it is crucial to understand these user types and how they interact with the applications and services running on the system. At the application level, different user types are often used to ensure the proper functioning, security, and isolation of processes. A. Categorize the any 3 types of users in Linux and explain how they are typically configured in a multi-application environment. (3 marks) **Root User (Superuser)** **Description:** ● The most powerful user in the system. ● Has unrestricted access to all commands, files, system settings, and resources. **Configuration in Multi-Application Environment:** ● Used for **system administration tasks** such as installing applications, managing users, configuring system services. ● Should **not** run application processes directly (for security reasons). ● Access is protected and limited, often using sudo for elevated privileges. **. Regular User (Normal User Account)** **Description:** ● Created by the system administrator for everyday tasks. ● Has limited access, mostly to their own files and permitted applications. **Configuration in Multi-Application Environment:** ● Used by developers, operators, or end-users to interact with specific applications.	CO1	Medium - - T

		● Cannot affect system-level files or other users' data. ● Can run applications and processes assigned to them, without risking the whole system. **System Users (Service Accounts)** **Description:** ● Non-login users created by the system or admin to run background services and daemons. ● Examples: www-data for web servers, mysql for database servers. **Configuration in Multi-Application Environment:** ● Applications and services run under dedicated system users to **isolate processes**, **increase security**, and **control permissions**. ● These users usually do not have home directories or login shells. ● Helps prevent services from interfering with each other and limits damage in case of compromise.		

5	1.1	Describe the structure of the Linux/Unix root directory and its common sub directories. A. Illustrate the purpose of the /home directory and its typical usage. (1 marks) B. Show the role of the /bin directory in Linux/Unix systems. (1 marks) C. Interpret the purpose of the /lib directory and provide an example of the type of files stored there. (1 marks) **A. /home Directory** **Purpose:** ● /home contains **personal directories** for all regular (non-system) users. ● Each user gets their own subdirectory under /home, for example: ○ /home/john ○ /home/alice **Typical Usage:** ● Stores **user-specific files**, documents, configurations, and personal scripts. ● Users can create, modify, and delete files within their own home directory. ● It keeps user data **separated from system files**, making backup and user management easier. **/bin Directory** **Purpose:** ● /bin stands for "**binary**" and contains **essential command-line binaries (executables)**. ● These are the most basic tools needed by the system and all users to operate. **Role:** ● Contains core commands required for basic system operation in **single-user mode** or when other directories are unavailable. ● Accessible by **all users** (root and regular users). **/lib Directory** **Purpose:** ● /lib contains **shared library files** that are required by the binaries in /bin and /sbin. ● These are similar to .dll files in Windows. **Role:** ● Stores **essential libraries** that help executables run. ● Also includes **kernel modules** and drivers necessary for hardware and system functionality. **/lib/modules/** → Kernel modules (device drivers).	CO1,CO1,CO1	Medium - - T

6	1.1	Analyze an absolute path and a relative path in Linux/Unix. A. Illustrate absolute and relative paths with examples. (1 marks) B. Evaluate the scenario that when would you use an absolute path over a relative path and why? (1 marks) C. Justify the situation that how can you navigate using relative paths from your current directory? (1 marks) **A. Absolute Path vs. Relative Path (Examples)** ● **Absolute Path:** ○ Starts from the **root directory** /. ○ Example: /home/user/documents/file.txt ● **Relative Path:** ○ Starts from the **current working directory**. ○ Example (if you are in /home/user/): documents/file.txt **When to Use Absolute Path Over Relative Path** ● When you need **certainty**, regardless of the current location. ● Useful in **scripts, cron jobs**, or system configurations. ● Ensures **consistency** in accessing system files and directories. **C. Navigating Using Relative Paths** ● Use cd directory_name/ to move into a subdirectory. ● Use cd .. to move **up one level** to the parent directory. ● Use cd ../../folder/ to move **up two levels**, then into another folder. ● Use ./file.txt to access files in the **current directory**.	CO1,CO1,CO1	Medium - - T
7	1.1	Demonstrate the different types of shells available in Linux and their key features. A. Predict the default shell in most Linux distributions, and why is it commonly used? (1 marks) B. Analyze the Bash shell and the Zsh shell. (1 marks) C. Examine the role does the shell play in Linux, and how does it interact with the kernel? (1 marks) **Default Shell in Most Linux Distributions** ● **Default Shell: Bash (Bourne Again Shell)** ● **Why Commonly Used:** ○ User-friendly and powerful. ○ Wide support for scripting and automation. ○ Backward compatible with the original Bourne shell. ○ Available by default in most Linux distributions. **B. Analysis: Bash Shell vs. Zsh Shell** ● **Bash Shell:** ○ Default shell in many systems. ○ Strong scripting capabilities. ○ Auto-completion for commands and filenames. ○ Supports command history. ● **Zsh Shell:** ○ Advanced features compared to Bash. ○ Superior auto-completion and suggestions. ○ Theme and plugin support (e.g., Oh-My-Zsh). ○ Customizable prompts and better globbing. **C. Role of the Shell in Linux & Interaction with Kernel** ● **Role of the Shell:** ○ Acts as a **command interpreter** between the user and the Linux system. ○ Translates human-readable commands into machine-readable instructions. ● **Interaction with Kernel:** ○ Shell sends user commands to the **kernel** via system calls. ○ Kernel manages hardware and executes the requested tasks. ○ Shell then displays the output back to the user.	CO1,CO1,CO1	Medium - - A

8	1.1	Demonstrate how to navigate between directories in Linux using command-line tools. A. Use a command that would you to move to a parent directory, and explain how does it work? (1 marks) B. How can you display the current working directory? Give an example. (1 marks) C. Use a command to list the contents of a directory, and what options can you use with it to display hidden files? (1 marks) **A. Move to Parent Directory** ● **Command:** cd .. ● **Points:** ○ cd is used to change directories. ○ .. represents the parent directory. ○ Using cd .. moves you up one level in the directory hierarchy. **Display Current Working Directory** ● **Command:** pwd ● **Points:** ○ pwd stands for "print working directory." ○ It shows the full path of your current directory. ○ Example: /home/user/documents **C. List Contents of Directory (Including Hidden Files)** ● **Command:** ls -a ● **Points:** ○ ls lists the contents of a directory. ○ -a option shows **all files**, including hidden ones (files starting with .). ○ Helps to view configuration files and directories that are hidden by default.	CO1,CO1,CO1	Medium - Applying - A
9	1.1	Demonstrate how Linux file manipulation commands can be used to manage files and directories effectively. A. Predict the command used to create a new empty file in Linux and illustrate an example of its usage. (1 marks) B. Show the process of deleting a file in Linux and discuss the safety precautions to consider before performing this operation. (1 marks) C. Illustrate the command used to move a file from one directory to another in Linux, and determine the role of the -i option in this command (1 marks) Using cat to create a file: cat > filename After typing this, the terminal waits for input. You can type something or leave it blank. To save and exit, press **Ctrl + D**. If you don't type anything and just press **Ctrl + D**, it creates an empty file. **B. Delete a File in Linux** ● **Command:** rm filename ● **Points:** ○ rm is used to remove (delete) files. ○ **Example:** rm old_file.txt ○ **Safety Precautions:** ▪ Double-check the file name before deletion. ▪ Use -i option for confirmation before deleting: rm -i file.txt ▪ Avoid using rm with wildcards (*) unless you're sure of the impact. **C. Move a File from One Directory to Another** ● **Command:** mv source_file destination_directory/	CO1,CO1,CO1	Challenging - - A

		● **Points:** ○ mv is used to move or rename files and directories. ○ **Example:** mv report.txt /home/user/documents/ moves the file to the documents folder. ○ **-i option:** ▪ Prompts before overwriting if a file with the same name exists in the destination. ▪ **Example:** mv -i report.txt /home/user/ asks for confirmation before overwriting.		

10	1.1	Analyze the process of changing file permissions in Linux.
		A. Examine the permissions of a file in Linux. Provide an example output. (1 marks)
		B. Evaluate and explain the following command chmod 755 sample.txt. (1 marks)
		C. Provide the exact command for setting read and write permissions to a specific user for a particular file. (1 marks)
		D. **Command to check permissions:**
		`ls -l filename`
		E. **Example output:**
		`-rw-r--r-- 1 user user 1234 Apr 7 10:00 sample.txt`
		F. **Explanation:**
		• `-` → Regular file
		• `rw-` → User (owner) has read (r) and write (w) permissions
		• `r--` → Group has read permission
		• `r--` → Others have read permission
		• `user user` → Owner and group names
		• `1234` → File size in bytes
		• `Apr 7 10:00` → Last modified date/time
		• `sample.txt` → File name
		Explain: chmod 755 sample.txt
		● **Command:**
		chmod 755 sample.txt
		● **Explanation of 755:**
		○ **7** → Owner: Read (4) + Write (2) + Execute (1) = **7**
		○ **5** → Group: Read (4) + Execute (1) = **5**
		○ **5** → Others: Read (4) + Execute (1) = **5**
		● **Result:**
		○ Owner: Full permissions (read, write, execute)
		○ Group & Others: Read and execute only (no write)
		● **Usage:** Commonly used for scripts or executable files.
		C. Command for setting read and write permissions for a specific user
		● **Command:**
		chmod u+rw filename
		● **Explanation:**
		○ chmod → Change permissions
		○ u+rw → Add read (r) and write (w) permissions for the user (owner)
		○ **Example:**
		chmod u+rw sample.txt
			CO1,CO1,CO1	Medium - - A

11	1.1	Examine how to manage file ownership and access control in Linux. A. Illustrate the role of file ownership in Linux, and explain how to change the ownership of a file using the chown command. Provide an example (1 marks) B. Examine the importance of file permissions in Linux and explain how to modify the permissions of a file using the chmod command. Include an example with different permission settings. (1 marks) C. Demonstrate how to use the chgrp command to manage group ownership of a file and explain its significance in access control. (1 marks)	CO1,CO1,CO1	Medium - - A

A. Role of File Ownership & Using chown Command

- **Role of File Ownership:**
  - In Linux, every file is owned by:
    - **User (Owner)**
    - **Group**
  - Ownership determines who can read, write, or execute the file.
  - Helps maintain security and control over files in a multi-user system.
- **Change ownership with chown:**

chown username filename

- **Example:**

chown john sample.txt

  - Changes the owner of sample.txt to user **john**.

B. Importance of File Permissions & Using chmod

- **Importance:**
  - File permissions control **who can read, write, or execute** a file.
  - Prevents unauthorized access and ensures data integrity.
- **Modify permissions with chmod:**

chmod permission filename

- **Example:**

chmod 644 sample.txt

  - **6** → User: Read + Write
  - **4** → Group: Read only
  - **4** → Others: Read only
- **Another example (more permissive):**

chmod 755 sample.sh

  - User: Read, Write, Execute
  - Group: Read, Execute
  - Others: Read, Execute

C. Using chgrp to Manage Group Ownership

- **Purpose:**
  - Changes the **group ownership** of a file.
  - Important for collaborative environments where multiple users belong to the same group.
- **Command:**

chgrp groupname filename

- **Example:**

chgrp developers project.txt

  - Changes the group of project.txt to **developers**.
- **Significance:**
  - Allows group members to access and modify files according to group permissions.
  - Helps manage shared access in team projects.

12	1.1	Analyze the usage of the head and tail commands in Linux for viewing file contents. A. Demonstrate how to display the first 10 lines of a file using head command. Provide an example (1 marks) B. Illustrate how to display the last 20 lines of a file. Include an example. (1 marks) C. Infer how the -f option with the tail command is used for monitoring real-time changes in a file. Provide a use case example. (1 marks) **A. Display First 10 Lines with head Command** ● **Command:** head -n 10 filename ● **Example:** head -n 10 sample.txt ● **Explanation:** ○ Displays the first 10 lines of the file sample.txt. ○ Useful for quickly checking the beginning of a file. **B. Display Last 20 Lines with tail Command** ● **Command:** tail -n 20 filename ● **Example:** tail -n 20 log.txt ● **Explanation:** ○ Shows the last 20 lines of the file log.txt. ○ Useful for viewing recent entries in a growing file like logs. **C. Using tail -f for Real-Time Monitoring** ● **Command:** tail -f filename ● **Example:** tail -f /var/log/syslog ● **Explanation:** ○ -f stands for **follow**. ○ Continuously displays new lines added to the file in real time. ○ **Use case:** Monitoring system logs, server logs, or application logs to see live updates as events happen.	CO1,CO1,CO1	Medium - - A
13	2.1	Explain the concept of local and global variables in Linux shell scripting. A. Discuss the difference between a local and a global variable in shell scripting. Provide examples of each. (1 marks) B. Restate the method for declaring and accessing a global variable in a shell script. (1 marks) C. Interpret the scope of a local variable within a function, and how is it different from a global variable's scope? (1 marks) **Difference Between Local and Global Variables (With Examples)** ● **Global Variable:** ○ Accessible **throughout the script**, including inside functions (unless shadowed). ○ **Example:** global_var="Hello" echo $global_var ● **Local Variable:** ○ Declared **inside a function**, accessible **only within that function**. ○ **Example:** my_function() { local local_var="Hi"	CO2,CO2,CO2	Medium - - T

		echo $local_var
		}
		my_function
		
		B. Declaring and Accessing a Global Variable
		● **Declaration:**
		
		global_var="Global Value"
		● **Accessing:**
		
		echo $global_var
		● **Usage in Script:**
		
		global_var="Script Wide"
		my_function() {
		echo "Inside function: $global_var"
		}
		my_function
		echo "Outside function: $global_var"
		
		C. Scope of Local Variable vs Global Variable
		● **Local Variable Scope:**
		○ Exists **only inside the function** where it is defined.
		○ Cannot be accessed outside the function.
		● **Global Variable Scope:**
		○ Accessible **anywhere** in the script (including functions).
		●		

14	2.1	Describe how conditional statements are used in shell scripting with examples. A. Demonstrate the syntax of a basic if statement in shell scripting. Provide a simple example of its use. (1 marks) B. Paraphrase how would you use an if-else statement to check if a file exists? Write a short script to demonstrate. (1 marks) C. In which situations would you use if..elif..else instead of a simple if-else statement? Provide an example of its usage. (1 marks) **A. Basic if Statement Syntax (With Example)** ● **Syntax:** if [condition]; then # commands fi ● **Example:** number=10 if [$number -gt 5]; then echo "Number is greater than 5" fi **if-else Statement to Check if a File Exists (With Script)** ● **Script Example:** filename="test.txt" if [-e "$filename"]; then echo "File exists." else echo "File does not exist." fi ● **Explanation:** ○ -e checks if the file exists. ○ If the condition is true, it prints "File exists."	CO2,CO2,CO2	Easy - - T

		○ Otherwise, it prints "File does not exist." **C. When to Use if..elif..else (With Example)** ● **Use Case:** Use when you need to **check multiple conditions**, not just one. ● **Example:** number=15 if [$number -lt 10]; then echo "Number is less than 10" elif [$number -eq 15]; then echo "Number is equal to 15" else echo "Number is greater than 10 but not equal to 15" fi ● **Explanation:** ○ if..elif..else lets you check multiple possibilities in a structured way. ○ Better than nesting multiple if-else for readability.		
15	2.1	Explain the case statement in shell scripting and provide a practical example. A. Illustrate the basic syntax of a case statement in Linux shell scripting? (1 marks) B. Show a scenario that you use the case statement to match multiple patterns for a variable. Provide an example where you match user input and perform actions based on the value. (1 marks) C. Examine the role of the ;; in a case statement, and what happens if you omit it?(1 marks) **A. Basic Syntax of case Statement in Shell Scripting** ● **Syntax:** case $variable in pattern1) # commands ;; pattern2) # commands ;; *) # default commands ;; esac ● **Explanation:** ○ case begins the block. ○ $variable is compared against multiple patterns. ○ esac ends the case block. **B. Example: Matching User Input with Multiple Patterns** ● **Scenario:** Prompt user to enter a day, then respond accordingly. ● **Example Script:** echo "Enter a day:" read day case $day in Monday|monday) echo "Start of the workweek!" ;; Friday|friday) echo "Almost weekend!" ;; Saturday|saturday|Sunday|sunday) echo "It's the weekend!" ;;	CO2,CO2,CO2	Medium - - T

		*) echo "Midweek days!" ;; esac ● **Explanation:** ○ The | symbol is used to match multiple patterns. ○ The * acts as the default case (like else). **C. Role of ;; in a Case Statement** ● **Purpose of ;;:** ○ Marks the **end of commands** for a specific pattern. ○ Without it, shell will continue executing the next case block (**fall-through**), which is not usually desired. ● **If Omitted:** ○ Commands from the next case pattern will execute, potentially leading to unexpected behavior case $choice in 1) echo "Option 1 selected" # missing ;; 2) echo "Option 2 selected" ;; esac If choice=1, both "Option 1 selected" and "Option 2 selected" will print. (Bad!)		
16	2.1	The test command in shell scripting is used to evaluate expressions and check various conditions, such as file attributes, string comparisons, and numerical comparisons. It returns a true or false result, depending on whether the condition being tested is satisfied. Describe the purpose and application of the test command in shell scripting. Specifically, explain how it is used to evaluate conditions and perform checks in a script, and provide an example demonstrating its syntax for checking whether a file exists. (3 marks) **Purpose and Application of the test Command in Shell Scripting:** The test command in shell scripting is used to evaluate expressions and determine whether specific conditions are true or false. It plays a crucial role in controlling the flow of execution in shell scripts, especially in conditional statements like if, while, and logical operations. The test command can check: ● **File attributes** (e.g., if a file exists, is readable, writable, etc.) ● **String comparisons** (e.g., if two strings are equal, not equal, etc.) ● **Numerical comparisons** (e.g., if numbers are equal, greater than, etc.) If the condition is **true**, test returns an exit status of 0. If the condition is **false**, it returns an exit status of 1. EXAMPLE : #!/bin/bash filename="example.txt" if test -e "$filename" then echo "The file '$filename' exists." else echo "The file '$filename' does not exist." Fi test -e "$filename" checks if the file named example.txt exists.	CO2	Challenging - - A

		◾ If it exists, the script prints a confirmation message. ◾ If it does not exist, it prints an alternative message.		
17	2.1	The test command in shell scripting is a utility used to evaluate conditional expressions. Illustrate how to use the test command to compare two numbers for equality in a shell script. Offer a concrete example that demonstrates this comparison. **Using test to Compare Two Numbers for Equality in Shell Script:** The test command can be used to compare numbers using specific operators. For **equality**, the operator is: ● -eq — **Equal to** **Example Script:** #!/bin/bash num1=10 num2=20 if test "$num1" -eq "$num2" then echo "The numbers are equal." else echo "The numbers are not equal." fi **Explanation:** ● test "$num1" -eq "$num2" checks if the values of num1 and num2 are equal. ● Since 10 is not equal to 20, the script will output: **"The numbers are not equal."** **Alternative Syntax (Using [ ]):** if ["$num1" -eq "$num2"] then echo "The numbers are equal." else echo "The numbers are not equal." fi Both versions work the same!	CO2	Easy - Applying - A
18	2.1	In Linux, different types of loops can be applied to automate repetitive tasks. Explain the function and structure of a while loop in shell scripting. Provide an example where a while loop is used to print numbers from 1 to 5. Analyze the use of this loop when you need to repeat an action as long as a certain condition is true. **Function and Structure of a while Loop in Shell Scripting:** The while loop in shell scripting is used to repeatedly execute a block of code **as long as a specified condition is true**. It is commonly used for tasks where you don't know beforehand how many times you need to run the loop, but you continue until a certain condition becomes false. **Structure:** while [condition] do # Commands to execute done ● **Condition:** Checked before every iteration. ● If the condition is **true**, the commands inside the loop are executed. ● The loop continues until the condition becomes **false**. **Example: Print Numbers from 1 to 5** *#!/bin/bash*	CO2	Medium - Analysing - A

		count=1 *while [$count -le 5]* *do* *echo "Number: $count"* *count=$((count + 1))* *done* **Explanation:** ● We start by initializing count=1. ● The condition [$count -le 5] checks if count is less than or equal to 5. ● Inside the loop: ○ It prints the current value of count. ○ Then increments count by 1. ● When count becomes 6, the condition fails, and the loop exits. ● **Output:** Number: 1 Number: 2 Number: 3 Number: 4 Number: 5 **Analysis of while Loop Usage:** ● The while loop is **ideal when you need to repeat an action as long as a condition remains true**. ● It is especially useful when: ○ You don't know in advance how many times you need to repeat the action. ○ You're waiting for a particular event or condition to change (e.g., a process to finish, a file to appear, user input, etc.). ● In this example, we repeated printing numbers **until** the condition (count <= 5) was no longer true.		
19	2.1	In shell scripting, both the while loop and the until loop are used to repeat actions based on a condition, but they function in different ways. Imagine you're tasked with automating a script that prints numbers in reverse order as part of a countdown for a scheduled event. You need to write a script that counts down from 5 to 1. Consider the key differences between a while loop and an until loop and how each could be applied to this task. **Difference Between while Loop and until Loop in Shell Scripting** 1. **Condition Check Logic** ○ **while** loop: Executes **as long as the condition is true**. ○ **until** loop: Executes **as long as the condition is false** (continues **until** the condition becomes true). 2. **Use Case Approach** ○ **while** loop: Used when you expect to run the loop while something remains true. ○ **until** loop: Used when you want to run the loop until something becomes true (opposite thinking). 3. **Readability & Preference** ○ **while** loop: More commonly used for regular looping tasks. ○ **until** loop: Useful for situations where you wait for a particular condition to occur (like waiting for a process to finish or a file to appear). 4. **Syntax Similarity** ○ Both use the same basic structure: while [condition] do	CO2	Challenging - Applying - A

		# commands done until [condition] do # commands done **Countdown from 5 to 1** **Using while loop:** #!/bin/bash count=5 while [$count -ge 1] do echo "Countdown: $count" count=$((count - 1)) done ● Condition: count -ge 1 (greater than or equal to 1). ● As long as this is true, the loop runs. **Using until loop:** #!/bin/bash count=5 until [$count -lt 1] do echo "Countdown: $count" count=$((count - 1)) done ● Condition: count -lt 1 (less than 1). ● The loop continues **until** this becomes true.		
20	2.1	You are working on a Linux-based system to edit configuration files, write scripts, and manage system settings. To carry out these tasks efficiently, you need to use the right text editor that suits your workflow. Classify the key features of at least three editors, focusing on their strengths and typical use cases in Linux-based environments. **1. Vim (Vi Improved)** **Key Features:** ● Modal editor (different modes for editing, command, and navigation). ● Extremely fast and lightweight. ● Powerful keyboard shortcuts and macros. ● Syntax highlighting for many languages. ● Highly customizable with plugins and .vimrc configuration. **Strengths:** ● Ideal for advanced users who prefer keyboard-driven workflows. ● Available by default on almost every Linux distribution. ● Great for editing files over SSH. **Typical Use Cases:** ● Quick editing of system configuration files. ● Writing shell scripts. ● Automating repetitive editing tasks. **2. Nano** **Key Features:**	CO2	Challenging - Analysing - T

		● Simple, beginner-friendly interface. ● Easy-to-use shortcut commands displayed at the bottom of the screen. ● No need to learn complex modes (like in Vim). ● Supports syntax highlighting (limited). **Strengths:** ● Very easy to learn for beginners. ● Lightweight and fast to open. ● Available by default on many Linux systems. **Typical Use Cases:** ● Quick edits to system files and scripts. ● Preferred by users who need a no-fuss editor. ● Ideal for straightforward editing tasks without advanced features. **3. Visual Studio Code (VS Code)** **Key Features:** ● Graphical text editor with a modern UI. ● Rich extension ecosystem (linting, version control, Docker, SSH). ● Built-in terminal and Git integration. ● IntelliSense for code completion. ● Supports remote development over SSH. **Strengths:** ● Feature-rich and highly extensible. ● Excellent for working on large projects and programming in multiple languages. ● Good balance between power and ease of use. **Typical Use Cases:** ● Writing and debugging scripts and applications. ● Managing configuration files in complex environments. ● Collaborative coding and version control workflows.		
21	2.1	You are working on a Linux system and need to manipulate the contents of files by shifting data or rearranging columns within them for better readability or processing. The shift command is an essential tool to modify the positional parameters in a shell script. A. Describe the function and purpose of the shift command in Linux. (1 marks) B. Examine how the shift command is used to shift positional parameters in shell scripting. (1 marks) C. Provide an example of a scenario where the shift command would be used effectively to process multiple input arguments in a script. (1 marks) **Describe the function and purpose of the shift command in Linux. (1 mark)** The shift command in Linux shell scripting is used to **move positional parameters to the left**, effectively discarding the current $1 and shifting the remaining parameters down by one. This is helpful when processing multiple arguments in a loop, as it allows you to work with the next argument without explicitly tracking position numbers. **B. Examine how the shift command is used to shift positional parameters in shell scripting. (1 mark)** When you run a script with arguments, they are stored as $1, $2, $3, etc. Using shift, the value of $2 becomes $1, $3 becomes $2, and so on. Each shift removes the first parameter, making it easier to iterate over all arguments. Syntax: bash CopyEdit shift [n] ● If n is specified, shift by n positions. ● If n is not specified, shift by 1 position by default.	CO2,CO2,CO2	Challenging - - A

		C. Example: Processing multiple input arguments in a script (1 mark) Here's a simple shell script that processes all input arguments using shift: bash CopyEdit #!/bin/bash echo "Processing arguments:" while ["$#" -gt 0]; do echo "Argument: $1" shift done **Explanation:** ● "$#" checks the number of arguments remaining. ● $1 prints the current argument. ● shift moves to the next argument. ● The loop continues until all arguments are processed.		
22	2.1	You are tasked with editing configuration files on a Linux server using the vi editor. As part of this task, you need to navigate, modify, and save files efficiently using various vi commands. A. Describe the key commands used in the vi editor for navigation, editing, and saving files. (1 marks) B. Provide an example where you would use specific vi commands to open a file, make changes, and save those changes, demonstrating their practical use in a real-life scenario. (2 marks) **A. Key commands used in the vi editor for navigation, editing, and saving files (1 mark)** **Function** — **Command** **Open file** — vi filename **Navigation** — h (left), l (right), j (down), k (up) **Move to line number** — :n (where n is line number) **Insert mode** — i (insert before cursor), a (append after cursor), o (open new **Delete text** — x (delete character), dd (delete line) **Undo** — u (undo last change) **Save changes** — :w (write/save) **Quit** — :q (quit), :wq (save and quit), :q! (quit without saving) **B. Example: Using vi to open a file, make changes, and save them (2 marks)** **Scenario:** You need to edit the server configuration file /etc/example.conf to change the port number from 8080 to 9090. **Steps:** 1. **Open the file:** vi /etc/example.conf 2. **Navigate to the line with the port number:** ○ Use arrow keys or j/k to move up and down. 3. **Enter insert mode:** ○ Move the cursor over the number 8080. ○ Press i to enter **insert mode**. 4. **Edit the number:** ○ Change 8080 to 9090. 5. **Exit insert mode:** ○ Press Esc. 6. **Save and exit:** ○ Type :wq and press **Enter**. **Summary command flow:** vi /etc/example.conf # Navigate to the line	CO2,CO2	Medium - - A

		i # Make changes Esc :wq		

23	2.1	The test command returns a status code: 0 (true) if the condition is met, and 1 (false) if it is not. A. Examine how to use the test command to check if a string is empty in a shell script. (1 marks) B. Provide a practical example showing how this check is implemented. (2 marks) **A. How to use the test command to check if a string is empty in a shell script (1 mark)** ● In shell scripting, to check if a string is **empty**, you use: bash CopyEdit test -z "$string" ● -z option checks if the length of the string is **zero**. ● If the string is empty, test returns **0** (true). **B. Practical example (2 marks)** Let's say you want to check if a user provided an argument to the script. If not, you want to print a message. #!/bin/bash input=$1 if test -z "$input" then echo "No input provided!" else echo "You entered: $input" fi **Explanation:** ● $1 is the first argument passed to the script. ● test -z "$input" checks if the variable input is empty. ● If it is empty, it prints "No input provided!" ● Otherwise, it echoes the entered value.	CO2,CO2	Easy - Applying - A
24	2.1	Discuss the syntax of iteration loop and demonstrate their practical use in various scenarios. A. Describe the use of a for loop in a shell script. Explain its syntax and provide a practical example of how a for loop can be used to iterate over a list of numbers from 1 to 5. (2 marks) B. Justify why the for loop is suitable for situations where you know the number of iterations in advance. (1 marks) **A. Describe the use of a for loop in a shell script (2 marks)** **Purpose:** ● The for loop in shell scripting is used to iterate over a list of items, such as numbers, strings, or filenames. ● It is especially useful when you want to repeat a block of code **a specific number of times** or process **a predefined list of items**. **Syntax:** for variable in list do commands	CO2,CO2	Challenging - Evaluating - A

		done **Practical Example:** Printing numbers from **1 to 5** #!/bin/bash for number in 1 2 3 4 5 do echo "Number: $number" done **Output:** javascript CopyEdit Number: 1 Number: 2 Number: 3 Number: 4 Number: 5 **Explanation:** ● The loop variable number takes each value from the list 1 2 3 4 5. ● The echo command prints each number. **B. Why the for loop is suitable for known iterations (1 mark)** ● The for loop is ideal when the **number of iterations is known in advance**, because: ○ You can explicitly define the list of items or range. ○ It provides clear and predictable flow control. ○ There is **no need for additional conditions** to control the loop — making scripts simpler and easier to read. **Example use-case:** ● Processing a fixed list of filenames. ● Iterating through a known range of numbers for reports. ● Running tests on a fixed set of inputs.		
25	3.1	You are working on a large project where files are scattered across various directories. You need to locate a file named config.txt in a particular directory structure to troubleshoot an issue. A. Examine how you would use the find command to locate the config.txt file within a specific directory. Provide an example of your command. (2 marks) B. Discuss the key differences between find and locate command. Which would you prefer for a quick search, and why? (1 marks) **A. Using the find command to locate config.txt in a specific directory (2 marks)** The find command searches for files and directories in a directory hierarchy based on conditions like name, type, size, etc. **Example command:** find /path/to/directory -name "config.txt" **Explanation:** ● /path/to/directory — starting directory for the search. ● -name "config.txt" — tells find to look for files with the name config.txt. If you want to search **case-insensitively**, you can use: find /path/to/directory -iname "config.txt" **B. Key differences between find and locate commands (in points):** ● **Search method:** ○ find: Searches **directly in the filesystem** in real-time. ○ locate: Searches in a **pre-built database** of file locations. ● **Speed:** ○ find: Slower, especially in large directories, because it scans live. ○ locate: Very fast, as it reads from an indexed database. ● **Accuracy:** ○ find: Always accurate, finds even newly created files. ○ locate: May miss recent files if the database is not updated. ● **Usage flexibility:** ○ find: More powerful, supports advanced filtering (by name, type,	CO3,CO3	Medium - - A

		size, permissions, etc.). ○ locate: Simple name-based searches, less flexible. ● **Database requirement:** ○ find: No database needed. ○ locate: Requires regular database updates (using updatedb). **Conclusion — Which to prefer for a quick search:** ● Prefer **locate** for speed, especially when you need quick results. ● Use **find** for up-to-date and precise results, especially for newly created or modified files.		
26	3.1	While working on a script, you realize that your program produces too much output, and you want to handle the output and errors separately. Differentiate between standard input, standard output, and standard error? **1. Standard Input (stdin)** ● **Definition:** The source from which a program reads its input data. ● **File Descriptor:** 0 ● **Default Source:** Keyboard (when running scripts manually). ● **Usage Example:** When you use read command in a script to accept user input. read name echo "Hello, $name" **2. Standard Output (stdout)** ● **Definition:** The destination where a program writes its regular output. ● **File Descriptor:** 1 ● **Default Destination:** Terminal/screen. ● **Usage Example:** Sending output to a file: echo "Operation completed successfully" > output.txt **3. Standard Error (stderr)** ● **Definition:** The destination where a program writes its error messages. ● **File Descriptor:** 2 ● **Default Destination:** Terminal/screen (separate from standard output). ● **Usage Example:** Redirecting errors to a file: ls non_existing_file 2> error.txt	CO3	Easy - Understanding - T
27	3.1	You are troubleshooting an application and need to capture any error messages produced during its execution. You want to redirect those errors into a separate file for later analysis. A. Evaluate how would you use redirection to capture the standard error output of a command into a file? Provide an example of the command. (2 marks) B. What is the difference between > and >> in redirection? Provide an example of how to use both. (1 marks) **How to capture error messages into a file** When a command fails, it shows error messages (called **standard error**). To save these errors to a file, use: command 2> errorfile.txt	CO3,CO3	Medium - - A

		✅ **Example:** ls wrongfile.txt 2> errors.txt ● ls wrongfile.txt tries to list a file that doesn't exist. ● 2> means: send error messages to a file. ● Now, the error message is saved inside errors.txt instead of showing on the screen. **Difference between > and >> in Shell Redirection:** 1. **> (Single Greater Than)** ○ Used to **overwrite** a file. ○ If the file exists, it **replaces** the existing content. ○ If the file doesn't exist, it **creates** a new file. ○ **Example:** echo "Hello" > file.txt # file.txt will contain only: Hello 2. **>> (Double Greater Than)** ○ Used to **append** to a file. ○ Adds new content to the **end** of the existing file. ○ If the file doesn't exist, it **creates** a new file. ○ **Example:** echo "World" >> file.txt # Adds "World" at the end of file.txt, keeps old content too		
28	3.1	You are working as a system administrator, and you need to quickly check all the error logs in a large syslog file. You are looking for specific occurrences of "error" to troubleshoot. A. Examine how would you use the grep command to search for the word "error" in the syslog file? Provide the command and explain how it works. (2 marks) B. What would you do if you wanted to search for "error" regardless of whether it's uppercase or lowercase? How can you modify the grep command to make the search case-insensitive? (1 marks) **A. Using grep to search for "error" in the syslog file** **Command:** grep "error" /var/log/syslog **Explanation:** ● grep — Command to search text in files. ● "error" — The word you are searching for. ● /var/log/syslog — The file where you are searching (common system log file). ● This command will display all lines in the syslog file that contain the word **"error"** (case-sensitive). **B. To make the search case-insensitive** **Command:** grep -i "error" /var/log/syslog **Explanation:** ● -i — This option makes the search **case-insensitive**. ● So, it will match **"error"**, **"Error"**, **"ERROR"**, etc.	CO3,CO3	Medium - - A
29	3.1	You have a CSV file where the first column contains employee names, and the second column contains their salaries. You want to extract just the names of employees who earn more than $50,000. A. Use the cut command to extract the names of employees from the CSV file, assuming the columns are separated by commas. Provide the command and explain the output. (2 marks) B. If the salary information is in the third column, how would you modify the cut command to include the salary along with the names? (1 marks) **A. Use the cut command to extract the names (First column)** **Command:** cut -d ',' -f 1 employees.csv **Explanation:**	CO3,CO3	Medium - - A

		• cut — Command to extract sections from lines of input. • -d ',' — Set the delimiter to **comma** (CSV file). • -f 1 — Select **field 1**, which is the **employee name**. • employees.csv — The CSV file name. **Output:** It will print only the first column — the employee names. Example: John Doe Jane Smith Alice Johnson **B. If the salary is in the third column and you want names and salaries** **Command:** cut -d ',' -f 1,3 employees.csv **Explanation:** • -f 1,3 — Select both **field 1 (name)** and **field 3 (salary)**. • Output will look like: John Doe,55000 Jane Smith,48000 Alice Johnson,62000		
30	3.1	You are a system administrator working on a Linux server. Your manager has asked you to identify all files within the /var/log/ directory that were modified within the last 7 days and contain the word "error" in their name. You need to list these files to prepare a report on recent system errors. Using the find command, write a command to display all files in the /var/log directory that: • Have been modified within the last 7 days. • Contain the word "error" in their name. Explain the output you would expect when running this command. **Command:** find /var/log -type f -mtime -7 -name "*error*" **Explanation of the command:** • find — Used to search for files and directories. • /var/log — Target directory to search in. • -type f — Look for **files only** (not directories). • -mtime -7 — Files **modified in the last 7 days**. • -name "*error*" — File names that contain the word **"error"** anywhere in the name. **Expected Output:** When you run this command, it will list all the files in /var/log/ that: • Are regular files • Have been modified in the last 7 days • Have "error" in the filename **Example Output:** lua CopyEdit /var/log/application_error.log /var/log/error_messages.log	CO3	Challenging - Applying - A

		/var/log/sys_error_2025.log		
31	3.1	You have a text file named data.txt containing the data as "apple","banana" and "cherry". A. Which command will you use to append the word "orange" to the data.txt file without overwriting its existing contents? (1 marks) B. What is the effect of using > instead of >> in this case? (1 marks) C. How can you verify that the word "orange" has been added to the file?(1 marks) a. echo '"orange"' >> data.txt b. The effect of using > instead of >> is: ● > **overwrites** the entire contents of the file with the new data. So, if you do: echo '"orange"' > data.txt — the original data ("apple","banana","cherry") will be **lost**, and only "orange" will remain. ● >> **appends** the new data to the end of the file **without deleting existing data**. **C.** To verify that the word **"orange"** has been added to the file, you can use the cat command to display the file contents: cat data.txt If successful, you should see something like: arduino CopyEdit "apple","banana","cherry" "orange"	CO3,CO3,CO3	Easy - - A

32	3.1	You have two files: file1.txt and file2.txt. You want to merge the contents of both files into a new file called merged.txt without any duplication, using only redirection and command-line utilities. A. Which command combination will you use to merge the contents of both files into merged.txt? (1 marks) To merge the contents of file1.txt and file2.txt into merged.txt, you can use: cat file1.txt file2.txt > merged.txt B. What would happen if you used > instead of >> when appending to merged.txt?(1 marks) If you use > instead of >>: ● > **creates or overwrites** the target file. ● >> **appends** to the target file. So, if you do: cat file1.txt > merged.txt cat file2.txt > merged.txt — the second command will overwrite the contents of merged.txt from the first command. As a result, **only the contents of file2.txt will remain** in merged.txt. To safely add both, you need either: cat file1.txt > merged.txt cat file2.txt >> merged.txt or just: cat file1.txt file2.txt > merged.txt C. How would you prevent adding any duplicate lines from the two files into merged.txt? (1 marks) **C.** To prevent adding duplicate lines, you can **sort and use uniq** like this: cat file1.txt file2.txt | sort | uniq > merged.txt	CO3,CO3,CO3	Medium - - A

		Explanation: • cat file1.txt file2.txt → combines the contents. • sort → sorts the lines (required for uniq to work properly). • uniq → filters out duplicate lines. • > → writes the final result into merged.txt.		
33	3.1	You need to list all the files in the /var/log directory sorted by their size, but you only want to display the top 5 largest files. A. Which command combination will you use to list the files sorted by size and display only the top 5 largest files? (1 marks) To list all files in /var/log sorted by size and show only the top 5 largest files, use: ls -lhS /var/log | head -n 5 B. What happens if you do not use the | operator and instead run a separate ls command followed by head? (1 marks) ls -lhS /var/log head -n 5 ➡️ The second command (head -n 5) will not have any input. The head command expects input from a file or standard input (stdin). Without the pipe (|), head won't know what to process and will just wait for input or return nothing. So, the | operator is essential to pass the output of ls as input to head. C. How does the sort command work in this scenario, and why is it important for sorting by size? (1 marks) du -ah /var/log | sort -rh | head -n 5 Explanation: du -ah → shows sizes of all files and directories in human-readable format. sort -rh → sorts by human-readable numbers in reverse order (largest first). head -n 5 → takes the top 5 largest. ✅ Importance of sorting: Without sorting, the files would be listed in default order (like by name or modification time). Sorting is crucial here because we want to identify the largest files, which requires arranging them by size.	CO3,CO3,CO3	Easy - - A
34	3.1	You want to display all files in the /etc directory, but only want to see files that have the word "config" in their name, sorted by the last modification date. A. Which command combination will you use to achieve this task using pipes? (1 marks) ls -lt /etc | grep config Explanation: ls -lt /etc → lists files in /etc, sorted by modification time (newest first). grep config → filters only lines (files) that contain the word "config". B. How does the grep command help in narrowing down the results in this case? (1 marks) The grep command filters the output by matching lines that contain the specific pattern — in this case, the word **"config"**. Since ls -lt /etc lists all entries in /etc, grep config ensures that only filenames containing **"config"** are shown. Without grep, you'd see **all** files and directories. So, **grep acts like a filter**, showing you only relevant results matching your keyword. C. What would happen if you replaced grep with find in this command? (1 marks) 🎬 **grep works as a text filter**, reading output from ls -lt /etc and showing only lines containing "config".	CO3,CO3,CO3	Easy - - A

		🎬 ⚠️ **find is not a text filter.** It searches the filesystem directly — it does **not** read from ls output or standard input.		
35	3.1	You have started a background job that processes a large dataset using a custom script process_data.sh. After a while, you want to check the status of this background job. A. Which command can you use to check the status of the background job?(1 marks) B. What is the difference between using jobs and ps commands to check background processes? (1 marks) C. If you want to bring a background job to the foreground, which command would you use? (1 marks) Command: *jobs* This will show the list of jobs running in the background in the current terminal session. B. Difference between jobs and ps commands to check background processes: jobs command: Shows background jobs only for the current shell session. Displays job IDs (like [1], [2]), status (Running/Stopped), and command. Limited to jobs started in this terminal. ps command: Displays all processes in the system or filtered by user/session. Can show detailed information: PID, CPU/memory usage, etc. Useful to check processes beyond the current shell, including orphaned or daemon processes. C. Command to bring a background job to the foreground: ✅ Command: fg If multiple background jobs, specify job number: fg %1	CO3,CO3,CO3	Medium - - A
36	3.1	You are running a script backup.sh in the background to backup a large directory. You want to save the output of the script to a log file to monitor the progress later. A. Which command will you use to run backup.sh in the background and redirect its output to backup.log? (2 marks) ./backup.sh > backup.log 2>&1 & **Explanation:** • ./backup.sh → Runs the script. • > → Redirects **standard output** to backup.log. • 2>&1 → Redirects **error output** to the same file (so you capture everything). • & → Runs the process in the **background**. B. How can you check the contents of backup.log while the job is running in the background? (1 marks) *tail -f backup.log* **Explanation:** • tail -f → Continuously monitors and displays new lines as they are written to the log. • You can stop monitoring anytime with **Ctrl + C**.	CO3,CO3	Challenging - - A
37	4.1	You are the system administrator for a small company that handles sensitive financial data for its clients. Over the past few weeks, there have been multiple instances of system crashes, causing data loss on some machines. Management is concerned about the potential risks of not having regular backups in place. After reviewing the situation, you are asked to explain why it is essential for the company to implement regular data backups. (1 marks) ✅ **To prevent data loss and ensure business continuity.** • Regular backups protect sensitive financial data from accidental loss due to system failures, human errors, or cyberattacks.	CO4,CO4	Easy - - A

		• They allow the company to quickly recover important files and maintain trust with clients, avoiding costly downtime and reputational damage. . In your explanation, outline two possible scenarios where the company could lose important data, and explain how having backups could mitigate these risks. (2 marks) 🎬 **Scenario 1: Hardware Failure (e.g., hard drive crash)** ● **Risk:** Sudden disk failure can erase all stored client financial records. ● **Backup Benefit:** Regular backups allow recovery of the latest data from backup storage, minimizing data loss and avoiding disruptions to client services. 🎬 **Scenario 2: Cyberattack (e.g., ransomware attack)** ● **Risk:** Attackers encrypt or delete critical financial data, making it inaccessible. ● **Backup Benefit:** With secure, offline backups, the company can restore clean copies of the data without paying ransom, ensuring business recovery and data integrity.		
38	4.1	Your company has just adopted a new data backup strategy to improve its disaster recovery procedures. The IT department is deciding between a full backup strategy and an incremental backup strategy, aiming to balance data security and storage efficiency. A. You are asked to define what an "Incremental Backup" is and how it differs from a "Full Backup" in terms of data storage and recovery time. (2 marks) 🎬 **Incremental Backup:** ● ✅ **Definition:** An incremental backup copies **only the data that has changed or been added since the last backup** (whether full or incremental). ● ✅ **Storage:** ○ Uses **less storage space** since it only saves new/changed data. ○ Faster to perform daily backups. ● ✅ **Recovery Time:** ○ Recovery can be **slower**, as you need to restore the last full backup **plus** all subsequent incremental backups in sequence. 🎬 **Full Backup:** ● ✅ **Definition:** A full backup copies **all data**, regardless of changes. ● ✅ **Storage:** ○ Requires **more storage space**, as it saves everything each time. ● ✅ **Recovery Time:** ○ **Faster recovery**, as all the data is in one complete backup set. B. Additionally, explain the advantages of using the "3-2-1" backup strategy for a growing organization and how it could optimize both data retention and recovery. (1 marks) **3-2-1 Backup Strategy:** ● Keep **3 copies** of your data. ● Store on **2 different media types** (e.g., hard drive + cloud or tape). ● Keep **1 copy offsite** (e.g., cloud storage or remote location). **Advantages:** ● ✅ **Data Redundancy:** Even if one copy fails, others are safe. ● ✅ **Disaster Recovery:** If on-site disaster (fire, flood) occurs, offsite copy ensures recovery. ● ✅ **Optimized Retention:** Balances local fast recovery and offsite long-term safety, perfect for growing organizations handling increasing data volumes.	CO4,CO4	Medium - - A

39	4.1	The company is looking to choose the most suitable backup medium for storing their data backups. Options include external hard drives, cloud storage, and magnetic tape. The data size is growing rapidly, and a reliable backup solution is needed to ensure data integrity. • From the available options, list three common backup mediums that could be used for the company's data backup strategy. Discuss the pros and cons of each in terms of cost, accessibility, and security. (2 marks) **External Hard Drives** • **Pros:** • ✅ Low upfront cost. • ✅ Easy to use and fast data transfer. • ✅ Good for local quick backups. • **Cons:** • ❌ Physical damage risk (fire, water, theft). • ❌ Limited scalability — as data grows, you'll need more drives. • ❌ Needs manual management for offsite storage. **Cloud Storage** • **Pros:** • ✅ Highly scalable — grows with data needs. • ✅ Accessible from anywhere with internet access. • ✅ Built-in encryption and redundancy by providers. • **Cons:** • ❌ Ongoing subscription cost. • ❌ Dependent on internet speed and availability. • ❌ Potential concerns over data privacy (need to ensure trusted provider). **Magnetic Tape** • **Pros:** • ✅ Very cost-effective for large, long-term archival storage. • ✅ Good durability (can last 30+ years if stored properly). • ✅ Secure offline storage — less risk from cyberattacks. • **Cons:** • ❌ Slow access and retrieval time. • ❌ Requires special hardware (tape drives). • ❌ Less convenient for frequent backups. Considering the rapid data growth, what key factors should be prioritized when selecting the most suitable backup medium for long-term use? (1 marks) **Scalability:** Can it handle rapid data growth efficiently? ✅ **Durability & Longevity:** Will the medium remain reliable for years? ✅ **Cost-effectiveness:** Balance between upfront cost and long-term maintenance. ✅ **Accessibility & Recovery Speed:** How fast and easily can you retrieve data when needed? ✅ **Security:** Protection from both cyber threats and physical damage.	CO4,CO4	Challenging - - A

40	4.1	The IT department is working on reducing the amount of storage space required for backing up large files. They are considering compressing files before creating backups to optimize storage usage and improve transfer speeds. A. You are tasked with explaining why it is important to compress files before backing them up, especially when dealing with large datasets. (1 marks) ● Reduces storage space needed, especially for large datasets. ● Speeds up data transfer during backup and restore operations. ● Lowers storage costs by optimizing space usage. ● Improves efficiency of data management and archiving. . There is a discussion on using different compression tools such as ZIP and TAR. What are the benefits and drawbacks of each method? How would you decide which one to use for various types of backup data? (2 marks) **ZIP:** ● ✅ Combines compression and archiving in one step. ● ✅ Widely supported across all platforms (Windows, macOS, Linux). ● ✅ Easy to use and extract files individually. ● ❌ Limited in preserving file permissions and timestamps. ● ❌ Compression ratio is good but not always the best. **TAR (with GZIP/BZIP2):** ● ✅ Better compression ratio, especially with BZIP2. ● ✅ Preserves file permissions, ownerships, and timestamps. ● ✅ Ideal for large datasets and Linux/UNIX systems. ● ❌ TAR alone does not compress (needs GZIP/BZIP2). ● ❌ Windows users may need extra tools to extract. **Decision Points:** ● ✔️ Choose **ZIP** for smaller backups, cross-platform use, or when simplicity matters. ● ✔️ Choose **TAR + GZIP/BZIP2** for large backups, system files, or when preserving metadata is important.	CO4,CO4	Medium - - A
41	4.1	Your company has recently expanded its network and is now using both IPv4 and IPv6 addressing. The networking team has been working on configuring the DNS to ensure proper domain resolution across the entire network. A. Explain what an IP address is and why it is crucial for the company's internal and external communications within the network. How does an IP address facilitate communication between different devices? (1 marks) 🍫 **IP address** (Internet Protocol address) is a unique identifier assigned to each device connected to a network. 🍫 It is essential for identifying devices and enabling data to be sent and received correctly over networks (internal or external). 🍫 **Facilitates communication** by ensuring that data packets reach the correct destination device through routing. 🍫 Devices use IP addresses to locate each other, just like home addresses in postal services. B. The IT team also needs to ensure that users can access websites using domain names instead of IP addresses. Describe how DNS (Domain Name System) works and how it supports internet browsing. (2 marks) ● **DNS (Domain Name System)** translates human-friendly domain names (like *example.com*) into IP addresses that computers understand. ● When a user enters a website name, the DNS server checks its records to find the corresponding IP address. ● The browser then uses this IP address to connect to the web server and load the website. ● **Supports browsing** by making it easier for users to remember domain names instead of complex numeric IP addresses. ● **Enhances efficiency** through caching, so frequently visited sites load faster. ● **Supports both IPv4 and IPv6**, ensuring smooth operation as networks expand.	CO4,CO4	Medium - - A

42	4.1	You have been asked to troubleshoot network connectivity issues between two servers in your company's network. The servers can communicate with each other intermittently, and the network seems to drop connection sporadically. A. You decide to use the ifconfig command to check the network interfaces on one of the servers. What information will ifconfig display, and how can this help you diagnose connectivity issues? (1 marks) ✒ **Displays network interface details** like IP address, MAC address, netmask, broadcast address. ✒ **Shows interface status** (e.g., UP, DOWN, or RUNNING). ✒ **Indicates packet statistics** (transmitted, received, errors, dropped packets). ✒ Helps identify issues like: ● ✖ Interface down (needs to be brought up). ● ⚠ IP misconfiguration (wrong subnet/gateway). ● 🚫 Errors or dropped packets indicating hardware or cable issues. B. You then use the ping command to test connectivity between the servers. Explain how ping works and how you would interpret the results when diagnosing network issues. (2 marks) ● **ping sends ICMP Echo Request packets** to a target IP and waits for Echo Reply. ● Measures **round-trip time (latency)** and **packet loss**. ● Useful for testing if the remote server is reachable and how stable the connection is. **Interpreting Results:** ● ✅ **Replies received with low latency** → Connection is good. ● ⚠ **High latency or timeouts** → Possible congestion or network delay. ● ❌ **Packet loss** → Network instability, faulty cables, or overloaded network devices. ● ❌ **100% packet loss / no replies** → Server down, wrong IP, or firewall blocking traffic.	CO4,CO4	Medium - - A
43	4.1	You have been asked to troubleshoot network connectivity issues between two servers in your company's network. The servers can communicate with each other intermittently, and the network seems to drop connection sporadically. A. You are also asked to use the netstat command to monitor network connections on the server. What kind of information does netstat provide, and how can it be helpful in identifying performance bottlenecks or suspicious activity? (2 marks) **A. Information provided by netstat and its usefulness (2 marks)** ● **Displays active network connections** (TCP, UDP). ● **Shows listening ports** and associated processes. ● **Reveals network statistics**, like the number of packets sent/received, errors, and dropped connections. ● **Identifies established, waiting, or closed connections**, helping to spot congestion or hanging connections. ● **Useful for detecting:** ○ 🔥 **Performance bottlenecks** (e.g., too many connections to one port). ○ ⚠ **Suspicious activity** (e.g., unknown external connections, port scanning, potential malware activity). ○ 🚦 **Traffic overload** on specific services causing intermittent connectivity issues. B. The team has asked you to verify the routing table on the server. How would you use the route command to check the routing table, and why is this important for diagnosing routing problems? (1 marks) ✒ **route (or ip route) displays the server's routing table**, showing destination networks, gateways, and interface associations. ✒ **Helps verify if correct routes are set** to reach the other server or external networks. ✒ **Important for diagnosing:** ● ❌ Missing or incorrect routes causing packet misdirection. ● ⚠ Conflicting routes leading to intermittent connection drops. ● ✅ Ensures traffic is sent via the correct network paths.	CO4,CO4	Medium - - A

44	4.1	You are a Linux system administrator and need to send a quick message to another user on the same system to inform them of upcoming maintenance. The other user is not currently available to respond immediately, but you want to leave them a message. A. Using the write command, describe how you would communicate with the other user and provide an example of how the system would handle the message. (2 marks) **Using write command to communicate (2 marks)** ● **Purpose:** write allows you to send a message to another user logged into the same system terminal. ● **Command structure:** write [username] [terminal] Example: write alice pts/2 ● **Usage:** ○ Start typing your message, press Enter after each line. ○ To end the message, press **Ctrl + D** (EOF). ● **System handling:** ○ The message appears directly on the target user's terminal in real time. B. Considering the system is used by multiple users, what security concerns would you need to address before sending messages to others, especially if sensitive data could be involved? (1 marks) ● **Message visibility:** Other users on shared terminals might see the message. ● **Sensitive data risk:** Avoid sending confidential information, as messages are not encrypted. ● **Permissions:** Ensure **mesg** permission allows writing (mesg y), but users can restrict it (mesg n). ● **Good practice:** Use secure communication methods (e.g., encrypted emails or secure messaging tools) for sensitive info.	CO4,CO4	Challenging - - A
45	4.1	You are a system administrator working on a Linux server, and you need to compress a directory named project_files which contains various sub directories and files. The goal is to create a compressed archive in .tar.gz format to save space and share it with other team members. A. What command would you use to compress the project_files directory into a .tar.gz archive, and what will be the result of using this command? Provide the command along with a brief explanation. (3 marks) **A. Command to compress project_files into a .tar.gz archive (3 marks)** ● **Command:** tar -czvf project_files.tar.gz project_files/ ● **Explanation of options:** ○ tar — Archive utility command. ○ -c — Create a new archive. ○ -z — Compress the archive using gzip (.gz format). ○ -v — Verbose mode, shows the progress of files being added. *(Optional, but helpful.)* ○ -f — Specifies the filename of the archive (project_files.tar.gz). **Result:** ✅ Creates a compressed file named **project_files.tar.gz** in the current directory. ✅ The archive contains all files and subdirectories from **project_files/**. ✅ Useful for saving storage space and for easy sharing or transferring to team members.	CO4	Medium - - A

46	4.1	You are managing a Linux system, and you need to send an email from the command line using the built-in mail command. The email needs to be sent from a specific user account, and it should include both a subject and body. You also need to ensure the email is sent securely to a recipient within your organization. Explain the steps to send an email using the mail command in Linux, including how to specify the subject and message body. 1. **Basic Command Structure:** echo "Email body text" | mail -s "Subject of the Email" recipient@example.com 2. **Explanation:** ○ echo "Email body text" — This is the content of the email body. ○ | — Pipes the message to the mail command. ○ mail — The command-line mail utility. ○ -s "Subject of the Email" — Specifies the subject line of the email. ○ recipient@example.com — The email address of the recipient. 3. **Sending from a Specific User:** ○ Use sudo -u [username] if you need to send as a specific user: echo "Maintenance scheduled at 10 PM" | sudo -u username mail -s "Server Maintenance Notification" recipient@example.com 4. **Sending Securely:** ○ To ensure secure sending, configure **mail** to use **SSL/TLS** via tools like **msmtp** or **sendmail** in the system's mail configuration. ○ Make sure the mail server (SMTP) you are relaying through is properly secured (authenticated and encrypted). 5. **Alternative: Send body from a file (Optional)** ○ If the message body is long, you can write it in a text file and send: mail -s "Subject here" recipient@example.com < /path/to/message.txt ✅ **Result:** ● An email will be sent with the specified subject and body, from the desired user account, to the recipient.	CO4	Easy - Applying - T
47	4.1	Describe how Network Address Translation (NAT) works to allow multiple devices on a private network to access the internet using a single public IP address. How would you configure NAT on a Linux server to achieve this? **How NAT Works:** 1. **Concept:** ○ **Network Address Translation (NAT)** allows multiple devices on a private network (using private IP addresses like 192.168.x.x) to share a single public IP address when accessing the internet. 2. **Process:** ○ When a device sends traffic to the internet, the NAT-enabled router (or server) **replaces the private IP address** in the packet with its **public IP address**. ○ The router keeps track of this connection in a NAT table (mapping internal IP:Port to external IP:Port). ○ When responses come back from the internet, the router uses the NAT table to **route the traffic back to the correct internal device**. 3. **Benefits:** ○ Conserves public IPv4 addresses. ○ Adds a layer of security by hiding internal IPs. ○ Allows seamless internet access for many devices behind one IP. **How to Configure NAT on a Linux Server:** ✅ **Step 1: Enable IP forwarding** echo 1 > /proc/sys/net/ipv4/ip_forward	CO4	Medium - Understanding - T

		• This allows the Linux server to forward packets between interfaces. ✅ **Step 2: Apply NAT using iptables** • Assuming: ○ Internal network interface: eth1 (connected to LAN) ○ External network interface: eth0 (connected to the internet) iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE • **Explanation:** ○ -t nat: Use the NAT table. ○ -A POSTROUTING: Apply rule after routing decision. ○ -o eth0: Outgoing traffic through the external interface. ○ -j MASQUERADE: Dynamically translate source IP to public IP. ✅ **Step 3: Save iptables rules** • To persist after reboot, save the rules: iptables-save > /etc/iptables/rules.v4 ✅ **Step 4: Ensure firewall rules allow traffic** **Result:** • Devices on the internal network can now access the internet through the Linux server using its single public IP address!		
48	4.1	You are a Linux system administrator tasked with managing disk space on a server that stores a large number of log files and backups. The server's disk space is running low, and you need to compress some of these files to save space while ensuring they can be easily extracted when needed. You are considering using different compression tools available in Linux. What is the difference between the .tar, .gz, .bz2, and .xz file formats in terms of compression and usage? **1. .tar (Tape Archive)** • **Purpose:** ○ Archiving only (combines multiple files into one). ○ **No compression** by itself. • **Usage:** ○ Used to bundle many files/directories together. • **Common Command:** tar -cf archive.tar folder/ **2. .gz (Gzip Compression)** • **Purpose:** ○ Compression format that reduces file size. • **Compression Speed:** ○ **Fast compression** and decompression. • **Compression Ratio:** ○ Moderate compression. • **Usage:** ○ Common for quick compression needs. • **Common Command:** gzip file.txt gunzip file.txt.gz **3. .bz2 (Bzip2 Compression)** • **Purpose:** ○ Higher compression ratio than .gz. • **Compression Speed:** ○ **Slower compression**, but decompression is fast.	CO4	Medium - Understanding - T

		● **Compression Ratio:** ○ Better than .gz, useful for saving more space. ● **Usage:** ○ When saving space is a bigger priority over speed. ● **Common Command:** bzip2 file.txt bunzip2 file.txt.bz2 **4. .xz (XZ Compression)** ● **Purpose:** ○ Very high compression ratio. ● **Compression Speed:** ○ **Slowest compression**, but decompression is fairly fast. ● **Compression Ratio:** ○ Best among .gz, .bz2, and .xz. ● **Usage:** ○ Ideal for archiving large files when maximum space saving is required. ● **Common Command:** xz file.txt unxz file.txt.xz		
49	5.1	You need to install a new application on your Linux system using the Red Hat Package Manager (RPM). The application is provided as an .rpm file. A. How would you install an .rpm package on a Red Hat-based system? (1 marks) **Command:** sudo rpm -ivh package_name.rpm **Explanation:** ● rpm — The Red Hat Package Manager command. ● -i — Install the package. ● -v — Verbose output (shows detailed installation progress). ● -h — Display hash marks (#) to show progress. B. What command would you use to ensure the package is properly installed? (1 marks) ● **Command:** rpm -q package_name ● **Explanation:** ○ rpm -q — Query the package. ○ package_name — The name of the package you installed. ○ **Result:** If the package is installed, it will return the version of the installed package. C. If there are any dependencies missing, how would you address that issue during the installation? (1 marks) **Command to address dependencies:** sudo yum install package_name.rpm **Explanation:** ● yum — The package manager for Red Hat-based systems. ● install — Installs the .rpm package. ● If dependencies are missing, yum will automatically try to resolve them by downloading and installing the required dependencies from enabled repositories.	CO5,CO5,CO5	Challenging - - A

50	5.1	The company's servers are running an outdated version of a critical software package. You are tasked with upgrading the package to the latest version using RPM. A. What command would you use to upgrade a package using RPM? (1 marks) sudo rpm -Uvh package_name.rpm Explanation: rpm — The Red Hat Package Manager command. -U — Upgrade the package to the latest version. -v — Verbose mode to show detailed output during the process. -h — Display progress with hash marks (#). B. How can you check if the upgrade was successful? (1 marks) rpm -q package_name Explanation: rpm -q — Query the package. package_name — The name of the package you upgraded. Result: If the upgrade was successful, the command will return the version of the upgraded package. C. If the upgrade fails, what steps would you take to troubleshoot the issue?(1 marks) 1. **Check for error messages:** Review the output of the rpm -Uvh command for any errors or warnings. These may provide clues about why the upgrade failed. 2. **Verify dependencies:** If dependencies are missing, use yum to check for and install them: sudo yum install package_name.rpm yum will automatically resolve and install any missing dependencies. 3. **Check if the package is already installed:** If the package is already at the latest version, the upgrade might not proceed. Verify the current version with: rpm -q package_name 4. **Log files:** Check system logs like /var/log/yum.log or /var/log/messages for more detailed error information related to the package installation or upgrade process.	CO5,CO5,CO5	Medium - - A
51	5.1	You are required to remove an obsolete package from your system to free up space. The package was installed via RPM. A. What command would you use to uninstall a package installed via RPM? (1 marks) sudo rpm -e package_name Explanation: rpm -e — The -e option stands for "erase" and is used to remove an installed package. package_name — The name of the package you want to remove. B. How can you verify that the package has been completely removed from the system? (1 marks) rpm -q package_name Explanation: rpm -q — Query the package. If the package has been successfully removed, the command will return something like "package_name is not installed" or no output. C. What potential risks or issues should you be aware of before removing a package? (1 marks) 1. **Dependency Issues:** ○ The package may be required by other installed software. Removing it could break other applications that depend on it.	CO5,CO5,CO5	Medium - - A

		○ **Solution:** Use the yum remove package_name command, as it will check and notify you of any dependencies before removing the package. 2. **System Stability:** ○ Some packages might be critical for system functionality, like libraries or kernel-related packages. ○ **Solution:** Ensure that the package is not a critical system component. 3. **Configuration Files:** ○ Sometimes, removing a package doesn't delete its configuration files. These files might take up space, or you might want to retain them for later use. ○ **Solution:** Use the --purge option if available (in case of yum), or manually check for leftover configuration files. 4. **Data Loss:** ○ If the package stores or manages important data, make sure to back up the data before removal.		
52	5.1	You need to schedule a task that runs every day at midnight and another task that runs once at a specified time using cron and atd. A. How would you use the cron daemon to schedule a task to run daily at midnight?(1 marks) 1. Open the crontab file for editing: crontab -e 2. Add the following line to schedule the task: 0 0 * * * /path/to/your/script.sh **Explanation:** ● 0 0 * * * — This means the task will run at 00:00 (midnight) every day. ● /path/to/your/script.sh — Replace this with the actual command or script you want to run at midnight. B. What is the difference between cron and atd, and when would you use each of them? (2 marks) **Cron:** ● **Purpose:** ○ **Cron** is used for scheduling recurring tasks. ● **Usage:** ○ Ideal for tasks that need to run on a regular basis, such as daily, weekly, or monthly. ○ Examples: Running backups every day at midnight, checking disk space every week. ● **Command:** ○ Managed through crontab files. ○ **When to use:** If you need to automate tasks that should repeat at regular intervals (e.g., daily, weekly, monthly). **atd (At Daemon):** ● **Purpose:** ○ **atd** is used for scheduling one-time tasks. ● **Usage:** ○ Ideal for tasks that should run once at a specified time in the future. ○ Example: Running a task once tomorrow at 2:00 PM or in an hour. ● **Command:** ○ Managed through at command to schedule a one-time task. ○ **When to use:** If you need to run a task once at a specific time and don't need it to recur.	CO5,CO5	Challenging - - A

53	5.1	You are tasked with managing network services on a Linux server. The server currently uses inetd to manage several network services. However, due to security concerns and the need for better control over each service, you decide to switch from inetd to xinetd, which provides enhanced features like access control, logging, and more granular service management. A. What are the key steps to switch from using inetd to xinetd for managing a service on the server? Provide the necessary commands and explain what each command does. (3 marks) **Step 1: Install xinetd** ● **Command:** sudo yum install xinetd **Step 2: Stop and disable inetd** ● **Commands:** sudo systemctl stop inetd sudo systemctl disable inetd **Step 3: Configure xinetd for the service** ● **Command to copy and edit config:** sudo cp /etc/inetd.conf /etc/xinetd.d/your_service sudo nano /etc/xinetd.d/your_service **Step 4: Start and enable xinetd** ● **Commands:** sudo systemctl start xinetd sudo systemctl enable xinetd **Step 5: Verify xinetd is running** ● **Command:** sudo systemctl status xinetd	CO5	Medium - - A
54	5.1	You are managing a Linux-based server, and during a routine system check, you notice that the server is experiencing a significant performance degradation. After reviewing the system logs, you see that the kthreadd daemon is consuming unusually high CPU resources. What is the role of the kthreadd daemon in the Linux kernel, and why is it important for system performance? The **kthreadd** daemon is a kernel thread in Linux responsible for creating and managing other kernel threads. It is the initial thread that is spawned when the system boots, and its primary role is to handle the creation of other threads that are required by the kernel for various tasks, such as managing hardware, handling system calls, or performing background system functions. **Role of kthreadd:** ● **Thread Management:** It is responsible for creating kernel threads that execute specific tasks within the kernel, such as I/O operations, memory management, and device management. ● **System Initialization:** It initializes and manages various internal kernel threads during boot time, which handle different background tasks. **Importance for System Performance:** ● **Efficiency in Task Management:** The **kthreadd** process allows the kernel to offload background tasks from the main user space, ensuring that the system remains responsive even under heavy loads. ● **Critical for Kernel Operations:** If the kthreadd daemon consumes too many resources, it could indicate that there is a problem with the creation or management of kernel threads, which could lead to system instability and performance degradation. **High CPU Usage by kthreadd:** ● **Possible Causes:** 1. **Kernel Bugs:** Issues with kernel threads or bugs in specific kernel modules might cause excessive CPU usage. 2. **Resource Exhaustion:** If the system is running out of resources like memory or I/O bandwidth, kthreadd may end up managing numerous kernel threads to try to compensate. 3. **Malfunctioning Drivers:** Faulty or misconfigured device drivers may create an unusually high number of threads.	CO5	Challenging - Analysing - A

55	5.1	A company uses a shared printer, and the cupsd daemon is responsible for managing all print jobs. Users report that print jobs are stuck in the queue and are not being processed even though the printer itself is operational. What is the primary role of the cupsd daemon in managing printing tasks on the server and how would you troubleshoot the issue of stuck print jobs in the queue? **Primary Role of the cupsd Daemon:** The **cupsd** daemon is the **CUPS (Common UNIX Printing System)** server daemon responsible for managing printing tasks on Linux and UNIX-like systems. Its primary role includes: 1. **Managing Print Jobs:** It controls and processes print jobs from users, sending them to the appropriate printer. 2. **Queue Management:** It manages the print queue, ensuring that jobs are processed in order, and maintains the status of each job. 3. **Printer Configuration and Control:** It handles printer setup, configuration, and communication with the printer hardware via the CUPS interface. 4. **Job Scheduling:** It schedules print jobs based on the printer's availability and resources. **Troubleshooting Stuck Print Jobs:** When print jobs are stuck in the queue, the issue could be with the **cupsd** daemon, the print job itself, or printer communication. Here's how you can troubleshoot the issue: **1. Check the status of cupsd:** ● **Command:** sudo systemctl status cups ● **Explanation:** ○ This checks if the **cupsd** daemon is running properly. If it's not running, start it with: sudo systemctl start cups **2. Check the print queue:** ● **Command:** lpq ● **Explanation:** ○ This command shows the current print queue. Check if there are jobs stuck or not processing. ○ If jobs are stuck, try to cancel them using: cancel <job-id> **3. Restart the cupsd daemon:** ● **Command:** sudo systemctl restart cups ● **Explanation:** ○ Restarting the **cupsd** daemon can clear temporary issues with the print job queue and the printer management system. **4. Check for Printer Issues:** ● **Command:** lpstat -p ● **Explanation:** ○ This shows the status of printers. Check if the printer is in a "paused" or "disabled" state. If so, resume it with: php-template CopyEdit cupsenable <printer-name> **5. Review CUPS Logs:** ● **Command:** sudo tail -f /var/log/cups/error_log ● **Explanation:** ○ Review the CUPS logs for any error messages or warnings related to the print jobs. This can help identify issues with the printer, permissions, or communication.	CO5	Medium - Evaluating - A

		6. Check for Resource Issues: ● Ensure the server has enough resources (CPU, memory) to process print jobs. Check system resource usage with: top **7. Clear the Print Queue:** ● If the jobs are stuck permanently, clearing the print queue manually may help: sudo cancel -a This cancels all jobs in the print queue.		

56	5.1	Users are unable to access files shared through the nfsd daemon on a network file server. The system administrator suspects that the nfsd daemon is not running correctly, and there are no errors displayed when attempting to access the server. What steps would you take to check the status of the nfsd daemon and ensure it is running properly? **1. Check if the nfsd daemon is running:** ● **Command:** sudo systemctl status nfs-server ● **Explanation:** ○ This command checks the status of the **nfsd** (NFS server) service. If it's not running, you'll need to start it. If it is running, the output will show the current state (active or inactive) and whether it is enabled on boot. **2. Start the nfsd daemon if it's not running:** ● **Command:** sudo systemctl start nfs-server ● **Explanation:** ○ This command starts the NFS server if it is not running. Ensure that the service is started properly. **3. Enable the nfsd daemon to start on boot:** ● **Command:** sudo systemctl enable nfs-server ● **Explanation:** ○ This command ensures that the **nfsd** daemon will start automatically when the server is rebooted. **4. Check if the NFS service is running:** ● **Command:** sudo systemctl status nfs ● **Explanation:** ○ This command checks the status of the NFS service, which is responsible for managing the network file system. If this service is not running, NFS clients will not be able to access shared files. **5. Check for active NFS exports:** ● **Command:** sudo exportfs -v ● **Explanation:** ○ This command shows the list of directories that are currently being exported by the NFS server. Ensure that the directories are correctly exported and available to clients. **6. Check NFS daemon logs for errors:** ● **Command:** sudo journalctl -u nfs-server ● **Explanation:** ○ This command shows the logs for the **nfsd** daemon. Review the logs for any error messages or warnings that might indicate why the daemon is not functioning properly.	CO5	Medium - Applying - A

		7. Verify firewall settings: ● **Command:** sudo firewall-cmd --list-all ● **Explanation:** ○ Ensure that the firewall is not blocking NFS traffic. If necessary, allow NFS traffic by running: sudo firewall-cmd --zone=public --add-service=nfs --permanent sudo firewall-cmd --reload **8. Verify NFS version compatibility:** ● **Command:** nfsstat -v ● **Explanation:** ○ This command checks the NFS version being used. If the clients are using a different version than the server, there could be compatibility issues. Ensure both the client and server are using compatible NFS versions. **9. Restart the NFS server:** ● **Command:** sudo systemctl restart nfs-server ● **Explanation:** ○ Restarting the NFS server can resolve many issues related to service failures or resource allocation. **10. Check client-side access:** ● **Command (on client machine):** sudo mount -v <nfs-server-ip>:/shared_directory /mnt/nfs ● **Explanation:** ○ Ensure the NFS share is accessible from the client machine. This command mounts the NFS share on the client and helps verify if the server is reachable and if the directory is properly exported.		
57	5.1	Your email server's sendmail daemon is working fine for outgoing emails, but incoming emails are not being received. Upon checking the logs, you find no errors but notice that the daemon seems unresponsive. What role does the sendmail daemon play in email delivery, both for incoming and outgoing messages and what steps would you take to troubleshoot the issue where only incoming emails are not being delivered? The **sendmail** daemon is responsible for handling **email delivery** on a Linux-based email server. It plays a crucial role in both **sending** and **receiving** emails: 1. **Outgoing Emails:** ○ When a user sends an email, **sendmail** processes the email, determines the destination server (via DNS lookup), and then sends the email using the SMTP protocol. 2. **Incoming Emails:** ○ When an email is received, **sendmail** listens on port **25** for incoming email traffic (SMTP). It then processes the message, verifies the destination, and places it in the recipient's mailbox. **If only outgoing emails are working**, the problem might be related to how **sendmail** is handling incoming messages. **Steps to Troubleshoot Incoming Email Delivery:** 1. **Check the sendmail status:** ○ Ensure the **sendmail** daemon is running correctly, particularly the part that listens for incoming mail on port **25**. ○ **Command:** sudo systemctl status sendmail ○ If it's not running, try restarting it: sudo systemctl restart sendmail 2. **Verify that sendmail is listening on port 25 (SMTP):** ○ **Command:** sudo netstat -tulnp | grep :25 ○ This command checks if **sendmail** is properly listening on port **25** for incoming connections. If there's no output, the daemon is not listening on that port.	CO5	Challenging - Applying - A

3. **Check firewall settings:**
    - Ensure that your firewall is not blocking incoming connections to **port 25** (SMTP).
    - **Command (for firewalld):**

```
sudo firewall-cmd --zone=public --add-port=25/tcp --permanent
sudo firewall-cmd --reload
```

    - **Command (for iptables):**

```
sudo iptables -A INPUT -p tcp --dport 25 -j ACCEPT
sudo service iptables save
```

4. **Check sendmail logs:**
    - Review the sendmail logs to see if there are any errors related to incoming emails. Logs are typically stored in:

```
/var/log/maillog
```

    - **Command:**

```
sudo tail -f /var/log/maillog
```

    - Check for any indications of why incoming connections might be rejected or not processed.
5. **Check DNS settings (MX records):**
    - Ensure that the server's **MX (Mail Exchange) records** are properly configured and pointing to the correct server for handling incoming emails.
    - **Command:**

```
dig +short MX <your-domain-name>
```

    - The output should show the correct mail server (hostname and IP) for the domain.
6. **Verify SELinux or AppArmor settings:**
    - If SELinux or AppArmor is enabled, they might be blocking incoming email connections.
    - **Check SELinux status:**

```
sudo getenforce
```

    - If SELinux is enforcing, you might need to adjust the policies:

```
sudo setsebool -P httpd_can_sendmail 1
sudo setsebool -P sendmail_can_network 1
```

7. **Check for DNS Resolution Issues:**
    - Ensure that the server can resolve the IP addresses of incoming mail servers correctly. If the server cannot resolve domain names to IP addresses, it won't be able to process incoming email.
    - **Command:**

```
nslookup <remote-mail-server>
```

8. **Test Mail from an External Source:**
    - Try sending an email from an external email account (e.g., Gmail, Yahoo) to your server's email address to confirm if the issue is isolated to certain senders or if all incoming mail is affected.
9. **Examine Spam Filters/Blacklists:**
    - Ensure that your server's IP address is not listed in any blacklists (which might prevent incoming mail). Use online tools to check if your server's IP is on any blacklists.
    - Check any spam filters or anti-spam software running on the server that could be rejecting or blocking incoming emails.
10. **Verify Mail Queue:**
    - Sometimes, emails can get stuck in the mail queue. Check the mail queue to see if there are any incoming messages that are not being processed.
    - **Command:**

```
mailq
```

Summary of Troubleshooting Steps:

1. Ensure **sendmail** is running using systemctl status sendmail.
2. Check if sendmail is listening on port **25** using netstat.
3. Verify firewall settings to ensure port **25** is open.
4. Review sendmail logs in /var/log/maillog.
5. Check DNS settings and MX records.

		6. Verify SELinux or AppArmor settings. 7. Test incoming mail from an external source. 8. Check for blacklisting or spam filters. 9. Examine the mail queue for stuck emails.		
58	5.1	After performing a system update, the server fails to boot, and you suspect that the systemd daemon, which is responsible for managing system services, may have encountered an issue during startup. What is the role of the systemd daemon in the boot process of a Linux system and how would you approach troubleshooting the boot failure related to systemd? What logs or tools would you use? **Role of systemd in the Boot Process:** ● **systemd** is the init system and service manager for Linux. ● During boot, systemd is the first userspace process (PID 1) that initializes the system: o Mounts file systems. o Starts essential system services (networking, logging, etc.). o Handles dependencies between services. o Manages targets (like runlevels in old systems). o Provides parallel service startup for faster boot times. If systemd has issues, the entire boot process can fail or hang. --- **Approach to Troubleshoot Boot Failure Related to systemd:** 1. **Boot into Emergency or Rescue Mode:** o When you see the GRUB menu, edit the boot entry: ▪ Add systemd.unit=rescue.target or systemd.unit=emergency.target at the end of the kernel line. o **Rescue Mode**: Minimal environment, basic services. o **Emergency Mode**: Even more minimal, no services, just root shell. 2. **Check systemd Boot Logs:** o Use journalctl to view logs related to boot. o **Command (after booting into rescue/emergency mode):** journalctl -xb o Look for errors (failed units, dependency issues). 3. **Check Failed Services:** o List failed services to see what caused the boot hang. systemctl --failed 4. **Use systemd-analyze (if booting is possible):** o This tool shows which parts of the boot process took the longest. systemd-analyze systemd-analyze blame 5. **Check for Dependency Loops:** o Sometimes, systemd units create a deadlock. o View dependency tree: systemctl list-dependencies 6. **Review Recent Changes:** o If the issue started after updates, check what was recently installed or upgraded: rpm -qa --last | head 7. **Rebuild initramfs (if corrupted):** o Initramfs might be corrupted during an update. dracut --force 8. **Rollback Updates (if necessary):** o Use package manager to downgrade problematic packages. **Summary of Useful Logs/Tools:** **Tool/Log** — **Purpose** journalctl -xb — View detailed boot logs systemctl --failed — List failed units	CO5	Challenging - Applying - A

		systemctl list-dependencies systemd-analyze /var/log/messages or /var/log/syslog Boot in **rescue/emergency mode**	Check unit dependency issues Analyze boot performance General system logs Safe mode for repair		

59	5.1	You notice that several new devices on the network are unable to obtain an IP address from the DHCP server, even though other devices are successfully getting IPs. The dhcpd daemon seems to be running but there may be a misconfiguration. What is the primary function of the dhcpd daemon in managing network configurations and how would you troubleshoot the issue where new devices are not receiving an IP address from the dhcpd daemon?
		
		Primary Function of dhcpd Daemon:
		● The dhcpd (DHCP Daemon) automatically assigns **IP addresses**, **subnet masks**, **default gateways**, and **DNS servers** to devices on the network.
		● It ensures devices get proper network configurations dynamically, without manual setup.
		● It manages the **IP address pool** and lease times.
		
		Troubleshooting Steps for New Devices Not Getting IPs:
		1. **Check DHCP Service Status:** Make sure dhcpd is running:
		
		systemctl status dhcpd
		2. **Check DHCP Configuration File (/etc/dhcp/dhcpd.conf):**
		○ Ensure there's an available IP pool for new devices.
		○ Confirm subnet and range settings:
		
		subnet 192.168.1.0 netmask 255.255.255.0 {
		range 192.168.1.100 192.168.1.200;
		option routers 192.168.1.1;
		option domain-name-servers 8.8.8.8;
		}
		○ Check if **MAC address filtering** is enabled (restricting new devices).
		3. **Check for IP Pool Exhaustion:**
		○ Ensure you have free IPs:
		▪ Look at the lease file:
		
		cat /var/lib/dhcpd/dhcpd.leases
		4. **Check Network Connectivity:**
		○ Ensure new devices are on the correct network/VLAN.
		○ Test with:
		
		ping <dhcp-server-ip>
		5. **Inspect DHCP Logs:**
		○ View logs for clues:
		
		journalctl -u dhcpd
		or
		
		tail -f /var/log/messages
		6. **Verify Firewall & SELinux:**
		○ Make sure ports (UDP 67 & 68) are open:
		
		firewall-cmd --list-all
		○ Check SELinux (temporarily set to permissive to test):
		
		setenforce 0
		7. **Restart the DHCP Service:**
			CO5	Medium - Analysing - A

		systemctl restart dhcpd **Summary Table:** **Step** — **Action** ✅ Check service — systemctl status dhcpd ✅ Review config — /etc/dhcp/dhcpd.conf ✅ Check IP pool — /var/lib/dhcpd/dhcpd.leases ✅ Inspect logs — journalctl -u dhcpd or /var/log/messages ✅ Check firewall/SELinux — Ensure ports 67/68 open ✅ Restart service — systemctl restart dhcpd		
60	5.1	You are tasked with managing a Linux server that uses systemd for service management. The server hosts a critical service, webapp.service, which is responsible for serving the company's website. Recently, you have received complaints that the website is intermittently down. • The webapp.service is running on the system, but sometimes it fails to start automatically after a reboot. • You notice that the service fails to start at boot, and sometimes it needs to be manually restarted using systemctl restart webapp.service. • After reviewing the system logs, you discover that the service crashes due to an issue with its configuration file. A. What command would you use to view the current status of webapp.service and identify if it's active or failed? (1 marks) To check the current status of webapp.service: systemctl status webapp.service B. How would you ensure that webapp.service is enabled to start automatically at boot time using systemctl? (1 marks) To enable the service to start automatically at boot: systemctl enable webapp.service C. After fixing the configuration issue, what command would you use to reload the service configuration and restart the service without rebooting the system? (1 marks) After fixing the configuration, to reload and restart the service: systemctl daemon-reload systemctl restart webapp.service	CO5,CO5,CO5	Challenging - Applying - A

Part B

#	Unit	Question	COS	Categorized
1	1.1	In Linux, directory management is a crucial task for system administrators and users to effectively organize and navigate the file system. Explain the key directory commands in Linux, focusing on their usage, syntax, and practical applications. In your answer, include the following: **1. Identifying the Current Directory Path** *(1 mark)* • **Command**: pwd • **Usage**: Displays the full path of the current working directory. **Syntax**: pwd **Example Output**:	CO1,CO1,CO1,CO1,CO1,CO1	Medium - - A

/home/user/Documents

2. Changing the Current Directory *(1 mark)*

- **Command**: `cd`
- **Usage**: Changes the current directory to another directory.

Syntax:
cd [directory_path]

Example:
cd /home/user/Desktop

3. Creating a Directory *(1 mark)*

- **Command**: `mkdir`
- **Usage**: Creates a new directory.

Syntax:
mkdir [directory_name]

Example:
mkdir Projects

4. Removing a Directory *(1 mark)*

- **Command**: `rmdir`
- **Usage**: Deletes an empty directory.

Syntax:
rmdir [directory_name]

Example:
rmdir Projects

- *(Use `rm -r` for non-empty directories)*

5. Listing the Contents of a Directory *(1 mark)*

- **Command**: `ls`
- **Usage**: Lists files and directories in the current directory.

Syntax:
ls [directory_name]

Example:
ls /home/user

6. Listing the Contents with File Access Permissions *(1 mark)*

- **Command**: `ls -l`
- **Usage**: Lists directory contents in long format, showing permissions, owner, size, and date.

		Syntax: ls -l [directory_name] **Example Output**: drwxr-xr-x 2 user user 4096 Apr 18 10:00 Documents -rw-r--r-- 1 user user 220 Apr 18 09:59 notes.txt		

2	1.1	You are working on a Linux server that contains a directory named projects with several text files of varying sizes and contents. Perform the following tasks using appropriate Linux file commands: A. Display the content of a file named report.txt. Ensure that you view the file's content without scrolling, and also handle cases where the content exceeds the terminal's window size. (1 marks) **1. Display the content of `report.txt` without scrolling** *(1 mark)* **Command**: less report.txt • **Usage**: Opens the file one screen at a time; allows navigation using arrow keys. Press `q` to quit. B. Using the head command, display the first 10 lines of the file logfile.txt (1 marks) **2. Display the first 10 lines of `logfile.txt` using `head`** *(1 mark)* **Command**: head logfile.txt C. Use the tail command to show the last 20 lines of the same file. (1 marks) **3. Show the last 20 lines of `logfile.txt` using `tail`** *(1 mark)* **Command**: tail -n 20 logfile.txt D. Copy the file project1.txt from the projects directory to a backup directory named backup_projects. Ensure that the file is copied correctly and verify the copy in the new directory. (1 marks) **4. Copy `project1.txt` to `backup_projects` and verify** *(1 mark)* **Command**: cp projects/project1.txt backup_projects/ ls backup_projects/ E. Identify a file named draft.txt that is no longer needed. Remove it from the projects directory. Verify that the file is deleted and cannot be found in the directory anymore. (1 marks) **5. Remove `draft.txt` and verify deletion** *(1 mark)* **Command**:	CO1,CO1,CO1,CO1,CO1,CO1	Challenging - - A

		rm projects/draft.txt ls projects/ • **Check**: Ensure `draft.txt` no longer appears in the listing. --- F. Move a file named summary.txt from the projects directory to a sub directory named archive. Also, rename the file to summary_old.txt while moving it. Verify both the file's location and the new name. (1 marks) **6. Move and rename `summary.txt` to `archive/summary_old.txt`** *(1 mark)* **Command**: mv projects/summary.txt projects/archive/summary_old.txt ls projects/archive/		
3	1.1	You are a system administrator tasked with ensuring that files on a Linux server are securely managed. The server contains a variety of files, and you must apply the appropriate file access permissions to protect sensitive data. Perform the following tasks related to securing files using File Access Permissions (FAPs) in Linux: A. You need to examine the access permissions of a file named project_report.txt located in the /home/user/projects/ directory. Use the appropriate command to view the file's permissions, and explain what each part of the output represents. (1 marks) **1. View permissions of `project_report.txt`** *(1 mark)* **Command**: ls -l /home/user/projects/project_report.txt • **Sample Output**: -rw-r--r-- 1 user user 1024 Apr 18 10:00 project_report.txt • **Explanation**: ○ `-`: Regular file ○ `rw-`: Owner can read & write ○ `r--`: Group can only read ○ `r--`: Others can only read ○ `1 user user`: Owner is *user*, group is *user* B. The file financial_data.csv located in /home/user/private/ needs to be modified so that only the owner can read and write to the file, and the group and others have no permissions. Change the file's permissions accordingly and explain the command you used. (1 marks) **2. Change permissions for `financial_data.csv` (owner: read & write only)** *(1 mark)* **Command**:	CO1,CO1,CO1,CO1,CO1	Challenging - - A

chmod 600 /home/user/private/financial_data.csv

- **Explanation**:
    - `6` (owner): read + write
    - `0` (group, others): no permission
    - **Result**: File is secured from other users.

C. The file public_image.jpg located in /home/user/images/ needs to be accessible by the owner for reading, writing, and executing, while the group should only have read and execute permissions, and others should have no access. Change the permissions using numeric representation (octal values). (1 marks)

3. Set permissions for `public_image.jpg` using octal notation *(1 mark)*

Command:

chmod 750 /home/user/images/public_image.jpg

- **Explanation**:
    - `7` (owner): read + write + execute
    - `5` (group): read + execute
    - `0` (others): no access

D. A script file named backup.sh located in /home/user/scripts/ is not executable by the owner. You are required to modify the permissions to allow the owner to execute the script. Describe how you would grant the owner execute permissions, and explain the impact of the changes. (1 marks)

4. Make `backup.sh` executable by the owner *(1 mark)*

Command:

chmod u+x /home/user/scripts/backup.sh

- **Explanation**:
    - `u+x`: Adds execute permission to the owner
    - **Impact**: Owner can now run the script as a command.

E. You are required to ensure that the directory /home/user/secure_folder/ is only accessible to the owner for reading, writing, and executing files, while the group and others have no access. After making the necessary changes, verify the directory's permissions. (2 marks)

5. Secure `secure_folder/` for owner-only access *(2 marks)*

Command:

chmod 700 /home/user/secure_folder/
ls -ld /home/user/secure_folder/

- **Explanation**:

		○ `7` (owner): read + write + execute ○ `0` (group, others): no access **Verification Output Example**: drwx------ 2 user user 4096 Apr 18 10:30 /home/user/secure_folder/		
4	1.1	Describe the process of managing files and directories in Linux using file manipulation commands, including common tasks like creating, moving, and deleting files. A. Explain how to create a new directory and provide the command. How can you create multiple directories at once? (2 marks) **1. Creating Directories** *(2 marks)* **Single Directory**: mkdir myfolder ○ This creates one directory named `myfolder`. **Multiple Directories at Once**: mkdir dir1 dir2 dir3 ○ This creates three directories in the current path. **Nested Directories (including parent folders if missing)**: mkdir -p parent/child/subchild ○ `-p` ensures that all parent directories are created as needed. B. How would you move a file to a new location, and what happens if the destination directory already contains a file with the same name? (1 marks) **2. Moving a File** *(1 mark)* **Command**: mv file1.txt /home/user/documents/ ○ Moves `file1.txt` to the specified directory. **Note**: If a file with the same name exists in the destination, it **will be overwritten without warning** (unless `-i` is used for interactive confirmation): mv -i file1.txt /home/user/documents/ C. Provide the commands for renaming a file and explain how these differ from moving files between directories. (1 marks) **3. Renaming a File** *(1 mark)* **Command**:	CO1,CO1,CO1,CO1	Challenging - - A

		mv oldname.txt newname.txt ○ Renames the file in the same directory. ● **Difference from Moving**: ○ When source and destination are in the same directory but have different names, `mv` renames the file. ○ When used with different paths, it moves the file. D. Describe the command and precautions you should take before using rm -rf to delete files and directories. (2 marks) **4. Deleting Files and Directories with `rm -rf` *(2 marks)*** **Command**: rm -rf foldername/ ○ `-r`: recursively removes directories and their contents ○ `-f`: forces deletion without confirmation ● **Precautions**: ○ Always **double-check the path** to avoid accidental deletion. ○ Never run `rm -rf /` or `rm -rf *` in a sensitive directory. Use `ls` first to confirm the target: ls foldername/		
5	1.1	Discuss file access permissions in Linux, including how to view, modify, and understand different types of permissions. A. Explain the file permission structure in Linux (owner, group, and others). Use an example to show how permissions are displayed with ls -l. (2 marks) ## A. File Permission Structure *(2 marks)* ● Every file has **three types of permissions** for: ○ **Owner**: the user who owns the file ○ **Group**: users who are in the same group ○ **Others**: all other users **Command to view permissions**: ls -l filename **Example Output**: -rwxr-xr-- 1 user user 2048 Apr 18 11:00 script.sh ● **Breakdown**:	CO1,CO1,CO1	Medium - - A

- `-`: regular file (`d` for directory)
- `rwx`: owner can read, write, execute
- `r-x`: group can read and execute
- `r--`: others can only read

B. What is the function of the chmod command in changing file permissions? Provide examples of how to assign read, write, and execute permissions using numeric and symbolic modes. (2 marks)

B. `chmod` Command for Changing Permissions *(2 marks)*

- **Function**: Changes the permission of a file or directory.

Symbolic Mode Examples:

Add execute to owner:

chmod u+x file.txt

Remove write from others:

chmod o-w file.txt

Numeric Mode Examples:

- `chmod 755 file.txt`
  - Owner: read + write + execute (`7`)
  - Group: read + execute (`5`)
  - Others: read + execute (`5`)
- `chmod 644 file.txt`
  - Owner: read + write (`6`)
  - Group/Others: read-only (`4`)

C. How would you modify file permissions to grant a specific user read and execute access, but not write access? Provide the exact chmod command and explain its impact. (2 marks)

C. Grant a User Read and Execute Only (No Write) *(2 marks)*

Command (using numeric mode):

chmod 555 file.txt

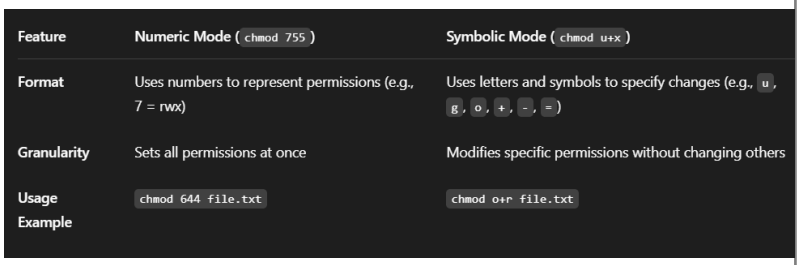
- **Explanation**:
  - `5 = r-x` (read + execute) for owner, group, and others
  - This prevents any user from writing to the file while allowing read and execution.

To apply only to a specific class (e.g., user/owner):

chmod u=rx file.txt

		○ Grants only the owner read and execute access ○ Removes write permission if it was present		
6	1.1	Explain how to navigate and manipulate directories in Linux using command-line tools. A. Describe the command to display the current directory and explain its output. (1 marks) **1. Display the Current Directory** *(1 mark)* **Command**: pwd ● **Explanation**: ○ Stands for **Print Working Directory** ○ It shows the **full path** of the directory you are currently in. **Example Output**: /home/user/Documents B. What command is used to change to a different directory in Linux? Provide an example of how to move from the current directory to the home directory. (1 marks) **2. Change to a Different Directory** *(1 mark)* **Command**: cd [directory_path] **Example (go to home directory)**: cd ~ or simply cd C. Demonstrate the usage of the mkdir command and explain how to create multiple directories at once, including an example. (2 marks) **3. Creating Multiple Directories with `mkdir`** *(2 marks)* **Command**: mkdir dir1 dir2 dir3 ● **Explanation**: ○ Creates multiple directories at once in the current location. To create nested directories: mkdir -p parent/child/grandchild ○ `-p` creates parent directories as needed.	CO1,CO1,CO1,CO1,CO1	Medium - - A

		D. Explain how you can list the contents of a directory, including hidden files. Provide an example. (1 marks) **4. List Directory Contents, Including Hidden Files** *(1 mark)* **Command**: ls -a ● **Explanation**: ○ Lists **all files**, including hidden files (those starting with `.`). Example: .bashrc .hiddenfile notes.txt E. What happens if you attempt to remove a non-empty directory using the rmdir command? (1 marks) **5. Removing a Non-Empty Directory with `rmdir`** *(1 mark)* ● **Behavior**: ○ The `rmdir` command **only removes empty directories**. If the directory is not empty, it will show an error: rmdir: failed to remove 'foldername': Directory not empty --- To remove a non-empty directory, use: rm -r foldername		
7	1.1	File access permissions in Linux control who can read, write, or execute files. They help secure files by restricting unauthorized access, ensuring that only authorized users or groups can perform specific actions on a file. A. Describe the steps to view and interpret the file access permissions of a file in Linux. (3 marks) **1. Steps to View and Interpret File Permissions** *(3 marks)* **Step 1: Use the `ls -l` command** ls -l filename **Step 2: Analyze the output** Example: -rwxr-xr-- ● **Interpretation**: ○ `-`: Indicates a regular file (`d` for a directory) ○ `rwx`: Owner has read, write, and execute permissions ○ `r-x`: Group has read and execute permissions ○ `r--`: Others have read-only permission **Step 3: Understand ownership**	CO1,CO1	Challenging - - A

		The next fields in the output show the **owner** and **group**: -rwxr-xr-- 1 user group 1024 Apr 18 12:00 filename B. How would you modify the permissions of a file to make it executable by everyone, but only the owner can modify it? (3 marks) **2. Modify Permissions: Executable by Everyone, Modifiable Only by Owner** *(3 marks)* ● **Goal**: ○ Everyone can **execute** the file ○ Only the **owner** can **modify** (write to) it ○ All users can **read** it (optional for execution visibility) **Command**: chmod 755 filename ● **Explanation**: ○ `7` (owner): read, write, execute (`rwx`) ○ `5` (group): read, execute (`r-x`) ○ `5` (others): read, execute (`r-x`) ● This ensures: ○ **Only the owner** can write (modify) ○ **Everyone** can execute (run) ○ **No one else** can alter the file		
8	1.1	In Linux, File Access Permissions (FAPs) determine who can read, write, or execute a file or directory. These permissions play a crucial role in the security and integrity of a system by ensuring that only authorized users or processes can access or modify certain files. A. Summarize the difference between the chmod numeric mode and symbolic mode for changing file permissions in Linux. (3 marks) **1. Difference Between `chmod` Numeric Mode and Symbolic Mode** *(3 marks)*  ● **Numeric Mode** is faster for full permission sets. ● **Symbolic Mode** is better for targeted permission changes. --- B. Demonstrate how to set file access permissions for a directory in Linux so that only the owner can modify it, but everyone	CO1,CO1	Medium - - A

		can read and execute files inside the directory. (3 marks) **2. Set Directory Permissions: Owner Modify, Everyone Read & Execute** *(3 marks)* • **Goal**: ○ Owner: full access (read, write, execute) ○ Group & Others: read and execute only (no write) **Command**: chmod 755 directory_name • **Explanation**: ○ `7` (owner): `rwx` → can modify and manage files ○ `5` (group): `r-x` → can read and access contents ○ `5` (others): `r-x` → same as group • **Result**: Only the owner can **modify** the directory (e.g., create/delete files), while everyone else can **read and access** the contents without changing anything.		
9	1.1	In Linux, there are various file commands used to manage, manipulate, and view files. A. Describe the different methods available in Linux to display the content of a file. How do you use the cat, more, and less commands to view file contents? Provide examples and explain when each command is best suited for use. (3 marks) **1. `cat` (Concatenate and display)** • **Use**: Shows the **entire content** of a file in one go. **Command**: cat filename.txt • **Best for**: Small files that can fit entirely in the terminal window. **2. `more` (Paged viewer)** • **Use**: Displays content **one screen at a time**. **Command**: more filename.txt • **Navigation**: Press `Enter` to scroll line by line, `Space` to go page by page. • **Best for**: Medium-sized files that don't fit in one screen. **3. `less` (Advanced paged viewer)** • **Use**: Similar to `more` but with **better navigation** features.	CO1,CO1	Medium - Analysing - A

		Command: less filename.txt ● **Navigation**: Use arrow keys, `q` to quit. ● **Best for**: Large files—scrolling and searching are easier. B. Examine the use of the head and tail commands in Linux. How do these commands help in managing and inspecting files? Include examples to show how to display the first and last few lines of a file. (3 marks) These commands show only specific parts of a file. **1. `head` (First lines)** **Command**: head -n 5 filename.txt ● ● **Use**: Displays the **first 5 lines** of the file. ● **Default**: Shows 10 lines if `-n` is not used. **2. `tail` (Last lines)** **Command**: tail -n 10 filename.txt ● ● **Use**: Displays the **last 10 lines** of the file. **Also useful for live monitoring**: tail -f logfile.txt ● ● **Best for**: ○ `head`: Quickly checking headers or introductions. ○ `tail`: Viewing logs or file endings.		
10	1.1	File manipulation in Linux involves a variety of commands to create, modify, copy, move, delete, and search files. A. Describe how the cp command works in Linux to copy files and directories. What are the different options available with the cp command, and how would you use them in various scenarios? Provide examples of copying both files and directories. (3 marks) **1. Using `cp` Command to Copy Files and Directories *(3 marks)*** The `cp` (copy) command is used to duplicate files and directories. **Basic Syntax:** cp [options] source destination **Common Options:**	CO1,CO1	Medium - - A

- `-r` or `--recursive`: Copy directories recursively
- `-v`: Verbose mode (shows progress)
- `-i`: Prompt before overwriting files
- `-u`: Copy only if the source file is newer

Examples:
Copy a file:

cp file1.txt backup.txt

Copy a file with confirmation:

cp -i report.txt /home/user/documents/

Copy a directory recursively:

cp -r myfolder /home/user/backup/

Copy with verbose output:

cp -rv myfolder /home/user/backup/

B. Demonstrate the usage of the rm command in Linux for removing files and directories. What are the different options you can use with rm, and how would you safely remove files? Provide examples for removing both files and directories. (3 marks)

2. Using `rm` Command to Remove Files and Directories *(3 marks)*

The `rm` (remove) command is used to delete files or directories.

Basic Syntax:
rm [options] filename

Common Options:

- `-i`: Interactive mode (asks before deletion)
- `-f`: Force delete without prompts
- `-r`: Remove directories recursively
- `-v`: Verbose mode

Examples:
Remove a file:

rm old_file.txt

Remove a file safely with confirmation:

rm -i draft.txt

Remove a directory and its contents:

rm -r old_folder/

Forcefully remove without prompts:

		rm -rf temp_folder/ **Precautions:** • Always double-check paths when using `rm -rf`, as it can permanently delete data. • Use `-i` for safer deletion in critical directories.		

11	1.1	Linux provides a set of powerful commands for manipulating files and directories in various ways, often allowing users to perform complex tasks with simple command-line tools. A. Describe how the mv command is used in Linux to move or rename files and directories. Provide examples for both moving files between directories and renaming files. (3 marks) **1. Using the `mv` Command in Linux *(3 marks)*** The `mv` (move) command is used to **move or rename** files and directories. **Basic Syntax:** mv [source] [destination] **Examples:** **Move a file to another directory**: mv file1.txt /home/user/documents/ **Rename a file**: mv oldname.txt newname.txt **Move and rename at the same time**: mv notes.txt /home/user/archive/notes_old.txt If a file with the same name exists in the destination, it will be overwritten without warning (unless `-i` is used). --- B. How would you display the contents of a file page by page in Linux? Explain how the more and less commands are used for paging file contents, and compare their functionalities. When should each be used? (3 marks) **2. Paging File Contents with `more` and `less` *(3 marks)*** **`more` Command:** • Displays file content **one screen at a time**. • Basic navigation: ◦ `Space`: next page ◦ `Enter`: next line ◦ `q`: quit	CO1,CO1	Medium - - A

Example:

more filename.txt

`less` Command:

- More advanced than `more`.
- Allows **forward and backward** navigation.
- Supports **searching** (`/text`).
- Navigation:
  - Arrow keys or Page Up/Page Down
  - `q`: quit

Example:

less filename.txt

Comparison:

Feature	`more`	`less`
Navigation	Forward only	Forward & backward
Search	Limited	Full search support
Performance	Slower on large files	Efficient and faster

When to Use:

- Use `more` for **quick viewing** of small or medium files.
- Use `less` for **large files** or when you need to search and scroll both ways.

12	1.1	In Linux, directories are essential for organizing and managing files. There are several commands for working with directories—creating, listing, navigating, and removing them. A. Analyze the scenario that you change the current directory in Linux using the cd command. Describe how the cd command works with relative and absolute paths, and give examples of both types of navigation. (3 marks) **1. Using `cd` to Change Directories (Relative & Absolute Paths) *(3 marks)*** The `cd` (change directory) command is used to move between directories in the Linux file system. **Absolute Path:** - Starts from the root `/` directory. **Example**:	CO1,CO1	Challenging - - A

cd /home/user/documents

- This takes you directly to the specified directory, no matter where you are currently.

Relative Path:

- Based on your current location.

Example:

cd projects/reports

- If you're in `/home/user`, this moves you to `/home/user/projects/reports`.

Special Navigation:

- `cd ~` or `cd`: Goes to the home directory.
- `cd ..`: Moves one level up.
- `cd -`: Goes back to the previous directory.

B. Examine how you can combine directory commands to perform more advanced tasks, such as creating a directory structure and listing its contents immediately afterward. Provide an example scenario. (3 marks)

2. Combining Directory Commands for Advanced Tasks *(3 marks)*

You can chain commands using `&&` or `;` to perform multiple actions in sequence.

Example Scenario:

Create a nested directory structure and list its contents:

mkdir -p ~/projects/2025/research && ls -R ~/projects

- `mkdir -p`: Creates multiple levels at once.
- `ls -R`: Lists all contents **recursively**.

Explanation:

This command:

1. Creates the full path `~/projects/2025/research`
2. Immediately lists everything inside `~/projects` including subdirectories.

13	2.1	You are tasked with creating a Bash script to perform two specific functions: A. Check if a Number is Even or Odd: • The script should prompt the user to input a number and check if the number is even or odd. • If the number is even, it should print "The number is even." • If the number is odd, it should print "The number is odd." • The script should handle invalid inputs (e.g., non-numeric inputs) by printing "Invalid input, please enter a number." (3 marks) **✅ Script Part 1: Check if Number is Even or Odd** **#!/bin/bash** **# Prompt user to enter a number** **echo "Enter a number:"** **read number** **# Check if the input is a valid number** **if [["$number" =~ ^-?[0-9]+$]]; then** **if ((number % 2 == 0)); then** **echo "The number is even."** **else** **echo "The number is odd."** **fi** **else** **echo "Invalid input, please enter a number."** **fi** B. Print a Message Based on the Day of the Week: • The script should prompt the user to input a day of the week (e.g., "Monday"). • Using a case construct, it should print a message depending on the day: ◦ If it's "Monday", print "Start of the work week." ◦ If it's "Friday", print "Almost weekend!" ◦ For any other day, print "Keep going!" (3 marks) **✅ Script Part 2: Print Message Based on Day of the Week** **#!/bin/bash** **# Prompt user to enter a day**	CO2,CO2	Challenging - - A

```
echo "Enter a day of the week:"
read day

# Convert input to lowercase
day_lower=$(echo "$day" | tr '[:upper:]' '[:lower:]')

# Check the day using case
case "$day_lower" in
  monday)
    echo "Start of the work week."
    ;;
  friday)
    echo "Almost weekend!"
    ;;
  *)
    echo "Keep going!"
    ;;
esac
```

📌 How to Use:

1. **Create two separate files**:
   - `even_or_odd.sh`
   - `day_message.sh`

Make them executable:

```
chmod +x even_or_odd.sh
chmod +x day_message.sh
```

Run them one at a time:

```
./even_or_odd.sh
./day_message.sh
```

14	2.1	You are developing a script to check the divisibility of a user-input number for a small utility program on a Linux system. The goal is to inform the user whether the number they entered is divisible by both 2 and 3, only by one of them, or by neither. This script should be capable of handling all possible cases effectively. A. Write the full shell script using an if-else statement to perform the divisibility check. Ensure that the script correctly handles the cases where the number is divisible by both 2 and 3, only by one, or by neither. (3 marks) **✅ Script 1: Divisibility by 2 and 3** **#!/bin/bash** **# Prompt the user for input** **echo "Enter a number:"** **read number** **# Check if the input is a valid number** **if [["$number" =~ ^-?[0-9]+$]]; then** **# Check divisibility by 2 and 3** **if ((number % 2 == 0 && number % 3 == 0)); then** **echo "The number is divisible by both 2 and 3."** **elif ((number % 2 == 0)); then** **echo "The number is divisible by 2 only."** **elif ((number % 3 == 0)); then** **echo "The number is divisible by 3 only."** **else** **echo "The number is divisible by neither 2 nor 3."** **fi** **else** **echo "Invalid input, please enter a valid number."** **fi** **Explanation:** • The script prompts the user for a number. • It checks divisibility by both 2 and 3 using an `if-else` statement. • It handles all possible cases: ○ Divisible by both 2 and 3. ○ Divisible by 2 only. ○ Divisible by 3 only. ○ Divisible by neither 2 nor 3. • If the user input is not a valid number, it handles the error. B. Modify the script to check for divisibility by 4 instead of 3. Provide the necessary changes and describe the impact of this modification on the script's functionality (3 marks) **✅ Script 2: Modified for Divisibility by 2 and 4** **#!/bin/bash** **# Prompt the user for input** **echo "Enter a number:"** **read number** **# Check if the input is a valid number**	CO2,CO2	Medium - - A

		if [["$number" =~ ^-?[0-9]+$]]; then ** # Check divisibility by 2 and 4** ** if ((number % 2 == 0 && number % 4 == 0)); then** ** echo "The number is divisible by both 2 and 4."** ** elif ((number % 2 == 0)); then** ** echo "The number is divisible by 2 only."** ** elif ((number % 4 == 0)); then** ** echo "The number is divisible by 4 only."** ** else** ** echo "The number is divisible by neither 2 nor 4."** ** fi** **else** ** echo "Invalid input, please enter a valid number."** **fi** **Changes:** • **Divisibility check for 4**: Instead of checking for divisibility by 3, the script now checks for divisibility by 4. • **Impact**: The primary change is replacing the divisibility check for 3 with 4 in the conditional structure: ○ `(( number % 4 == 0 ))` checks if the number is divisible by 4. ○ This affects the output where divisibility by 3 has been replaced with divisibility by 4. --- **📌 How to Use:** 1. **Create the scripts**: ○ Save the first script as `divisibility_2_3.sh`. ○ Save the second script as `divisibility_2_4.sh`. **Make them executable**: chmod +x divisibility_2_3.sh chmod +x divisibility_2_4.sh **Run each script**: ./divisibility_2_3.sh ./divisibility_2_4.sh		
15	2.1	Create a shell script using the case statement that accepts a month number (1-12) from the user and prints the corresponding month name. For example, if the user enters 1, the script should print "January". Here's a simple shell script using a `case` statement that takes a month number (1-12) as input and prints the corresponding month name: **#!/bin/bash** **# Prompt the user for a month number** **echo "Enter a month number (1-12):"** **read month** **# Use the case statement to print the corresponding month name**	CO2	Medium - Creating - A

		case $month in **1) echo "January" ;;** **2) echo "February" ;;** **3) echo "March" ;;** **4) echo "April" ;;** **5) echo "May" ;;** **6) echo "June" ;;** **7) echo "July" ;;** **8) echo "August" ;;** **9) echo "September" ;;** **10) echo "October" ;;** **11) echo "November" ;;** **12) echo "December" ;;** ***) echo "Invalid month number. Please enter a number between 1 and 12." ;;** **esac** ## 📌 How to Use: 1. **Save the script** as `month_name.sh`. **Make the script executable**: chmod +x month_name.sh **Run the script**: ./month_name.sh This script will prompt the user to enter a month number and then use a `case` statement to display the corresponding month name. If the user enters a number outside the range of 1-12, the script will print an error message.		
16	2.1	Compose a shell script using if-elif that prompts the user for a number and assigns a rating based on the input. If the number is between 90 and 100, the rating should be "Excellent". Between 70 and 89, it should be "Good", and below 70 should be "Needs Improvement". Here's a shell script that uses an `if-elif` structure to assign a rating based on the input number: **#!/bin/bash** **# Prompt the user to enter a number** **echo "Enter a number:"** **read number** **# Use if-elif structure to assign ratings based on the number** **if ((number >= 90 && number <= 100)); then** **echo "Rating: Excellent"** **elif ((number >= 70 && number < 90)); then** **echo "Rating: Good"** **elif ((number < 70)); then** **echo "Rating: Needs Improvement"** **else** **echo "Invalid input. Please enter a valid number."** **fi**	CO2	Medium - Creating - A

		📌 How to Use: 1. **Save the script** as `rating.sh`. **Make the script executable**: chmod +x rating.sh 2. **Run the script**: ./rating.sh 3. **Explanation:** ● The script prompts the user to input a number. ● It then checks: ○ If the number is between 90 and 100, it prints "Excellent." ○ If the number is between 70 and 89, it prints "Good." ○ If the number is below 70, it prints "Needs Improvement." ● If the input doesn't match any condition (though unlikely with numerical input), it outputs an error message.		

17	2.1	Design a shell script that uses a for loop to iterate through the files in a directory and checks if each file is readable, writable, and executable. Print the permissions for each file in a human-readable format (e.g., "Readable, Writable, Executable" or the combination that applies). **#!/bin/bash** **# Prompt user for the directory** **echo "Enter the directory path:"** **read directory** **# Check if the directory exists** **if [! -d "$directory"]; then** **echo "Directory does not exist."** **exit 1** **fi** **# Iterate through the files in the directory** **for file in "$directory"/*; do** **# Skip directories and non-files** **if [! -f "$file"]; then** **continue** **fi**	CO2	Medium - Creating - A

		# Initialize permission status **permission=""** **# Check permissions** **[-r "$file"] && permission+="Readable "** **[-w "$file"] && permission+="Writable "** **[-x "$file"] && permission+="Executable "** **# If no permissions, mark as "No permissions"** **if [-z "$permission"]; then** **permission="No permissions"** **fi** **# Output the result** **echo "$file: $permission"** **done** ## 📌 How to Use: 1. **Save the script** as `check_permissions.sh`. **2.Make it executable**: chmod +x check_permissions.sh **3. Run the script**: ./check_permissions.sh ## Explanation: - **Directory Input**: The script prompts the user for a directory path and checks if it exists. - **File Iteration**: The script loops through each file in the directory, skipping directories. - **Permission Checks**: It uses simple conditionals (`[ -r ]`, `[ -w ]`, `[ -x ]`) to check if the file is readable, writable, or executable. - **Output**: For each file, it prints out the permission status in a simple format like "Readable", "Writable", and "Executable" or "No permissions" if none apply.		
18	2.1	Formulate a shell script using a while loop that prints all the prime numbers between 1 and 100. Use a nested while loop to check if a number is divisible by any number other than 1 and itself. ## ✅ Shell Script: prime_numbers.sh **#!/bin/bash** **# Initialize the number** **num=2** **# Loop through numbers from 2 to 100** **while [$num -le 100]; do** **# Check if the number is prime** **is_prime=1** **div=2** **# Nested loop to check divisibility**	CO2	Challenging - Creating - A

		while [$div -lt $num]; do **if [$((num % div)) -eq 0]; then** **is_prime=0** **break** **fi** **div=$((div + 1))** **done** **# If the number is prime, print it** **if [$is_prime -eq 1]; then** **echo $num** **fi** **# Increment the number** **num=$((num + 1))** **done** ## 📌 How to Use: 1. **Save the script** as `prime_numbers.sh`. 2. **Make it executable**: chmod +x prime_numbers.sh 3. **Run the script**: ./prime_numbers.sh ## Explanation: - **Outer Loop**: The `while` loop iterates over numbers from 2 to 100. - **Prime Check**: For each number, the script checks if it is divisible by any number other than 1 and itself using a nested `while` loop. - **Nested Loop**: The nested `while` loop checks divisibility starting from 2 up to one less than the current number. - **Prime Identification**: If a number is divisible by any number other than itself, it is marked as not prime. If it is not divisible by any other number, it is considered prime and printed.		
19	2.1	You have a directory containing multiple text files. You need to create a script that checks the size of each file. If the file size exceeds 1 MB, the script should print the file name and its size. Otherwise, it should print a message saying the file is smaller than 1 MB. A. Show how to use a for loop to iterate over files in a directory.(3 marks) ## Part 1: Using a `for` loop to iterate over files in a directory In this part of the script, we use a `for` loop to iterate through all text files in the specified directory. **Code for part 1:** **# Iterate through each text file in the directory** **for file in "$directory"/*.txt; do** **# Check if it's a file and exists** **if [-f "$file"]; then** **# Further actions will be explained below**	CO1,CO2,CO1,CO2	Medium - - A

```
    fi
done
```

- **Explanation**:
    - The `for` loop iterates over all `.txt` files in the given directory (`"$directory"/*.txt`).
    - The `if [ -f "$file" ]` condition ensures that only regular files (not directories or other types) are processed.

B. Measure how to use conditional checks to compare the file size (using commands like ls) (3 marks)

Part 2: Using conditional checks to compare the file size

This part of the script checks the size of each file and compares it to 1 MB. If the file size exceeds 1 MB, it prints the file's name and size; otherwise, it prints a message stating that the file is smaller than 1 MB.

Code for part 2:

```
# Get the file size in bytes
filesize=$(stat -c %s "$file")

# Check if the file size exceeds 1 MB (1 MB = 1048576 bytes)
if [ $filesize -gt 1048576 ]; then
    # Print file name and its size in kilobytes
    echo "$file: $(($filesize / 1024)) KB"
else
    # Print message if the file is smaller than 1 MB
    echo "$file is smaller than 1 MB."
fi
```

- **Explanation**:
    - `stat -c %s "$file"` retrieves the size of the file in bytes.
    - The script compares the size (`$filesize`) with 1 MB (1048576 bytes).
    - If the file size is greater than 1 MB, the script prints the file name and its size in kilobytes.
    - If the file size is smaller than 1 MB, it prints a message stating so.

Complete Script (Both Parts Combined):

```
#!/bin/bash

# Prompt user for the directory path
echo "Enter the directory path:"
read directory

# Check if the directory exists
if [ ! -d "$directory" ]; then
    echo "Directory does not exist."
    exit 1
fi
```

		# Iterate through each text file in the directory **for file in "$directory"/*.txt; do** ** # Check if it's a file and exists** ** if [-f "$file"]; then** ** # Get the file size in bytes** ** filesize=$(stat -c %s "$file")** ** # Check if the file size exceeds 1 MB (1 MB = 1048576 bytes)** ** if [$filesize -gt 1048576]; then** ** # Print file name and its size in kilobytes** ** echo "$file: $(($filesize / 1024)) KB"** ** else** ** # Print message if the file is smaller than 1 MB** ** echo "$file is smaller than 1 MB."** ** fi** ** fi** **done** ## Summary: • **Part 1**: The script iterates over all `.txt` files in the directory using a `for` loop. • **Part 2**: It then checks each file's size using `stat` and compares it to 1 MB, printing appropriate messages depending on the file's size.		
20	2.1	Create a bash script that uses a for loop to iterate through all .txt files in the directory /home/user/documents/ and create a backup of each file with a _backup suffix. Explain the logic behind the iteration and how you handle the file copying process. Here is a simple Bash script that iterates through all `.txt` files in the directory `/home/user/documents/` and creates a backup of each file with a `_backup` suffix. ## ✅ Shell Script: create_backups.sh **#!/bin/bash** **# Directory containing the .txt files** **directory="/home/user/documents/"** **# Iterate through each .txt file in the directory** **for file in "$directory"/*.txt; do** ** # Check if the file exists (in case there are no .txt files in the directory)** ** if [-f "$file"]; then** ** # Create the backup file name by appending _backup to the original file name** ** backup_file="${file%.txt}_backup.txt"** ** # Copy the file to create a backup** ** cp "$file" "$backup_file"** ** # Print a confirmation message** ** echo "Backup created for: $file -> $backup_file"** ** fi** **done** ## 📌 How to Use: 1. **Save the script** as `create_backups.sh`.	CO1,CO2	Medium - Creating - A

2. **Make it executable**:

   chmod +x create_backups.sh

3. **Run the script**:

   ./create_backups.sh

Explanation of the Logic:

1. **Directory Path**:
   - The script defines the path of the directory containing the `.txt` files (`/home/user/documents/`).
2. **For Loop**:
   - The `for` loop iterates over all `.txt` files in the specified directory (`"$directory"/*.txt`). This pattern will match all files with the `.txt` extension in the directory.
   - The variable `file` holds the name of each `.txt` file in each iteration.
3. **File Existence Check**:
   - The `if [ -f "$file" ]` condition checks if the current item is a regular file (not a directory or other type) before proceeding with the backup. This ensures the script only processes actual files.
4. **Backup File Name**:
   - The line `backup_file="${file%.txt}_backup.txt"` creates a new backup file name by removing the `.txt` extension and appending `_backup.txt`.
   - The `${file%.txt}` part strips the `.txt` extension from the original file name.
5. **File Copying**:
   - The `cp "$file" "$backup_file"` command copies the original file to the new backup file. This effectively creates a backup of the original file with the `_backup.txt` suffix.
6. **Confirmation Message**:
   - After successfully copying the file, the script prints a message confirming the backup operation for each file.

Example Output:

Backup created for: /home/user/documents/file1.txt -> /home/user/documents/file1_backup.txt
Backup created for: /home/user/documents/file2.txt -> /home/user/documents/file2_backup.txt

Key Points:

- **Iteration**: The `for` loop is used to iterate over all `.txt` files in the directory. If there are no `.txt` files, the loop will simply not run.

		• **Backup Creation**: The `cp` command is used to copy each file, and the script adds a `_backup` suffix to the file name for easy identification of backup files. This script will create backups of all `.txt` files in the specified directory, appending `_backup` to the original file names.		
21	2.1	Create a shell script using the case statement that accepts a grade (A, B, C, D, or F) from the user and prints the corresponding description. For example, if the user enters "A", the script should print "Excellent". Here is a simple shell script that uses the `case` statement to accept a grade from the user and prints the corresponding description. ## ✅ Shell Script: grade_description.sh **#!/bin/bash** **# Prompt the user to enter a grade** **echo "Enter your grade (A, B, C, D, or F):"** **read grade** **# Use the case statement to determine the description based on the grade** **case $grade in** **A)** **echo "Excellent"** **;;** **B)** **echo "Good"** **;;** **C)** **echo "Average"** **;;** **D)** **echo "Poor"** **;;** **F)** **echo "Fail"** **;;** ***)** **echo "Invalid grade entered. Please enter A, B, C, D, or F."** **;;** **esac** ## 📌 How to Use: 1. **Save the script** as `grade_description.sh`. 2. **Make it executable**: chmod +x grade_description.sh 3. **Run the script**: ./grade_description.sh ## Explanation of the Script: 1. **Prompting the User**:	CO2	Medium - Creating - A

		○ The script first prompts the user to enter a grade (`A`, `B`, `C`, `D`, or `F`) by displaying a message and reading the input into the variable `grade`. 2. **Using the `case` Statement**: ○ The `case` statement is used to match the user's input against the possible grade values (`A`, `B`, `C`, `D`, `F`). ○ For each grade, the script prints a corresponding description: ■ `A` → "Excellent" ■ `B` → "Good" ■ `C` → "Average" ■ `D` → "Poor" ■ `F` → "Fail" 3. **Default Case**: ○ If the user enters anything other than `A`, `B`, `C`, `D`, or `F`, the script outputs `"Invalid grade entered. Please enter A, B, C, D, or F."`. ## Example Output: Enter your grade (A, B, C, D, or F): A Excellent ## Key Points: • **Case Statement**: This allows handling different inputs in a clean and readable way. It's ideal for situations where a variable can take one of a limited set of values. • **User Input Handling**: The script ensures that it handles valid grade inputs and provides an error message for invalid ones.		
22	2.1	Create a shell script using the case statement that accepts a month number (1-12) from the user and prints whether the month is in the "first quarter," "second quarter," "third quarter," or "fourth quarter" of the year. Here is a shell script that uses the `case` statement to accept a month number (1-12) from the user and prints which quarter of the year the month belongs to. ## ✅ Shell Script: month_quarter.sh **#!/bin/bash** **# Prompt the user to enter a month number (1-12)** **echo "Enter a month number (1-12):"** **read month** **# Use the case statement to determine the quarter of the year** **case $month in** **1|2|3)** **echo "This month is in the first quarter."**	CO2	Challenging - Creating - A

```
    ;;
  4|5|6)
    echo "This month is in the second quarter."
    ;;
  7|8|9)
    echo "This month is in the third quarter."
    ;;
  10|11|12)
    echo "This month is in the fourth quarter."
    ;;
  *)
    echo "Invalid month number. Please enter a number between 1 and 12."
    ;;
esac
```

📌 How to Use:

1. **Save the script** as `month_quarter.sh`.

2. **Make it executable**:

   chmod +x month_quarter.sh

3. **Run the script**:

   ./month_quarter.sh

Explanation of the Script:

1. **Prompting the User**:
   - The script prompts the user to enter a month number (1-12) and stores the input in the variable `month`.
2. **Using the `case` Statement**:
   - The `case` statement checks the value of `$month` and matches it to the respective quarter:
     - Months 1, 2, and 3 correspond to the **first quarter** (Q1).
     - Months 4, 5, and 6 correspond to the **second quarter** (Q2).
     - Months 7, 8, and 9 correspond to the **third quarter** (Q3).
     - Months 10, 11, and 12 correspond to the **fourth quarter** (Q4).
3. **Default Case**:
   - If the user enters a number other than 1-12, the script outputs `"Invalid month number. Please enter a number between 1 and 12."`.

Example Output:

Enter a month number (1-12):

		3 This month is in the first quarter. ## Key Points: - **Quarter Classification**: The `case` statement groups the months into quarters, making the script easy to understand and modify if necessary. - **Error Handling**: The script ensures that only valid month numbers (1-12) are accepted and provides an error message for invalid inputs.		
23	2.1	Write a shell script using the case statement that accepts a user's choice of a browser (e.g., "Chrome," "Firefox," "Safari") and prints the corresponding default website (e.g., "Chrome - google.com," "Firefox - mozilla.org"). Here is a shell script that uses the `case` statement to accept a user's choice of browser and prints the corresponding default website for that browser. ## ✅ Shell Script: browser_website.sh **#!/bin/bash** **# Prompt the user to enter the browser name** **echo "Enter your preferred browser (Chrome, Firefox, Safari):"** **read browser** **# Use the case statement to print the corresponding default website** **case $browser in** **Chrome)** **echo "Chrome - google.com"** **;;** **Firefox)** **echo "Firefox - mozilla.org"** **;;** **Safari)** **echo "Safari - apple.com"** **;;** ***)** **echo "Invalid browser entered. Please enter one of the following: Chrome, Firefox, or Safari."** **;;** **esac** ## 📌 How to Use: 1. **Save the script** as `browser_website.sh`. 2. **Make it executable**: chmod +x browser_website.sh 3. **Run the script**: ./browser_website.sh ## Explanation of the Script:	CO2	Medium - Creating - A

		1. **Prompting the User**: ○ The script prompts the user to enter a browser name (`Chrome`, `Firefox`, or `Safari`) and reads the input into the variable `browser`. 2. **Using the `case` Statement**: ○ The `case` statement checks the value of `$browser` and matches it to the corresponding website: ■ If the user enters `Chrome`, the script prints `Chrome - google.com`. ■ If the user enters `Firefox`, the script prints `Firefox - mozilla.org`. ■ If the user enters `Safari`, the script prints `Safari - apple.com`. 3. **Default Case**: ○ If the user enters something other than the three expected options, the script outputs `"Invalid browser entered. Please enter one of the following: Chrome, Firefox, or Safari."`. ## Example Output: Enter your preferred browser (Chrome, Firefox, Safari): Chrome Chrome - google.com ## Key Points: • **Browser Selection**: The `case` statement helps to associate specific browsers with their default websites in a clear and organized manner. • **Error Handling**: The script includes a default case that handles invalid input by reminding the user of the valid browser options.		
24	2.1	Create a shell script that uses a for loop to iterate through all files in a given directory and checks if each file is a regular file, directory, or symbolic link. Print the type of each file. You should iterate through files using a for loop, use -f, -d, and -L to check file types, and output the file type (e.g., "Regular File," "Directory," "Symbolic Link"). Here's a shell script that uses a `for` loop to iterate through all files in a given directory and checks if each file is a regular file, directory, or symbolic link. The script uses the `-f`, `-d`, and `-L` options to check the file types. ## ✅ Shell Script: file_type_check.sh **#!/bin/bash** **# Prompt the user to enter a directory path** **echo "Enter the directory path:"** **read directory** **# Check if the entered directory exists** **if [! -d "$directory"]; then**	CO2	Challenging - Creating - A

```
    echo "The directory does not exist. Please enter a valid directory."
    exit 1
fi

# Iterate through each file in the directory
for file in "$directory"/*; do
    if [ -f "$file" ]; then
        echo "$file is a Regular File."
    elif [ -d "$file" ]; then
        echo "$file is a Directory."
    elif [ -L "$file" ]; then
        echo "$file is a Symbolic Link."
    else
        echo "$file is of unknown type."
    fi
done
```

📌 How to Use:

1. **Save the script** as `file_type_check.sh`.
2. **Make it executable**:

   chmod +x file_type_check.sh

3. **Run the script**:

   ./file_type_check.sh

Explanation of the Script:

1. **Prompt the User for Directory**:
   - The script first asks the user to input a directory path. The entered path is stored in the variable `directory`.
2. **Directory Existence Check**:
   - The script checks whether the entered path is a valid directory using `if [ ! -d "$directory" ]`. If the directory doesn't exist, the script exits with an error message.
3. **For Loop for Iterating Files**:
   - The script then iterates through all files in the specified directory using the `for` loop.
   - It checks the type of each file using:
     - `-f` for checking if it's a **regular file**.
     - `-d` for checking if it's a **directory**.
     - `-L` for checking if it's a **symbolic link**.
   - Based on the file type, it prints the appropriate message.
4. **Unknown File Types**:
   - If a file doesn't match any of the above types, it will be labeled as "unknown type."

		## Example Output: Enter the directory path: /home/user/documents /home/user/documents/file1.txt is a Regular File. /home/user/documents/folder1 is a Directory. /home/user/documents/link1 is a Symbolic Link. ## Key Points: - **File Type Checking**: The script uses `-f`, `-d`, and `-L` options to determine whether the file is a regular file, directory, or symbolic link, respectively. - **Directory Path Validation**: The script checks if the provided directory path is valid and exists before attempting to iterate over files.		
25	3.1	You are a system administrator working on a large Linux file server. The server contains multiple directories and a large number of files with varying sizes. Your task is to perform different searches for files based on specific conditions for maintenance and optimization purposes. A. Write a find command to locate all directories (not files) within the /home/ directory that have been accessed in the last 30 days. Explain how the find command works in this scenario. (2 marks) ## 1. Find All Directories Accessed in the Last 30 Days To locate all directories (not files) within the `/home/` directory that have been accessed in the last 30 days, you can use the following `find` command: find /home/ -type d -atime -30 ### Explanation: - `find`: The command used to search files and directories. - `/home/`: The directory where the search will be conducted. - `-type d`: This option ensures that only directories are selected (not files). - `-atime -30`: This option filters directories that were accessed within the last 30 days. The `-` before `30` means "less than 30 days ago." B. Write a find command to locate all files within the /data/ directory that are larger than 500MB. (2 marks) ## 2. Find All Files Larger Than 500MB in /data/ Directory To find all files in the `/data/` directory that are larger than 500MB, use this command: find /data/ -type f -size +500M	CO3,CO3,CO3	Challenging - - A

Explanation:

- `find`: The command used to search files and directories.
- `/data/`: The directory where the search will be conducted.
- `-type f`: This option ensures that only files are selected (not directories).
- `-size +500M`: This option filters files that are larger than 500MB. The `+` before `500M` means "greater than 500MB."

C. Write a find command to locate all files in the /var/www/ directory that were modified exactly 60 days ago. (2 marks)

3. Find All Files Modified Exactly 60 Days Ago in /var/www/ Directory

To find all files in the `/var/www/` directory that were modified exactly 60 days ago, use the following command:

find /var/www/ -type f -mtime 60

Explanation:

- `find`: The command used to search files and directories.
- `/var/www/`: The directory where the search will be conducted.
- `-type f`: This option ensures that only files are selected.
- `-mtime 60`: This option filters files that were modified exactly 60 days ago. The number `60` is used without a `+` or `-`, meaning it will match files modified exactly 60 days ago.

These `find` commands are powerful for locating files and directories based on various criteria, such as type, size, and modification/access times.

26	3.1	You are a system administrator managing a server that logs activities of different users. A. Using find with grep to Search for Content in Specific Files: • Write a command that combines find and grep to search for the term "critical" only in .log files modified within the last 7 days. • Explain how the find and grep commands work together to achieve this. (3 marks) **1. Using `find` with `grep` to Search for Content in Specific Files** To search for the term "critical" in `.log` files modified within the last 7 days, you can use the following combined `find` and `grep` command: find /path/to/logs/ -type f -name "*.log" -mtime -7 -exec grep -H "critical" {} \; **Explanation:** • **`find /path/to/logs/`**: This command starts searching from the `/path/to/logs/` directory. You should replace `/path/to/logs/` with the actual path where your log files are stored. • **`-type f`**: Ensures that only files are considered (not directories). • **`-name "*.log"`**: Filters files with a `.log` extension. • **`-mtime -7`**: Searches for files that have been modified in the last 7 days. The `-7` means "less than 7 days ago." • **`-exec grep -H "critical" {} \;`**: For each file that matches the `find` conditions, the `grep` command is executed. ◦ `grep -H "critical" {}`: The `grep` command searches for the term "critical" inside the files. The `-H` option ensures the file name is printed with the matching lines. ◦ `{}` is replaced by the path of each file found by `find`, and `\;` tells `find` to execute the command on each file. **How `find` and `grep` work together:** • **`find`** is used to locate files that meet the conditions (i.e., `.log` files modified in the last 7 days). • **`grep`** is used to search within those files for the term "critical." The output will show the file names along with the lines where "critical" appears. --- B. Using find, grep, and wc Together to Analyze Error Logs: • Write a command that uses find, grep, and wc together to search for "failure" and count the number of occurrences. • Explain how each command contributes to the overall process and the expected output	CO3,CO3	Challenging - - A

(3 marks)

2. Using `find`, `grep`, and `wc` Together to Analyze Error Logs

To search for the term "failure" in log files and count the number of occurrences, you can use the following command:

```
find /path/to/logs/ -type f -name "*.log" -exec grep -o "failure" {} \; | wc -l
```

Explanation:

- **`find /path/to/logs/`**: This command searches from the `/path/to/logs/` directory. Replace it with the actual directory containing your log files.
- **`-type f`**: Ensures only files are considered.
- **`-name "*.log"`**: Filters the files to include only `.log` files.
- **`-exec grep -o "failure" {} \;`**: For each file found, `grep -o "failure"` searches for the term "failure" and returns only the matches (not the full line). The `-o` option makes `grep` print each occurrence of the word "failure" on a new line.
- **`| wc -l`**: The output from `grep` is passed through a pipe (`|`) to `wc -l`, which counts the number of lines (i.e., the number of times "failure" appears in the files).

How `find`, `grep`, and `wc` work together:

- **`find`** locates the files that match the search criteria (`*.log` files).
- **`grep`** searches for the term "failure" within those files. The `-o` option makes `grep` output each match on a new line.
- **`wc -l`** counts the number of lines produced by `grep`, which corresponds to the number of occurrences of the term "failure."

Expected Output:

The output will be a single number that represents how many times the term "failure" appears across all `.log` files in the specified directory.

Summary:

Find with `grep`: The command finds specific files (`*.log`) and uses `grep` to search for the term "critical" within those files, printing the matching lines and file names.

Find with `grep` and `wc`: The command locates `.log` files, searches for occurrences of the term "failure," and counts how many times it appears across all the files using `wc -l`.

27	3.1	You are tasked with analyzing large logs, filtering out redundant data, counting specific entries, and sorting information in a meaningful way. A. You have a log file /var/log/messages that contains multiple lines of repeated entries for the system start time. You want to remove consecutive duplicate lines and display the unique entries. Write the command using uniq and explain what the result will be. (1 marks) **1. Removing Consecutive Duplicate Lines in a Log File using `uniq`** To remove consecutive duplicate lines in the `/var/log/messages` file and display only the unique entries, you can use the following command: uniq /var/log/messages **Explanation:** • **`uniq`**: This command filters out consecutive duplicate lines from a sorted file or input. It only removes duplicates that appear immediately next to each other. • When you run `uniq`, the command will read the `/var/log/messages` file, remove consecutive duplicate lines, and display only the unique entries in the terminal. **Result:** The output will be a list of unique entries in the log file, excluding any consecutive duplicates. If there are lines that repeat in the log, only the first occurrence of each will be displayed. B. You need to count how many error messages (lines containing the word "ERROR") appear in the log file /var/log/syslog. Write the command using grep and wc and explain the expected result. (1 marks) **2. Counting "ERROR" Messages in a Log File Using `grep` and `wc`** To count how many error messages appear in the `/var/log/syslog` file, you can use the following command: grep -i "ERROR" /var/log/syslog | wc -l **Explanation:** • **`grep -i "ERROR" /var/log/syslog`**: This searches the `/var/log/syslog` file for lines containing the word "ERROR", using the `-i` flag to make the search case-insensitive (so it will match "ERROR", "error", "Error", etc.). • **`| wc -l`**: The output of the `grep` command (all the lines that contain "ERROR") is piped into `wc -l`, which counts the number of lines. **Result:** The output will be a single number representing how many lines in `/var/log/syslog` contain the word "ERROR". This is the total count of	CO3,CO3,CO3,CO3	Medium - - A

error messages in the file.

C. You are tasked with generating a sorted list of all unique usernames from the /var/log/auth.log file. The usernames are stored in a specific field in the log entries. After sorting the usernames, you want to store the output in a file called sorted_usernames.txt while also displaying it on the terminal. Write the command using sort and tee to achieve this, and explain how the command works. (2 marks)

3. Generating a Sorted List of Unique Usernames from `/var/log/auth.log` Using `sort` and `tee`

To generate a sorted list of all unique usernames from `/var/log/auth.log`, sort them, and store the output in a file (`sorted_usernames.txt`) while displaying it on the terminal, you can use the following command:

```
awk '{print $1}' /var/log/auth.log | sort | uniq | tee
sorted_usernames.txt
```

Explanation:

- **`awk '{print $1}' /var/log/auth.log`**: This command uses `awk` to extract the first field (which typically contains the usernames) from each line of the `/var/log/auth.log` file.
- **`sort`**: This sorts the extracted usernames alphabetically.
- **`uniq`**: After sorting, the `uniq` command removes duplicate usernames.
- **`tee sorted_usernames.txt`**: The `tee` command takes the output and writes it to `sorted_usernames.txt` while also displaying it on the terminal.

Result:

The command will display the sorted list of unique usernames in the terminal and simultaneously save the list in the file `sorted_usernames.txt`.

D. You need to know the total number of lines in the log file /var/log/dmesg. Write a command using wc to determine this number. Additionally, explain how you can redirect the output to a file called line_count.txt while still displaying it on the terminal. (2 marks)

4. Counting the Total Number of Lines in `/var/log/dmesg` Using `wc`

To count the total number of lines in the `/var/log/dmesg` file and save the output to a file while displaying it on the terminal, you can use the following command:

```
wc -l /var/log/dmesg | tee line_count.txt
```

Explanation:

- **`wc -l /var/log/dmesg`**: The `wc -l` command counts the

		number of lines in the `/var/log/dmesg` file. • **`tee line_count.txt`**: The `tee` command takes the output of `wc -l` (the line count) and writes it to the file `line_count.txt` while also displaying it on the terminal. **Result:** The output will display the total number of lines in `/var/log/dmesg` and store this count in the file `line_count.txt`.		
28	3.1	You need to extract, transform, and save specific information from logs in various ways, ensuring both accuracy and efficiency. A. You need to extract all user login attempts (lines containing the word "login") from the log file /var/log/auth.log and display them on the terminal, as well as save them to a file called login_attempts.log. Write the command using tee and explain how it works. (3 marks) **1. Extracting User Login Attempts and Saving to a File Using `tee`** To extract all user login attempts (lines containing the word "login") from the `/var/log/auth.log` file, display them on the terminal, and save them to a file called `login_attempts.log`, you can use the following command: grep "login" /var/log/auth.log | tee login_attempts.log **Explanation:** • **`grep "login" /var/log/auth.log`**: This command searches the `/var/log/auth.log` file for lines that contain the word "login". The `grep` command filters out all lines that do not match this search term, leaving only the lines that are related to login attempts. • **`| tee login_attempts.log`**: The `|` (pipe) takes the output from the `grep` command and passes it to the `tee` command. The `tee` command writes this output to the file `login_attempts.log` and simultaneously displays it on the terminal. **Result:** The `grep` command will extract all lines containing "login" from `/var/log/auth.log`, and the `tee` command will both display the result in the terminal and save it to `login_attempts.log`. B. You are required to convert all uppercase letters to lowercase in the file /var/log/messages and store the transformed text in a new file called messages_lowercase.log. Write the command using tr and explain how the command works. (3 marks) **2. Converting Uppercase to Lowercase in a File Using `tr`** To convert all uppercase letters to lowercase in the `/var/log/messages` file and store the transformed text in a new file called	CO3,CO3	Medium - - A

		`messages_lowercase.log`, you can use the following command: cat /var/log/messages | tr 'A-Z' 'a-z' > messages_lowercase.log **Explanation:** • **`cat /var/log/messages`**: This command outputs the contents of the `/var/log/messages` file. • **`| tr 'A-Z' 'a-z'`**: The `|` (pipe) passes the content of `/var/log/messages` to the `tr` (translate) command. The `tr 'A-Z' 'a-z'` command converts all uppercase letters (from A to Z) to lowercase (from a to z). • **`> messages_lowercase.log`**: The `>` symbol redirects the output to a new file named `messages_lowercase.log`, overwriting the file if it exists. **Result:** This command will read the contents of `/var/log/messages`, convert all uppercase letters to lowercase, and save the result to `messages_lowercase.log`. The original file remains unchanged.		
29	3.1	You are working as a system administrator managing log files on a Linux server. These log files need to be processed for analysis, including converting text formats and saving output for reporting. A. You want to count the number of times the word "error" appears in the /var/log/syslog file, converting all text to lowercase before counting (to ensure case-insensitive counting). After counting, you need to display the result on the terminal and also save it to a file called error_count.txt. Write the command using tr, grep, tee, and explain the expected result. (3 marks) **1. Counting "error" (Case-Insensitive) in `/var/log/syslog` and Saving the Result** To count the number of times the word "error" appears in `/var/log/syslog`, converting all text to lowercase to ensure case-insensitive counting, and displaying the result on the terminal while also saving it to `error_count.txt`, use the following command: cat /var/log/syslog | tr 'A-Z' 'a-z' | grep -o "error" | wc -l | tee error_count.txt **Explanation:** • **`cat /var/log/syslog`**: This command outputs the contents of the `/var/log/syslog` file. • **`| tr 'A-Z' 'a-z'`**: The `tr` (translate) command is used to convert all uppercase letters to lowercase, ensuring case-insensitive searching for the word "error". • **`| grep -o "error"`**: The `grep -o` command searches for occurrences of the word "error" and prints each instance on a new line. This allows `wc -l` to count the occurrences accurately. • **`| wc -l`**: The `wc -l` command counts the number of lines, which corresponds to the number of occurrences of the word "error".	CO3,CO3	Challenging - - A

- `| tee error_count.txt`: The `tee` command writes the output to both the terminal and the file `error_count.txt`.

Expected Result:

This command will count the number of times the word "error" (in any case) appears in `/var/log/syslog`, print the count to the terminal, and save the result in the file `error_count.txt`.

B. You need to extract all unique usernames from the /var/log/auth.log file, but the usernames are in uppercase, and you need to convert them to lowercase before saving the unique usernames. After processing, you should save the output to unique_usernames.txt while displaying it on the terminal. Write the command using tr, sort, uniq, and tee, and explain how it works. (3 marks)

2. Extracting Unique Usernames from `/var/log/auth.log`, Converting to Lowercase, and Saving to a File

To extract all unique usernames from `/var/log/auth.log` (which are in uppercase) and convert them to lowercase before saving the unique usernames to `unique_usernames.txt`, while also displaying them on the terminal, use the following command:

cat /var/log/auth.log | tr 'A-Z' 'a-z' | awk '{print $1}' | sort | uniq | tee unique_usernames.txt

Explanation:

- **`cat /var/log/auth.log`**: This command outputs the contents of the `/var/log/auth.log` file.
- **`| tr 'A-Z' 'a-z'`**: The `tr` command converts all uppercase letters to lowercase.
- **`| awk '{print $1}'`**: The `awk` command extracts the first field (assumed to be the username) from each log entry.
- **`| sort`**: The `sort` command orders the usernames alphabetically.
- **`| uniq`**: The `uniq` command filters out duplicate usernames, leaving only unique ones.
- **`| tee unique_usernames.txt`**: The `tee` command writes the output to both the terminal and the file `unique_usernames.txt`.

Expected Result:

This command will extract unique usernames from `/var/log/auth.log`, convert them to lowercase, sort them, remove duplicates, and then save the list to `unique_usernames.txt` while displaying it on the terminal.

Summary:

Counting "error" Case-Insensitive: The `tr` command converts text to

		lowercase, `grep` counts occurrences of "error", and `tee` saves the result to a file. **Extracting Unique Usernames**: The `tr` command converts text to lowercase, `awk` extracts usernames, `sort` sorts them, `uniq` removes duplicates, and `tee` saves the result to a file.		

30	3.1	You are running a script download_data.sh that takes a long time to download a large dataset. You want to run it in the background so that you can continue using the terminal for other tasks. A. Which command will you use to run download_data.sh in the background? (1 marks) ## 1. Running `download_data.sh` in the Background To run `download_data.sh` in the background, you can use the following command: ./download_data.sh & ### Explanation: - The **&** at the end of the command puts the `download_data.sh` script in the background, allowing you to continue using the terminal for other tasks while the script runs. B. How can you verify that download_data.sh is running in the background? (2 marks) ## 2. Verifying that `download_data.sh` is Running in the Background You can verify that `download_data.sh` is running in the background by using either of these commands: **Using `jobs` command:** jobs - This will list all background jobs associated with the current terminal session. If `download_data.sh` is running, it will show up as a background job. **Using `ps` command:** ps aux | grep download_data.sh - This will show you all running processes and filter the results to only display lines related to `download_data.sh`. ### Explanation: - **`jobs`** will display job IDs for background tasks running in the current shell session.	CO3,CO3,CO3	Medium - - A

		• **`ps aux`** lists all processes running on the system. **`grep download_data.sh`** filters the list to show processes related to the script. C. If you want the process to keep running even if you log out, how can you modify the command to ensure that it doesn't terminate when the session ends? (3 marks) ## 3. Ensuring `download_data.sh` Keeps Running After Logging Out To ensure that the script keeps running even if you log out of the session, you can use **`nohup`** (short for "no hang up"): nohup ./download_data.sh & ### Explanation: • **`nohup`**: This command prevents the process from being terminated when the terminal session is closed or the user logs out. • **`&`**: This still places the script in the background. • **`nohup` output**: By default, `nohup` saves any output to a file called `nohup.out`. You can specify a different output file by redirecting it, e.g., `nohup ./download_data.sh > output.log &`. --- ## Summary: . **Running in Background**: Use `./download_data.sh &`. . **Verifying Background Process**: Use `jobs` or `ps aux | grep download_data.sh`. **Keep Running After Logout**: Use `nohup ./download_data.sh &` to prevent the process from terminating when the session ends.		
31	3.1	You have started a long-running process in the background using a script process_data.sh. After some time, you want to check the status of the background process. A. Which command can you use to check the status of background jobs running in your session? (1 marks) ## 1. Checking the Status of Background Jobs To check the status of background jobs running in your session, you can use the following command: jobs ### Explanation: • **`jobs`**: This command lists all background jobs associated with the current terminal session. It shows the job number, the status	CO3,CO3,CO3	Medium - - A

(such as running or stopped), and the command associated with the job.

B. How can you bring the background process to the foreground to monitor its progress? (2 marks)

2. Bringing a Background Process to the Foreground

To bring a background process to the foreground, you can use the following command:

```
fg %job_number
```

Example:

If the job number is 1, you would use:

```
fg %1
```

Explanation:

- **`fg`**: This command brings a background job to the foreground.
- **`%job_number`**: Specifies the job number you want to bring to the foreground. You can find the job number by using the `jobs` command.

C. If you have multiple background jobs, how can you check the process IDs (PIDs) associated with each job? (3 marks)

3. Checking Process IDs (PIDs) for Multiple Background Jobs

To check the process IDs (PIDs) associated with each job, you can use the following command:

```
ps -ef | grep process_data.sh
```

Explanation:

- **`ps -ef`**: This command lists all running processes on the system.
- **`grep process_data.sh`**: Filters the output to show only the processes related to `process_data.sh`. This will display the PID (process ID) of the running process.

Alternatively, if you are working with multiple background jobs in the current shell session, and you want to find the PIDs of jobs managed by your shell, you can use:

```
jobs -l
```

Explanation:

		• **`jobs -l`**: This command shows all background jobs with their associated PIDs, along with their job numbers and statuses. --- ## Summary: . **Check the status of background jobs**: Use `jobs`. . **Bring a background job to the foreground**: Use `fg %job_number`. **Check process IDs for background jobs**: Use `ps -ef | grep process_data.sh` or `jobs -l`.		
32	3.1	You want to monitor the resource usage of processes running on your Linux system, including CPU and memory usage. You decide to use the top command for this. A. Which command will you use to start the top command and monitor the running processes? (1 marks) ## 1. Starting the `top` Command To start the `top` command and monitor the running processes, you can simply use: top ### Explanation: • **`top`**: This command starts the interactive process viewer, displaying a real-time view of running processes, along with information about CPU usage, memory usage, and more. B. How can you filter the top command output to only display processes using a high percentage of CPU? (2 marks) ## 2. Filtering the `top` Command Output by High CPU Usage To filter the `top` command output to only display processes using a high percentage of CPU, you can do the following: • Press **`Shift + P`** while in the `top` interface. This will sort processes by CPU usage in descending order. Alternatively, to filter processes that are using more than a specific percentage of CPU, you can press **`Shift + F`** (to enter the field selection mode) and choose the CPU usage field. After that, you can filter based on values interactively. For more advanced filtering before opening `top`, you can use the following command: top -o %CPU ### Explanation: • **`top -o %CPU`**: This command sorts the processes by CPU usage, in descending order.	CO3,CO3,CO3	Medium - - A

		If you want to only show processes that use more than a certain percentage of CPU, you can use: top -b -n 1 | awk '$9 > 50' This will show processes where the CPU usage (column 9 in `top`) is greater than 50%. C. If you want to sort the processes by memory usage, how can you achieve this in top? (3 marks) **3. Sorting Processes by Memory Usage in `top`** To sort the processes by memory usage in `top`, you can press: • **`Shift + M`** within the `top` command interface. This sorts the processes by memory usage, showing the most memory-intensive processes at the top. Alternatively, you can start `top` with sorting by memory usage: top -o %MEM **Explanation:** • **`top -o %MEM`**: This command sorts the processes by memory usage, in descending order. --- **Summary:** . **Start `top` command**: Use `top`. . **Filter by high CPU usage**: Press `Shift + P` to sort by CPU usage or use `top -o %CPU`. **Sort by memory usage**: Press `Shift + M` or use `top -o %MEM`.		
33	3.1	You have started a background process process_large_file.sh that has been running for several hours. You now want to terminate this background process because it is taking too long. A. Which command can you use to list all running background jobs? (1 marks) **1. Listing All Running Background Jobs** To list all running background jobs, use the following command: jobs **Explanation:** • **`jobs`**: This command displays all background jobs running in the	CO3,CO3,CO3	Medium - - A

current shell session, along with their job numbers and statuses (such as running or stopped).

B. How can you terminate the background process by referencing its job number or process ID (PID)? (2 marks)

2. Terminating the Background Process by Job Number or PID

To terminate the background process, you can either reference the job number or the process ID (PID).

By Job Number:

If the background job is listed by `jobs` with a job number (e.g., `[1]`, `[2]`, etc.), you can use the `kill` command followed by the job number prefixed with a `%` symbol:

```
kill %1
```

This will terminate the job with the job number `1`.

By Process ID (PID):

If you prefer using the PID directly, first, find the PID of the process using the `ps` command or `jobs -l`. Once you have the PID, you can use `kill` with the PID:

```
kill <PID>
```

For example, if the PID is `12345`:

```
kill 12345
```

Explanation:

- **`kill %1`**: This sends a termination signal (SIGTERM) to the background job with job number `1`.
- **`kill <PID>`**: This sends the SIGTERM signal to the process with the specified PID.

C. If you want to terminate the process immediately, what is the command to send a SIGKILL signal? (3 marks)

3. Sending a SIGKILL Signal to Terminate Immediately

If you want to terminate the process immediately without giving it a chance to clean up, use the SIGKILL signal. The SIGKILL signal cannot be ignored by the process, and it forces the process to terminate.

To send a SIGKILL signal, use:

```
kill -9 <PID>
```

For example, if the PID is `12345`:

```
kill -9 12345
```

		Explanation: • **`kill -9 <PID>`**: The `-9` option sends the SIGKILL signal, which immediately kills the process without giving it a chance to clean up or handle the termination. --- ## Summary: . **List all background jobs**: Use `jobs`. . **Terminate a job**: • By job number: `kill %1` • By PID: `kill <PID>` **Terminate a process immediately**: Use `kill -9 <PID>` to send a SIGKILL signal.		
34	3.1	You want to know how long it takes for the command find / -name "*.log" to search through all directories for log files. You wish to measure the time taken for the command to complete. A. Which command can you use to measure the time taken to execute the find command? (1 marks) ## 1. Measuring the Time Taken for the `find` Command To measure the time taken for a command to execute, use the `time` command: time find / -name "*.log" **Explanation:** • **`time`**: This command measures how long the specified command takes to run, displaying the real time, user CPU time, and system CPU time. B. How would you modify the command if you want to measure the total system time, user time, and real time for the execution of the find command? (2 marks) ## 2. Measuring Total System Time, User Time, and Real Time The `time` command provides information on: • **Real time**: The total elapsed time. • **User time**: The amount of CPU time spent in user mode. • **System time**: The amount of CPU time spent in kernel mode. The basic `time` command already provides these three times (real, user, and system). Here's how you can use it: time find / -name "*.log"	CO3,CO3,CO3	Medium - - A

Explanation:

- The `time` command outputs three values:
    - **Real time**: Total elapsed time from start to finish.
    - **User time**: Time the CPU spent on user-level code (outside of the kernel).
    - **System time**: Time the CPU spent on kernel-level operations (system calls).

Example output:

```
real    0m10.236s
user    0m2.314s
sys     0m0.781s
```

C. If the command is taking too long, how can you limit the search to a specific directory to speed up the process? (3 marks)

3. Limiting the Search to a Specific Directory to Speed Up the Process

To speed up the search and limit it to a specific directory, simply provide the path of that directory in the `find` command. For example, to limit the search to the `/var/log` directory, use:

```
find /var/log -name "*.log"
```

Explanation:

- **`/var/log`**: Specifies the directory where the search should be conducted. This limits the search to that directory and its subdirectories.
- By specifying a more specific directory, the search scope is reduced, which speeds up the execution time.

If you want to limit it to a specific subdirectory, simply adjust the path accordingly:

```
find /home/user/logs -name "*.log"
```

Summary:

. **Measure execution time**: Use `time find / -name "*.log"`.

. **Measure system, user, and real time**: `time` already gives you real, user, and system times by default.

Limit search to a directory: Modify the path in the `find` command, e.g., `find /var/log -name "*.log"`, to narrow down the search.

35	3.1	You need to schedule a backup script backup.sh to run automatically every day at 2 AM. You want to use cron to automate this task. A. Which command will you use to open the crontab file and schedule the task? (1 marks) **1. Opening the Crontab File to Schedule the Task** To open the crontab file and schedule a task, you can use the following command: crontab -e **Explanation:** • `crontab -e`: This command opens the user's crontab file in an editor (such as `vi` or `nano` depending on your environment). From here, you can schedule cron jobs. B. What would the cron entry look like to schedule backup.sh to run at 2 AM every day? (2 marks) **2. Cron Entry to Schedule `backup.sh` to Run at 2 AM Every Day** To run the script `backup.sh` every day at 2 AM, add the following line to the crontab file: 0 2 * * * /path/to/backup.sh **Explanation:** • `0 2 * * *`: This cron schedule means "at 2:00 AM every day." Here's the breakdown: ○ `0`: The minute (0th minute). ○ `2`: The hour (2 AM). ○ `*`: Every day of the month. ○ `*`: Every month. ○ `*`: Every day of the week. • `/path/to/backup.sh`: The full path to the `backup.sh` script you want to run. Make sure to specify the absolute path. C. How can you verify that the cron job has been successfully scheduled and is working as expected? (3 marks) **3. Verifying the Cron Job Has Been Scheduled and Is Working** To verify that the cron job has been successfully scheduled and is running as expected, follow these steps: **a) List the Scheduled Cron Jobs** You can list the current user's cron jobs with: crontab -l	CO3,CO3,CO3	Medium - - A

Explanation:

- **`crontab -l`**: This lists all the cron jobs scheduled for the current user. You should see the cron entry for `backup.sh` if it has been scheduled correctly.

b) Check Cron Service Status

Ensure the cron service is running with the following command:

systemctl status cron

Explanation:

- **`systemctl status cron`**: This command checks the status of the cron service. It should be active (running). If the service is not running, cron jobs won't execute.

c) Check Cron Logs

To check whether the job is executing correctly, you can check the cron logs (if logging is enabled):

grep CRON /var/log/syslog

Explanation:

- **`grep CRON /var/log/syslog`**: This searches for cron-related entries in the system log (`/var/log/syslog`). You should see an entry for `backup.sh` when it runs.

d) Manual Verification

To verify manually, you can modify `backup.sh` to log its execution to a file (for debugging):

echo "Backup job ran at $(date)" >> /path/to/backup_log.txt

After the cron job is supposed to have run (2 AM), check the contents of `/path/to/backup_log.txt` to see if the log entry is there.

Summary:

. **Open crontab file**: Use `crontab -e` to edit the crontab file.

. **Cron entry**: Schedule the job with `0 2 * * * /path/to/backup.sh` to run `backup.sh` at 2 AM every day.

Verify cron job:

- List cron jobs with `crontab -l`.
- Check cron service status with `systemctl status cron`.
- Check cron logs using `grep CRON /var/log/syslog`.

		• Add logging to the script for manual verification.		

36	3.1	You are a system administrator responsible for managing large amounts of log files on a Linux server. To streamline the process of finding relevant information in these files, you need to use the find, grep, and wc commands to search for specific content, count occurrences, and analyze the file structure. Complete the following tasks: A. Locate Files by Name: • Write a command to search for all .log files and explain how it works. • Provide the expected output format. (2 marks) **1. Locate Files by Name** **Command:** find / -type f -name "*.log" **Explanation:** • `find`: Command to search for files and directories. • `/`: Start search from the root directory. • `-type f`: Only look for regular files. • `-name "*.log"`: Match files that end with `.log`. **Expected Output Format:** /var/log/syslog.log /home/user/app.log /opt/data/debug.log B. Search for Specific Content in Files: • Write a command to search for "error" in these files and explain the function of the command. • How would you modify the command to perform a case-insensitive search? (2 marks) **2. Search for Specific Content in Files** **Command:** grep "error" $(find / -type f -name "*.log" 2>/dev/null) **Explanation:** • `grep "error"`: Searches for the word **"error"** in files. • `$(find ...)`: Substitutes the list of `.log` files found. • `2>/dev/null`: Suppresses permission denied errors.	CO3,CO3,CO3	Challenging - - A

Case-insensitive Search:

grep -i "error" $(find / -type f -name "*.log" 2>/dev/null)

- `-i`: Ignores case (matches "Error", "ERROR", etc.).

C. Count Lines, Words, and Characters:

- Write a command that pipes the results of the grep search into wc to get the line, word, and character count.
- Explain the purpose of each option used with the wc command.

(2 marks)

3. Count Lines, Words, and Characters

Command:

grep -i "error" $(find / -type f -name "*.log" 2>/dev/null) | wc

Explanation:

- `| wc`: Counts lines, words, and characters from the grep output.

Output example:

15 100 900

-
  - **15** lines matched.
  - **100** words in those lines.
  - **900** total characters.

Each Option in `wc`:

- `-l`: Count lines.
- `-w`: Count words.
- `-c`: Count characters (bytes).

You can run them individually as well:

```
... | wc -l   # just lines
... | wc -w   # just words
... | wc -c   # just characters
```

37	4.1	You are a network administrator at a company that is experiencing connectivity issues. Several users are unable to access certain websites, and some devices are not showing up in the network. Your task is to troubleshoot the network issue using basic networking commands. A. What does the ifconfig command do in Linux? (1 marks) **1. What does the `ifconfig` command do in Linux?** **Answer:** `ifconfig` displays and configures network interfaces. It shows IP addresses, MAC addresses, and interface status. B. If you use the ping command and it returns "Request Timed Out," what could be the reason? (1 marks) **2. If you use the `ping` command and it returns "Request Timed Out," what could be the reason?** **Answer:** It may indicate: • The destination is unreachable or offline. • A firewall is blocking ICMP packets. • Network issues like broken routes or DNS failure. C. What information can the route command provide when troubleshooting network routing issues? (1 marks) **3. What information can the `route` command provide when troubleshooting network routing issues?** **Answer:** `route` shows the kernel's IP routing table. It reveals default gateways, subnet masks, and destination networks—useful for checking if routes are correctly set. D. How can the netstat command help in checking network connections on your system? (1 marks) **4. How can the `netstat` command help in checking network connections on your system?** **Answer:** `netstat` lists open ports, active connections, routing tables, and interface statistics. It helps detect which services are listening or which connections are active. E. How would you use nslookup to check if a DNS server is resolving a domain name correctly? (2 marks) **5. How would you use `nslookup` to check if a DNS server is resolving a domain name**	CO4,CO4,CO4,CO4,CO4	Challenging - - A

		correctly? (2 marks) **Command:** nslookup google.com **Explanation:** It queries the DNS to see if the domain name (`google.com`) resolves to an IP address. If it does, DNS is working correctly. If not, there may be a DNS issue.		
38	4.1	You are the IT administrator of a growing company. The company has critical data stored on several servers, and your manager has asked you to create a backup strategy to ensure the safety of important files and prevent data loss. The company cannot afford to lose any data, and downtime must be minimized in case of an emergency. A. In the given scenario, which backup strategy would you recommend for this company and why? (2 marks) **1. Recommended Backup Strategy (2 marks)** **Answer:** I would recommend a **hybrid backup strategy** combining **incremental daily backups** with **full weekly backups**, using **automated scheduling** and **offsite/cloud storage**. ● **Why?** It ensures regular backups with minimal data loss and fast recovery, while offsite copies protect against disasters. B. What factors should you consider when selecting a backup medium for the company's data? (2 marks) **2. Factors to Consider When Selecting a Backup Medium (2 marks)** **Answer:** ● **Data volume and growth rate** ● **Cost and budget** ● **Speed of backup and recovery** ● **Security and encryption support** ● **Accessibility and reliability** ● **Compliance with regulations** C. Based on the company's needs, what backup medium would you choose (e.g., external hard drives, cloud storage, etc.) and why? (2 marks) **3. Recommended Backup Medium and Why (2 marks)** **Answer:** I recommend using **cloud storage** (e.g., AWS S3, Google Cloud, or Azure). ● **Why?** Cloud storage is scalable, secure, accessible from anywhere, supports automation, and offers high durability with minimal downtime.	CO4,CO4,CO4	Challenging - - A

39	4.1	You have a large text file containing server logs that need to be compressed for storage. The file is not critical, and some loss of precision is acceptable in exchange for better compression ratios. The client requests a compression format that achieves a good balance between compression ratio and speed. A. Describe how you would use the gzip command to compress the text file. After compressing the file, the client needs to extract and view the log data. What command would you use to decompress the file and view the contents directly in the terminal without saving the decompressed file? Provide an explanation of the command used. (6 marks) ## A. Compressing and Viewing a Text File Using `gzip` (6 marks) ### 1. Compressing the Log File To compress the text file using `gzip`, use the following command: gzip server_logs.txt **Explanation:** • `gzip` compresses the file `server_logs.txt` and replaces it with `server_logs.txt.gz`. • It uses the DEFLATE algorithm, offering a good balance between compression ratio and speed. • Original file is removed after compression by default. --- ### 2. Viewing Contents Without Decompressing to Disk To view the contents of the compressed file directly in the terminal: zcat server_logs.txt.gz **Explanation:** • `zcat` is like `cat` but for `.gz` files. • It decompresses the file in-memory and displays the content in the terminal. • It **does not** save the decompressed file to disk — ideal for quick viewing without extraction.	CO4	Medium - Analysing - T
40	4.1	You are managing a Linux-based system that supports multiple users in a corporate environment. The system needs to enable users to communicate with each other efficiently through different methods, such as messaging, file sharing, and notifications. As an administrator, you must configure and monitor communication tools to ensure proper functionality and security. Explain how you would configure and use the write and wall commands to allow users to send messages to other users or all users on the system. ## 1. `write` Command – Send a Message to a Specific User ### Usage:	CO4	Easy - Analysing - T

		write username **Example:** write alice You can now type a message. Press `Ctrl+D` to end the message. **Configuration Notes:** • The target user must be logged in. The target user's terminal must allow messages. Use: mesg y • This enables messages. If they run `mesg n`, they will block messages. ## 2. `wall` Command – Broadcast a Message to All Users **Usage:** wall "System will reboot in 5 minutes" **Alternative (Interactive):** wall Then type the message and end with `Ctrl+D`. **Configuration Notes:** • Sends the message to all logged-in users. • Useful for system-wide announcements. • Messages are sent only to terminals that allow messages (`mesg y`).		
41	4.1	You are managing a multi-user Linux server where employees often need to communicate in real-time. The talk command is available on the system and can be used to allow users to have interactive, real-time conversations. However, you need to ensure that the talk service is properly configured and secure, as not all users should have access to this communication tool. Explain how the talk command works to allow two users to communicate interactively on a Linux system. Include the basic steps for initiating and accepting a talk session. The `talk` command in Linux enables **real-time, interactive text communication** between two users logged into the same or different systems, provided the service is properly configured and allowed. --- ## 🧩 How `talk` Works: • `talk` establishes a **split-screen session**. • Each user types in their half of the screen, and the other can see	CO4	Easy - Analysing - T

it immediately.

- It uses **UDP port 517** and requires the **`talk` daemon (`talkd`)** to be running on both systems.

🔄 Basic Steps to Use `talk`:

✅ Step 1: Ensure `talk` Is Installed

sudo apt install talk

✅ Step 2: Allow Messaging

Both users must allow messages on their terminals:

mesg y

✅ Step 3: Initiate a Talk Request

talk username[@hostname]

Example (same system):

talk john

Example (remote system):

talk john@server.company.com

✅ Step 4: Accept a Talk Request

The recipient gets a prompt like:

Message from Talk_Daemon@host... talk: connection requested by user@host.
talk: respond with: talk user@host

They respond with:

talk user@host

🔐 Security & Access Control:

Restrict access: You can remove execute permissions for `talk` from unauthorized users using:

chmod o-x /usr/bin/talk

- **Audit usage:** Monitor talk-related messages in system logs.
- **Firewall config:** Allow UDP 517 if used across networked systems.
- **mesg restrictions:** Users can opt out with `mesg n`.

42	4.1	Different departments require different types of FTP connections depending on their security needs and network configuration. Some users need basic FTP access, while others require secure connections for transmitting sensitive data. As the administrator, you must ensure the FTP server is configured to handle both regular and secure FTP connections. Explain the two primary types of FTP connections: Active FTP, and Passive FTP. How do these different connection types impact the FTP communication process? In FTP (File Transfer Protocol), there are **two primary connection modes** that determine how data is exchanged between client and server: **Active FTP** and **Passive FTP**. Understanding these modes is crucial for configuring FTP servers to accommodate different network environments and security requirements. --- ## 🔄 1. Active FTP (PORT Mode) • **Client initiates a connection** to the **server's command port (TCP 21)**. • The **server initiates a connection back** to the client on a random high-numbered **data port**. • The client must **allow incoming connections**, which can be blocked by firewalls or NAT (Network Address Translation). **Impact:** • **Less firewall-friendly**, especially for clients behind NAT or firewalls. • Suitable when the client's network allows incoming connections. **How it works:** Client -> Server : TCP 21 (Command) Server -> Client : Random TCP port (Data) --- ## 📤 2. Passive FTP (PASV Mode) • **Client initiates both connections**: command (TCP 21) and data. • The **server opens a random port** for data and tells the client which port to connect to. • No need for the server to initiate any connection back to the client. **Impact:** • **Firewall- and NAT-friendly**. • Preferred for **secure or complex network environments**. **How it works:** Client -> Server : TCP 21 (Command) Client -> Server : Random TCP port (Data)	CO4	Medium - Analysing - T

🔐 Summary of Differences

Feature	Active FTP	Passive FTP
Data Channel	Server → Client	Client → Server
Firewall/NAT	Difficult (incoming to client)	Easier (all from client)
Default Mode	Traditional/Default	Modern/Preferred
Security	Can expose client ports	Safer for clients behind NAT

As an admin, enabling **both modes** on the FTP server allows flexibility, but **passive mode is generally more secure and reliable** in modern network setups. You can configure your FTP server (like vsftpd or proftpd) to support both. Let me know if you want a sample configuration file.

43	4.1	You are tasked with configuring a system where you need to send automated emails with specific requirements from a Linux server. The system should send emails with attachments, as well as to multiple recipients using CC (carbon copy) and BCC (blind carbon copy) fields. You are using the mail command to send these emails directly from the terminal. A. How would you send an email with an attachment, a specific subject, and specify recipients in the CC field using the mail command? (3 marks)	CO4,CO4	Medium - Applying - A

A. Sending Email with Attachment and CC using `mail` (3 marks)

To send an email with an **attachment**, a **subject**, and include **CC recipients**, use:

mail -s "Subject Text" -c cc@example.com -a /path/to/attachment.txt recipient@example.com

- `-s "Subject Text"`: Sets the subject of the email.
- `-c cc@example.com`: Adds a carbon copy recipient (visible to all).
- `-a /path/to/attachment.txt`: Attaches a file.
- `recipient@example.com`: Main recipient.

B. Explain how to send an email with an attachment and include recipients in the BCC field using the mail command. What is the difference between CC and BCC, and why would you use BCC in certain scenarios? (3 marks)

B. Sending Email with Attachment and BCC using `mail` (3 marks)

		To send an email with **attachment** and **BCC**, use: mail -s "Subject Text" -b bcc@example.com -a /path/to/attachment.txt recipient@example.com • `-b bcc@example.com`: Sends a blind carbon copy (hidden from other recipients). --- ## Difference Between CC and BCC: **Field** | **Visible to Others?** | **Purpose** **CC** (Carbon Copy) | ✅ Yes | Inform others; visible to all. **BCC** (Blind Carbon Copy) | ❌ No | Keep recipients private; useful for confidentiality. **Use BCC** when: • You want to **hide recipient emails**. • You're **emailing a large group** without revealing all addresses.		
44	4.1	You are the system administrator for a Linux-based server that hosts several applications, including a web server (Apache) and a database server (MySQL). The server needs to be secured using a firewall to prevent unauthorized access while allowing legitimate traffic to reach the necessary services. You are tasked with configuring the firewall to allow specific services while blocking everything else by default. Describe how you would use firewalld to achieve the configuration - allowing HTTP and HTTPS while blocking other traffic ,and compare its advantages over iptables. To **secure your Linux server using `firewalld`**, while **allowing only HTTP (port 80) and HTTPS (port 443)** traffic and **blocking everything else**, follow these steps: --- ## ✅ Steps to Configure `firewalld` 1. **Start and Enable firewalld**: sudo systemctl start firewalld sudo systemctl enable firewalld 2. **Set the default zone (optional but recommended)**: sudo firewall-cmd --set-default-zone=public 3. **Allow HTTP and HTTPS traffic**: sudo firewall-cmd --zone=public --add-service=http --permanent sudo firewall-cmd --zone=public --add-service=https --permanent 4. **Reload the firewall to apply changes**:	CO4	Medium - Applying - A

sudo firewall-cmd --reload

5. **(Optional) Verify rules**:

   sudo firewall-cmd --list-all

🛡️ What's Happening?

- Only **HTTP** and **HTTPS** traffic are allowed.
- By **default**, all other ports/services are **blocked**.
- The `--permanent` flag ensures changes persist after reboot.

🔄 firewalld vs iptables: Key Advantages

Feature	`firewalld`	`iptables`
Ease of Use	High-level, service-based commands	Low-level, rule-by-rule handling
Zones	Supports zone-based policies	No built-in zone concept
Dynamic Rules	Changes rules **without restart**	Requires restarting service
Rich Rules	Provides abstractions (services, interfaces)	Manual rule definitions
Better for Automation	Yes (D-Bus API available)	Less user-friendly for automation

✅ **Conclusion**:
`firewalld` is more **modern**, **manageable**, and **dynamic** than `iptables`, making it ideal for **real-time service management** on production servers.

45	4.1	You are managing a Linux-based network that includes both internal (private) and external (public) IP addresses. The network is configured with multiple subnets, and several servers are hosted behind a firewall and a Network Address Translation (NAT) device. Your goal is to ensure that the internal network can access external resources, while also securing the internal network by keeping it isolated from the public internet. What is the difference between a private IP address and a public IP address in a network setup? How would these addresses typically be used in a Linux-based network environment?	CO4	Medium - Analysing - A

✅ Difference Between Private and Public IP Addresses

Feature	Private IP Address	Public IP Address
Scope	Used within internal networks (LAN)	Used to identify devices on the public internet
IP Ranges	- 10.0.0.0 – 10.255.255.255 - 172.16.0.0 – 172.31.255.255 - 192.168.0.0 – 192.168.255.255	Assigned by ISPs (outside of private ranges)
Routable on Internet?	❌ Not routable (requires NAT to reach internet)	✅ Routable on internet
Security	More secure by default (isolated)	Exposed to public access unless firewalled
Cost	Free (not registered)	May incur cost (must be globally unique)

🛠️ How These Addresses Are Used in a Linux-Based Network

1. Private IP Usage

- Assigned to internal systems like servers, workstations, and printers.
- Helps keep internal traffic **isolated** from the public internet.
- Example:
  - Web server on `192.168.1.100`
  - Database server on `192.168.1.101`
- Internal communication (e.g., between web and DB servers) uses these private IPs.

2. Public IP Usage

		• Assigned to the firewall/router or NAT device connecting the internal network to the internet. • Acts as the **gateway** for outbound and inbound traffic. • The Linux server may receive a public IP if it needs to be directly accessible (e.g., public web server), but this should be protected via firewall. **3. NAT (Network Address Translation)** • Converts **private IP addresses to a public IP** when internal devices access the internet. • Ensures that many private devices can **share a single public IP**. • Linux routers or firewalls (using `iptables` or `nftables`) often perform this NAT function. --- 🧠 **Summary** • **Private IPs**: Used internally, secure, not routable on the internet. • **Public IPs**: Used to access the internet or expose services. • **Linux networks** use private IPs for internal communication and NAT/firewall to access external resources while staying secure.		
46	4.1	You are a Linux system administrator tasked with maintaining the network configuration and troubleshooting network-related issues for a Linux-based server. The server is connected to a local area network (LAN) and requires access to both internal resources and external services via the internet. You need to verify the current network configuration, diagnose connectivity issues, and make adjustments to the network interface if needed. Explain how you would use the following Linux networking commands to troubleshoot and manage network connectivity: • ifconfig • ip • ping • traceroute • netstat • nslookup **1. `ifconfig` / `ip`** • **Purpose**: These commands help you view and configure network interfaces on the system. **`ifconfig`** • **Usage**: `ifconfig` is an older command (deprecated but still widely used) for displaying and managing network interfaces. • **Verify IP addresses and network interfaces**:	CO4	Challenging - Evaluating - A

ifconfig

- This will list all active network interfaces along with details like IP addresses, netmasks, and MAC addresses.

Enable/Disable an interface:
To bring an interface up:

sudo ifconfig eth0 up

To bring an interface down:

sudo ifconfig eth0 down

Set a static IP:

sudo ifconfig eth0 192.168.1.100 netmask 255.255.255.0 up

`ip`

- **Usage**: `ip` is the modern replacement for `ifconfig` and provides a more robust and flexible set of options.

View network interfaces:

ip addr show

- **Bring up/down an interface**:

To bring an interface up:

sudo ip link set eth0 up

To bring an interface down:

sudo ip link set eth0 down

Assign an IP address:

sudo ip addr add 192.168.1.100/24 dev eth0

2. `ping`

- **Purpose**: Used to check the availability of a network host (either a local or remote machine) and measure the round-trip time for packets.

Usage:To test connectivity to a remote server (e.g., Google's DNS server):

ping 8.8.8.8

- This command sends ICMP Echo Request packets to the target and displays the responses.

To test connectivity to a hostname (e.g., `google.com`):

ping google.com

- **Stop the ping** by pressing `Ctrl + C`.

- **Troubleshooting**: If you get "Request Timed Out" or no response, the issue might be with the network, firewall, or the target device.

3. `traceroute`

- **Purpose**: This command helps you track the route taken by packets to reach a destination and diagnose where delays or packet losses occur.
  Usage:

  To trace the route to `google.com`:

  traceroute google.com

  - **Troubleshooting**: If a route hangs at a specific hop, it indicates where packets are getting delayed or dropped, helping you identify the problematic network segment.

4. `netstat`

- **Purpose**: `netstat` provides detailed information about network connections, routing tables, and network statistics, allowing you to see which ports are open and the status of active network connections.
  Usage:

  List all active connections:

  netstat -tuln

  - This shows all listening ports and active connections (TCP/UDP) with associated IP addresses and port numbers.

Check for open connections on a specific port:

netstat -an | grep ':80'

- This shows all connections involving port 80 (HTTP).

Show routing table:

netstat -r

- **Troubleshooting**: You can check if the desired ports are open and which services are using them. If no service is listening on a specific port, it might indicate an issue with the service.

5. `nslookup`

- **Purpose**: `nslookup` is used for querying DNS to obtain domain name or IP address mappings and troubleshooting DNS issues.
    - **Usage**:

To resolve a domain name to an IP address:

nslookup google.com

- **Troubleshooting DNS**: If `nslookup` fails, it could indicate a DNS configuration issue or a problem with the DNS server.

Reverse lookup:

nslookup 8.8.8.8

- This will query the DNS to resolve the given IP address back to a domain name.

Example Troubleshooting Workflow

1. **Check Network Configuration**:
    - First, use `ip addr show` (or `ifconfig`) to check the network interfaces' IP addresses.
2. **Verify Connectivity**:
    - Use `ping` to check if you can reach a known server (e.g., Google DNS `8.8.8.8` or a local host).
3. **Diagnose Path**:
    - If ping fails, use `traceroute` to find where the issue is occurring in the path to the destination.
4. **Check Network Connections**:
    - Use `netstat` to ensure that services are listening on the correct ports and check for any unusual connections.
5. **Check DNS Resolution**:
    - Use `nslookup` to verify that DNS resolution is working as expected, especially if you're unable to reach websites by hostname.

47	4.1	You are managing a Linux server that stores critical company data. As part of your disaster recovery plan, you need to implement an efficient and reliable backup strategy. The server contains large amounts of data, and you need to select the appropriate backup medium considering factors such as cost, speed, data volume, and access needs. You have several options, including external hard drives, network-attached storage (NAS), cloud storage, and tape backups. A. How would you determine whether a cloud storage solution is suitable for backing up your server data? (2 marks) To determine whether a cloud storage solution is suitable for backing up server data, you can consider these factors: 1. **Cost**: Cloud storage is often charged based on the amount of data you store and the frequency of access. Compare this to other solutions like external hard drives or NAS. 2. **Speed and Bandwidth**: Uploading large data to the cloud can be slow if you have a slow internet connection, so consider the time it would take to backup or restore data. 3. **Access**: Cloud storage allows easy remote access to data, making it a good option for off-site backups or for accessing your data from multiple locations. 4. **Reliability**: Ensure that the cloud service offers high availability and data redundancy to avoid data loss. B. What are the advantages and disadvantages of using external hard drives for backups compared to network-attached storage (NAS)? (4 marks) **External Hard Drives vs. Network-Attached Storage (NAS)**: 1. **External Hard Drives**: ○ **Advantages**: ■ Simple and inexpensive. ■ No need for network infrastructure. ■ Portable and easy to move. ○ **Disadvantages**: ■ Limited capacity compared to NAS. ■ Less secure—if lost or damaged, data may be unrecoverable. ■ Requires manual connection to your system. 2. **Network-Attached Storage (NAS)**: ○ **Advantages**: ■ Offers centralized storage and allows multiple devices to access it over a network. ■ More secure, often with RAID support to protect against disk failure. ■ Easier to scale and expand as data grows. ○ **Disadvantages**:	CO4,CO4	Medium - - A

		■ More expensive. ■ Requires network setup and maintenance. ■ Less portable than external hard drives.		
48	4.1	You are managing a Linux server that is used for file sharing among different departments in the company. The server uses FTP to transfer files between the server and client machines. You need to perform various operations like retrieving and uploading multiple files, as well as creating and removing directories on the FTP server. Explain how to connect to an FTP server in Linux and perform the following operations: • Get and Mget (Download files from the FTP server) • Put and Mput (Upload files to the FTP server) • Mkdir and Rmdir (Create and remove directories on the FTP server) To manage file transfers and directories on an FTP server from a Linux client, you can use the `ftp` command. Here's a guide to connect to the FTP server and perform the necessary operations. **1. Connecting to an FTP Server** To connect to the FTP server, use the following command in your terminal: ftp <ftp-server-address> For example, if your FTP server's address is `ftp.example.com`, you would use: ftp ftp.example.com You will be prompted to enter your username and password to authenticate. **2. Get (Download a Single File)** To download a single file from the FTP server, use the `get` command: get <remote-file> <local-file> • `<remote-file>`: The name of the file you want to download from the server. • `<local-file>`: The name you want to save the file as on your local machine (optional, it will default to the same name). For example: get example.txt This will download the file `example.txt` from the FTP server to your current local directory. **3. Mget (Download Multiple Files)**	CO4	Medium - Applying - A

To download multiple files, use the `mget` command:

```
mget <file-pattern>
```

- `<file-pattern>`: Use a wildcard (e.g., `*.txt`) to specify multiple files.

For example, to download all `.txt` files:

```
mget *.txt
```

You will be prompted to confirm each file download, but you can press `a` to download all files automatically.

4. Put (Upload a Single File)

To upload a single file to the FTP server, use the `put` command:

```
put <local-file> <remote-file>
```

- `<local-file>`: The path to the file you want to upload.
- `<remote-file>`: The name to save the file as on the FTP server (optional, it will default to the same name).

For example, to upload `example.txt`:

```
put example.txt
```

5. Mput (Upload Multiple Files)

To upload multiple files at once, use the `mput` command:

```
mput <file-pattern>
```

For example, to upload all `.txt` files:

```
mput *.txt
```

You will be prompted to confirm each file upload, but you can press `a` to upload all files automatically.

6. Mkdir (Create a Directory on the FTP Server)

To create a new directory on the FTP server, use the `mkdir` command:

```
mkdir <directory-name>
```

		For example, to create a directory called `new_folder`: mkdir new_folder ## 7. Rmdir (Remove a Directory on the FTP Server) To remove an empty directory on the FTP server, use the `rmdir` command: rmdir <directory-name> For example, to remove the directory `old_folder`: rmdir old_folder ## Summary of Commands: - **get** : Download a single file. - **mget** : Download multiple files. - **put** : Upload a single file. - **mput** : Upload multiple files. - **mkdir** : Create a new directory on the server. - **rmdir** : Remove an empty directory on the server. ## Exiting the FTP Session When you're done with your FTP session, you can exit by typing: bye Or: quit		
49	5.1	A colleague has sent you an RPM package, and you need to ensure it is legitimate and hasn't been tampered with. A. How would you verify the integrity of the RPM package using RPM commands to ensure it has not been corrupted? (2 marks) ## 1. Verifying the Integrity of the RPM Package Using RPM Commands To verify the integrity of an RPM package, you can use the `rpm` command to check the package's checksum, which ensures that the package hasn't been corrupted or altered. The command for verifying an RPM package is: rpm -K <package-name>.rpm	CO5,CO5,CO5	Challenging - - A

- `<package-name>.rpm`: Replace this with the name of the RPM package file you received.

The `-K` option checks the package's signature and integrity. If the package is legitimate and hasn't been tampered with, you will see an output like:

<package-name>.rpm: md5 OK

If the package has been corrupted or altered, you'll see an error message instead, such as:

<package-name>.rpm: FAILED (md5).

This indicates that the checksum doesn't match, and the package might have been tampered with.

B. How can you check the GPG signature of the package to confirm its authenticity? (2 marks)

2. Checking the GPG Signature of the Package

RPM packages often come with a GPG (GNU Privacy Guard) signature to confirm their authenticity and ensure that they were signed by a trusted source. To check the GPG signature of an RPM package, you can use the following command:

rpm --checksig <package-name>.rpm

- `<package-name>.rpm`: Replace this with the name of the RPM package file.

If the GPG signature is valid and the package is authentic, you will see:

<package-name>.rpm: rsa sha1 (key ID <key-id>) signed <date>

If the GPG signature is invalid or not found, you will see something like:

<package-name>.rpm: ** NOT YET SIGNED **

or

<package-name>.rpm: signature verification failed

C. If the signature check fails, what steps would you take to diagnose the issue and resolve it? (2 marks)

3. Diagnosing and Resolving Issues When the Signature Check Fails

If the GPG signature check fails, here's how you can diagnose and resolve the issue:

a. Ensure the GPG Key is Imported

The package may be signed with a GPG key that is not yet imported into your system. You can import the missing key by obtaining it from a trusted source (e.g., the software vendor or repository). You can import the key using the `rpm` command:

```
rpm --import /path/to/gpg/key
```

Alternatively, you might download the key from a repository or vendor's website and import it.

b. Verify the Package Source

If the GPG key is correctly imported and the signature still fails, verify that the package was obtained from a trusted source. Ensure that you are downloading RPMs from trusted, secure repositories. If you suspect the package is from an untrusted or unreliable source, do not install it.

c. Re-download the Package

If the integrity check fails, and you are certain the GPG key is correct, it's possible that the package was corrupted during download. You should re-download the package from a secure source and verify the GPG signature again.

d. Manually Verify the Key

If the signature check still fails, manually verify that the correct public key is associated with the software vendor or repository. Sometimes, you may need to install the public key from an additional repository or update it.

Summary:

- **Integrity Check**: Use `rpm -K <package-name>.rpm` to verify the package's integrity.
- **GPG Signature Check**: Use `rpm --checksig <package-name>.rpm` to check the authenticity of the package.
- **Troubleshooting**:
    - **Import Missing Keys**: Use `rpm --import <key-path>` to import missing GPG keys.
    - **Re-download Package**: If corruption is suspected, re-download the package.
    - **Verify Trusted Sources**: Ensure the package is from a reliable, trusted source.

50	5.1	You notice that a package on your system isn't working as expected, and you need to troubleshoot the issue. A. What RPM command would you use to query the installed package and check for any possible issues, such as missing files or dependencies? (2 marks) **1. Querying the Installed Package for Issues** To query an installed RPM package and check for any possible issues, such as missing files or dependencies, you can use the following RPM command: rpm -q <package-name> • Replace `<package-name>` with the name of the installed package. This will show you the version of the package that is installed. If the package is missing or not installed, you'll get a message like: package <package-name> is not installed If you suspect missing files or issues within the package, you can query for specific files installed by the package: rpm -ql <package-name> This lists all the files that were installed by the package. You can check for missing or incorrectly installed files here. Additionally, you can check for missing dependencies with: rpm -qR <package-name> This will show you the required dependencies for the package. If any dependencies are missing, they will be listed here. B. How would you use RPM to check if the package is properly installed and verify its integrity? (2 marks) **2. Verifying the Integrity of the Package** To verify that a package is properly installed and check its integrity (i.e., ensure no files have been altered or corrupted), you can use the following command: rpm -V <package-name> The `-V` (verify) option compares the installed files to the RPM database and shows any discrepancies. If everything is intact, there will be no output. If there are issues (such as missing or altered files), it will output a list of problems, indicating which files have discrepancies. For example, the output might look like this if a file is missing or corrupted: S.5....T. c /etc/someconfig.conf	CO5,CO5,CO5	Challenging - - A

Where:

- `S` means the file size has changed.
- `T` means the modification time is different.

C. If there are any missing dependencies or conflicts, how would you resolve them using RPM tools? (2 marks)

3. Resolving Missing Dependencies or Conflicts

If you find missing dependencies or conflicts with the installed package, you can resolve them using RPM tools in the following ways:

a. Resolve Missing Dependencies

To check for missing dependencies, you can use the `-qR` option (as shown above). If any dependencies are missing, you can install them with RPM or `yum` (or `dnf` for newer versions of Linux). For example:

```
yum install <missing-package-name>
```

or, for systems using `dnf`:

```
dnf install <missing-package-name>
```

If the dependencies are not available from the default repository, you may need to find the appropriate repository or manually download and install the required RPM packages.

b. Resolve Conflicts

If you suspect conflicts between packages (such as multiple versions of the same package), you can use the following command to check for package conflicts:

```
rpm -qa | grep <package-name>
```

This will list all installed versions of the specified package. If there are multiple conflicting versions, you can remove the unnecessary ones using:

```
rpm -e <conflicting-package-name>
```

Alternatively, if you're dealing with dependency issues, you might want to reinstall the package to ensure it is correctly installed:

```
rpm -Uvh <package-name>.rpm
```

This will upgrade the package if it’s already installed or install it if it’s not.

Summary of Solutions:

1. **Query Installed Package**: Use `rpm -q <package-name>` to

		query installed packages and `rpm -ql <package-name>` to list the files installed by the package. 2. **Verify Integrity**: Use `rpm -V <package-name>` to check the integrity of the package and verify if any files have been altered. 3. **Resolve Issues**: ○ **Missing Dependencies**: Use `yum install <missing-package-name>` or `dnf install <missing-package-name>`. ○ **Conflicts**: Use `rpm -e <conflicting-package-name>` to remove conflicting packages, or reinstall with `rpm -Uvh <package-name>.rpm`.		
51	5.1	You need to remove an outdated package called example-old-1.0-1.x86_64.rpm to make room for a newer version. However, you are concerned about the potential impact on other packages that might depend on it. A. What RPM command would you use to remove the old version of the example-old package from your system? Explain the steps to ensure that the removal is safe. (3 marks) **1. Removing the Outdated Package** To safely remove the outdated package `example-old-1.0-1.x86_64.rpm`, you can use the `rpm` command as follows: rpm -e example-old-1.0-1.x86_64 ● The `-e` option stands for **erase**, which removes the specified package from your system. ● Replace `example-old-1.0-1.x86_64` with the actual name of the package. **Steps to Ensure Safe Removal:** 1. **Check Dependencies**: Before removing the package, it is crucial to ensure that no other packages depend on it to avoid breaking other packages or software on the system. You should perform a dependency check to see if any other packages are linked to `example-old`. 2. **Backup the Package (Optional)**: If you are uncertain, you can back up the package or create a snapshot of the system, so you can easily revert changes if anything goes wrong. 3. **Remove the Package**: After verifying that no critical dependencies exist, proceed with the removal using the command above. 4. **Install New Version (if required)**: After safely removing the old version, you can proceed to install the new version of the package. B. Before removing the package, how would you check if any other packages depend on example-old-1.0-1.x86_64.rpm? (3 marks)	CO5,CO5	Challenging - - A

		## 2. Checking for Dependent Packages Before removing the package, you can check if any other packages depend on `example-old-1.0-1.x86_64.rpm` using the following `rpm` command: rpm -q --whatrequires example-old-1.0-1.x86_64 - The `-q` option queries the system for the package. - The `--whatrequires` option lists all packages that require the specified package as a dependency. If the output shows that other packages depend on `example-old`, you'll need to handle these dependencies before removal. If the output is empty, it means no other packages depend on `example-old`, and you can safely remove it. ### Example Output: If no package depends on `example-old`, the output will be empty or state that no packages require it. Example: No package requires example-old-1.0-1.x86_64 - If there are dependent packages, the output will list the names of those packages. Example: package1-2.3-1.x86_64 package2-4.0-2.x86_64 --- ## Summary of Solutions: 1. **To remove the old package**: Use `rpm -e example-old-1.0-1.x86_64`. 2. **To check if other packages depend on it**: Use `rpm -q --whatrequires example-old-1.0-1.x86_64`.		
52	5.1	You are trying to install the package-1.0-1.x86_64.rpm on a system, but RPM reports that several dependencies are missing. You need to resolve this issue while maintaining system stability. A. What steps would you take to identify and resolve the missing dependencies for the package-1.0-1.x86_64.rpm? Include commands and tools you would use to find the missing dependencies. (3 marks) ## 1. Identifying and Resolving Missing Dependencies To resolve missing dependencies for `package-1.0-1.x86_64.rpm`, you	CO5,CO5	Challenging - - A

can follow these steps:

Step 1: Attempt to Install the Package

First, try to install the package to see which dependencies are missing. Run:

```
rpm -ivh package-1.0-1.x86_64.rpm
```

This will show an error message indicating which dependencies are missing. It might look like this:

```
error: Failed dependencies:
    dependency1 is needed by package-1.0-1.x86_64
    dependency2 is needed by package-1.0-1.x86_64
```

Step 2: Check for Missing Dependencies

Once you know which dependencies are missing, you can check if those dependencies are available on your system or need to be installed manually.

Use the `rpm` command to check if the dependency is already installed:

```
rpm -q dependency1
rpm -q dependency2
```

1.
    - If the dependencies are installed, you will see the version of the package.
    - If the dependency is missing, you will see an output like `package 'dependency1' is not installed`.

Use a Package Manager (YUM or DNF): If dependencies are missing, you can use YUM (for RHEL/CentOS) or DNF (for Fedora) to automatically resolve and install them from the repositories:

```
sudo yum install dependency1 dependency2
```

or for Fedora:

```
sudo dnf install dependency1 dependency2
```

2.

Check for Available Dependencies in Repositories: If the required dependencies are not available on your system, search for them in your package repositories:

```
yum search dependency1
```

or:

```
dnf search dependency1
```

3.

Manual Installation: If the dependencies are not available via YUM or DNF, you may need to manually download the RPMs for these dependencies from a trusted source or vendor's website and install them:

rpm -ivh dependency1.rpm

4. Repeat this for each missing dependency.

B. How would you ensure that the installation of these missing dependencies does not conflict with or break any existing packages? (3 marks)

2. Ensuring No Conflicts or Breakage of Existing Packages

To ensure the installation of dependencies does not break or conflict with existing packages, you can take the following steps:

Step 1: Use `yum` or `dnf` to Resolve Dependencies

Using YUM or DNF automatically checks for dependencies and resolves conflicts. They will not install conflicting packages unless you explicitly force it. If a conflict occurs, they will display an error message.

Installing Dependencies Using `yum` or `dnf`: As mentioned earlier, using `yum` or `dnf` to install the missing dependencies will automatically manage conflicts:

```
sudo yum install dependency1 dependency2
```
or:
```
sudo dnf install dependency1 dependency2
```

1. These package managers will handle dependency resolution and avoid installing packages that would cause conflicts.

Step 2: Use the `--test` Option (YUM/DNF)

You can test the installation of the dependencies first to ensure there are no conflicts before actually installing them. For example:

```
sudo yum install --test dependency1 dependency2
```

or:

```
sudo dnf install --test dependency1 dependency2
```

This will simulate the installation process without actually applying any changes, allowing you to review potential conflicts.

Step 3: Use `rpm` to Verify Package Integrity

If you are manually installing RPM packages (outside of YUM/DNF), use the `rpm -V` command to verify the integrity of the installed packages and check for conflicts.

```
rpm -V package-1.0-1.x86_64.rpm
```

This will verify the files in the package and show any mismatches or conflicts with existing packages.

		Step 4: Backup and Snapshot (Optional) If you're working on a critical system, it's always a good idea to create a backup or snapshot of your system before installing new dependencies. This way, you can easily revert to a working state if anything goes wrong. --- ## Summary of Steps: 1. **Identify Missing Dependencies**: ○ Use `rpm -ivh` to install and identify missing dependencies. ○ Use `rpm -q` to check if dependencies are already installed. ○ Use YUM or DNF to install the missing dependencies. ○ If not available, manually download and install RPMs. 2. **Ensure No Conflicts**: ○ Use YUM or DNF for automatic conflict resolution. ○ Use the `--test` option with YUM/DNF to simulate the installation. ○ Use `rpm -V` to verify package integrity. ○ Create backups or snapshots if working in a critical environment.		
53	5.1	After installing the mysql-server package using RPM, you encounter an issue where the MySQL service fails to start. You suspect that the installation might have missing dependencies or configuration problems. A. What RPM command would you use to check if the mysql-server package has any missing dependencies or problems with the installation? Provide the exact syntax for the command. (3 marks) ## 1. Check for Missing Dependencies or Installation Problems To check if the `mysql-server` package has missing dependencies or installation problems, you can use the following `rpm` command: rpm -qR mysql-server ### Explanation: - `rpm -qR <package_name>`: This command queries the package and lists all the dependencies (required packages) for the specified package. It will show you if any dependencies are missing or if there are issues with the installation. If you see missing dependencies listed, it means that those packages are required for MySQL to function properly. ---	CO5,CO5	Medium - - A

B. If the RPM diagnostics report missing dependencies, how would you resolve these issues while minimizing the risk of breaking other packages on the system? (3 marks)

2. Resolving Missing Dependencies

If the RPM diagnostics report missing dependencies, you can follow these steps to resolve them without breaking other packages on the system:

Step 1: Verify the Missing Dependencies

Once you know which dependencies are missing (from the `rpm -qR mysql-server` output), use the `rpm` command to check if those packages are available on your system:

```
rpm -q <missing_dependency>
```

If the package is not installed, you'll get an output like:

```
package <missing_dependency> is not installed
```

Step 2: Install the Missing Dependencies

To install the missing dependencies, you can use `yum` or `dnf` (depending on your distribution). These tools will ensure that the dependencies are resolved properly and will handle potential conflicts. For example:

```
sudo yum install <missing_dependency>
```

Or for Fedora-based systems:

```
sudo dnf install <missing_dependency>
```

Step 3: Check the System for Existing Package Conflicts

If you manually install a missing dependency, it is crucial to make sure that you are not conflicting with already installed packages. To do this, you can use:

```
sudo yum check
```

or:

```
sudo dnf check
```

These commands check the system for package issues like conflicts or broken dependencies.

Step 4: Reinstall MySQL Server Package

If you resolved all missing dependencies, you can reinstall the `mysql-server` package to ensure that it is installed correctly and all dependencies are satisfied:

```
sudo rpm -Uvh mysql-server.rpm
```

		This will upgrade or install the `mysql-server` package while ensuring it gets reinstalled properly. **Step 5: Verify and Start MySQL Service** Once dependencies are resolved, verify that MySQL starts correctly: sudo systemctl start mysqld If it starts without issues, you've successfully resolved the problem. You can also check the status with: sudo systemctl status mysqld **Step 6: Test System Stability** After installing the missing dependencies, test the system to make sure everything works and that no other packages are affected by the installation of the dependencies. --- **Summary:** . **Check Missing Dependencies:** Use `rpm -qR mysql-server` to list the dependencies and check for missing ones. . **Resolve Dependencies:** • Use `yum` or `dnf` to install missing dependencies. • Check for conflicts using `yum check` or `dnf check`. • Reinstall the `mysql-server` package if necessary. . **Ensure Stability:** • Verify the service with `systemctl start mysqld` and `systemctl status mysqld` to ensure MySQL starts correctly.		
54	5.1	You have received the example-2.0-1.x86_64.rpm package from a third-party vendor. You want to verify that the package is legitimate and hasn't been tampered with by checking its GPG signature before installation. A. What command would you use to check the GPG signature of the example-2.0-1.x86_64.rpm package, and why is this important? (3 marks) **1. Check the GPG Signature of the RPM Package** To verify the GPG signature of the `example-2.0-1.x86_64.rpm` package, you can use the following command: rpm --checksig example-2.0-1.x86_64.rpm **Explanation:**	CO5,CO5	Medium - - A

- `rpm --checksig <package_file>`: This command checks the GPG signature of the specified RPM package to ensure that it was signed by a trusted source.
- The output will indicate whether the signature is **valid** or **invalid**. If the signature is valid, it means the package has not been tampered with and comes from the expected vendor.

Why Is This Important?

Verifying the GPG signature is crucial because it ensures that:

- The package comes from a legitimate source (the vendor).
- The package has not been altered or tampered with during transfer.
- This adds a layer of security, protecting against the installation of malicious software.

B. If the signature check fails, what steps would you take to either obtain a valid package or address the issue (e.g., re-download, contact the vendor)? (3 marks)

2. What to Do if the Signature Check Fails

If the GPG signature check fails, it means the package has either been corrupted or tampered with, or the GPG key used to sign the package is not trusted. Here are the steps you can take to address the issue:

Step 1: Verify the GPG Key

If the signature fails, it could be due to the missing or untrusted GPG key used to sign the package. You can try importing the vendor's GPG key to your system:

sudo rpm --import <path_to_vendor_gpg_key>

If you don't have the key, you can often obtain it from the vendor's website or from a trusted source. After importing the key, you can try checking the signature again.

Step 2: Re-download the Package

If importing the GPG key does not resolve the issue, it is possible that the package was corrupted during the download process. In this case:

- **Re-download the RPM package** from the vendor's official website or repository to ensure that you have an untampered version.
- After re-downloading, repeat the signature check to verify if it is valid.

Step 3: Contact the Vendor

If the signature still fails after importing the correct GPG key and re-downloading the package, it's possible that the package itself is compromised or there is an issue with the vendor's signing process. In this

		case: • **Contact the vendor** to inform them of the issue. • Request a fresh or properly signed package. ### Step 4: Do Not Install the Package If you cannot resolve the signature issue or contact the vendor, **do not install the package**. Installing an unsigned or tampered package can pose a significant security risk to the system, including introducing malware or unauthorized modifications. --- ## Summary: . **Verify the GPG Signature:** Use the `rpm --checksig example-2.0-1.x86_64.rpm` command to verify the package's authenticity. . **If the Signature Check Fails:** • **Verify the GPG key** by importing it from the vendor. • **Re-download the package** from the vendor's official source. • **Contact the vendor** if the issue persists and request a properly signed package. . **Security Implications:** • If you cannot verify the signature, **do not install** the package as it could be compromised.		
55	5.1	You have a Linux server that requires certain tasks to be performed at specific intervals, such as backing up logs every morning 5AM, and sending a system status report every 6 hours. A. Explain how you would schedule the backup task to run at 5 AM every day using the Cron daemon. (3 marks) ## 1. Schedule the Backup Task to Run at 5 AM Every Day To schedule the backup task to run at 5 AM every day using the **Cron daemon**, follow these steps: **Open the crontab file** for editing: crontab -e 1. **Add the cron job** for the backup task. The syntax for a cron job is: * * * * * command_to_execute - - - - - | | | | | | | | | +---- Day of week (0-7) (Sunday=0 or 7) | | | +------ Month (1-12) | | +-------- Day of month (1-31) | +---------- Hour (0-23)	CO5,CO5	Medium - - A

+---------- Minute (0-59)
For **running the backup at 5 AM every day**, the cron expression will be:

0 5 * * * /path/to/backup_script.sh

2.
   - `0`: Minute (0th minute, exactly at the start of the hour).
   - `5`: Hour (5 AM).
   - `*`: Day of the month (any day).
   - `*`: Month (any month).
   - `*`: Day of the week (any day of the week).

3. **Explanation**: This cron job will execute the script `backup_script.sh` at 5:00 AM every day.

B. Schedule another Corn job as the System Status Report to Be Sent Every 6 Hours.Provide the specific Cron expression used to schedule this task. (3 marks)

2. Schedule the System Status Report to Be Sent Every 6 Hours

To schedule the system status report to be sent every 6 hours, you can use the following cron expression:

1. **Open the crontab file** for editing (if not already open):

   crontab -e

2. **Add the cron job** for sending the system status report. The cron expression for every 6 hours would be:

   0 */6 * * * /path/to/system_status_report.sh
   - `0`: Minute (0th minute).
   - `*/6`: Hour (every 6th hour, meaning at 0:00, 6:00, 12:00, and 18:00).
   - `*`: Day of the month (any day).
   - `*`: Month (any month).
   - `*`: Day of the week (any day of the week).

Explanation: This cron job will execute the script `system_status_report.sh` every 6 hours (at midnight, 6 AM, 12 PM, and 6 PM).

Summary of Cron Expressions:

1. **Backup Task at 5 AM Every Day**:

   0 5 * * * /path/to/backup_script.sh

		2. **System Status Report Every 6 Hours**: 0 */6 * * * /path/to/system_status_report.sh These cron expressions will ensure that the tasks are performed at the specified times without manual intervention.		
56	5.1	You need to schedule a one-time system update that should run exactly 30 minutes after a system restart. A. How would you schedule the system update using the Atd daemon? (3 marks) **1. Scheduling a One-Time System Update with Atd** The **Atd** daemon allows you to schedule one-time tasks that will run once at a specified time, which is useful when you want to schedule a task to occur **after a system restart**. To schedule a system update to run exactly **30 minutes after a system restart**, you can use the `at` command. **Steps:** **Ensure the Atd service is running**: If the Atd service is not already running, start it using: sudo systemctl start atd 1. **Schedule the task** using the `at` command: Use the following command to schedule the system update 30 minutes after the system restarts: sudo at now + 30 minutes -f /path/to/system_update_script.sh 2. ○ `now + 30 minutes`: This tells `at` to schedule the task for 30 minutes after the current time (i.e., after the system restart). ○ `-f /path/to/system_update_script.sh`: This option specifies the path to the script or command to run. **Verify the scheduled job**: To verify that the job is scheduled, you can use the `atq` command: atq 3. This will list all the scheduled tasks, including the one-time system update. B. Compare and contrast the functionality of Atd with Cron. When would Atd be more appropriate than Cron? (3 marks) **2. Comparing Atd with Cron** **Cron** and **Atd** are both used to schedule tasks, but they differ in functionality and use cases.	CO5,CO5	Easy - - A

Cron:

- **Functionality**: Cron is used to schedule recurring tasks at fixed times or intervals (e.g., daily, weekly, monthly, etc.).
- **Use Case**: Cron is best suited for tasks that need to run on a regular schedule. For example, database backups every day at midnight or sending system reports every Monday at 8 AM.
- **Example**: Schedule a job to run at 5 AM every day: `0 5 * * * /path/to/script.sh`

Atd:

- **Functionality**: Atd is used for one-time tasks that should run at a specified time in the future. It is not meant for recurring tasks.
- **Use Case**: Atd is best used for one-time tasks that need to be executed once, such as running a script 30 minutes after a reboot, or scheduling a task after a specific delay.
- **Example**: Schedule a job to run 30 minutes after system startup: `at now + 30 minutes -f /path/to/script.sh`

When to Use Atd Over Cron:

- **One-Time Tasks**: Use **Atd** for one-time tasks that should run once after a specific event, like a system reboot or a certain amount of time after a particular event.
- **Cron** would be more appropriate when you need a **repetitive, recurring task**, such as hourly, daily, or weekly scheduled jobs.

Summary:

. **Schedule a System Update Using Atd**: Use the command `sudo at now + 30 minutes -f /path/to/system_update_script.sh` to run the update 30 minutes after the system restarts.

. **Comparison Between Atd and Cron**:

- **Cron**: For **recurrent tasks** on a regular schedule.
- **Atd**: For **one-time tasks** at a specific future time or after a certain delay (such as after a system restart).

57	5.1	You are a system administrator managing a Linux server that hosts several critical network services such as FTP, SSH, and Telnet. The server needs to handle incoming requests securely and efficiently while ensuring that each service has appropriate resource management and logging. You are deciding between using Xinetd and Inetd to manage these services. Given the scenario, explain why you would choose Xinetd over Inetd to manage these services, considering security, performance, and logging requirements. When managing critical network services such as FTP, SSH, and Telnet, it's essential to ensure security, efficient resource management, and detailed logging. Between **Xinetd** and **Inetd**, **Xinetd** is often the better choice due to several important advantages: ## 1. Security Features: ● **Xinetd** provides **better security** controls compared to **Inetd**. It includes advanced access controls, such as: ○ **Access Control Lists (ACLs)**: You can define who can access specific services based on IP addresses, subnet ranges, and other criteria. ○ **Limiting Access**: Xinetd can limit the number of simultaneous connections to a service, preventing denial of service (DoS) attacks. ○ **Service Timeouts**: Xinetd allows you to configure timeouts for idle connections, reducing the risk of open connections being hijacked or abused. ○ **Rate Limiting**: You can configure limits on how many times a service can be accessed in a short period, which helps to mitigate brute-force attacks. **Inetd**, on the other hand, lacks these robust security features. While it can manage services, its security mechanisms are far less granular and less configurable than those of **Xinetd**. ## 2. Performance and Resource Management: ● **Xinetd** is designed to be more efficient in managing services. It supports features like **TCP wrappers**, which can be used to restrict service access more efficiently than Inetd. ● **Dynamic Service Management**: Xinetd is able to dynamically manage services based on conditions, like the number of active connections or system load. It can start and stop services based on system resources, ensuring optimal resource usage. ● **Connection Handling**: Xinetd is more robust in handling incoming connections. For example, it can spawn a new process only when a request is received, rather than keeping services running continuously, as **Inetd** might. **Inetd** is an older tool and is not as efficient in handling modern requirements for high-performance networking. It spawns a process for each connection, which can be less efficient when managing a large number of simultaneous requests. ## 3. Logging Capabilities: ● **Xinetd** provides **better logging** capabilities. It supports detailed logging for each service, such as: ○ **Per-service logging**: Logs can be customized per	CO5	Easy - Analysing - T

service, which helps to troubleshoot issues and track suspicious activity.

- **Syslog Integration**: Xinetd can log to the system's logging service (like **syslog**), which allows for centralized log management.
- **Log Rotation**: It offers more flexible log rotation options, helping to manage large log files efficiently.

Inetd offers much less detailed logging. While it can log information, its capabilities are basic and less flexible. For security-critical services, this limitation is a drawback because detailed and customizable logs are crucial for monitoring suspicious activities.

4. Service Control and Management:

- **Xinetd** provides more **granular control over services**. You can configure options like:
  - **Wait and Nowait settings**: Control whether the service should wait for an incoming connection before starting or if it should run multiple instances of the service simultaneously.
  - **Service Configuration**: Xinetd allows you to fine-tune configuration parameters, including timeouts, retries, and connection handling, ensuring each service behaves optimally according to its security and performance needs.

Inetd offers basic service management but lacks the advanced configuration options and flexibility that **Xinetd** provides, making it harder to fine-tune service behavior for optimal performance and security.

5. Modern Support and Development:

- **Xinetd** is actively maintained and developed, which means it receives regular updates and security patches.
- **Inetd** is a much older tool, and while still functional, it is no longer actively developed, making it less suitable for modern security and performance requirements.

Conclusion:

Given the critical nature of the network services (FTP, SSH, Telnet) on the Linux server, **Xinetd** is the better choice for managing these services. It provides enhanced **security features**, such as access control, rate limiting, and connection timeout handling. It also offers **better performance**, resource management, and detailed **logging capabilities** that help in monitoring and troubleshooting. Additionally, Xinetd's flexibility and modern support make it a more robust solution than **Inetd**, which is outdated and lacks the necessary features for managing modern, high-security, high-performance network services.

58	5.1	You are a system administrator responsible for managing a file-sharing server that provides shared access to directories across a corporate network. The server uses the NFS daemon (nfsd) to allow clients on the network to mount and access specific directories remotely. However, the organization has strict security policies, and you need to ensure that the shared directories are accessible only to authorized users and specific departments. What security measures should be taken into account when configuring nfsd to prevent unauthorized access to shared directories? When configuring **nfsd** (NFS daemon) for secure file sharing, especially in a corporate environment where strict security policies must be followed, it's essential to implement various security measures to restrict unauthorized access and ensure that only authorized users and departments can access shared directories. Below are the key security measures to take into account: ## 1. Access Control with `/etc/exports`: **Define Client Access Control**: The **/etc/exports** file controls which directories are shared and who has access to them. It is crucial to specify **trusted IP addresses** or **hostnames** of the clients that are allowed to mount the shared directories. For example: /shared_directory client1.example.com(rw,sync,no_root_squash) /shared_directory 192.168.1.0/24(ro,sync) ○ `rw`: Read-write access for the client. ○ `ro`: Read-only access for the client. ○ `sync`: Ensure that changes are written to disk before responding. ○ `no_root_squash`: Disables the mapping of the root user on the client to a non-privileged user on the server. By specifying only trusted IPs or network ranges, you can prevent unauthorized machines from accessing the shared directories. ## 2. NFS Version and Encryption: ● **Use NFSv4**: Ensure that the system uses **NFSv4** instead of NFSv3 or older versions, as NFSv4 includes significant improvements in security, including better **authentication** and **authorization** mechanisms (e.g., Kerberos). ● **Encrypt NFS Traffic**: NFS traffic is typically transmitted in clear text, which may expose sensitive data. Consider using **Kerberos authentication** and **NFS over SSH** or implement an **IPsec tunnel** to encrypt the traffic between clients and the NFS server. ## 3. Use Kerberos Authentication: ● **Enable Kerberos Authentication**: Set up **Kerberos** for NFS, which will help to ensure that only users with valid Kerberos tickets can access the NFS shares. This adds an extra layer of security by providing strong authentication. To configure Kerberos authentication for NFS, modify the `/etc/exports` file to include the Kerberos option: /shared_directory client1.example.com(rw,sync,sec=krb5) ○ This ensures that the client must authenticate with	CO5	Easy - Analysing - T

Kerberos before accessing the directory.

4. Use Firewall and Network Restrictions:

- **Restrict NFS Ports**: NFS uses several ports for communication (e.g., port 2049 for NFS, port 111 for RPC). Use a **firewall** (e.g., **iptables**, **firewalld**) to only allow traffic from authorized IP addresses on these ports.

Example of blocking all except certain IP addresses:

```
sudo iptables -A INPUT -p tcp -s 192.168.1.0/24 --dport 2049 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 2049 -j DROP
```

5. Restrict NFS Root Access:

- **Use `root_squash`**: For security purposes, ensure that root users from client machines are not granted superuser access on the NFS server. Use the **`root_squash`** option in the `/etc/exports` file to map the root user on clients to a non-privileged user on the NFS server.

Example:

```
/shared_directory 192.168.1.0/24(rw,sync,root_squash)
```

6. Set Appropriate Permissions:

- **File Permissions**: Make sure the shared directories have the correct file permissions. Set **read**, **write**, and **execute** permissions based on the needs of the authorized users and groups. For example, use the `chmod` command to set appropriate permissions.

Example:

```
chmod 750 /shared_directory
chown root:staff /shared_directory
```

- This limits access to only users in the **staff** group, with only root having full control.

7. Limit Access to Specific Users and Groups:

- **Use `map_static` for user mapping**: When sharing directories, map the NFS users to specific users and groups on the server side. This ensures that only users in specific groups or with certain UID/GID can access the directory. You can configure this through `idmapd.conf` (for NFSv4).

- **Exporting Directories for Specific Groups**: You can ensure that only specific groups or users have access to certain directories by using **`/etc/exports`** options.

Example:

```
/shared_directory
192.168.1.0/24(rw,sync,secure,all_squash,anonuid=1001,anongid=1001)
```

		○ This configuration ensures that even if the client tries to access the NFS share as root, the access will be mapped to the specified anonymous UID/GID. ## 8. Audit NFS Access: ● **Log NFS Access**: Consider enabling **logging** for NFS activity. This can be achieved by enabling auditing services such as **auditd** to monitor all NFS-related activities and ensure that no unauthorized access occurs. ● You can also use **syslog** to capture logs related to NFS access: Example: echo "nfsd" >> /etc/rsyslog.d/50-default.conf ## 9. Disable Unused NFS Services: ● **Disable Unnecessary NFS Services**: Disable NFS services that aren't in use (e.g., NFSv2 or NFSv3) to reduce the potential attack surface. You can do this by disabling older versions of NFS in the `/etc/nfs.conf` file or directly in the `nfsd` configuration file. ## 10. Use SELinux for Additional Security: ● **Configure SELinux**: Ensure **SELinux** is enabled and configured correctly for NFS. SELinux provides an additional layer of security by restricting the actions that can be performed by services running on the system. ○ Ensure that the NFS server runs with the correct SELinux context and access control rules for NFS file shares. Example: semanage fcontext -a -t nfs_t "/shared_directory(/.*)?" restorecon -R /shared_directory ## Conclusion: By taking these security measures into account when configuring **nfsd**, you can ensure that the shared directories are only accessible by authorized users, specific departments, and trusted machines. Key aspects include restricting client access, using Kerberos for authentication, enabling firewalls, controlling root access, setting appropriate file permissions, and utilizing auditing for monitoring. These practices will help protect your NFS server from unauthorized access and ensure compliance with your organization's security policies.		
59	5.1	Your organization requires a file server that can serve files across different departments using NFS for Unix/Linux systems and Samba for Windows-based clients. You are tasked with configuring and securing both file service daemons to ensure reliable file access while enforcing strict access control for different departments. Explain the role of file service daemons like nfsd (NFS daemon) and smbd (Samba daemon) in enabling file sharing across different operating systems, and how each service interacts with its clients.	CO5	Medium - Analysing - T

In your organization, the need to serve files across both Unix/Linux and Windows-based systems requires configuring and securing both **NFS** (Network File System) and **Samba** to ensure reliable file access and strict access control. These file service daemons enable file sharing between different operating systems, with each having its own methods of sharing, authentication, and access control. Here's an explanation of the role of **nfsd** and **smbd** and how they interact with their clients:

NFS (Network File System) - nfsd Daemon:

1. **Role of nfsd:**
    - **nfsd** (NFS daemon) is the server-side process responsible for managing file access over the **NFS protocol**. It allows Unix/Linux systems to share directories and files over a network. NFS operates primarily on Unix-like systems, allowing seamless file sharing between systems running similar operating systems, such as Linux, AIX, Solaris, and macOS.
    - NFS allows clients to mount directories from the server as if they are local file systems, making the remote files accessible to the clients.
2. **How nfsd Interacts with Clients:**
    - Clients initiate connections to the NFS server, requesting access to directories and files shared via the NFS protocol. The server responds by either allowing or denying access based on the client's IP address, authentication mechanisms (like **Kerberos**), and the configuration in the **/etc/exports** file.
    - When a client requests access to a file on the server, **nfsd** handles the request, ensuring that the correct file permissions are applied, and data is transmitted over the network.
    - **Exporting Directories:** Directories are shared on the NFS server by specifying them in the **/etc/exports** file, where access permissions and client specifications (IP, hostname, etc.) are defined.
3. **Security Considerations with nfsd:**
    - **Root Squashing**: By default, root users on clients are mapped to a non-privileged user on the server to avoid potential security risks.
    - **Kerberos Authentication**: To enhance security, NFS can be configured to use **Kerberos** for strong authentication, ensuring that only authorized users can access shared directories.
    - **Firewall Restrictions**: The server's firewall can restrict access to NFS to specific IP ranges or networks, ensuring that only trusted clients can connect.

Samba (SMB/CIFS) - smbd Daemon:

1. **Role of smbd:**
    - **smbd** (Samba daemon) is the server-side process responsible for providing file and print services to **Windows-based clients** using the **SMB/CIFS protocol**. This protocol is native to Windows operating systems and is widely used for file sharing and printer sharing in

Windows networks. Samba allows Unix/Linux systems to interact with Windows clients and vice versa, enabling **cross-platform file sharing**.

- Samba can also act as a **domain controller** or integrate with an existing Windows Active Directory environment, enabling centralized authentication and user management.

2. **How smbd Interacts with Clients:**

   - **Windows Clients**: Windows machines access Samba shares as if they were part of the native Windows network. They can browse the shared directories, read and write files, and perform file operations such as copying and deleting, all with their native Windows file explorer or applications.

   - **Linux Clients**: Linux systems can also access Samba shares using **mount.cifs** or SMB clients like **smbclient**, making it possible to mount Windows shares on Linux systems for seamless access.

   - **Share Configuration**: Shared directories and printers are configured in the **/etc/samba/smb.conf** file. Within this file, administrators specify share names, access permissions, authentication settings, and various other options such as whether to allow guest access or require passwords for clients.

3. **Security Considerations with smbd:**

   - **User Authentication**: Samba provides several modes of authentication, including using **UNIX user credentials**, **Windows NTLM** authentication, or integration with **Active Directory** for centralized authentication. Samba can restrict access based on users, groups, or even specific network IP ranges.

   - **Access Control**: Through the **smb.conf** configuration file, access to Samba shares can be controlled with granular permissions (read-only, read-write) and restrictions (IP addresses, user groups). Samba also supports **file ACLs** for more advanced permission management.

   - **Encryption**: Samba can encrypt traffic between the server and client using **SMB3** for secure communication, protecting sensitive data from eavesdropping over the network.

Key Differences between nfsd and smbd:

1. **Protocol:**

   - **nfsd** uses the **NFS protocol** (mainly for UNIX-like systems).

   - **smbd** uses the **SMB/CIFS protocol** (mainly for Windows-based systems).

2. **Client Compatibility:**

   - **nfsd** is designed for **Linux/Unix clients**.

   - **smbd** is designed to provide file access to **Windows clients**, but also supports **Linux/Unix clients** through **mount.cifs** or **smbclient**.

3. **Authentication:**
    - **nfsd** can use **UNIX-based authentication** or **Kerberos**.
    - **smbd** uses **Windows-style authentication** (either local Samba users or integration with Active Directory).
4. **File Permissions:**
    - **nfsd** follows **UNIX file permissions**, which are based on **UID** and **GID**.
    - **smbd** uses **Windows ACLs** for finer-grained permissions or can use **UNIX-style permissions** when interacting with Linux/Unix clients.
5. **Cross-Platform Support:**
    - **nfsd** works best for sharing files between **Unix/Linux systems**.
    - **smbd** is designed for **Windows** and also supports **Linux/Unix clients** using Samba shares.

Best Practices for Securing Both Services:

- **NFS**:
    - Use **Kerberos** for secure authentication.
    - Restrict access to specific IP addresses via **firewall rules**.
    - Use **root squashing** to prevent root access from client systems.
    - Limit shares and access using **/etc/exports**.
- **Samba**:
    - Restrict access to Samba shares based on user/group permissions.
    - Integrate Samba with **Active Directory** or use **UNIX authentication** for more control.
    - Enable **SMB encryption** for secure data transfer.
    - Set **strong passwords** and use **access control lists** (ACLs) for detailed access management.

Conclusion:

In summary, **nfsd** and **smbd** both serve the essential function of enabling file sharing across different systems. **nfsd** is designed for sharing files among Unix/Linux systems, while **smbd** enables Windows systems to access files on Unix/Linux servers. By configuring and securing both daemons, you can ensure that your file server supports both Windows and Unix/Linux clients, while maintaining strict access control to meet your organization's security policies.

60	5.1	As part of a security audit, you are tasked with enhancing the security of SSH to prevent unauthorized access and ensure that remote logins are controlled and monitored. Explain how the sshd daemon facilitates remote login and command execution, and describe the key components involved in its configuration (e.g., configuration file, authentication methods). The **sshd** (SSH daemon) is a critical component of the SSH (Secure Shell) service in Linux/Unix systems. It facilitates secure remote login and command execution over an encrypted connection, providing a robust mechanism for administrators and users to interact with the system remotely. Below is an explanation of how **sshd** works and the key components involved in its configuration: ## How sshd Facilitates Remote Login and Command Execution: 1. **Remote Login:** ○ **sshd** listens on a specified port (by default, port 22) for incoming SSH connection requests from remote clients. When a user initiates an SSH connection, **sshd** authenticates the user based on the configured authentication methods and grants or denies access accordingly. ○ Once authenticated, **sshd** opens an encrypted channel for the user's session, ensuring that all communication between the client and the server is securely encrypted, preventing eavesdropping and man-in-the-middle attacks. 2. **Command Execution:** ○ After a successful login, the user can execute commands remotely through the SSH session. The **sshd** daemon provides the environment for command execution, allowing the user to interact with the system as if they were directly logged in. ○ The system's shell (e.g., **bash**) is invoked, and the user can execute commands, interact with files, and perform administrative tasks, depending on the permissions granted. ## Key Components of the sshd Configuration: The configuration of the **sshd** daemon is controlled primarily through the **/etc/ssh/sshd_config** file. This file contains various settings that determine how the SSH service behaves, including security features, authentication methods, and access control. Below are the key components: 1. **Port and Address Binding:** ○ **Port**: Specifies the port on which the **sshd** service listens for incoming connections. By default, this is port 22. ■ Example: `Port 22` ○ **ListenAddress**: Specifies which IP address(es) **sshd** should listen on. It is often used for limiting access to specific network interfaces or IP ranges. ■ Example: `ListenAddress 0.0.0.0` (Listen on all network interfaces)	CO5	Medium - Evaluating - T

2. **Authentication Methods:**

    - **Password Authentication**: This allows users to authenticate using a password. By default, this is enabled, but for stronger security, it is recommended to disable it in favor of key-based authentication.

        - Example: `PasswordAuthentication no`

    - **Public Key Authentication**: This is the preferred authentication method for SSH, where the user's public key is stored in the **~/.ssh/authorized_keys** file, and the private key is kept on the client system. This is more secure than password authentication, as it is resistant to brute force attacks.

        - Example: `PubkeyAuthentication yes`

    - **ChallengeResponseAuthentication**: This enables or disables challenge-response authentication methods (e.g., one-time passwords, two-factor authentication).

        - Example: `ChallengeResponseAuthentication no`

    - **PermitEmptyPasswords**: This specifies whether login is allowed for accounts with empty passwords. It is recommended to set this to **no** for increased security.

        - Example: `PermitEmptyPasswords no`

3. **Access Control:**

    - **AllowUsers** and **DenyUsers**: These directives allow or deny access to specific users or groups. This can be used to restrict who can log in via SSH.

        - Example: `AllowUsers user1 user2`

    - **AllowGroups** and **DenyGroups**: Similar to **AllowUsers** and **DenyUsers**, these directives control access based on group membership.

        - Example: `AllowGroups sshusers`

    - **PermitRootLogin**: This setting controls whether the **root** user can log in directly via SSH. For security reasons, it is recommended to disable root login and use **sudo** or **su** for administrative tasks.

        - Example: `PermitRootLogin no`

4. **Timeout and Session Settings:**

    - **ClientAliveInterval**: This setting determines how often the server will send a message to the client to check if the connection is still alive. If the client does not respond, the connection is closed.

        - Example: `ClientAliveInterval 300` (Send alive message every 5 minutes)

    - **ClientAliveCountMax**: Specifies the number of client alive messages that can be sent without receiving a response before the connection is terminated.

        - Example: `ClientAliveCountMax 3` (Allow 3 missed responses)

- **LoginGraceTime**: Specifies the amount of time the server will wait for a successful login before disconnecting the client.
    - Example: `LoginGraceTime 60` (1 minute grace period)

5. **Logging:**
    - **LogLevel**: This setting controls the verbosity of logging for **sshd**. It can be set to levels such as **INFO**, **VERBOSE**, or **DEBUG**. For security audits, **VERBOSE** or **DEBUG** levels can provide detailed logs of SSH activity.
        - Example: `LogLevel VERBOSE`
    - Logs are stored in **/var/log/auth.log** (depending on the distribution), providing valuable information on login attempts, successful logins, and errors.
6. **Key Management:**
    - **AuthorizedKeysFile**: Specifies the location of the file containing the public keys for key-based authentication. The default is **~/.ssh/authorized_keys**.
        - Example: `AuthorizedKeysFile .ssh/authorized_keys`
    - **HostKey**: Specifies the location of the server's private keys used for SSH. These keys are used to establish a secure connection with the client.
        - Example: `HostKey /etc/ssh/ssh_host_rsa_key`

Enhancing Security with sshd:

To enhance the security of SSH access, consider implementing the following:

1. **Disable Password Authentication**: Prefer **public key** authentication, and set **PasswordAuthentication** to **no** to prevent brute-force password attacks.
2. **Disable Root Login**: Set **PermitRootLogin** to **no** to prevent direct root access over SSH. Users can use **sudo** for administrative tasks instead.
3. **Use Strong Encryption Algorithms**: Set **Ciphers** and **MACs** in the **sshd_config** to use strong algorithms.
4. **Limit Access by IP Address**: Use **AllowUsers** or **AllowGroups** to restrict SSH access to specific users or groups.
5. **Two-Factor Authentication**: Configure **ChallengeResponseAuthentication** and use **Google Authenticator** or similar methods for additional authentication layers.
6. **SSH Banner**: Display a legal notice or banner on login attempts using the **Banner** option in **sshd_config**.

Conclusion:

		The **sshd** daemon plays a crucial role in securing remote login and command execution on a Linux/Unix system. By configuring the **sshd_config** file correctly and implementing best practices like disabling root login, using key-based authentication, and configuring appropriate logging and access controls, you can significantly enhance the security of SSH on your system and minimize the risk of unauthorized access.		