# **Day 6 - 31 May 2025**

**Document Name:**Day 6 - hmuvvala@ - Hari Gopal Muvvala

**Task 1: RegEx Symbols in Linux**

Regular expressions (RegEx) are used to search for patterns in text.

1. . – Matches any single character.  
   Example: a.c → matches abc, a1c, etc.
2. ^ – Matches the start of a line.  
   Example: ^Hari → matches lines starting with “Hari”.
3. $ – Matches the end of a line.  
   Example: end$ → matches lines ending with “end”.
4. \* – Matches 0 or more of the previous character.  
   Example: lo\* → matches l, lo, loo.
5. + – Matches 1 or more of the previous character.  
   Example: lo+ → matches lo, loo.
6. ? – Matches 0 or 1 of the previous character.  
   Example: colou?r → matches color or colour.
7. [abc] – Matches any one character inside brackets.  
   Example: matches a, b, or c.
8. [^abc] – Matches any character not in brackets.  
   Example: [^0-9] → matches non-digits.
9. {n} – Matches exactly n times.  
   Example: a{3} → matches aaa.
10. {n,m} – Matches between n and m times.  
    Example: a{2,4} → matches aa, aaa, aaaa.
11. \d – Matches any digit.  
    Example: \d\d → matches 42, 99.
12. \w – Matches any letter, digit, or underscore.  
    Example: \w+ → matches Hari\_123.
13. \s – Matches whitespace (space, tab, newline).
14. | – Matches either of two patterns.  
    Example: cat|dog → matches cat or dog.

Examples:

* grep '^A' file.txt → lines starting with A
* grep '[0-9]' file.txt → lines with digits
* grep -E 'go+gle' file.txt → matches gogle, google, gooogle

**Task 2: Features of Linux OS**

1. Open-source and free to use.
2. Multi-user and multitasking support.
3. Strong security with permissions and roles.
4. Stable and reliable – used in servers.
5. Command Line and GUI both supported.
6. Can run on many types of hardware.
7. Highly customizable.
8. Active community support.
9. Supports multiple file systems.
10. Easy software management with package managers.

**Task 3: What is Kernel?**

* The kernel is the core of the Linux OS.
* It manages system resources like memory, CPU, and devices.
* Acts as a link between user programs and hardware.
* Without the kernel, the OS can’t function.

**Task 4: BASH – Full Form and Explanation**

* Full form: Bourne Again SHell
* BASH is the default shell in most Linux systems.
* Used to run commands and navigate through the system.
* Also supports writing small scripts to automate tasks.

**Task 5: Difference between Linux and Windows**

* **Linux** is open-source, while **Windows** is proprietary (paid license).
* Linux is mostly command-line-based; Windows is GUI-based.
* Linux is more secure and used in servers; Windows is more user-friendly and used in desktops.
* Linux supports many distributions (Ubuntu, Fedora, etc.); Windows has fixed versions.
* Linux allows full control and customization; Windows is limited in customization.
* Linux rarely needs reboots; Windows requires frequent updates and restarts.
* Software installation in Linux uses package managers; in Windows, it’s mostly manual installers (.exe/.msi).

**Task 6: Basic Components of Linux**

1. **Kernel** – Core part of the OS that controls everything (hardware, memory, processes).
2. **Shell** – Interface between the user and the kernel (like BASH).
3. **File System** – Organizes files and directories (like ext4, xfs, etc.).
4. **System Libraries** – Prewritten functions that programs use.
5. **System Utilities** – Basic tools and commands used by users (ls, cp, mkdir, etc.).
6. **Applications** – Software installed by the user (editors, browsers, etc.).

**Task 7: Is it legal to edit Kernel? When do we need to?**

* Yes, it is legal to edit the Linux kernel because it is open-source under the GNU General Public License (GPL).
* People edit the kernel to add new features, support new hardware, or optimize performance.
* Usually done by advanced users, developers, or contributors to Linux.
* Most normal users don’t need to edit the kernel unless working on kernel-level development or custom builds.

**Task 8: What is LILO?**

* **LILO** stands for **Linux Loader**.
* It is a bootloader used to load the Linux operating system into memory when the computer starts.
* It allows booting Linux or other operating systems (like Windows) from the same machine.
* LILO is older; newer systems use **GRUB** (GRand Unified Bootloader).

Task 9: What is Shell? How many shells are there and what are they?

* A shell is a program that takes commands from the user and sends them to the operating system to execute.
* It acts as a bridge between the user and the Linux kernel.
* Shells can be used to run commands, manage files, and automate tasks using scripts.

Types of Shells in Linux:

1. BASH (Bourne Again SHell) – Most commonly used default shell.
2. SH (Bourne Shell) – The original Unix shell.
3. KSH (Korn Shell) – A powerful scripting shell.
4. CSH (C Shell) – Syntax similar to the C programming language.
5. ZSH (Z Shell) – Feature-rich and user-friendly, supports themes and plugins.
6. Fish (Friendly Interactive SHell) – Modern shell with helpful suggestions and color output.

Task 10: What is Swap Space?

* Swap space is a portion of the hard drive used as virtual memory when the RAM is full.
* It helps the system continue running even when physical memory (RAM) is completely used.
* It is slower than RAM but useful to prevent crashes or freezing during high memory usage.

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

* Mounting means attaching a storage device (like a USB, hard disk, or partition) to a directory so that its contents can be accessed.
* Unmounting is the process of detaching the file system so it can be safely removed.

To mount a file system:

* Command: mount /dev/sdX /mnt/myfolder  
    
   (Here, /dev/sdX is the device and /mnt/myfolder is the mount point)

To unmount a file system:

* Command: umount /mnt/myfolder
* Mounting is essential for accessing external drives and additional storage in Linux.

**Task 12: What is chmod command? How to use it?**

chmod is used to change the permissions of files or folders in Linux.

Each file has 3 types of users:

* Owner
* Group
* Others

And 3 types of permissions:

* Read (r)
* Write (w)
* Execute (x)

Changing permissions controls who can open, modify, or run the file.

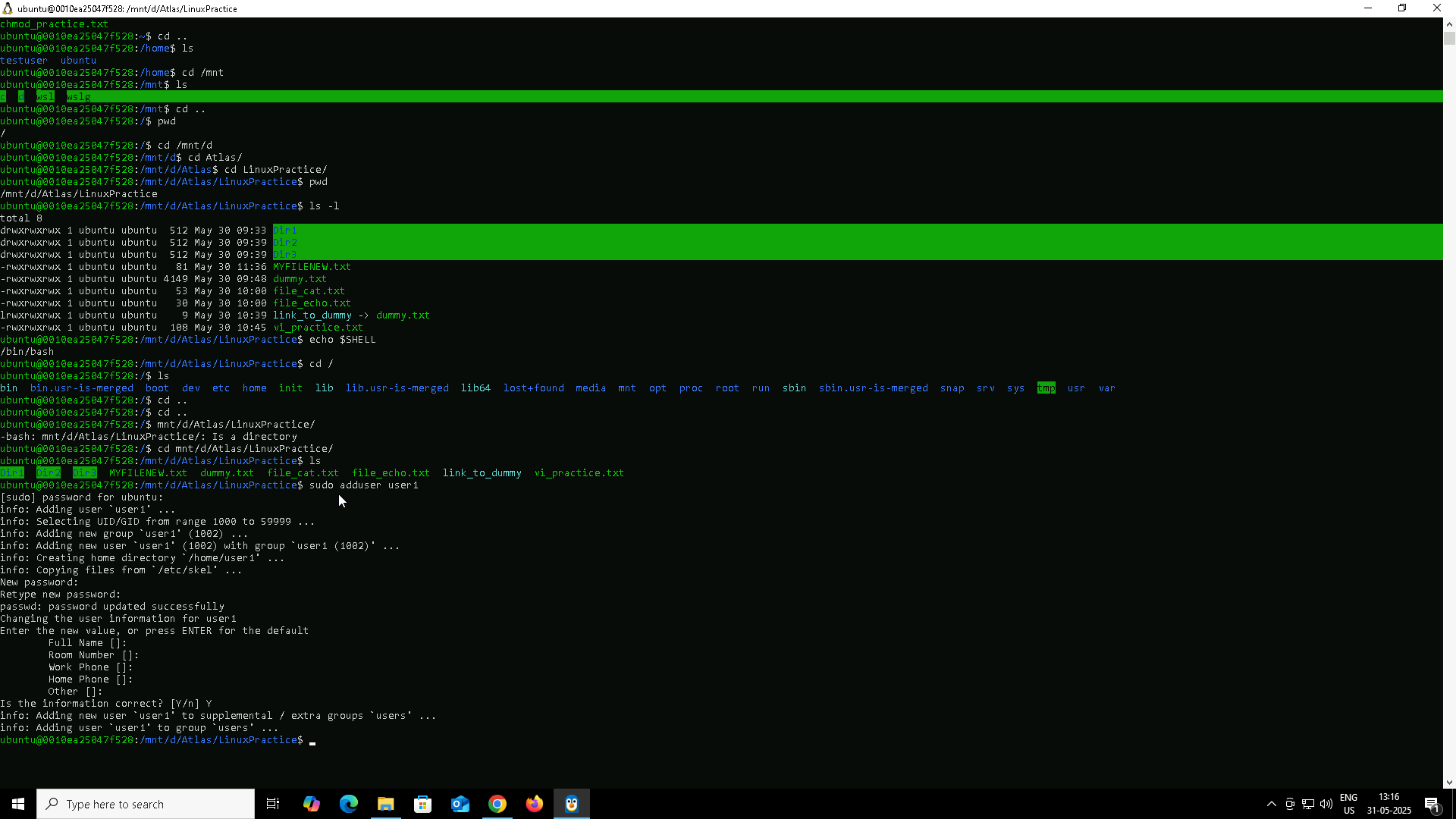
It’s like giving or removing keys to a file.

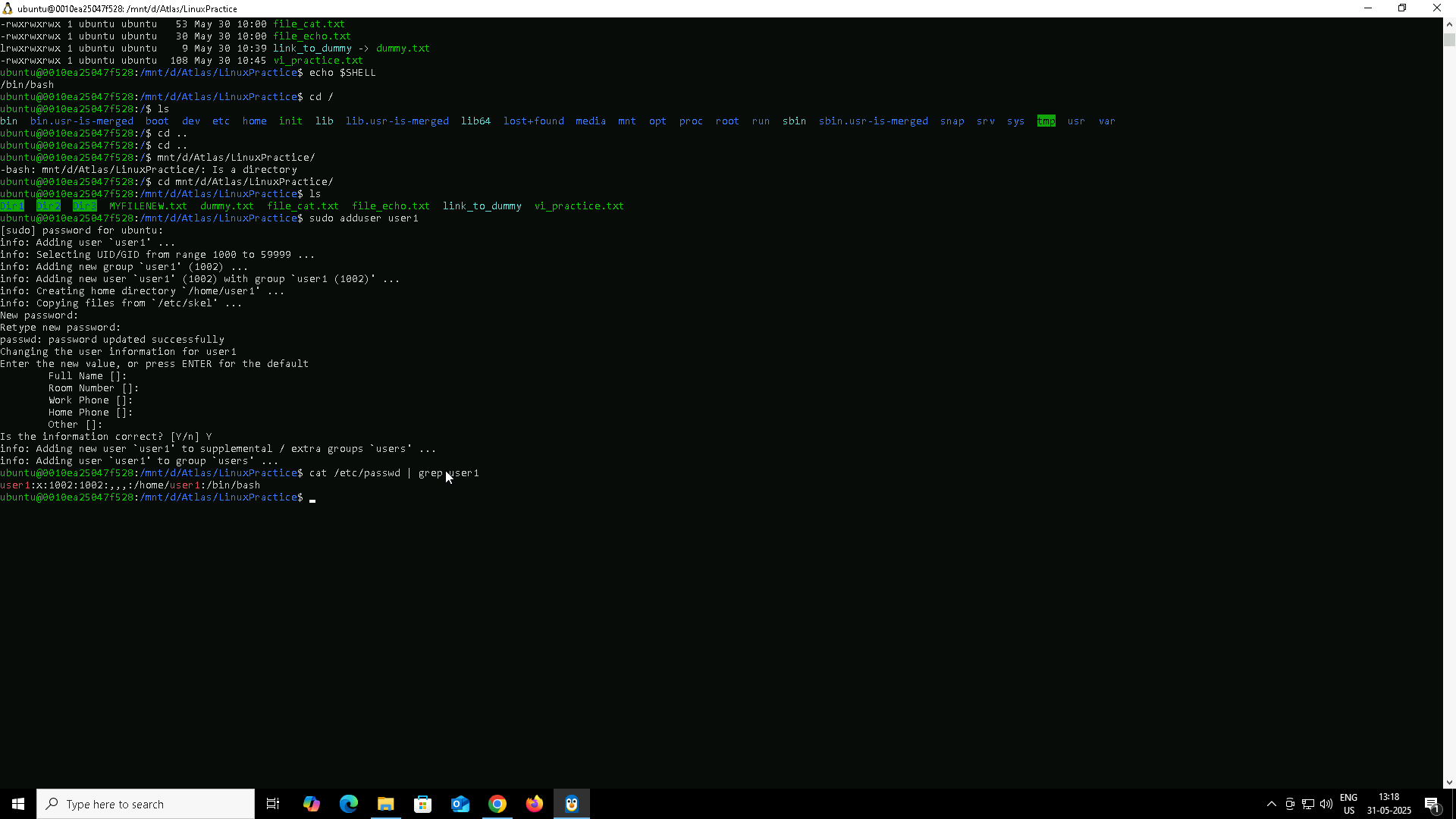
**Examples:**

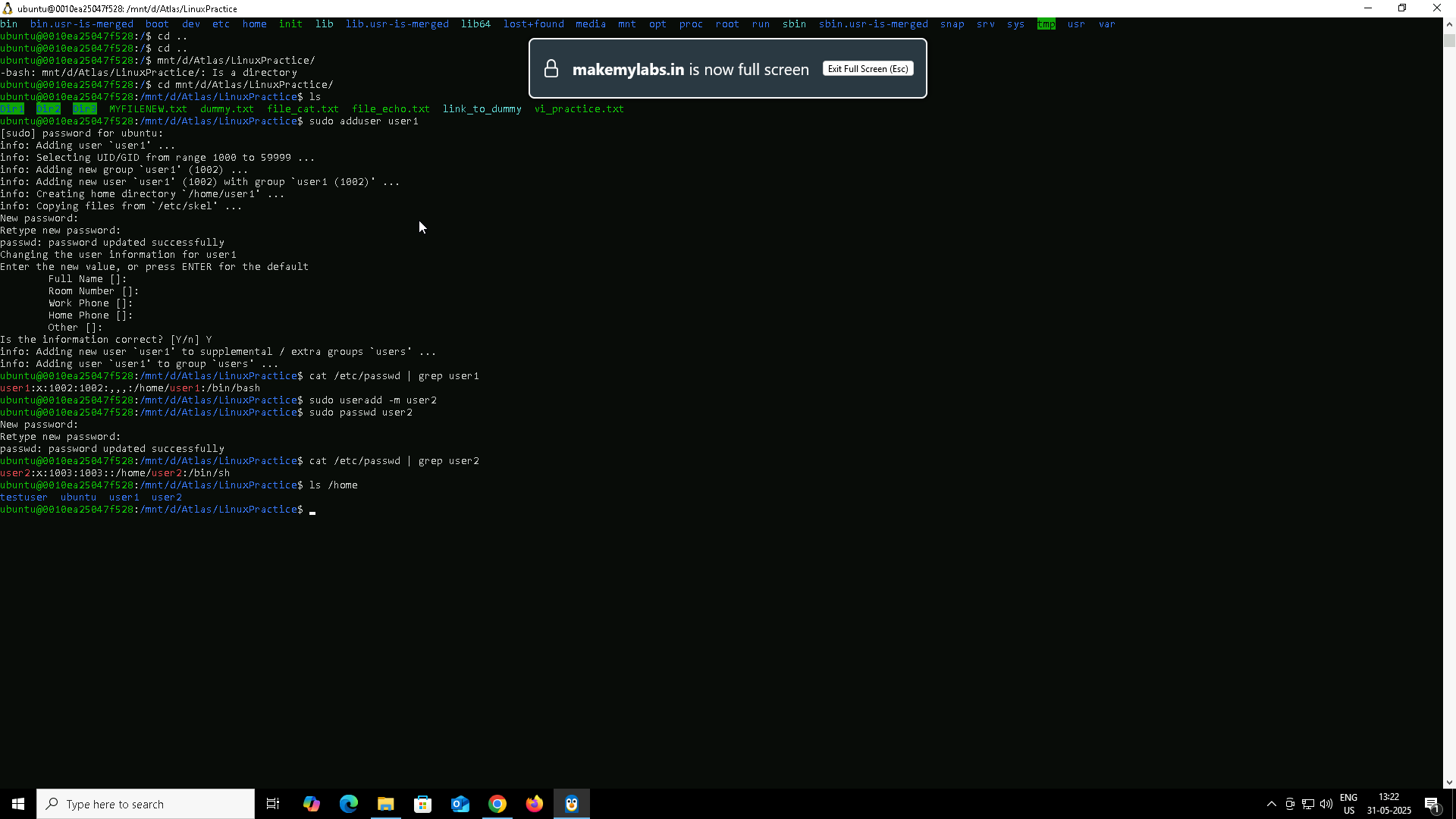
* chmod 777 file.txt → Full access to everyone (owner, group, others)
* chmod 755 file.txt → Owner: all permissions; Group & Others: only read and execute
* chmod +x script.sh → Adds execute permission

Task 13:

Adding a new user in different ways



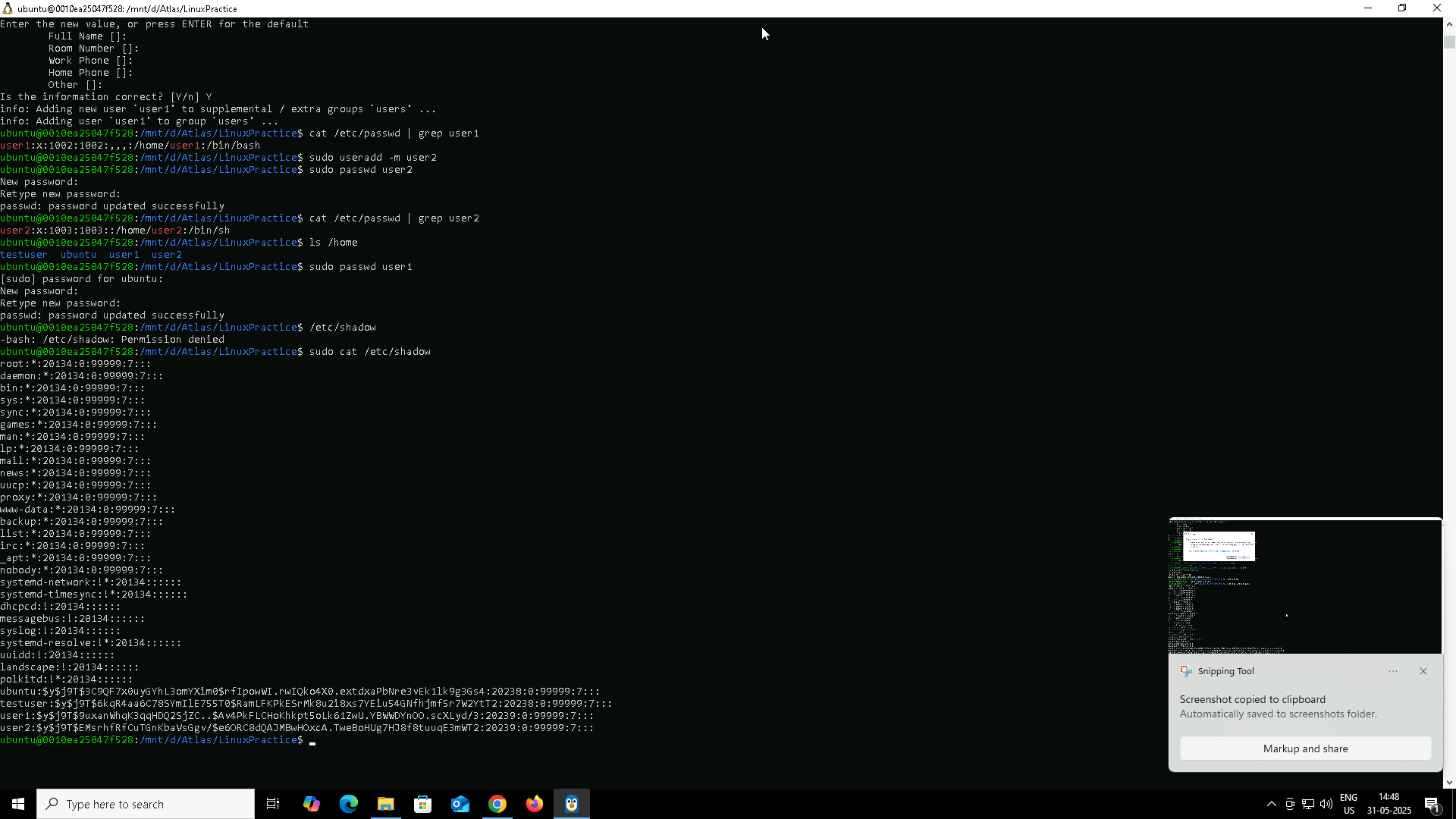




* adduser is a high-level command that’s easier and interactive.
* useradd is a low-level command, more manual, often used in scripting.
* adduser automatically creates a home directory and prompts for user details.
* useradd needs -m option to create the home directory.
* After useradd, I must run passwd separately to set a password.
* Both commands update /etc/passwd once the user is created.
* I can verify user creation using:  
  cat /etc/passwd | grep username
* I can check if the home directory is created using:  
  ls /home

Task 14:

Can you change the password of a user ?



* To change a user’s password, I use:  
  sudo passwd username
* It asks for the new password and confirms it by asking twice.
* If I don’t use sudo, and try to change another user’s password, it will be denied.
* Passwords are stored in encrypted format inside /etc/shadow (not in /etc/passwd).
* Only the **root user** has permission to access /etc/shadow.
* Trying to read /etc/shadow directly without sudo gives:  
  Permission denied
* I can verify password changes by switching to the user using:  
  su - username and logging in with the new password.
* I can also **force** a user to change password on next login using:  
  sudo chage -d 0 username

**Task 15: What is the difference between Process and Thread?**

From what I’ve understood:

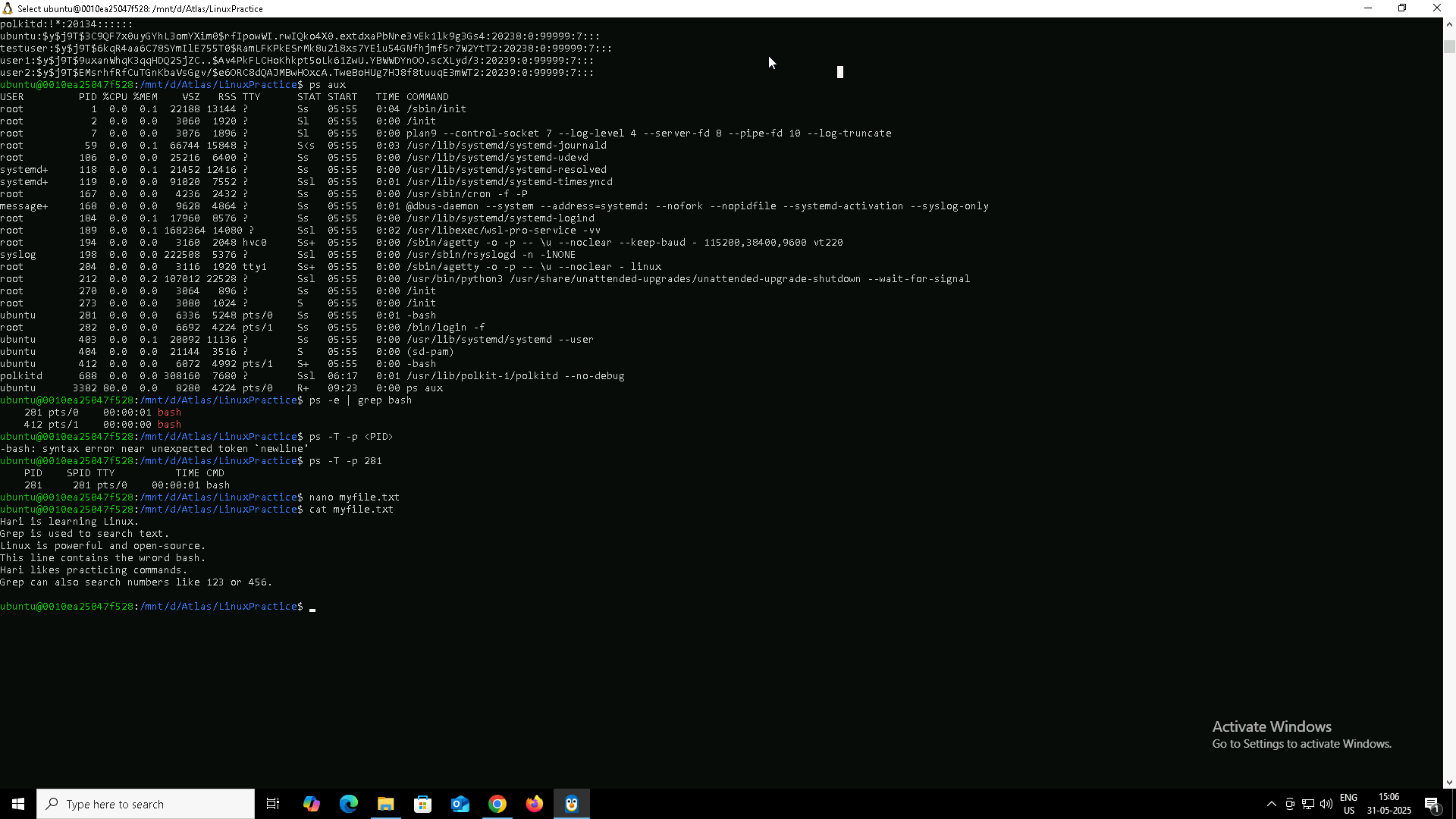
* A **process** is a complete running program.
* A **thread** is a smaller task that runs inside a process.
* One process can have **multiple threads** doing different work at the same time.

### **What I should remember:**

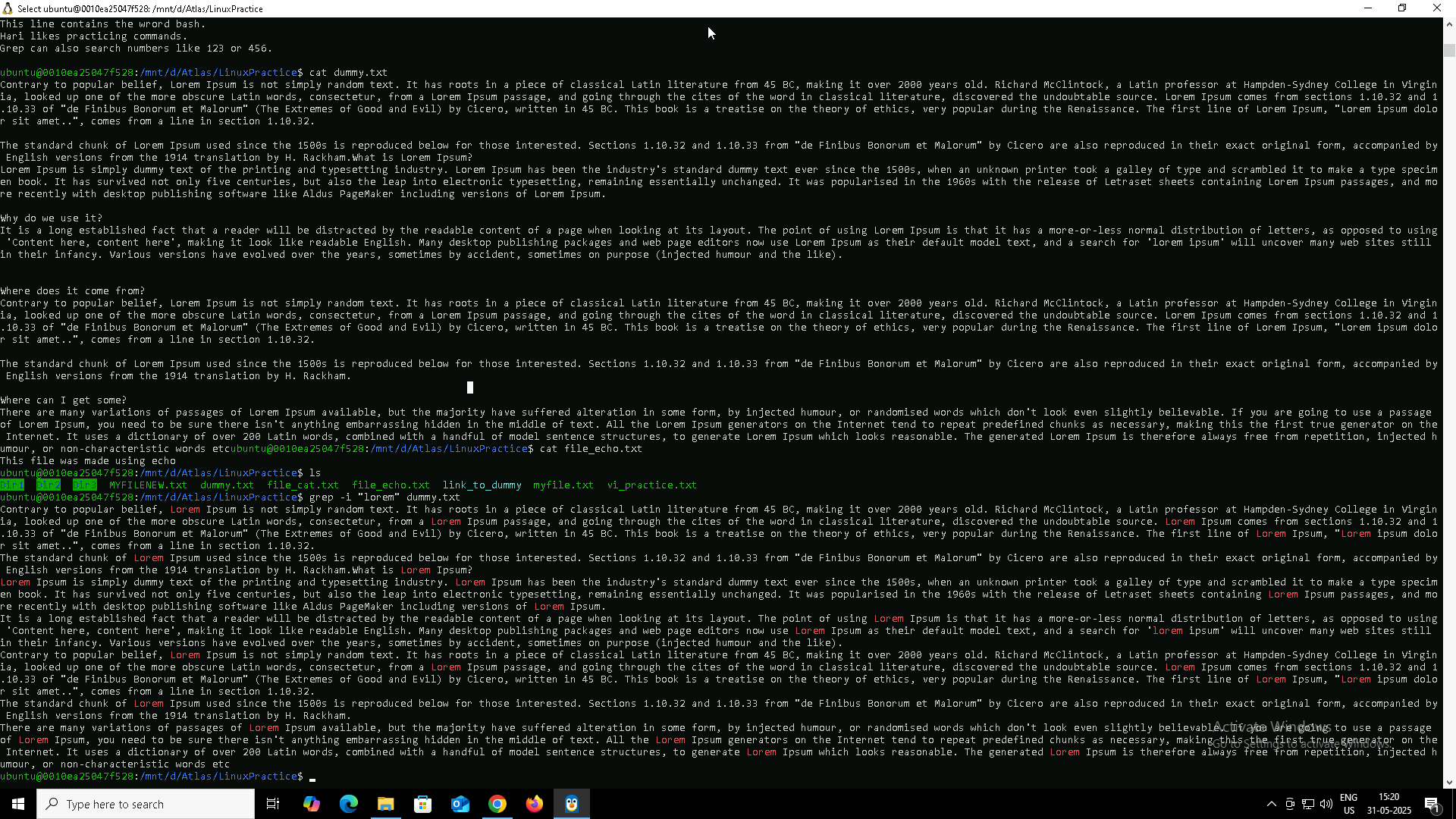
* **Process = big container**, **Thread = smaller unit inside it**
* Each process has **its own memory**. Threads **share memory** with each other.
* If one **process crashes**, others are not affected.
* If a **thread crashes**, it can affect the whole process.
* Processes take **more time and resources** to create. Threads are **faster and lightweight**.
* Communication between processes is harder. Threads can talk to each other easily.
* I can use ps and ps -T -p <PID> to check active processes and threads in Linux.

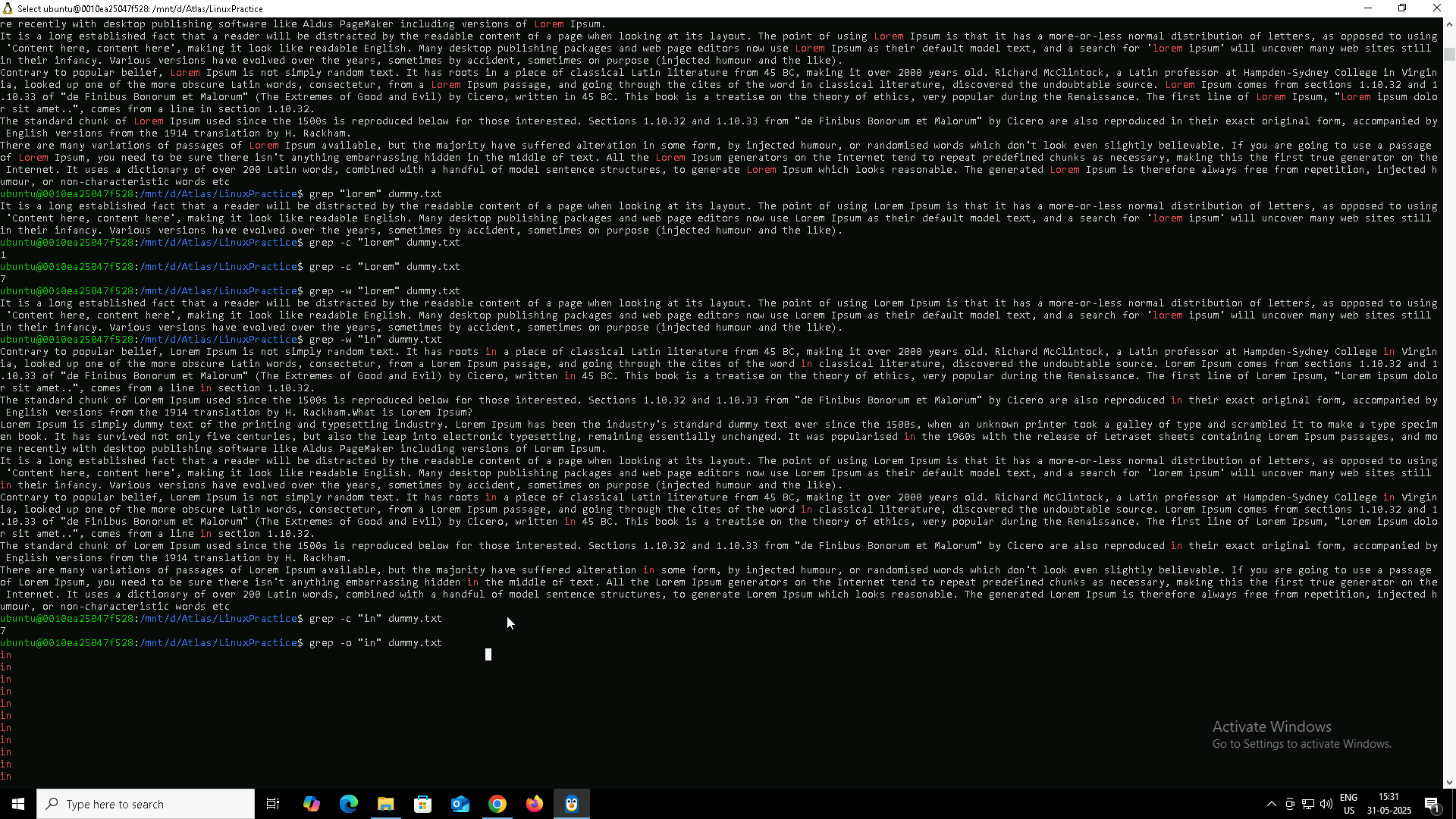
### **Example to relate:**

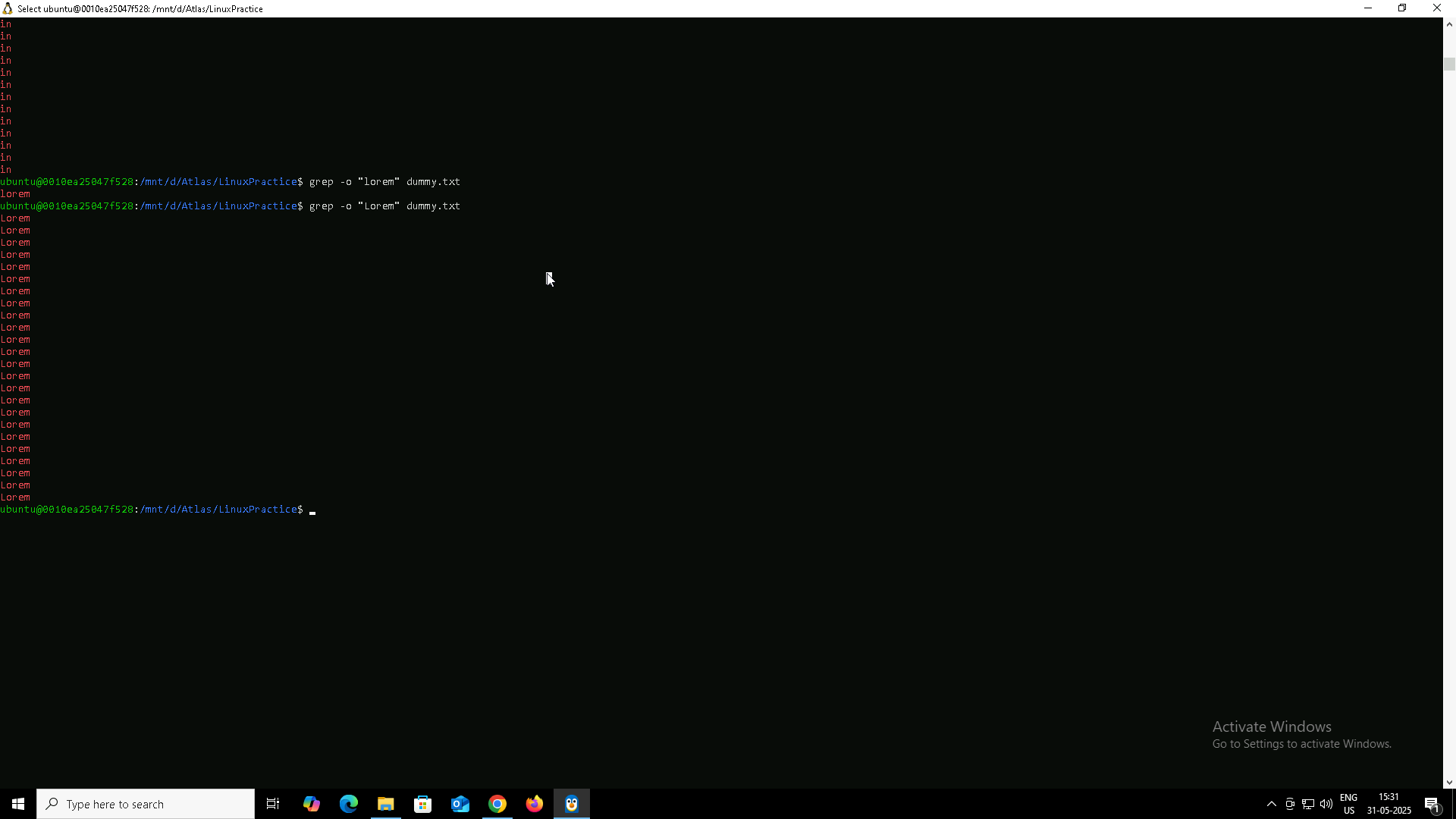
* If Chrome is the **process**, each tab is like a **thread**.
* Opening Chrome = one process.
* Opening 5 tabs = 5 threads working inside the same process.

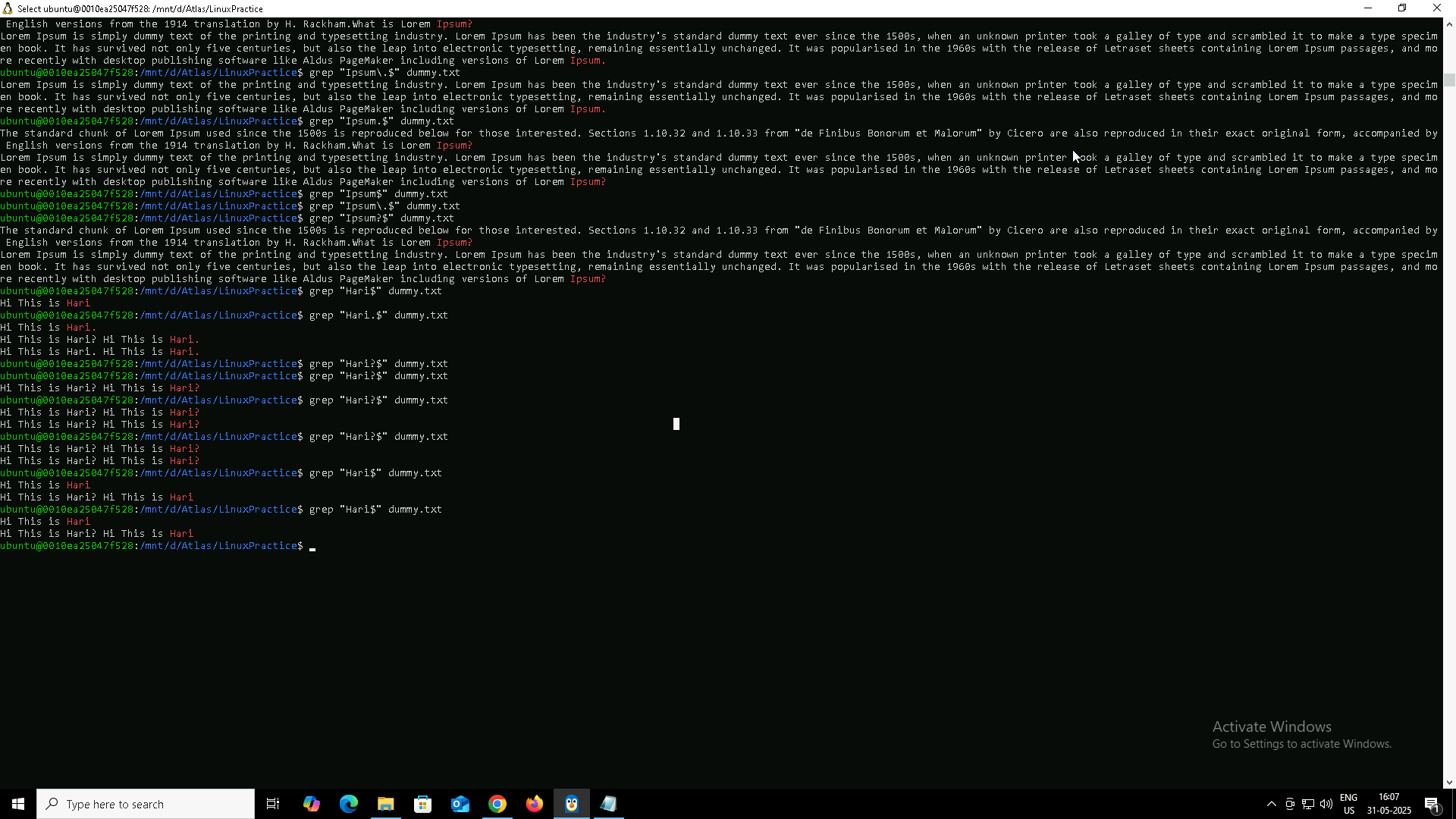


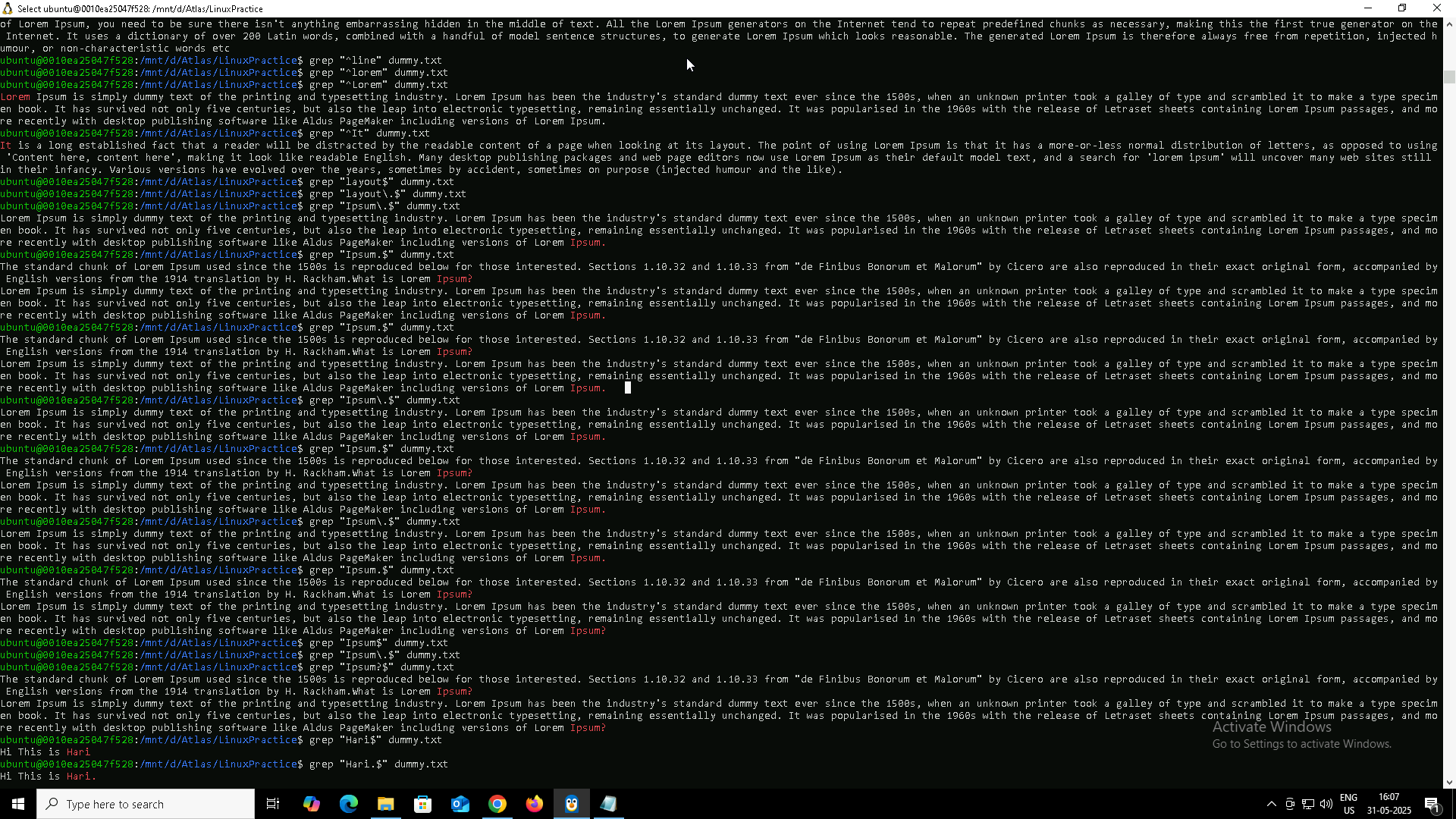
**Task 16: Linux Grep Commands**

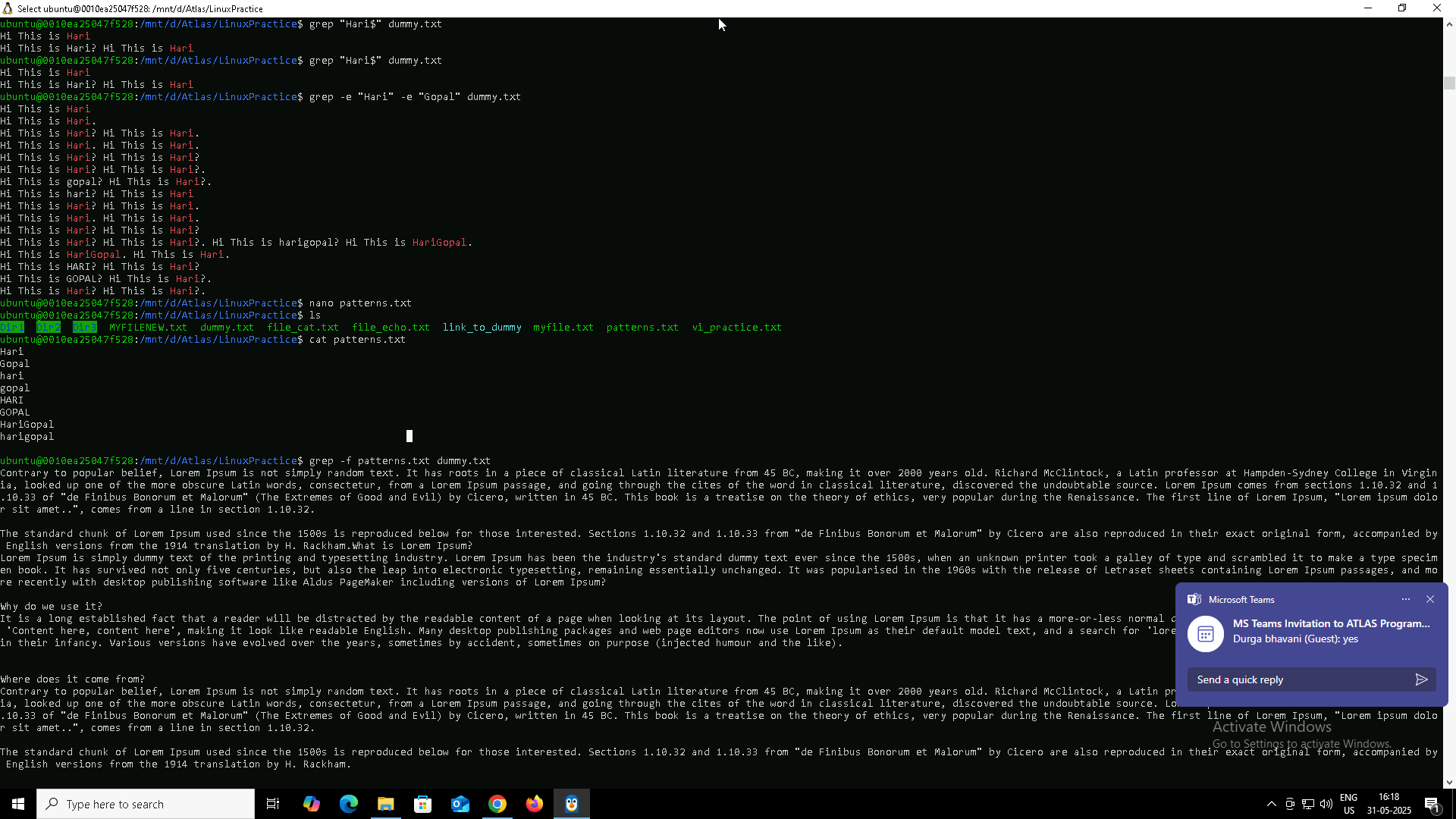


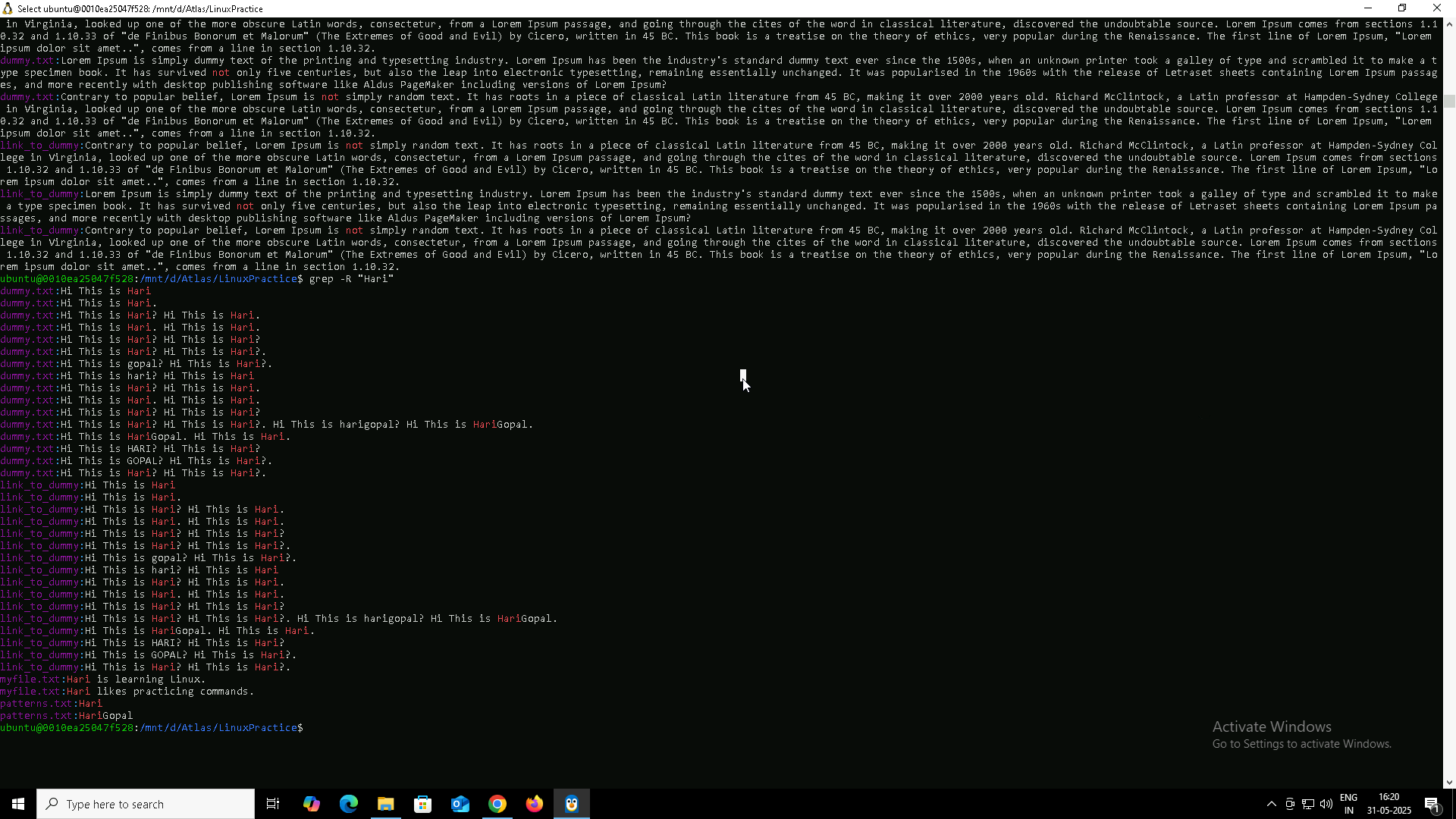












### **Basic grep options I tried:**

* grep -i "lorem" dummy.txt  
  → Case-insensitive search (matches “lorem”, “Lorem”, “LOREM”, etc.)
* grep -c "lorem" dummy.txt  
  → Count how many lines match
* grep -w "lorem" dummy.txt  
  → Match the exact word only (ignores words like “loremasters”)
* grep -o "lorem" dummy.txt  
  → Show only the matched word, one per line
* grep -n "lorem" dummy.txt  
  → Show matched lines with their line numbers
* grep -v "lorem" dummy.txt  
  → Show lines that do not contain the word
* grep "^lorem" dummy.txt  
  → Match lines that start with the word
* grep "lorem\.$" dummy.txt  
  → Match lines that end with the word (with dot escaped)

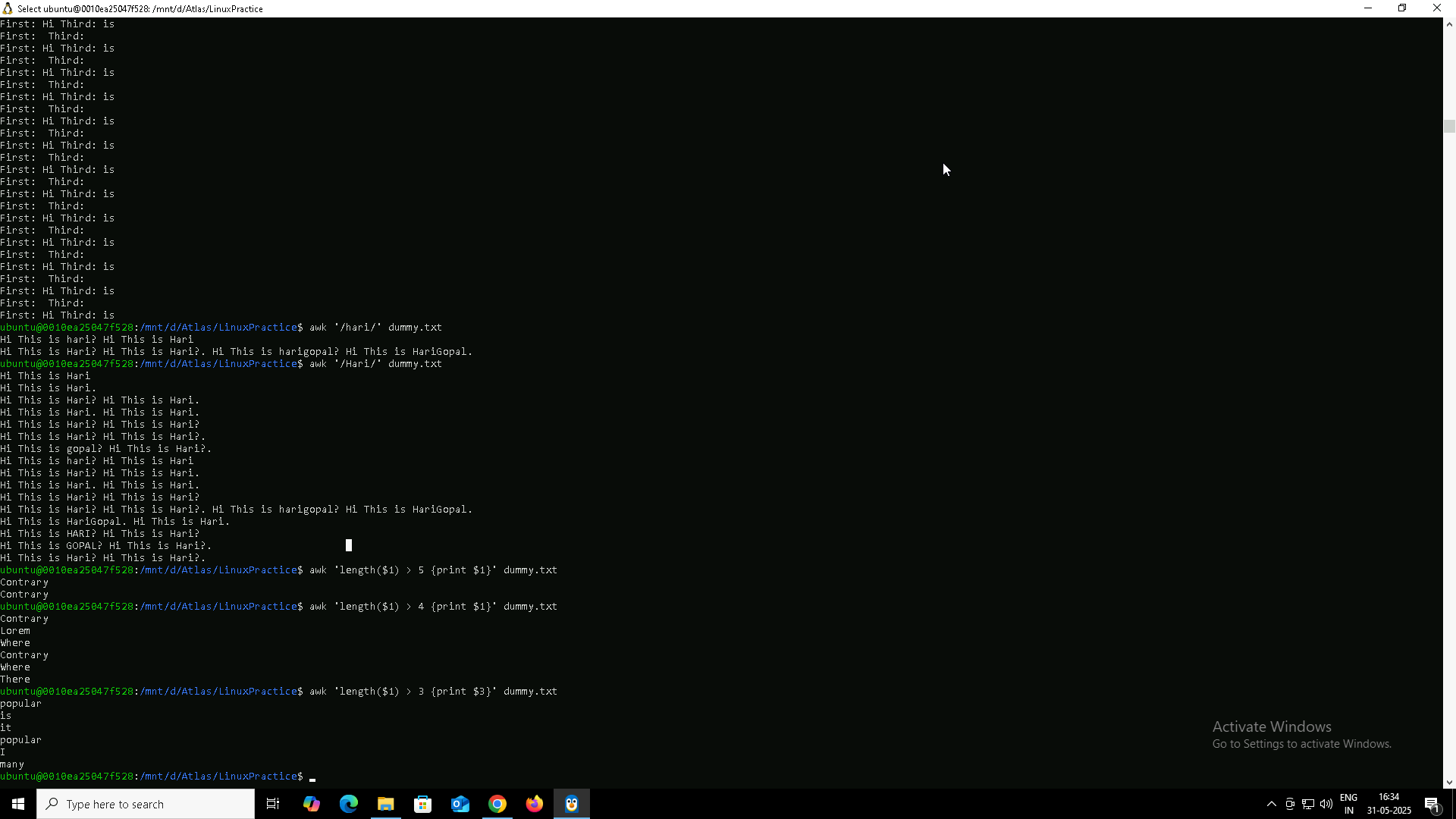
### **🔁 Advanced usage:**

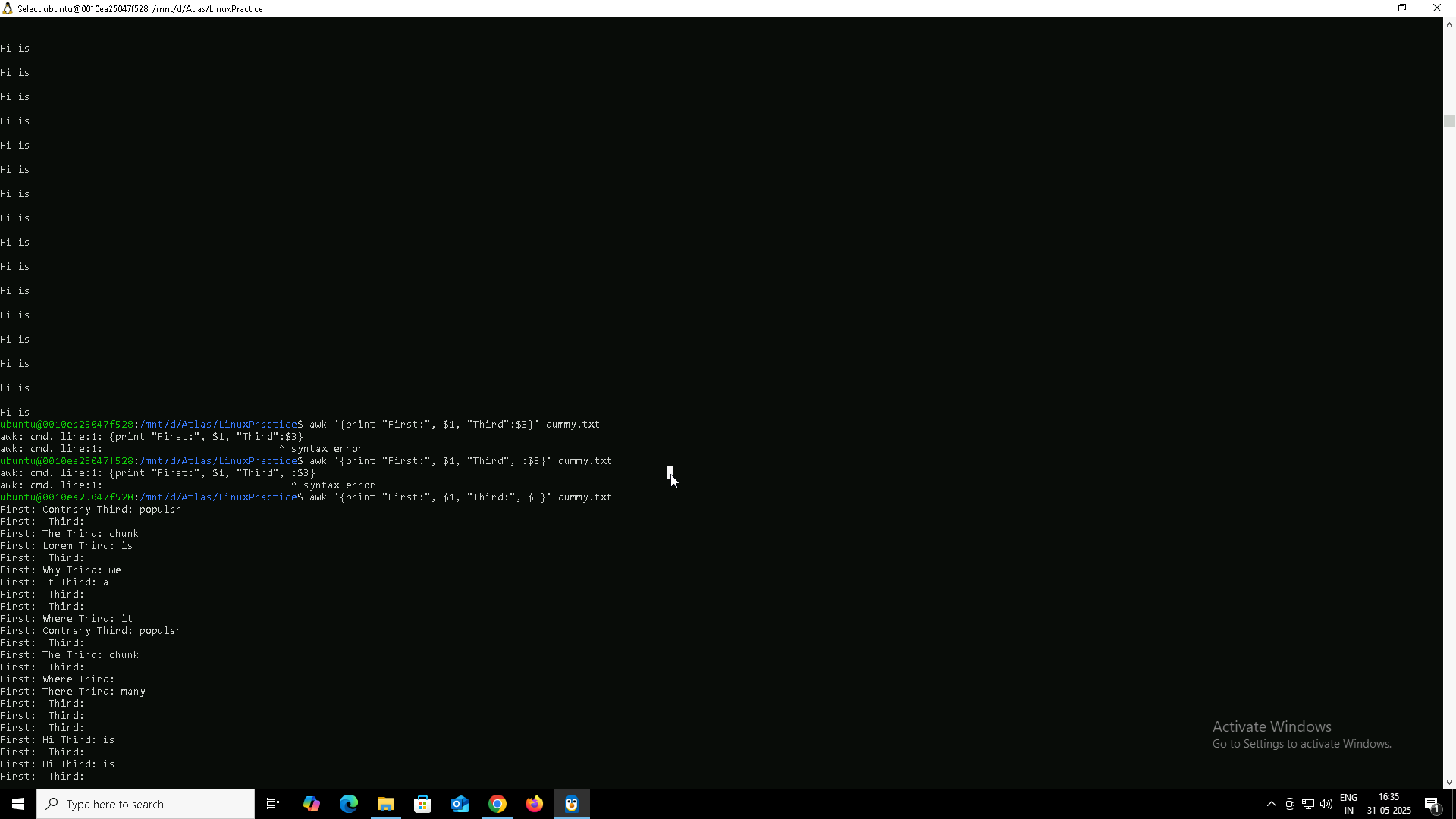
* grep -e "lorem" -e "ipsum" dummy.txt  
  → Match multiple patterns in a single command
* grep -f patterns.txt dummy.txt  
  → Match patterns listed in a separate file (patterns.txt)
* grep -A1 "lorem" dummy.txt  
  → Show the matched line and 1 line after it
* grep -R "lorem" .  
  → Recursively search all files in the current directory and subfolders

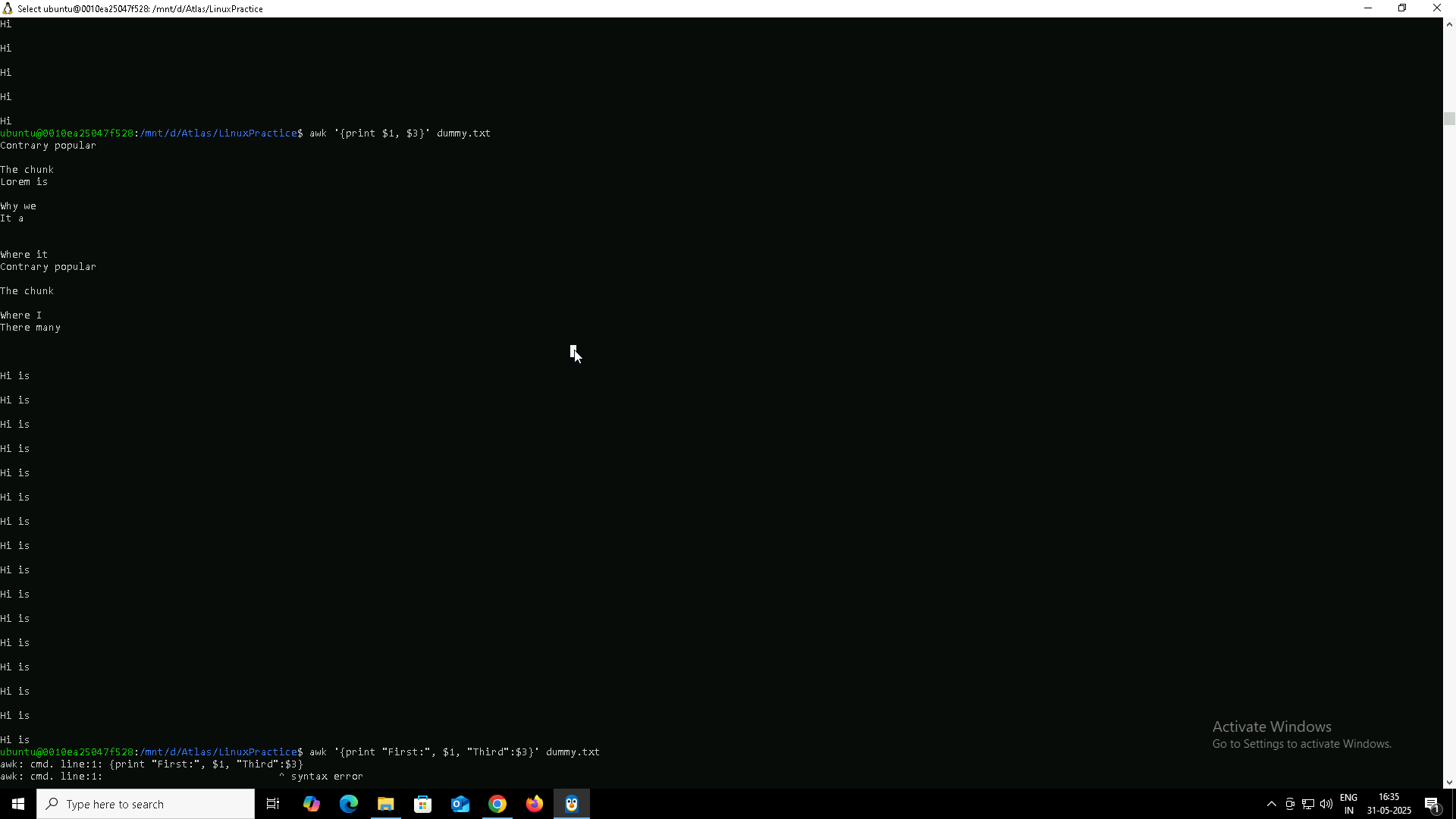
I have also tried with words Hari Gopal hari gopal which I have later added as a text in my file

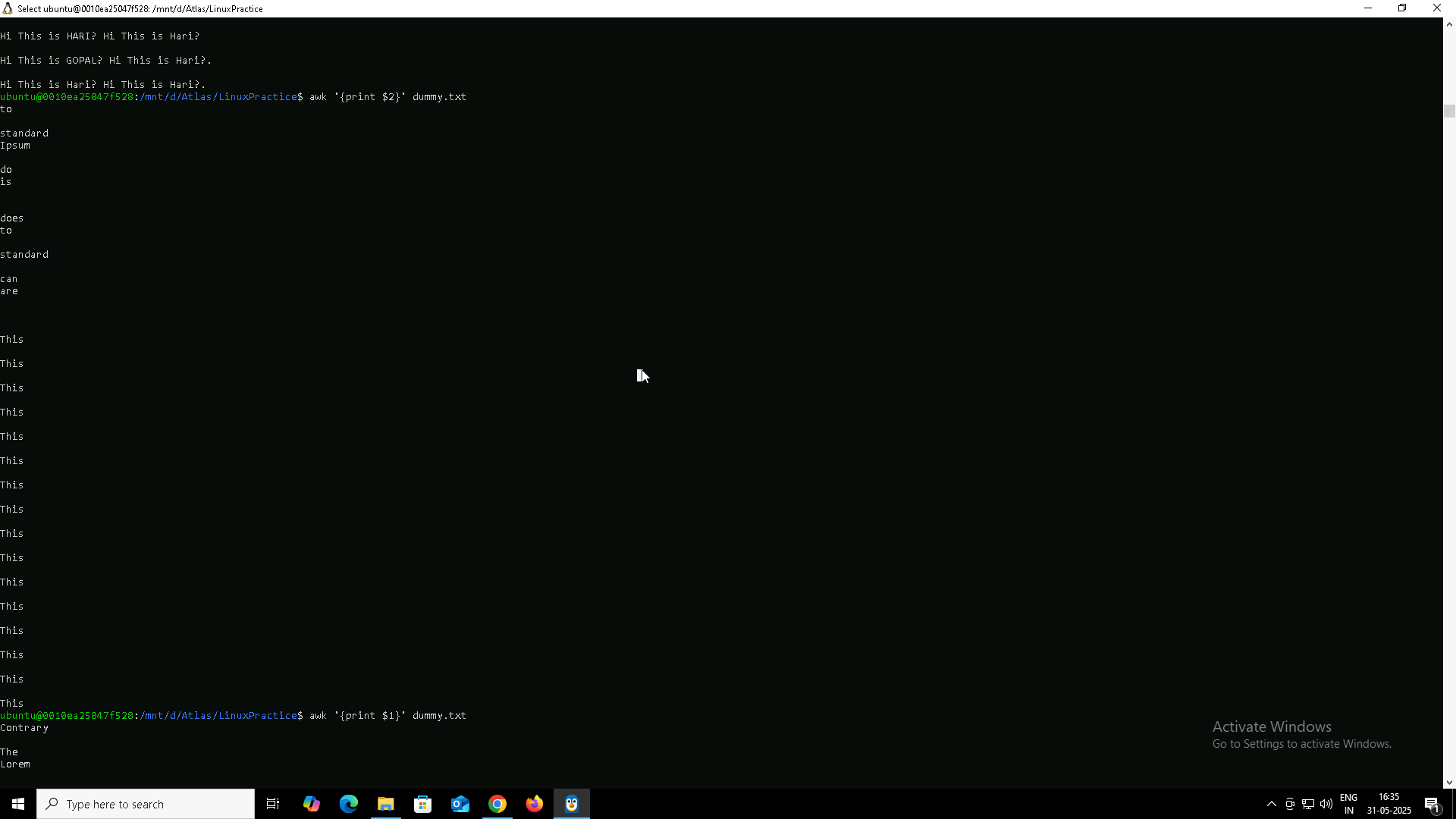
awk is a powerful command-line tool used for **pattern scanning** and **text processing** in Linux. It’s especially useful when working with **columns in files** or output.

I practiced using the awk command to filter and display specific parts of text files based on columns and conditions.









* awk is used to read a file **line by line** and work on **specific columns**.
* By default, it separates fields (words) based on **spaces**.

awk '{print}' dummy.txt

Prints the entire line.

awk '{print $2}' dummy.txt

Prints only the second word from each line.

awk '{print $1, $3}' dummy.txt

Prints the first and third words from each line.

awk '{print "First:", $1, "Third:", $3}' dummy.txt

Adds custom labels while printing selected columns.

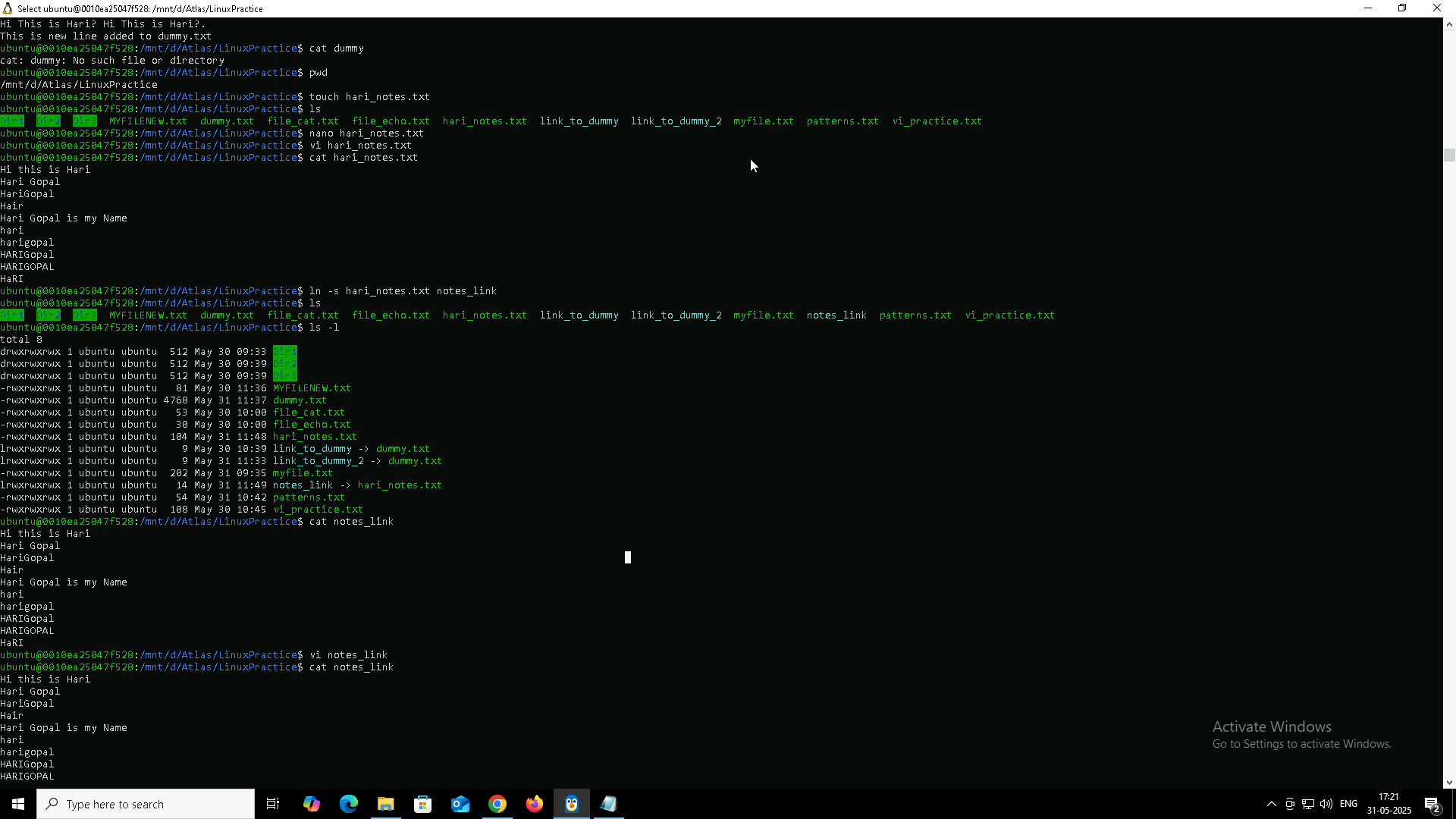
awk '/lorem/' dummy.txt

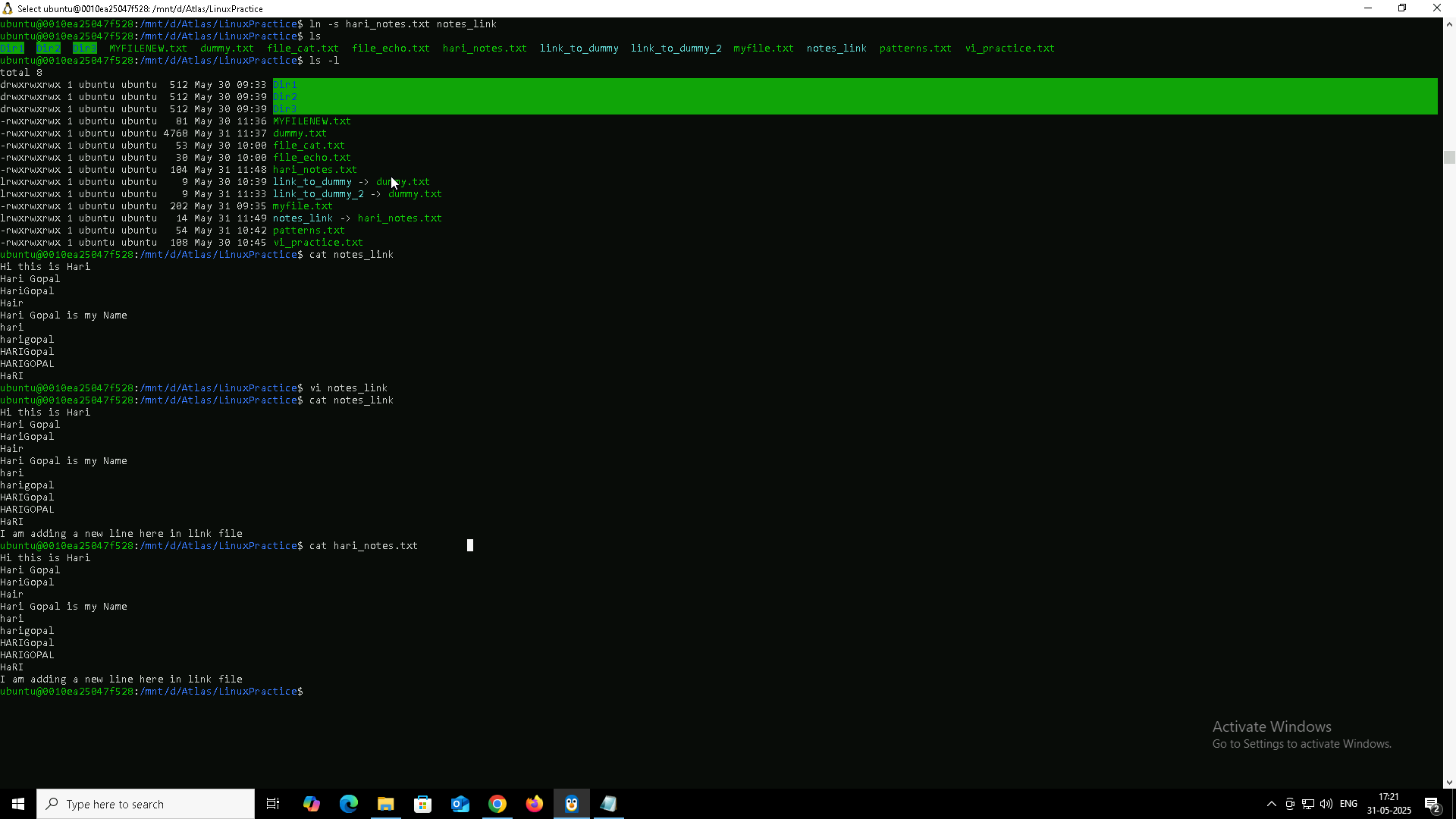
Prints only those lines that contain the word “lorem”.

awk 'length($1) > 5 {print $1}' dummy.txt

Prints the first word only if its length is greater than 5 characters.

#### **What I should remember:**

* $1, $2, $3, etc. refer to the 1st, 2nd, 3rd word in each line.
* awk works best when the file has a space or tab-separated structure.
* Very helpful in filtering and formatting output from text files.
* 



* Create/Edit file with nano:  
    
   nano filename → Type → Ctrl + O (save), Ctrl + X (exit)
* Create/Edit file with vi:  
    
   vi filename  
  + Press i to insert
  + Press Esc to exit insert mode
  + Type :wq to save and exit
* Other ways to create file:  
  + touch filename → Creates an empty file
  + cat > filename → Starts writing directly (End with Ctrl + D)
* Symbolic link:  
    
   ln -s original\_file link\_name
* View file content:  
    
   cat filename or cat link\_name (works for symlinks too)

### **Task 18 – File Permissions (using ls -l)**

* Use ls -l to view file permissions:

ls -l

* Example output:  
  -rw-r--r-- 1 ubuntu ubuntu 0 May 31 11:57 perms\_check.txt
* Breakdown of -rw-r--r--:
  + - → regular file
  + rw- → owner: read & write
  + r-- → group: read-only
  + r-- → others: read-only
* To check a specific file:

ls -l perms\_check.txt

### **Task 19: Default Permissions of a New File**

(From default\_check.txt created using touch)

* Owner → rw-
* Group → r--
* Others → r--

### **Task 20: What is the command to change the permission to read only for the owner, group, and all other users?**

* Command used → chmod 444 perms\_check.txt
* Effect → Only read permission is given to owner, group, and others. No write or execute allowed.

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

Owner: Read and Write

Group: Read

Other: No permissions (None)

* Command used → chmod 640 perms\_check.txt

### **Task 22: What was the command for changing the file permissions to -rw-r-----?**

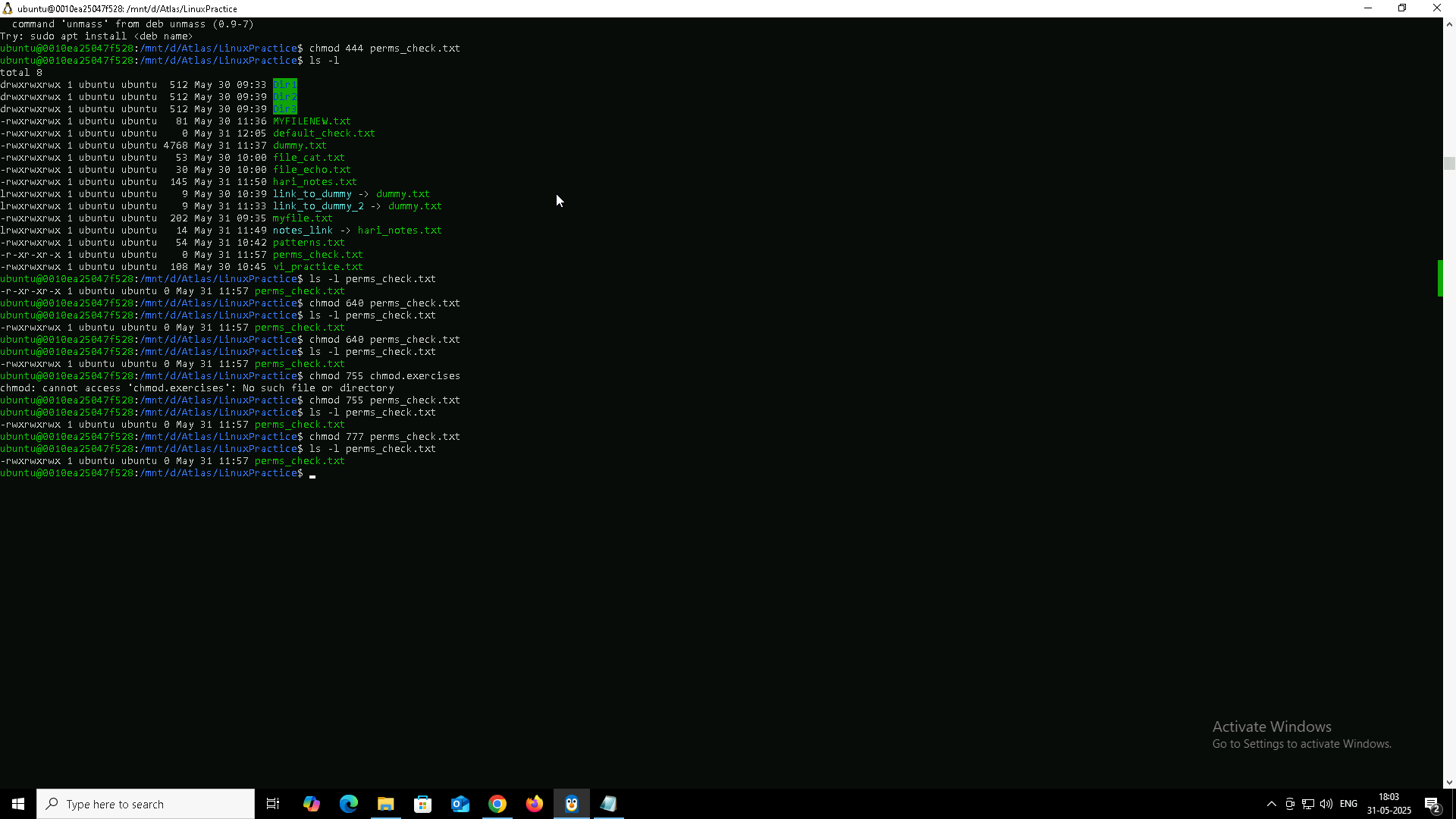
* Command used → chmod 640 perms\_check.txt
* This sets:  
  + Owner → rw-
  + Group → r--
  + Others → ---

### **Task 23: Change file permissions to -rwxr-x-x**

* File used → perms\_check.txt
* Command used → chmod 755 perms\_check.txt
* This sets:  
  + Owner → rwx
  + Group → r-x
  + Others → r-x

### **Task 24: What was the command for changing the file permissions to -rwxr-x--x?**

* Command used → chmod 751 perms\_check.txt
* This sets:  
  + Owner → rwx
  + Group → r-x
  + Others → --x



### **Task 25: What will this command do?**

chown -c master file1.txt

* **chown** is used to change the **ownership** of a file or directory.
* **-c** stands for **verbose mode** – it shows a message **only if** a change **was actually made**.
* master is the **new owner**.
* file1.txt is the **target file**.

#### **What will happen?**

* This command will **change the owner of file1.txt to master**, and will **print a message only if the ownership is updated**.

#### **Example output**

changed ownership of 'file1.txt' from ubuntu to master

#### **Important Note:**

* I **must be sudo or root** to execute this command, otherwise I’ll get a **“Permission denied”** error

### **Task 26: What is a process?**

* A process is a program in execution.
* Each process gets a unique **PID (Process ID)**.
* It runs in its own memory space and is managed by the **kernel**.
* Example: Running gedit, top, or ls creates a new process.

### **Task 27: Command to list current processes**

ps

* Use ps -ef for detailed list of all system processes.

Task 28:

Can you list all the running processes?

Shared ss below

Task 29:

What will ps -f command do ?

