Task 1:

RegEX Symbols in linux

List them down with description

| **Symbol** | **Meaning** | **Supported in [[ =~ ]]?** | **Supported in grep/awk/sed?** | **Example Command** |
| --- | --- | --- | --- | --- |
| . | Any character except newline | ✅ | ✅ | [[ "abc" =~ a.c ]] && echo "Match" |
| ^ | Start of string | ✅ | ✅ | [[ "abc" =~ ^a ]] && echo "Starts with a" |
| $ | End of string | ✅ | ✅ | [[ "abc" =~ c$ ]] && echo "Ends with c" |
| [...] | Match any one character in set | ✅ | ✅ | [[ "a2c" =~ a[0-9]c ]] && echo "Match" |
| [^...] | Match any character NOT in set | ✅ | ✅ | [[ "abc" =~ a[^0-9]c ]] && echo "Match" |
| \* | Zero or more of previous token | ✅ | ✅ | [[ "aaaa" =~ a\* ]] && echo "Match" |
| + | One or more of previous token | ✅ (Bash ≥ 3.2) | ✅ (-E) | [[ "aaa" =~ a+ ]] && echo "Match" |
| ? | Zero or one of previous token | ✅ | ✅ | [[ "ab" =~ ab? ]] && echo "Match" |
| {n} | Exactly n repetitions | ❌ | ✅ (-E) | `echo 111 |
| {n,} | n or more repetitions | ❌ | ✅ (-E) | `echo 1111 |
| {n,m} | Between n and m repetitions | ❌ | ✅ (-E) | `echo 111 |
| (...) | Grouping | ✅ | ✅ | `[[ "cat" =~ (cat |
| ` | ` | Alternation (OR) | ✅ | ✅ |
| \ | Escape character | ✅ | ✅ | [[ "." =~ \. ]] && echo "Dot matched" |
| \d | Digit ([0-9]) | ❌ | ✅ (grep -P) | `echo 123 |
| \w | Word char ([a-zA-Z0-9\_]) | ❌ | ✅ (grep -P) | `echo name\_1 |
| \s | Whitespace | ❌ | ✅ (grep -P) | `echo " " |
| (?=...) | Positive lookahead | ❌ | ✅ (grep -P) | `echo "abc123" |
| (?<=...) | Positive lookbehind | ❌ | ✅ (grep -P) | `echo "123abc" |

Task 2:

What are the imp features of Linux os ?

| **Feature** | **Description** |
| --- | --- |
| **Open Source** | Freely available code that anyone can view, modify, and share. |
| **Multiuser Capability** | Supports multiple users working simultaneously without conflicts. |
| **Multitasking** | Can run many processes at the same time efficiently. |
| **Security** | Strong protection via permissions, authentication, and firewalls. |
| **Stability and Reliability** | Runs for long periods without crashing or rebooting. |
| **Portability** | Works on a variety of hardware platforms and devices. |
| **Powerful Command-Line Interface (CLI)** | Robust shell tools for control, scripting, and automation. |

Task 3:

WHAT IS Kernal and can you explain its functions

### **What is Kernel?**

* The **kernel** is the core part of an operating system.
* It manages communication between hardware and software.
* It controls system resources and operations.

### **Functions of Kernel**

1. **Process Management** – Handles creation, scheduling, and termination of processes.
2. **Memory Management** – Manages RAM allocation and swapping.
3. **Device Management** – Controls and communicates with hardware devices via drivers.
4. **File System Management** – Manages files, directories, and storage devices.
5. **Security and Access Control** – Enforces permissions and user authentication.
6. **System Calls Handling** – Provides an interface for programs to interact with hardware.

Task 4:

What is BASH? Full form with explaination

**Bash** (Bourne Again SHell) is a command-line shell and scripting language.

It allows users to interact with the Linux/Unix operating system by typing commands.

Bash can also run scripts to automate tasks.

It’s the default shell on many Linux distributions.

Task 5:

What is the diffrenece between window and linux

| **#** | **Feature** | **Linux** | **Windows** |
| --- | --- | --- | --- |
| 1 | **Source Code** | Open-source and free | Proprietary but well-maintained by Microsoft |
| 2 | **User Interface (UI)** | Highly customizable with multiple environments | Consistent and user-friendly GUI |
| 3 | **Command Line** | Powerful and essential (Bash, Zsh, etc.) | Improved with PowerShell, easier for beginners |
| 4 | **Software Availability** | Great open-source ecosystem | Excellent commercial software support |
| 5 | **Security** | Very secure by design | Good security with frequent updates |
| 6 | **Performance** | Lightweight, great for older machines | Optimized for modern hardware |
| 7 | **System Updates** | Manual and flexible | Automated and integrated |
| 8 | **Hardware Support** | Broad but may need manual drivers | Plug-and-play, great driver support |
| 9 | **Community & Support** | Large open-source community, forums | Strong vendor support and professional help |
| 10 | **Use Cases** | Ideal for servers, development, and scripting | Excellent for business, productivity, and home use |

Task 6:

Basic components of Linux are Kernel (core system), Shell (command interpreter), File System (data management), System Libraries (reusable code), and System Utilities (tools).

Task 7:

Yes, it is legal to edit the Linux kernel under the GNU GPL license, which allows modification and redistribution of the source code.

Task 9:

Shell is a command-line interface to interact with the OS. Common shells: Bash, sh, csh, ksh, zsh, fish.

Task 10:

Swap space is disk space used as virtual memory when RAM is full to avoid system crashes.

Task 11:

Mounting attaches a filesystem to the directory tree using `mount`; unmounting detaches it using `umount`.

Task 12:  
 chmod changes file or directory permissions in Linux. Use: chmod 755 file.txt sets rwxr-xr-x permissions.

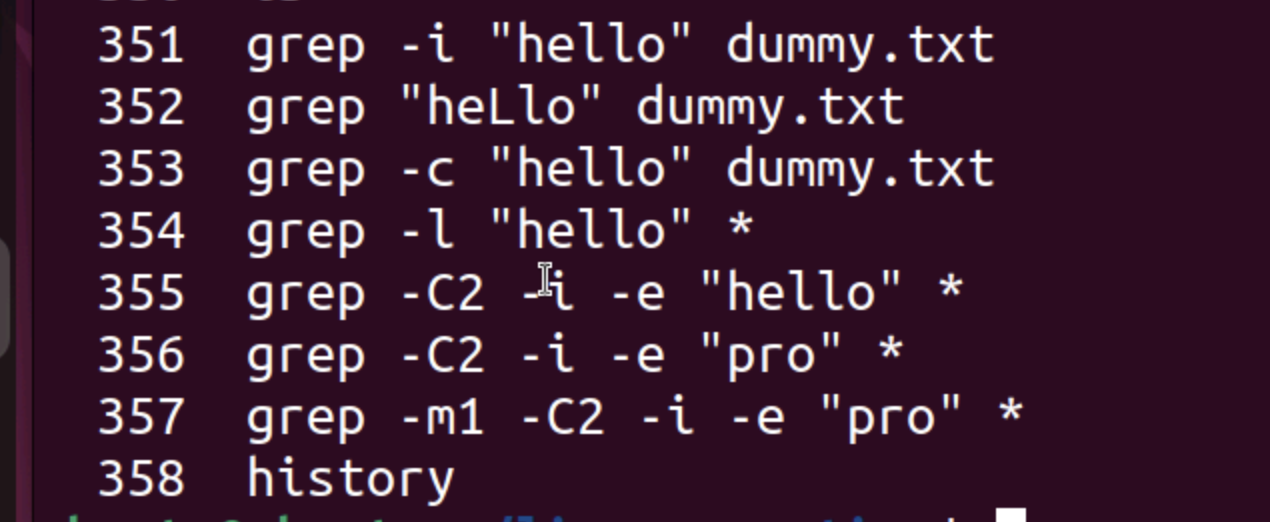
Task 13:  
 Yes, you can add users using useradd username, or adduser username (interactive). GUI tools can also be used.  
 (Screenshots can't be shared here; try the command in terminal to see output.)

Task 14:  
 Yes, use passwd username to change a user’s password, then enter the new password when prompted.  
 (Again, try in terminal to view prompts; screenshots not available in this format.)

Task 15:

| Feature | Process | Thread |
| --- | --- | --- |
| Definition | Program in execution | Lightweight process inside a process |
| Memory | Has separate memory space | Shares memory with other threads |
| Communication | Uses IPC (slow) | Faster via shared memory |
| Creation | Slow, resource-intensive | Faster, less overhead |
| Termination | Terminating kills all threads | One thread can end without killing others |
| Scheduling | Handled by OS | Managed within the process |
| Isolation | Strongly isolated | Weak isolation (shared data) |

Grep



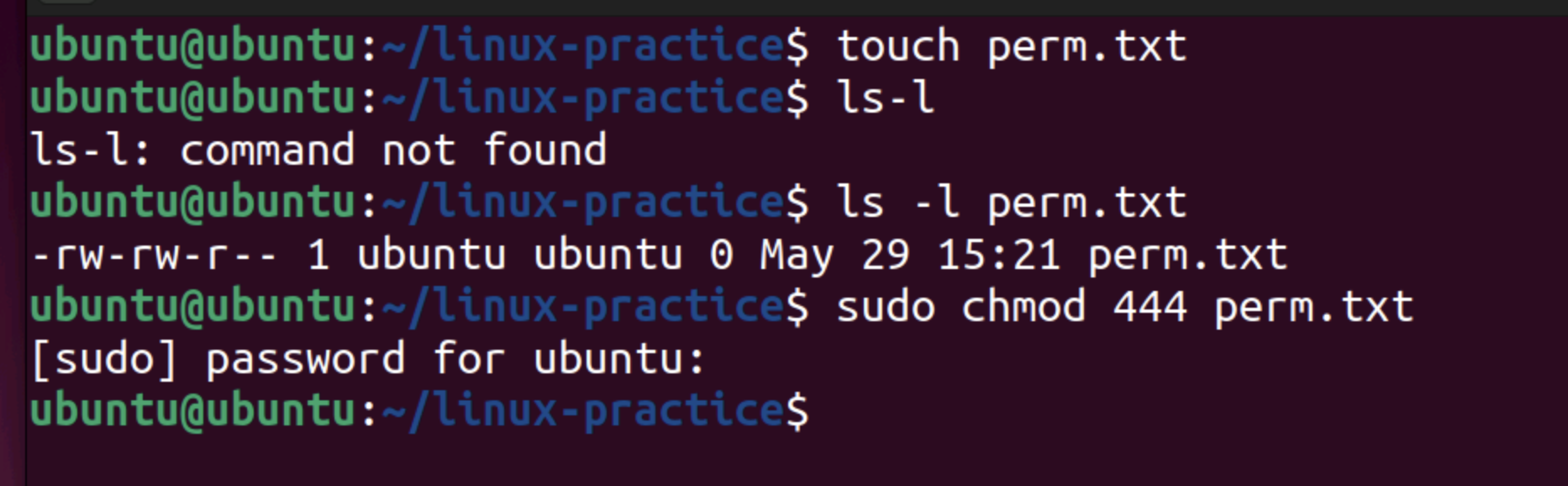
Task 19:

What are the default permissions for a new file ?

666

Task 20:

What is

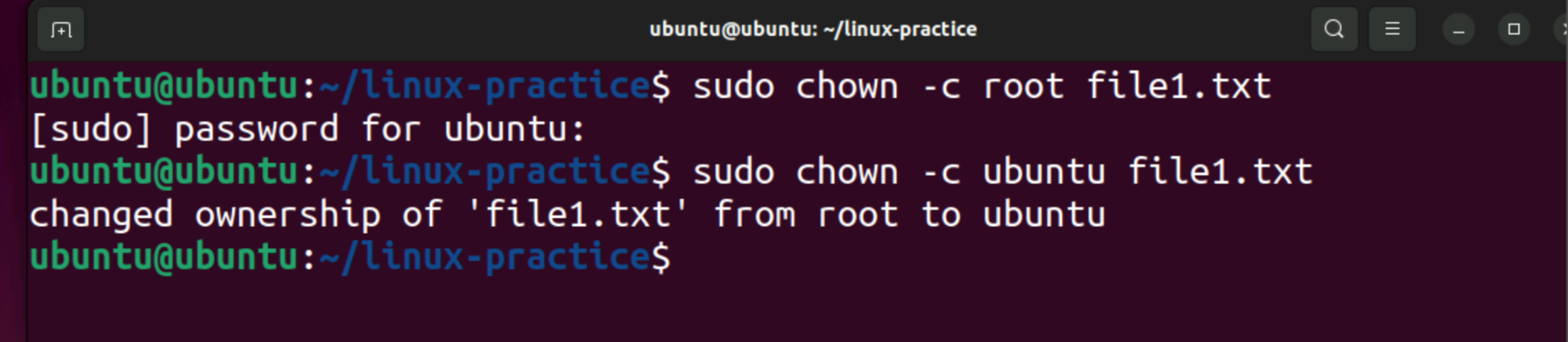


**Task 21:** chmod 640 filename

**Task 22:** chmod 640 filename

**Task 23:** chmod 751 chmod.exercises

**Task 24:** chmod 751 chmod.exercises

Task 25:  


Can you define what is a process

What is command to check foreground process and background process

Can you list all the running processes?

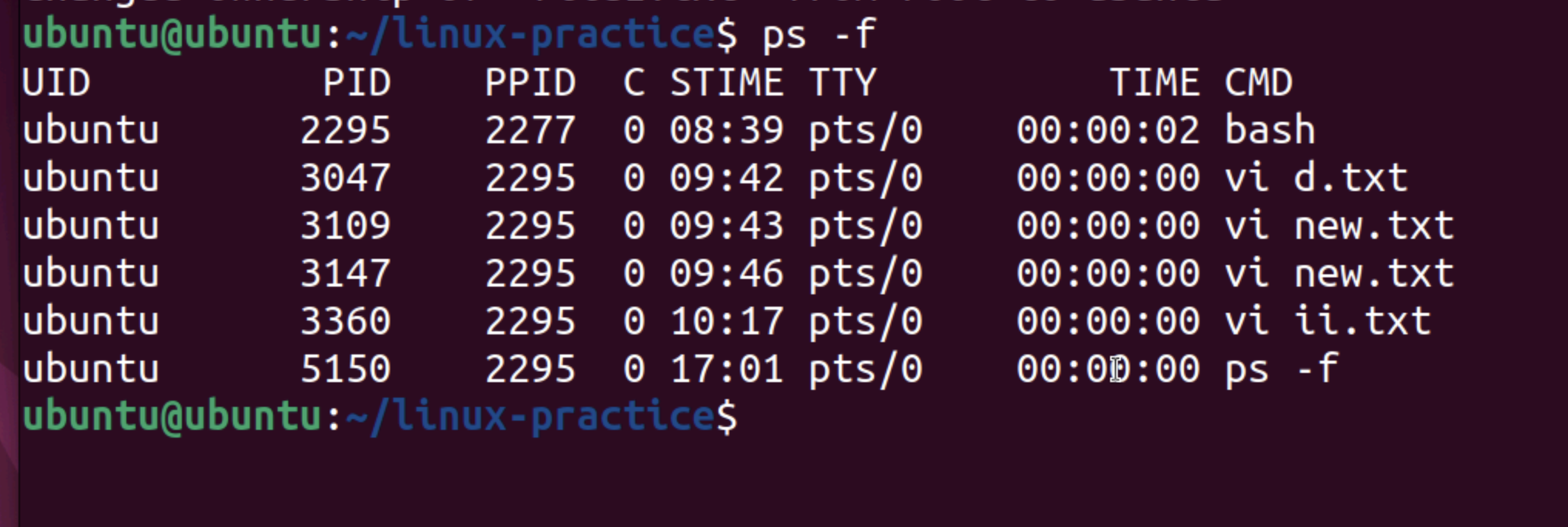
A process is an instance of a program in execution. It includes the program code, its current activity, and resources like open files and memory.

**Command to Check Foreground and Background Processes:**

* **Foreground process:** Automatically visible in the terminal. Use jobs to list jobs started in the current shell.
* **Background process:** Also listed with jobs or ps to see all processes:  
  + jobs — shows background and stopped jobs.
  + ps or ps aux — shows all running processes

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

ps -f displays a full-format list of running processes, showing details like UID, PID, PPID, start time, and the command that started each process.



# Task 30:

Name="prasunamba"

Id=10001

echo $Name

# Output: prasunamba

# Task 31:

Name="prasunamba"

readonly Name

Name="Meher"

# Output: bash: Name: readonly variable

# Explanation: Cannot change a readonly variable.

32::

unset Name

unset Id

echo $Name # No output, variable deleted

echo $Id # No output, variable deleted

# Trying to unset a readonly variable

Name="prasunamba"

readonly Name

unset Name

# Output: bash: unset: Name: cannot unset: readonly variable

# Task 33:

# Create an array of friends' names and print specific indexes

NAME[0]="Ram"

NAME[1]="Sita"

NAME[2]="Tina"

NAME[3]="Veena"

NAME[4]="Tim"

echo "First Index: ${NAME[0]}"

echo "Second Index: ${NAME[1]}"

# Output:

# First Index: Ram

# Second Index: Sita

# Task 34:

# Print all elements of the array at once

echo "${NAME[\*]}"

echo "${NAME[@]}"

# Output (both lines print all names separated by space):

# Ram Sita Tina Veena Tim

# Ram Sita Tina Veena Tim

