Day 6 - 31st May 2025

Name - Rishabh Manchanda

Regular Expressions

Regular Expressions (Regex) Regular Expression, or regex or regexp in short, is extremely and amazingly powerful in searching and manipulating text strings, particularly in processing text files. One line of regex can easily replace several dozen lines of programming codes.

* The **dot (.)** is a wildcard that matches any single character except for a newline.

It’s super useful when you’re not sure what character might come in the middle. For example, if you're searching through a document and want to find both "hot" and "hat", you can just write h.t and you'll catch both.

* **Asterisk (\*)** matches the preceding character zero or more times.The asterisk means the character before it can repeat any number of times, even zero.It just keeps going, as long as the character before is repeating.
* The **plus sign (+)** matches the preceding character one or more times. It means one or more times.It tells the pattern to match the character (or group) at least once, but it can be repeated any number of times.
* To match the preceding character zero or one time, the **question mark (?)** is used.It makes the character (or group) just before it optional like saying, Hey, you can be here once, or not at all. I’m fine either way.

* **Caret (^)** asserts the position at the start of a line or string.The caret is used to mean start of a line or string. When you put it at the beginning of your regex pattern.
* The **dollar sign ($)** asserts the position at the end of a line or string. It matches the end of string or line.
* For grouping expressions or creating sub expressions, **parentheses( )**are used.
* **Square brackets []**define a character set, matching any one character within the set.
* A **hyphen -** inside square brackets denotes a range of characters.
* To escape a special regex character and treat it as a literal character, a **backslash \** is used.
* The **vertical bar |** acts as an OR operator, allowing you to match either one pattern or another (e.g., cat|dog matches "cat" or "dog").
* **curly braces {}**specify the exact number or range of occurrences of the preceding character or group.

**What are features of linux OS?**

**Open Source and Free:** This is arguably Linux's most defining feature. Its source code is freely available to anyone, allowing users to view, modify, and distribute it under licenses like the GNU General Public License (GPL). This fosters a massive, global community of developers who continuously contribute to its improvement, bug fixes, and feature additions. Being free also eliminates licensing costs, making it a highly cost-effective solution for individuals and businesses.

**Stability and Reliability:** Linux is celebrated for its exceptional stability and reliability. It can run for extended periods without needing reboots, which is critical for servers and mission-critical systems. Its well-engineered design and the constant scrutiny of its open-source codebase contribute to fewer crashes and increased uptime.

**Security:** Linux is inherently more secure than many proprietary operating systems. Its robust permission system, user privilege management, and the transparency of its open-source code allow for rapid identification and patching of vulnerabilities by the community. While not entirely immune to malware, it's significantly less targeted than other popular operating systems, and its architecture makes it more resistant to common threats.

**Multitasking and Multi-User:** Linux is a true multitasking operating system, capable of running multiple applications and processes concurrently without significant performance degradation. It's also a multi-user system, meaning multiple users can access the system's resources (like memory, CPU, and applications) simultaneously, each with their own secure environment.

**Customizability and Flexibility:** Linux offers unparalleled customization. Users have complete control over their operating system, from choosing different desktop environments (like GNOME, KDE, Xfce) that dictate the look and feel, to selecting specific kernels and even building their own Linux distributions tailored to their exact needs. This flexibility extends to installing only the necessary components, making it adaptable to various hardware configurations.

**Command-Line Interface (CLI) and Graphical User Interface (GUI):** While Linux is famous for its powerful and efficient command-line interface (Shell), which is a favorite among developers and system administrators for automation and precise control, it also fully supports various graphical user interfaces (GUIs). This provides a user-friendly visual experience similar to other popular operating systems, making it accessible to a wider audience.

**Portability:** Linux is highly portable and can run on a vast array of hardware platforms, from tiny embedded devices (like routers and smart TVs) to powerful supercomputers. This adaptability is a significant advantage for cross-platform development and deployment.

**Efficient Package Management:** Linux distributions utilize sophisticated package management systems (like APT for Debian/Ubuntu, YUM/DNF for Red Hat/CentOS) that simplify the installation, updating, and removal of software. These systems provide access to vast repositories of applications, ensuring that users can easily find and manage the software they need.

**What are the basic components of Linux?**

The Linux Kernel:

* What it is: This is the absolute core of the Linux operating system. It's the "brain" that interfaces directly with your computer's hardware (CPU, memory, storage, network cards, peripherals, etc.).

**The Shell:**

* **What it is:** A command-line interpreter that provides an interface between the user and the kernel. It's where you type commands.
* **What it does:** The shell (like Bash, Zsh, or Fish) takes your commands, interprets them, and passes them to the kernel for execution. It also provides features for scripting, history, tab completion, and environment management.

**System Utilities / GNU Core Utilities:**A collection of essential command-line tools that perform basic tasks. The most common set is the GNU Core Utilities.These are the programs you use every day in the terminal:eg, mkdir, ls, cp, mv,cat,etc.

**Bootloader:**A small program that runs when your computer starts up, *before* the operating system loads.Its primary function is to load the Linux kernel into memory and pass control to it, allowing the operating system to begin its boot process. It often provides a menu (like GRUB) if you have multiple operating systems installed, allowing you to choose which one to boot.

**What is Kernel ?**

The kernel is the central, fundamental component of the operating system. It's the "brain" or "heart" of the system, acting as the crucial interface between your computer's hardware (CPU, memory, storage, peripherals, etc.) and all the software applications and processes running on it.

**Core of the OS:** While a full Linux distribution includes many components like a graphical user interface (GUI), system libraries, and user-space utilities, the kernel is the very core. Everything else relies on the kernel to interact with the hardware.

**Hardware Abstraction and Management:** The kernel's primary role is to manage and abstract the underlying hardware. This means that applications don't need to know the specific details of how a particular piece of hardware works. Instead, they make requests to the kernel, and the kernel handles the low-level communication with the hardware through device drivers. This abstraction allows for great flexibility, as different hardware can be supported without changing the applications.

**Resource Management:** The kernel is responsible for efficiently managing system resources, including:

* **CPU Management (Process Scheduling):** It determines which processes get access to the CPU, when, and for how long. It ensures fair distribution of CPU time among multiple running applications.
* **Memory Management:** It allocates and deallocates memory to processes, keeping track of what memory is being used by whom. It also implements virtual memory, allowing processes to believe they have more memory than physically available.
* **Device Management:** It controls access to all connected hardware devices (keyboard, mouse, hard drives, network cards, printers, etc.) through device drivers.
* **File System Management:** It provides an interface for interacting with various file systems (like ext4, NTFS, FAT), handling file storage, access, and permissions.
* **Network Management:** It implements network protocols (like TCP/IP) and manages network interfaces, enabling communication with other systems.

**System Calls:** Applications interact with the kernel through "system calls." These are special requests that applications make to the kernel to perform privileged operations, such as reading a file, writing to a device, creating a new process, or accessing network resources.

**Kernel Space vs. User Space:** The operating system is divided into two main modes or "spaces":

* **Kernel Space:** This is a highly privileged area where the kernel code runs. It has direct and unrestricted access to the computer's hardware. A crash in kernel space can bring down the entire system.
* **User Space:** This is where all user applications and regular processes run. They have restricted access to hardware and must make system calls to the kernel to perform operations that require hardware access. This separation provides security and stability; if an application in user space crashes, it typically doesn't affect the entire system.

Task 4:

**BASH in Linux full form and Explanation-**

BASH stands for Bourne Again Shell

**Shell:** In the context of operating systems like Linux, a "shell" is a command-line interpreter. It's a program that provides an interface between the user and the operating system's kernel. When you type commands into your terminal, it's the shell that interprets those commands and passes them to the kernel for execution. Think of it as a translator that takes your human-readable commands and turns them into something the computer can understand.

**Bourne:** The "Bourne" part refers to the original Unix shell, developed by Stephen Bourne at Bell Labs. This shell, often referred to as sh, was one of the earliest and most influential command interpreters for Unix-like systems. It introduced many fundamental features that are still prevalent in modern shells today, such as variables, control flow (loops, conditionals), and command substitution.

**Again:** The "Again" part signifies that BASH is a *replacement* and an *enhancement* of the original Bourne shell. It was developed by Brian Fox for the GNU Project in 1989 as a free software alternative to sh. It was designed to be largely compatible with the Bourne shell, meaning scripts written for sh would often run directly in BASH, but it also incorporated numerous useful features from other popular shells of the time, such as the Korn shell (ksh) and the C shell (csh).

A Command-Line Interpreter: When you open a terminal in most Linux distributions (or macOS, or even Windows Subsystem for Linux), you're typically interacting with BASH. You type commands (like ls to list files, cd to change directories, pwd to print the current working directory), and BASH executes them.

**BASH is like a special translator or a control panel for your computer.**

Here's the super easy breakdown:

1. **It's a "Shell":** Think of a shell as the "outer layer" that lets you talk to the computer's core (the "kernel"). When you open a "terminal" or "command prompt" on a Linux computer, you're usually looking at a Bash shell.
2. **It understands "commands":** Instead of clicking icons, you *type* instructions into Bash. Like:

A Scripting Language: BASH is also a powerful scripting language. You can write sequences of commands in a plain text file (called a "shell script" or "bash script," usually with a .sh extension) and then execute that file. This allows you to automate repetitive tasks, create complex workflows, and build custom tools. Bash scripts are widely used for system administration, deployment, and various automation purposes.

Feature-Rich: BASH includes many features that enhance both interactive use and scripting, such as:

**Command history**: Easily recall and re-execute previous commands.

**Tab completion**: Autocomplete commands, file paths, and options, saving time and reducing typos.

**Variables and arrays**: Store and manipulate data.

**Control flow**: if statements, for loops, while loops for conditional execution and iteration.

Input/output redirection: Direct the output of a command to a file or use a file as input for a command.

Piping: Connect the output of one command to the input of another.

Job control: Manage background and foreground processes.

**What do you think is the difference between LInux and Windows?**

Linux and Windows are both operating systems, but they are fundamentally different in their philosophy, development, usage, and overall user experience. Here's a breakdown of the key distinctions:

1. **Licensing & Source Code:**

* **Linux**: Is open source and free. This means its source code is publicly available, allowing anyone to view, modify, and distribute it. Most Linux distributions are free to download and use, fostering a massive global community of developers who contribute to its continuous improvement.
* **Windows**: Is proprietary and commercial. Its source code is owned and controlled by Microsoft and is not publicly accessible. Users must purchase a license to use Windows, and its development and distribution are managed by Microsoft.

2. **Customization and Flexibility:**

* **Linux**: Offers unparalleled customization. Due to its modular design and open-source nature, users can choose from a vast array of desktop environments (GNOME, KDE, Xfce, etc.), window managers, and other components to tailor the OS to their exact preferences and needs. This makes it highly flexible and adaptable to various hardware, including older machines.
* Windows: Is less customizable. While you can change themes and some settings, the core user interface and system components are fixed. Microsoft aims for a consistent user experience across all its devices, which limits deep customization.

**3. User Interface (GUI vs. CLI):**

* **Linux**: While modern Linux distributions offer user-friendly Graphical User Interfaces (GUIs) that are comparable to Windows, it's also renowned for its powerful and efficient Command-Line Interface (CLI), typically accessed through a "shell" like Bash. The CLI offers granular control and is favored by developers and system administrators for automation and advanced tasks.
* **Windows**: Primarily focuses on a highly user-friendly Graphical User Interface (GUI). Most tasks can be performed visually, making it very accessible for casual users. While it has a command prompt (CMD) and PowerShell, they are generally less central to the typical user experience compared to the Linux CLI.

4. **Security**:

* Linux: Generally considered more secure. Its permission system, user privilege management (requiring sudo for administrative tasks), and the transparency of its open-source code allow for rapid identification and patching of vulnerabilities by the community. Its diversity across distributions also makes it a less attractive and harder target for widespread malware.
* **Windows**: Historically has been more susceptible to malware and viruses due to its larger market share, making it a prime target for malicious actors. While Microsoft has made significant strides in security with features like Windows Defender, its popularity still makes it a more common target.

**Software Availability and Compatibility:**

* **Linux:** Has a **growing software ecosystem**, with many popular open-source applications (like Firefox, LibreOffice, GIMP) available. Many Windows applications can also be run using compatibility layers like Wine, but native support for some proprietary software (especially games and professional creative suites) can be limited.
* **Windows:** Boasts the **largest software ecosystem** due to its widespread adoption. Almost all commercial software, including productivity suites, games, and specialized applications, is developed for Windows first. This makes it the go-to choice for users who rely on specific Windows-only applications.

**What is LILO? Explain?**

**LILO** stands for **LInux LOader**. It is a **boot loader** that was very prominent in the early days of Linux. While largely superseded by GRUB (GRand Unified Bootloader) in most modern Linux distributions, understanding LILO is important for appreciating the evolution of Linux and the boot process.

When you power on your computer, the BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface) performs a Power-On Self-Test (POST) to initialize hardware. After the POST, the BIOS/UEFI looks for a bootable device (like a hard drive, SSD, or USB drive). Once it finds one, it hands over control to the **boot loader**

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

Shell is a command-line interpreter. It's a program that provides a user interface for accessing the operating system's services. Think of it as a translator: it takes your commands (which are human-readable text strings), interprets them, and then passes them to the operating system's kernel for execution.

**Bourne Shell-**This is the most dominant and widely used family of shells in the Unix/Linux world. They are characterized by their scripting syntax, which originated from the original Bourne Shell. They are generally considered more robust and reliable for scripting.

**Bourne Again SHell (bash)-**

The most popular and default shell on most Linux distributions and macOS (until recently, where zsh became default for new users). It's "Bourne Again" because it's fully compatible with the sh syntax but incorporates numerous enhancements from ksh and csh.

**C Shell (csh) Family-**

These shells adopt a syntax that is more similar to the C programming language. While they introduced some important interactive features early on, their scripting capabilities are often considered less robust and more prone to errors compared to Bourne-compatible shells. They are generally not recommended for complex scripting.

**TENEX C Shell (tcsh)-**

Built upon csh by adding more user-friendly interactive features such as advanced command-line editing, programmable completion, and history expansion. It aimed to address some of the interactive shortcomings of the original.

**What is swap space?**

Swap space in Linux is a designated area on a hard drive or SSD that the operating system uses as a temporary extension of your computer's physical RAM (Random Access Memory). It's a crucial component of virtual memory management.

RAM is extremely fast and where the CPU primarily works with data and programs. However, RAM is a finite resource. When your system runs out of physical RAM due to too many applications running, very large applications, or memory leaks, it needs a way to continue functioning without crashing. This is where swap space comes in.

**Prevents Out-of-Memory (OOM) Errors:** The primary purpose is to prevent your system from freezing or crashing when RAM is exhausted. Instead of outright failing, the system can offload less critical data to disk.

**Memory Overcommitment:** Swap space allows the system to allocate more virtual memory to processes than is physically available in RAM. This is useful because many applications request a lot of memory upfront but might not use all of it immediately.

**Supports Large Programs and Multitasking:** It enables you to run more applications simultaneously or run very memory-intensive applications that might otherwise exceed your physical RAM.

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

Mounting is the process of making a file system available for use by the operating system and its users. It's how you access the data stored on various storage devices.

When you plug in a USB drive, insert a CD/DVD, or have a separate hard drive partition, these devices contain their own file systems (their own "trees" of directories and files). For the operating system to see and interact with these separate file systems, they need to be "attached" to a specific point within the main file system tree. This attachment point is called a **mount point**.

The primary command for mounting is mount. You typically need root privileges (using sudo) to mount file systems.

sudo mount [device] [mount\_point]

Just as important as mounting is **unmounting**. Unmounting detaches the file system from its mount point, making it safe to remove the physical device or perform maintenance. Crucially, you should always unmount a device before physically disconnecting it to prevent data corruption.

**What is chmod command ? how to use it?**

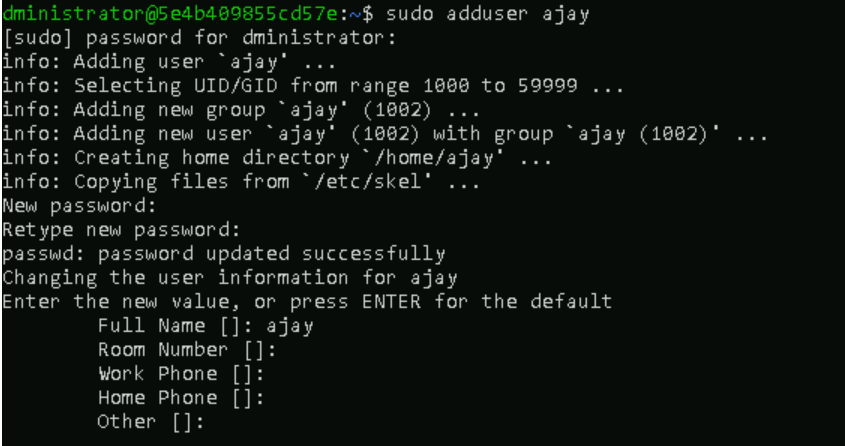
The chmod command in Linux (and other Unix-like operating systems) is short for "change mode". Its primary purpose is to modify the file system permissions of files and directories. These permissions control who can read, write, or execute a file or access a directory.

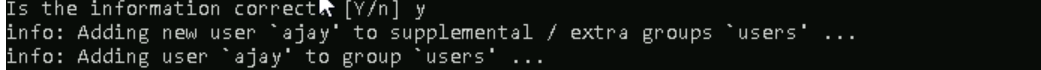
chmod u+x file.sh → Adds **execute** permission for the **user (owner)**

chmod g-w file.txt → Removes **write** permission from the **group**

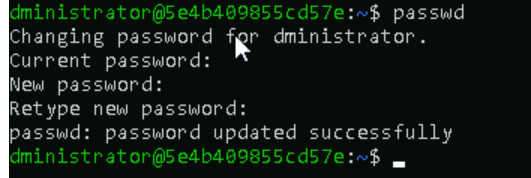
chmod o=r file.txt → Sets **read-only** for **others**

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





**Can you change the password of a user?**

****

### 

**What is diff between Process and Thread?**

In an operating system, both processes and threads are fundamental units of execution, but they differ significantly in their scope, resources, independence, and how they interact. Think of them as different levels of "work units" that your computer handles.

### **Process (Heavyweight Process)**

A **process** is an **independent instance of a running program**. When you double-click an application icon (like a web browser, a word processor, or a game), the operating system creates a new process for that program.

### **Thread (Lightweight Process)**

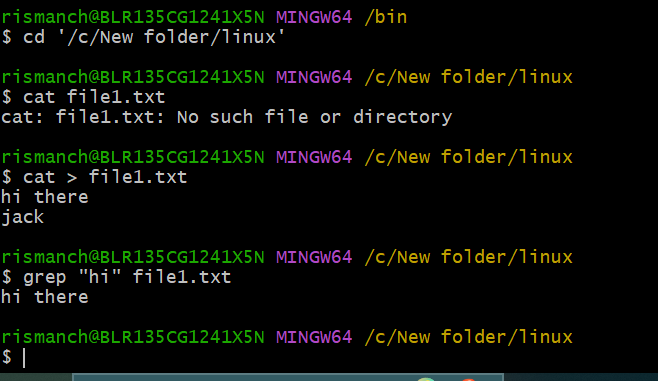
A **thread** is a **single sequence of execution within a process**. Every process starts with at least one main thread, and it can create multiple additional threads. All threads *within the same process* share the process's resources.

**Part of a Process:** A thread cannot exist independently; it always belongs to a process.

**Shared Resources:** Threads *within the same process* share the process's:

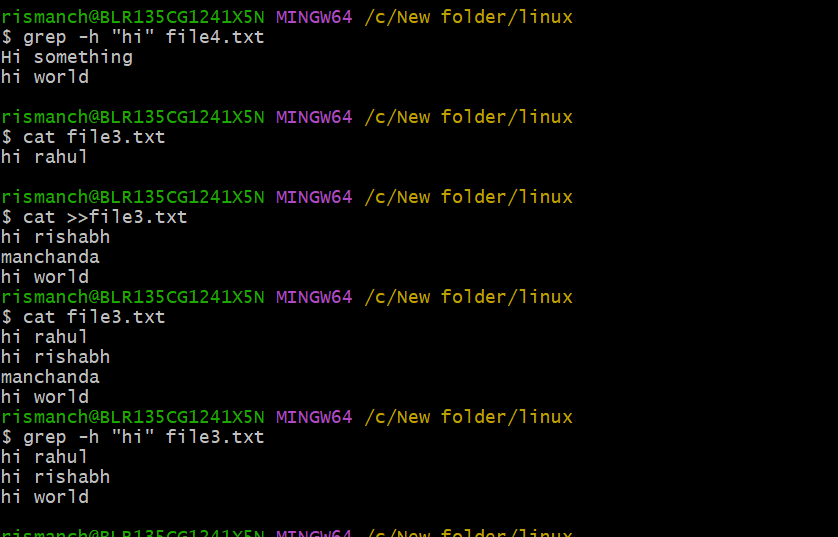
* **Code/Text Section:** The executable code.
* **Data Section:** Global and static variables.
* **Heap:** Dynamically allocated memory.
* **File Descriptors:** Open files, network connections

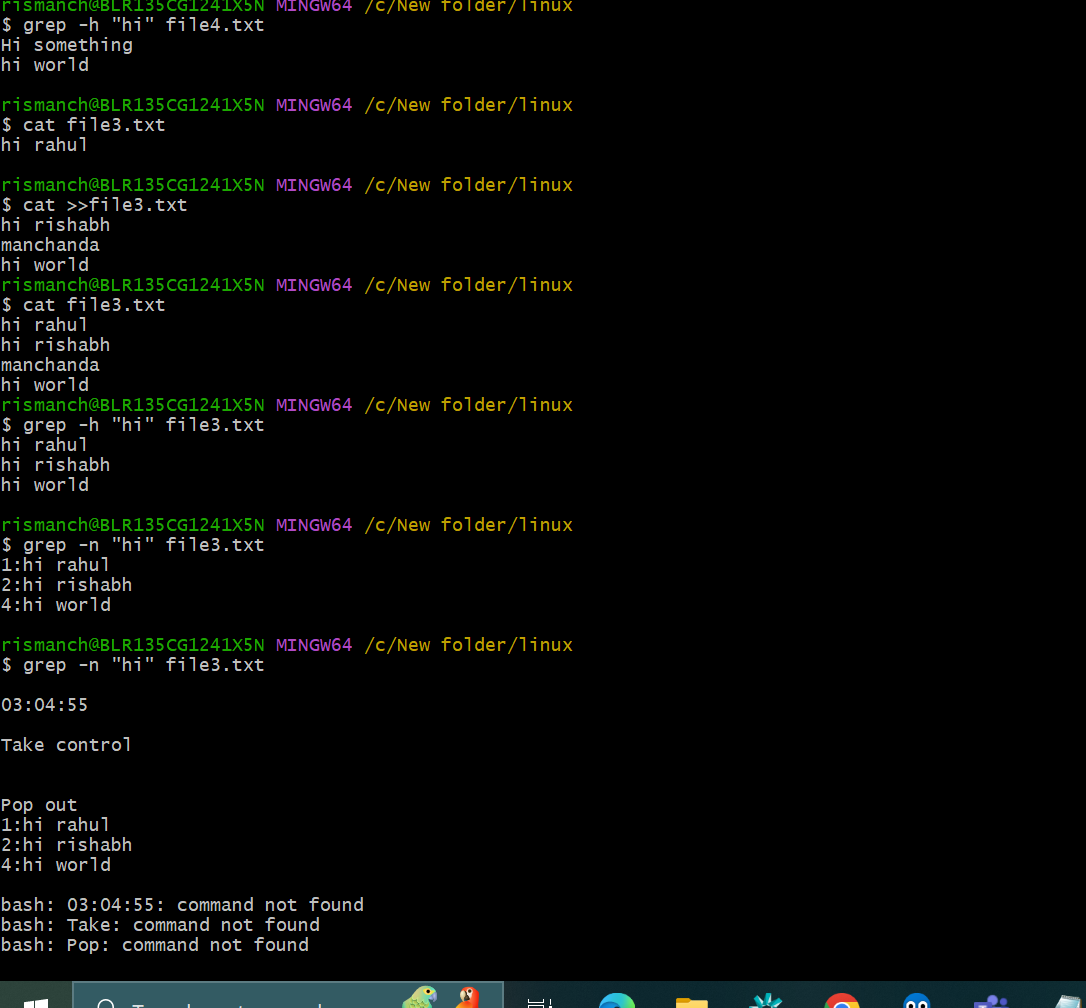
**keep a file ready with some content in it for Grep command.**

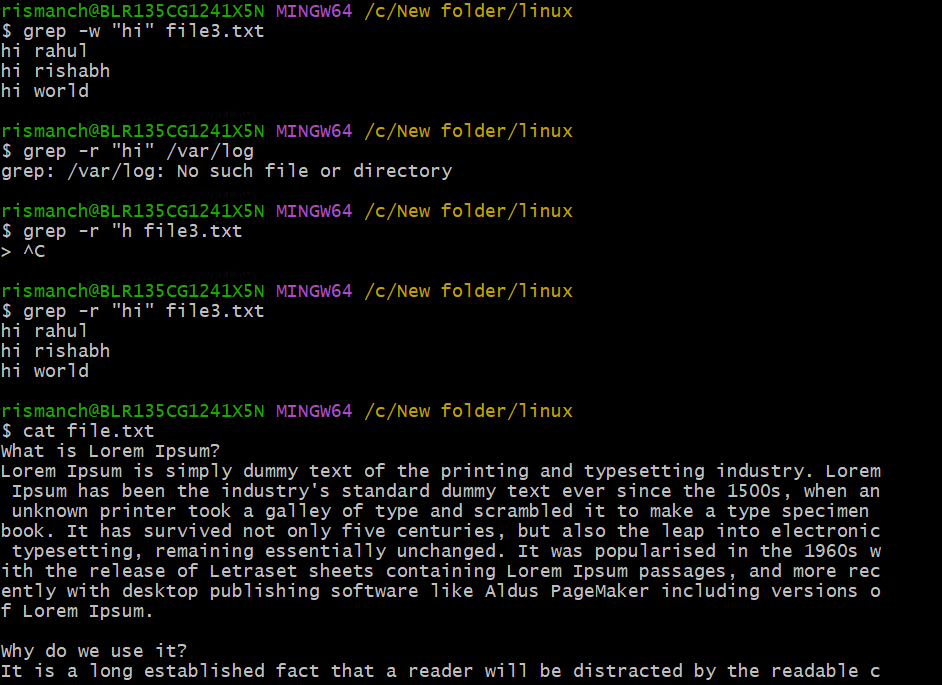


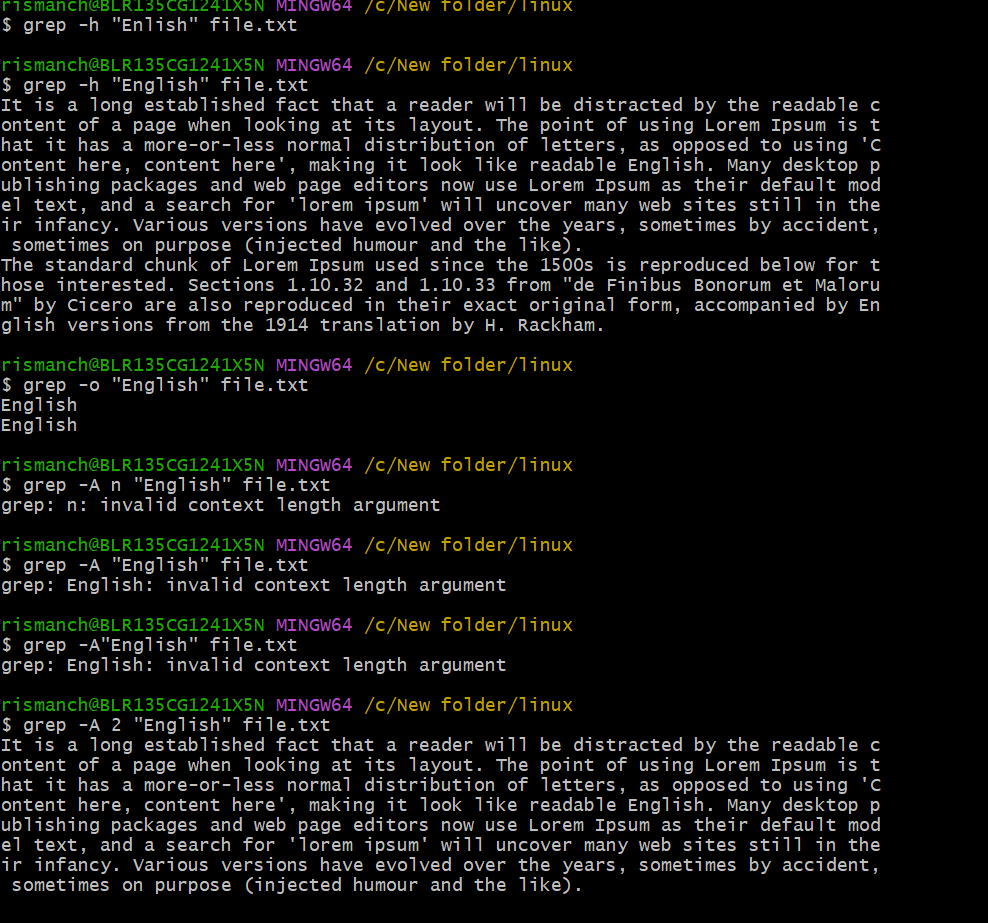
Task 16:

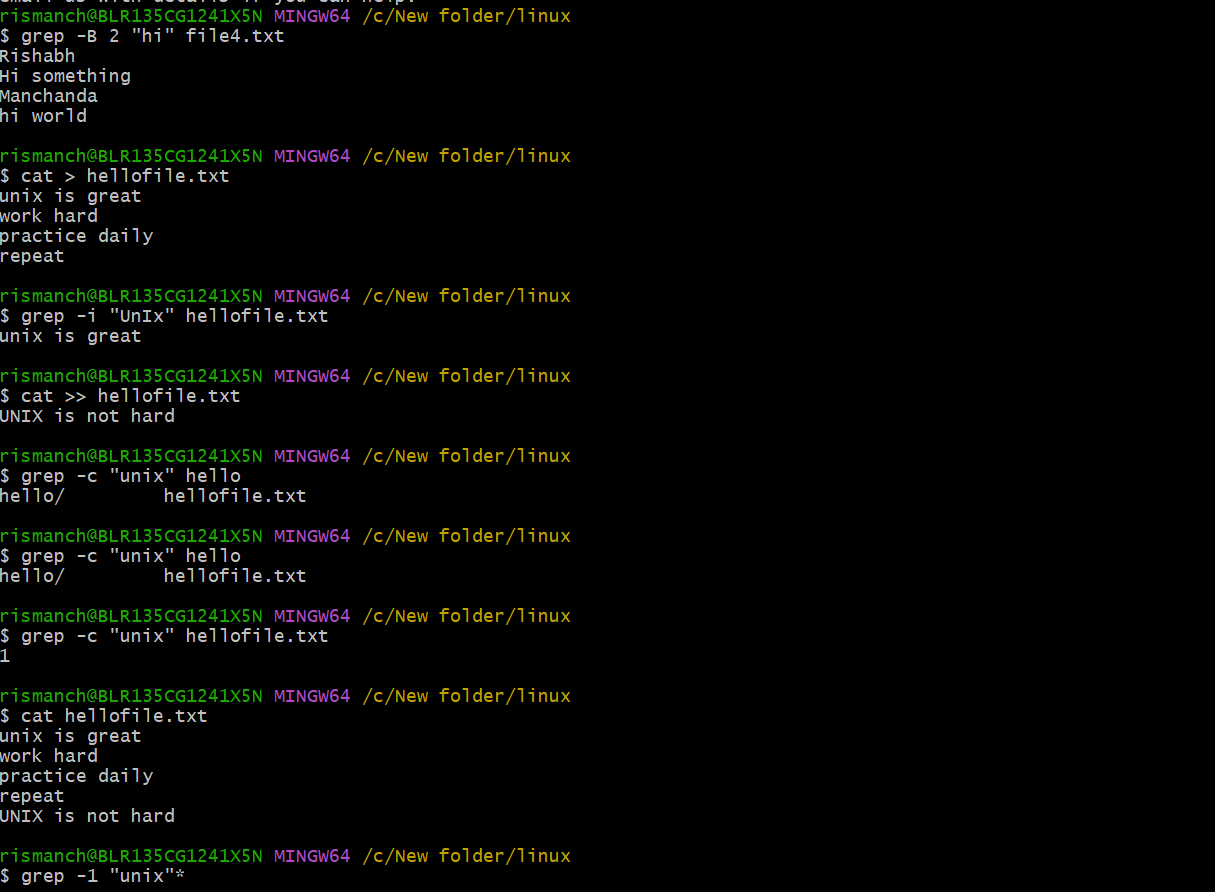
**Doc 14 Linux Grep commands in docs to study folder**

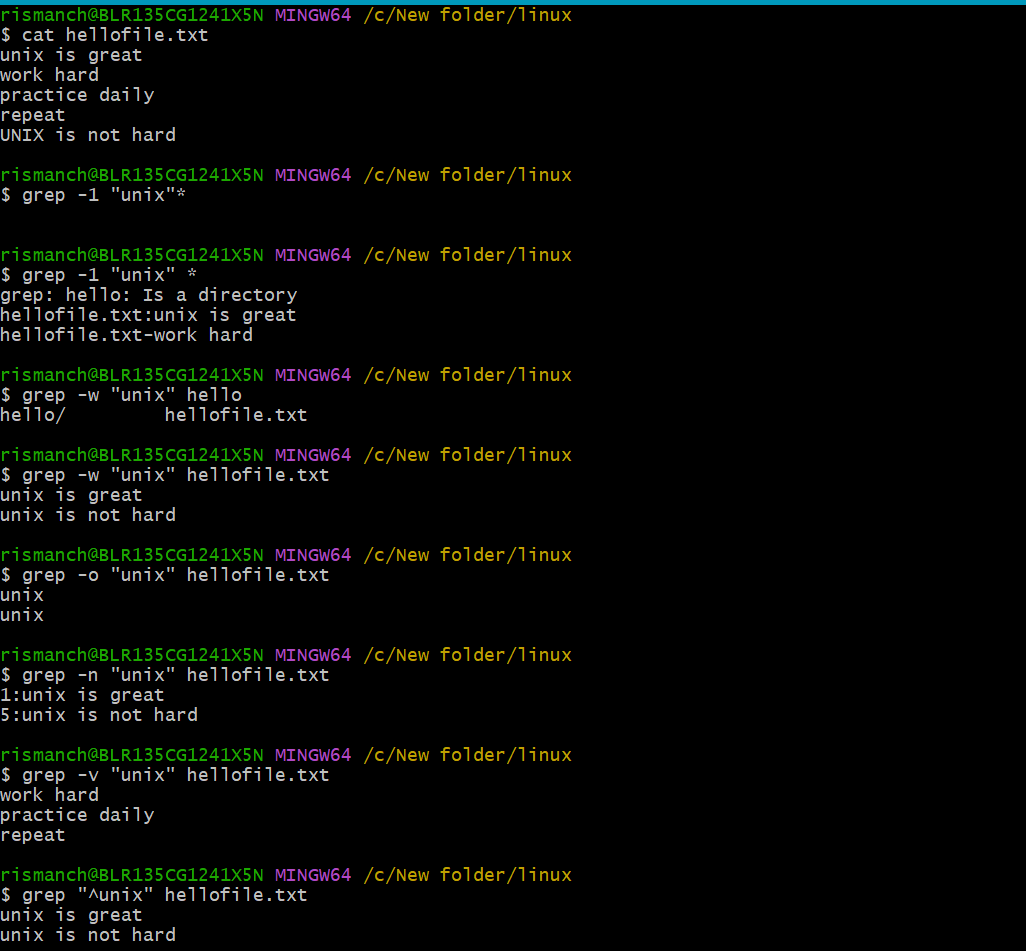
****

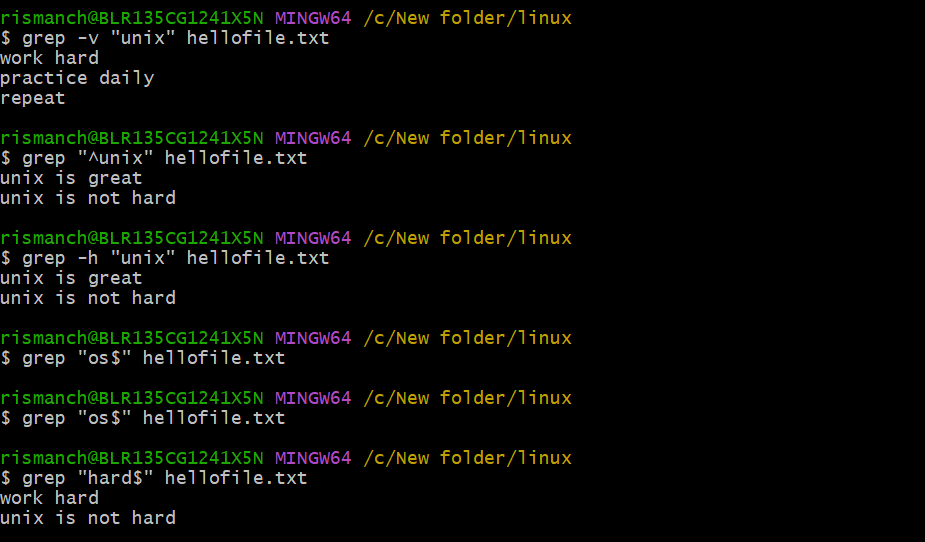
****

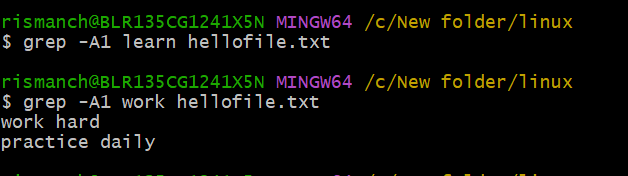
****

****

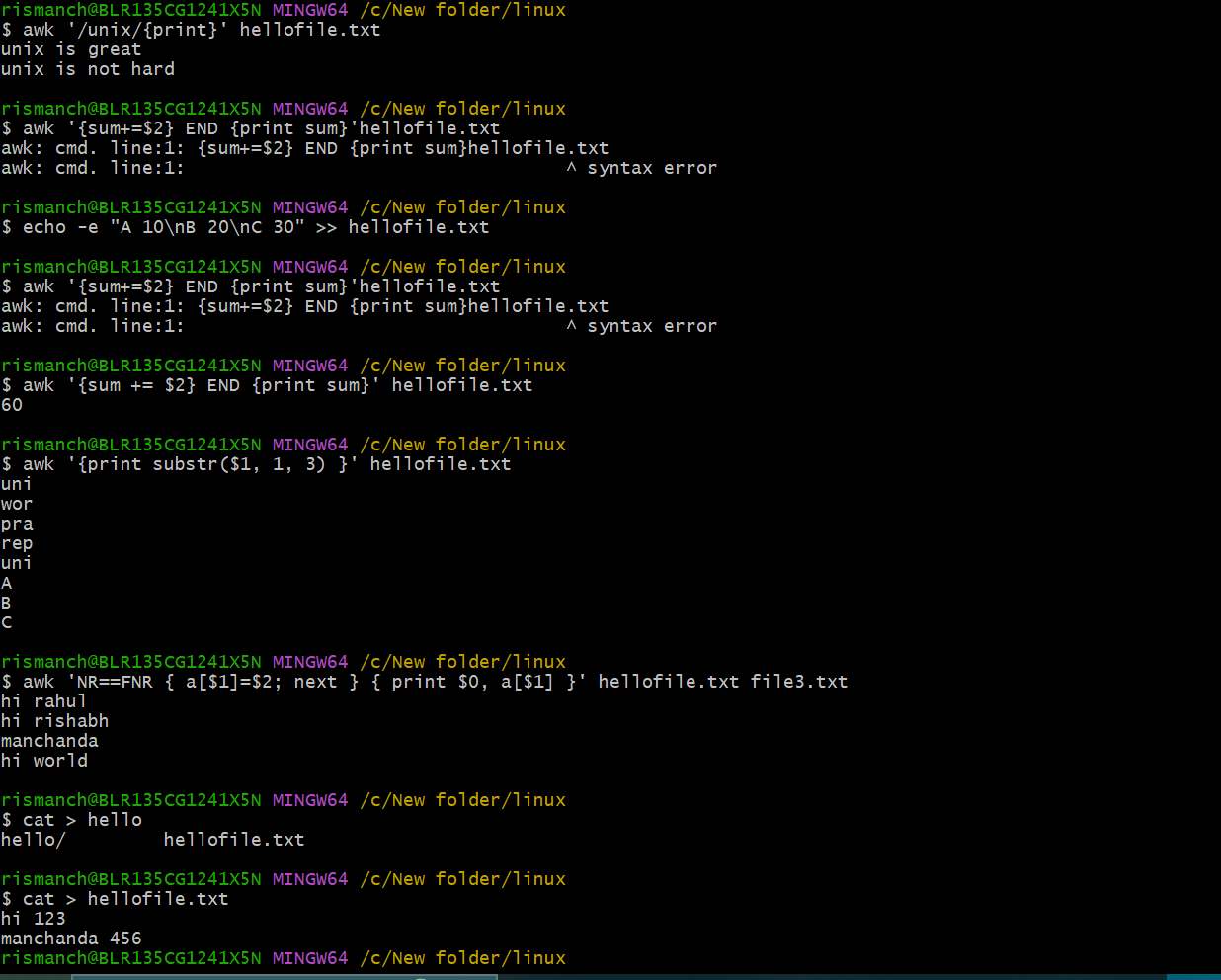
****

****

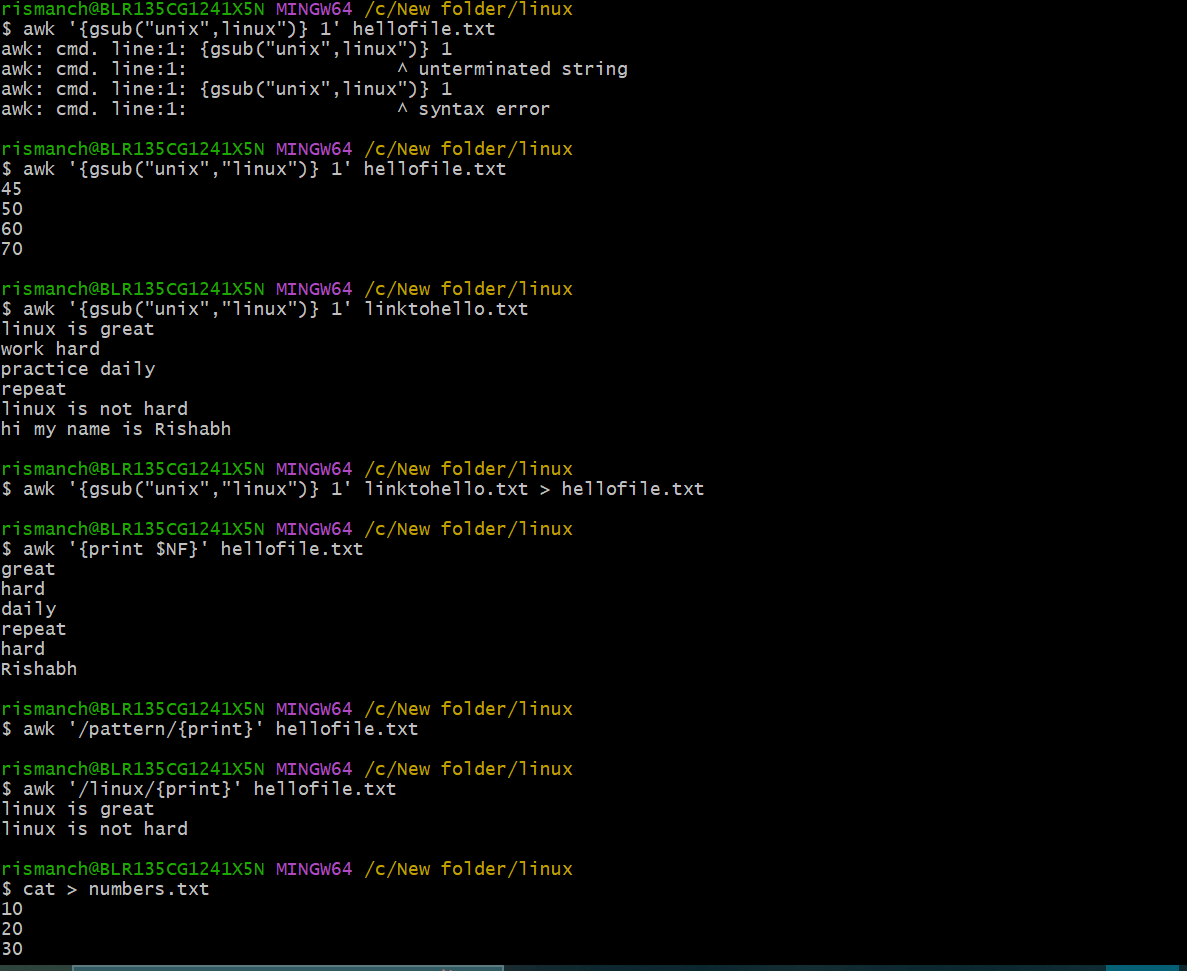
****

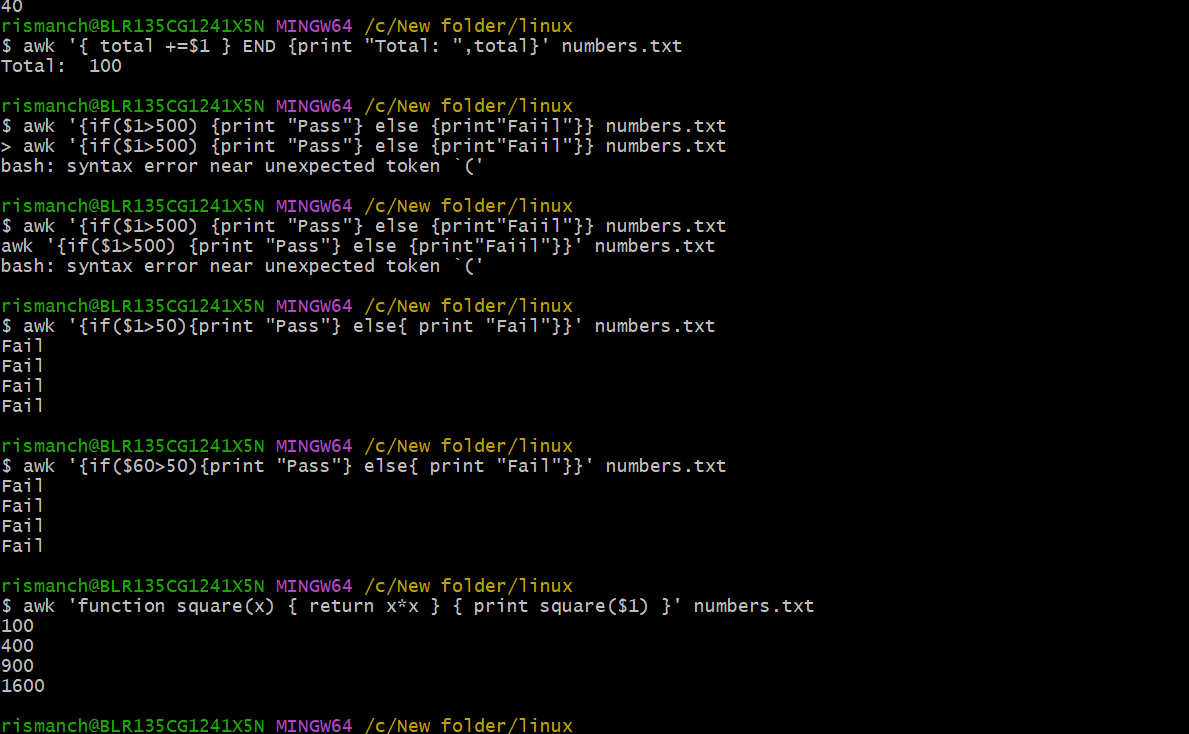
****

**AWK Commands**

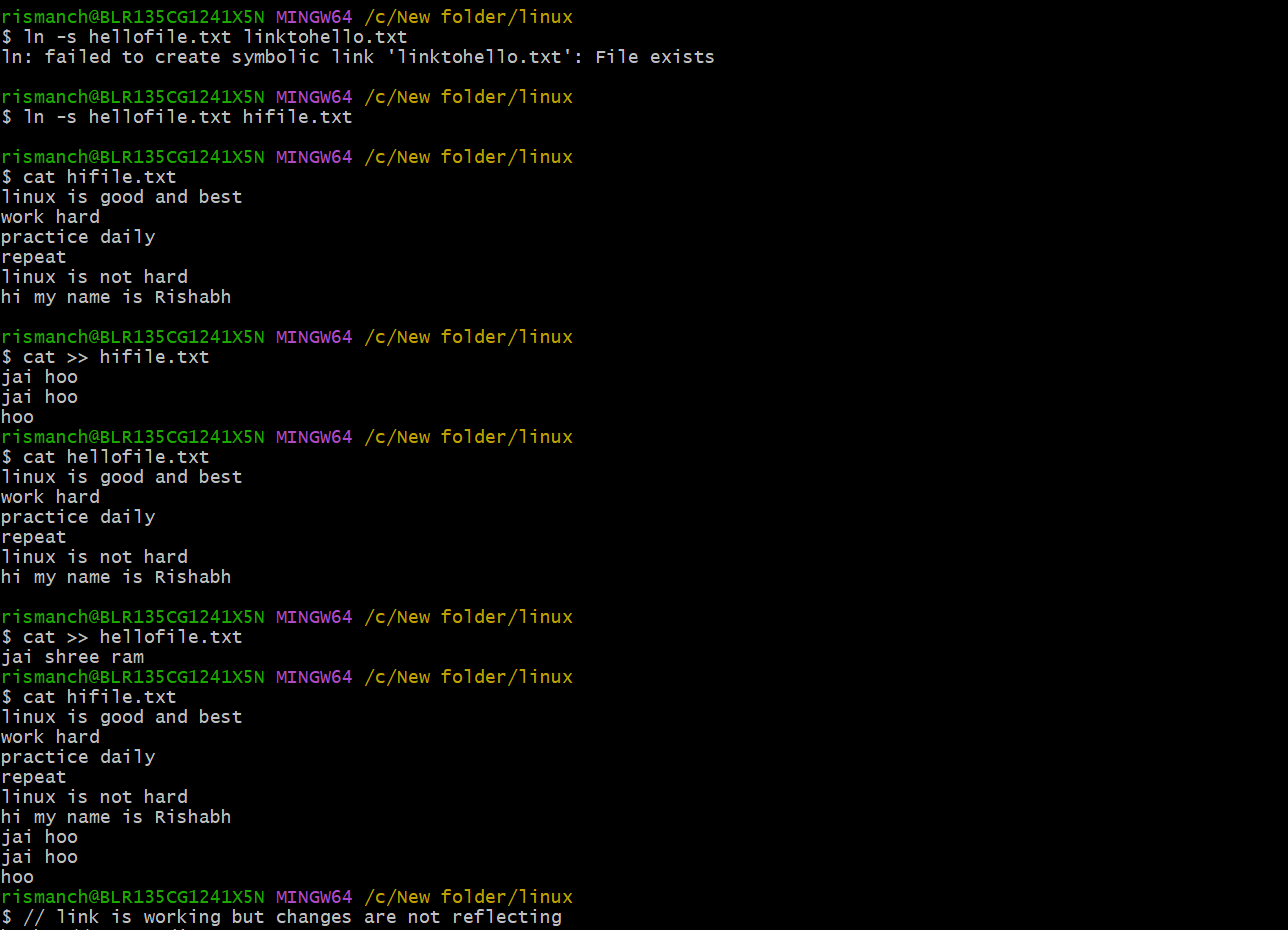
****

****

****

****

**Symbolic link**

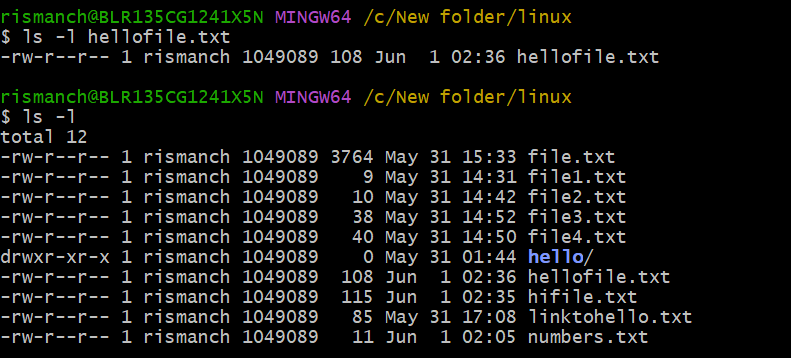
****

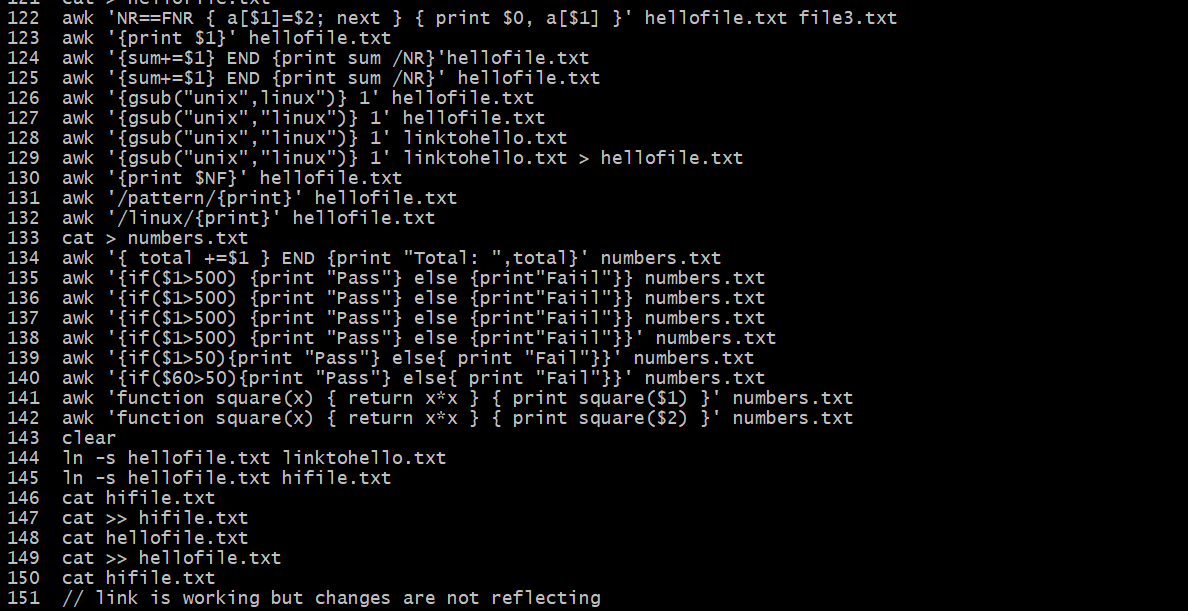
**Owner-group-all & others**

**r- read, w- write, x-execute**

**Task 19:**

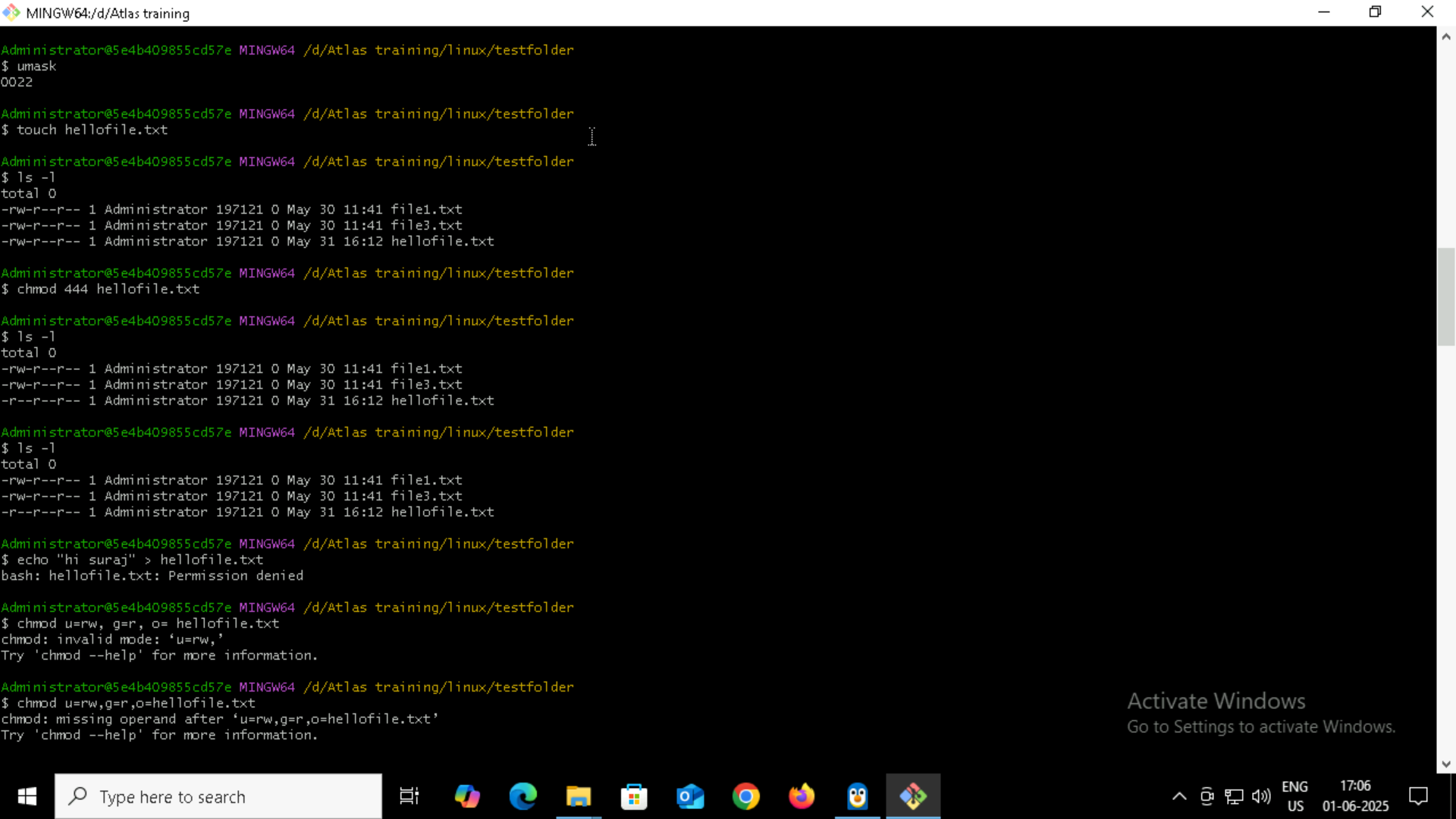
**Owner: rw**

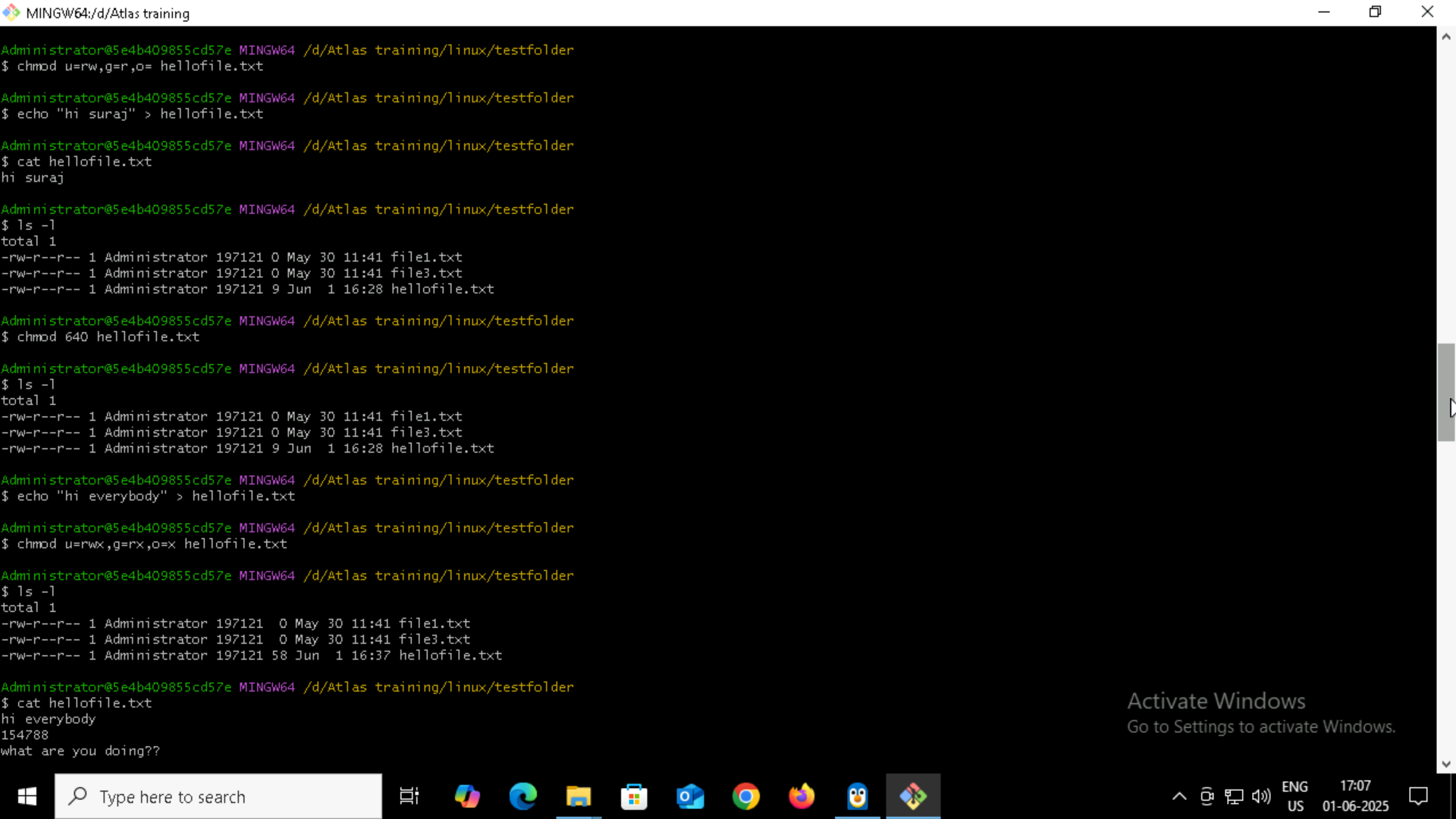
**Group: rw**

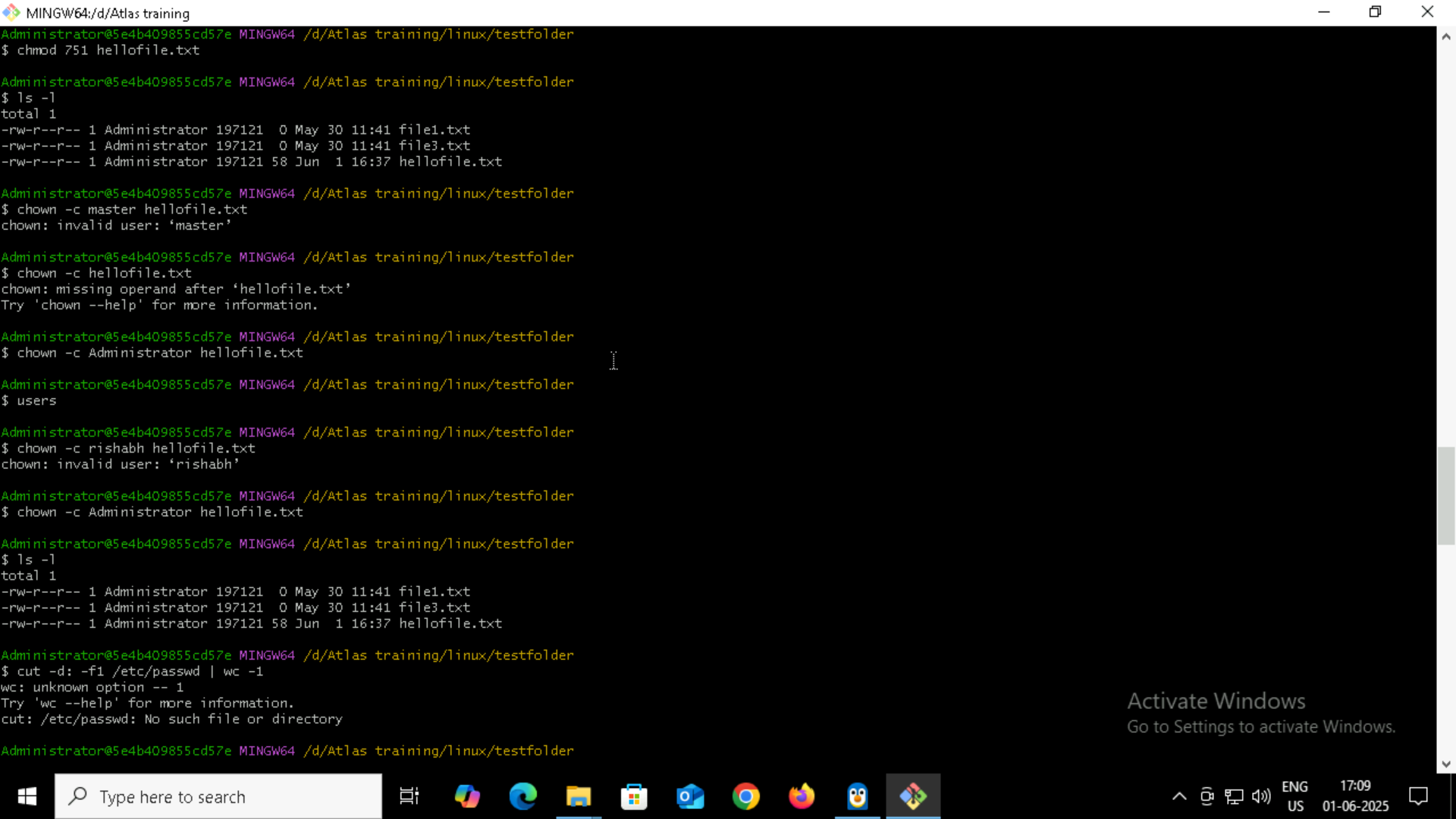
****

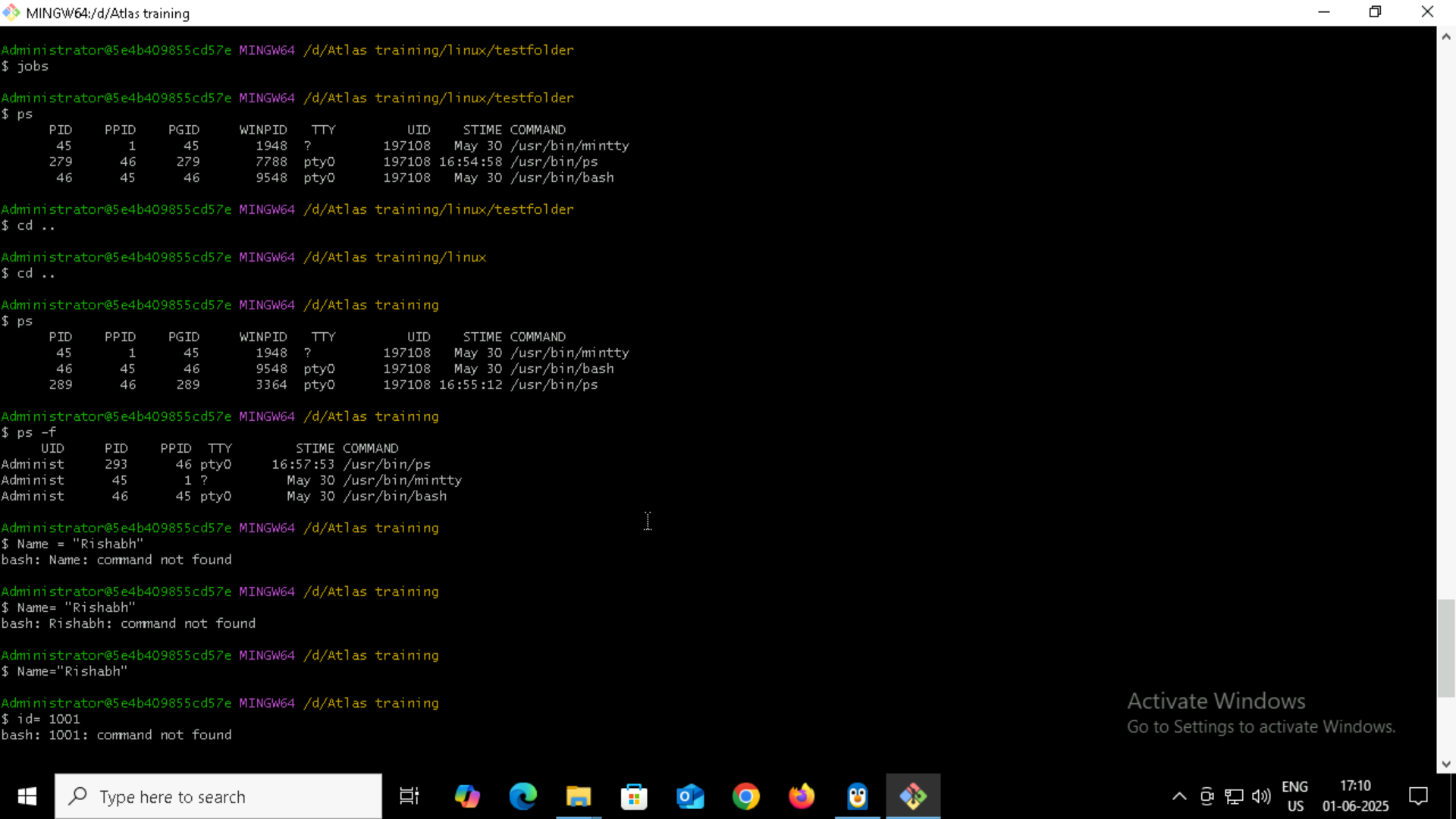
**Task 26 to 35**

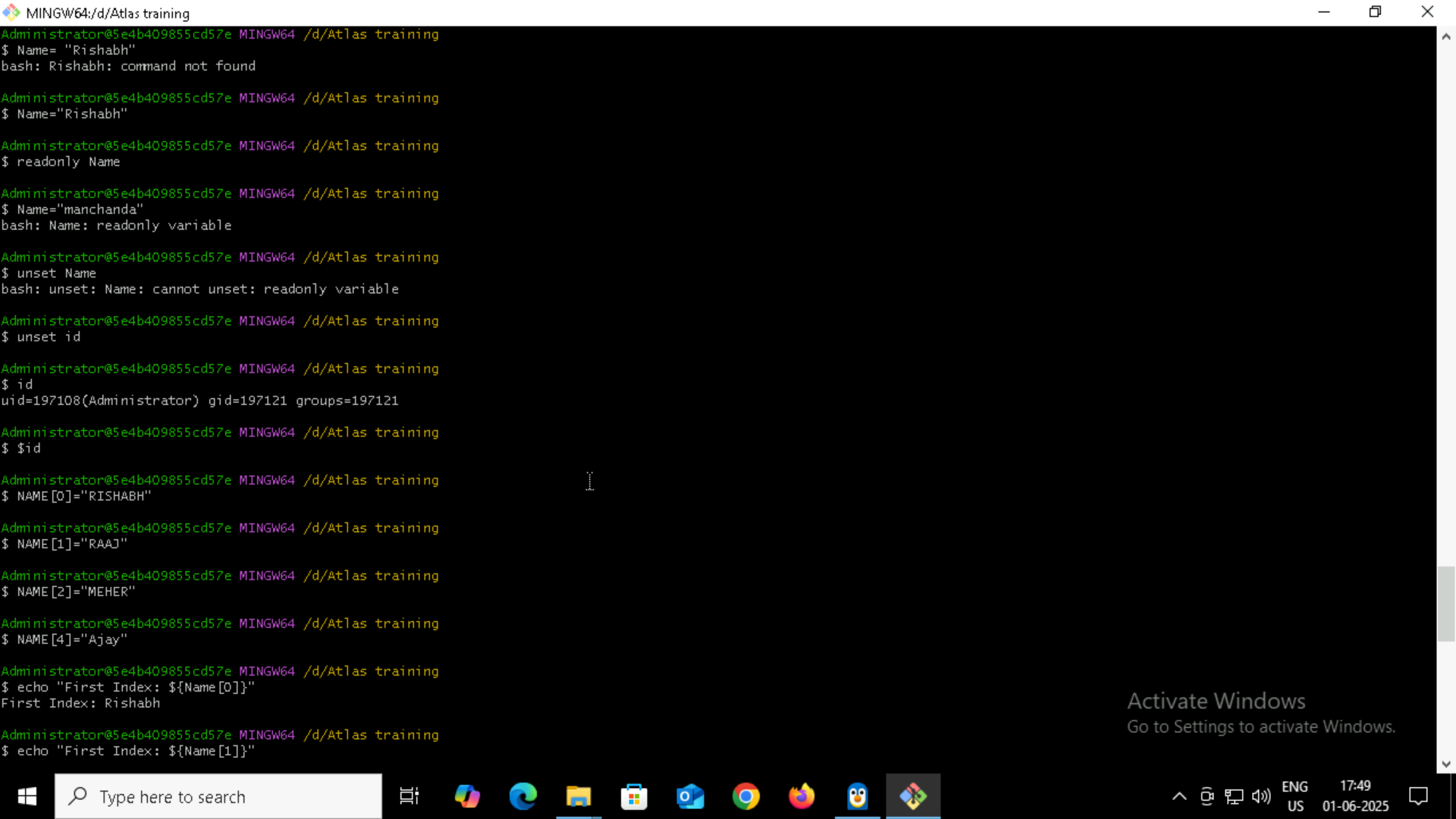
A process is running instance of a program within its own memory space, resources, and execution context.

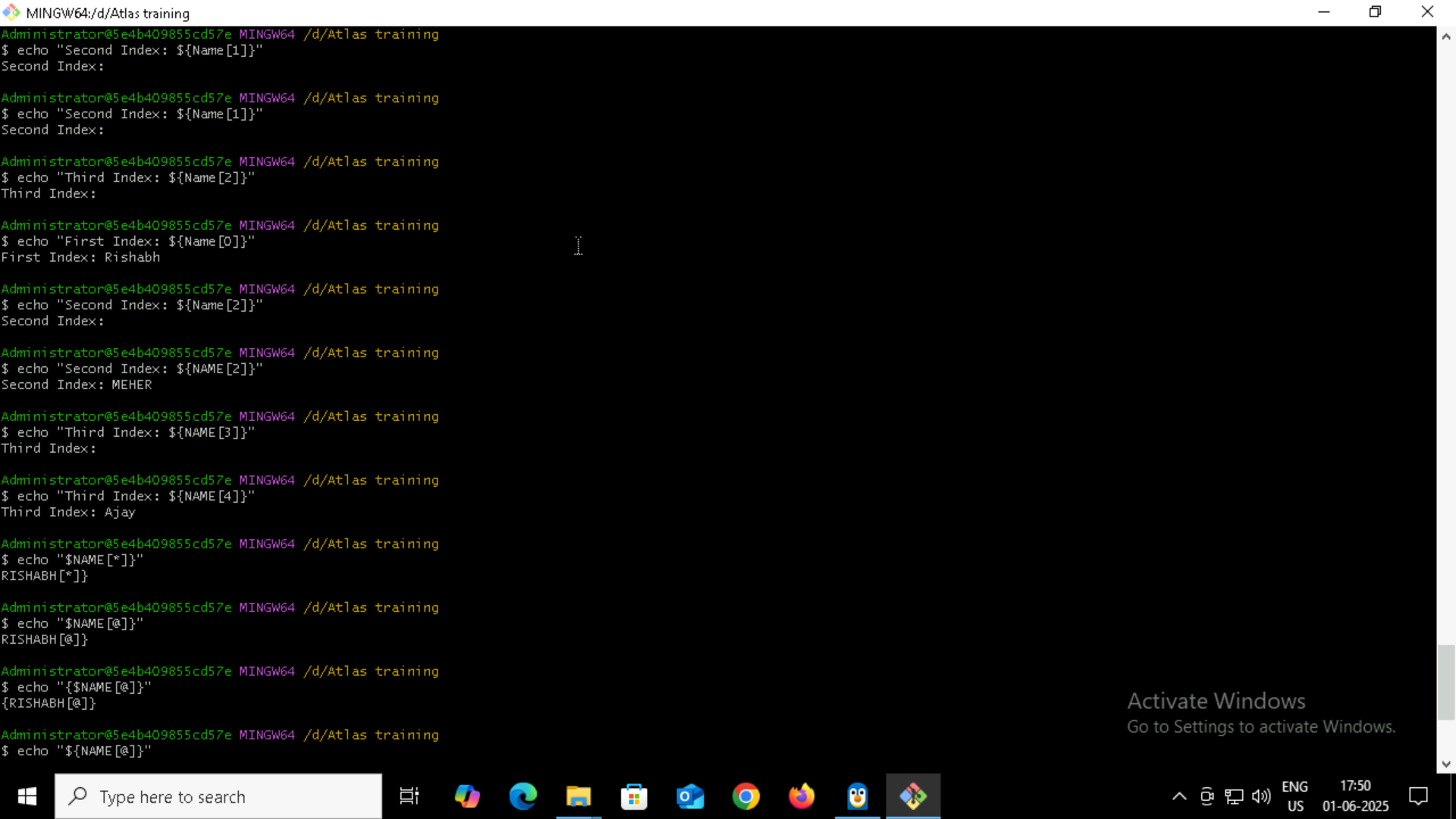
****

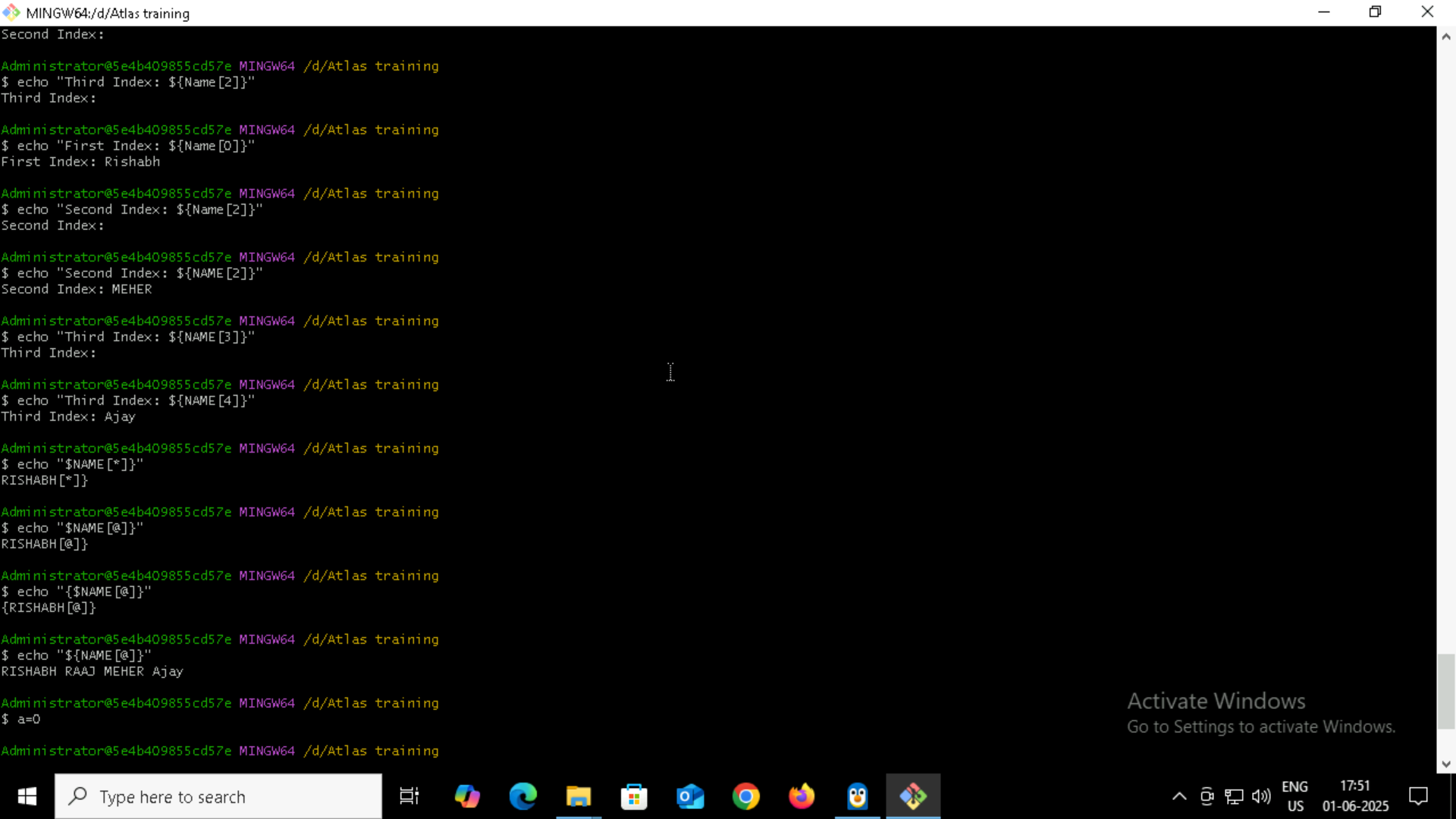
****

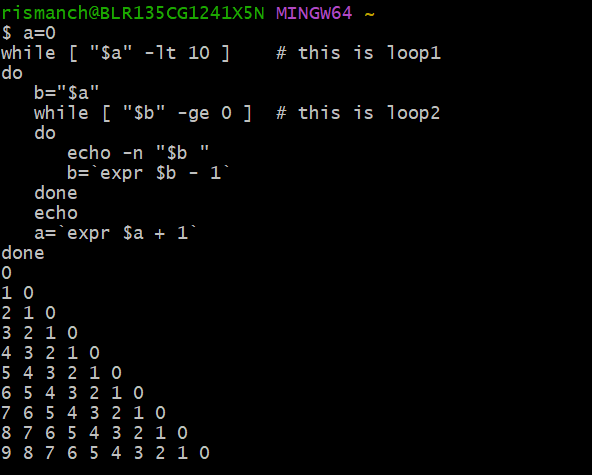
****

****

****

****

****

****