**Ramdeobaba University, Nagpur**

**School of Computer Science and Engineering**

**Session: 2024-2025**

**Fundamentals of Linux OS I Semester**

**PRACTICAL NO. 4**

**Aim: Viewing and managing processes in Linux.**

**Tasks:**

* 1. View running processes using ps, top, and htop.
  2. Manage processes using commands like kill, pkill, killall, nice, and renice.

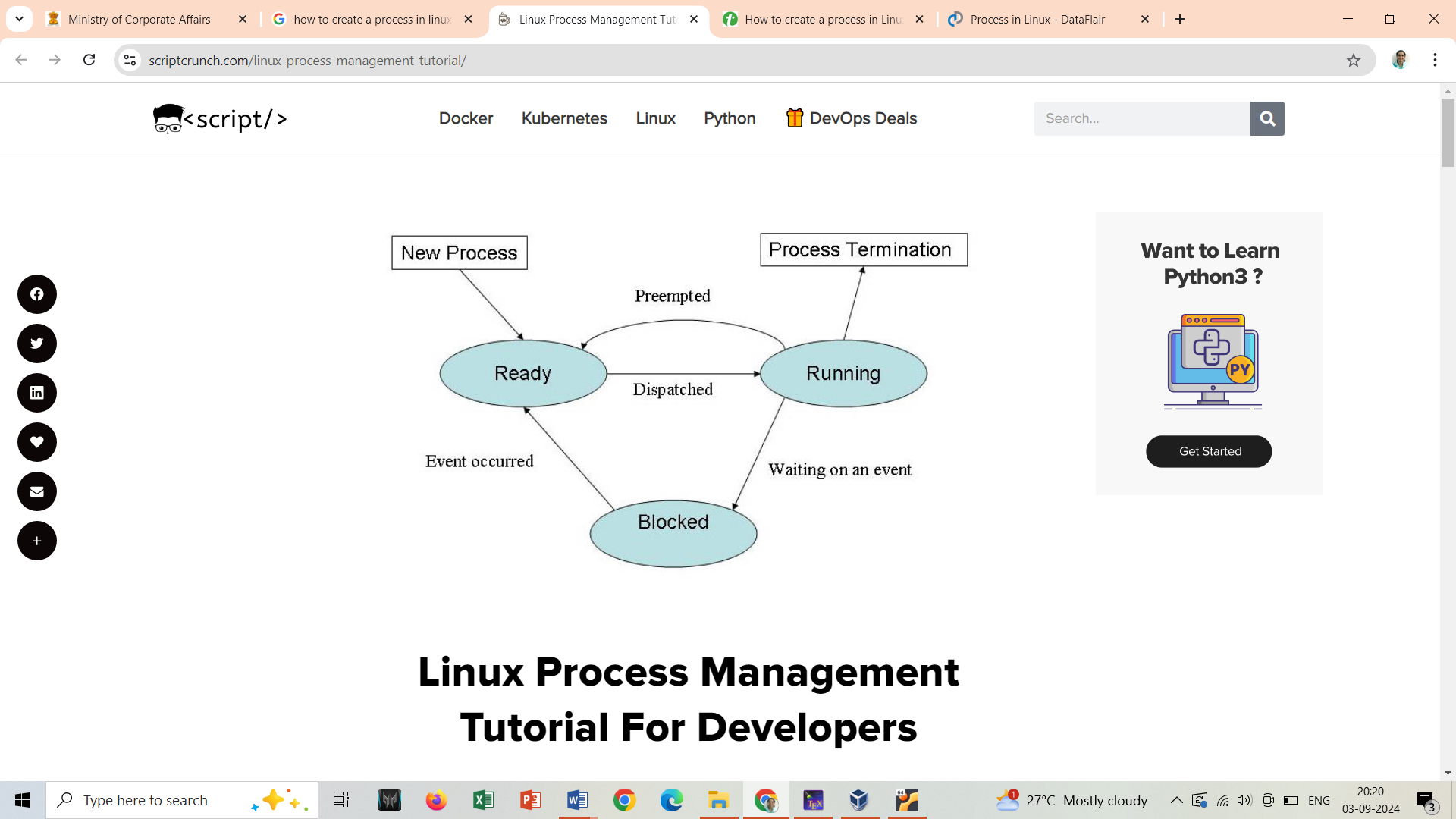
**Theory:**

### Introduction to Process Management in Ubuntu

In any operating system, including Ubuntu, process management is crucial for ensuring that programs and services run smoothly and efficiently. A process is essentially an instance of a running program, and managing these processes involves viewing, controlling, and optimizing them.

A process in Linux starts every time you start an application or run a program or command. A program or command creates only one process, but an application on the other hand has the capability to start multiple processes to accomplish various tasks.

**Process States:**



### Why Process Management is Important:

* **Resource Utilization:** Efficiently managing processes helps in optimal utilization of system resources like CPU and memory.
* **System Stability:** Properly handling processes prevents system crashes and ensures that critical services are running smoothly.
* **Performance Monitoring:** Monitoring processes helps in identifying performance bottlenecks and improving system responsiveness.

### Key Concepts:

1. **Viewing Processes:**
   * **ps Command:** Displays a snapshot of current processes.
   * **top Command:** Provides a real-time, interactive view of process activities.
   * **htop**
2. **Managing Processes:**
   * **kill Command:** Terminates a process using its PID (Process ID).
   * **pkill Command:** Sends a signal to processes based on their name.
   * **killall Command:** Similar to pkill, but can target all processes with a specific name.
   * **nice and renice Commands:** Adjust the priority of processes to manage their CPU usage.

**Different commands**

#### 1. Viewing Running Processes

**Commands:**

1. **List all running processes:**

ps aux - Displays detailed information about all running processes.

1. **Real-time process monitoring:**

top - Provides a dynamic, real-time view of system processes and their resource usage.

1. **Interactive process viewer:**

htop - An enhanced version of top with a user-friendly interface (requires installation via sudo apt install htop).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Experiment (PART-A):**

1. Run ps aux and note down the process IDs (PIDs) and commands of a few running processes.
2. Open top and observe the real-time updates of CPU and memory usage. Press q to exit.
3. Install htop if not already installed and run it. Explore its features, including process sorting and searching.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### 2. Managing Processes

**Commands:**

1. **Terminate a process by PID:**

kill PID - Sends the SIGTERM signal to terminate the process with the specified PID.

1. **Forcefully terminate a process:**

kill -9 PID - Sends the SIGKILL signal to forcefully terminate the process.

1. **Terminate processes by name:**

pkill process\_name - Sends the SIGTERM signal to all processes with the given name.

1. **Terminate all processes by name:**

killall process\_name - Sends the SIGTERM signal to all processes matching the given name.

1. **Start a process with a specific priority:**

nice -n 10 command - Starts the command with a niceness of 10 (lower priority).

[The nice command is used to start a process with a defined priority (from -20 to 19, where -20 is the highest priority and 19 is the lowest).]

1. **Change the priority of a running process:**

renice 10 -p PID - Changes the niceness of the process with the given PID to 10.

**Experiment (PART-B):**

1. Open a new terminal [say Terminal-2] and create a file (fileA.txt) using cat command with redirection (>) allowing text entry. Let the data entry in file be kept open (that is do not press ctrl+ D to exit from cat)
2. Find the PID of this cat process using ps aux or pgrep on the previous terminal [say Terminal-1].
3. Use kill PID to terminate the cat process gracefully. What happens in Terminal-2? Are the fileA.txt contents updated?
4. Again start cat command to append data in the fileA.txt [on Terminal-2]
5. Find the PID for cat command and use kill -9 PID to forcefully terminate the same process [from Terminal-1]. What happens in Terminal-2? Are the fileA.txt contents updated?
6. On Terminal-2 start entry of text in fileA.txt again
7. On Terminal-1 test pkill by terminating processes based on their name.
8. Run a CPU-intensive command with nice to observe its lower priority and change the priority of a running process using renice.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_