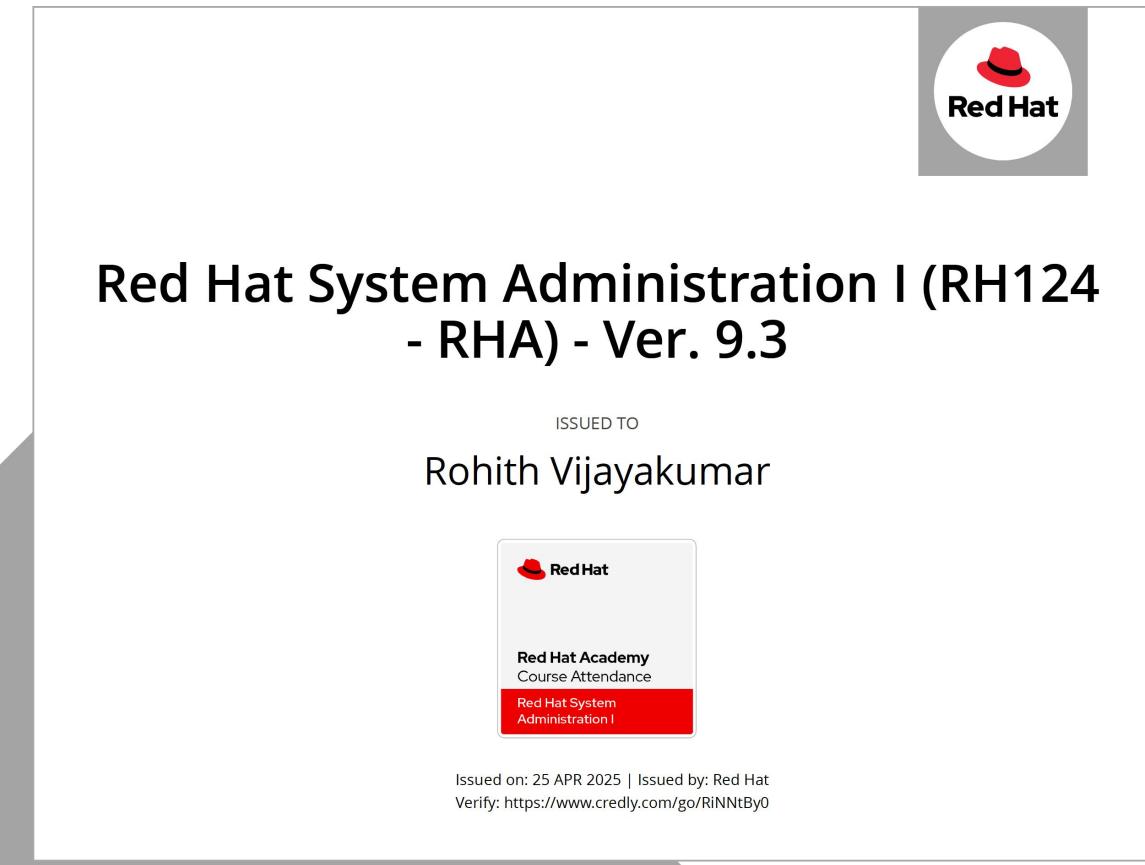


Red Hat System Administration I 9.3(RH-124)**Certificate of Completion***Fig 0.1: Certificate of completion*

Note: Certificate of completion received post completion of course.

Chapter 1: Get Started with Red Hat Enterprise Linux**Summary:**

- Fedora: Community-driven, cutting-edge updates every 6 months, suitable for developers but not for production use.
- CentOS Stream: A continuous delivery platform upstream of RHEL, great for testing and contributing but not ideal for production.
- RHEL: Stable, commercially supported, used by enterprises with long-term support, requires a subscription (free Developer version available).
- The command line allows user input via a shell, where commands consist of the program, options (modifying behavior), and arguments (specifying input).
- SSH (Secure Shell): Secure login protocol using public key authentication to encrypt connections, replacing password-based login.
- Workspaces: Virtual desktops that allow separate open applications/windows for better multitasking and organization.

Chapter 2: Access the Command Line

Lab: Access the Command Line

Fig 2.1: Lab Initialization and Command-Line Output Execution

```

student@workstation:~$ ssh workstation; exit
Activate this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Thu Apr 17 01:37:21 2025 from 10.88.0.2
student@workstation:~$ PS1="Rohith Yellambalase Vijayakumar|${D%Y-%m-%d %H:%M}:\$ "
Rohith Yellambalase Vijayakumar|2025-04-17 01:54:51 ~$lab start cli-review
Starting lab.
  • Checking lab systems ..... SUCCESS
  • Ensuring that the student user password is set ..... SUCCESS
  • Copying files required for lab tasks ..... SUCCESS
Rohith Yellambalase Vijayakumar|2025-04-17 01:51:08 ~date
Thu Apr 17 01:51:08 UTC 2025
Rohith Yellambalase Vijayakumar|2025-04-17 01:51:31 ~date +R
Rohith Yellambalase Vijayakumar|2025-04-17 01:51:42 >file zcat
zcat: a /usr/bin/sh script, ASCII text executable
Rohith Yellambalase Vijayakumar|2025-04-17 01:51:51 >wc zcat
32
Rohith Yellambalase Vijayakumar|2025-04-17 01:52:01 >head zcat
#!/usr/bin/sh
# Uncompress files to standard output.

# Copyright (C) 2007, 2010-2022 Free Software Foundation, Inc.
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 3 of the License, or
# (at your option) any later version.

Rohith Yellambalase Vijayakumar|2025-04-17 01:52:10 >tail zcat
With no FILE, or when FILE is -, read standard input.
Report bugs to <bug-gzip.gnu.org>.

case $1 in
--help) printf "%s\n" "Usage"
        printf "%s\n" "version" || exit 1; exit;
esac

exec gzip -cd "$0"
Rohith Yellambalase Vijayakumar|2025-04-17 01:52:20 >!!
tail zcat
With no FILE, or when FILE is -, read standard input.

Report bugs to <bug-gzip.gnu.org>.

student@workstation:~$ 

```

Note: This figure highlights the successful execution of simple programs from the Bash shell command line. It also emphasises the ability to execute commands to identify file types and display parts of text files. Additionally, it showcases the use of Bash command history shortcuts for efficiently repeating commands or parts of commands.

Fig 2.2: CLI Review Lab – Zcat File Analysis and Output Commands

```

student@workstation:~$ ssh workstation; exit
31 ssh workstation; exit
32 ssh workstation; exit
33 lab start cli-review
34
35 date +R
36 file zcat
37 wc Esc-
38 zcat
39 head zcat
40 tail zcat
41 tail -n 20
42 lab start cli-review
43 date
44 wc zcat
45 file zcat
46 wc zcat
47 head zcat
48 tail zcat
49 tail -n 20 zcat
50 date
51 date +R
52 zcat
53 head zcat
54 tail zcat
55 tail -n 20 zcat
56 tail -n 20 zcat
57 Rohith Yellambalase Vijayakumar|${D%Y-%m-%d %H:%M}:\$ >
58 lab start cli-review
59 date
60 file zcat
61 wc zcat
62 head zcat
63 tail zcat
64 tail -n 20 zcat
65 history
Rohith Yellambalase Vijayakumar|2025-04-17 01:53:31 >lab grade cli-review

Grading lab.
  • Checking lab systems ..... SUCCESS
Overall lab grade: PASS
Rohith Yellambalase Vijayakumar|2025-04-17 01:53:53 >lab finish cli-review

Finishing lab.
  • Checking lab systems ..... SUCCESS
  • Removing temporary files ..... SUCCESS
Rohith Yellambalase Vijayakumar|2025-04-17 01:54:04 >

student@workstation:~$ 

```

Note: This demonstrates the execution of CLI commands including file, wc, head, and tail on the zcat utility, along with environment setup and timestamp logging for the lab session passed successfully.

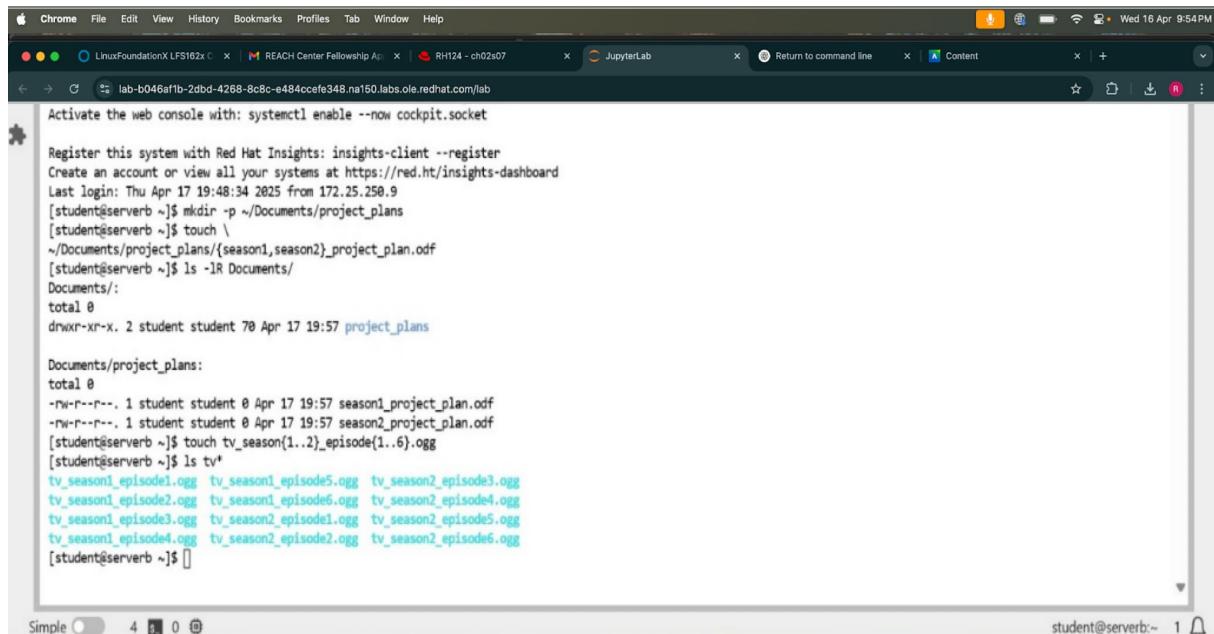
Summary:

- **Command Basics and Multiline Commands:** You can execute multiple commands on a single line using command1 ; command2, which allows for efficient task execution. Additionally, long commands can be split into multiple lines using the backslash \, helping keep your terminal commands clean and readable.
- **File Type and Content Commands:** The file command is used to determine the type of a file, such as whether it's text or binary. To view or concatenate file contents, the cat command is used, which prints the contents of one or more files in order. The head and tail commands allow you to view specific parts of a file, with head showing the beginning and tail displaying the end, providing flexibility when working with large files.
- **Word Count Tool (wc):** The wc command counts the number of lines, words, and characters in a file. It offers options such as -l to count lines, -w to count words, and -c for characters, helping users quickly gather statistics on text files.
- **Command History and Navigation:** The history command displays previously executed commands, making it easier to review and repeat past actions. You can use !number to run a specific command from the history list, or !string to repeat the last command that started with a certain string. Navigation through the history can also be done using the arrow keys (UpArrow, DownArrow, and Left/Right Arrows), allowing for fast corrections and re-execution of commands.
- **Tab Completion:** Tab completion is a helpful feature that auto-completes file paths, command names, and options. Pressing the Tab key once automatically completes the command or filename if it's unique, and pressing it twice shows all possible matches, improving productivity.
- **Password Management:** You can change your system password using the passwd command, a crucial tool for maintaining secure access to your system. This command helps ensure that users can easily update their login credentials.

Chapter 3: Manage Files from the Command Line

Lab: Manage Files from the Command Line

Fig. 3.1 File command to find file format



```

Activate the web console with: systemctl enable --now cockpit.socket

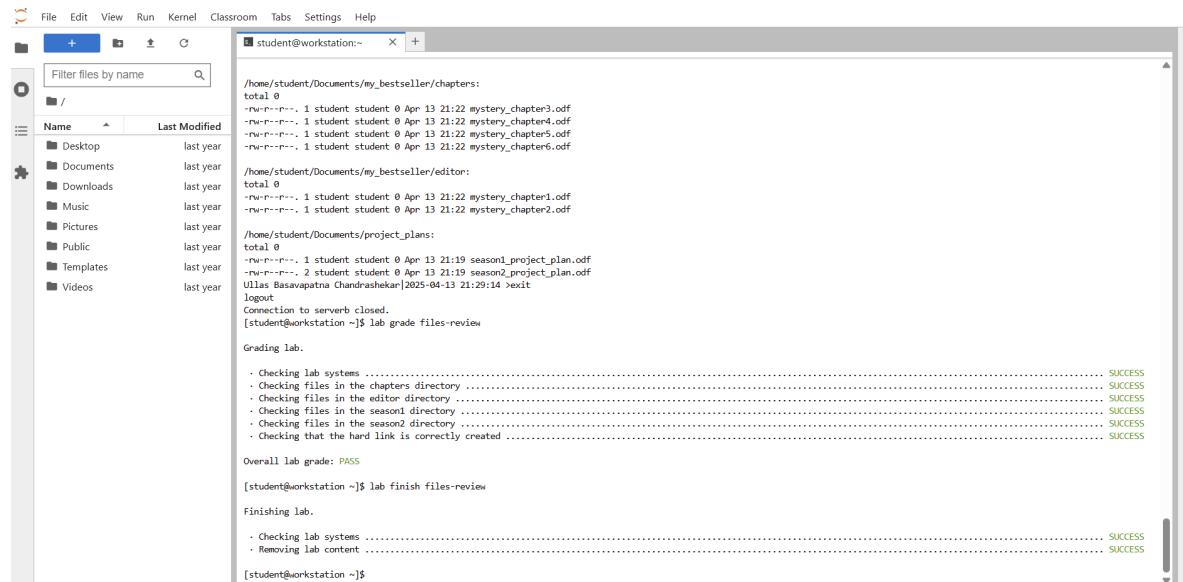
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Thu Apr 17 19:48:34 2025 from 172.25.250.9
[student@serverb ~]$ mkdir -p ~/Documents/project_plans
[student@serverb ~]$ touch \
~/Documents/project_plans/season1,season2}_project_plan.odf
[student@serverb ~]$ ls -lR Documents/
Documents/:
total 0
drwxr-xr-x. 2 student student 70 Apr 17 19:57 project_plans

Documents/project_plans:
total 0
-rw-r--r--. 1 student student 0 Apr 17 19:57 season1_project_plan.odf
-rw-r--r--. 1 student student 0 Apr 17 19:57 season2_project_plan.odf
[student@serverb ~]$ touch tv_season{1..2}_episode{1..6}.ogg
[student@serverb ~]$ ls tv*
tv_season1_episode1.ogg tv_season1_episode5.ogg tv_season2_episode3.ogg
tv_season1_episode2.ogg tv_season1_episode6.ogg tv_season2_episode4.ogg
tv_season1_episode3.ogg tv_season2_episode1.ogg tv_season2_episode5.ogg
tv_season1_episode4.ogg tv_season2_episode2.ogg tv_season2_episode6.ogg
[student@serverb ~]$ 

```

Note : Usage of the file command to determine the file format

Fig 3.2: Screenshot showing the successful execution of file creation, movement, and grading for the lab task.



```

student@workstation:~$ ls
Filter files by name
student@workstation:~$ cd /home/student/Documents/my_bestseller/chapters
student@workstation:~/Documents/my_bestseller/chapters$ ls
total 0
-rw-r--r--. 1 student student 0 Apr 13 21:22 mystery_chapter3.odf
-rw-r--r--. 1 student student 0 Apr 13 21:22 mystery_chapter4.odf
-rw-r--r--. 1 student student 0 Apr 13 21:22 mystery_chapter5.odf
-rw-r--r--. 1 student student 0 Apr 13 21:22 mystery_chapter6.odf
student@workstation:~/Documents/my_bestseller/editor$ ls
total 0
-rw-r--r--. 1 student student 0 Apr 13 21:22 mystery_chapter1.odf
-rw-r--r--. 1 student student 0 Apr 13 21:22 mystery_chapter2.odf
student@workstation:~/Documents/project_plans$ ls
total 0
-rw-r--r--. 1 student student 0 Apr 13 21:19 season1_project_plan.odf
-rw-r--r--. 2 student student 0 Apr 13 21:19 season2_project_plan.odf
Ullas Basavapatna Chandrashekhar[2025-04-13 21:29:14] >exit
Connection to serverb closed.
[student@workstation ~]$ lab grade files-review

Grading lab.
  • Checking lab systems ..... SUCCESS
  • Checking files in the chapters directory ..... SUCCESS
  • Checking files in the editor directory ..... SUCCESS
  • Checking files in the season1 directory ..... SUCCESS
  • Checking files in the season2 directory ..... SUCCESS
  • Checking that the hard link is correctly created ..... SUCCESS

Overall lab grade: PASS
[student@workstation ~]$ lab finish files-review

Finishing lab.
  • Checking lab systems ..... SUCCESS
  • Removing lab content ..... SUCCESS

[student@workstation ~]$ 

```

Note: This image illustrates the terminal output for tasks performed during the "Manage Files from the Command Line" lab, where directories and files were created and organized using various Bash commands. The task also involved moving files to specific directories and using symbolic and hard links, with the final step showing the grading results.

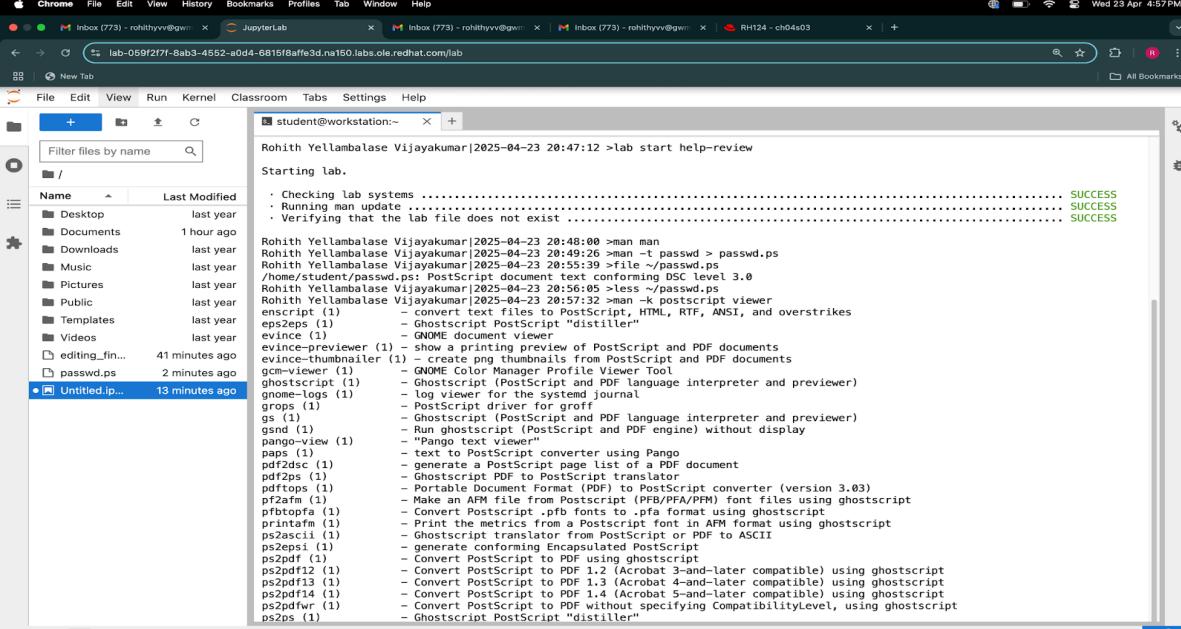
Summary:

- Linux directories are structured hierarchically with directories like /bin, /usr, /home, and /var, each serving specific system and user-related purposes.
- The pwd command shows the current directory, while cd allows navigation between directories using absolute or relative paths.
- File operations include commands like cp (copy), mv (move), and rm (remove), which can be used with various options for more specific tasks like recursive actions or forced deletion.
- Links in Linux, both hard and symbolic (soft), allow multiple names or shortcuts for the same file. Hard links directly point to the same data, while symbolic links are shortcuts that may break if the original file is removed.
- Shell features like pathname expansion (wildcards), brace expansion, tilde expansion, and variable expansion provide flexibility and efficiency in working with file paths and variables in Linux commands.

Chapter 4: Get Help in Red Hat Enterprise Linux

Lab: Get Help in Red Hat Enterprise Linux

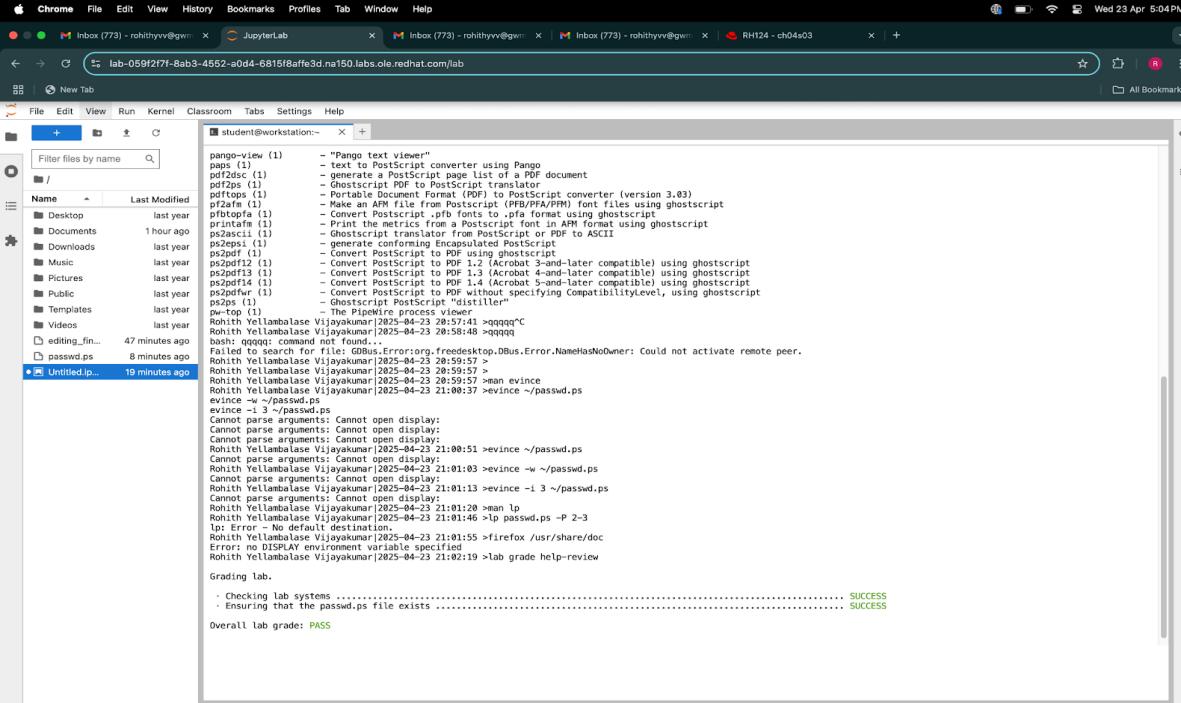
Fig. 4.1 PostScript Tools and Man Page Formatting in Linux



```
Rohith Yellambalase Vijayakumar|2025-04-23 20:47:12 >lab start help-review
Starting lab.
.
.
.
Rohith Yellambalase Vijayakumar|2025-04-23 20:49:26 >man -t passwd >passwd.ps
Rohith Yellambalase Vijayakumar|2025-04-23 20:49:26 >man -t passwd >passwd.ps
Rohith Yellambalase Vijayakumar|2025-04-23 20:49:26 >man -t passwd >passwd.ps
Rohith Yellambalase Vijayakumar|2025-04-23 20:56:05 >less ~/passwd.ps
Rohith Yellambalase Vijayakumar|2025-04-23 20:57:32 >man -k postscript viewer
enscript (1)           - convert text files to PostScript, HTML, RTF, ANSI, and overstrikes
eps2ps (1)             - Ghostscript PostScript "distiller"
evince (1)             - GNOME document viewer
evince-viewer (1)      - show a running preview of PostScript and PDF documents
evince-thumbnails (1)  - create png thumbnails from PostScript and PDF documents
gcm-viewer (1)          - GNOME Color Manager Profile Viewer Tool
ghostscript (1)         - Ghostscript (PostScript and PDF language interpreter and previewer)
gnome-log (1)          - log viewer for the systemd journal
grops (1)              - PostScript driver for groff
gsnd (1)               - Run ghostscript (PostScript and PDF engine) without display
pango-view (1)          - "Pango text viewer"
paps (1)               - text to PostScript converter using Pango
pdf2dsc (1)            - generate a PostScript page list of a PDF document
pdf2ps (1)              - Ghostscript PDF to PostScript translator
pdf2ps (1)              - Portable Document Format (PDF) to PostScript converter (version 3.03)
pdf2afm (1)             - Make an AFM file from Postscript (.PFB/.PFA/.PFM) font files using ghostscript
pf2afm (1)              - Convert Postscript .pfb fonts to .pfa format using ghostscript
pf2topfa (1)            - Convert Postscript .pfb fonts to .pfa format using ghostscript
printafm (1)            - Print the metrics from a Postscript font in AFM format using ghostscript
ps2ascii (1)            - Ghostscript translator from PostScript or PDF to ASCII
ps2eps (1)              - generate conforming Encapsulated PostScript
ps2eps (1)              - Convert PostScript to PDF
ps2pdf12 (1)            - Convert PostScript to PDF 1.2 (Acrobat 3-and-later compatible) using ghostscript
ps2pdf13 (1)            - Convert PostScript to PDF 1.3 (Acrobat 4-and-later compatible) using ghostscript
ps2pdf14 (1)            - Convert PostScript to PDF 1.4 (Acrobat 5-and-later compatible) using ghostscript
ps2pdfw (1)              - Convert PostScript to PDF without specifying CompatibilityLevel, using ghostscript
ps2ps (1)               - Ghostscript PostScript "distiller"
```

Note : This Linux terminal session demonstrating how to convert man pages into PostScript format using the man -t command and how to explore PostScript-related utilities like ghostscript, enscript, and document viewers.

Fig. 4.2: Verifying PostScript File Output in Linux Terminal



```
pango-view (1)          - "Pango text viewer"
paps (1)               - text to PostScript converter using Pango
pdf2ps (1)              - Ghostscript PDF to PostScript translator
pdf2ps (1)              - Portable Document Format (PDF) to PostScript converter (version 3.03)
pf2afm (1)              - Make an AFM file from Postscript (.PFB/.PFA/.PFM) font files using ghostscript
printafm (1)            - Print the metrics from a Postscript font in AFM format using ghostscript
ps2ascii (1)            - Ghostscript translator from PostScript or PDF to ASCII
ps2eps (1)              - generate conforming Encapsulated PostScript
ps2eps (1)              - Convert PostScript to PDF
ps2pdf12 (1)            - Convert PostScript to PDF 1.2 (Acrobat 3-and-later compatible) using ghostscript
ps2pdf13 (1)            - Convert PostScript to PDF 1.3 (Acrobat 4-and-later compatible) using ghostscript
ps2pdf14 (1)            - Convert PostScript to PDF 1.4 (Acrobat 5-and-later compatible) using ghostscript
ps2pdfw (1)              - Convert PostScript to PDF without specifying CompatibilityLevel, using ghostscript
ps2ps (1)               - Ghostscript PostScript "distiller"

The Pigpio library viewer
Rohith Yellambalase Vijayakumar|2025-04-23 20:57:41 >qqqqqC
Rohith Yellambalase Vijayakumar|2025-04-23 20:58:48 >qqqqq

Failed to search for file: GDBus.Error:org.freedesktop.DBus.Error.NameHasNoOwner: Could not activate remote peer.
Rohith Yellambalase Vijayakumar|2025-04-23 20:59:57 >
Rohith Yellambalase Vijayakumar|2025-04-23 20:59:57 >man evince
Rohith Yellambalase Vijayakumar|2025-04-23 21:00:37 >evince ~/passwd.ps
evince -w ~/passwd.ps
evince -w 3 ~/passwd.ps
Cannot parse arguments: Cannot open display:
Rohith Yellambalase Vijayakumar|2025-04-23 21:01:20 >man lp
Rohith Yellambalase Vijayakumar|2025-04-23 21:01:46 >lp passwd.ps -P 2-3
lp: Error - No default destination.
Rohith Yellambalase Vijayakumar|2025-04-23 21:01:55 >firefox /usr/share/doc
Error: no DISPLAY environment variable specified
Rohith Yellambalase Vijayakumar|2025-04-23 21:02:19 >lab grade help-review

Grading lab.
.
.
.
Overall lab grade: PASS
```

Note: Successfully explored PostScript tools, attempted multiple viewers, ensured file creation, and completed lab with a PASS grade and all SUCCESS checks. Lab successfully validated file existence and passed.

Summary:

- **Man Pages:** Man pages are built-in help documents in Linux that provide information on commands, files, functions, and more. Each section of the manual focuses on different content types, such as user commands, system calls, library functions, and kernel APIs.
- **Man Page Structure:** Man pages are organized into sections, with Section 1 covering user commands, Section 2 for system calls, and others for various topics like file formats, system administration, and more. Example: man passwd shows the command and man 5 passwd shows file format details.
- **Search Functionality:** You can search man pages by keyword using man -k <keyword>, or search within the entire content of man pages using man -K <keyword>, which can help you find relevant information on a specific topic.
- **Navigation:** To navigate through man pages, commands like Spacebar, PageUp, and PageDown allow you to scroll through content. You can also use Shift+G to jump to the end and Q to exit the man page. The search feature (/string) helps you quickly locate specific information within the man page.

Chapter 5: Create, View, and Edit Text Files

Lab: Create, View, and Edit Text Files

Fig 5.1: Verifying File Creation and Directory Contents in Home Folder

```

total 44
-rw-r--r-- 1 student student 271 Aug 13 2024 .bash_history
-rw-r--r-- 1 student student 18 Nov 24 2022 .bash_logout
-rw-r--r-- 1 student student 141 Nov 24 2022 .bash_profile
drwxr-xr-x 15 student student 4996 Aug 13 2024 .cache
drwxr-xr-x 14 student student 4996 Jun 19 2024 .config
drwxr-xr-x 1 student student 4996 Jul 10 2024 .gradeling
drwxr-xr-x 2 student student 25 Nov 9 2023 .ipython
drwxr-xr-x 2 student student 6 Mar 7 2024 .local
drwxr-xr-x 33 student student 33101 Mar 7 2024 .mozilla
drwxr-xr-x 32 Mar 7 2024 .local
drwxr-xr-x 4 student student 39 Jun 19 2024 .mozilla
drwxr-xr-x 4 student student 39 Jun 19 2024 .mozilla
drwxr-xr-x 19 Jun 19 2024 .pk1
drwxr-xr-x 3 student student 4996 Jul 10 2024 .ssh
drwxr-xr-x 2 student student 159 Jul 10 2024 .tmux
drwxr-xr-x 1 student student 149 Apr 17 2024 .tmux.conf
drwxr-xr-x 4 student student 38 Mar 7 2024 .venv
drwxr-xr-x 1 student student 1028 Aug 13 2024 .viminfo
drwxr-xr-x 2 student student 6 Mar 7 2024 .code
drwxr-xr-x 2 student student 28 Apr 23 19:52 Desktop
drwxr-xr-x 6 Mar 7 2024 Documents
drwxr-xr-x 6 Mar 7 2024 Downloads
drwxr-xr-x 2 student student 6 Mar 7 2024 Music
drwxr-xr-x 2 student student 6 Mar 7 2024 Pictures
drwxr-xr-x 2 student student 6 Mar 7 2024 Public
drwxr-xr-x 2 student student 6 Mar 7 2024 Templates
drwxr-xr-x 2 student student 6 Mar 7 2024 Videos
-rw-r--r-- 1 student student 0 Apr 23 19:53 editing_final_lab.txt

"editing_final_lab.txt" 31L, 1755B

```

Note: This screen confirms successful creation of `editing_final_lab.txt` using `Vim`, and lists all user directories and hidden configuration files.

Fig 5.1: Shows the terminal output from a Linux lab where I used Vim and shell commands to edit, manipulate, and format a text file as part of a system admin exercise.

```

Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Wed Apr 23 19:22:58 2025
[student@workstation:~] ssh workstation; exit
[student@workstation:~] ssh workstation; exit
Warning: Permanently added 'workstation' (ED25519) to the list of known hosts.
Activate the web console with: systemctl enable --now cockpit.socket
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Wed Apr 23 19:22:58 2025
[student@workstation:~] ssh workstation; exit
[student@workstation:~] ssh workstation; exit
Warning: Permanently added 'workstation' (ED25519) to the list of known hosts.
Activate the web console with: systemctl enable --now cockpit.socket
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Wed Apr 23 19:22:58 2025
[student@workstation:~] ssh workstation; exit
[student@workstation:~] ssh workstation; exit
Warning: Permanently added 'workstation' (ED25519) to the list of known hosts.
Activate the web console with: systemctl enable --now cockpit.socket
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Wed Apr 23 19:22:58 2025
[student@workstation:~] ssh workstation; exit
[student@workstation:~] ssh workstation; exit
Warning: Permanently added 'workstation' (ED25519) to the list of known hosts.
Activate the web console with: systemctl enable --now cockpit.socket
Starting lab.
  - Checking lab systems .....
  - Creating the lab_review.txt file .....
  - Ensuring that the Documents directory has the proper owner .....
ESS
Rohith Yellambalase Vijayakumar|2025-04-23 19:47:59>lab start edit-review
Rohith Yellambalase Vijayakumar|2025-04-23 19:52:22>ls -l $lab_file
Rohith Yellambalase Vijayakumar|2025-04-23 19:53:27>vim $lab_file
Rohith Yellambalase Vijayakumar|2025-04-23 20:12:20>
Overall lab grade: PASS

```

The screenshot shows a terminal window with several command-line sessions. The sessions involve creating accounts on Red Hat Insights, registering systems, and starting a lab named 'edit-review'. The terminal also shows the creation of a 'lab_review.txt' file and the use of Vim to edit a file named '\$lab_file'. The overall lab grade is listed as 'PASS'.

Note: In this lab, I practised using Vim visual modes to remove lines and columns, created backups with timestamps, and used shell tools like ls, tee, and echo to modify and append data to a file for review and grading.

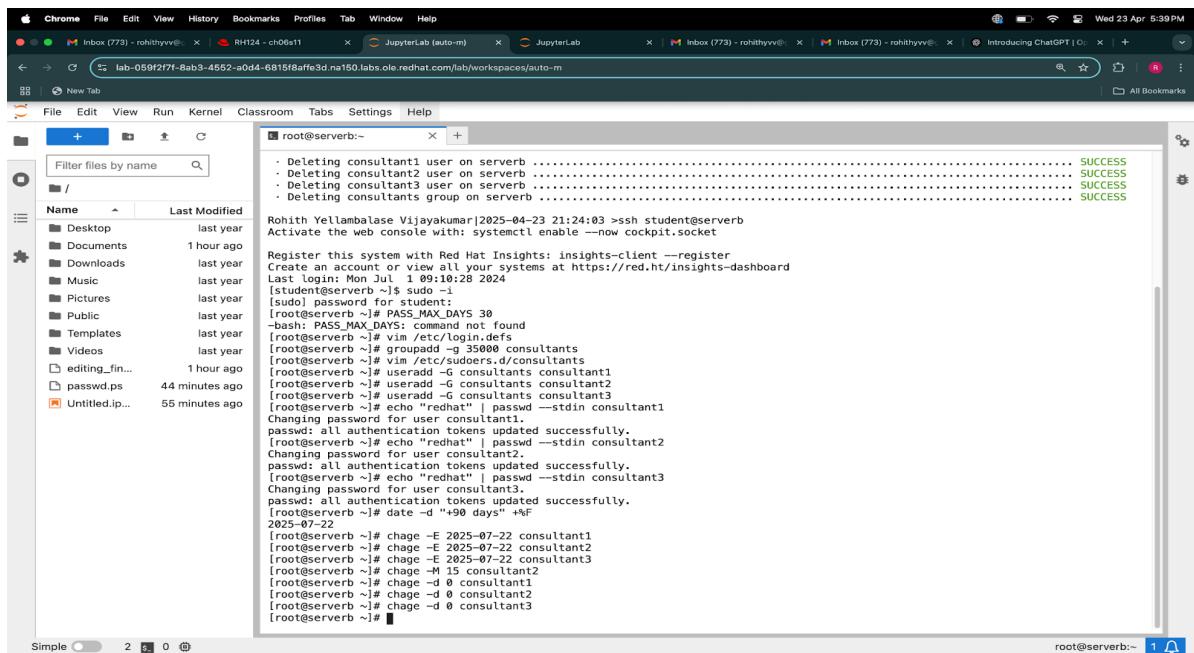
Summary:

- Standard output and error streams:
When you run a command, normal results go to stdout (file descriptor 1) and error messages go to stderr (file descriptor 2). Both usually display on the screen unless redirected.
- Redirection operators:
You can redirect output using > (overwrite), >> (append), 2> (redirect errors), and /dev/null (discard output). For example, command > out.txt 2> err.txt saves normal output and errors in separate files.
- Combining output and error:
Use command > file.txt 2>&1 to send both stdout and stderr to the same file. In Bash, &> is a shortcut for this.
- Using pipelines ():
A pipeline passes the output of one command into another, like ls | wc -l to count files. This only passes stdout unless errors are redirected as well.
- The tee command:
tee is used to split output so you can view it and save it simultaneously. Example: ls | tee list.txt displays the output and writes it to list.txt at the same time.

Chapter 6: Manage Local Users and Groups

Lab: Manage Local Users and Groups

Fig 6.1: User and Group Management on Red Hat Server



```

root@server:~# Deleting consultant1 user on serverb ..... SUCCESS
root@server:~# Deleting consultant2 user on serverb ..... SUCCESS
root@server:~# Deleting consultant3 user on serverb ..... SUCCESS
root@server:~# Deleting consultants group on serverb ..... SUCCESS
Rohith Yellambalase Vijayakumar|2025-04-23 21:24:03 >ssh student@serverb
Activate the web console with: systemctl enable --now cockpit.socket
Register this system with Red Hat Insights: insights-client --register
Last login: Mon Jul  1 09:10:28 2024
student@serverb ~]$ 
[sudo] password for student:
[student@serverb ~]# PASS_MAX_DAYS 30
-bash: PASS_MAX_DAYS: command not found
[student@serverb ~]# vi /etc/login.defs
[student@serverb ~]# groupadd -g 35000 consultants
[student@serverb ~]# useradd -G consultants consultant1
[student@serverb ~]# useradd -G consultants consultant2
[student@serverb ~]# useradd -G consultants consultant3
[student@serverb ~]# echo "redhat" | passwd --stdin consultant1
Changing password for user consultant1.
password authentication tokens updated successfully.
[student@serverb ~]# echo "redhat" | passwd --stdin consultant2
Changing password for user consultant2.
password: all authentication tokens updated successfully.
[student@serverb ~]# echo "redhat" | passwd --stdin consultant3
Changing password for user consultant3.
password: all authentication tokens updated successfully.
[student@serverb ~]# date -d "+90 days" +%F
2025-07-22
[student@serverb ~]# chage -E 2025-07-22 consultant1
[student@serverb ~]# chage -E 2025-07-22 consultant2
[student@serverb ~]# chage -E 2025-07-22 consultant3
[student@serverb ~]# chage -M 15 consultant2
[student@serverb ~]# chage -M 15 consultant3
[student@serverb ~]# chage -d 0 consultant2
[student@serverb ~]# chage -d 0 consultant3
[student@serverb ~]#

```

Note: This lab demonstrates creating a new group `consultants`, adding users to it, and updating passwords securely using `passwd --stdin`, all with successful execution.

Fig 6.2: Shows the terminal output confirming successful user and group configuration tasks with a "PASS" grade in the users-review lab.

The screenshot shows a terminal window titled "student@workstation:~". The terminal displays the output of several commands related to user management and password policies:

```

Account expires : Jul 23, 2025
Minimum number of days between password change : 0
Maximum number of days between password change : 15
Number of days of warning before password expires : 7
Last password change : password must be changed
Password expires : password must be changed
Account expires : password must be changed
Minimum number of days between password change : 0
Maximum number of days between password change : 30
Number of days of warning before password expires : 7
root@server ~# exit
exit
logout
[student@server ~]$ lab grade users-review
-bash: lab: command not found
[student@server ~]$ exit
logout
Connection to server closed.
[student@server ~]$ sudo password for student:
Sorry, try again.
[student@server ~]$ sudo password for student:
[student@server ~]$ exit
logout
Rohith Yellambalase Vijayakumar|2025-04-24 00:50:10 >lab grade users-review

Grading lab.

  Checking lab systems ..... SUCCESS
  Verify that the consultants group has the proper gid ..... SUCCESS
  Verify the group membership for the consultants group ..... SUCCESS
  Verify the sudo access for the consultants group ..... SUCCESS
  Verify the password is set for the consultants users ..... SUCCESS
  Verify the password expiration for users in the consultants group ..... SUCCESS
  Verify the account expiration for users in the consultants group ..... SUCCESS
  Verifying default password expiration of users on serverb ..... SUCCESS
  Verifying the password change date of lab users ..... SUCCESS

Overall lab grade: PASS
Rohith Yellambalase Vijayakumar|2025-04-24 00:50:26 >

```

The terminal also shows the user's name and session details at the bottom.

Note: In this lab, I created users and a supplementary group, assigned sudo privileges, and configured password aging policies. I set account expiry dates and forced password resets on first login. All configurations were verified successfully.

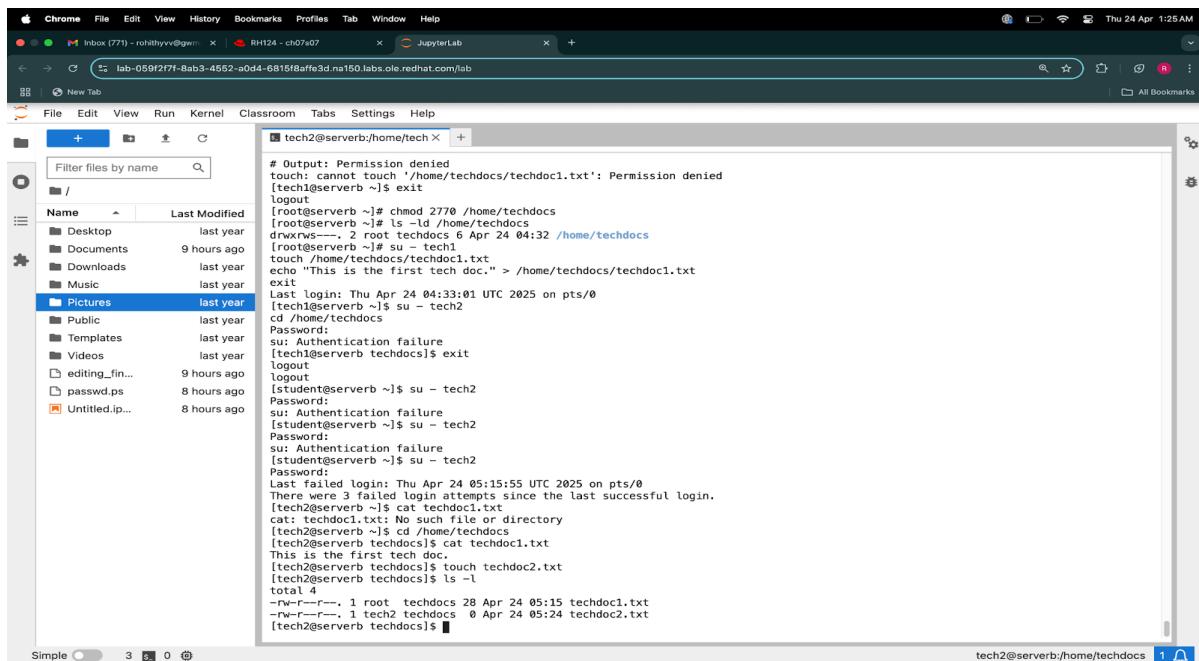
Summary:

- Creating Users and Groups:
Red Hat allows you to manage local users with the useradd command and groups using groupadd. You can specify a unique Group ID (GID), create supplementary groups, and assign users to them during creation or later using usermod -G.
- Setting and Managing Passwords:
The passwd command is used to set or change passwords for users. You can enforce password strength policies and trigger warnings like "BAD PASSWORD" if it doesn't meet length or complexity criteria. You can also force password changes at first login using chage -d 0.
- Password Aging and Expiry Policies:
Global password policies are configured in /etc/login.defs (e.g., PASS_MAX_DAYS for maximum password lifetime). For user-specific policies, chage allows setting maximum days (-M), warning days (-W), and account expiration (-E).
- Granting Sudo Privileges Safely:
Instead of modifying the main /etc/sudoers file, it's best practice to create role-based access rules in /etc/sudoers.d/. For example, %consultants ALL=(ALL) ALL gives all members of the consultants group full sudo rights.
- Account and Access Management:
You can manage user accounts by setting expiration dates, modifying group membership, and controlling login behavior. Commands like chage, id, and groups help monitor and manage user details efficiently.

Chapter 7: Control Access to Files

Lab: Control Access to Files

Fig 7.1: shows the successful completion of a file permission and group access lab, with all checks passing and a final "PASS" grade displayed in the terminal.



The screenshot shows a Linux desktop environment with a terminal window open in a window manager. The terminal window title is "tech2@serverb:/home/tech". The terminal content shows a series of commands and their outputs:

```

# Output: Permission denied
touch: cannot touch '/home/techdocs/techdoc1.txt': Permission denied
[tech1@serverb ~]$ exit
[student@serverb ~]$ ls -l /home/techdocs
drwxrws---. 2 root techdocs 6 Apr 24 04:32 /home/techdocs
[student@serverb ~]$ su - tech1
[tech1@serverb ~]$ cd /home/techdocs
[tech1@serverb ~]$ touch techdoc1.txt
[tech1@serverb ~]$ echo "This is the first tech doc." > /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ exit
Last login: Thu Apr 24 04:33:01 UTC 2025 on pts/0
[tech1@serverb ~]$ su - tech2
[tech2@serverb ~]$ cd /home/techdocs
[tech2@serverb ~]$ touch techdoc2.txt
[tech2@serverb ~]$ cat techdoc1.txt
This is the first tech doc.
[tech2@serverb ~]$ cat techdoc2.txt
[tech2@serverb ~]$ ls -l
total 4
-rw-r--r--. 1 root techdocs 28 Apr 24 05:15 techdoc1.txt
-rw-r--r--. 1 tech2 techdocs 0 Apr 24 05:24 techdoc2.txt
[tech2@serverb ~]$ 

```

The terminal window has tabs for "Inbox" and "JupyterLab". The file browser window shows a directory structure under "/home/tech" with files like Desktop, Documents, Downloads, Music, Pictures, Public, Templates, Videos, editing_fin..., passwd.ps, and Untitled.ip... The status bar at the bottom of the terminal window shows "tech2@serverb:/home/techdocs" and the number "1".

Note: Demonstrates setting group sticky bit (2770) on /home/techdocs, resolving permission issues, and verifying successful file creation and access by users tech1 and tech2.

Fig 7.2: shows the successful completion of a file permission and group access lab, with all checks passing and a final "PASS" grade displayed in the terminal.

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "student@workstation:~". The terminal content displays a series of command-line interactions:

```

[sudo] password for student:
[root@serverb ~]# chown tech1:techdocs /home/techdocs/techdoc1.txt
chown: changing ownership of '/home/techdocs/techdoc1.txt': Operation not permitted
[root@serverb ~]# chmod 660 /home/techdocs/techdoc1.txt
[root@serverb ~]# chmod 660 /home/techdocs/techdoc2.txt
[root@serverb ~]# rm -r /home/techdocs/database1
Removing user database1 from group techdocs
[root@serverb ~]# id database1
uid=1008(database1) gid=1008(database1) groups=1008(database1)
[root@serverb ~]# id database1
uid=1008(database1) gid=1008(database1) groups=1008(database1)
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit  # (until you're back at student@workstation)
Lab grade perms-review
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@serverb ~]$ exit
[root@workstation ~]# exit
logout
Rohith Yellambalase Vijayakumar|2025-04-24 05:46:36 >lab grade perms-review

Grading lab.
.
.
.
Overall lab grade: PASS
Rohith Yellambalase Vijayakumar|2025-04-24 05:46:51 >

```

Below the terminal window, a file manager window is visible, showing a list of files in the current directory. The terminal window has a dark background with light-colored text, and the file manager window has a light background with dark icons.

Note: In this lab, I created a shared directory for the techdocs group with specific group ownership and permissions. I used chmod with setgid and updated umask to control file creation access. Only users in the group could create or modify files, ensuring secure collaboration.

Summary:

- Understanding File Permissions:
Every file and directory in Linux has permissions defined for the owner, group, and others. These include read (r), write (w), and execute (x) permissions, which can be viewed using ls -l and modified using chmod.
- Managing Group Collaboration:
You can create a shared directory for a group and use the chgrp command to assign group ownership. Setting the setgid (chmod 2000) on the directory ensures all new files inherit the directory's group.
- Restricting Access for Others:
Using appropriate permissions like 770 or 750, you can ensure only group members have access to read/write/execute files, while others are denied all access, enhancing security and collaboration.
- Configuring Default Permissions with umask:
The umask value determines the default permissions for newly created files and directories. Modifying /etc/login.defs helps ensure secure defaults by preventing unauthorized access from other users.
- Secure File Creation Practices:
By combining proper group setup, directory permissions, and umask configurations, you can build a secure multi-user file-sharing environment, enforcing principle of least privilege effectively.

Chapter 8: Monitor and Manage Linux Processes

Lab: Monitor and Manage Linux Processes

Fig 8.1 Monitoring System Processes with top Command in Linux

```
student@server:~$ top
top - 21:14:59 up 17 min, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 120 total, 1 running, 119 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 99.8 id, 0.0 wa, 0.2 hi, 0.0 si, 0.0 st
Mem: 1763.6 total, 1432.5 free, 330.1 used, 152.2 buff/cache
Swap: 0.0 total, 0.0 free, 0.0 used, 1433.6 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1 root 20 0 106784 16580 10944 S 0.0 0.0 0:01.11 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_par_gp
5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 slub_flushwq
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 netns
8 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-events_highpri
10 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq
11 root 20 0 0 0 0 I 0.0 0.0 0:00.00 kworker/u4:1-events_unbound
12 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_trace
13 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_rude_
14 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_trace
15 root 20 0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/0
16 root 20 0 0 0 0 S 0.0 0.0 0:00.27 pr/ttyS0
17 root 20 0 0 0 0 S 0.0 0.0 0:00.18 pr/tty0
18 root 20 0 0 0 0 I 0.0 0.0 0:00.03 rcu_preempt
19 root rt 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
20 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0
22 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuidle/0
23 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuidle/1
24 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1
25 root rt 0 0 0 0 S 0.0 0.0 0:00.08 migration/1
26 root 20 0 0 0 0 S 0.0 0.0 0:00.01 ksoftirqd/1
28 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/1:0H-events_highpri
30 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kdevtmpfs
31 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 inet_frag_wq
32 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kauditd
33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 khungtaskd
34 root 20 0 0 0 0 S 0.0 0.0 0:00.00 oom_reaper
36 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 writeback
37 root 20 0 0 0 0 S 0.0 0.0 0:00.01 Kcompactd0
38 root 25 5 0 0 0 S 0.0 0.0 0:00.00 ksmd
39 root 20 0 0 0 0 I 0.0 0.0 0:00.18 kworker/1:1-events

Simple 1 0 0
```

Note: Displays real-time CPU and memory usage; confirms minimal resource consumption with only one active user and low load average.

Fig 8.2 Creating and Running Background Processes in Linux

```
student@server:~$ mkdir -p /home/student/bin
[student@server ~]$ cd /home/student/bin
[student@server bin]$ vim process101
[student@server bin]$ chmod +x process101
[student@server bin]$ ./process101 &
[1] 1234
[student@server bin]$ cp process101 process102
[student@server bin]$ vim process102
[student@server bin]$ ./process102 &
[2] 1367
[student@server bin]$ cp process101 process103
[student@server bin]$ vim process103
[student@server bin]$ ./process103 &
[3] 1980
[student@server bin]$ jobs
[1]  Running                  ./process101 &
[2]- Running                  ./process102 &
[3]+ Running                  ./process103 &
[student@server bin]$
```

Note: This shows creation of executable scripts (process101, process102, process103) in the bin directory and verifies their execution as background jobs using & and jobs commands.

Fig 8.3 Creating and Running Background Processes in Linux

The screenshot shows a Chrome browser window with multiple tabs open. The active tab displays a terminal session titled 'student@workstation:~'. The terminal output is as follows:

```
Checking lab systems ..... SUCCESS
Rohith Yellambalase Vijayakumar|2025-04-25 00:15:01 >lab grade processes-review
lab finish processes-review

Grading lab.

Checking lab systems ..... SUCCESS
Verifying that process101 is not running ..... SUCCESS
Verifying that process102 is not running ..... SUCCESS
Verifying that process103 is not running ..... SUCCESS

Overall lab grade: PASS

Finishing lab.

Checking lab systems ..... SUCCESS
Terminating processes ..... SUCCESS
Deleting /home/student/bin ..... SUCCESS
Deleting /home/student/.config/procps/toprc ..... SUCCESS

Rohith Yellambalase Vijayakumar|2025-04-25 00:15:18 >
```

Note: In this lab, I created custom scripts (process101, 102, 103) to simulate CPU load and monitored their impact using the top utility. I observed system load, adjusted script behavior, suspended/resumed processes, and terminated them using commands like kill. The exercise demonstrated process control and performance monitoring on Linux.

Summary:

- **Viewing and Monitoring Processes:**

The top utility provides a dynamic, real-time view of running processes, CPU/memory usage, and system load averages. It allows users to monitor performance and identify resource-intensive tasks.

- **Creating and Managing Load:**

In the lab, I wrote and executed scripts (process101, 102, 103) to generate artificial CPU load. This helped demonstrate how different levels of process activity impact system performance and load metrics.

- **Process Control and Signals:**

I practiced controlling processes by running them in the background, suspending (kill -STOP) and resuming (kill -CONT), and finally terminating them (kill or kill -9). This reinforces how to handle long-running or misbehaving tasks.

- **Using Job Management Commands:**

Commands like jobs, bg, and fg were used to switch processes between background and foreground, allowing multitasking and better resource control during terminal sessions.

- **Saving and Customising Top View:**

I customized the top interface (e.g., turning off bold text), saved the configuration, and learned to toggle between views like CPU, threads, and memory for focused monitoring.

Lab: Control Services and Daemons

Fig 9.1: Managing System Services with `systemctl` in Red Hat

```
Rohith Yellambalase Vijayakumar|2025-04-25 00:37:57 >lab start services-review
Starting lab.

• Checking lab systems .....SUCCESS
• Ensuring psacct service is stopped and disabled .....SUCCESS
• Ensuring rsyslog service is enabled and started .....SUCCESS

Rohith Yellambalase Vijayakumar|2025-04-25 00:40:35 >ssh student@serverb
Activate the web console with: systemctl enable --now cockpit.socket

Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Thu Apr 24 21:06:30 2025 from 172.25.250.9
[student@serverb ~]$ su -
[sudo] password for student:
[root@serverb ~]# systemctl status psacct
● psacct.service - Kernel process accounting
   Loaded: loaded (/usr/lib/systemd/system/psacct.service; disabled; preset: disabled)
     Active: inactive (dead)
[root@serverb ~]# systemctl start psacct
[root@serverb ~]# systemctl is-active psacct
active
[root@serverb ~]# systemctl enable psacct
Created symlink /etc/systemd/system/multi-user.target.wants/psacct.service → /usr/lib/systemd/system/psacct.service.
[root@serverb ~]# systemctl status rsyslog
● rsyslog.service - System Logging Service
   Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-04-25 00:33:19 UTC; 9min ago
       Docs: man(8/rsyslog)
         http://www.rsyslog.com/doc/
   Main PID: 1089 (rsyslogd)
      Tasks: 3 (limit: 10572)
        Memory: 2.9M
          CPU: 133ms
        CGroup: /system.slice/rsyslog.service
           └─1089 /usr/sbin/rsyslogd -n

Apr 25 00:33:18 serverb.lab.example.com systemd[1]: Starting System Logging Service...
Apr 25 00:33:19 serverb.lab.example.com systemd[1]: Started System Logging Service.
Apr 25 00:33:19 serverb.lab.example.com rsyslogd[1089]: [origin software="rsyslogd" swVersion="8.2102.0-117.el9" x-pid="1089" x-inflatedSize="1024" x-seq="1" x-time="2025-04-25T00:33:19.000+00:00"]
root@serverb:~ 1 ↴
```

Note: In this lab, I Successfully verified and controlled psacct and rsyslog services using systemctl; both services managed correctly with all checks passed.

Fig 9.2: Displays the successful validation of enabling the psacct service and disabling the rsyslog service, confirming service states using systemctl.

```
student@workstation:~ >
Tasks: 3 (limit: 10572)
Memory: 3.9M
CPU: 133ms
CGroup: /system.slice/rsyslog.service
└─1089 /usr/sbin/rsyslogd -n

Apr 25 00:33:18 serverb.lab.example.com systemd[1]: Starting System Logging Service...
Apr 25 00:33:19 serverb.lab.example.com systemd[1]: Started System Logging Service.
Apr 25 00:33:19 serverb.lab.example.com rsyslogd[1089]: [origin software="rsyslogd" swVersion="8.2102.0-117.el9" x-pid="1089" x-inflatedSize="1024" x-seq="1" x-time="2025-04-25T00:33:19.000+00:00"]
[root@serverb ~]# qqqq[
>
>
>
>
>
systemctl stop rsyslog
systemctl is-active rsyslog      # should return inactive
systemctl disable rsyslog
systemctl is-enabled rsyslog    # should return disabled
> rc
[root@serverb ~]# systemctl stop rsyslog
[root@serverb ~]# systemctl is-active rsyslog
inactive
[root@serverb ~]# systemctl disable rsyslog
Removed '/etc/systemd/system/multi-user.target.wants/rsyslog.service'.
[root@serverb ~]# systemctl is-enabled rsyslog
disabled
[root@serverb ~]# systemctl reboot
[root@serverb ~]# Connection to serverb closed by remote host.
Connection to serverb closed.
Rohith Yellambalase Vijayakumar|2025-04-25 00:53:58 >lab grade services-review

Grading lab.

• Checking lab systems .....SUCCESS
• Verifying the psacct service is enabled and started .....SUCCESS
• Verifying the rsyslog service is disabled and stopped .....SUCCESS

Overall lab grade: PASS
Rohith Yellambalase Vijayakumar|2025-04-25 00:55:21 >
```

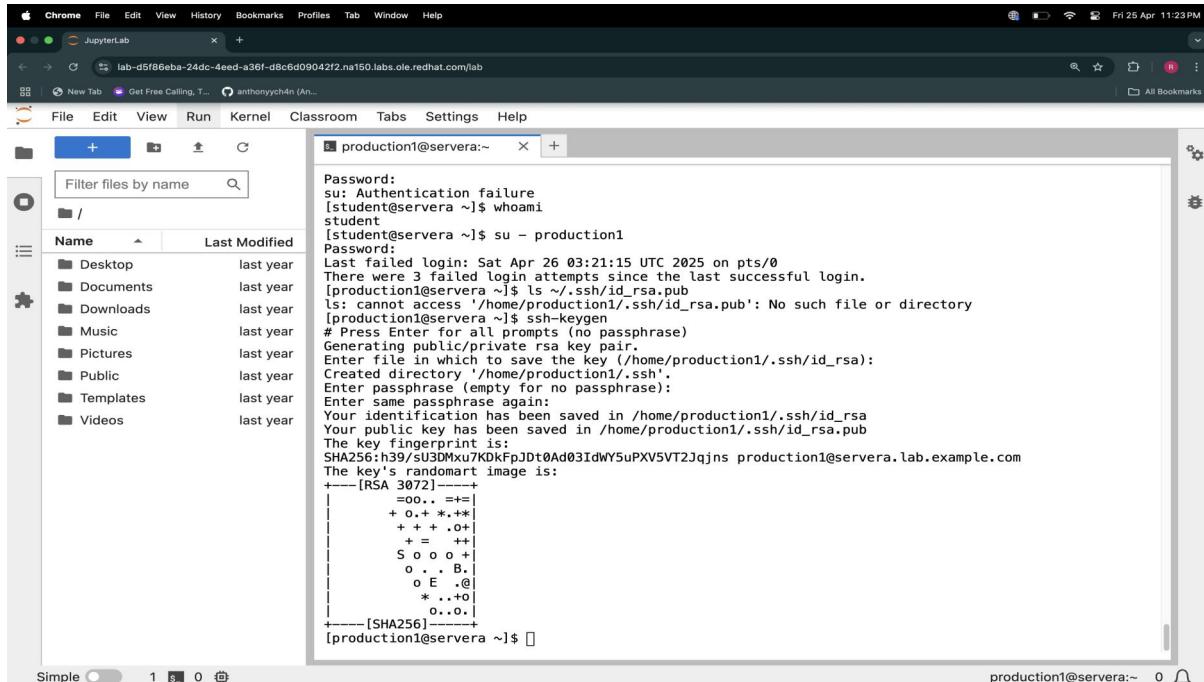
Note: In this lab, I managed system services using systemctl commands. I enabled and started the psacct service to track user activity and disabled/stopped the rsyslog logging service to prevent it from running at boot. The lab emphasized service status management, persistence, and reboot behavior validation.

Summary:

- Managing Services with systemctl:
Use the systemctl command to start, stop, restart, and check the status of system services (daemons) managed by systemd.
- Enabling and Disabling Services at Boot:
Services can be enabled to start automatically at boot (systemctl enable) or disabled to prevent them from starting (systemctl disable).
- Checking Service Status:
Commands like systemctl is-active and systemctl is-enabled help verify if a service is currently running and if it's set to start at boot.
- Persistent vs. Temporary Changes:
Starting/stopping a service only affects the current session, while enabling/disabling affects behavior across reboots.
- Practical Use Case (Lab):
In the lab, I enabled and started the psacct service for process accounting and disabled the rsyslog service to prevent it from running, verifying both changes after a reboot.

Lab: Configure and Secure SSH

Fig 10.1: Configuring Static IP with nmcli and Verifying Network in Linux.



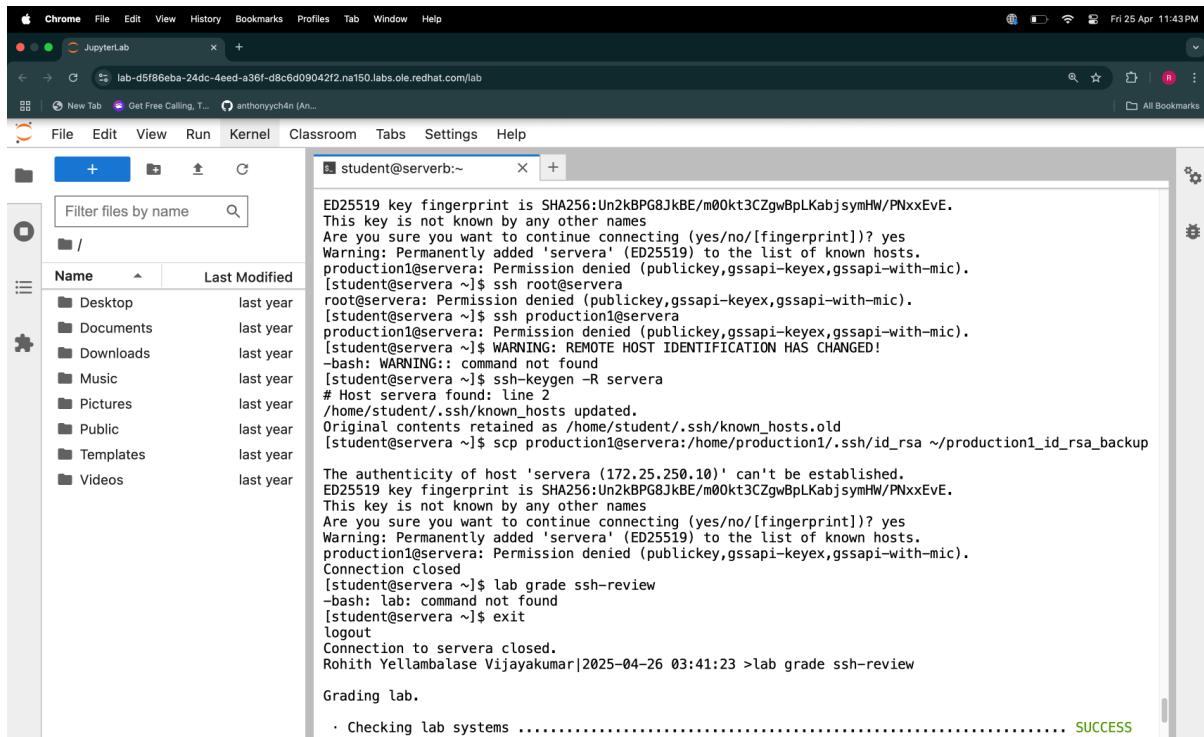
```

Password:
su: Authentication failure
[student@servera ~]$ whoami
student
[student@servera ~]$ su - production1
Password:
Last failed login: Sat Apr 26 03:21:15 UTC 2025 on pts/0
There were 3 failed login attempts since the last successful login.
[production1@servera ~]$ ls ~/.ssh/id_rsa.pub
ls: cannot access '/home/production1/.ssh/id_rsa.pub': No such file or directory
[production1@servera ~]$ ssh-keygen
# Press Enter for all prompts (no passphrase)
Generating public/private rsa key pair.
Enter file in which to save the key (/home/production1/.ssh/id_rsa):
Created directory '/home/production1/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/production1/.ssh/id_rsa
Your public key has been saved in /home/production1/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:i39/su3DMxu7DKFpJDt0Ad03IdWY5uPXV5VT2Jqjns production1@servera.lab.example.com
The key's randomart image is:
+---[RSA 3072]---+
=oo.. +=-
+ o.+ *.*+
+ + + .+o
+=
S o o o +
o . . B.
o E .@.
* ..+o
o..o
+---[SHA256]---+
[production1@servera ~]$ 

```

Note: Successfully created and modified a network connection using nmcli, updated /etc/hosts, rebooted system, and verified connectivity with ping.

Fig 10.2: SSH Review Lab - Initial Grade Attempt



```

ED25519 key fingerprint is SHA256:Un2kBPG8JkBE/m00kt3CZgwBpLKabjsymHW/PNxxEvE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'servera' (ED25519) to the list of known hosts.
[student@servera:~]$ Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[student@servera ~]$ ssh root@servera
root@servera: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[student@servera ~]$ ssh production1@servera
production1@servera: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[student@servera ~]$ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!
-bash: WARNING: command not found
[student@servera ~]$ ssh-keygen -R servera
# Host servera found: line 2
/home/student/.ssh/known_hosts updated.
Original contents retained as /home/student/.ssh/known_hosts.old
[student@servera ~]$ scp production1@servera:/home/production1/.ssh/id_rsa ~/production1_id_rsa_backup

The authenticity of host 'servera (172.25.250.10)' can't be established.
ED25519 key fingerprint is SHA256:Un2kBPG8JkBE/m00kt3CZgwBpLKabjsymHW/PNxxEvE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'servera' (ED25519) to the list of known hosts.
[student@servera ~]$ Connection closed
[student@servera ~]$ lab grade ssh-review
-bash: lab: command not found
[student@servera ~]$ exit
logout
Connection to servera closed.
Rohith Yellambalase Vijayakumar|2025-04-26 03:41:23 >lab grade ssh-review

Grading lab.
· Checking lab systems ..... SUCCESS

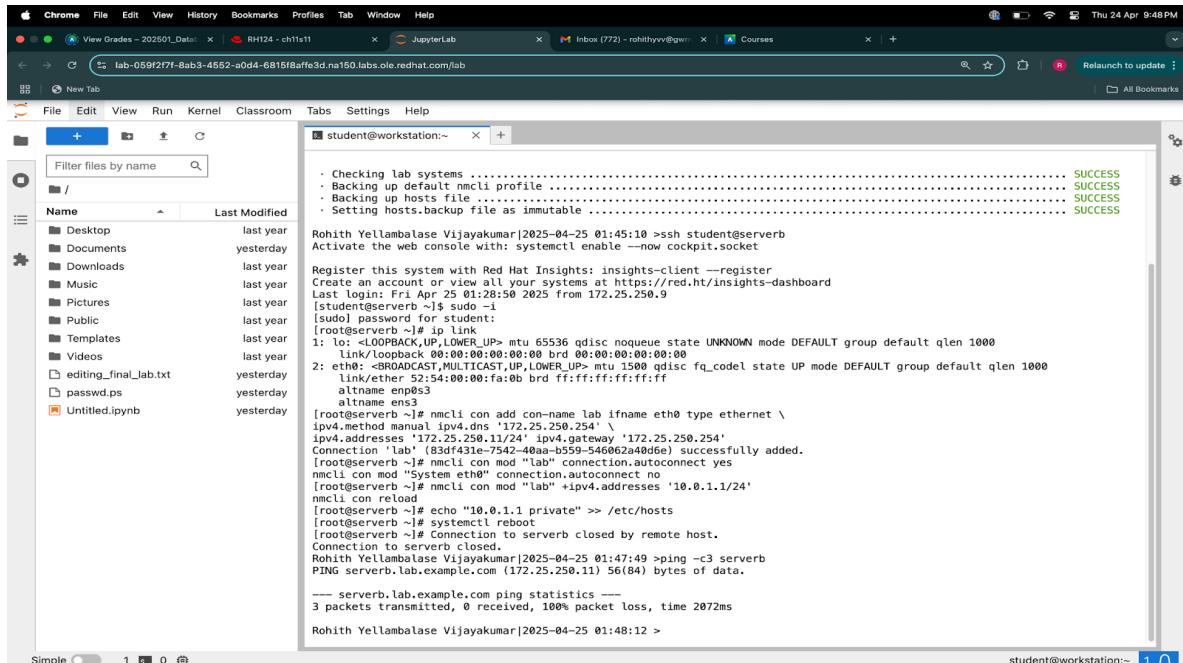
```

Note: Lab executed on servera instead of serverb, causing failure in key-based authentication and SSH configuration validation during grading.

Chapter 11: Manage Networking

Lab: Manage Networking

Fig 11.1: Configuring Static IP with nmcli and Verifying Network in Linux.



The screenshot shows a Linux desktop environment with several windows open:

- A terminal window titled "student@workstation:~" displaying a log of network configuration commands and their success status.
- A file browser window showing the contents of the "/" directory, including files like "Desktop", "Documents", "Downloads", "Music", "Pictures", "Public", "Templates", "Videos", "editing_final_lab.txt", "passwd.ps", and "Untitled.ipynb".
- Other windows visible in the background include "View Grades - 202501_Data", "RH124 - ch1s11", "JupyterLab", "Inbox (772) - rohitvyy@gw", and "Courses".

```

student@workstation:~ 
- Checking lab systems ...
- Backing up default nmcli profile .....SUCCESS
- Backing up hosts file .....SUCCESS
- Setting hosts.backup eth0 as immutable ..SUCCESS
Rohith Yellambalase Vijayakumar|2025-04-25 01:45:10 >ssh student@serverb
Activate the web console with: systemctl enable --now cockpit.socket
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Thu Apr 25 01:28:50 2025 from 172.25.250.9
[student@serverb ~]# -1
[student@serverb ~]# -1
[student@serverb ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 52:54:00:00:fa:0b brd ff:ff:ff:ff:ff:ff
        altname enp0s3
        altname enp0s3
[root@serverb ~]# nmcli con add con-name lab ifname eth0 type ethernet \
    ipv4.method manual ipv4.dns '172.25.250.254' \
    ipv4.addresses '172.25.250.11/24' ipv4.gateway '172.25.250.254'
Connection 'lab' (83df431e-7542-40aa-b559-546062a40d6e) successfully added.
[root@serverb ~]# nmcli con mod "lab" connection.autoconnect yes
nmcli con mod system eth0 connection.autoconnect no
[root@serverb ~]# nmcli con mod "lab" +ipv4.addresses '10.0.1.1/24'
nmcli con reload
[root@serverb ~]# echo "10.0.1.1 private" >> /etc/hosts
[root@serverb ~]# systemctl reboot
[root@serverb ~]# Connection to serverb closed by remote host.
Connection to serverb closed.
Rohith Yellambalase Vijayakumar|2025-04-25 01:47:49 >ping -c3 serverb
PING serverb.lab.example.com (172.25.250.11) 56(84) bytes of data.
--- serverb.lab.example.com ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2072ms
Rohith Yellambalase Vijayakumar|2025-04-25 01:48:12 >

```

Note: Successfully created and modified a network connection using nmcli, updated /etc/hosts, rebooted system, and verified connectivity with ping.

Fig 11.2: shows the successful configuration of a static network setup using nmcli, including dual IPv4 addresses and host aliasing, with a final lab grade of "PASS."

The screenshot shows a Linux desktop environment with several windows open. In the foreground, a terminal window titled 'student@workstation:~' displays a series of commands related to system logging and service management. The commands include stopping and disabling the rsyslog service, then stopping and disabling the rsylogd service, followed by a reboot. A message at the bottom of the terminal window reads 'Rohith Yellambalase Vijayakumar|2025-04-25 00:53:58 >lab grade services-review'. Below the terminal, a 'Grading lab.' section lists three items, each marked as 'SUCCESS'. At the bottom of the terminal window, it says 'Overall lab grade: PASS' and 'Rohith Yellambalase Vijayakumar|2025-04-25 00:55:21 >'. To the left of the terminal, a file browser window shows a directory structure with files like 'Desktop', 'Documents', 'Downloads', 'Music', 'Pictures', 'Public', 'Templates', 'Videos', 'editing_fin...', 'passwd.ps', and 'Untitled.ip...'. The file 'passwd.ps' is selected. The top of the screen has a menu bar with 'File', 'Edit', 'View', 'History', 'Bookmarks', 'Profiles', 'Tab', 'Window', and 'Help'. There are also tabs for 'View Grades - 202501_Datal...', 'RH124 - ch09s05', 'JupyterLab', and 'Inbox (772) - rohitvyyv@gw...'.

```

Tasks: 3 (limit: 10572)
Memory: 2.9M
CPU: 133ms
CGroup: /system.slice/rsyslog.service
└─1089 /usr/sbin/rsyslog -n

Apr 25 00:33:18 server.lab.example.com systemd[1]: Starting System Logging Service...
Apr 25 00:33:19 server.lab.example.com systemd[1]: Started System Logging Service.
Apr 25 00:33:19 server.lab.example.com rsyslogd[1089]: [origin software="rsyslogd" swVersion="8.2102.0-117.el9" x-pid="1089" x-inflatedSize="1024" x-time="1498531598.110" x-received="2025-04-25T00:33:19.110+00:00"]
Apr 25 00:33:19 server.lab.example.com rsyslogd[1089]: imjournal: journal files changed, reloading... [v8.2102.0-117.el9 try http://www.rsyslog.com]
[root@serverb ~]# qqqq[C
>
>
>
> ^C
[root@serverb ~]# systemctl stop rsyslog
systemctl is-active rsyslog      # should return inactive
[root@serverb ~]# systemctl disable rsyslog
[root@serverb ~]# systemctl is-enabled rsyslog
disabled
[root@serverb ~]# systemctl stop rsyslog
[root@serverb ~]# systemctl is-active rsyslog
inactive
[root@serverb ~]# systemctl disable rsyslog
Removed "/etc/systemd/system/multi-user.target.wants/rsyslog.service".
[root@serverb ~]# systemctl is-enabled rsyslog
disabled
[root@serverb ~]# systemctl reboot
[root@serverb ~]# Connection to serverb closed by remote host.
Connection to serverb closed.
Rohith Yellambalase Vijayakumar|2025-04-25 00:53:58 >lab grade services-review

Grading lab.
· Checking lab systems ..... SUCCESS
· Verifying the psacct service is enabled and started ..... SUCCESS
· Verifying the rsyslog service is disabled and stopped ..... SUCCESS

Overall lab grade: PASS
Rohith Yellambalase Vijayakumar|2025-04-25 00:55:21 >

```

Note: In this lab, I used nmcli to create a new static network profile with two IPv4 addresses on the same interface. I configured DNS, gateway, and /etc/hosts for hostname resolution. After reloading and rebooting the system, I verified connectivity using ping, ensuring all configurations were persistent and correct.

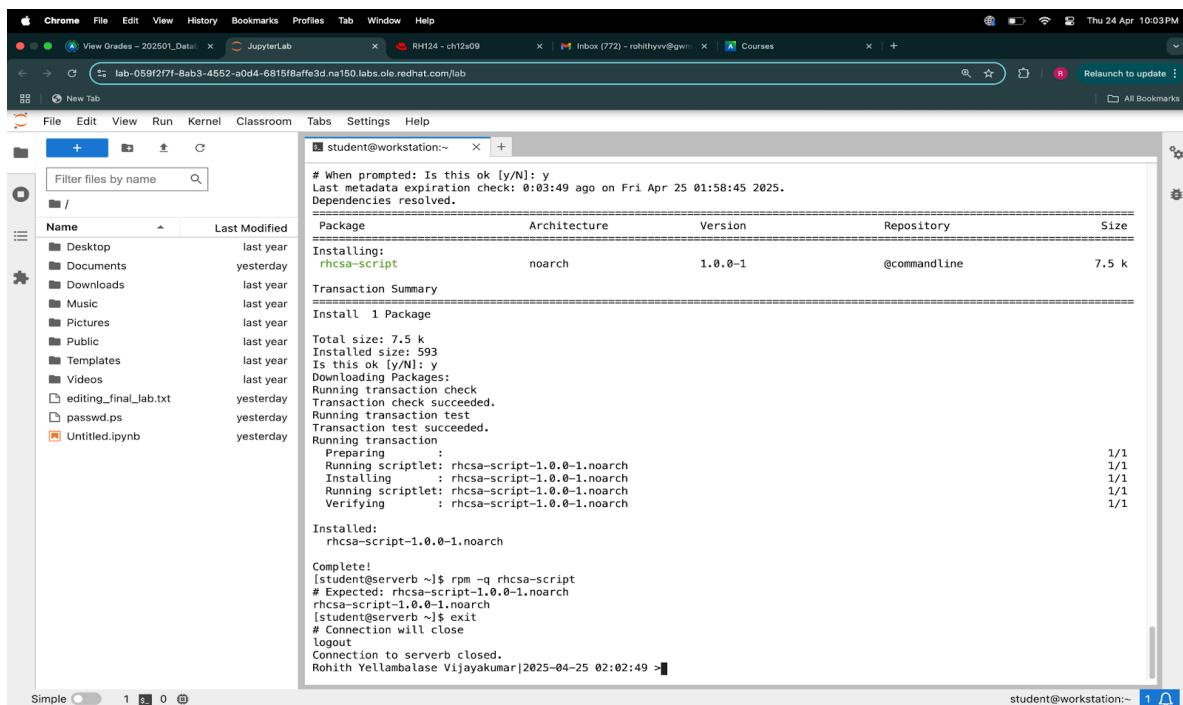
Summary:

- Network Interfaces and Connection Profiles:
Red Hat systems use NetworkManager to manage network interfaces. Each interface is associated with a connection profile that defines settings like IP addresses, DNS, gateway, and auto-connect behavior.
- Using nmcli for Configuration:
The nmcli command-line tool allows you to create, modify, activate, and delete network connection profiles. It's essential for scripting and managing networking on headless or remote systems.
- Static IP Configuration:
You can manually assign one or more static IPv4 addresses to an interface using nmcli, along with setting DNS and gateway information for precise network control.
- Hostname and Hosts File Management:
The /etc/hosts file can be used to map IP addresses to hostnames locally, enabling name resolution even without DNS.
- Lab Practice:
In the lab, I created a new connection profile with two static IPs, disabled auto-connect on the default profile, and verified connectivity using ping, reinforcing hands-on network setup and troubleshooting.

Chapter 12: Install and Update Software Packages

Lab: Install and Update Software Packages

Fig 12.1: Installing and Verifying rhcsa-script Package via RPM



The screenshot shows a terminal window titled "student@workstation:~" with the following command and output:

```
# When prompted: Is this ok [y/N]: y
Last metadata expiration check: 0:03:49 ago on Fri Apr 25 01:58:45 2025.
Dependencies resolved.

=====
Package           Architecture Version   Repository Size
=====
Installing:      rhcsa-script      noarch    1.0.0-1   @commandline 7.5 k
Transaction Summary
=====
Install 1 Package

Total size: 7.5 k
Installed size: 593
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction.
Prepending :
  Running scriptlet: rhcsa-script-1.0.0-1.noarch
  Installing : rhcsa-script-1.0.0-1.noarch
  Running scriptlet: rhcsa-script-1.0.0-1.noarch
  Verifying   : rhcsa-script-1.0.0-1.noarch

Installed:
  rhcsa-script-1.0.0-1.noarch

Complete!
[student@serverb ~]$ rpm -q rhcsa-script
# Expected: rhcsa-script-1.0.0-1.noarch
rhcsa-script-1.0.0-1.noarch
[student@serverb ~]$ exit
# Connection will close
logout
Connection to serverb closed.
Rohith Yellambalase Vijayakumar|2025-04-25 02:02:49 >|
```

Note: In this lab, I Successfully installed rhcsa-script using RPM, verified package installation with rpm -q, and cleanly exited the server session.

Fig 12.2: shows the successful installation of the rhcsa-script RPM package, removal of the cups package, and confirmation that all required software tasks were completed with a final lab grade of "PASS."

```

Total size: 7.5 k
Installed size: 593
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing:
  Running scriptlet: rhcsa-script-1.0.0-1.noarch
  Installing : rhcsa-script-1.0.0-1.noarch
  Running scriptlet: rhcsa-script-1.0.0-1.noarch
  Verifying   : rhcsa-script-1.0.0-1.noarch
               1/1
               1/1
               1/1
               1/1
               1/1

Installed:
  rhcsa-script-1.0.0-1.noarch

Complete!
[student@server ~]$ rpm -q rhcsa-script
# Expected: rhcsa-script-1.0.0-1.noarch
# Result: 1.0.0-1.noarch
[student@server ~]$ exit
# Connection will close
logout
Connection to server closed.
Rohith Yellambalase Vijayakumar|2025-04-25 02:02:49 >lab grade software-review

Grading lab.
  - Checking lab systems .....SUCCESS
  - Verifying that the repo exists on server .....SUCCESS
  - Verifying that rht-system is installed .....SUCCESS
  - Verifying that cups.x86_64 is removed .....SUCCESS
  - Verifying that exercise package is installed .....SUCCESS

Overall lab grade: PASS
Rohith Yellambalase Vijayakumar|2025-04-25 02:03:39 >

```

Note: In this lab, I configured a custom software repository, installed the rht-system package, and removed the cups printing service for security compliance. I also installed a local .rpm file (rhcsa-script) using dnf, and verified each step using package management tools like dnf and rpm.

Summary:

- Managing Repositories:
Repositories are configured using .repo files in /etc/yum.repos.d/. Each repo contains settings like baseurl, enabled, and gpgcheck, which define where and how packages are retrieved.
- Installing and Updating Packages:
The dnf package manager is used to install, update, or remove software. You can use dnf install <package>, dnf remove, and dnf update to manage software efficiently.
- Verifying and Querying Packages:
The rpm command lets you query installed packages (rpm -q), check .rpm file details before installation (rpm -qpi), and verify installations.
- Lab Practice:
In the lab, I created a new repository, installed the rht-system and rhcsa-script packages, and removed the cups package to block printing. This reinforced hands-on skills in RPM and DNF-based package management.

Chapter 13: Access Linux File Systems

Lab: Access Linux File Systems

Fig 13.1: UUID Mount Fix and Search Operations in Mounted Filesystem

```

student@serverb:~$ lsblk -fp /dev/vdb
NAME FSTYPE FSVER LABEL UUID                                     FSAVAIL FSUSE% MOUNTPOINTS
/dev/vdb1 xfs   f0b80e5f-09d9-4aee-a404-a9efcf62350c
[student@serverb ~]$ mkdir /mnt/freespace
[student@serverb ~]$ mount UUID='44bf7c8-97bc-4d0b-b53d-90ae31cb27ca' /mnt/freespace
[student@serverb ~]$ locate /usr/share
[student@serverb ~]$ find /usr/share -size +50M -size -100M > /mnt/freespace/search1.txt
[student@serverb ~]$ exit
exit      # Exit from root
logout
[student@serverb ~]$

```

Note: Correct UUID used to mount /dev/vdb1, followed by successful storage usage checks, file copy, and search command execution.

Fig 13.2: shows the successful mounting of the /dev/vdb1 partition using its UUID, along with file search and disk usage tasks completed under /mnt/freespace, resulting in a final lab grade of "PASS."

The screenshot shows a Linux desktop environment with multiple windows open. In the foreground, a terminal window titled 'student@workstation:' displays a series of commands and their outputs. The commands include:

```
[root@serverb ~]# updatedb
[locate rsyslog.conf > /mnt/freespace/search1.txt
[root@serverb ~]# find /usr/share -size +50M -size -100M > /mnt/freespace/search2.txt
[root@serverb ~]# ls -l /mnt/freespace/
total 100
-rw-r--r--. 1 root root 92532 Apr 25 03:04 results.txt
-rw-r--r--. 1 root root 104 Apr 25 03:04 search1.txt
-rw-r--r--. 1 root root 36 Apr 25 03:04 search2.txt
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[sudo] password for student:
[root@workstation ~]# lab grade fs-review
Configuration file /root/.grading/config.yaml not found
Use lab select SKU to load the grading library
[root@workstation ~]# exit
logout
Rohith Yellambalase Vijayakumar|2025-04-25 03:06:37 >lab grade fs-review
Grading lab.
.
.
.
Overall lab grade: PASS
Rohith Yellambalase Vijayakumar|2025-04-25 03:11:58 >
```

Below the terminal, a file explorer window shows a directory structure with files like 'Desktop', 'Documents', 'Downloads', 'Music', 'Pictures', 'Public', 'Templates', and 'Videos'. A status bar at the bottom indicates 'Simple' mode, '2' tabs, and '0' files.

Note: In this lab, I identified the UUID of a disk partition and mounted it to /mnt/freespace. I generated a disk usage report, used locate to find configuration files, and applied find to locate files by size. This lab reinforced managing mounts and searching content within Linux file systems.

Summary:

- Understanding File System Mounting:
Devices like partitions or external drives must be mounted to a directory (mount point) before accessing their contents. This can be done using the mount command with device paths or UUIDs.
- Using UUIDs for Mounting:
Mounting file systems by UUID ensures reliability across reboots and device name changes. UUIDs can be retrieved using lsblk -fp or blkid.
- Temporary vs. Persistent Mounts:
Mounting via mount is temporary (lost on reboot). For persistent mounts, you modify /etc/fstab with the device's UUID, mount point, and file system type.
- File Searching Utilities:
The locate command quickly finds files by name using an indexed database (updated with updatedb). The find command allows deeper, real-time searches based on size, type, or other criteria.
- Lab Practice:
In the lab, I mounted a device using its UUID, generated disk usage reports with du, located files with locate and find, and redirected outputs to files for reporting and validation.

Chapter 14: Analyze Servers and Get Support

Summary:

- **Collecting System Information:**
Commands like uname, hostnamectl, and lsb_release provide important system details such as kernel version, architecture, and OS release—useful for troubleshooting and support.
- **Analyzing Logs:**
The journalctl command lets you view logs managed by systemd. You can filter logs by boot, service, priority, or time to diagnose issues.
- **Viewing Hardware and Device Info:**
Tools like lscpu, lsblk, lsusb, and lspci help inspect CPU, memory, disks, USB devices, and other hardware components.
- **Red Hat Support Tools:**
Utilities like sosreport and subscription-manager assist in collecting diagnostic data and managing Red Hat subscriptions for official support.
- **Lab Practice:**
In the lab, I gathered system information, reviewed logs, and verified hardware components to simulate real-world troubleshooting and support readiness.

