|  |
| --- |
| **DAY – 06** |
| **Task 1:** |
| RegEX Symbols in linux |
| List them down with description. |
| |  |  |  | | --- | --- | --- | | **Character** | **Description** | **Example** | | Basic syntax | | | | non-special character | Matches the specified characters literally (providing they are not any of the regex special characters). | abc matches abc, ABC, aBC etc | | (...) | Groups a set of matching characters together. Multiple groups can be specified. Each group can then be referenced in order (from left to right) using the characters \1, \2, etc. as part of a replace string. |  | | \ | Escapes a regular expression special character: {}[]()^ $.|\*+?\  Also used to reference a group in a replace expression. | ab\+ matches ab+ | | | | Matches either one expression or an alternative expression. | .\*@example\.(net|com) matches against any URI for the domain example.com or the domain example.net | | Wildcard and character matching | | | | . | Matches any single character. | a.c matches aac, abc, azc, a2c, a$c etc. | | \d | Matches a decimal digit character (i.e. 0-9). | a\d matches a1, a2, a3 etc. but not aa, ab etc. | | \D | Matches a non-digit character. | a\D matches ab but not a1, a2, a3 etc. | | \s | Matches any whitespace character (space, tab, newline). | ab\sd matches ab d but not abcd, abxd etc. | | \S | Matches any non-whitespace character. | ab\Sd matches abcd, abxd etc. but not ab d | | \w | Shorthand for [a-zA-Z0-9\_].  Matches any alphabetical or digit character, or underscore. | \w+ matches bob and bob\_jones but not bob.jones.  [\w.-]+ matches bob, bob.jones, bob\_jones, and bob-jones. | | [...] | Matches the characters specified in the brackets.  You can specify a range of characters by specifying the first and last characters in the range, separated by a hyphen.  You cannot use other special characters within the [] — they will be taken literally. | 9[aeiou] matches 9a, 9e, 9i etc. and 9A, 9E, 9I etc, but not 9b, 9c, 9B, 9C etc  9[a-z] matches 9a, 9b, 9z etc. and 9A, 9B, 9Z etc. but not strings such as 91, 99 or 9(  [0-9#\*] matches any single E.164 character (digits 0-9, hash key or asterisk) | | [^...] | Matches anything except the set of specified characters. | [^a-z] matches any non-alphabetical character  [^0-9#\*] matches anything other than an E.164 character | | Repetition factors | | | | \* | Matches 0 or more repetitions of the previous character or expression. | ab\*c matches ac, abc, abbc, but not ab or abd | | + | Matches 1 or more repetitions of the previous character or expression. | ab+c matches abc and abbc, but not ab, ac or abd  [a-z.]+ matches any string containing the letters a to z, A to Z, or dots (periods) | | ? | Matches 0 or 1 repetitions of the previous character or expression. | ab?c matches ac and abc, but not ab, abbc or abd | | .\* | Matches against any sequence of characters. | a.\* matches everything beginning with a | | {n} | Matches n repetitions of the previous character or expression. | ab{2}c matches abbc, but not abc or abbbc  a\d{3} matches a123and a789, but not a12 or 456 | | {n,m} | Matches n to m repetitions of the previous character or expression. | ab{2,4}c matches abbc, abbbc and abbbbc, but not abc or abbbbbc | | {n,} | Matches at least n repetitions of the previous character or expression. | ab{2,}c matches abbc, abbbc, abbbbc etc, but not abc | | Position matching | | | | ^ | Matches the beginning of a line. | ^meet\.(.\*) matches any string that starts with meet. and places the rest of the string into a group (which could be used in a replace string). | | $ | Matches the end of the line. | .\*\.com$ matches any string that ends in .com | | (?!...) | Negative lookahead. Defines a subexpression that must not be present immediately after the current match position, for example regex1(?!regex2) where a match is found if regex1 matches and regex2 does not match. | (?!.\*@example\.com$).\* matches any string that does not end with @example.com  (?!meet).\* matches any string that does not start with meet  meet(?!\.).\* matches any string that starts with meet providing it is not followed by a period | | (?<!...) | Negative lookbehind. Defines a subexpression that must not be present immediately before the current match position, for example (?<!regex1)regex2 where a match is found if regex1 does not match and regex2 matches. | .\*(?<!@)example\.com.\* matches any string containing example.com providing it is not preceded by @ |   **Task 2:**  What are the imp features of Linux os ?   * **Open Source & Free:** No cost, modifiable code, community-driven. * **Multitasking & Multi-User:** Efficiently runs multiple programs and supports many users simultaneously. * **High Security:** Robust permissions, less prone to malware, quick updates. * **Stable & Reliable:** Runs for long periods without reboots, very dependable. * **Flexible & Customizable:** Adaptable to various uses, wide choice of desktop environments. * **Powerful CLI:** Command-line interface for efficient scripting and control. * **Efficient Package Management:** Easy software installation, updates, and removal. *  **Broad Hardware Support:** Runs on a wide range of devices, including older hardware. |

**Task 3:**

What is Kernal and can you explain its functions.

A kernel is the core component of an operating system. It's the central module that essentially acts as a bridge between the computer's hardware and the software applications running on it. Think of it as the "brain" or the "heart" of the OS.

Its key functions: The kernel is the OS's core, bridging hardware and software. It manages processes (scheduling, creation/termination, communication), memory (allocation, virtual memory, protection), and devices (drivers, I/O, resource allocation). It also handles system calls for application services, responds to interrupts, and ensures security through mode separation and access control. Essentially, the kernel makes the computer functional and usable.

1. Process Management

* Scheduling: The kernel decides which processes (running programs) get access to the CPU and for how long. It uses various scheduling algorithms (e.g., round-robin, priority-based) to ensure fair and efficient use of the processor, allowing multiple programs to appear to run simultaneously.4
* Process Creation/Termination: It handles the creation of new processes when you launch an application and the termination of processes when they complete or are closed.
* Process Synchronization and Communication: It provides mechanisms for processes to communicate with each other (Inter-Process Communication - IPC) and to synchronize their activities to avoid conflicts when accessing shared resources.5

2. Memory Management

* Memory Allocation: The kernel allocates memory to different processes as they need it, ensuring that each process has enough space to run without interfering with others.6
* Virtual Memory: It implements virtual memory, which allows programs to use more memory than is physically available.7 This is done by swapping data between RAM and disk storage.
* Memory Protection: It protects the memory space of one process from being accessed or corrupted by another, ensuring system stability and security.8

3. Device Management (I/O Management)

* Device Drivers: The kernel interacts with hardware devices (like keyboards, mice, printers, hard drives, network cards) through device drivers.9 It loads and manages these drivers.10
* I/O Operations: It manages all input and output operations, coordinating the flow of data between the CPU and peripheral devices. This includes handling interrupts generated by devices when they need attention.
* Resource Allocation: It allocates and deallocates hardware resources to different processes.11

4. System Calls

* Interface for Applications: The kernel provides a set of system calls (APIs) that user-level applications can use to request services from the OS.12 For example, an application might use a system call to read from a file, write to the screen, or create a new process. The kernel then executes these privileged operations on behalf of the application.

5. Interrupt Handling

* Responding to Events: The kernel is responsible for handling interrupts, which are signals from hardware or software indicating that an event has occurred that requires immediate attention (e.g., a key press, a disk read completion, a division-by-zero error).The kernel pauses its current activity, handles the interrupt, and then resumes.

6. Security and Protection

* User/Kernel Mode Separation: The kernel operates in a privileged "kernel mode" (or supervisor mode), while user applications run in an unprivileged "user mode."This separation protects the kernel from malicious or faulty user programs.
* Access Control: It enforces security policies, controlling access to system resources and ensuring that only authorized users and processes can perform certain operations.

In essence, the kernel is what makes a computer usable. It abstracts the complexities of the hardware, provides a consistent interface for applications, manages system resources, and ensures the stable and secure operation of the entire system. Without a kernel, your computer would just be a collection of inert hardware components.

**Task 4:**

What is BASH? Full form with explanation.

**BASH** stands for **Bourne-Again SHell**.

It's a widely used command-line interpreter and scripting language for Unix-like operating systems, including Linux and macOS. Essentially, BASH is the program that allows you to type commands into a terminal (like the black-screen-with-text window) and have your computer execute them.

BASH is like a universal translator between you and your computer's operating system. You speak to it in commands or scripts, and it translates those instructions into actions for the computer to perform. It's a fundamental tool for system administrators, developers, and anyone who wants to have powerful control over their Unix-like system.

**Bourne-Again:** The "Bourne-Again" part is a pun on the "Bourne Shell", which was an earlier and influential Unix shell developed by Stephen Bourne. BASH was created as a free and improved replacement for the Bourne Shell, incorporating features from other popular shells like the C shell (csh) and the Korn shell

**Task 5**

What is the difference between window and Linux.

Windows is a commercial, proprietary OS known for user-friendliness and broad software compatibility, it lacks built-in security tools while Linux is an open-source OS favored for its flexibility, security, and stability.

Linux is often preferred for its cost-effectiveness, customized experience, and enhanced security. It is preferred by hackers and security experts due to its open-source nature and control.

**Task 6:**

Define the basic components of Linux

The core components of a Linux operating system include the kernel, system libraries, and system utilities.

1. Kernel: This is the absolute heart of Linux. It's the low-level program that directly interacts with the computer's hardware. It manages the CPU, memory, and devices, and acts as the bridge between hardware and software.
2. Shell: This is the command-line interpreter (like Bash) that allows users to interact with the kernel. It takes commands typed by the user or from scripts, interprets them, and passes them to the kernel for execution.
3. System Utilities (User-Space Tools): These are a collection of programs and tools that perform various system-level tasks. Examples include ls (list files), cp (copy files), mv (move files), grep (search text), cat (display file content), ping (network test), etc. They are essential for managing the system and often interact with the kernel via system calls.
4. System Libraries: These are collections of pre-written code that applications and utilities can use to perform common tasks, such as accessing the kernel's features. They provide a standardized way for programs to interact with the underlying system without having to write complex low-level code from scratch. The GNU C Library (glibc) is a prominent example.

**Task 7:**

Is it legal to edit Kernel?

Yes, it is absolutely legal to edit the Linux kernel.

The Linux kernel is licensed under the GNU General Public License (GPLv2), which explicitly grants you the freedom to:

* Study how it works.
* Change it to suit your needs.
* Redistribute your modified versions.

**Task 8:**

Can you explain LILO

LILO, or the Linux Loader, is an older bootloader used in Linux systems to load the operating system into memory during the boot process. It's responsible for loading the kernel and initial RAM disk (initrd). LILO has been largely replaced by more advanced bootloaders like GRUB due to limitations in support for modern file systems and configuration flexibility.

Unlike modern boot loaders, LILO does not have a user-friendly interface or advanced features but offers a straightforward and dependable way to load the operating system kernel.

**Task 9:**

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

A **shell** is a program that provides an interface between the user and the operating system. It allows users to interact with the computer by entering commands, which the shell interprets and executes.

In simple terms:

* It's like a translator between you and the operating system's kernel.
* It can be command-line based (CLI) or graphical (GUI).

The most common types of shells:

1. **Bourne Shell (sh):** The original, basic Unix shell; known for simplicity and portability.
2. **Bourne-Again SHell (bash):** The most common default shell; enhanced sh with features like tab completion and history.
3. **C Shell (csh) & TENEX C Shell (tcsh):** Use C-like syntax; tcsh is an enhanced version for interactivity.
4. **Korn Shell (ksh):** Combines sh and csh features; robust for scripting and efficiency.
5. **Z Shell (zsh):** Highly customizable and powerful, building on bash with advanced features (e.g., smart completion, themes).
6. **Friendly Interactive SHell (fish):** User-friendly, focused on interactive experience with auto-suggestions and syntax highlighting.
7. **Debian Almquist Shell (dash):** A very lightweight and fast shell, often used for system scripts due to its efficiency.

**Task 10:** What is Swap space ?

Swap space is a portion of a hard drive or SSD that is used as virtual memory when a system’s physical RAM (Random Access Memory) is full. When the system runs out of RAM, it temporarily moves less-used data from RAM to the swap space, freeing up RAM for other tasks. This process is called "paging" or "swapping." It helps the system stay stable and multitask better. It’s slower than RAM but essential for heavy workloads or low-memory systems.

**Task 11:**

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

In Linux, mounting is the process of attaching a filesystem (like a hard drive, USB stick, CD, or network storage) to a specific directory in the system's directory tree so that its contents can be accessed. When you mount a filesystem you assign it to a directory (called a mount point) like /mnt, /media/usb, or /home/user/drive. **Unmount:** The act of **detaching** a mounted file system from its mount point. This makes the device's files no longer accessible through that directory and safely prepares the device for removal or other operations. It's crucial to unmount before physically removing a device to prevent data corruption.

**How to Mount:**

1. **Create Mount Point:** sudo mkdir /mnt/mydevice (a directory where the device will be attached).
2. **Mount Device:** sudo mount /dev/sdb1 /mnt/mydevice (replace /dev/sdb1 with your device path).

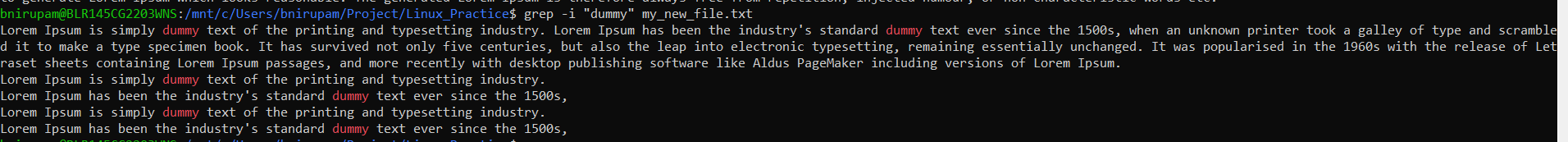
|  |
| --- |
| * Mount a device |

|  |
| --- |
| sudo mount /dev/sdX1 /mnt/point |
| **How to Unmount:**  sudo umount /mnt/mydevice (unmount using the mount point).  sudo umount /dev/sdb1 (unmount using the device path, less common) |
| **Task 12:**  What is chmod command ? how to use it?  The chmod (short for **change mode**) command is used to **change the permissions** of files and directories in Linux/Unix.  **Permission Levels**   * **r** = read * **w** = write * **x** = execute   Set for:   * **u** = user (owner) * **g** = group * **o** = others   Use:  chmod [permissions] [file]  **Numeric Mode:**  chmod 755 file  # 7 = rwx (user), 5 = r-x (group), 5 = r-x (others)  **Symbolic Mode:**  chmod u+x file (Add execute for user)  chmod go-w file (Remove write for group/others)  **Check Permissions**  ls -l file  **Task 13:**  Can you add a new user account? Crate a new user in different ways and paste ss  Yes, you can add a new user account in several ways, depending on the operating system and your needs  **Using adduser**  adduser is a high-level command that automates many steps. It creates the user, creates a home directory, copies default configuration files, and sets up a new group for the user by default. |
| **Using useradd**  useradd is a lower-level command that provides more fine-grained control but requires more manual setup. If you just use useradd without options, it typically *won't* create a home directory or copy default files, which often leads to a non-functional user unless you add flags.  sudo useradd newuser2 |
| **Task 14:**  Can you change the password of a user?  How do you do that? Plz share ss  Yes, you can change a user's password in Linux using the passwd command.    **Task 15:**  What is diff between Process and Thread?   * **Process:** An independent running program with its own dedicated, isolated memory space and resources. It's "heavyweight" and failures typically don't affect other processes. * **Thread:** A lightweight unit of execution *within* a process. Threads share the parent process's memory space and resources, making them faster to create and communicate, but a failure can impact other threads in the same process. |

**Task16:**

**Practical Example of grep Command in Linux**

1. **Case insensitive search (grep -i)**



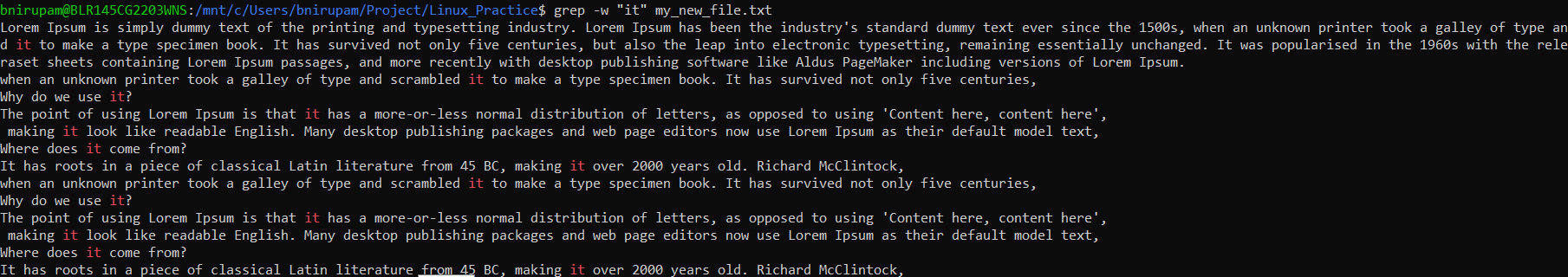
**2. Displaying the Count of Number of Matches Using grep (grep -c)**



**3. Display the File Names that Matches the Pattern Using grep (grep -l)**

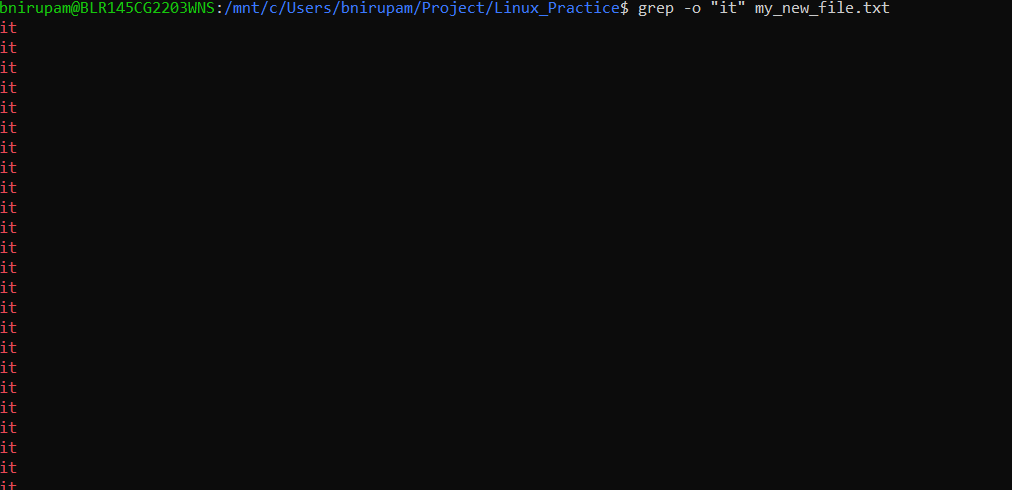


**4. Checking for the Whole Words in a File Using grep (grep -w)**



**5. Displaying only the matched pattern Using grep**

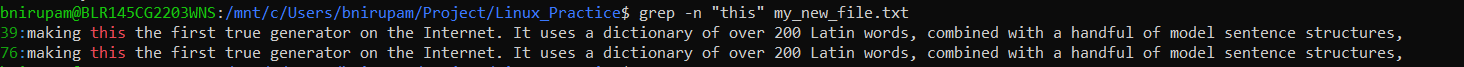
By default, grep displays the entire line which has the matched string. We can make the grep to display only the matched string by using the -o option.



**6. Show Line Number While Displaying the Output Using grep -n**

To show the line number of file with the line matched.

grep -n "unix" hellofile.txt

**Output:** 

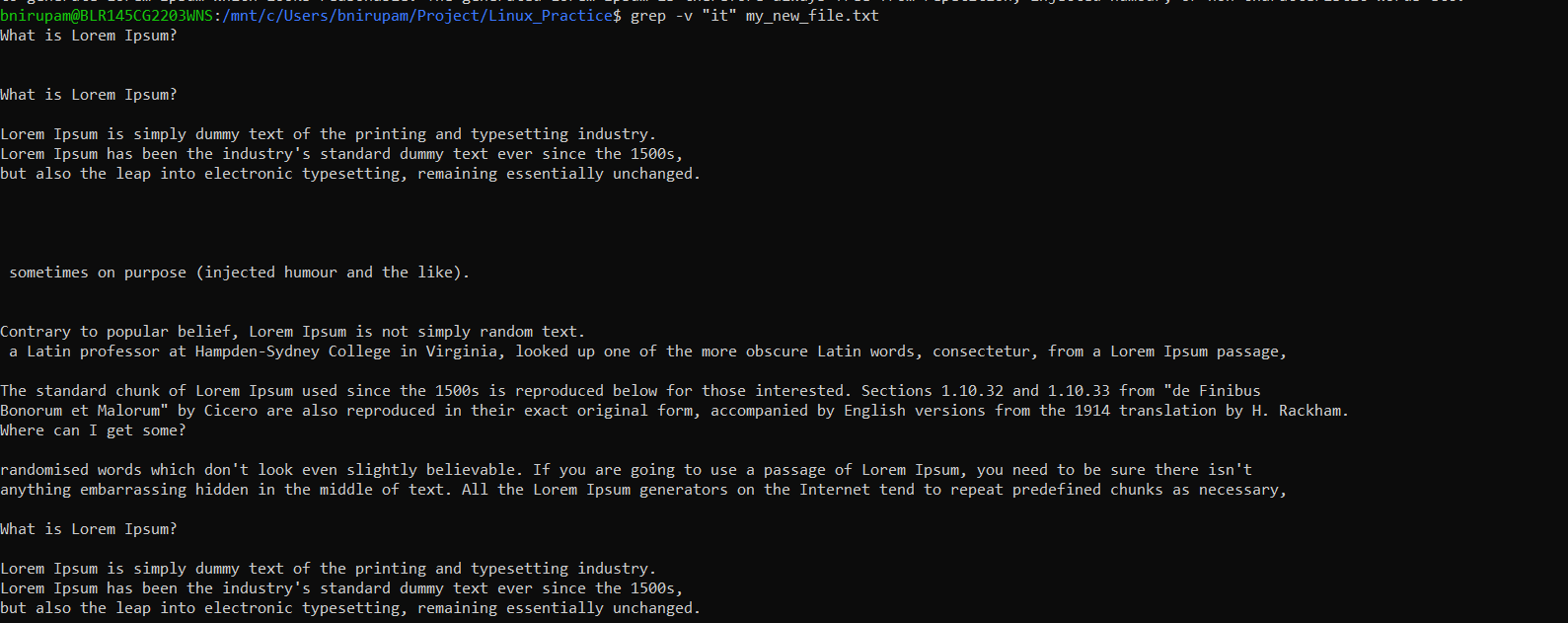
Show line number while displaying the output

**7. Inverting the Pattern Match Using grep**

You can display the lines that are not matched with the specified search string pattern using the -v option.

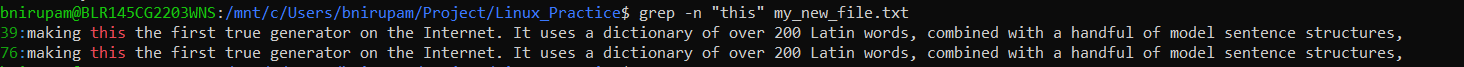
grep -v "unix" hellofile.txt

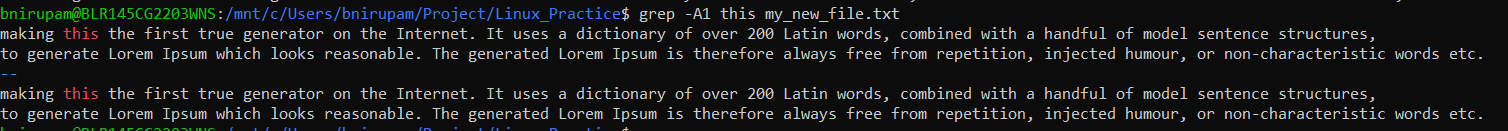
**Output:**



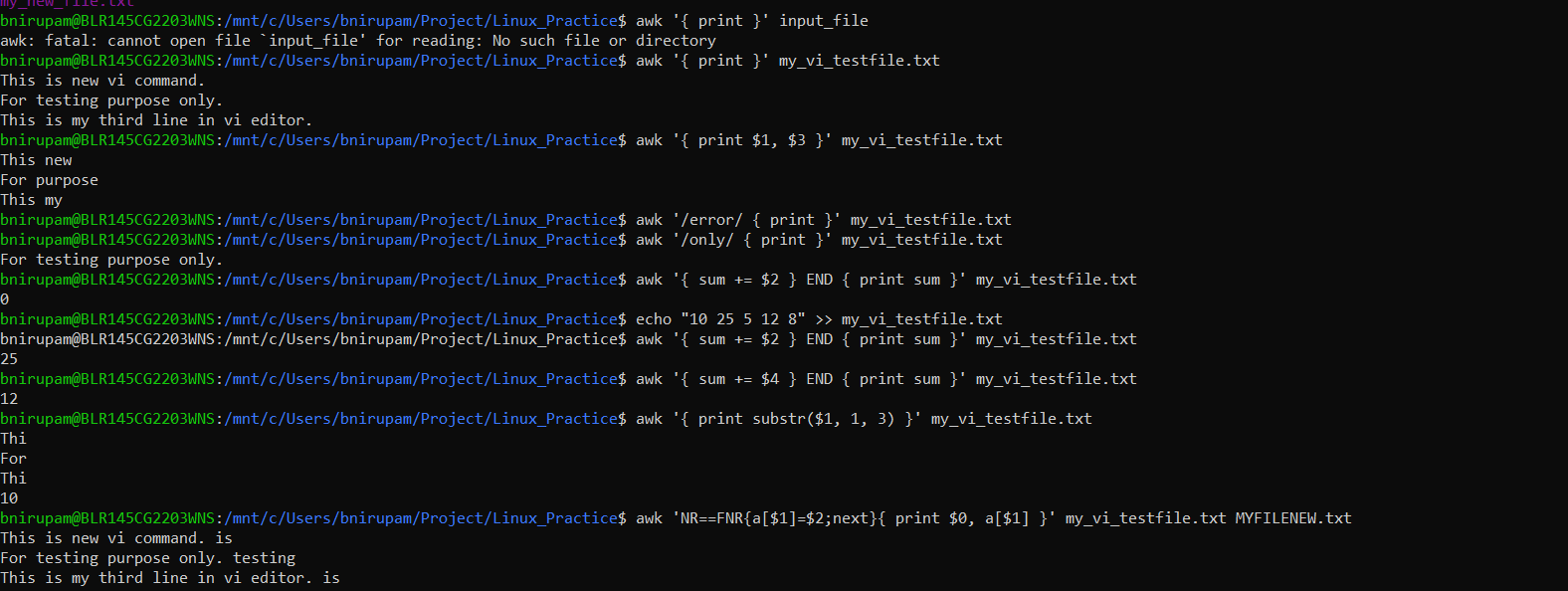


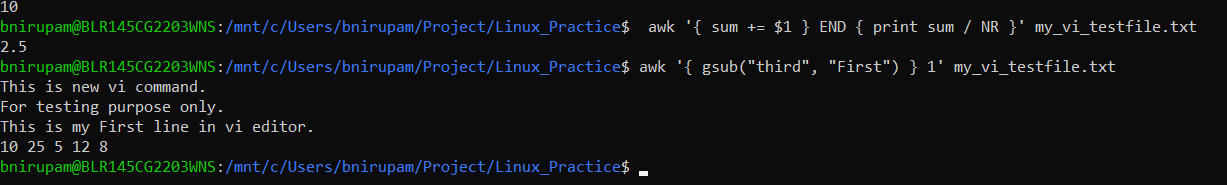


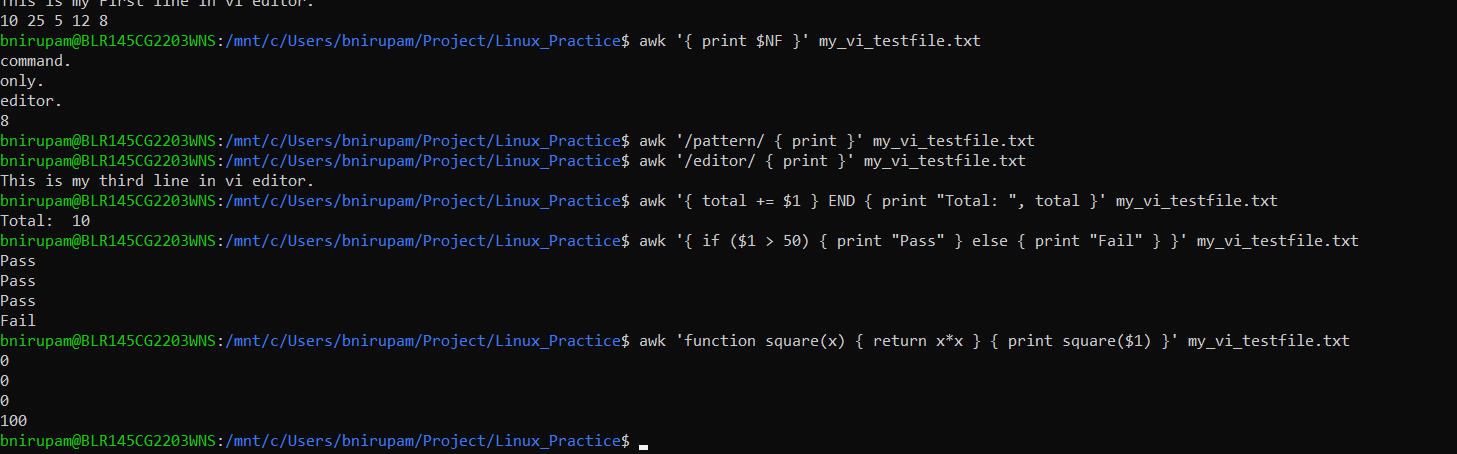




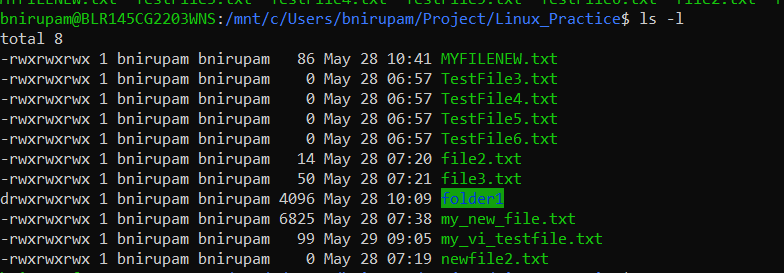
**Task 17 AWK commands in doc 15 Linux AWK commands..**







**Task 18:**

How to check file access permission in Linux? 

**Task 19:**

What are the default permissions for a new file ?

Plz find out for

Owner → Read & Write

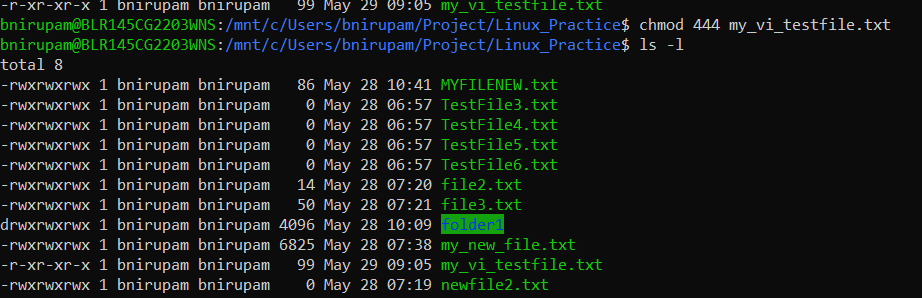
Group → Read

All and others → Read

**Task 20:**

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

HInt: chmod 444



It’s in read & execute as I’m using windows rn.

Chmod 777 for rwx

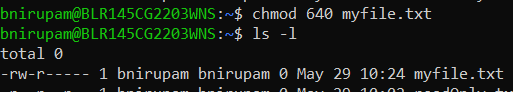
**Task 21:**

Can you change the file permissions to match the following:

* owner: Read and Write
* group: Read

other: no permissions (None)

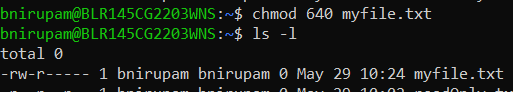
Yes! To change the file permissions so that: we can use chmod 640



Task 22:

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

Hint : use chmod 640 filename



Task 23:

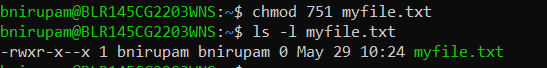
Change chmod.exercises permissions to -rwxr-x--x

Change the file permissions to match the following:

owner: Read, Write and Execute

group: Read and Execute

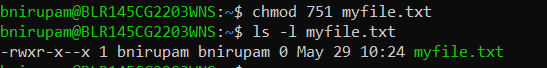
other: Execute



**Task 24:**

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

Hint : use chmod 751 filename



**Task 25:**

Guys what will this command do? chown -c master file1.txt

It is used to change the ownership of file1.txt to the user master. 

**Task 26**  
**Can you define what is  a process?**

A process is an instance of a running program in Linux. Each process is an independent, isolated environment with its own dedicated memory, resources (like open files), and a unique Process ID (PID).

**What is command to check foreground process and background process**

Check foreground & background jobs (in current shell): jobs

**Can you list all the running processes?**

To see all processes across the entire system, use ps with options:

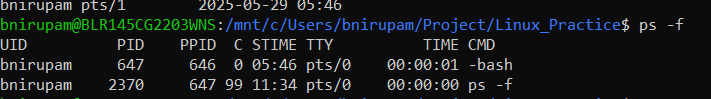
**ps aux**: Shows all user-oriented processes, including those without a controlling terminal.

**ps -ef**: Displays all processes in a full-format listing.

Both commands provide details like USER, PID, %CPU, %MEM, TTY, START time, and the COMMAND that started the process.

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

The ps -f command gives a full-format listing of processes specifically associated with your current terminal (tty). It includes key details like UID, PID, PPID (Parent Process ID), STIME (Start time), and the full CMD (command with arguments).



**Task 30:**

Can you createa  a variable name with your name in it

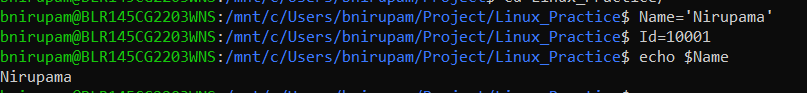
Ex:

Name =  “prasunamba”

Id  = 10001

And check

Echo $Name

Chek the output

**Task 31:**

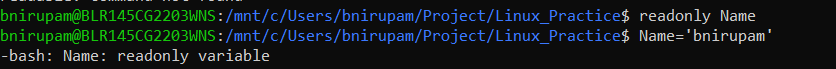
Can you make the above name variable read only..

Ex:

Name = “Prasunamba”

Readonly Name

Name = “Meher” —>what will this display.. Is it saying read only?? Pl check

 **Task 32:**

Now will unset or delete the variables

Use the below command and check

Unset Name 

Now check for

 echo $Name   —> this should not print anything.. Plz try also specify the reason

It will give output as it’s not unset and it’s in read only. You cannot unset a variable that has been marked as readonly. You will get an error.

**Task 33:**

Can u try to add a list of your friends names in an array and try to printout

Ex:

NAME[0]="Ram"

NAME[1]="Sita"

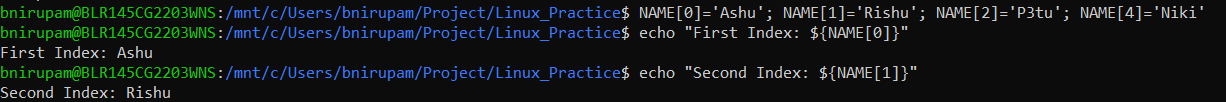
NAME[2]="Tina"

NAME[3]="Veena"

NAME[4]="Tim"

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

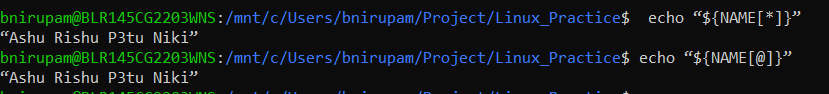
echo "Second Index: ${NAME[1]}"



**Task 34:**

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

Echo “${array\_name[\*]}”

Echo “${array\_name[@]}”

**Task 35:**

Plz let me know whats the output of the below snippet:

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`

**ANS:**

