Task 1: RegEX Symbols in Linux List them down with discription?

Ans: . (Dot)

Matches any single character (except newline).

^ (Caret)

Matches the start of a line.

$ (Dollar)

Matches the end of a line.

\* (Asterisk)

Matches zero or more repetitions of the previous character or group.

+ (Plus)

Matches one or more repetitions of the previous character or group.

? (Question mark)

Matches zero or one occurrence of the previous character or group.

[] (Square brackets)

Matches any one character inside the brackets. Example: [abc] matches "a", "b", or "c".

| (Pipe)

Acts as OR. Example: a|b matches "a" or "b".

() (Parentheses)

Groups parts of the pattern for applying operators or capturing.

\ (Backslash)

Escapes special characters to match them literally or introduces special sequences.

Task 2: what is regular expression in linux?

Ans: A regular expression (often abbreviated as regex or regexp) in Linux is a sequence of characters that defines a search pattern. It is used for pattern matching within text, allowing you to search, replace, or manipulate strings based on specific patterns. Regular expressions are powerful tools in Linux for pattern matching and text processing, enabling efficient and flexible manipulation of text data.

Literals: Match exact characters (e.g., abc matches "abc").

Meta-characters:

. (dot): matches any single character.

The kernel in Linux is the core component of the Linux operating system. It acts as a bridge between the hardware of a computer and the software applications that run on it. The kernel manages hardware resources, such as CPU, memory, storage devices, and peripheral devices, and provides essential services to other parts of the operating system and applications.

Key Functions of the Linux Kernel:

ANS:

* Process Management: Controls the creation, scheduling, and termination of processes and threads.
* Memory Management: Handles allocation and deallocation of memory for processes, ensuring efficient use of RAM.
* Device Management: Interfaces with hardware devices through device drivers, enabling communication with peripherals like disks, printers, and network cards.
* File System Management: Manages data storage, retrieval, and organization on disk drives.
* Security and Access Control: Implements permissions and user authentication to secure system resources.
* Networking: Provides networking capabilities for communication between computers and devices.

Task 4: BASH in Lonux full form and Explanation?

ANS: BASH in Linux stands for Bourne Again SHell.

Full Form:

Bourne Again SHell

Explanation:

Bourne: Refers to the original UNIX shell called Bourne Shell (sh), created by Stephen Bourne.

Again: Indicates that BASH is an enhanced, improved version of the original Bourne Shell.

SHell: A command-line interpreter that allows users to interact with the operating system by typing commands.

What is BASH?

BASH is a popular command-line shell used in Linux and many Unix-like systems.

It provides a user interface to interact with the system, run commands, write scripts, and automate tasks.

BASH supports features like variables, loops, functions, command history, and scripting.

In simple words:

BASH is the default text-based interface in Linux that helps you control your computer by typing commands, and it also allows you to create scripts for automating tasks.

Task 5: Now that you know Linux is also an OS like windows.

What do you think the difference between Linux and Windows?

ANS: Linux is free, customizable, and often used by developers and servers.

Windows is user-friendly, widely used on personal computers, and supports many popular apps.

Certainly! Here's a simple comparison between Linux and Windows:

| Feature | Linux | Windows |

|------------------------------ |------------------------------------------------ |----------------------------------------------|

| \*\*Type of OS\*\* | Open Source (Free to use and modify) | Proprietary (Paid, owned by Microsoft) |

| \*\*Source Code\*\* | Open Source | Closed Source |

| \*\*User Interface\*\* | Customizable (various desktop environments) | Standardized (Windows GUI) |

| \*\*Performance\*\* | Usually faster and more efficient for servers | May require more resources |

| \*\*Security\*\* | More secure, less prone to viruses | More targeted by malware, needs antivirus|

| \*\*Compatibility\*\* | Better for programming and servers | Better for gaming and mainstream apps |

| \*\*Software Availability\*\* | Many free and open-source apps | Commercial software, some free apps |

| \*\*Usage\*\* | Popular in servers, programming, tech field | Common for personal and office use |

\*\*In simple words:\*\*

- Linux is free, customizable, and often used by developers and servers.

- Windows is user-friendly, widely used on personal computers, and supports many popular apps.

Task 6: what are the basic componens of Linux? Describe each in details with diagrams

**Basic Components of Linux:**

**1. Kernel**

The **kernel** is the core part of any operating system. It manages hardware resources and provides services for all the other parts of the system. It acts as an intermediary between hardware and software.

* **Responsibilities**:
  + **Process Management**: Handles the execution of processes.
  + **Memory Management**: Allocates and manages memory for processes.
  + **Device Drivers**: Manages input and output devices.
  + **File System Management**: Provides file systems to organize and store data.

**Diagram:**

sql

CopyEdit

+-------------------------------+

| Kernel |

| - Process Management |

| - Memory Management |

| - Device Drivers |

| - File System Management |

+-------------------------------+

↑ ↑ ↑

| | |

Hardware Software

**2. Shell**

The **shell** is a command-line interface (CLI) that allows users to interact with the kernel. It interprets user commands and translates them into system calls for the kernel to execute.

* **Types of Shells**:
  + **Bash** (Bourne Again Shell): Most commonly used shell in Linux.
  + **Zsh**, **Ksh**, **Fish**, etc.
* **Responsibilities**:
  + Accept user input.
  + Interpret and execute commands.
  + Allow scripting (shell scripts).

**Diagram:**

pgsql

CopyEdit

+-------------------------------+

| Shell |

| - Accepts user input |

| - Executes commands |

| - Supports scripting |

+-------------------------------+

↑

|

+----------------------------+

| Kernel (Handles |

| Hardware, Memory, etc.) |

+----------------------------+

**3. System Library**

The **system libraries** are a set of programs that provide functions to applications. They serve as an intermediary between user applications and the kernel.

* **Responsibilities**:
  + Provide functions for user applications (e.g., standard input/output, memory allocation).
  + Facilitate communication between user space and kernel space.
  + Standardize interactions with the system.

**Diagram:**

pgsql

CopyEdit

+-------------------------------+

| System Libraries |

| - Provide system functions |

| - Intermediary between |

| user applications & kernel|

+-------------------------------+

↑

|

+----------------------------+

| User Space |

+----------------------------+

↑

|

+----------------------------+

| Kernel Space |

+----------------------------+

**4. System Utilities**

**System utilities** are tools that perform specific tasks like managing files, monitoring processes, and configuring system settings. They are often invoked via the command line or shell.

* **Examples**:
  + ls, cp, mv, rm (file management)
  + top, ps (process management)
  + chmod, chown (permission management)

**Diagram:**

sql

CopyEdit

+-------------------------------+

| System Utilities |

| - File management |

| - Process management |

| - Permission management |

+-------------------------------+

↑

|

+----------------------------+

| Kernel Space |

+----------------------------+

**5. File System**

The **file system** in Linux manages how data is stored, retrieved, and organized. It defines the structure in which files and directories are arranged.

* **Common File Systems**:
  + **Ext4** (Fourth Extended File System)
  + **XFS**
  + **Btrfs**
* **Responsibilities**:
  + Organizing files in directories.
  + Handling file permissions.
  + Providing access to files for users and programs.

**Diagram:**

pgsql

CopyEdit

+-------------------------------+

| File System |

| - Organizes files |

| - Manages file permissions |

+-------------------------------+

↑

|

+----------------------------+

| Storage Devices |

| (HDD, SSD, etc.) |

+----------------------------+

**6. User Space**

**User space** is where all the user-level applications and processes run, outside of the protected kernel space. The user space includes all of the utilities, libraries, and programs that we interact with.

* **Responsibilities**:
  + Run user-level programs.
  + Maintain user-specific data and processes.

**Diagram:**

sql

CopyEdit

+-------------------------------+

| User Space |

| - Applications |

| - User processes |

+-------------------------------+

↑

|

+----------------------------+

| Kernel Space |

+----------------------------+

**7. Hardware**

The **hardware** represents the physical devices that make up the system (e.g., CPU, memory, storage devices).

* **Responsibilities**:
  + Perform computational tasks.
  + Store data.
  + Provide input/output interfaces.

Task 7: is it legal to edit Kernal? when do you think we have to in case?

Yes, **it is legal** to edit and modify the Linux kernel, and it is actually a major feature of the open-source model that Linux follows. The Linux kernel is released under the **GNU General Public License (GPL) version 2**. This license permits users to:

* **Modify the kernel code**: You are allowed to edit and modify the kernel to fit your needs.
* **Redistribute modified versions**: You can distribute the modified version, provided you adhere to the terms of the GPL, including sharing the source code for any modifications you make.
* **Commercial Use**: You can use the kernel for commercial purposes, but again, any modified version you distribute must be accompanied by the same rights to modify and share the source code.
* **Freedom to use, modify, and distribute**: The license ensures that you can freely use the kernel in any way you want and also allows modifications.
* **Share modifications**: If you redistribute a modified version of the kernel, you must make the source code of your modifications available to others under the same GPL terms.
* **No warranty**: The kernel comes without any warranty, and you are using and modifying it at your own risk.

This model encourages **collaboration and innovation**, as developers around the world are allowed to contribute to the Linux kernel.

Task 8: What is LILO? Explain

ANS; **LILO (LInux LOader)** is a **boot loader** for Linux systems. It is a program that is responsible for loading the operating system when the computer starts up.

Here’s a simple breakdown:

1. **Boot Loader**: LILO is the first program that runs when you power on your computer. It loads the Linux kernel into memory and starts the system.
2. **What LILO Does**:
   * It shows you a **menu** to choose which operating system to boot (if you have more than one installed, like Linux and Windows).
   * It **loads** the kernel (the core part of the Linux OS) into memory to start the system.
3. **Where LILO is Stored**: It is usually installed in the **MBR (Master Boot Record)** of the hard drive or in the **boot sector**.
4. **How It Works**:
   * When the computer starts, LILO is the first thing that runs.
   * It gives you a choice to boot into Linux or another operating system.
   * After selecting an OS, it loads the kernel into memory and passes control to it.

### Key Points:

* **LILO** stands for **LInux LOader**.
* It **loads the Linux kernel** during system startup.
* It can show a **menu** to select different operating systems in a dual-boot setup.
* LILO is **simple**, but has been largely replaced by **GRUB** (GRand Unified Bootloader) in modern systems.

Task 9: **What is a Shell? How Many Shells Are There and What Are They?**

#### ANS: **1. What is a Shell?**

In Linux (and Unix-like systems), the **shell** is a **command-line interface (CLI)** that allows users to interact with the operating system by typing commands. The shell acts as a bridge between the **user** and the **kernel** (the core part of the OS), translating the user’s commands into system calls that the kernel can execute.

**Key Functions of a Shell**:

* **Command Interpreter**: It interprets and executes the commands you type.
* **File Management**: It allows you to manage files and directories (e.g., creating, deleting, copying files).
* **Program Execution**: It can execute programs and scripts.
* **Scripting**: It supports **shell scripting**, which allows users to automate tasks by writing scripts.

### ****2. Types of Shells and Their Descriptions****

There are several different types of shells, each with its own features and characteristics. Below are the most common shells used in Linux:

#### **a. Bourne Shell (sh)**

* **Description**: The **Bourne Shell** (abbreviated as sh) is the original Unix shell developed by **Stephen Bourne**. It was the default shell for many years and is the foundation for most other shells.
* **Key Features**:
  + Basic scripting capabilities.
  + Simple syntax.
* **Common Usage**: Historically used as the default shell for many Unix systems.

#### **b. Bourne Again Shell (Bash)**

* **Description**: **Bash** (Bourne Again Shell) is the most widely used shell today. It is an **improved version** of the Bourne Shell (sh), developed by **Brian Fox** for the GNU Project. It's the default shell on many Linux distributions.
* **Key Features**:
  + **Command-line editing**: You can edit commands before executing them.
  + **Tab completion**: Automatically completes file names and commands.
  + **History feature**: Keeps a history of commands you've typed.
  + **Scripting capabilities**: Advanced scripting and automation features.
* **Common Usage**: Default shell for most Linux distributions.

#### **c. C Shell (csh)**

* **Description**: The **C Shell** (csh) was developed by **Bill Joy**. It has a syntax that is more similar to the **C programming language**, which makes it easier for C programmers to use.
* **Key Features**:
  + **C-like syntax**: Its commands resemble C programming constructs.
  + **Job control**: Allows you to suspend, resume, and background tasks.
  + **History and aliases**: Supports command history and aliases.
* **Common Usage**: Used mainly by developers who are familiar with C programming.

#### **d. TC Shell (tcsh)**

* **Description**: The **TC Shell** (tcsh) is an enhanced version of the C Shell (csh). It includes additional features such as **command-line editing** and **better history management**.
* **Key Features**:
  + **Command-line editing**: You can edit commands with vi or emacs keybindings.
  + **Improved history**: More advanced history features compared to csh.
  + **Better scripting**: Additional features that make scripting easier.
* **Common Usage**: Often used by developers who need C-like syntax but with enhanced features.

#### **e. Korn Shell (ksh)**

* **Description**: The **Korn Shell** (ksh) was developed by **David Korn**. It combines features of the Bourne Shell (sh) with C Shell (csh) features and introduces new functionalities.
* **Key Features**:
  + **Command-line editing**: Supports both Emacs and vi editing modes.
  + **Job control**: Allows you to control background and suspended processes.
  + **Array variables**: Supports arrays, which is useful for scripting.
  + **Scripting**: Provides advanced scripting capabilities and powerful built-in features.
* **Common Usage**: Preferred in enterprise environments and by professional system administrators.

#### **f. Z Shell (zsh)**

* **Description**: **Zsh** (zsh) is an advanced shell that incorporates features from **Bash**, **Ksh**, and **Csh**. It is highly customizable and offers advanced features not found in other shells.
* **Key Features**:
  + **Improved tab completion**: More powerful than in Bash.
  + **Advanced globbing**: Allows more flexible file matching.
  + **Customizability**: Highly configurable with many themes and plugins.
  + **Scripting**: Strong scripting features, similar to Bash and Ksh.
* **Common Usage**: Gaining popularity among developers, particularly in the development of **configurable environments**.

#### **g. Fish Shell (fish)**

* **Description**: The **Fish Shell** (Friendly Interactive Shell) is designed to be **user-friendly**, with an emphasis on being easy to use and visually appealing.
* **Key Features**:
  + **Autocompletion**: Extremely smart and helpful autocompletion.
  + **Syntax highlighting**: Highlights commands and syntax in real-time.
  + **User-friendly**: Designed for ease of use with helpful error messages and suggestions.
* **Common Usage**: Favored by users who want an easy-to-use shell with modern features, particularly for interactive use.

Task 10: What is swap space?

ANS: **Swap space** is a portion of your computer's storage (usually on the hard disk or SSD) that is used when the physical **RAM** (Random Access Memory) is full. It acts as **virtual memory**, allowing the operating system to move inactive processes or data from RAM to swap space, freeing up RAM for active processes.

#### **How Swap Space Works:**

1. **RAM (Physical Memory)**: RAM is fast but limited in size. When you run out of RAM, your system needs to make room for more active processes.
2. **Swap Space (Virtual Memory)**: When your RAM is full, the kernel (the core of the operating system) will move inactive data or processes to the swap space on the disk. This process is called **paging** or **swapping**.
3. **When It's Used**: The swap space is only used when there is no available RAM left to store active programs or data. It acts as an extension of RAM, allowing the system to continue running.

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

#### Ans: **1. What is Mount?**

In Linux, **mounting** refers to the process of making a storage device (such as a hard drive, USB drive, CD/DVD, or network drive) accessible to the system by attaching it to a specific location in the **file system hierarchy** (called a **mount point**). Once mounted, the device or file system becomes a part of the system’s overall file structure, allowing users to access its contents.

#### **Key Concepts**:

* **Mount Point**: This is the directory in the Linux file system where the storage device or file system will be attached. It could be something like /mnt, /media, or any directory.
* **File System**: A file system is the structure that organizes data on the storage device. Common file systems include **ext4**, **NTFS**, **FAT32**, and **exFAT**.

**Example**:

* When you plug in a USB drive, you mount it to a directory (e.g., /media/usb) so you can access the files on the drive.

### ****2. How to Mount a File System in Linux****

To mount a file system in Linux, you typically use the mount command.

#### **Basic Syntax of the mount command:**

bash

CopyEdit

mount [options] <device> <mount\_point>

**Create a Mount Point**:  
If you don't already have a directory where you want to mount the device, create one using the mkdir command.

**Mount the Device**:  
Now, use the mount command to mount the device (e.g., /dev/sdb1) to the mount point (/mnt/myusb).

 After running this command, the files on /dev/sdb1 will be accessible at /mnt/myusb.

 **Verify the Mount**:  
You can verify that the device is mounted correctly by using the df or mount command:

### ****3. How to Unmount a File System in Linux****

To **unmount** a file system means to safely disconnect it from the system. It is important to unmount a device before physically removing it to prevent data loss or corruption.

#### **Unmount Command:**

The command used to unmount a file system is umount (note that it is spelled **without** an "n" after the "u").

* **<mount\_point>**: The directory where the device is mounted (e.g., /mnt/myusb).
* **<device>**: The device that you want to unmount (e.g., /dev/sdb1).

#### **Steps to Unmount a File System**:

1. **Unmount the File System**:  
   To unmount the device mounted at /mnt/myusb, use the umount command:

**Verify the Unmount**:  
To verify that the device has been unmounted, you can check the output of the df or mount command:

Task 12: **What is the chmod Command? How to Use It?**

Ans: The chmod command in Linux is used to **change the permissions** of files and directories. Permissions determine who can read, write, or execute a file. chmod stands for **"change mode"** and is an essential command for managing access control to files and directories.

The chmod command can be used in **two modes**:

1. **Symbolic mode**: Uses letters to represent users and permissions.
2. **Numeric (octal) mode**: Uses numbers to represent permissions.

#### **A. Symbolic Mode** (Using Letters)

In symbolic mode, you specify the permission you want to modify by using letters:

* **User categories**:
  + **u**: User (Owner)
  + **g**: Group
  + **o**: Others
  + **a**: All users (Owner + Group + Others)
* **Permissions**:
  + **r**: Read
  + **w**: Write
  + **x**: Execute
* **Operations**:
  + **+**: Adds the specified permission.
  + **-**: Removes the specified permission.
  + **=**: Sets the specified permission and removes others.

#### **B. Numeric Mode** (Using Numbers)

In numeric mode, you assign **numbers** to permissions. Each permission (read, write, execute) is represented by a number:

* **Read (r)** = **4**
* **Write (w)** = **2**
* **Execute (x)** = **1**

The numbers are **added together** for each category:

* **Owner** = First digit
* **Group** = Second digit
* **Others** = Third digit

**Examples of Numeric Permissions**:

* **7** = rwx (4 + 2 + 1)
* **6** = rw- (4 + 2)
* **5** = r-x (4 + 1)
* **4** = r-- (4)
* **3** = wx- (2 + 1)
* **2** = w-- (2)
* **1** = x-- (1)
* **0** = --- (no permissions)

### ****Conclusion****

* **chmod** is a powerful command used to modify file permissions in Linux.
* You can change permissions using **symbolic mode** (using letters like r, w, x) or **numeric mode** (using numbers like 7, 6, 5).
* **Special permissions** like setuid, setgid, and the **sticky bit** can be used to grant more advanced permissions for files and directories.
* Proper file permissions are crucial for system security, ensuring only authorized users can access or modify important files.

Task 13: **Can You Add a New User Account? Create a New User in Different Ways and Paste as**

Ans: In Linux, you can create new user accounts using several methods. The most common methods involve using the useradd command and the adduser command. Both commands are used to create user accounts.

For ex: sudo useradd john , sudo adduser john

Task 14: **Can You Change the Password of a User? How Do You Do That?**

Ans: Yes, we can change the password of a user in Linux using several commands. The most commonly used command for changing a user's password is passwd.

For ex:

* $ passwd
* Changing password for user john.
* Current password:
* New password:
* Retype new password:
* passwd: password updated successfully

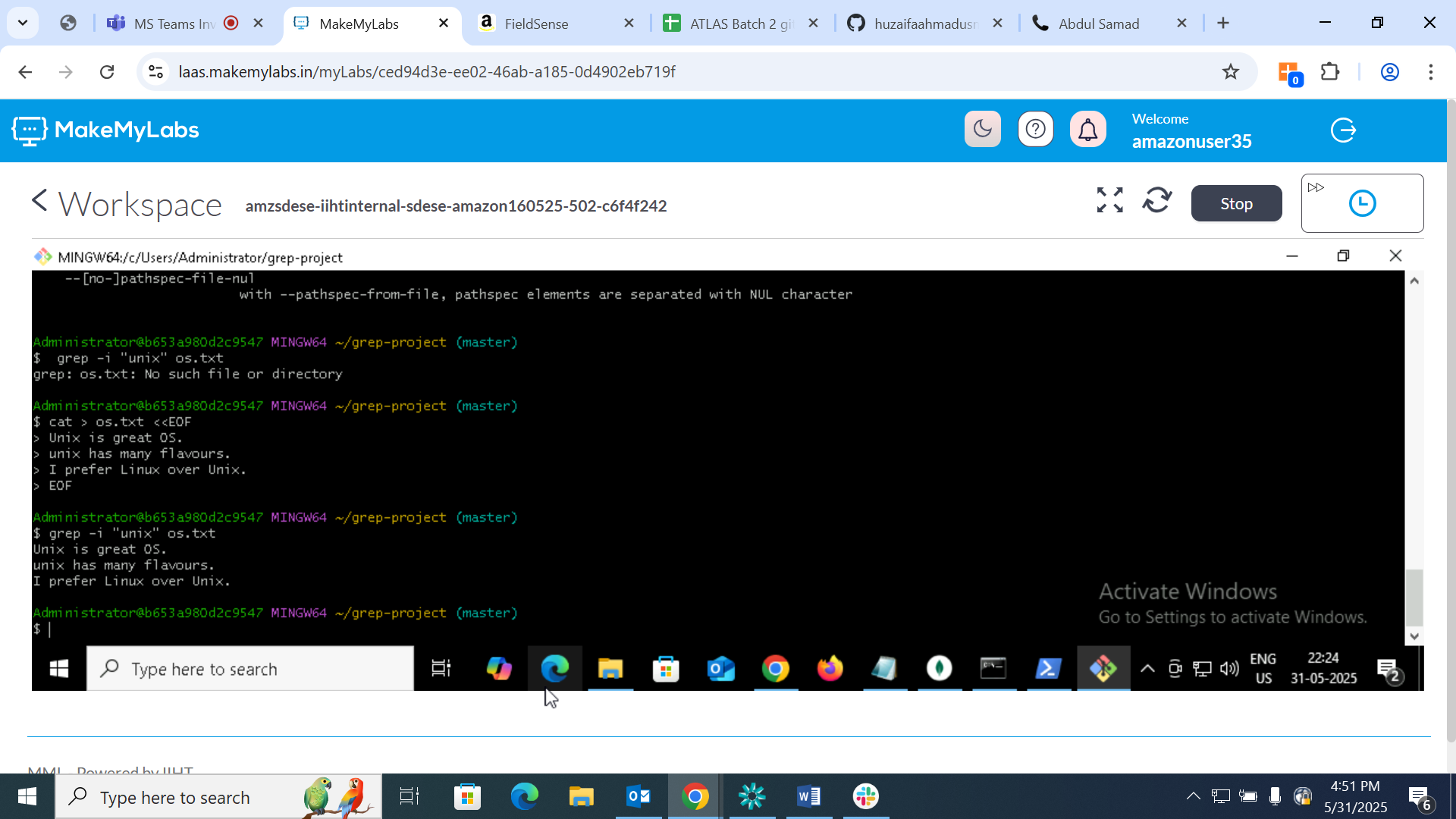
Task 15: **What is the Difference Between a Process and a Thread?**

Ans:  **Process**:  
A **process** is an **instance of a running program**. It has its own memory space, system resources, and a **lifetime**. A process is the container that holds an execution environment, including program code, data, stack, and registers. Processes are isolated from each other, meaning that one process cannot directly access the memory or data of another process without specific mechanisms like inter-process communication (IPC).

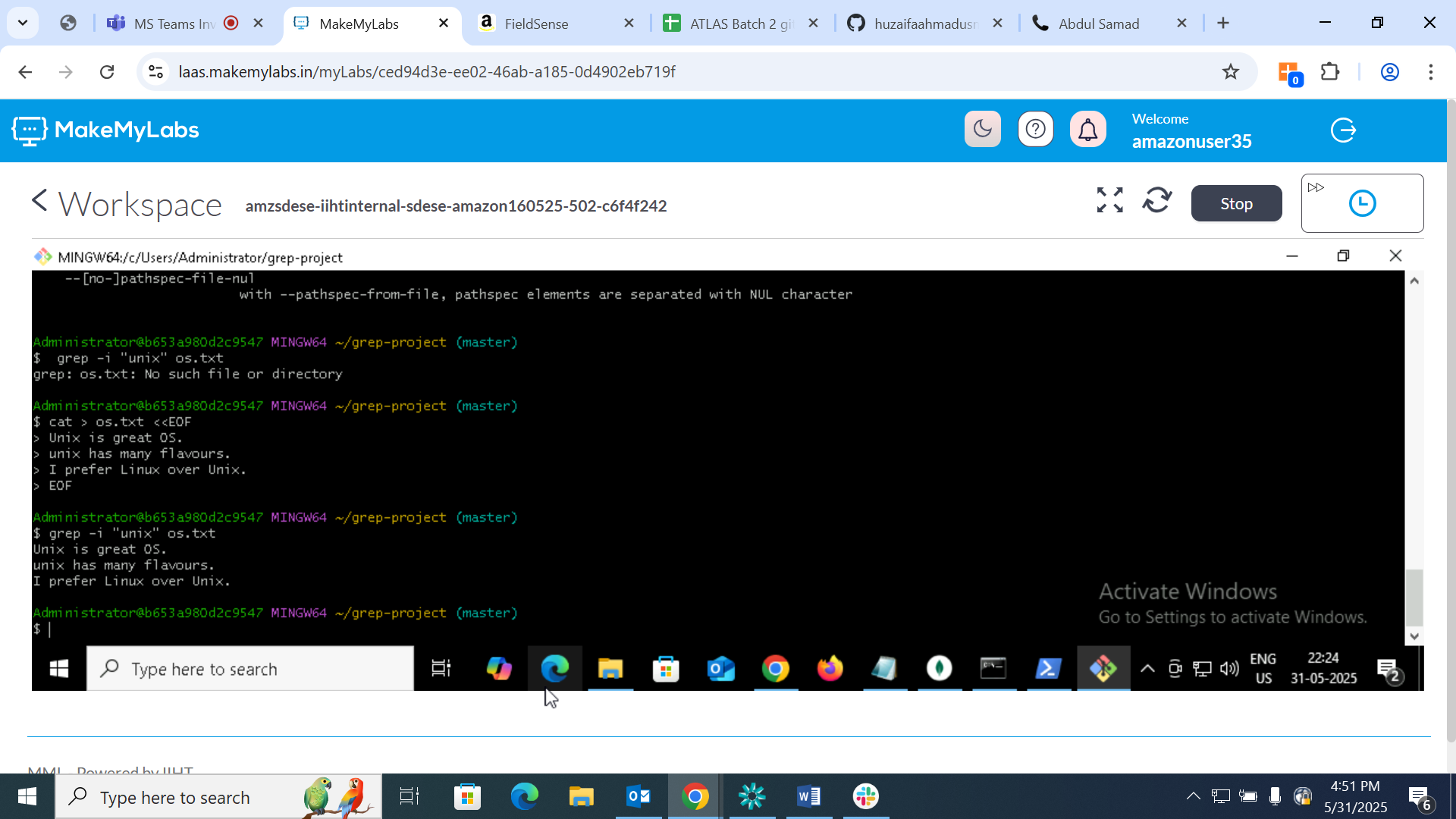
 **Thread**:  
A **thread** is the **smallest unit of execution** within a process. Every process can have **multiple threads**. Threads within the same process share the same memory space and resources. A thread includes a program counter, a stack, and registers. Threads allow a process to perform multiple tasks concurrently (in parallel) within the same process context.

Task 16: Doc 14 Linux Grep commands in docs to study folder

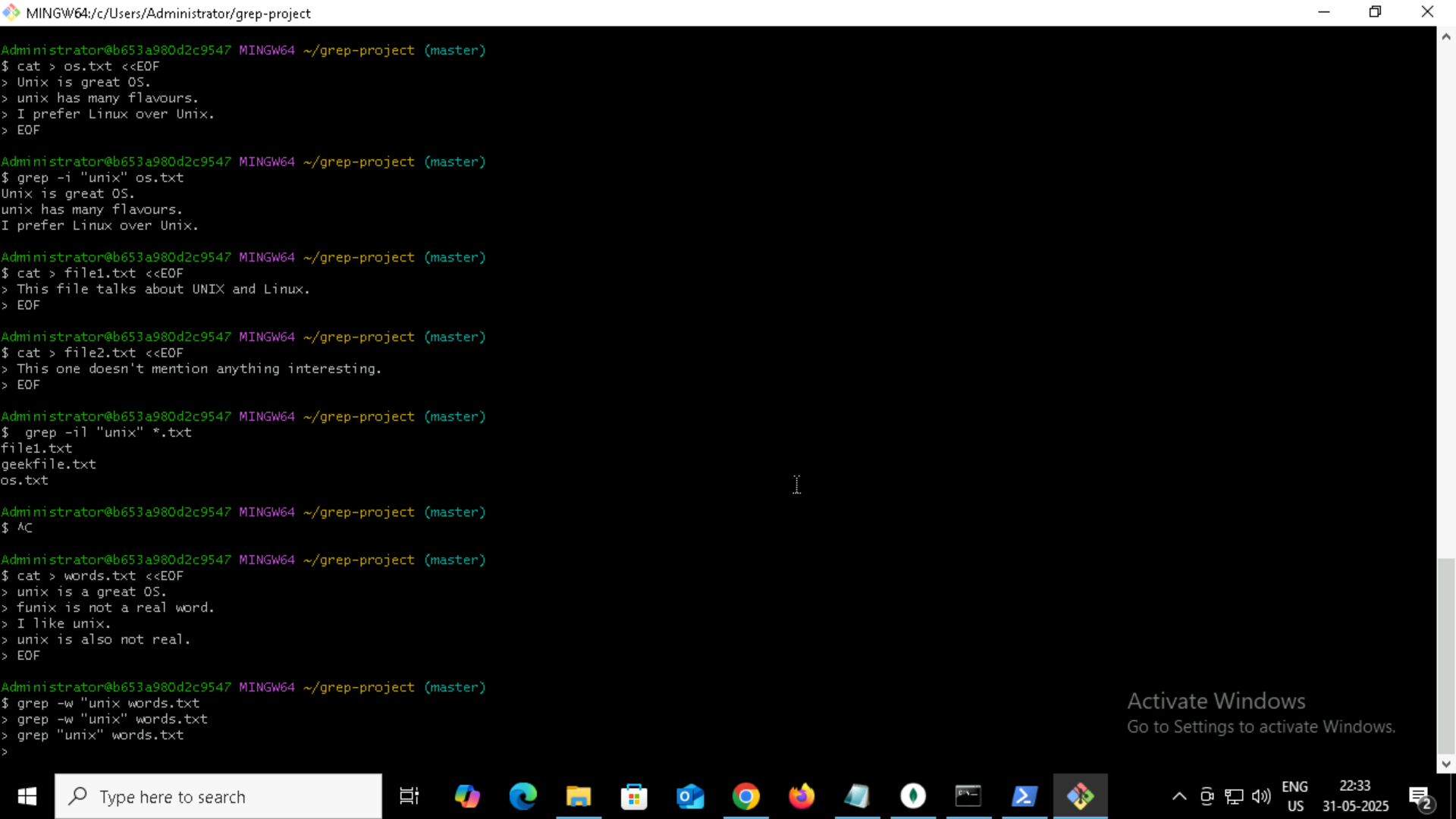
Task 18: How to check file access permission in Linux (promp commnad)



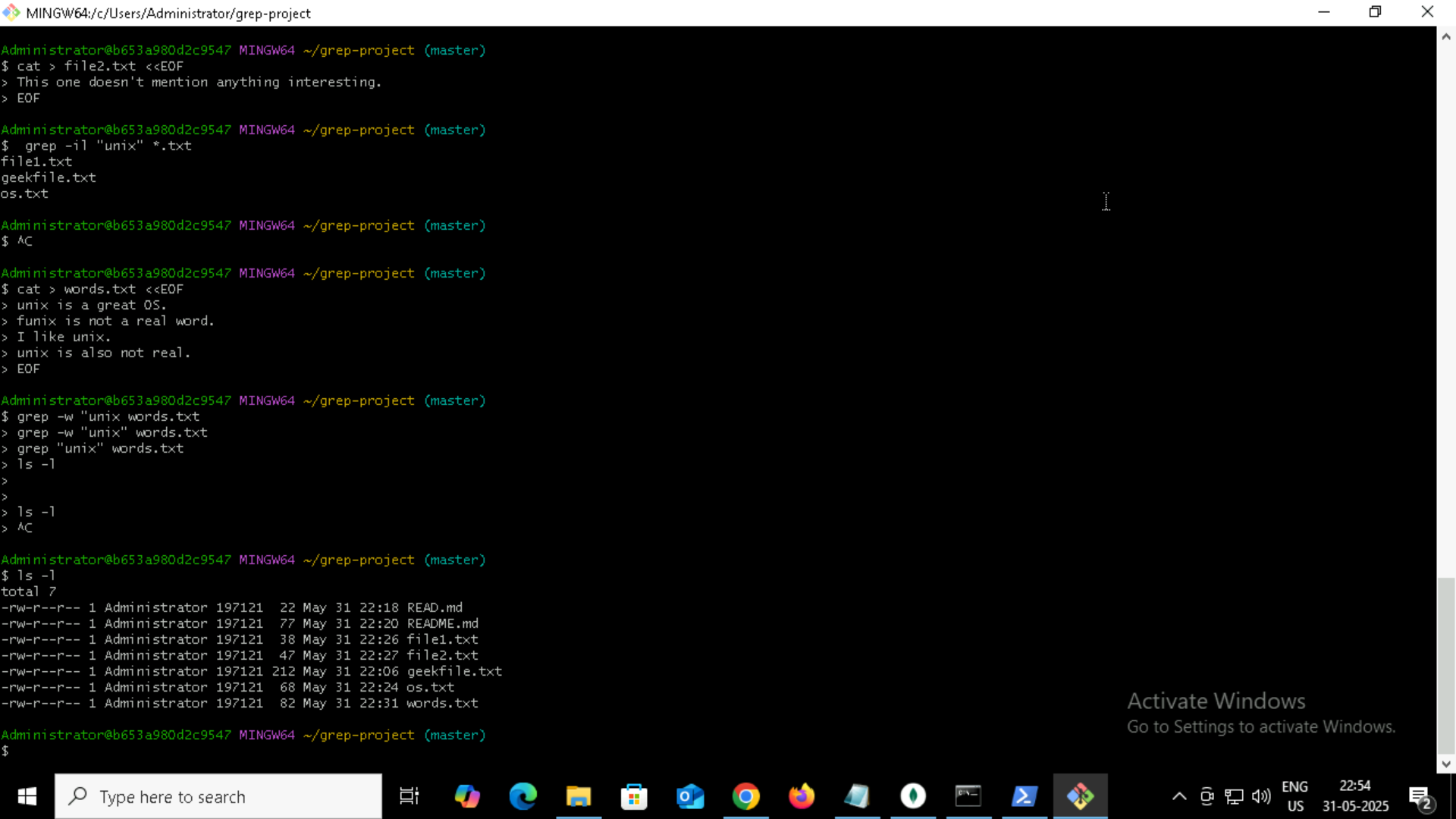
Now display the file name that matches pattern using grep



**Checking for the whole world in a file using Grep**



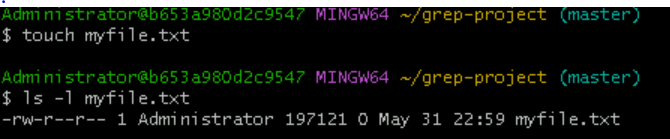
**Task 18: How to check file access permission in Linux**



**Task 19: What are default permission for a new file ?**

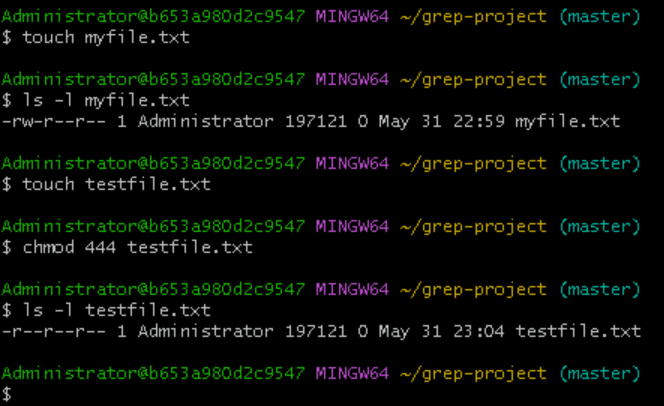
**Pls find out for**

* **owner ?**
* **group?**
* **All and others?**



**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**



**Task 21: Can u change the file persmission to match the following:**

* **Owner:read and write**
* **Group:read**
* **Other: no permissions(none)**



**Task 22 what is the command for changing the file permission to -rw r-----?**



**Task 23 change chmod excercises permission to rwxr-x—x**

**Change file permissions to match the following:**

* **Owner read write execute**
* **Group: read and execute**
* **Other: execute**

