Day 6 - 31st May 2025

Linux (cont.)

Plz refer :-

Doc 11, 13, 14,15 - in Docs to Study

Task 1:

RegEX Symbols in linux

List them down with description

Basic Symbols:

. - Matches any single character

\* - Matches zero or more occurrences of previous character

^ - Matches beginning of line

$ - Matches end of line

[] - Matches any single character in brackets

[^] - Matches any single character NOT in brackets

Quantifiers:

? - Matches zero or one occurrence

+ - Matches one or more occurrences

{n} - Matches exactly n times

{n,} - Matches at least n times

{n,m} - Matches at least n times, but not more than m times

Character Classes:

[0-9] - Matches any digit

[a-z] - Matches any lowercase letter

[A-Z] - Matches any uppercase letter

[a-Z] - Matches any letter

\w - Matches word characters [a-zA-Z0-9\_]

\W - Matches non-word characters

\d - Matches digits [0-9]

\D - Matches non-digits

\s - Matches whitespace (space, tab, newline)

\S - Matches non-whitespace

Special Characters:

\ - Escape character

| - Alternation (OR)

() - Groups expressions

\b - Word boundary

\B - Non-word boundary 9

5 min 9.43 to 9.48

Task 2:

If you are aware of Linux OS.. can you tell me the feature of Linux.

Linux is like a free, customizable, and secure digital Swiss Army knife for computers. It's open-source, meaning anyone can see and improve its code, making it highly transparent and trustworthy. Think of it as a system that can run on almost anything - from tiny smartphones to massive supercomputers! It's incredibly stable (rarely crashes), efficient (runs well even on older computers), and secure (very resistant to viruses). Users can completely customize how it looks and works, unlike the one-size-fits-all approach of other operating systems. Linux is great at multitasking and can update most of its parts without needing to restart. It comes in different flavors (called distributions) like Ubuntu, Fedora, or Mint, each designed for different types of users. The best part? It's completely free, and there's a massive community of users ready to help if you get stuck. Whether you're a beginner or a tech expert, Linux has something for everyone while keeping your data private and secure.

5 min 9.49 to 9.53

Task 3:

What is Kernal ? can you explain about it in your words..

Kernel IS the heart of your computer system - a tireless worker that never sleeps. When you turn on your computer, the kernel springs to life first, like a master coordinator waking up before everyone else. It immediately starts mapping out the entire system, discovering what hardware is available - from the CPU's processing power to the last byte of RAM, from the smallest USB port to the largest hard drive. As the system comes alive, the kernel creates an efficient system of communication highways between all these components. It's like having a universal translator who understands every language - when a program needs something, it speaks to the kernel, and the kernel translates this request into a language the hardware understands. The kernel maintains strict security at all times, acting as a vigilant gatekeeper - no program can directly access hardware or memory without going through the kernel's security checks. It's constantly juggling multiple tasks: managing memory like a strict librarian (deciding who gets what space and for how long), scheduling processes like a precise timekeeper (ensuring every program gets its fair share of CPU time), and handling hardware requests like a skilled traffic controller (making sure data flows smoothly between components). When resources run low, it makes smart decisions about what to prioritize, and when errors occur, it handles them gracefully to maintain system stability. All of this happens in microseconds, in a continuous cycle, creating the seamless computing experience we take for granted.

5 min 9.54 to 9.59

Oral —>

Task 4:

BASH in Lonux full form and Explanation

BASH (Bourne Again Shell)

BASH is like Linux's command interpreter and default interface. Think of it as the translator between you and your computer - when you type commands, BASH understands them and tells the computer what to do. It's called "Bourne Again" because it's an improved version of the original Bourne Shell (sh).

5 min 10 to 10.05

Oal →

Task 5:

Now that you know Linux is also an Operating System like Windows..

What do you think is the difference between LInux and Windows

Think of Linux as a free, open cookbook where anyone can see and modify recipes (source code), while Windows is like a secret recipe book owned by Microsoft. Linux is more customizable, generally more secure, and free to use. It rarely needs reboots and is less vulnerable to viruses. Windows is more user-friendly for beginners and has better software compatibility, especially for games and business applications.

8 min 10.06 to 10.14

Task 6:

What are the basic components of Linux? Describe each in detail with diagrams.

Linux is built like a layer cake: At the bottom is the Kernel (the core that talks to hardware), then the Shell (the interpreter of your commands), followed by Applications (programs you use). There's also the Bootloader (starts the system), System Services (background tasks), and the Desktop Environment (the visual interface you see).

10.15 to 10.25 10 min

Oral →

Task 7:

Is it legal to edit Kernal ? when do you think we have to in case?

Yes, it's legal to edit the kernel - that's the beauty of open source! You might need to edit it when adding support for new hardware, improving performance for specific tasks, or fixing security issues. However, it's like performing heart surgery - only do it if you really know what you're doing!

10.26 to 10.31 5 min

Task 8:

What is LILO? Explain

LILO is like a starter key for your Linux system - it's one of the original bootloaders that helps start your computer and load Linux. Though mostly replaced by GRUB now, it's historically significant and still used in some systems.

10.32 to 10.36 5 min

how many of you have gone through techadamy Linux plz raise ur hand

19 pax done.. Completely —> Good Job Guys

Task 9:

What is shell? How many shells are there and what are they ? can you explain.

A shell is your command interpreter - like a translator between you and the computer. Main types include:

Bash (most common)

sh (original shell)

csh/tcsh (C shell)

ksh (Korn shell)

zsh (Z shell, very modern and feature-rich

10.37 to 10.47 10 min

Task 10:

What is swap space?

Think of swap space as extra RAM on your hard drive. When your computer runs out of physical memory, it uses this space like an overflow parking lot - slower than RAM but prevents system crashes from memory shortages.

2 min 10.48 to 10.50

Task 11:

What is Mount ? how do you mount and unmount file system in Linux?

Mounting is like plugging in a USB drive - it's the process of making storage devices accessible to your Linux system. You can mount drives using the mount command and unmount them with umount. It's how Linux organizes all storage devices into one unified file system.

10 min 10.51 to 11.02

Plz raise hand once done till here.. So that we can go for oral..

Task 12:

What is chmod command ? how to use it?

chmod (Change Mode) controls file permissions in Linux

# Numeric Method

chmod 755 file.txt # rwxr-xr-x

chmod 644 file.txt # rw-r--r--

# Symbolic Method

chmod u+x file.txt # Add execute permission for user

chmod go-w file.txt # Remove write permission for group and others

Numbers mean:

4 = read (r)

2 = write (w)

1 = execute (x)

5 min

Task 13:

Can you add a new user account? Crate a new user in different ways and paste ss

*# Method 1: useradd*

*sudo* useradd newuser

*sudo* passwd newuser

*# Method 2: adduser (more user-friendly)*

*sudo* adduser newuser

*# Method 3: With home directory*

*sudo* useradd -m -d /home/newuser newuser

5 min

Task 14:

Can you change the password of a user?

How do you do that? Plz share ss

*# For current user*

passwd

*# For other user (needs sudo)*

*sudo* passwd username

*# Force user to change password at next login*

*sudo* passwd -e username

5 min

Task 15:

What is diff between Process and Thread?

Process:

* Like a complete restaurant with its own kitchen, staff, and resources
* Independent program running in memory
* Has its own resources and memory space
* Heavier and more resource-intensive
* Example: Running Chrome browser

Thread:

* Like different chefs working in the same kitchen
* Lightweight units within a process
* Shares resources with parent process
* Faster to create and switch between
* Example: Chrome tabs within the browser

Plz explain

8 min

Task 16:

Doc 14 Linux Grep commands in docs to study folder .. plz work on it..

33 pax done out of 44

If all 16 tasks done.. Plz raise your hands

Task 17:

AWK commands in doc 15 Linux AWK commands.

Task 18:

How to check file access permission in Linux?

Hint use:

Ls -l

ls -l filename

# Example output:

# -rw-r--r-- 1 user group 123 May 31 10:00 filename

Task 19:

What are the default permissions for a new file ?

Plz find out for

Owner → ?

Group → ?

All and others → ?

Owner → rw- (read, write) = 6

Group → r-- (read only) = 4

Others → r-- (read only) = 4

Default is typically: 644

Juz write no ss req

Task 20:

What is the command to change the permisssion to read only for the owner, group and all other users

HInt: chmod 444 filename

chmod 444 filename

# Results in: r--r--r--

# Everyone can read, nobody can write or execute

Task 21:

Can you change the file permissions to match the following:

* owner: Read and Write
* group: Read
* other: no permissions (None)

chmod 640 filename

# Results in: rw-r-----

# 6 (rw-) for owner

# 4 (r--) for group

# 0 (---) for others

Task 22:

What was the command for changing the file permissions to -rw-r-----?

Hint : use chmod 640 filename

chmod 640 filename

# Creates: -rw-r-----

Task 23:

Change chmod.exercises permissions to -rwxr-x--x

Change the file permissions to match the following:

chmod 751 filename

# Owner: rwx (7)

# Group: r-x (5)

# Others: --x (1)

owner: Read, Write and Execute

group: Read and Execute

other: Execute

Task 24:

What was the command for changing the file permissions to -rwxr-x--x

Hint : use chmod 751 filename

chmod 751 filename

# Creates: -rwxr-x--x

\*\*Task 25:\*\*

\*\*Question:\*\* Guys what will this command do? `chown -c master file1.txt`

\*\*Answer:\*\*

This command will change the ownership of `file1.txt` to user `master`. The `-c` flag means "verbose" - it will only display output when a change is actually made. If the file ownership is successfully changed, it will show a message like "changed ownership of 'file1.txt' from [old\_owner] to master". If no change occurs, it displays nothing.

---

\*\*Task 26:\*\*

\*\*Question:\*\* Can you define what is a process?

\*\*Answer:\*\*

A process is a running instance of a program in memory. Key characteristics:

- Has a unique Process ID (PID)

- Contains program code, data, and system resources

- Can be in different states: running, sleeping, stopped, zombie

- Has parent-child relationships

- Managed by the operating system kernel

- Examples: when you run `ls`, `cat`, or any command, each becomes a process

---

\*\*Task 27:\*\*

\*\*Question:\*\* What is command to check foreground process and background process?

\*\*Answer:\*\*

- \*\*Foreground processes:\*\* `jobs -l` or simply `jobs`

- \*\*Background processes:\*\* `jobs` or `ps` commands

- \*\*All processes:\*\* `ps aux` or `ps -ef`

- \*\*Current shell jobs:\*\* `jobs -r` (running) and `jobs -s` (stopped)

- \*\*Move to background:\*\* `command &` or `Ctrl+Z` then `bg`

- \*\*Move to foreground:\*\* `fg %jobnumber`

---

\*\*Task 28:\*\*

\*\*Question:\*\* Can you list all the running processes? Hint use ps

\*\*Answer:\*\*

```bash

ps aux # All processes with detailed info

ps -ef # All processes in full format

ps -A # All processes

ps -e # All processes

ps # Processes in current terminal only

```

Most commonly used: `ps aux` shows all running processes with user, PID, CPU%, memory%, and command details.

---

\*\*Task 29:\*\*

\*\*Question:\*\* What will ps -f command do? plz try n check .. ss required.

\*\*Answer:\*\*

`ps -f` displays processes in full format showing:

- UID (User ID)

- PID (Process ID)

- PPID (Parent Process ID)

- C (CPU utilization)

- STIME (Start time)

- TTY (Terminal)

- TIME (CPU time used)

- CMD (Command with arguments)

Example output:

```

UID PID PPID C STIME TTY TIME CMD

user 1234 1000 0 10:30 pts/0 00:00:01 bash

user 5678 1234 0 10:35 pts/0 00:00:00 ps -f

```

---

\*\*Task 30:\*\*

\*\*Question:\*\* Can you create a variable name with your name in it? Ex: Name = "prasunamba", Id = 10001. And check Echo $Name

\*\*Answer:\*\*

```bash

Name="prasunamba" # No spaces around =

Id=10001

echo $Name # Output: prasunamba

echo $Id # Output: 10001

echo "My name is $Name and ID is $Id" # Output: My name is prasunamba and ID is 10001

```

---

\*\*Task 31:\*\*

\*\*Question:\*\* Can you make the above name variable read only? Ex: Name = "Prasunamba", Readonly Name, Name = "Meher" —>what will this display.. Is it saying read only?

\*\*Answer:\*\*

```bash

Name="Prasunamba"

readonly Name

Name="Meher" # This will give error: bash: Name: readonly variable

echo $Name # Output: Prasunamba (unchanged)

```

Yes, it will display an error message saying the variable is read-only and cannot be modified.

---

\*\*Task 32:\*\*

\*\*Question:\*\* Now will unset or delete the variables. Use: Unset Name, Now check for echo $Name

\*\*Answer:\*\*

```bash

unset Name # Note: cannot unset readonly variables

echo $Name # Output: (nothing/blank)

```

\*\*Reason:\*\* `unset` removes the variable from memory. When you echo an undefined variable, it returns empty/nothing. However, if the variable was made readonly, `unset` will fail with error "bash: unset: Name: cannot unset: readonly variable".

---

\*\*Task 33:\*\*

\*\*Question:\*\* Can u try to add a list of your friends names in an array and try to printout?

\*\*Answer:\*\*

```bash

NAME[0]="Ram"

NAME[1]="Sita"

NAME[2]="Tina"

NAME[3]="Veena"

NAME[4]="Tim"

echo "First Index: ${NAME[0]}" # Output: First Index: Ram

echo "Second Index: ${NAME[1]}" # Output: Second Index: Sita

echo "Third Index: ${NAME[2]}" # Output: Third Index: Tina

echo "Fourth Index: ${NAME[3]}" # Output: Fourth Index: Veena

echo "Fifth Index: ${NAME[4]}" # Output: Fifth Index: Tim

```

---

\*\*Task 34:\*\*

\*\*Question:\*\* Can you print all the list at once in an array.. Try the below cmds and check

\*\*Answer:\*\*

```bash

echo "${NAME[\*]}" # Output: Ram Sita Tina Veena Tim

echo "${NAME[@]}" # Output: Ram Sita Tina Veena Tim

```

Both commands print all array elements, but:

- `${NAME[\*]}` treats all elements as single string

- `${NAME[@]}` treats each element separately (better for loops)

---

\*\*Task 35:\*\*

\*\*Question:\*\* Plz let me know whats the output of the below snippet:

\*\*Answer:\*\*

```bash

a=0

while [ "$a" -lt 10 ] # this is loop1

do

b="$a"

while [ "$b" -ge 0 ] # this is loop2

do

echo -n "$b "

b=`expr $b - 1`

done

echo

a=`expr $a + 1`

done

```

\*\*Output:\*\*

```

0

1 0

2 1 0

3 2 1 0

4 3 2 1 0

5 4 3 2 1 0

6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

8 7 6 5 4 3 2 1 0

9 8 7 6 5 4 3 2 1 0

```

\*\*Explanation:\*\*

- Outer loop runs from a=0 to a=9

- Inner loop counts down from current value of 'a' to 0

- Creates a countdown pattern for each iteration

Add ons:

—----------------------------------------------------------------------------------------------------------

Stopping Processes

Ending a process can be done in several different ways. Often, from a console-based command, sending a CTRL &plus; C keystroke (the default interrupt character) will exit the command. This works when the process is running in the foreground mode.

If a process is running in the background, you should get its Job ID using the **ps** command. After that, you can use the **kill** command to kill the process as follows −

$ps -f

UID      PID  PPID C STIME    TTY   TIME CMD

amrood   6738 3662 0 10:23:03 pts/6 0:00 first\_one

amrood   6739 3662 0 10:22:54 pts/6 0:00 second\_one

amrood   3662 3657 0 08:10:53 pts/6 0:00 -ksh

amrood   6892 3662 4 10:51:50 pts/6 0:00 ps -f

$kill 6738

Terminated

Here, the **kill** command terminates the **first\_one** process. If a process ignores a regular kill command, you can use **kill -9** followed by the process ID as follows −

$kill -9 6738

Terminated

Parent and Child Processes

Each unix process has two ID numbers assigned to it: The Process ID (pid) and the Parent process ID (ppid). Each user process in the system has a parent process.

Most of the commands that you run have the shell as their parent. Check the **ps -f** example where this command listed both the process ID and the parent process ID.

Zombie and Orphan Processes

Normally, when a child process is killed, the parent process is updated via a **SIGCHLD** signal. Then the parent can do some other task or restart a new child as needed. However, sometimes the parent process is killed before its child is killed. In this case, the "parent of all processes," the **init** process, becomes the new PPID (parent process ID). In some cases, these processes are called orphan processes.

When a process is killed, a **ps** listing may still show the process with a **Z** state. This is a zombie or defunct process. The process is dead and not being used. These processes are different from the orphan processes. They have completed execution but still find an entry in the process table.

Daemon Processes

Daemons are system-related background processes that often run with the permissions of root and services requests from other processes.

A daemon has no controlling terminal. It cannot open **/dev/tty**. If you do a **"ps -ef"** and look at the **tty** field, all daemons will have a **?** for the **tty**.

To be precise, a daemon is a process that runs in the background, usually waiting for something to happen that it is capable of working with. For example, a printer daemon waiting for print commands.

If you have a program that calls for lengthy processing, then its worth to make it a daemon and run it in the background.

The top Command

The **top** command is a very useful tool for quickly showing processes sorted by various criteria.

It is an interactive diagnostic tool that updates frequently and shows information about physical and virtual memory, CPU usage, load averages, and your busy processes.

Here is the simple syntax to run top command and to see the statistics of CPU utilization by different processes −

$top

Job ID Versus Process ID

Background and suspended processes are usually manipulated via **job number (job ID)**. This number is different from the process ID and is used because it is shorter.

In addition, a job can consist of multiple processes running in a series or at the same time, in parallel. Using the job ID is easier than tracking individual processes.

—-------------------------------------------------------------------------------

—-------------------------------

Mounting

Fstab

Bash

Man

Tac

Find

Export

Printenv

Zip unzip

Sed

Uniq

Diff

Grep

Cut

Awk

Sort

Split

Tr