Day 6

**Task 1: Regex uses in Linux.**

Character Matching

. Matches any single character except newline

[...] Matches any one character in the set

[^...] Matches any character not in the set

^ Anchors the match at the start of a line

$ Anchors the match at the end of a line

\ Escape character (used to treat special characters literally)

Quantifiers and Repetition

\* Matches the preceding character 0 or more times

\+ Matches 1 or more times

\? Matches 0 or 1 time

{n} Exactly n times

{n,} n or more times

{n,m} Between n and m times

**TASK 2: LINUX OS, features of LINUX**

Linux is an operating system, just like Windows or macOS. It's the part of your computer that helps all the programs and hardware (like your keyboard, mouse, and internet) work together smoothly.

But here's the special thing about Linux:

It's free, open-source, and super flexible. That means anyone can use it, change it, and even build their own version of it.

**1. It’s Free**

You don’t have to pay anything to use Linux. No license fees. It’s just... free forever.

**2. It’s Open**

Anyone can see how Linux works under the hood. Techies can tweak it, improve it, or build cool things on top of it.

**3. It’s Super Reliable**

Linux rarely crashes. That’s why it’s used to run big websites, companies, and even space missions!

**4. It’s Safe**

Viruses are very rare on Linux. It’s built to be secure, and users don’t usually have admin powers unless they really need them.

**5. It’s Fast and Light**

Linux doesn’t need a powerful computer. It runs well even on older machines.

**6. You’re the Boss**

You can customize everything — from how it looks to how it works. Want a purple terminal? You got it. Want to make your own commands? Go for it!

**7. It Can Do a Lot**

Whether you're browsing the web, coding, editing photos, or running a server — Linux can handle it all.

**8. Great for Developers**

If you’re into programming, Linux is like heaven. It supports almost every language and tool you can think of.

**9. Strong Community Support**

Stuck with something? Just ask online — thousands of Linux users around the world are happy to help.

**10. Runs Everywhere**

From phones (like Android), smart TVs, supercomputers, to your personal laptop — Linux is everywhere!

**TASK 3: What is KARNEL.**

Think of the kernel as the brain of your operating system. It's the middleman between your computer's hardware (like CPU, RAM, and disk) and the software you run (like browsers, games, or apps).

**How Karnel works:**

Let’s say you want to open a file:

Your app says, “Hey computer, I need this file.” The kernel takes that request and says, “Okay, I’ll get it from the hard drive for you.” It talks directly to the hardware, gets the file, and gives it back to your app. Your programs don’t talk to hardware directly — they always go through the kernel.

**In Simple Terms:**

Your software: “Can I use the printer?” The kernel: “Sure, let me ask the printer and make sure it's ready.”

**Why Is It Important?**

Without the kernel, your computer wouldn't know how to run anything.

**It manages:**

Memory (what program uses how much RAM)

Processes (which programs run and in what order)

Hardware access (keyboard, mouse, printer, etc.)

Security (who can do what on the system)

In Linux, the Linux kernel is what powers all Linux-based systems — from Ubuntu laptops to Android phones to huge servers.

It’s open-source, meaning people all over the world help make it better.

Quick Analogy:

You = software

Hardware = tools

Kernel = translator

> You say “cut this wood,” the kernel figures out how to use the saw.

**TASK 4: BASH in LINUX full form and Explanation**

**BASH = Bourne Again SHell**

Bash is a command-line interpreter — it lets you talk to the Linux system using commands.

When you open a terminal in Linux (like Ubuntu or WSL), you're usually using Bash.

It understands your commands (like ls, cd, mkdir, echo, etc.), runs them, and shows the output.

**Example:**

#!/bin/bash

echo "Hello, World!"

Run this and it prints: Hello, World!

**Features:**

Command history (press ↑ to see previous commands)

Auto-completion using TAB

Variables and loops

If-else conditions

Built-in math

Powerful scripting capabilities

**BASH used?**

Terminal in Linux/Mac

Server maintenance

Automation scripts (backups, deployments)

System administration

**TASK 5: LINUX Vs Windows**

Linux and Windows are both operating systems, but they differ greatly in how they function and who they’re built for. Windows is a closed-source, paid system developed by Microsoft, known for its user-friendly interface and wide compatibility with software and games. It’s widely used in homes, schools, and offices. Linux, on the other hand, is free and open-source, meaning anyone can view, modify, and share its code. It’s highly customizable, more secure, and lightweight, making it popular among developers, tech-savvy users, and for use in servers and cybersecurity. While Windows focuses on ease and familiarity, Linux offers more control, flexibility, and performance, especially for programming and system management tasks.

**Task 6: Components of LINUX? Describe in detail with diagram.**

**1. Kernel**

Core of the Linux system. Manages hardware like CPU, memory, and devices. Handles process control, memory management, device control, and system calls. Acts as a bridge between software and hardware.



**2. Shell**

The command-line interpreter. Takes user input (commands), passes it to the kernel, and shows output.

Can be:

BASH (Bourne Again SHell) – the most common

Others: ZSH, KSH, CSH, etc.

Supports scripting to automate tasks.

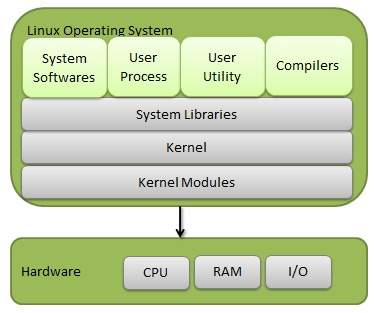


**3. System Libraries**

These are special programs that help applications interact with the kernel.

Provide functions that apps can use without needing to talk to the kernel directly.

Example: glibc – GNU C Library used in many Linux programs.



**4. System Utilities**

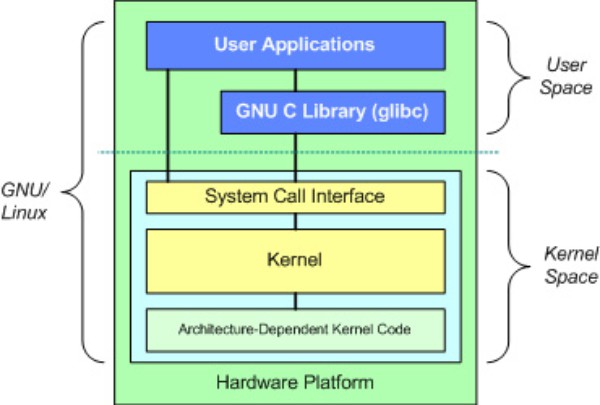
Basic tools and software provided by the OS to perform tasks like:

File management (cp, mv, rm)

Disk management

Network configuration

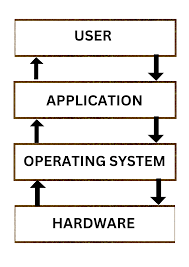
These tools make it easier to use the system.



**5. User Applications**

Programs that users run: text editors, browsers, games, etc.

These interact with the shell or directly with libraries to function.



**TASK 7: Is it legal to edit Kernal ? when do you think we have to in case?**

**Yes**, it’s absolutely legal!

The Linux kernel is released under the GNU General Public License (GPL), which means:

You are free to view, edit, and modify the kernel’s source code.

You can even share your modified version with others — as long as you also share your changes (i.e., keep it open-source).

So, modifying the Linux kernel is 100% legal, unlike proprietary systems like Windows where the core system is locked.

When would you need to edit the kernel?

Most users never need to touch the kernel. But in advanced or special cases, it might be useful:

**1. Adding Support for New Hardware**

If you build a custom device (like a robot or embedded system), you might need to modify the kernel to make it work with your hardware.

**2. Custom Performance Tweaks**

Developers or companies sometimes tweak the kernel to improve performance for specific applications (e.g., high-speed trading, gaming systems, or servers).

**3. Creating a Custom OS**

If you're building a Linux-based operating system (like Android or a Linux distro), you’ll likely make kernel modifications.

**4. Adding New Features or Drivers**

You may add support for a new filesystem, network protocol, or create a custom kernel module for your application.

**5. Security Research or Education**

If you’re learning how OS internals work, modifying the kernel helps you understand how systems manage memory, processes, and hardware.

**TASK 8: What is LILO? Explain**

LILO stands for Linux Loader.

It is a boot loader used in Linux systems to load the operating system into memory when the computer starts up.

What does a boot loader do?

When you turn on your computer, the hardware doesn’t automatically know how to start Linux (or any OS). That’s where the boot loader comes in — it helps load the kernel (the core part of Linux) into memory so that the operating system can start running.

**How LILO Works:**

1. BIOS/UEFI starts when the computer is powered on.

2. It loads LILO, which is installed in the MBR (Master Boot Record) or the boot sector of a partition.

3. LILO then loads the Linux kernel into memory.

4. The kernel takes over and boots the system.

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

In Linux, a shell is a program that allows us to communicate with the operating system by typing commands.

When we open the Terminal and write something like ls to list files, it’s actually the shell that reads that command, passes it to the kernel, and then shows us the result.

So, the shell works like a translator between the user and the system. It takes the commands we type and helps the operating system understand and act on them.

Why do we need a shell?

We can’t directly talk to the Linux kernel (the heart of the system), because it only understands low-level instructions. The shell makes it easier for us to control the system by using simple, human-readable commands.

How many shells are there and what are they?

There are several types of shells in Linux. All of them do the same basic job — they take our commands and interact with the system — but each has its own features and style.

1. **BASH** (Bourne Again Shell) – This is the most commonly used shell in Linux. It’s user-friendly, powerful, and supports features like command history, scripting, and tab completion. Most Linux systems use BASH by default.

2. **SH** (Bourne Shell) – This is the original Unix shell. It’s very simple and is mostly used for basic scripting or compatibility with older systems.

3. **KSH** (Korn Shell) – This shell is faster and includes many features useful for advanced scripting. It’s popular in enterprise environments.

4. **CSH** (C Shell) – This shell uses a syntax that’s similar to the C programming language. It includes job control and command history but isn’t used much nowadays.

5. **ZSH** (Z Shell) – This is a modern shell that offers a lot of helpful features like auto-correction, powerful customization, and themes. Many developers love using ZSH.

6. **Fish** (Friendly Interactive Shell) – This shell is designed to be simple and helpful. It shows suggestions as you type and highlights commands, making it great for beginners.

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

Swap space is a portion of the hard disk that Linux uses as extra memory when your RAM (Random Access Memory) is full.

Think of RAM as your system’s main working area — it stores data and programs while you’re using them. But when the RAM runs out of space (especially if you’re running many applications), the system needs somewhere else to store the extra data. That’s where swap space comes in.

Example to understand:

Imagine RAM as your office desk — it’s fast and easy to reach.

When the desk is full, you move some papers to a drawer (swap space). The drawer isn’t as quick to access, but it gives you more room to keep working.

Uses in SWAP:

Temporarily stores inactive data from RAM.

Helps prevent system crashes due to low memory.

Can be used for hibernation (saving the entire RAM to disk).

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

**What is Mount in Linux?**

In Linux, mounting means making a storage device (like a hard drive, USB, or DVD) accessible to the system and users.

Linux does not automatically show your drives (like Windows does). Instead, you have to "mount" the device to a directory (called a mount point) so you can access its contents.

**Think of it like this:**

> Mounting is like plugging in a USB and choosing a folder where you want to view and use its files.

**Example:**

If you insert a USB drive, Linux doesn’t automatically open it. You need to mount it (manually or automatically) to something like /mnt/usb or /media/usb so you can see the files.

**How to Mount a Drive in Linux:**

You need root (admin) access to mount manually.

**Step to Mount:**

sudo mount /dev/sdb1 /mnt

/dev/sdb1 is the device name (your USB, hard drive, etc.)

/mnt is the mount point (a folder where files will appear)

After this, you can go to /mnt and access the files on the device.

**How to Unmount (remove) a Drive:**

Unmounting means disconnecting the device from the system safely, so no data is lost or corrupted.

**Step to Unmount:**

sudo umount /mnt

or

sudo umount /dev/sdb1

This safely removes the connection.

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

chmod stands for "change mode".

It's a command used to change the permissions of a file or folder (read, write, and execute).

In Linux, every file has permissions that control who can read, write, or run it. The chmod command lets you modify those permissions.

Types of Permissions:

There are three types of permissions:

r = read (view the file)

w = write (edit or delete the file)

x = execute (run the file, if it's a program or script)

And there are three types of users:

u = user (owner)

g = group

o = others (everyone else)

Two Ways to Use chmod:

1. Symbolic Method (using letters)

You can add (+), remove (-), or set (=) permissions like this:

chmod u+x filename # Give execute permission to the owner

chmod g-w filename # Remove write permission from group

chmod o=r filename # Set only read permission for others

2. Numeric (Octal) Method (using numbers)

Each permission is given a number:

Read (r) = 4

Write (w) = 2

Execute (x) = 1

You add the numbers for each user type: Example:

chmod 755 filename means:

User: 7 → 4+2+1 = read, write, execute

Group: 5 → 4+0+1 = read, execute

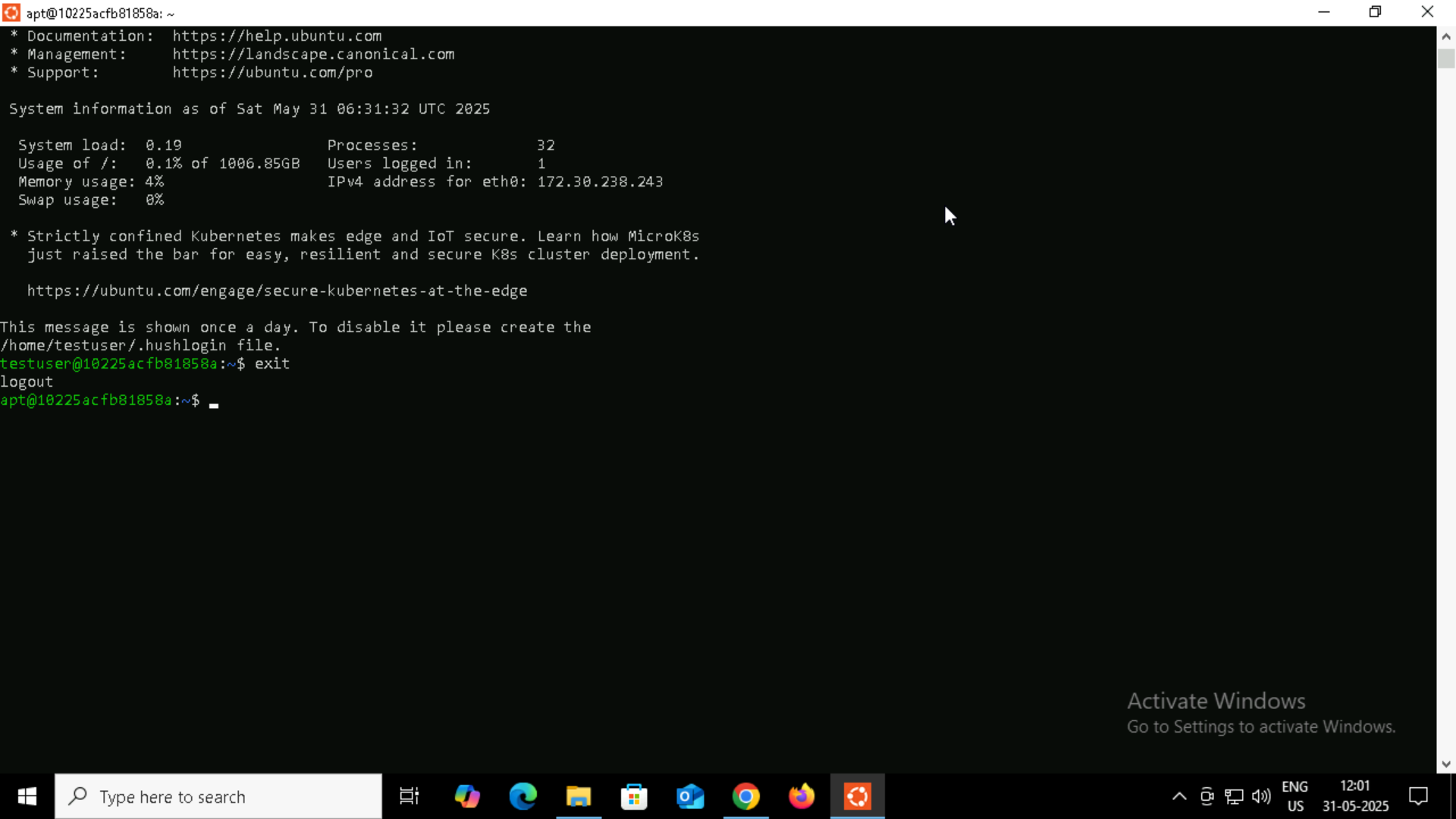
Others: 5 → 4+0+1 = read, execute

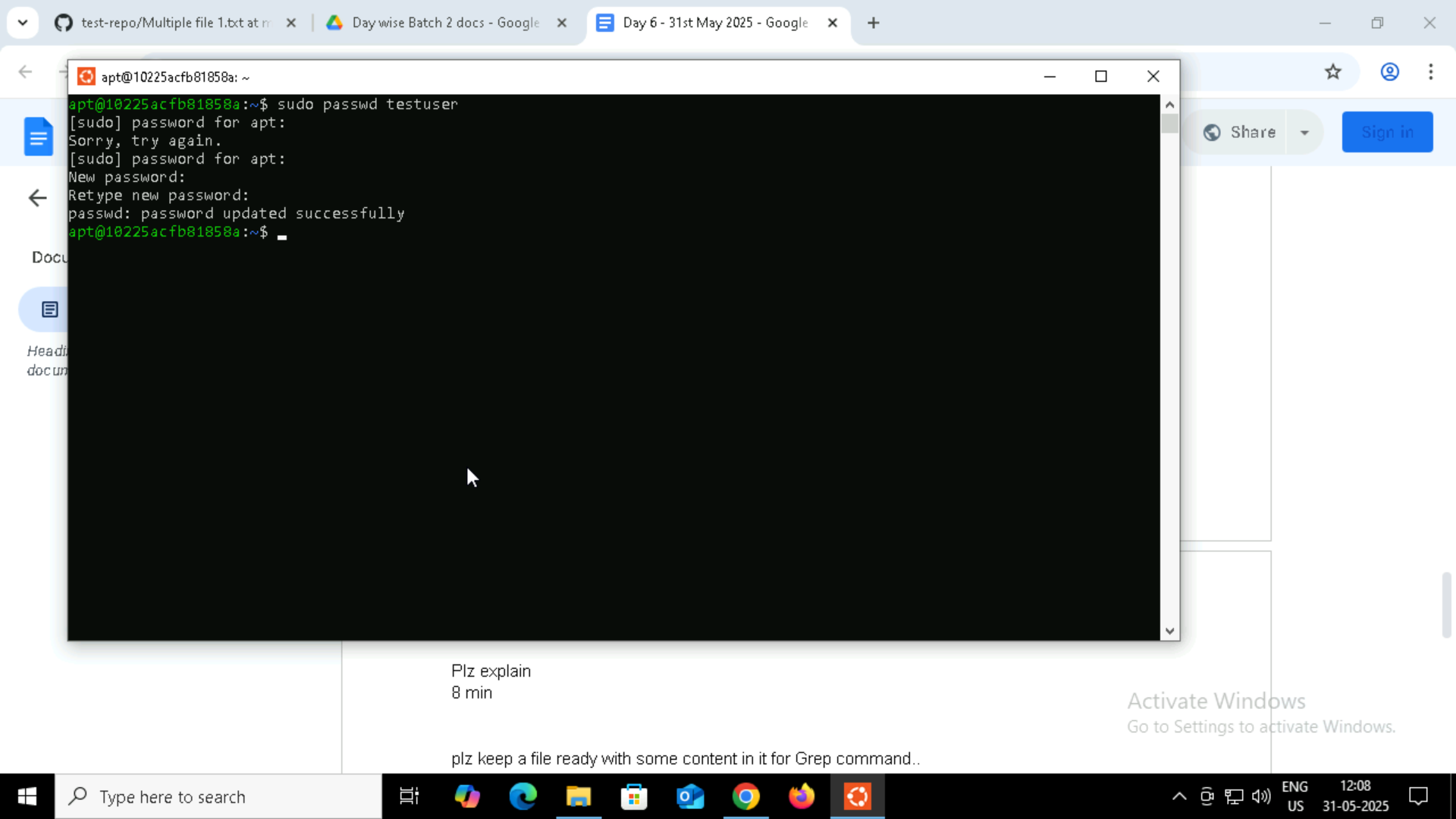
Common Examples:

chmod 777 filename # Everyone can read, write, and execute

chmod 644 filename # Owner can read/write, others can only read

chmod 700 filename # Only owner has full access

**TASK 13: Can you add a new user account? Crate a new user in different ways and paste ss**

**Task 14: Change psswerd for users:** 

**TASK 15: What is diff between Process and Thread?**

A process is an independent program running in memory.

Each process has its own memory space, files, variables, and resources.

When you open a browser or a terminal, each one is a separate process.

Think of a process like a person working in their own office — they have their own desk, files, and tools. They don’t share with others.

What is a Thread?

A thread is a smaller part inside a process.

Multiple threads share the same memory and resources of the process.

They are used to do multiple tasks at the same time within the same program.

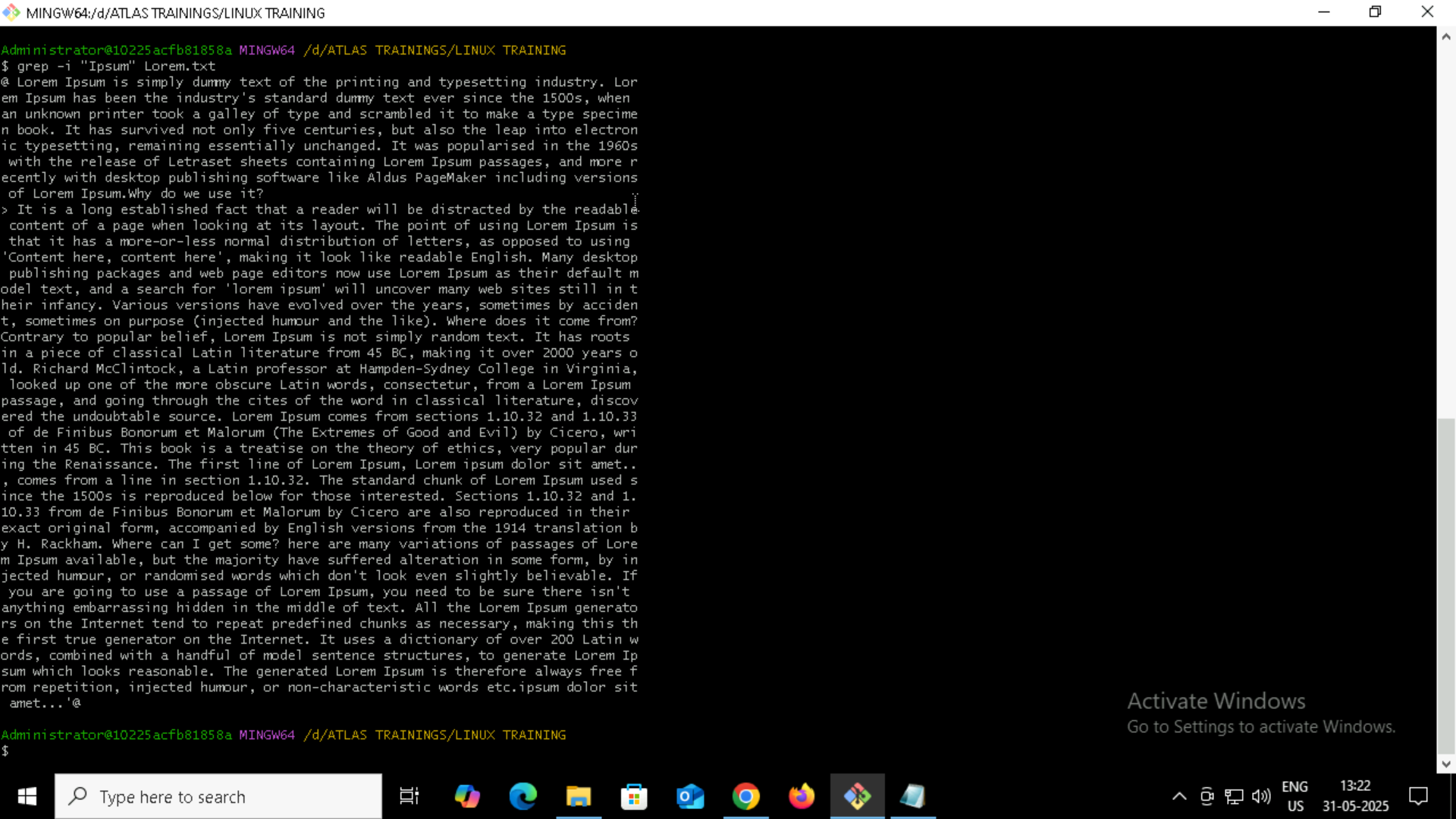
Think of threads like workers inside the same office. They all work together using the same files and space, but do different tasks at the same time.

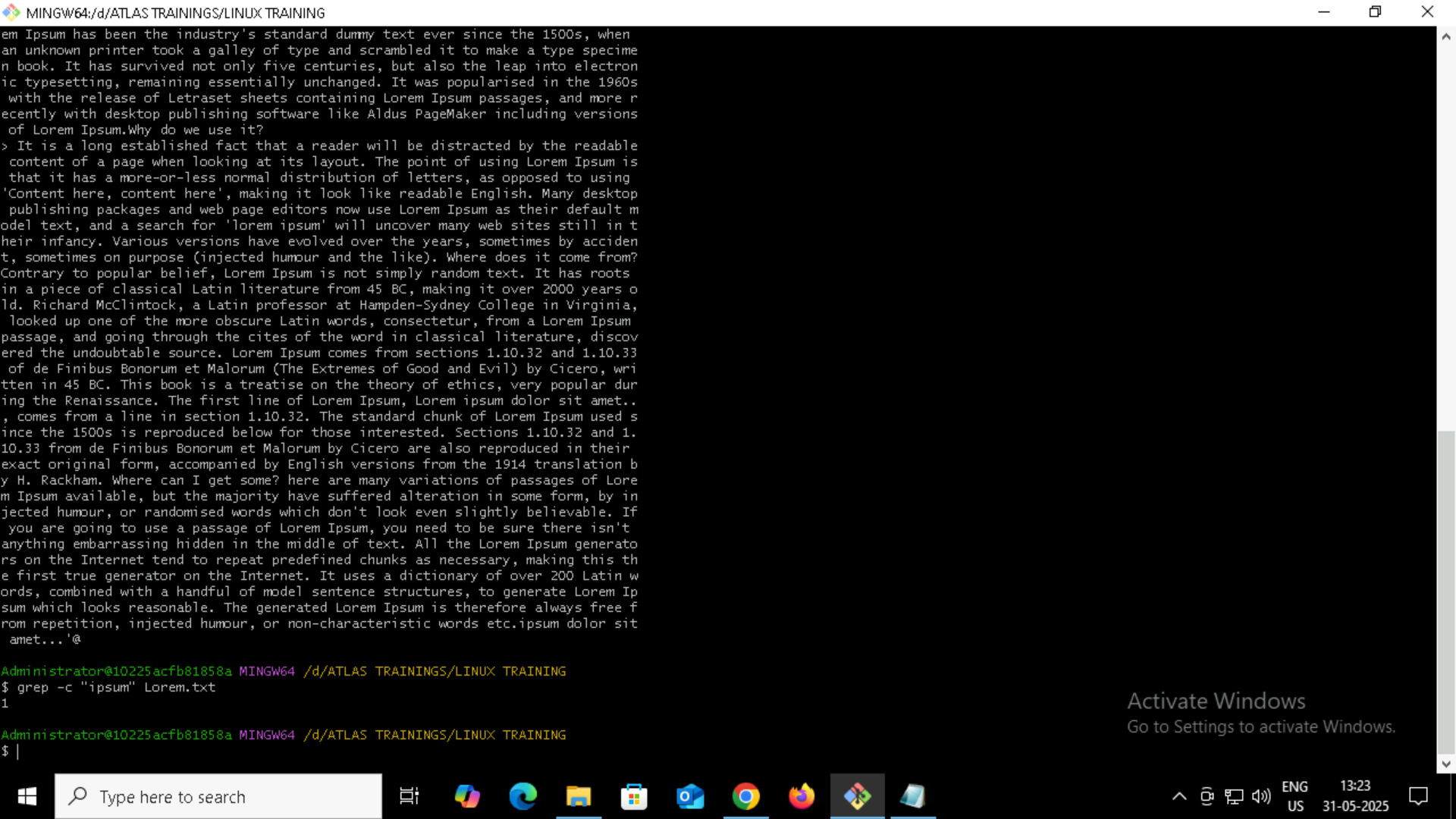
Example:

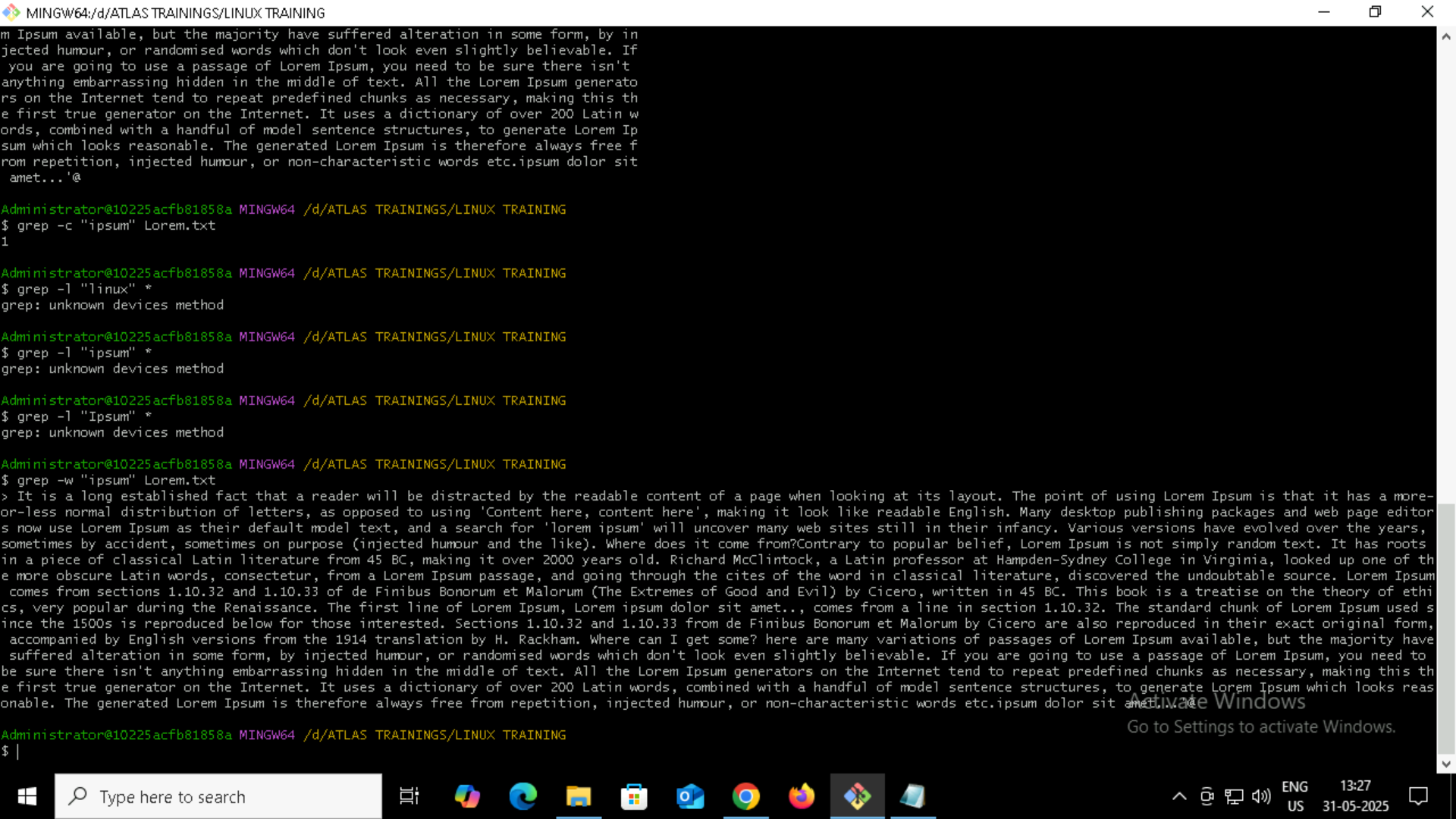
You open a browser → That’s a process.

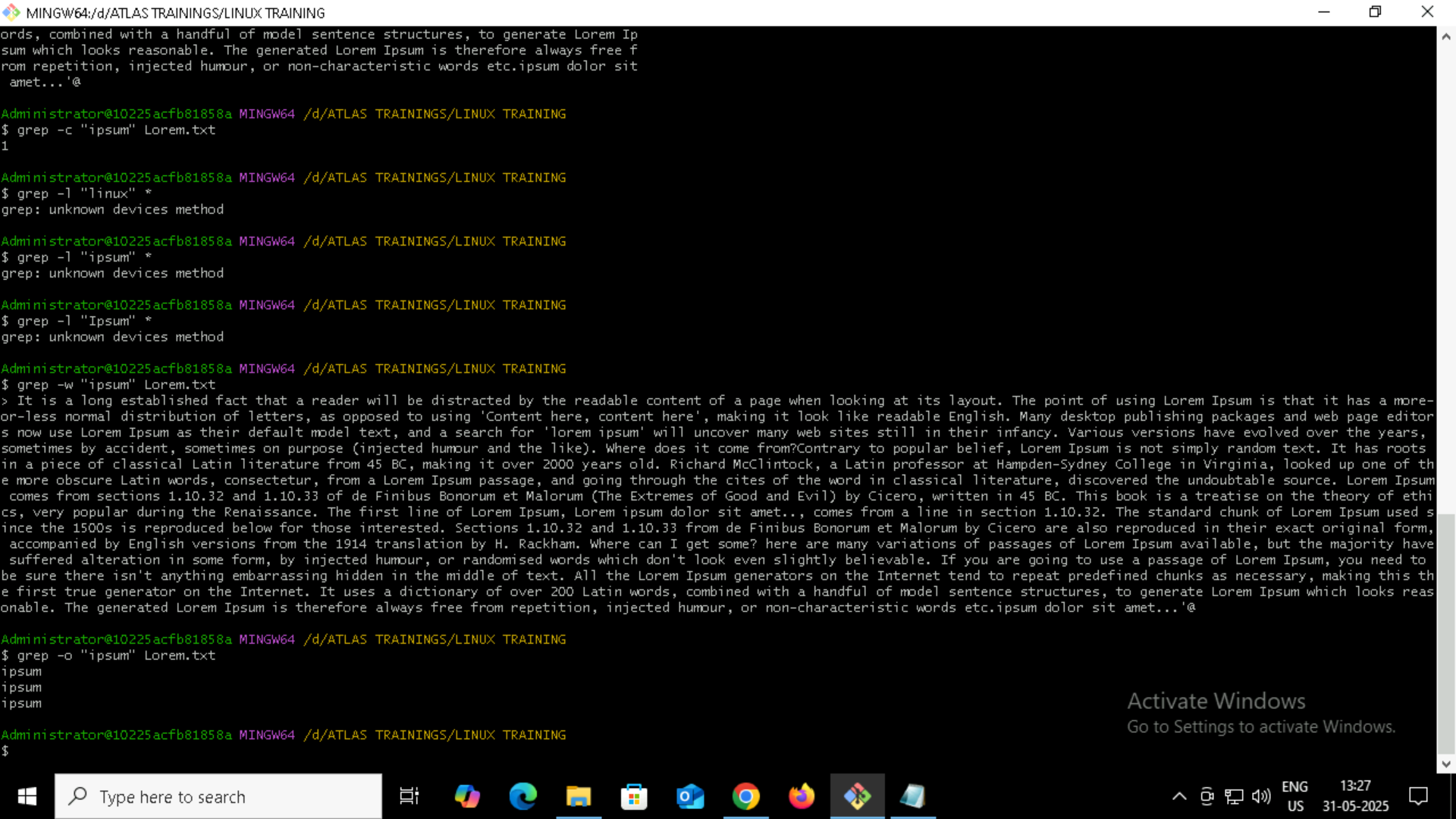
You open multiple tabs → Each tab might be a thread doing something different (loading a page, running a script, etc.).

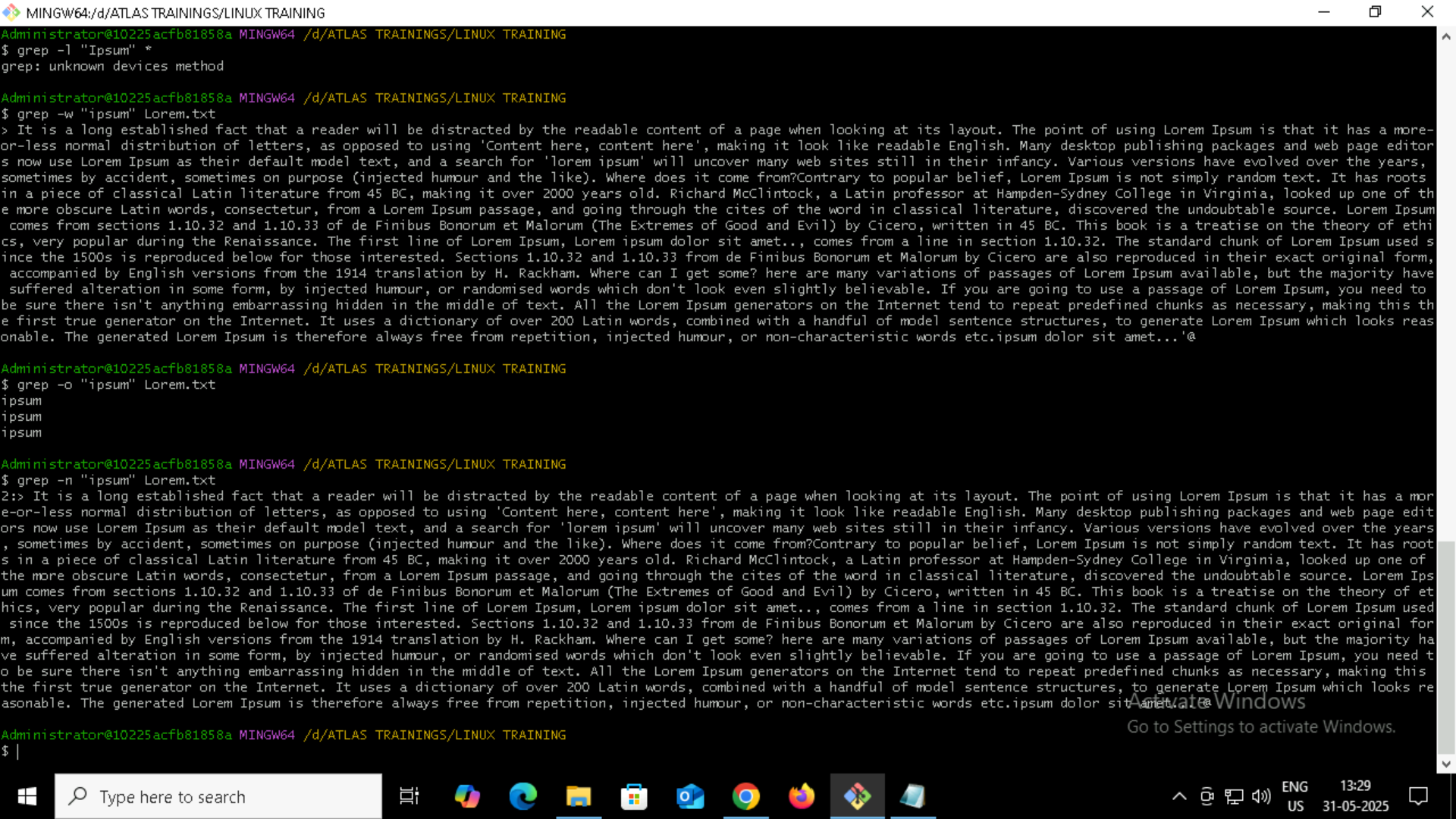
<https://drive.google.com/drive/folders/1vsL_jdQIsTMz92S9s5LDqMrYeDuuwIfx>

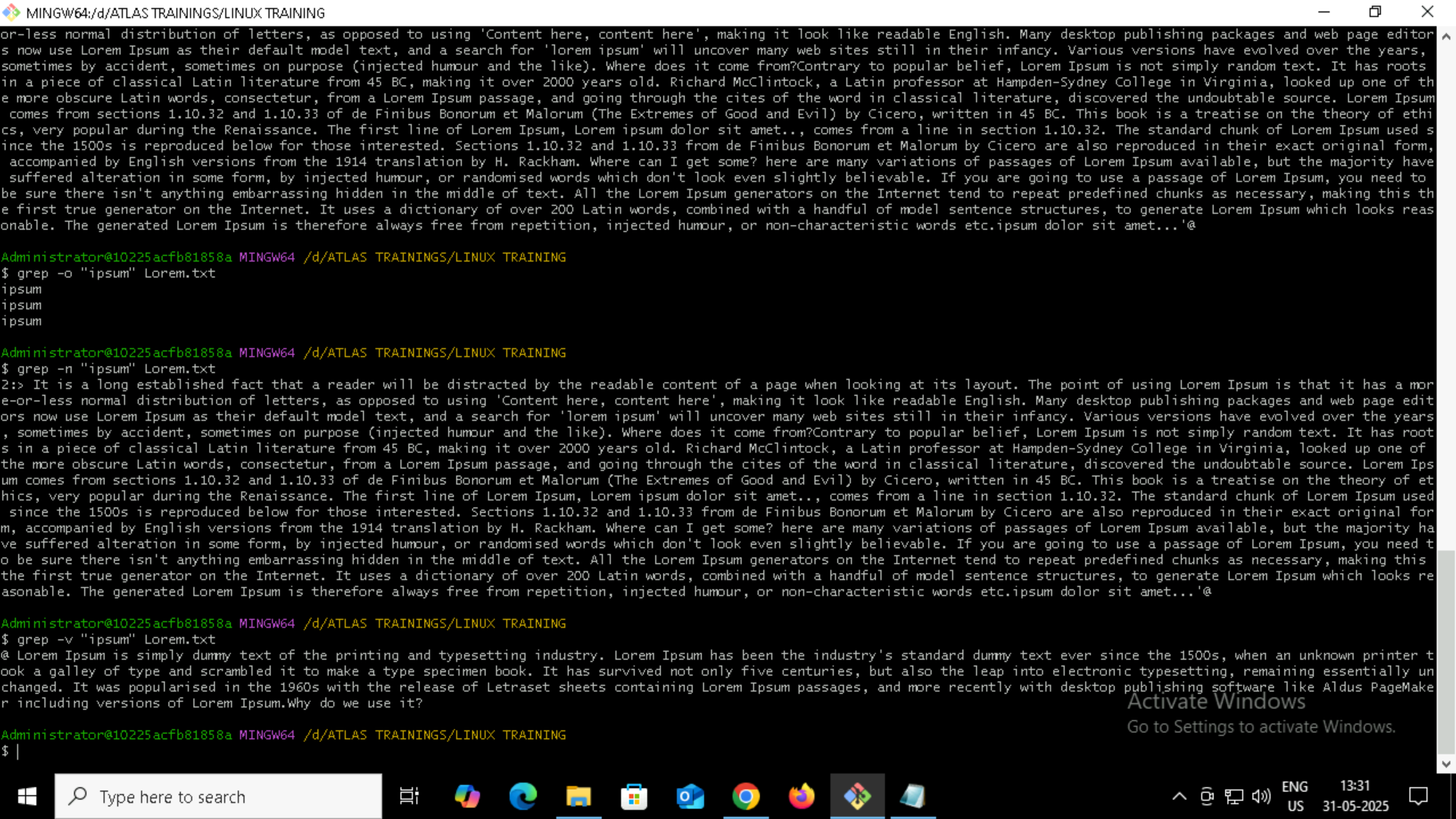
TASK 16: Grep -i 

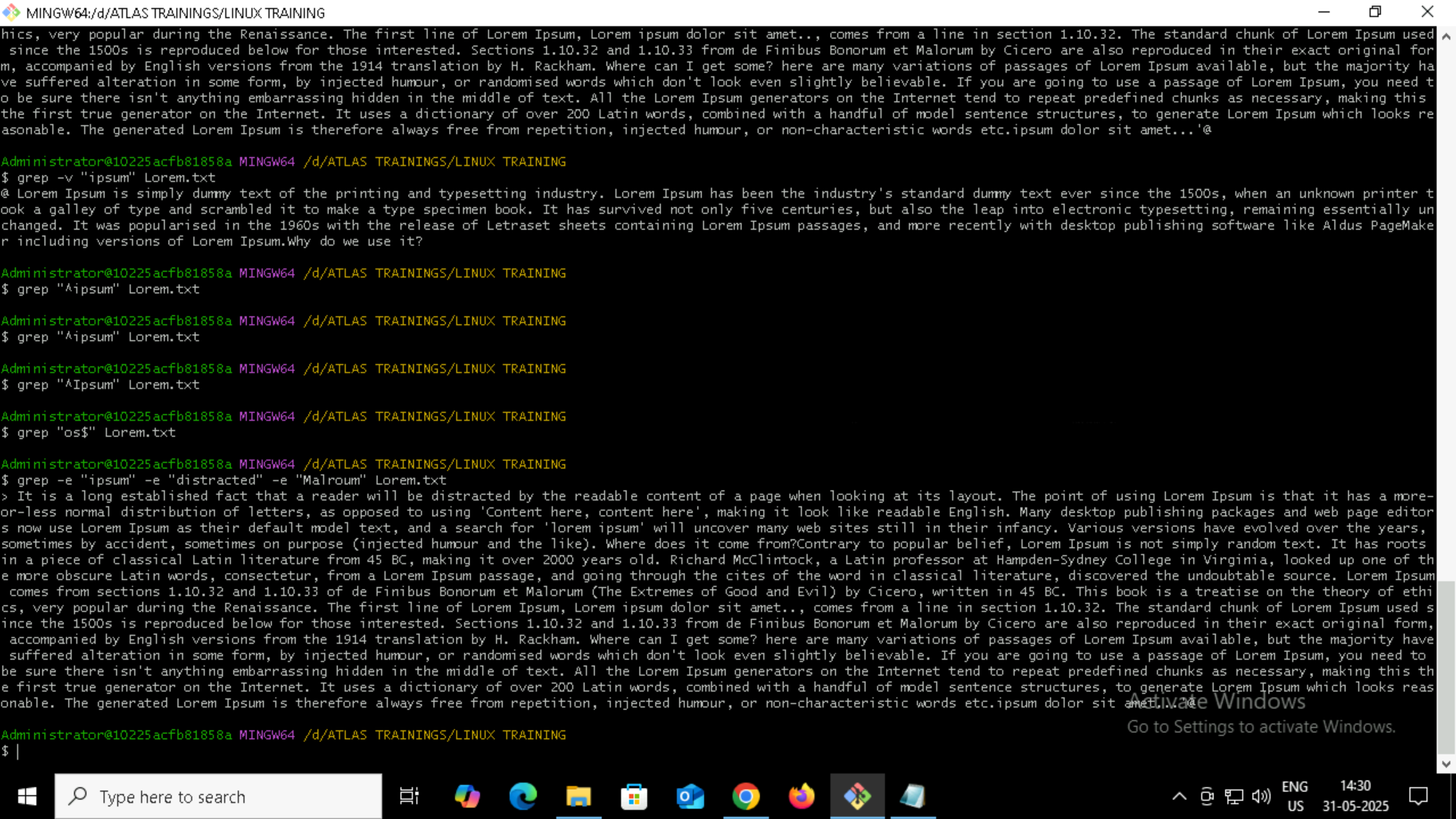
Count of no of matches grep -c: 

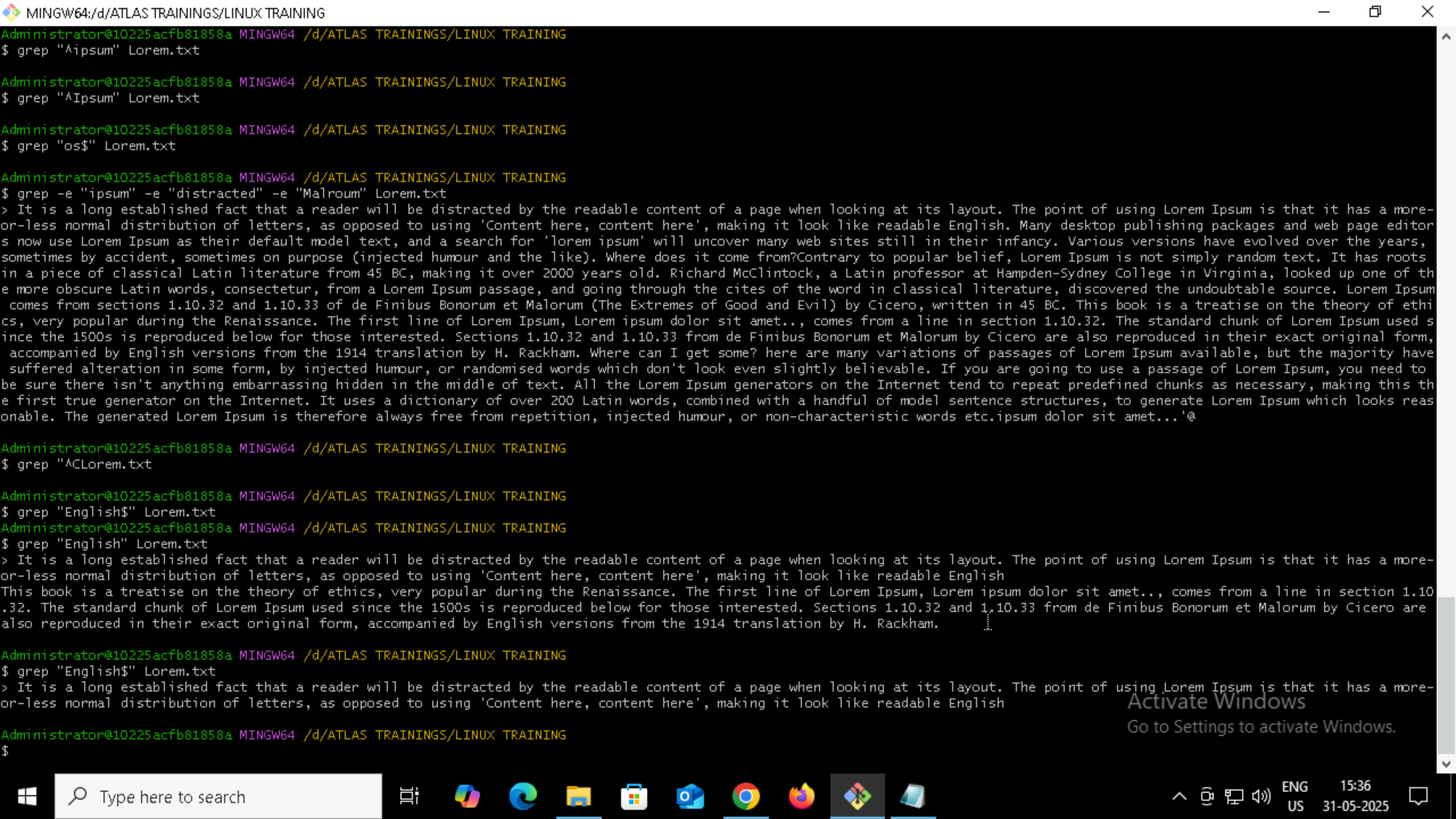
Checking for whole word grep -w: 

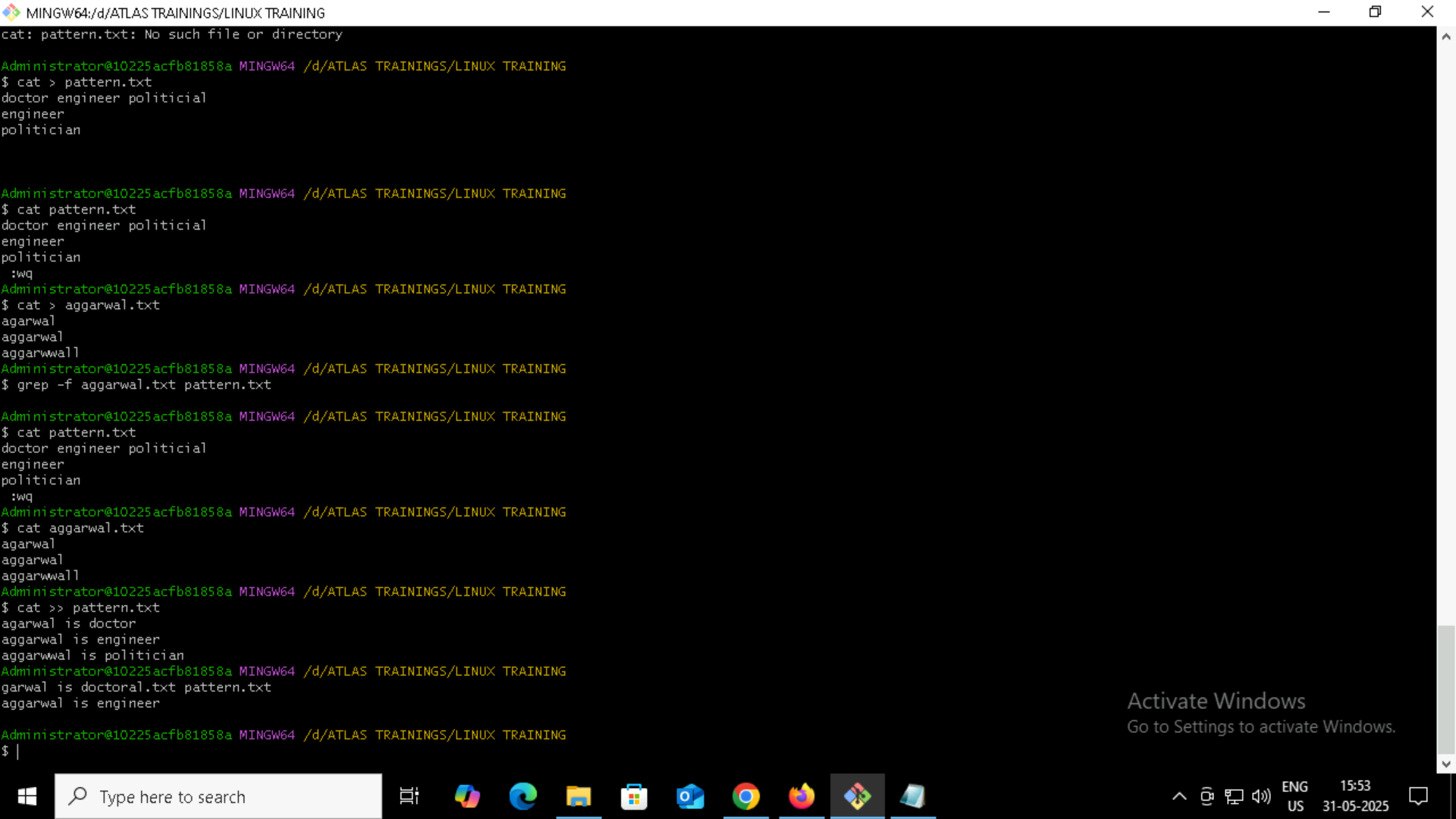
Only match pattern grep -o: 

Show line no while displaying the output grep -n: 

Investing pattern match using grep grep -v: 

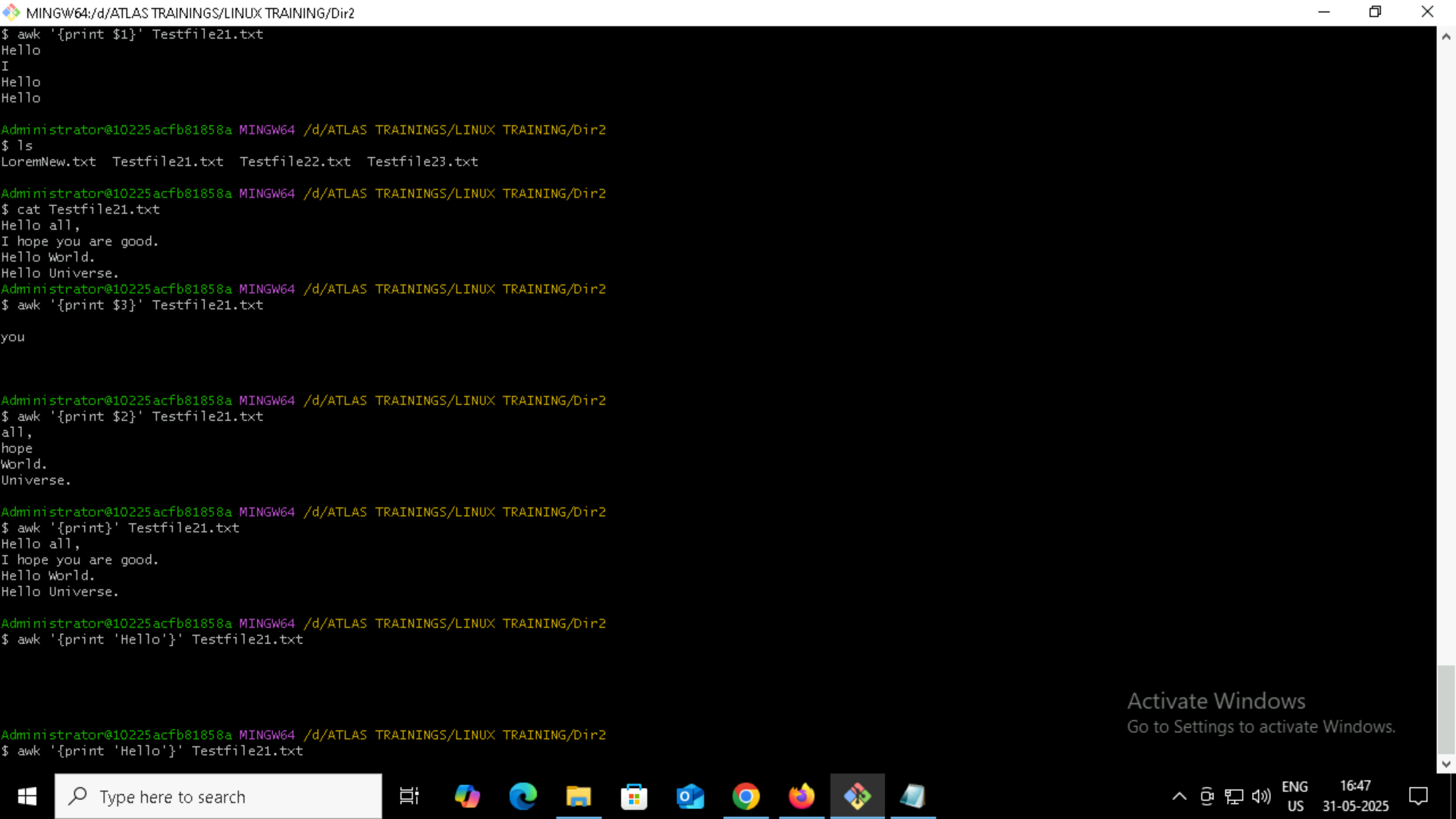
-e with multiple: 

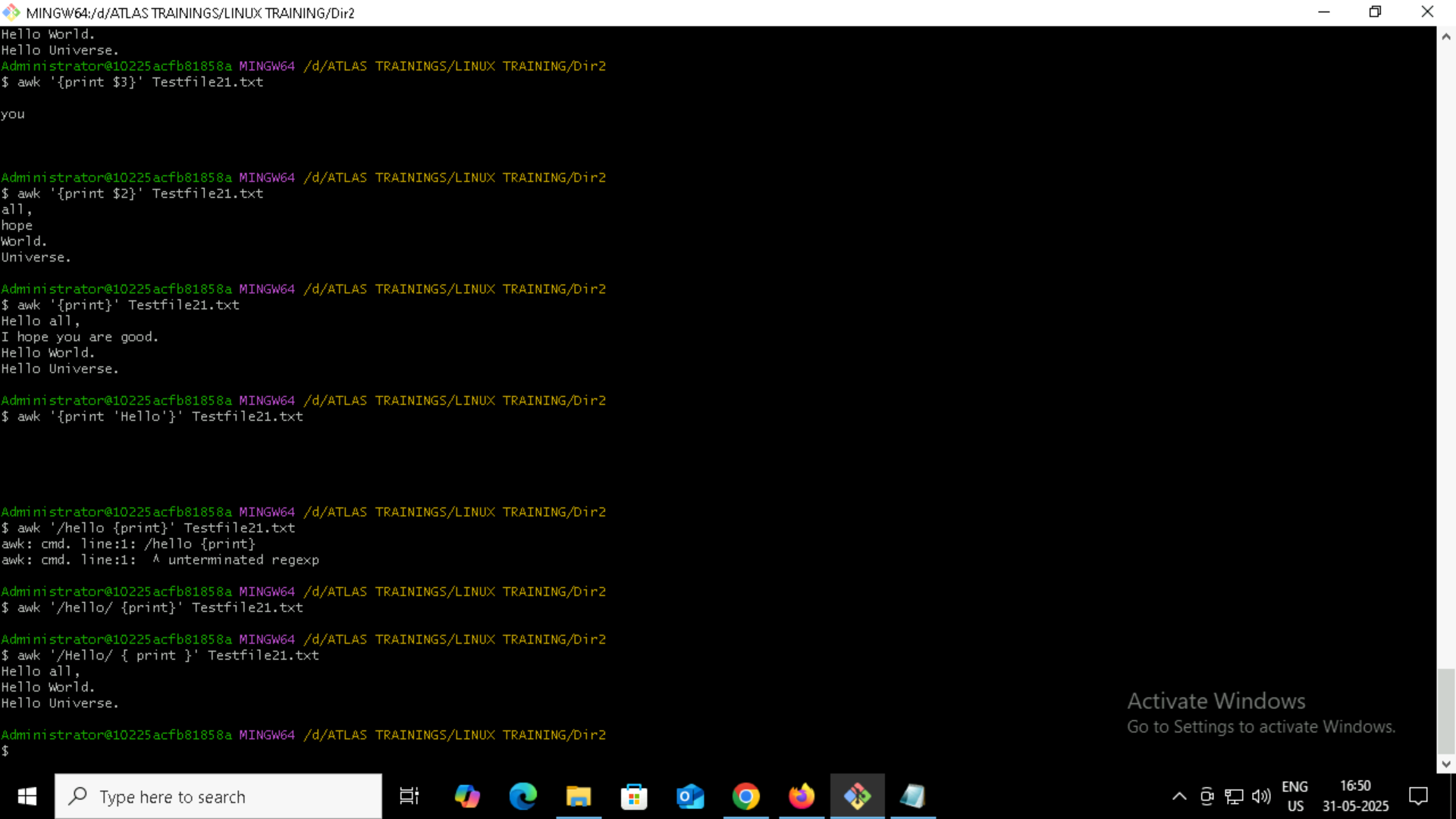
9. Line ending with : 

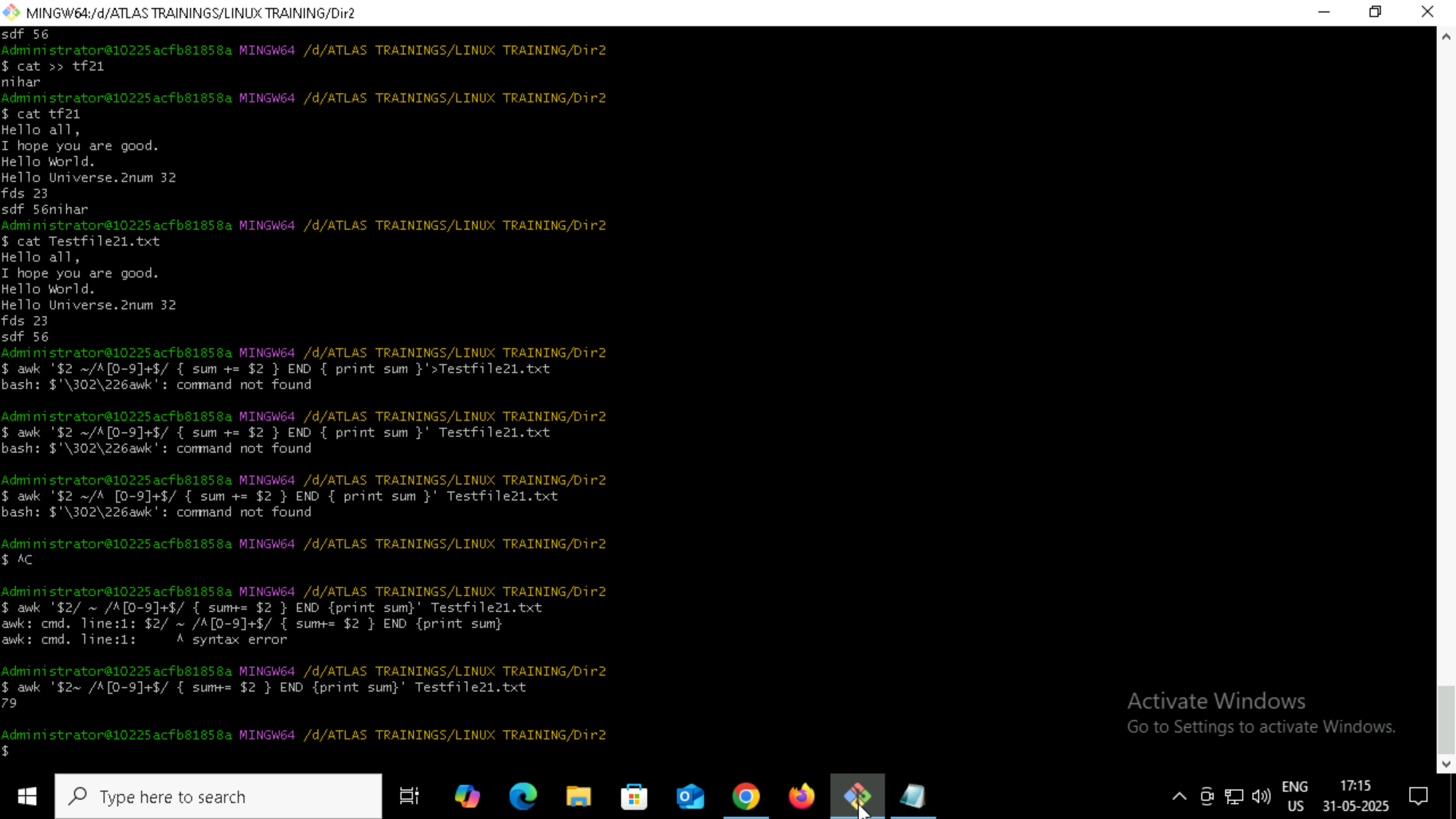
Grep -f: 

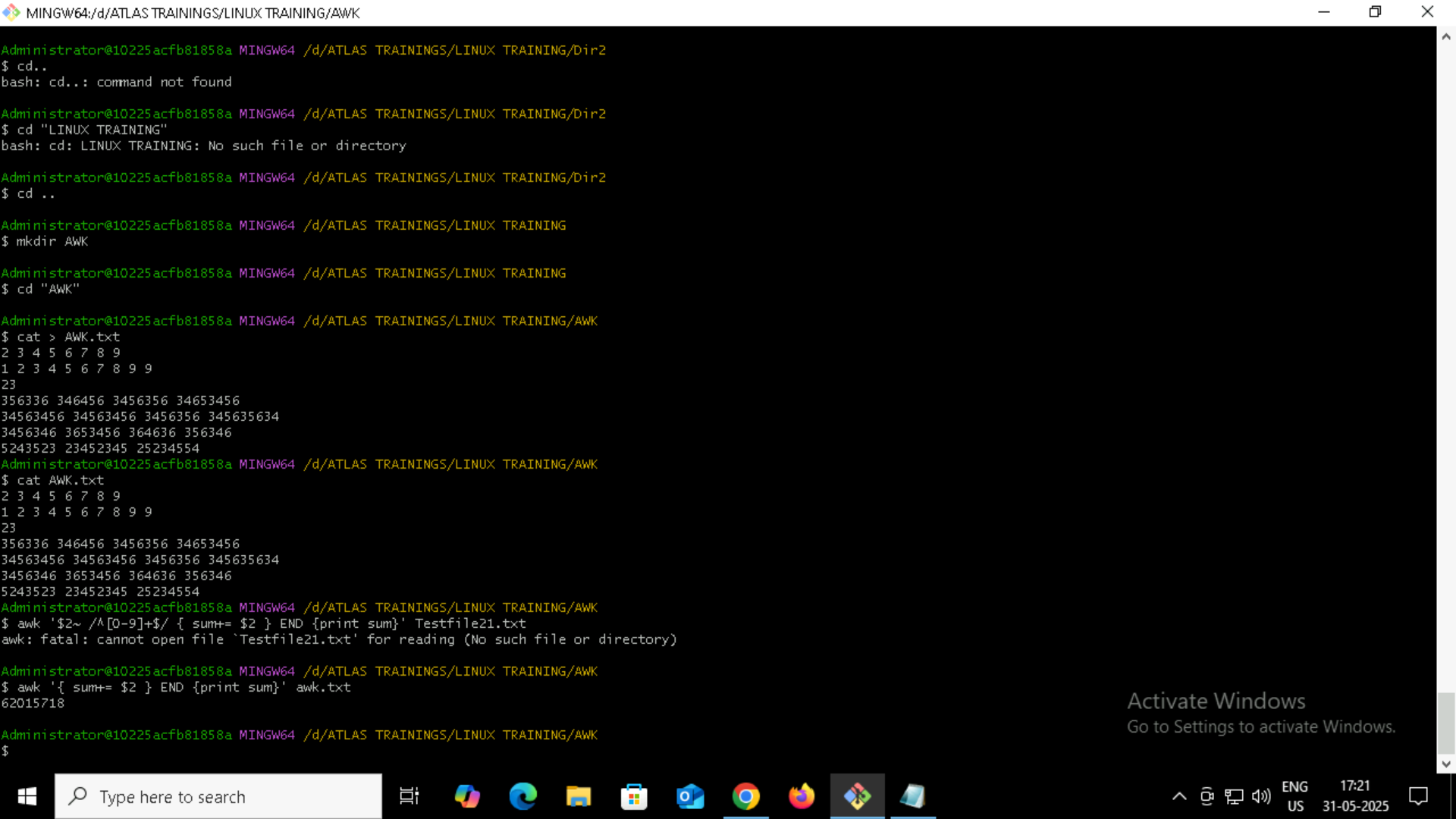
<https://docs.google.com/document/d/1UR3ZUlqKAt7hLXO2G_2f7MUu8GnRSbvPhik9nxODLeA/edit?tab=t.0>

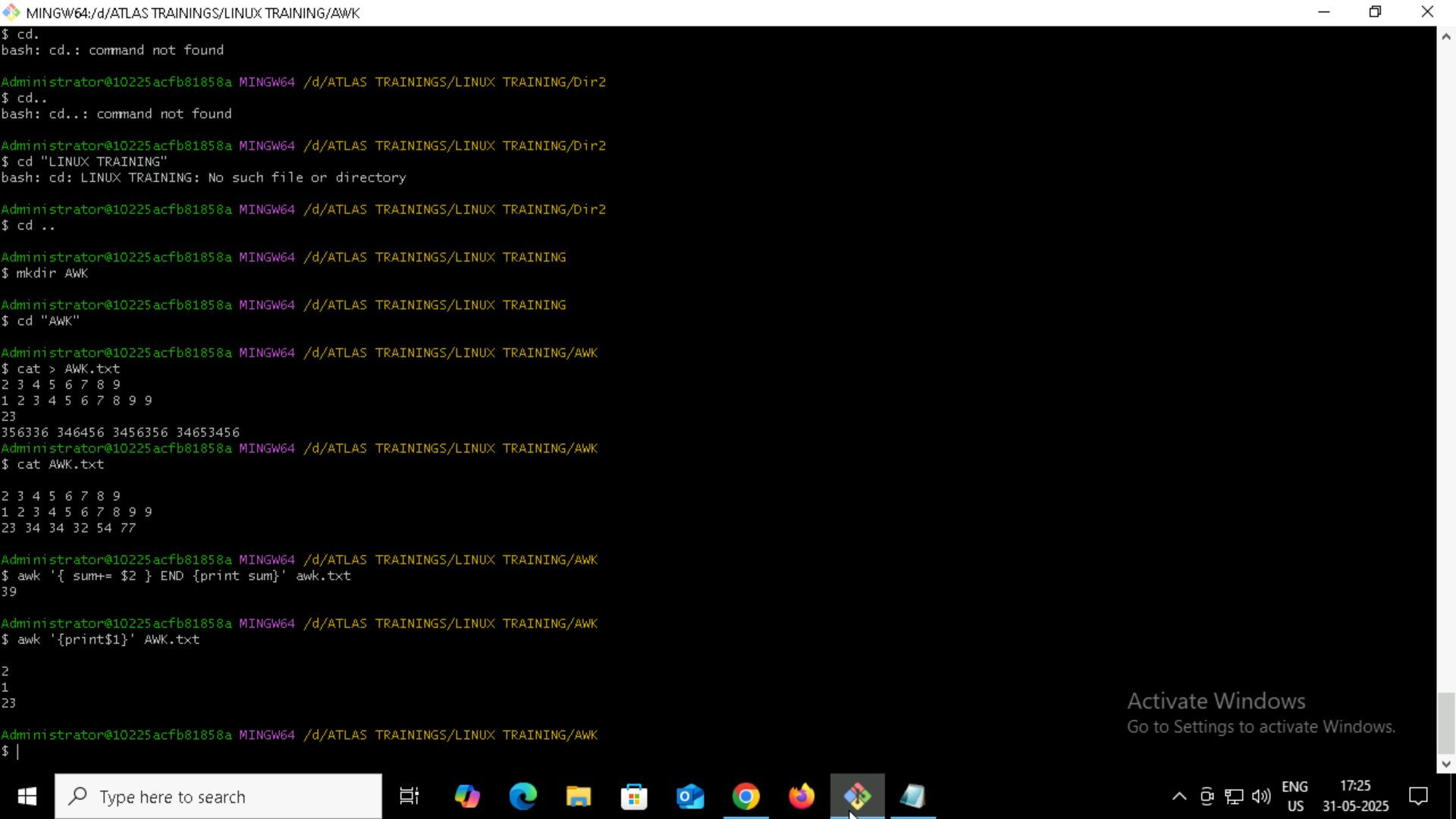
AWK Command:

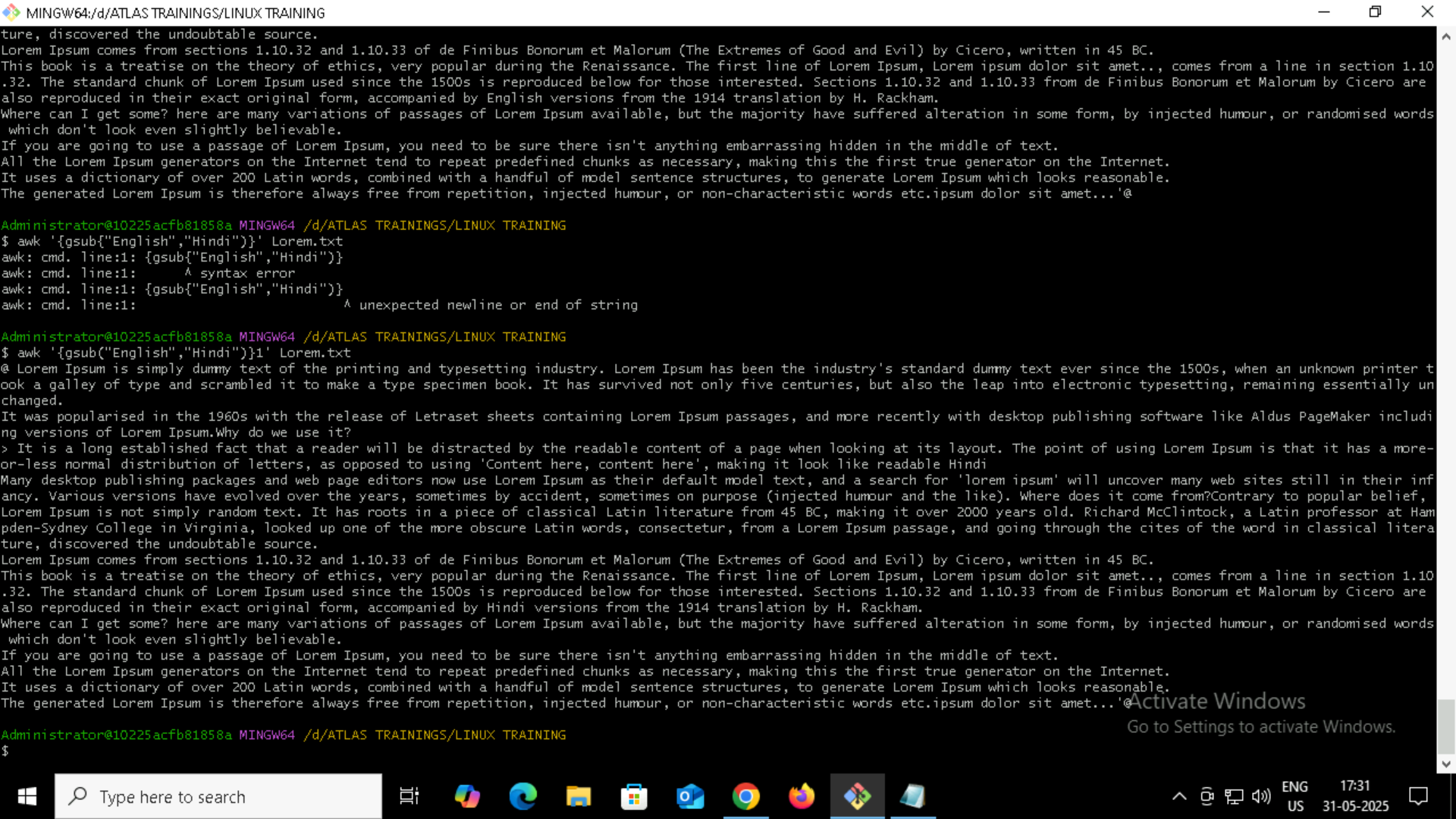




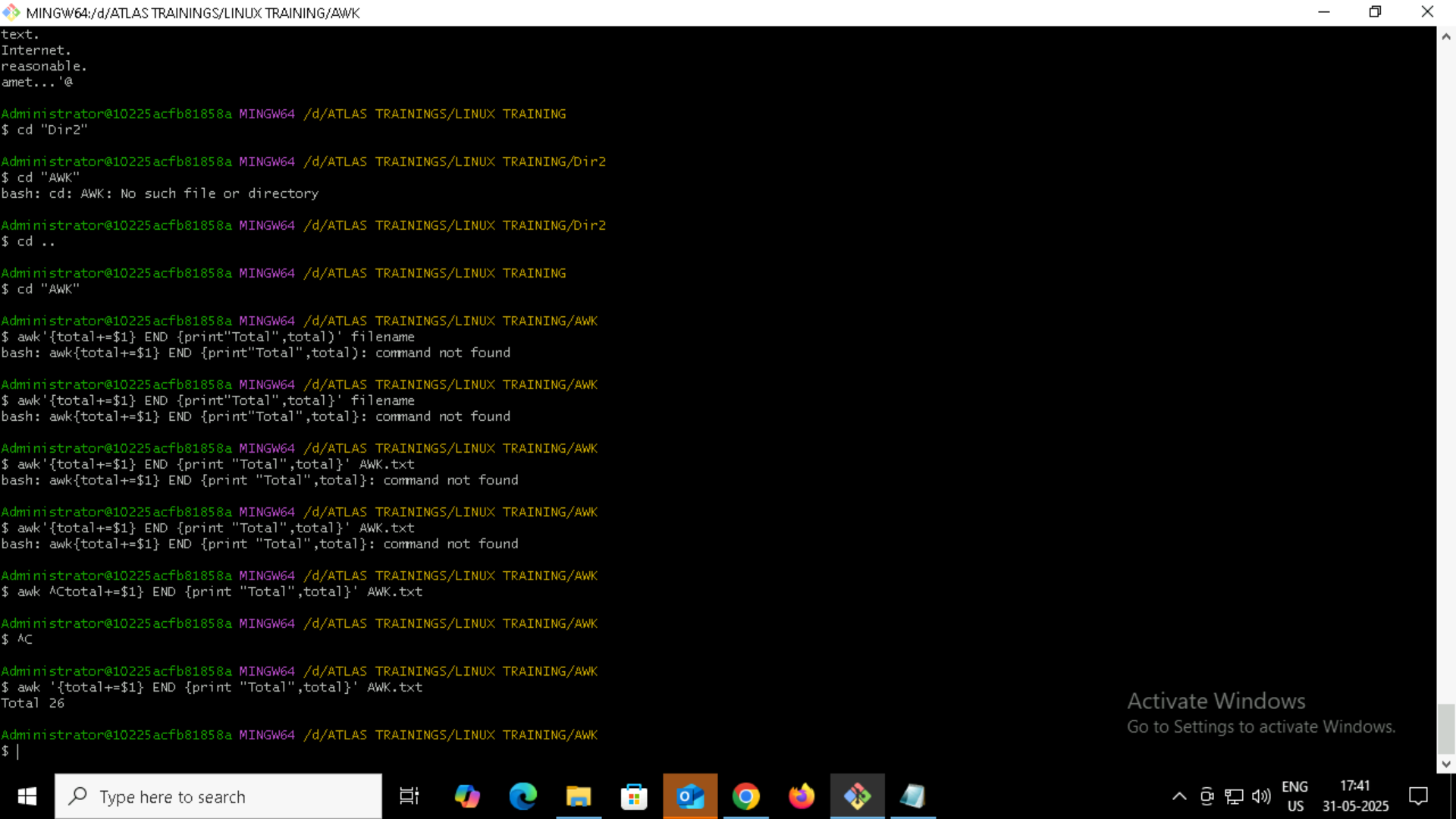


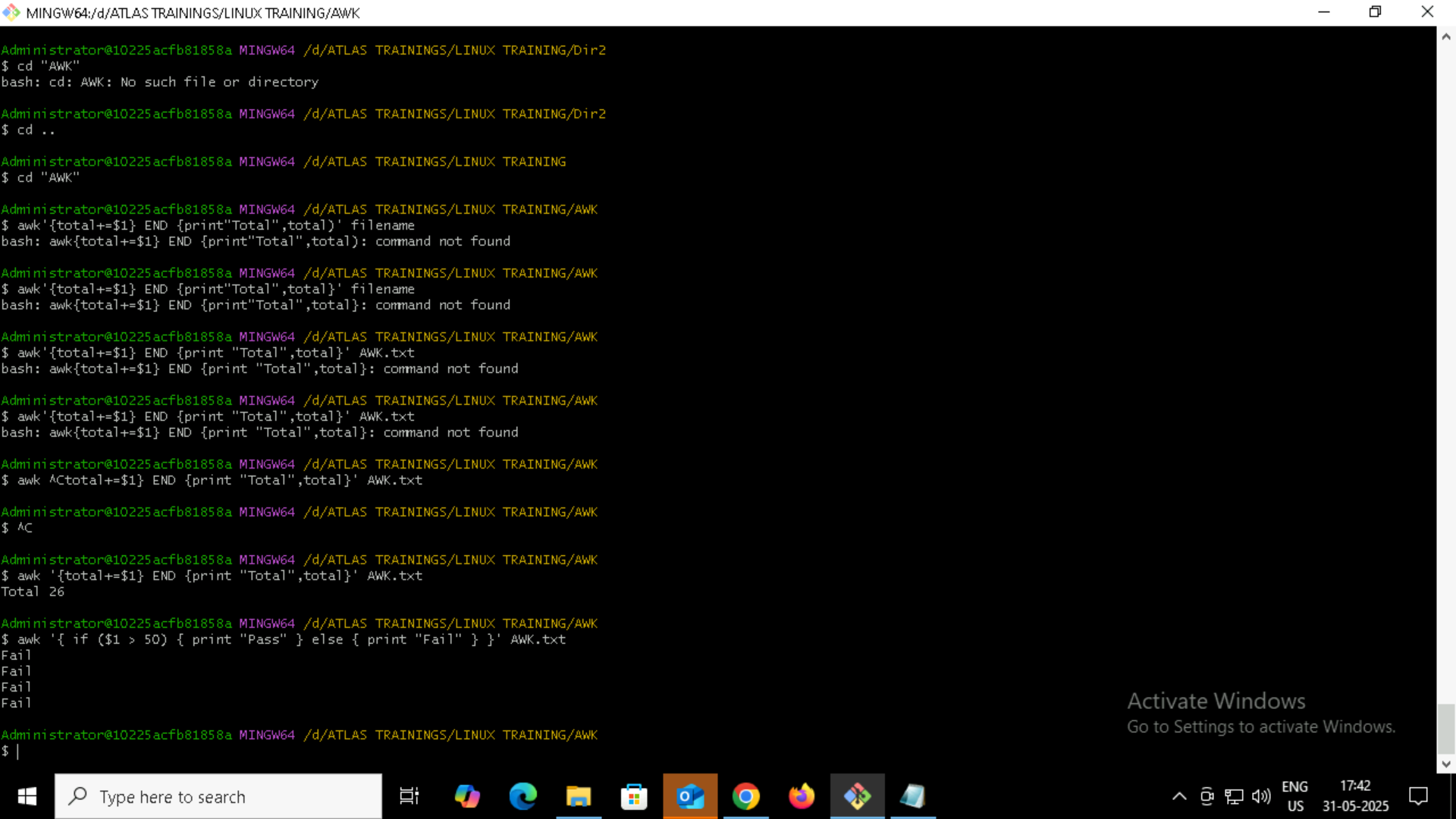


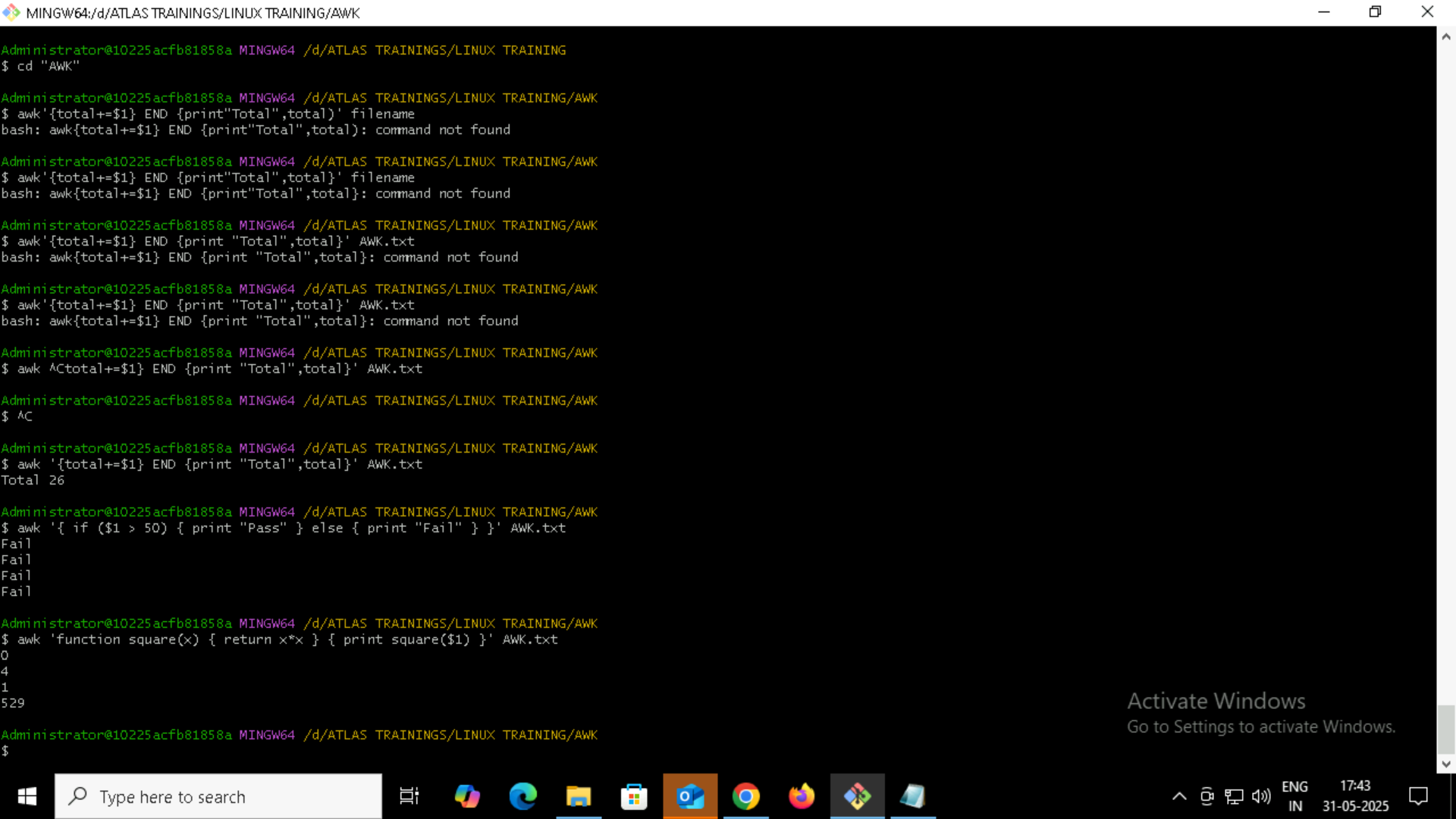


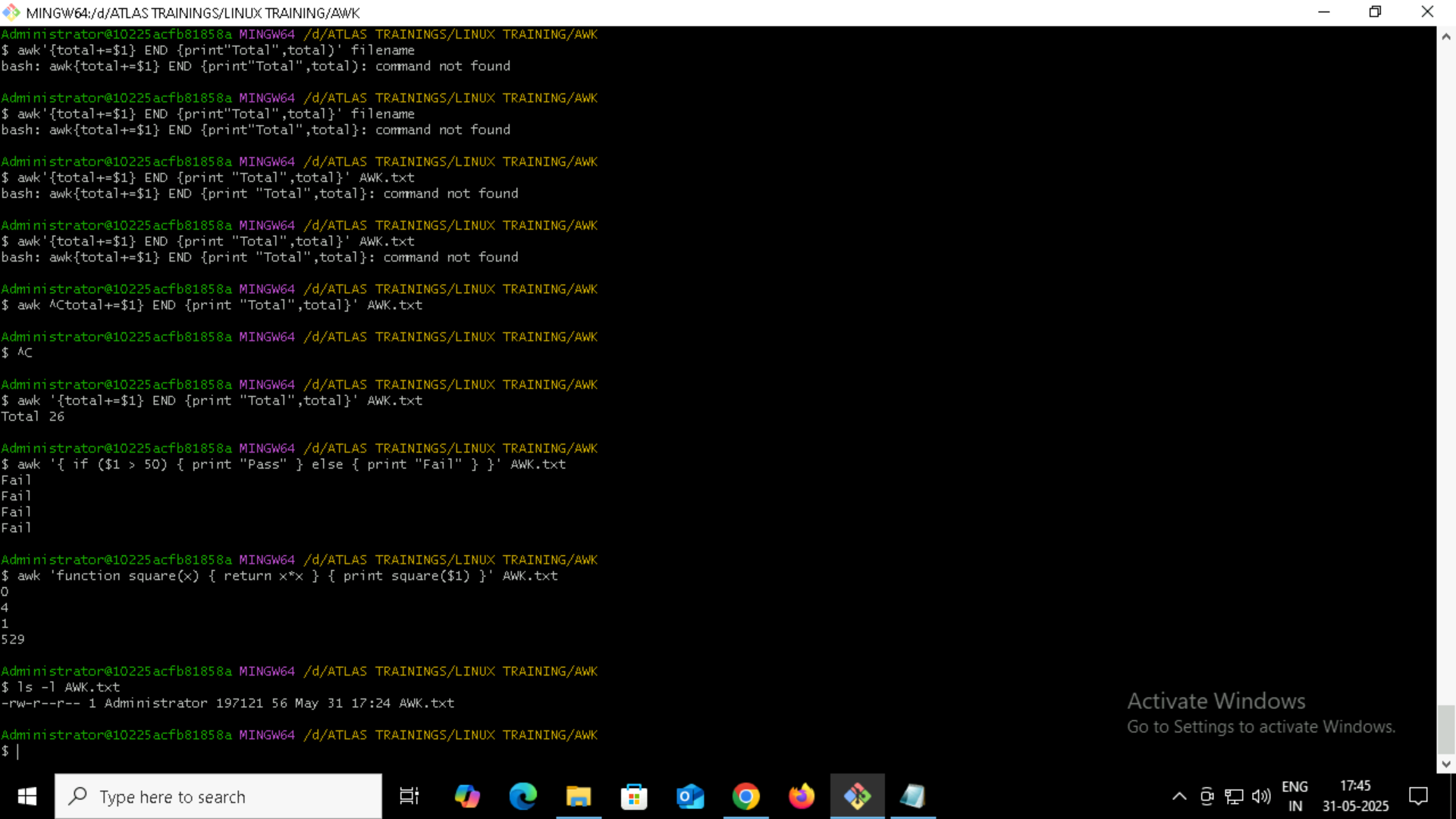






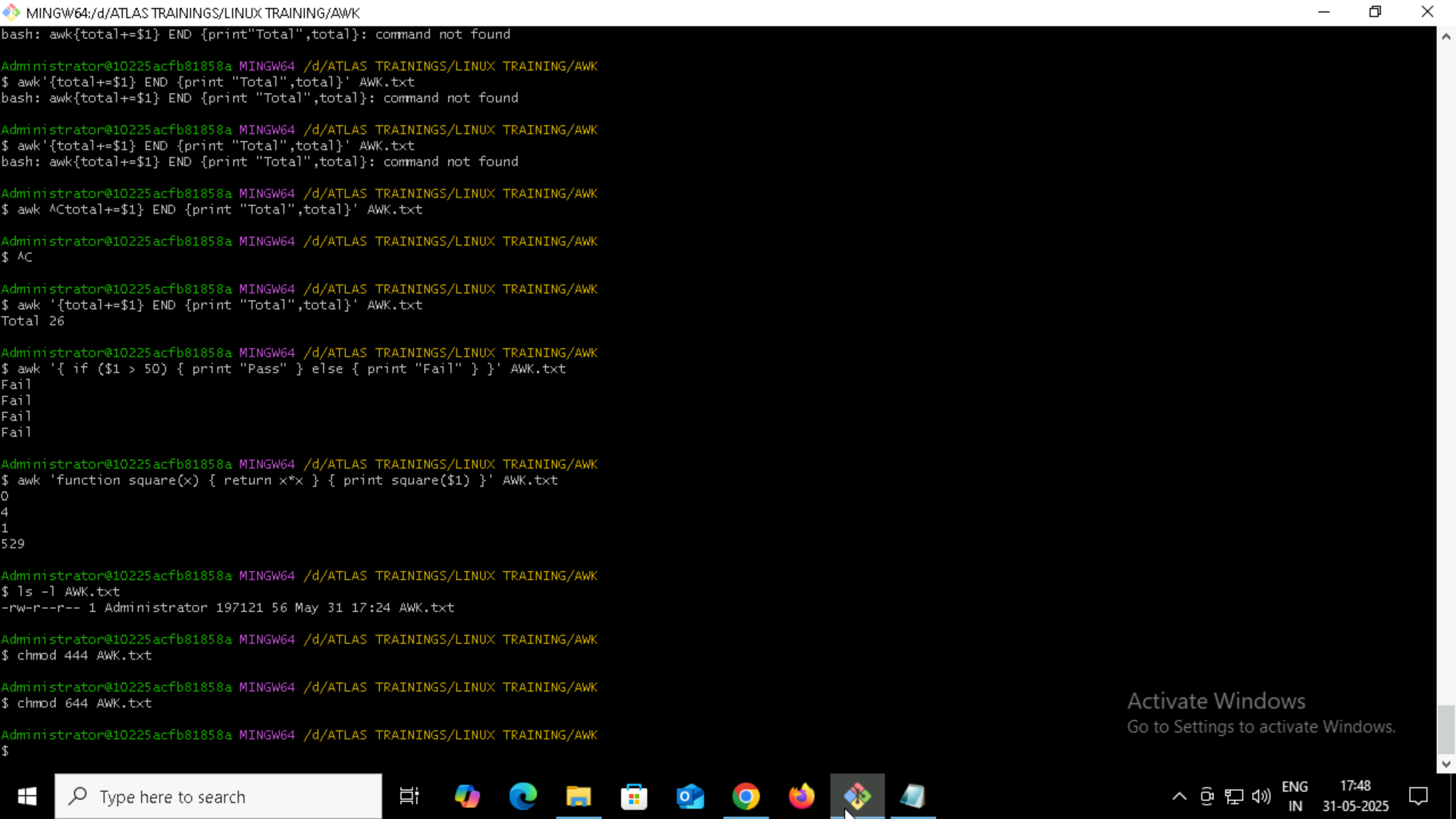




Check Permission ls -l: 

Default permissions:



Command to change file only to read only for owner group and all: 

**Can you change the file permissions to match the following:**

* owner: Read and Write (rw- )
* group: Read (r--)

other: no permissions (None)

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

Hint : use chmod 640 filename

ANS: Read ONLY

**Guys what will this command do?**

chown -c master file1.txt

changes the owner of file

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

Foreground process is the one that is connected to the terminal and interacting with the user. There isin’t a direct command to list the current foreground process because it’s usually the one you are actively running

However you can use: ps -o pid,tty,stat,cmd

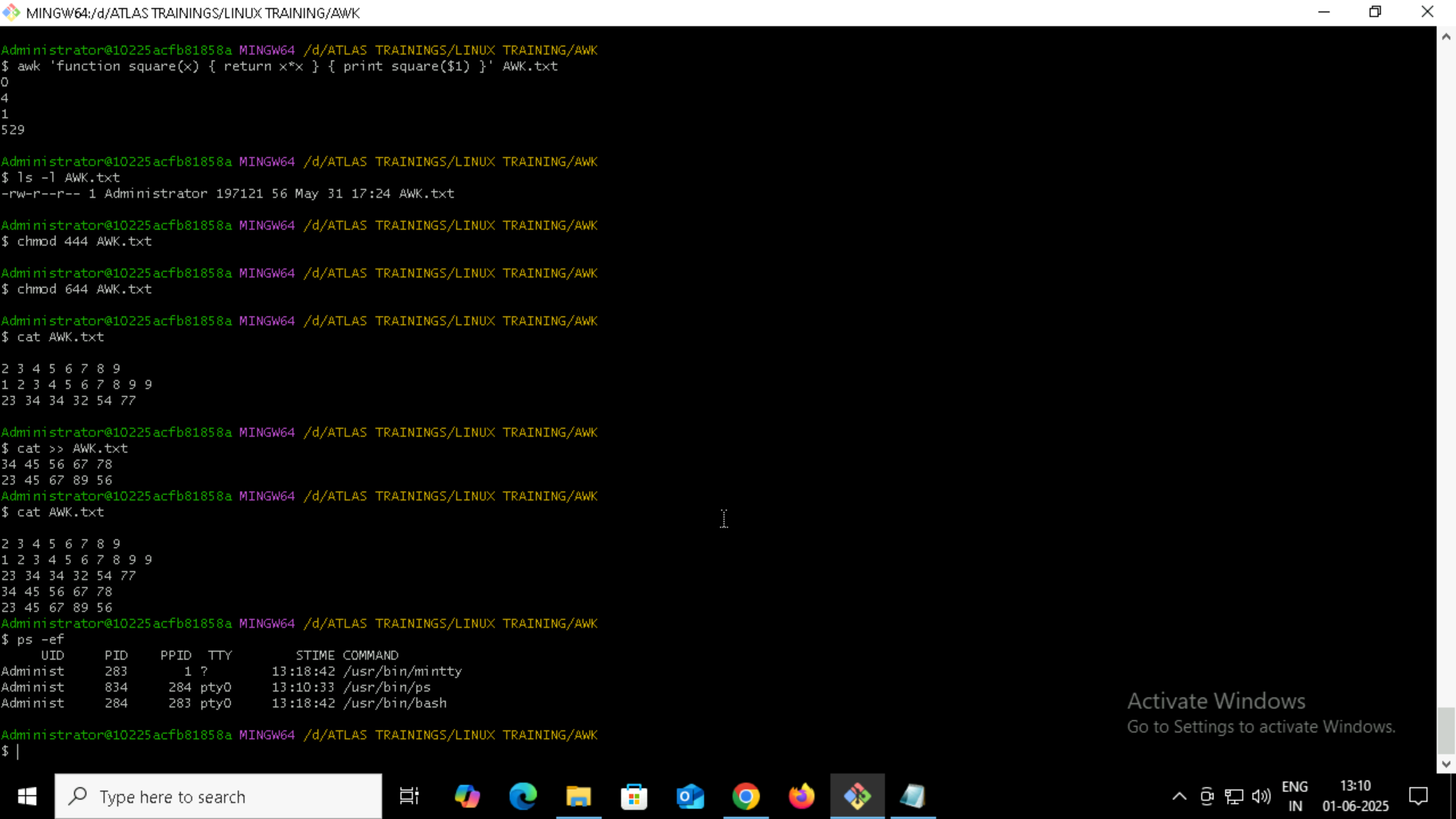
Bash

Ps -u $USER

Ps aux | grep your-username

Ps -eco pid,stat,cmd | grep ‘^.\*T’

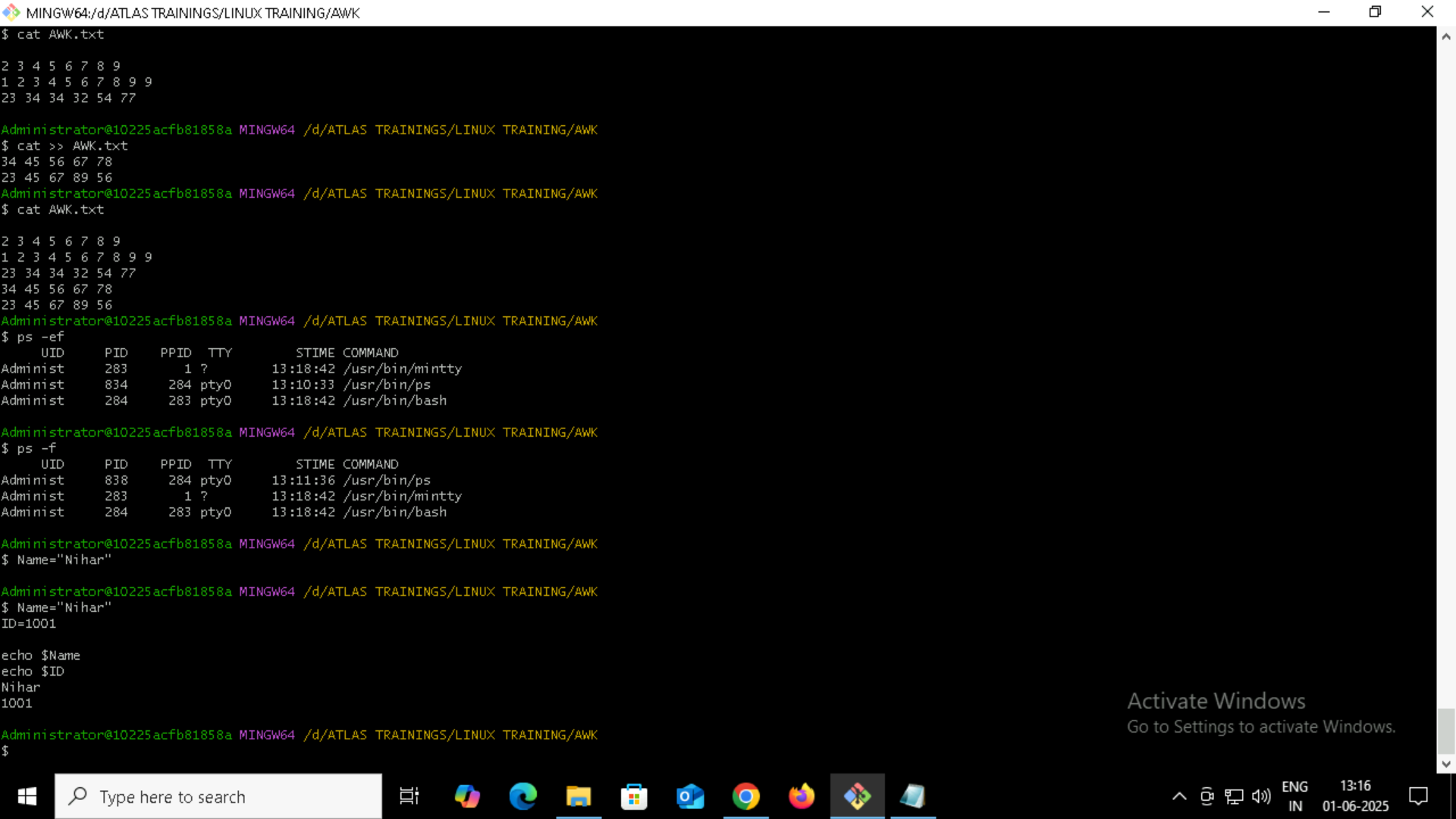
**Can you list the running process? Ps -ef**

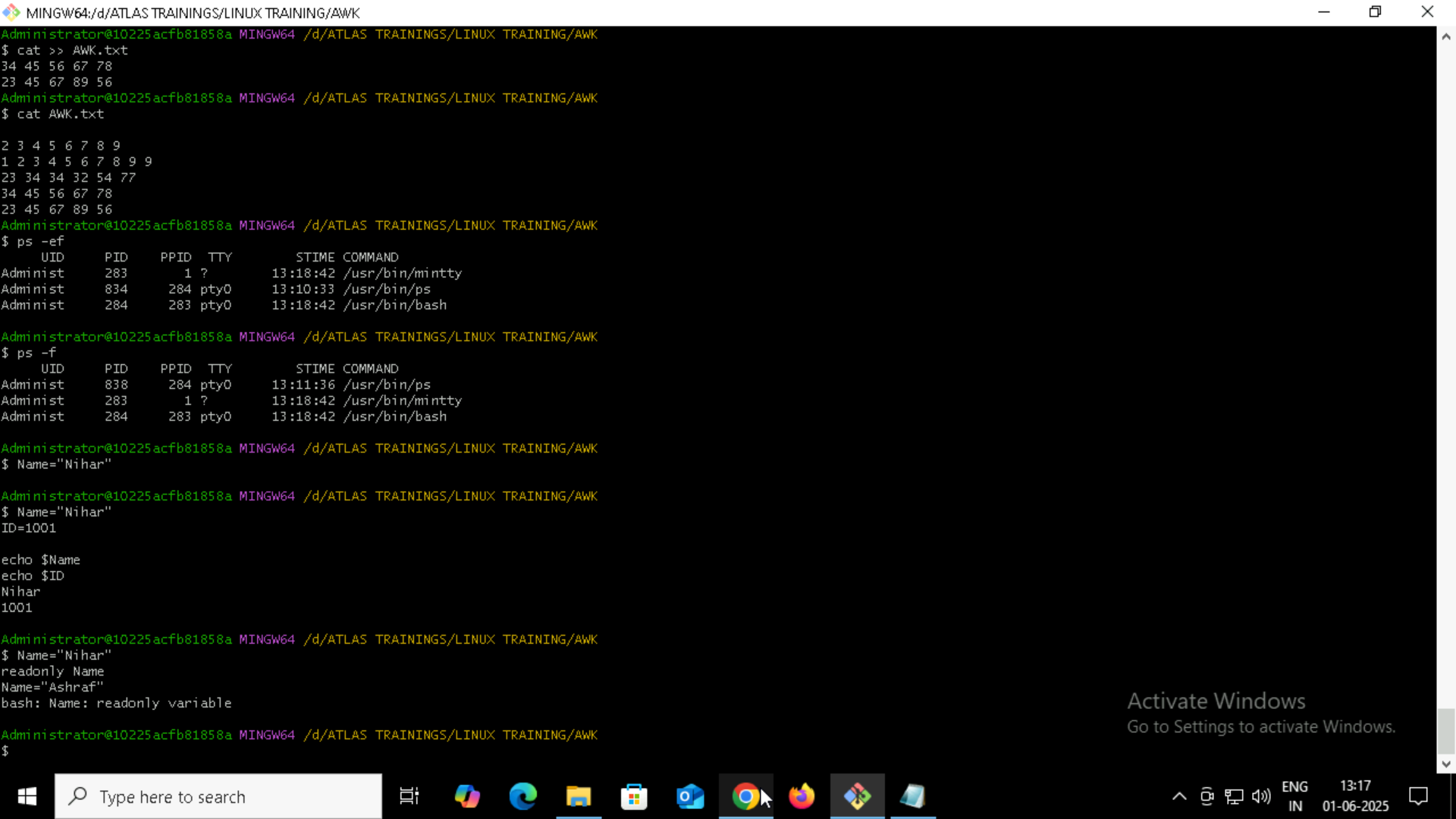


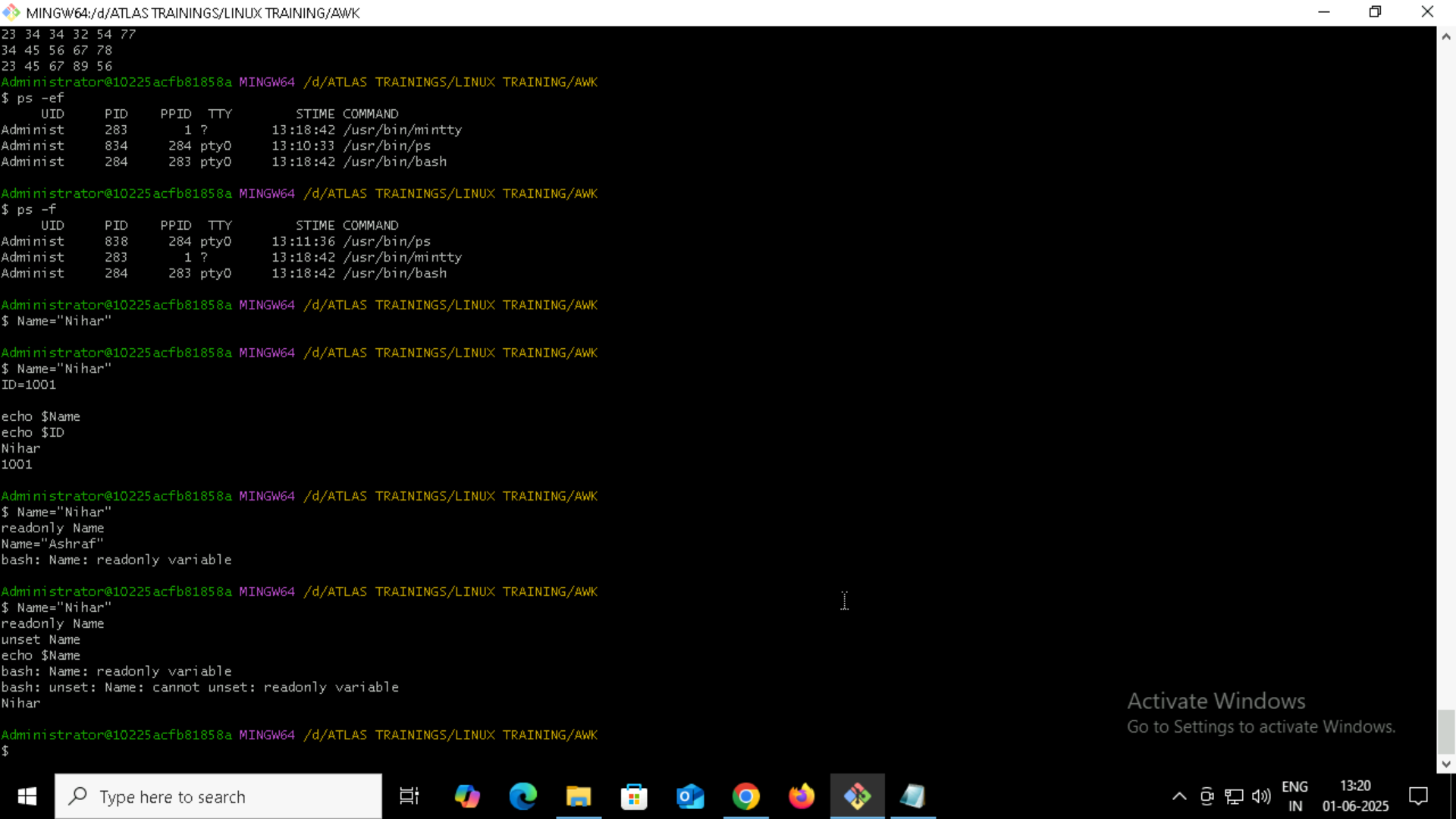
**Ps-f what does it do?**

Outputs the following colums:

UID, PID, PPID, C, STIME, TTY, TIME, CMD

TASK 30: 

Task 31: 

Task 31: Checking if we can unset read only variables. ANS: We can not unset read only Variables. 

Creating Shortcut: