LINUX FUNDAMENTALS PART 2

RegEx (Regular Expressions): Regular expressions are a powerful tool for pattern matching and text manipulation. They provide a concise and flexible way to search, match, and manipulate text data. Programming languages like Bash, Perl, Python, and many others support regular expressions

In Linux, regular expressions are supported by various command-line utilities and programming languages, such as:

sed: The stream editor, which uses RegEx for find-and-replace operations on text.

grep: The global regular expression print utility, which searches for and prints lines that match a given RegEx pattern.

awk: A programming language that uses RegEx for advanced text processing and data manipulation.

vim: The popular text editor, which uses RegEx for search and replace operations.

RegEx can be used for a wide range of text-related tasks, such as:

Searching and replacing text

Validating user input (e.g., email addresses, phone numbers)

Extracting specific data from text (e.g., URLs, dates, IP addresses)

Splitting and parsing text data

Automating text-processing tasks

Regular expressions can be complex and take some time to master, but they are an invaluable tool for working with text data in Linux and many other programming environments.

Task 1: RegEx Symbols in Linux. List them down with description

Solution: Here are the common regular expression (RegEx) symbols used in Linux, along with brief descriptions:

1. `. `: Matches any single character, except a newline character.

2. `^`: Matches the beginning of a line or string.

3. `$`: Matches the end of a line or string.

4. `\*`: Matches zero or more occurrences of the preceding character or group.

5. `+`: Matches one or more occurrences of the preceding character or group.

6. `? `: Matches zero or one occurrence of the preceding character or group.

7. ` [] `: Matches any one of the characters within the brackets.

8. ` [^] `: Matches any one character that is not within the brackets.

9. ` () `: Groups multiple characters together for use with other operators, such as `\*`, `+`, or `|`.

10. `|`: Matches either the expression before or after the pipe.

11. `\`: Escapes special characters, allowing you to match literal characters like `\*` or `? `.

12. `\b`: Matches a word boundary (the start or end of a word).

13. `\d`: Matches any digit character (0-9).

14. `\s`: Matches any whitespace character (space, tab, newline, etc.).

15. `\w`: Matches any word character (a-z, A-Z, 0-9, \_).

These RegEx symbols provide powerful pattern-matching capabilities that are widely used in various Linux utilities and programming languages for tasks such as text processing, file searching, and data validation.

Task 2: What are the imp features of Linux?

Solution: Linux, as an open-source operating system, has several unique features that distinguish it from other operating systems like Windows or macOS. Here are some of the most important and unique features of Linux, briefly:

1. Open-Source & Free: Linux is open-source and available for free, allowing customization and community development.

2. Flexibility: Linux offers a wide variety of desktop environments, window managers, and distributions for customization.

3. Command-Line Interface: Linux provides a powerful CLI for advanced tasks, automation, and system administration.

4. Security & Stability: Linux has robust security features and is generally more stable than other OSes.

5. Multi-Tasking & Multi-User: Linux supports multiple users and tasks simultaneously, suitable for servers and enterprises.

6. Hardware Compatibility: Linux runs on a variety of hardware architectures and supports older hardware.

7. Package Management: Linux distributions use package managers for easy software installation, upgrade, and removal.

8. Performance: Linux is known for efficient resource utilization and high performance, even on older hardware.

9. Community & Ecosystem: Linux benefits from a large, active community and a wide range of software and tools.

10. Embedded Systems & IoT: Linux is widely used in embedded systems and IoT devices due to its small footprint, flexibility, and security.

Task 3: What is Kernel and can you explain its functions?

Solution: At the heart of any operating system, including Linux, is the kernel - the core component that acts as the intermediary between the software and the computer's hardware.

The kernel is responsible for managing system resources, such as memory and CPU time, as well as facilitating communication between applications and the underlying hardware. It ensures that programs can access the necessary hardware components, like storage devices and network interfaces, in a secure and organized manner.

Additionally, the kernel handles vital tasks like process scheduling, memory management, and device management. It's the gatekeeper that enables the rest of the operating system and user-level applications to function smoothly and efficiently.

While the inner workings of the kernel can be complex, its primary role is to provide a stable, abstracted interface that allows the rest of the system to operate without needing to understand the intricate details of the computer's hardware. It's the foundation upon which the entire operating system is built.

Task 4: What is BASH? Full form with explanation.

Solution: BASH, or the Bourne-Again Shell, is the default command-line interface in most Linux systems. It's essentially the gateway through which users interact with the operating system, allowing them to execute commands, run programs, and even write small scripts to automate repetitive tasks.

Essentially, BASH builds upon the original Bourne Shell, providing a more robust and feature-rich environment for users to work within. Its scripting capabilities make it a powerful tool for system administrators and advanced users who need to streamline their workflows and automate various processes.

While the technical details of BASH can get quite complex, at its core, it's a straightforward and essential component of the Linux ecosystem, serving as the primary means of direct interaction between the user and the underlying operating system.

Task 5: What is the difference between window and Linux?

Solution: The key difference between Windows and Linux is the level of user control and customization. Windows is designed for ease of use, while Linux offers more flexibility and freedom for those who are willing to invest time in learning the system.

Windows:

Windows is the most popular operating system, developed and maintained by the Microsoft corporation. It's the default choice for many personal computers and is known for its user-friendly graphical interface. Windows provides a familiar and intuitive experience, making it easy for most people to navigate and use their computers. However, Windows is a proprietary system, meaning users have limited control and customization options compared to other operating systems.

Linux:

Linux, on the other hand, is an open-source operating system, meaning its underlying code is freely available for anyone to view, modify, and distribute. This makes Linux highly customizable and flexible, allowing users to tailor the system to their specific needs. Linux is often preferred by tech-savvy individuals, developers, and those who value security and privacy, as it generally offers more control and fewer restrictions than Windows. While the learning curve may be steeper for beginners, Linux provides a powerful and versatile platform for a wide range of computing tasks.

Task 6: Define the basic components of Linux

Solution: At the core of the Linux operating system, you have the kernel - the fundamental software that manages all the essential functions of the computer. It's the backbone that allows the rest of the system to operate smoothly.

Sitting on top of the kernel are the various user-level applications and utilities that you interact with directly. This includes the command line interface, known as the shell, which is your gateway to giving instructions to the computer.

Then there are the desktop environments, like GNOME or KDE, which provide the graphical user interface (GUI) that many Linux users are familiar with. These visual layers sit on top of the underlying system, making it easier for everyday users to navigate and use their computers.

Finally, you have the vast collection of software packages and tools that users can install to expand the functionality of their Linux system. From web browsers to office suites, the flexibility of the Linux ecosystem allows people to customize their experience to their specific needs.

While the technical details can get complex, at its core, Linux is built on these fundamental components - the kernel, the shell, the desktop, and the software ecosystem. Together, they form a powerful and adaptable operating system.

Task 7: Is it legal to edit Kernel?

Solution: Editing the Linux kernel is perfectly legal, thanks to the open-source nature of the operating system. The GNU General Public License (GPL) allows users to access, modify, and distribute the kernel code freely, as long as they follow a few basic guidelines.

This openness is a core principle of Linux, giving technically-inclined users the freedom to customize and optimize the system's core components. However, making changes to the complex kernel requires a deep understanding to avoid potential issues.

So, in short, the ability to legally edit the Linux kernel is a unique feature that empowers users, but should be approached with caution by those without the necessary expertise.

Task 8: Can you explain LILO?

Solution: LILO is a boot loader - the first software that runs when you turn on a Linux computer. Its job is to load the Linux kernel and start the operating system.

LILO provided a text-based menu where users could choose which OS to boot into, like Linux or Windows. While an early and important tool, LILO has largely been replaced by newer boot loaders like GRUB in modern Linux systems.

But LILO remains an important part of Linux history, as one of the pioneering utilities that helped make the OS more accessible in its early days.

Task 9: What is shell? How many shells are there and what are they? can you explain.

Solution: The shell in Linux is the command-line interface that allows users to interact with the operating system. It's the gateway through which you can type commands, run programs, and automate tasks.

There are several different shell options available in the Linux world, with the most common ones being Bash (Bourne-Again Shell), Zsh, Fish, and Tcsh. Each shell has its own unique features and capabilities, catering to the preferences and needs of different users.

Bash is by far the most widely used shell, as it's the default option in most Linux distributions. It provides a powerful and flexible environment for both interactive use and scripting. Other shells like Zsh offer enhanced features and user-friendliness for specific use cases.

So, in summary, the shell is a critical component of the Linux experience, giving users direct control over their system. And with multiple shell options to choose from, Linux users can find the one that best suits their workflow and preferences.

Task 10: What is Swap space?

Solution: Swap space in Linux is a designated area on your computer's hard drive used as an overflow for the system's memory (RAM).

When your computer's available RAM is fully utilized, the operating system can temporarily move some data to the swap space, freeing up RAM for more immediate needs. This helps prevent your system from crashing or freezing when running multiple programs.

Think of swap space as a backup storage area that your computer can use when the main memory gets too full. It's an essential feature that enhances the overall performance and stability of a Linux system, especially on machines with limited RAM.

Task 11: What is Mount? how do you mount and unmount file system in Linux?

Solution: Mounting in Linux refers to the process of making a file system accessible to the operating system. It's how you connect a storage device or network share to your Linux system's directory structure.

To mount a file system, you typically use the 'mount' command and specify the device or network location, along with the mount point (the directory where you want to access the content).

Conversely, unmounting is the process of detaching a file system from your Linux system. You use the 'unmount' command to safely disconnect a mounted device or network share when you're done with it.

Mounting and unmounting are essential Linux operations that allow you to access and manage various storage resources on your system, whether they're local drives, USB devices, or network-attached storage.

TASK 12: What is chmod command? how to use it?

Solution: The chmod command is used to change the access permissions of files and directories in a Linux system.

To use chmod, you specify the desired permissions, followed by the file or directory you want to modify. For example, 'chmod 755 myfile.txt' grants the owner full access while allowing others to read and execute.

chmod is a powerful tool for managing the security and accessibility of your Linux system's resources by setting appropriate file and folder permissions.

Task 13: Can you add a new user account? How to Create a new user in different ways?

Solution: Regardless of the method, adding a new user in Linux is a straightforward process that gives someone the ability to log in and access the system with their own account and permissions.

1. Using the 'useradd' command:

- The basic syntax is 'useradd username'

- This creates a new user account with the specified username.

2. Using the 'adduser' command:

- The 'adduser' command is a more user-friendly interface for adding new users.

- It prompts you for additional user information, such as the full name, password, and other details.

3. Through the graphical user interface (GUI):

- Many Linux distributions provide a graphical tool for managing users and groups.

- This allows you to add new users through a point-and-click interface, without needing to use the command line

Task 14: Can you change the password of a user? How do you do that?

Solution: Controlling password permissions is an important aspect of Linux security, ensuring users can securely update their credentials while also allowing authorized administrators to manage passwords for other accounts as needed.

1. Using the 'passwd' command:

- The 'passwd' command allows users to change their own passwords.

- Users can simply run 'passwd' and follow the prompts to update their password.

2. Granting permission to change other users' passwords:

- To let an administrator, change another user's password, you'd use the 'passwd username' command.

- This requires the necessary permissions, typically granted to users with sudo or root access.

3. Configuring password policies:

- Linux systems have configuration files that define password policies, such as minimum length, complexity, and expiration.

- Administrators can modify these settings to enforce stronger password requirements across the system.

Task 15: What is difference between Process and Thread?

Solution: Processes and threads work together on a CPU, with the operating system managing their execution and resource allocation. Processes provide isolation, while threads enable parallelism and better utilize the available CPU cores. Understanding the difference between processes and threads is essential for optimizing application performance and understanding the inner workings of a Linux system's CPU utilization.

Processes:

In a Linux system, a process represents a running instance of a program. Each process has its own memory space and resources allocated by the operating system.

Threads:

Threads are lightweight sub-processes that run within a parent process. They share the same memory space but can execute tasks independently, improving overall system efficiency.

## Task 16 : Doc 14 Linux Grep commands(Execute them & share SS)

Solution : Consider the below file as an input.

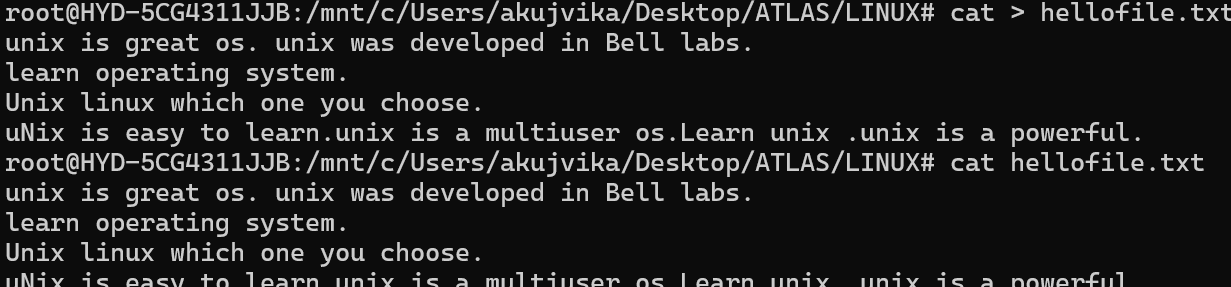
cat > hellofile.txt

*unix is great os. unix was developed in Bell labs.*

*learn operating system.*

*Unix linux which one you choose.*

*uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.*

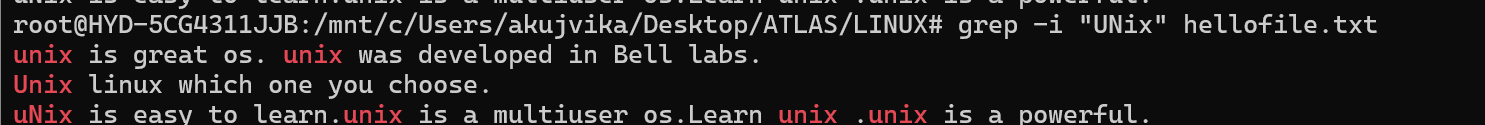


**1. Case insensitive search**

The -i option enables to search for a string case insensitively in the given file. It matches the words like "UNIX", "Unix", "unix".

grep -i "UNix" hellofile.txt

**Output :**



Case insensitive search

## 2. Displaying the Count of Number of Matches Using grep

We can find the number of lines that matches the given string/pattern

grep -c "unix" hellofile.txt

**Output:** 

Displaying the count number of the matches

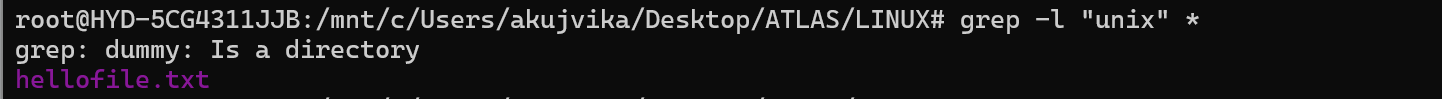
## 3. Display the File Names that Matches the Pattern Using grep

We can just display the files that contains the given string/pattern.

grep -l "unix" \*

**or**

grep -l "unix" f1.txt f2.txt f3.xt f4.txt

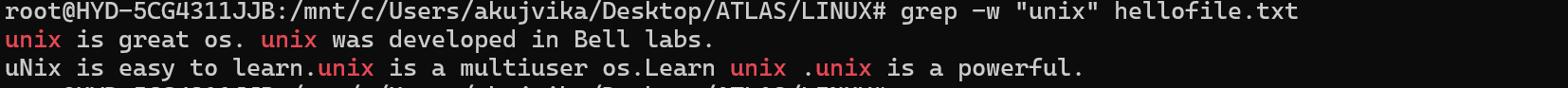
**Output:** 

The file name that matches the pattern

## 4. Checking for the Whole Words in a File Using grep

By default, grep matches the given string/pattern even if it is found as a substring in a file. The -w option to grep makes it match only the whole words.

grep -w "unix" hellofile.txt

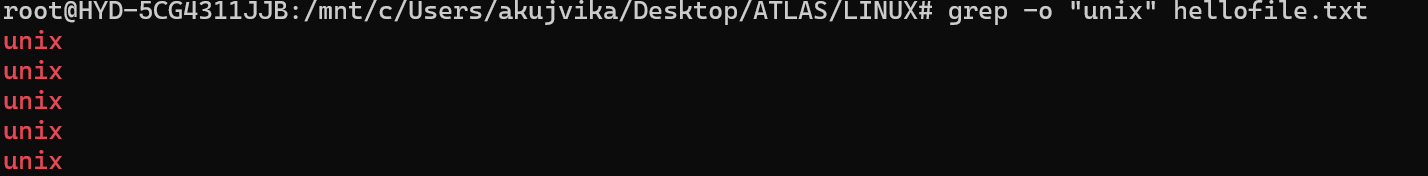
**Output:** 

checking whole words in a file

## 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.

grep -o "unix" hellofile.txt

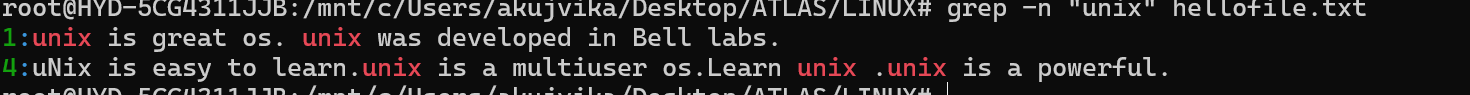
**Output:** 

Displaying only the matched pattern

## 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

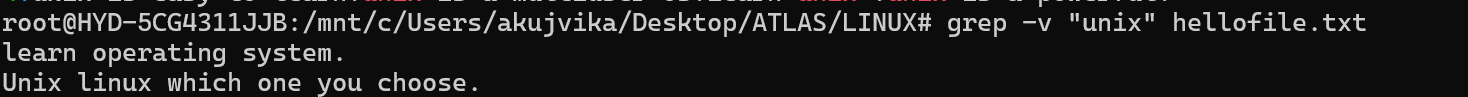
**Output:** 

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:** 

Inverting the pattern match

## 8. Matching the Lines that Start with a String Using grep

The ^ regular expression pattern specifies the start of a line. This can be used in grep to match the lines which start with the given string or pattern.

grep "^unix" hellofile.txt

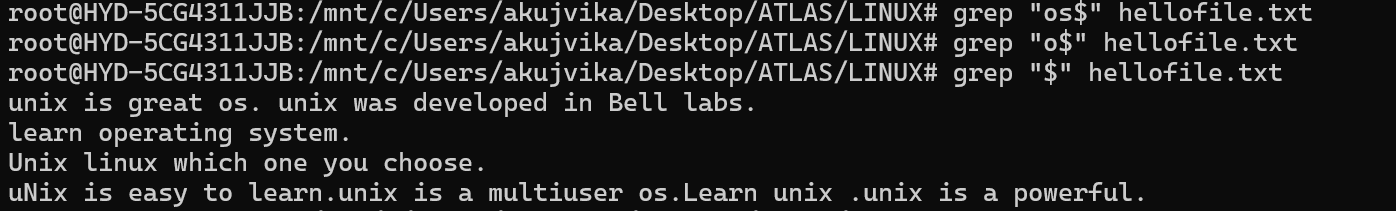
**Output:** 

Matching the lines that start with a string

## 9. Matching the Lines that End with a String Using grep

The $ regular expression pattern specifies the end of a line. This can be used in grep to match the lines which end with the given string or pattern.

grep "os$" hellofile.txt



## 10.Specifies expression with -e option

Can use multiple times :

grep –e "Agarwal" –e "Aggarwal" –e "Agrawal" hellofile.txt

## 

## 

## 11. -f file option Takes patterns from file, one per line

cat pattern.txt

*Agarwal  
Aggarwal  
Agrawal*

grep –f pattern.txt  hellofile.txt

## 

## 12. Print n Specific Lines from a File Using grep

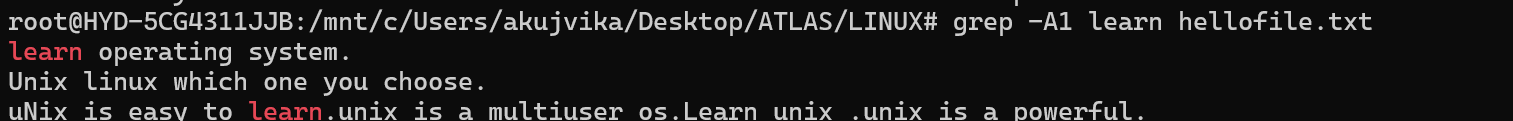
-A prints the searched line and n lines after the result, -B prints the searched line and n lines before the result, and -C prints the searched line and n lines after and before the result.

**Syntax:**

grep -A[NumberOfLines(n)] [search] [file]    
grep -B[NumberOfLines(n)] [search] [file]    
grep -C[NumberOfLines(n)] [search] [file]

**Example:**

grep -A1 learn hellofile.txt

**Output:**  

Print n specific lines from a file

## 13. Search Recursively for a Pattern in the Directory

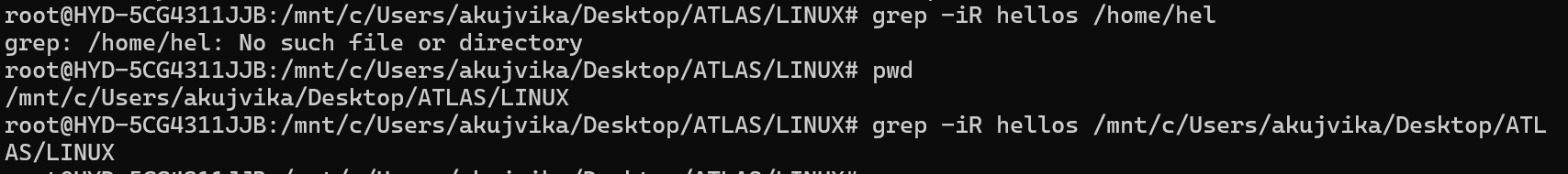
**-R**prints the searched pattern in the given directory recursively in all the files.

**Syntax:**

grep -R [Search] [directory]

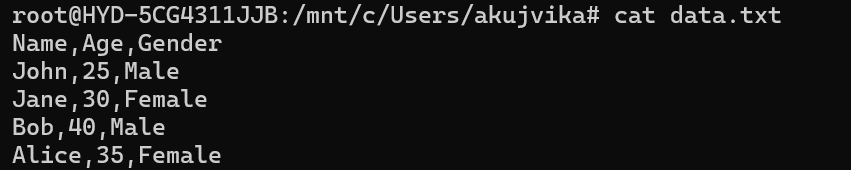
**Example :**

grep -iR hellos /home/hel



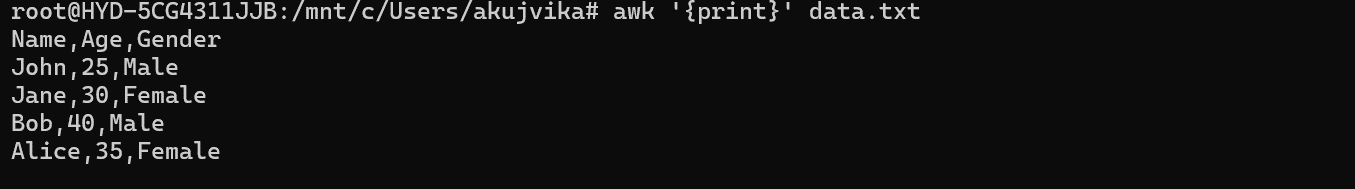
## Task 17 : AWT commands in doc 15 Linux AWK commands

Solution : To try out the AWK commands, we have created a dummy file called data.txt

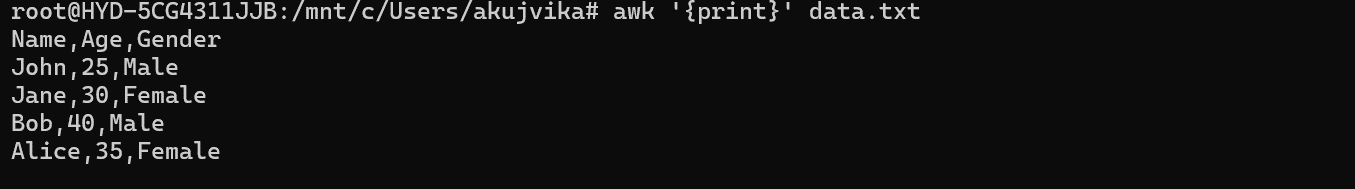


Example 1: Print Contents of a File

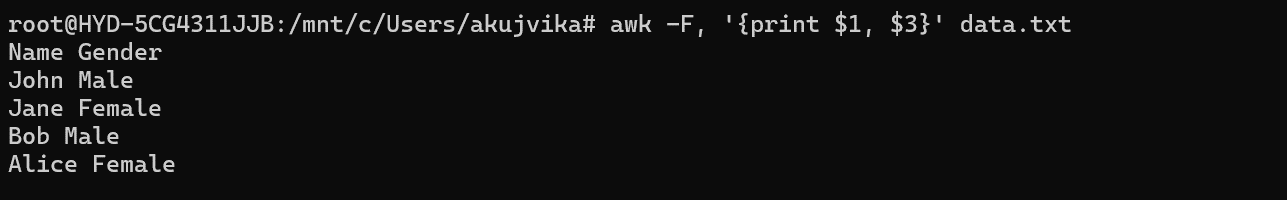
awk '{print}' data.txt



Example 2: Print Specific Columns of a File

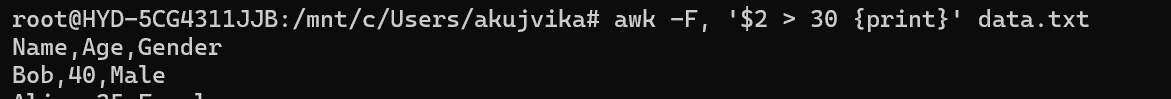
awk -F, '{print $1, $3}' data.txt 

Example 3: Filter Lines Based on a Condition

awk -F, '$2 > 30 {print}' data.txt

Example 4: Sum Values in a Column

awk -F, '{sum += $2} END {print "Total age:", sum}' data.txt



Example 5: Extract Substring from a Column

awk -F, '{sum += $2} END {print "Average age:", sum/NR}' data.txt



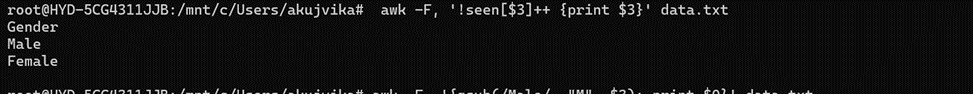
Example 6: Join Two Files Based on a Common Column

awk -F, 'END {print "Number of records:", NR}' data.txt



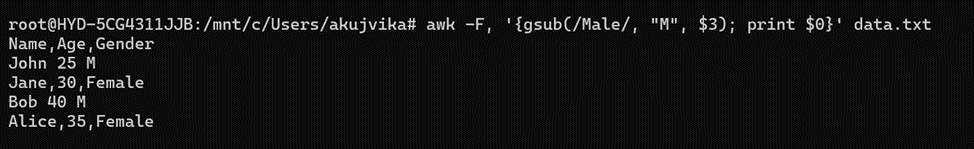
Example 7: Extract First Column of a File

awk -F, '!seen[$3]++ {print $3}' data.txt



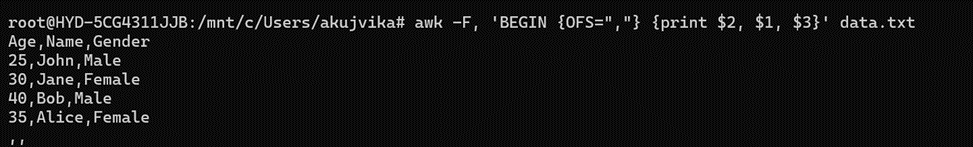
Example 8: Calculate Average of a Column

awk -F, '{gsub(/Male/, "M", $3); print $0}' data.txt



Example 9: Replace a String in a File

awk -F, 'BEGIN {OFS=","} {print $2, $1, $3}' data.txt



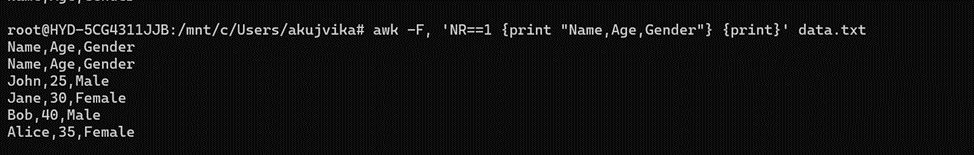
Example 10: Display Last Field of a File

awk -F, 'NR==1 {print} END {print}' data.txt



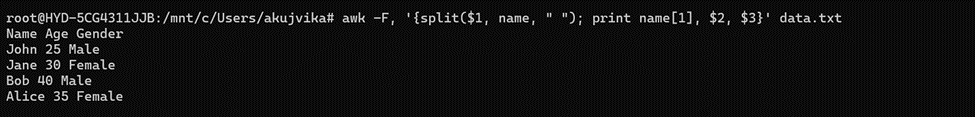
Example 11: Using Regular Expressions

awk -F, 'NR==1 {print "Name,Age,Gender"} {print}' data.txt



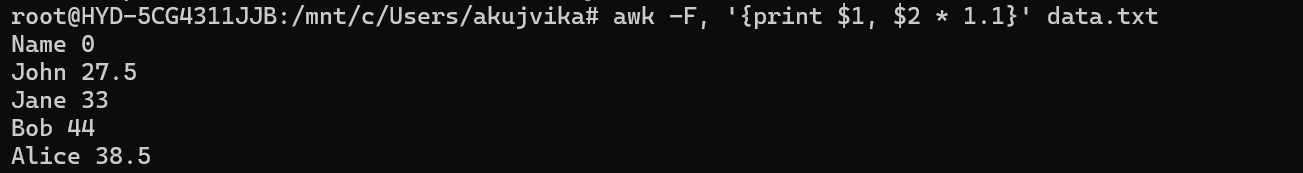
Example 12: Using Variables

awk -F, '{split($1, name, " "); print name[1], $2, $3}' data.txt



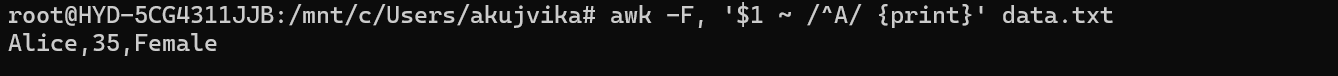
Example 13: Using Built-in Variables

awk -F, '{print $1, $2 \* 1.1}' data.txt



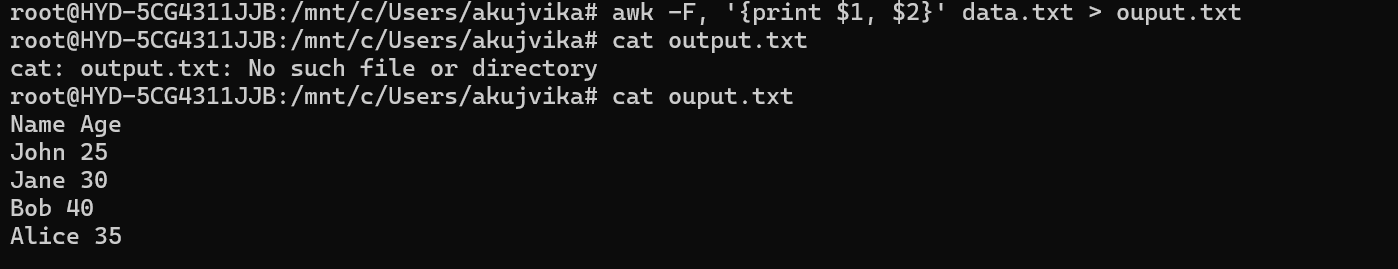
Example 14: Using Control Statements

awk -F, '$1 ~ /^A/ {print}' data.txt

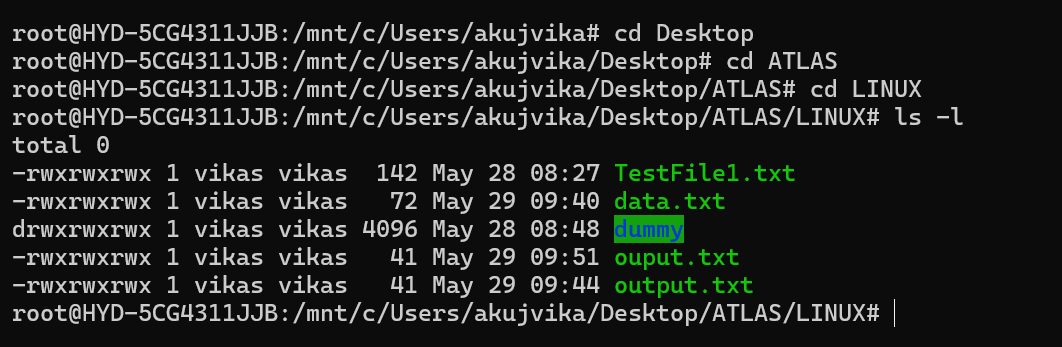


Example 15: Using Functions

awk -F, '{print $1, $2}' data.txt > ouput.txt



Task 18: How to check file access permission in Linux?(Hint use:Ls -l)

Solution : 

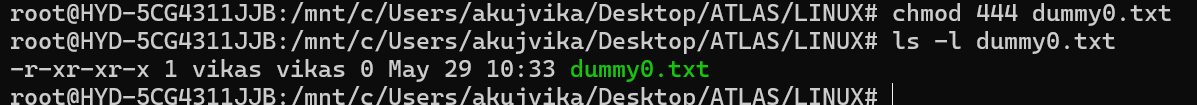
Task 19: What are the default permissions for a new file? (Plz find out for Owner   → ? , Group → ? , All and others → ? )

Solution: The default permissions for a new file in Linux are pretty straightforward. As the owner of the file, I have the ability to both read and write to it. That makes sense since I created the file, and I should be able to work with it however I need to.

For the group I'm associated with, they're granted read-only access. That means they can view the contents of the file, but they can't make any changes to it. And for everyone else who isn't part of my group, they also have read-only permissions.

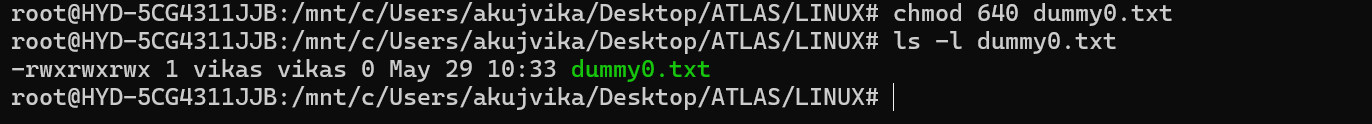
This default setup gives me, as the file owner, the most control, while still allowing a limited level of access for my group and other users. It's a pretty standard and sensible way to handle permissions when creating new files in Linux

Task 20: What is the command to change the permission to read only for the owner, group and all other users? (Hint: chmod 444 filename)

Solution : 

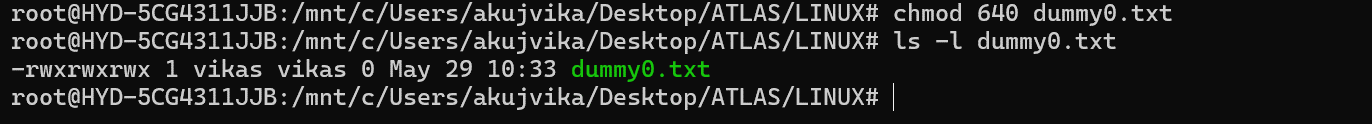
Task 21: Can you change the file permissions to match the following:

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

Solution : 

Task 22: What was the command for changing the file permissions to -rw-r-----?(Hint : use chmod 640 filename)

Solution :

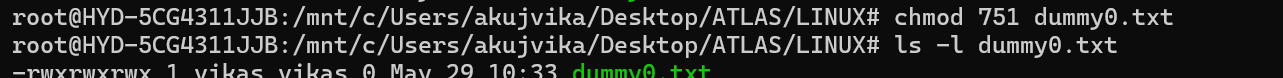


Task 23: Change chmod.exercises permissions to -rwxr-x—x & Change the file permissions to match the following:

owner: Read, Write and Execute

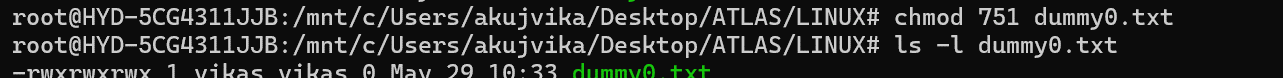
group: Read and Execute

other: Execute

Solution : 

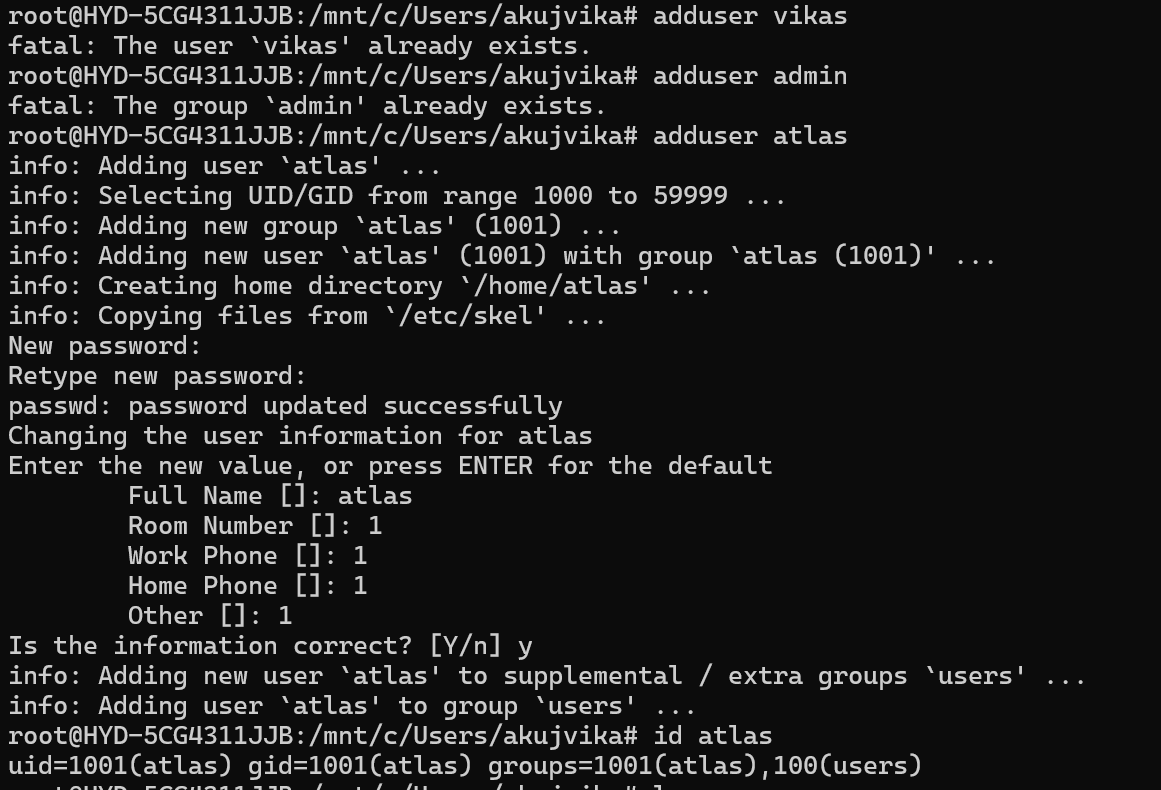
Task 24: What was the command for changing the file permissions to -rwxr-x—x(Hint : use chmod 751 filename)

Solution :



Task 25: what will this command do? (chown -c master file1.txt)

Solution : The command `chown -c master file1.txt` will change the owner of `file1.txt` to the user "master" and print a line indicating the change. In this scenario, we have added a new user named ATLAS & changed owner from Vikas to ATLAS for chmod.exercises.txt

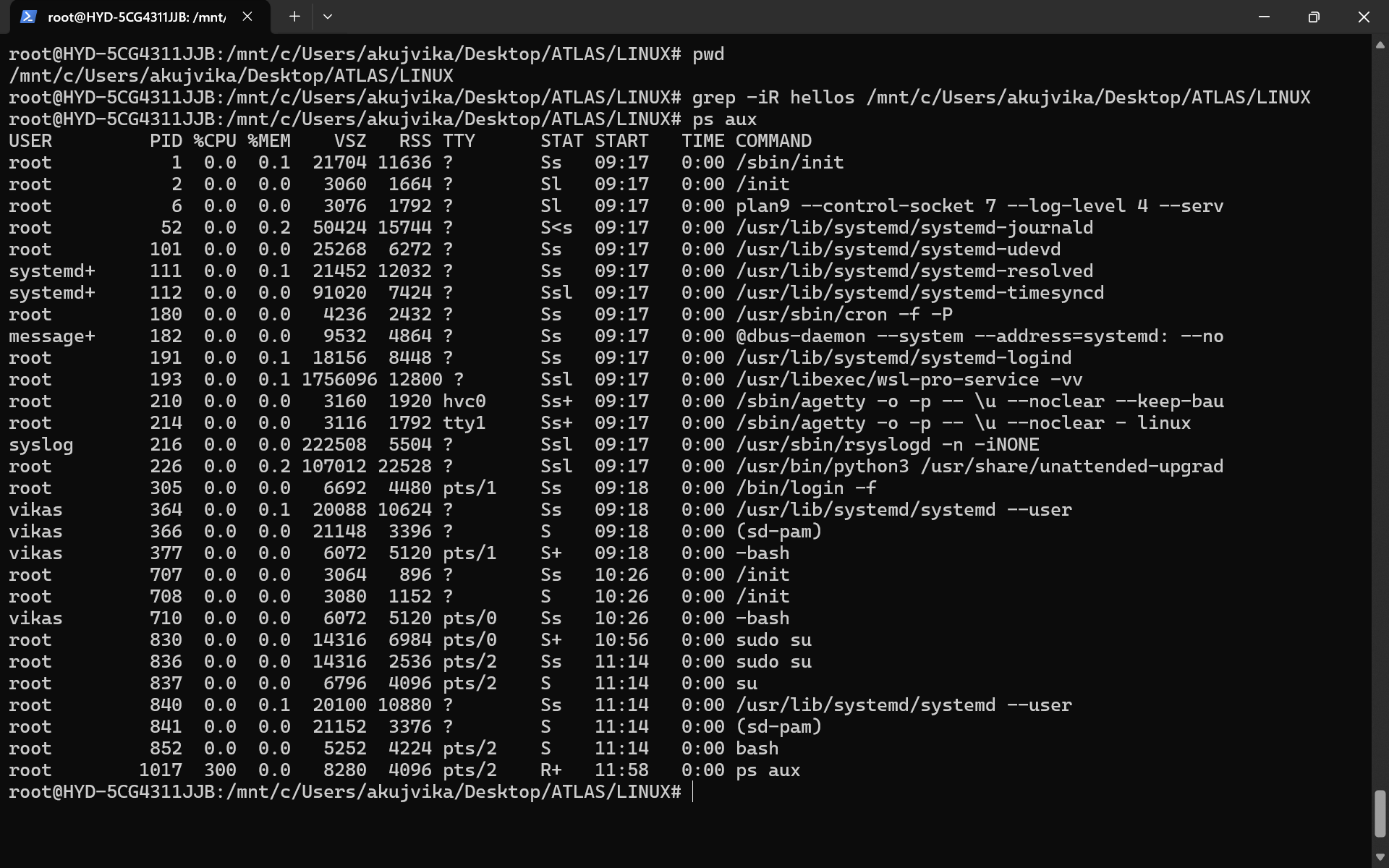




## Processes

Task 26: Can you define what is a process?

Solution : A process is just a running program on your Linux system. Every app, command, or background service you start becomes a process with its own ID (PID) and memory space. The system manages these like a traffic controller - starting, stopping, and keeping them separate so they don’t crash each other.



Task 27: What is command to check foreground process and background process

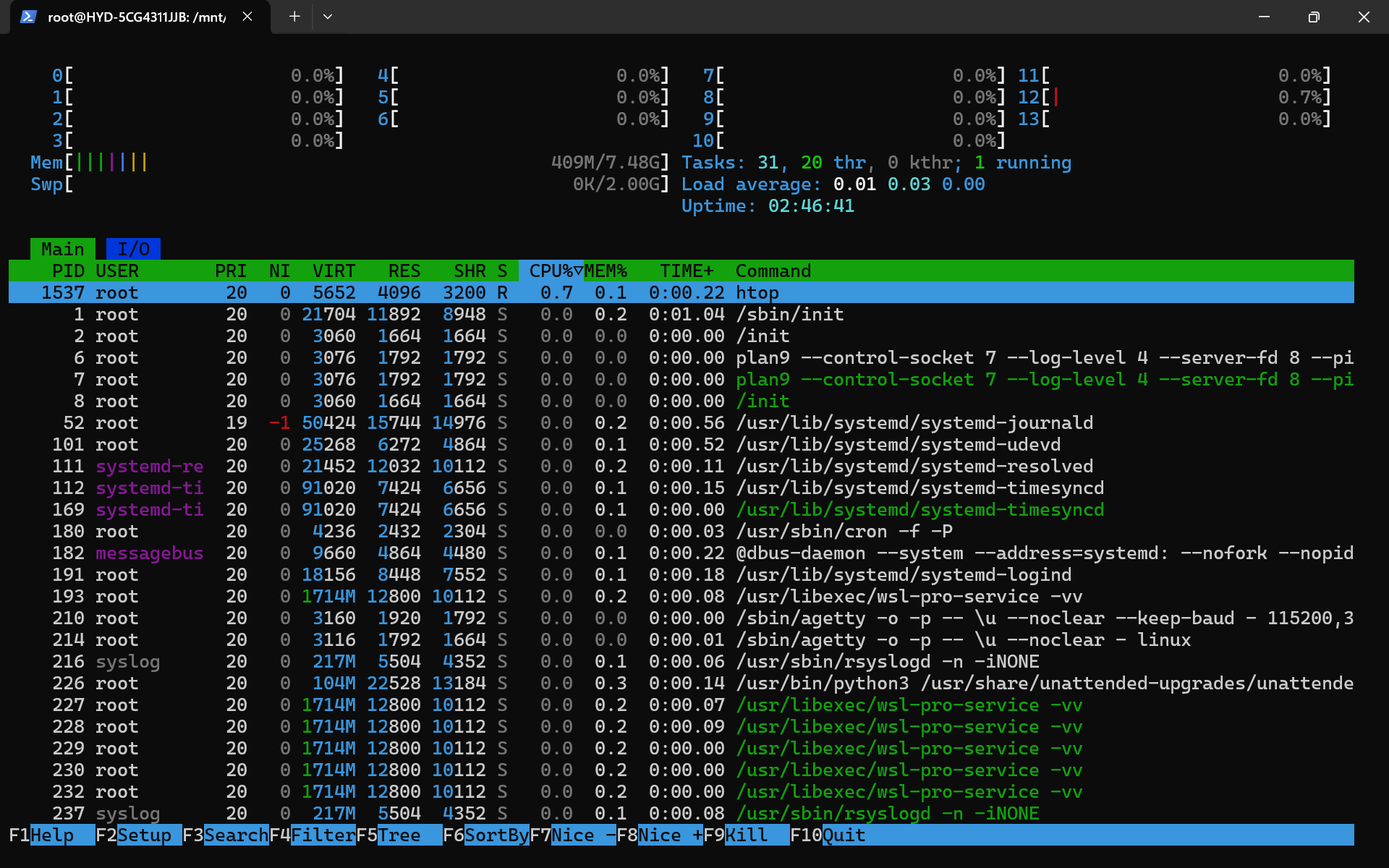
Solution : Foreground processes will stop if you close the terminal & Background processes (started with &) keep running . Command to show user processes : ps aux | grep yourname

Foreground: jobs -l (current terminal's active jobs)

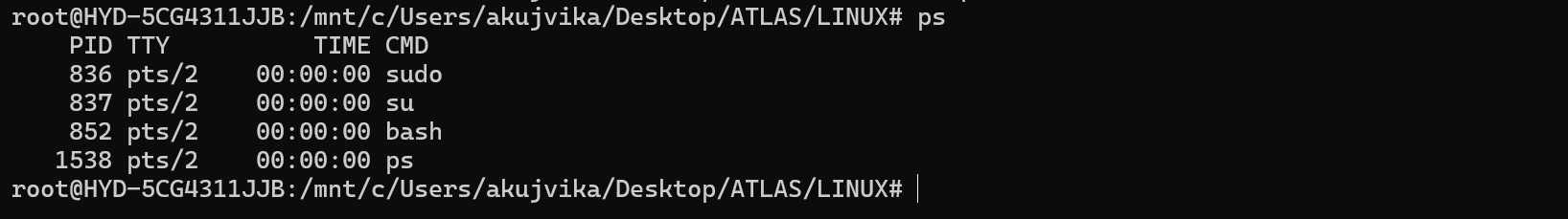
Background: ps aux (all system processes)

Task 28: Can you list all the running processes?(Hint use ps)

Solution : using htop



Using ps :



Task 29: What will ps -f command do? plz try n check .. ss required.

Solution : The ps -f command shows a full-format listing of processes associated with your current terminal session. Here's what it displays in a clean, columnized view:

UID: User running the process

PID: Process ID

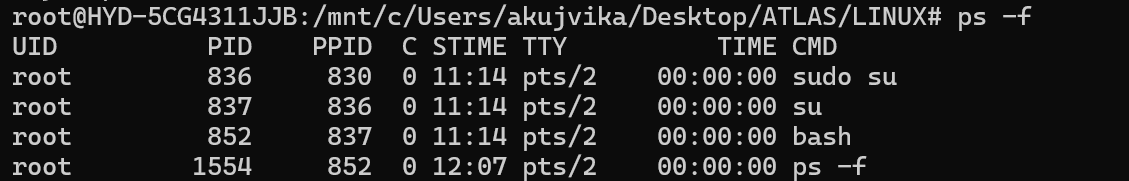
PPID: Parent Process ID

C: CPU utilization

STIME: Start time

TTY: Terminal associated

CMD: Full command (shows arguments)



## Let’s play with shell variables

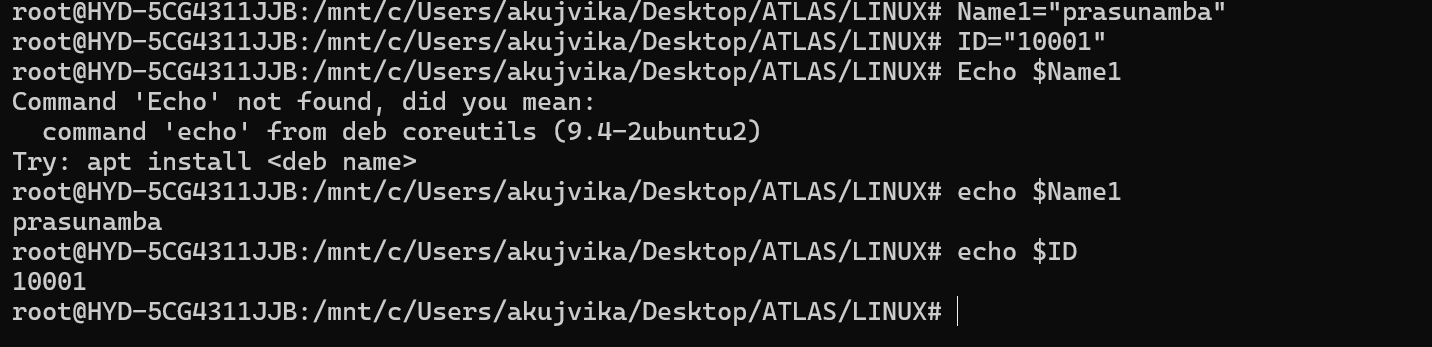
Task 30: Can you create a variable name with your name in it & Check the output

Ex:

Name =  “prasunamba”

Id  = 10001

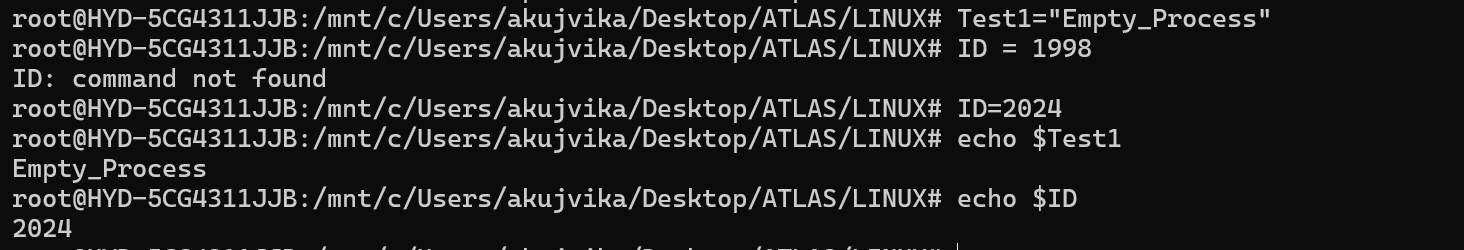
Echo $Name

Solution : 

Example 2 : Created variable Test1 as Empty\_Process

Set a numeric ID 2024

“Echo $variablename” prints their values



Task 31: Can you make the above name variable read only . what will this display? Is it saying read only

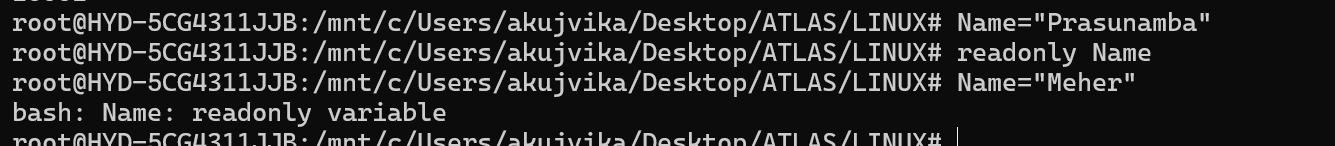
Ex:

Name = “Prasunamba”

Readonly Name

Name = “Meher” —>

Solution : Yes . readonly made it unchangeable & Trying to modify it fails with an error



Task 32: Now will unset or delete the variables

Use the below command and check

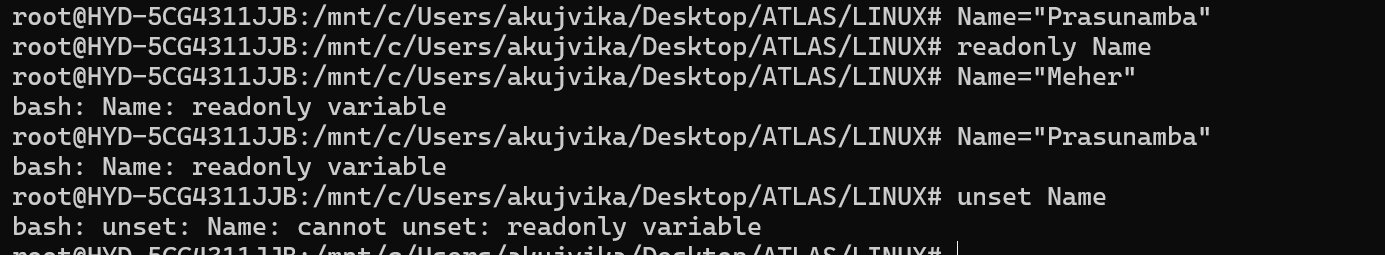
Unset Name

Now check for

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

Solution : readonly makes the variable permanent for the current shell session while unset can't remove read-only variables (protection feature) . The variable and its value persist until you close the terminal

Note : Normal variables can be unset, but read-only ones are locked



## Variable Types

When a shell is running, three main types of variables are present −

Local Variables − A local variable is a variable that is present within the current instance of the shell. It is not available to programs that are started by the shell. They are set at the command prompt.

Environment Variables − An environment variable is available to any child process of the shell. Some programs need environment variables in order to function correctly. Usually, a shell script defines only those environment variables that are needed by the programs that it runs.

Shell Variables − A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly. Some of these variables are environment variables whereas others are local variables.

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]}"

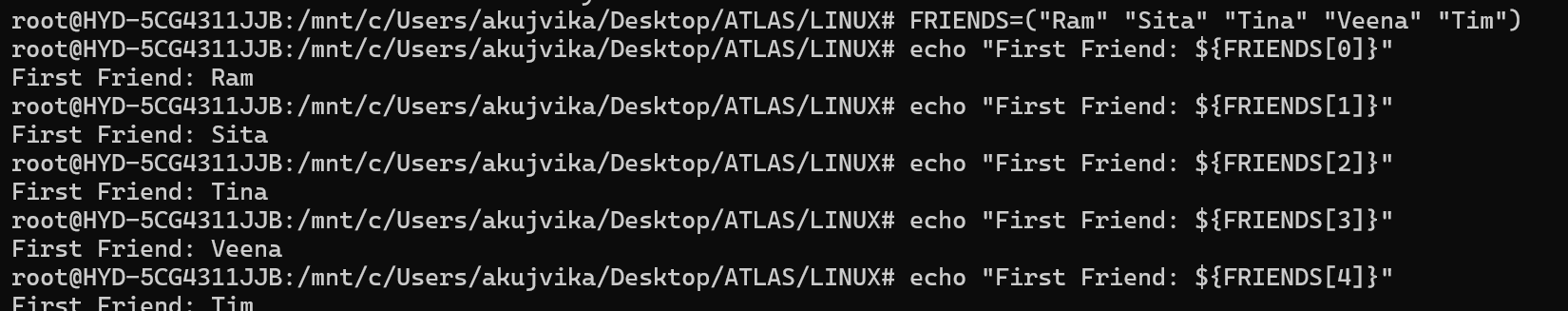
Solution : Here the following steps are taking place :

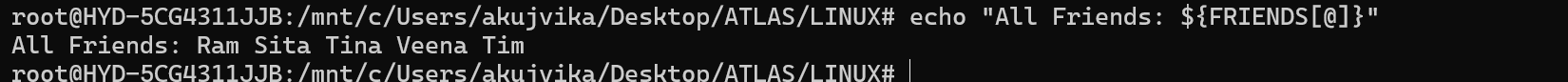
Arrays start at index 0

Use ${array[index]} syntax to access values

No commas between elements - just spaces

Works in bash (not all shells support arrays)



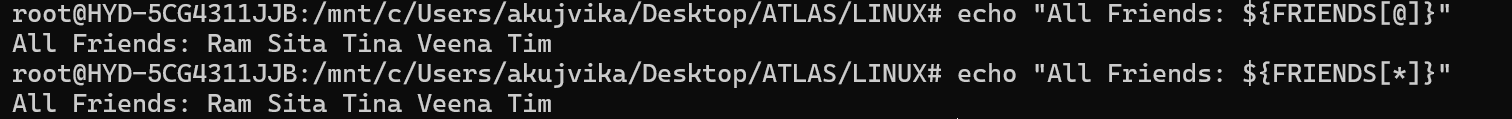


Task 34: Can you print all the list at once in an array. Try the below cmds and check

Echo “${array\_name[\*]}”

Echo “${array\_name[@]}”

Solution : yes we can print all of them. While [\*] combines all elements into one string ; [@] treats each element separately (better for loops)



Operators 👍

* Arithmetic Operators
* Relational Operators
* Boolean Operators
* String Operators
* File Test Operators

If else

if...fi statement

if...else...fi statement

if...elif...else...fi statement

case...esac statement

The while loop

The for loop

The until loop

The select loop

Task 35: Plz let me know what’s 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`

Solution :

