**Ramdeobaba University, Nagpur**

**School of Computer Science and Engineering**

**Session: 2024-2025**

**Fundamentals of Linux OS I Semester**

**PRACTICAL NO. 5**

**Aim: To understand the management of background and foreground processes in Linux, and to manage system services**

**Tasks:**

1. Experiment with background and foreground processes (&, fg, bg).
2. Explore the /proc directory to understand process information.
3. Managing System Services

**Theory:**

**Types of Processes in Linux**

In Linux, processes are of 2 types, let us go through each of them:

**1. Foreground processes**

Foreground processes are also called as interactive processes. These processes are launched and controlled through the command line in a terminal session. There has to be a user in the system to start such foreground processes as they cannot start automatically by themselves.

2.**Background processes**

Background processes are also called as non-interactive processes. These processes are the complete opposite of the Foreground process. They don’t need any user input and are not connected to the terminal. One such example of background processes is daemons.

**What are Daemons?**

Since we touched on the topic of Daemons, might as well learn what they are. Daemons are a special type of background process that starts when the system boots and keeps running forever, till eternity — they never die. Daemons can be controlled by a user through the init process.

## Why manage jobs?

By default, all jobs in Linux execute in the foreground. Foreground jobs run directly from the shell. When you run one foreground job, you will have to wait for shell access until the job completes before you can run other jobs.

Waiting for jobs to complete is fine when the jobs complete quickly. But in cases where the current job is going to take a long time (even several hours) to complete, it becomes a challenge, especially when you have a single terminal window (SSH session, web console) to work with.

In such cases, you can run the job in the background and free the shell for other jobs and tasks. When you run a Job in the background, it will execute at low priority, which will, in turn, allow typing of other commands in the terminal window while the job runs.

**Key Concepts**

1. **Background and Foreground Processes:**
   * **Running in Background:** Allows you to continue using the terminal while a process runs in the background.
   * **Foreground/Background Management:** Commands like fg, bg, and Ctrl+Z help manage processes between the background and foreground.
2. **Exploring Process Information:**
   * **/proc Directory:** Contains detailed information about system processes and their resource usage.
3. **Managing System Services:**
   * **systemctl Command:** Used to manage system services, including starting, stopping, and configuring services to start on boot.

#### 1. Background and Foreground Processes

**Commands:**

1. **Run a process in the background:**

& - Runs the specified command in the background.

1. **Bring a background process to the foreground:**

fg %job\_number - Brings the specified job from the background to the foreground.

1. **Send a foreground process to the background:**
2. **Suspending a foreground job with CTRL + Z**
   * Let’s say you want to suspend the job for a while to run other commands; you can do that with CTRL + Z. CTRL + Z puts the job in a “stopped” mode and doesn’t terminate it.
   * **Resume in the background:** bg %job\_number - Suspends a foreground process and resumes it in the background.

## Running jobs even after closing the terminal window

When you have jobs running in the background or foreground, all the jobs will be terminated if you close the terminal window. The job termination is because they are associated with that particular terminal session.

To make jobs continue running even after closing the terminal window, you can use **nohup** command.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Experiment (PART-A):**

1. Run sleep command for 1000 seconds. What do you observe? Terminate it using ctrl+C.

sleep 1000

1. Start a long-running process [say Sleep] in the background using command &

sleep 1001 &

1. List the running background jobs. Do you see the job you started running in

background?

jobs

1. Bring the process to the foreground.

fg %1

1. Suspend the process and then resume it in the background.

bg %1

1. Create more two jobs with sleep and view those using jobs command. Conclude about + and – sign you see after the job number.

sleep 1002 &

sleep 1003 &

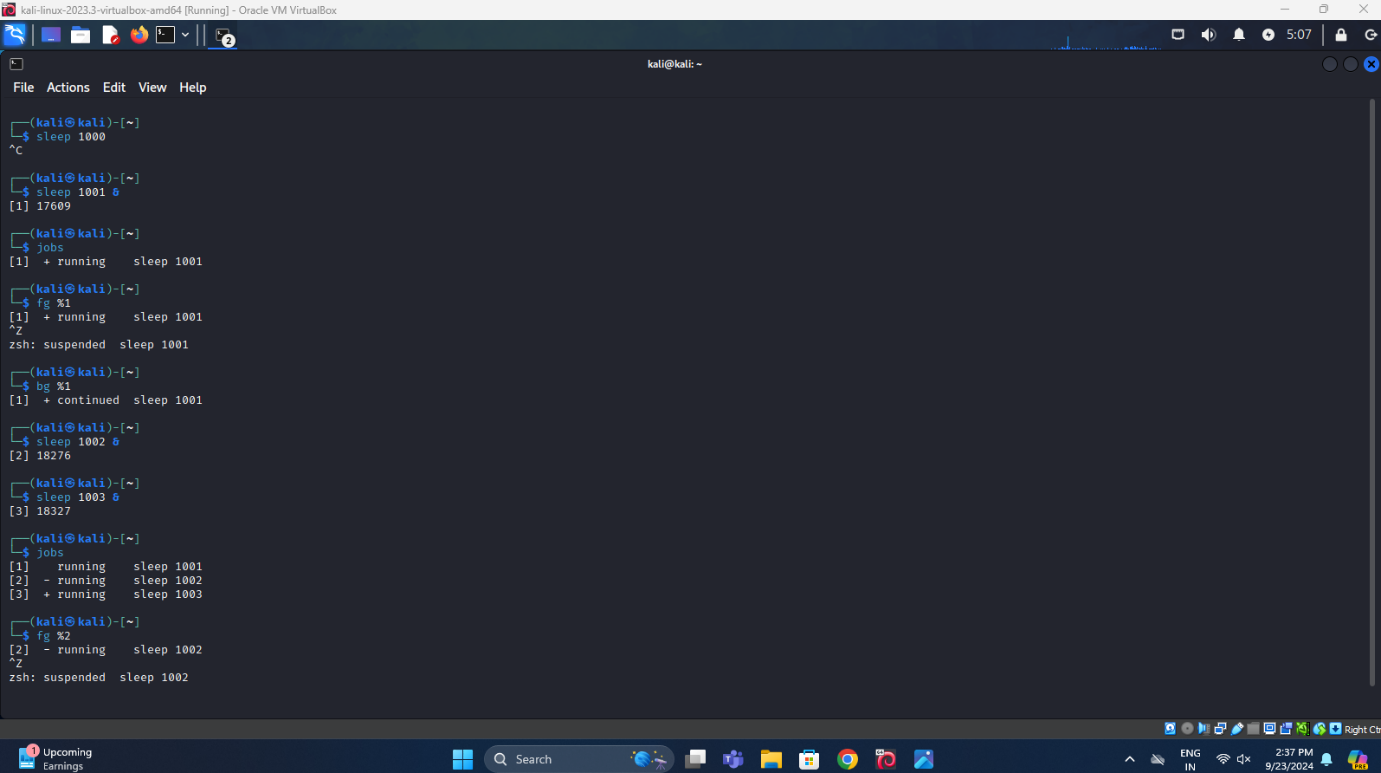
jobs

1. Bring the 2nd job to the foreground and suspend it. Conclude about what you observe about the running processes.

fg %2

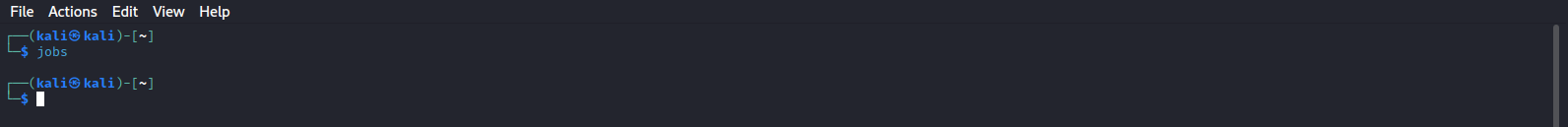
ctrl+Z

The running process 2 is suspended.



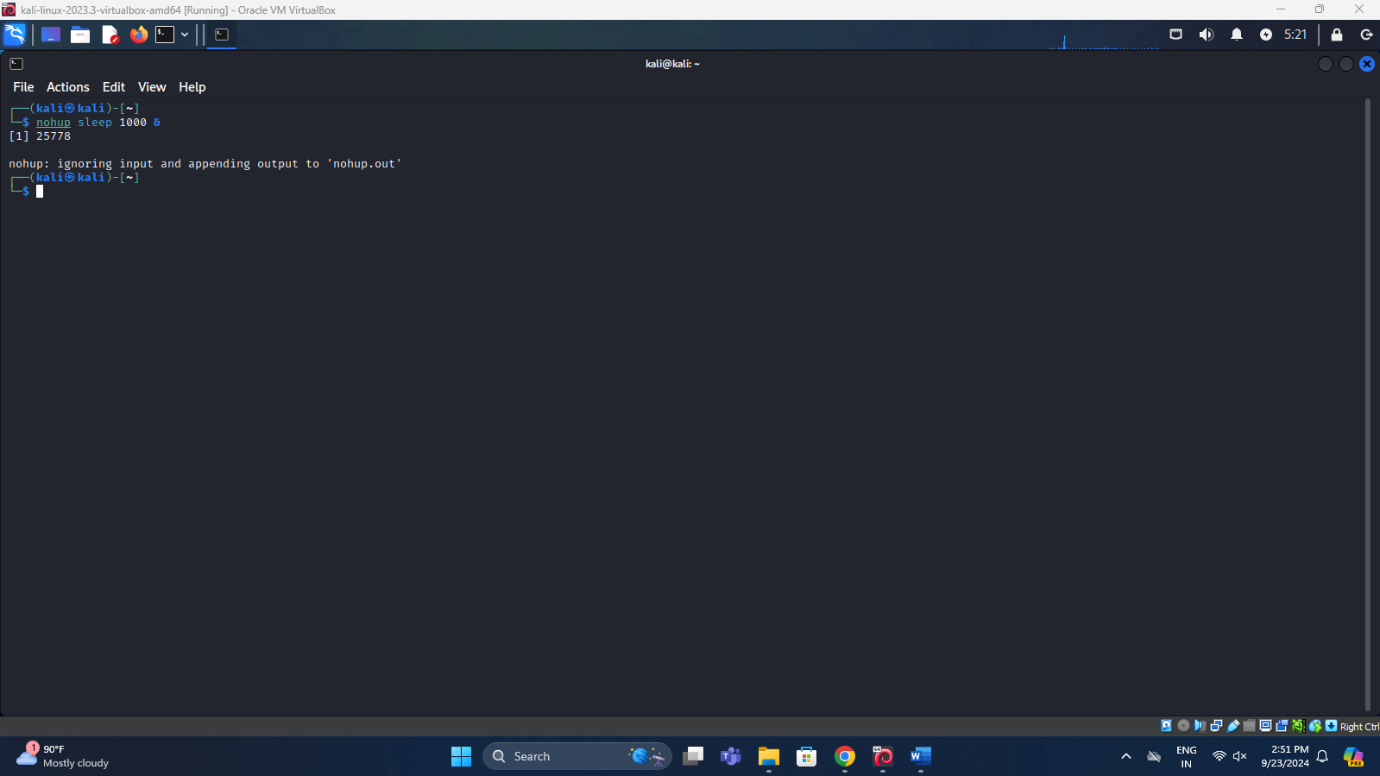
1. Open a new terminal and run jobs. What do you observe and why?

No job is created in the new terminal.\



1. Create a new job [sleep] that continue running even after closing the terminal window and close the terminal.

nohup sleep 1000 &



1. Open a new terminal and check jobs. What do you observe? Use ps aux to find the running process and kill it.

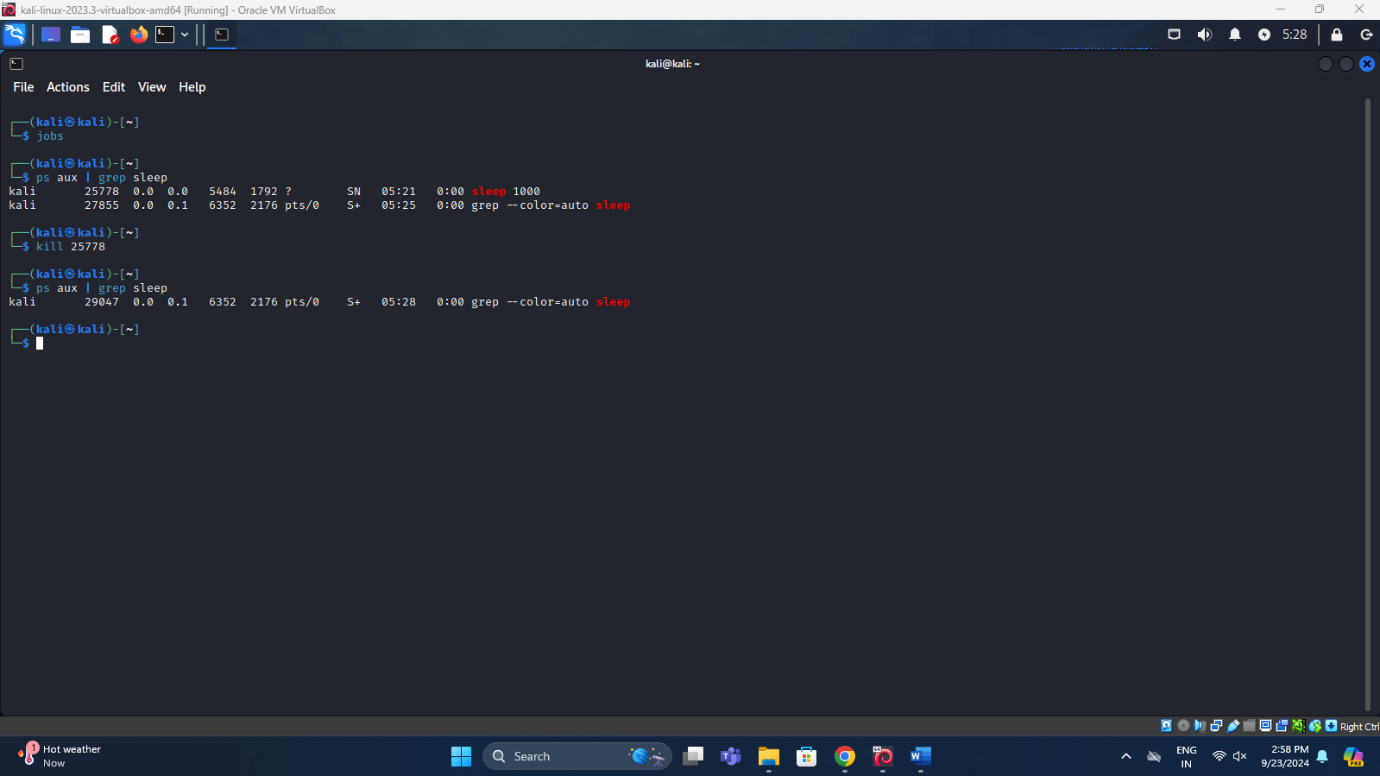
jobs

ps aux | grep sleep

pkill sleep

jobs

ps aux | grep sleep

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

#### 2. Exploring the /proc Directory

**Commands:**

1. **List process directories:**

ls /proc **-** Lists the contents of the /proc directory, where each subdirectory is named after a PID.

1. **View process information:**

cat /proc/PID/status - Displays detailed status information about the process with the specified PID.

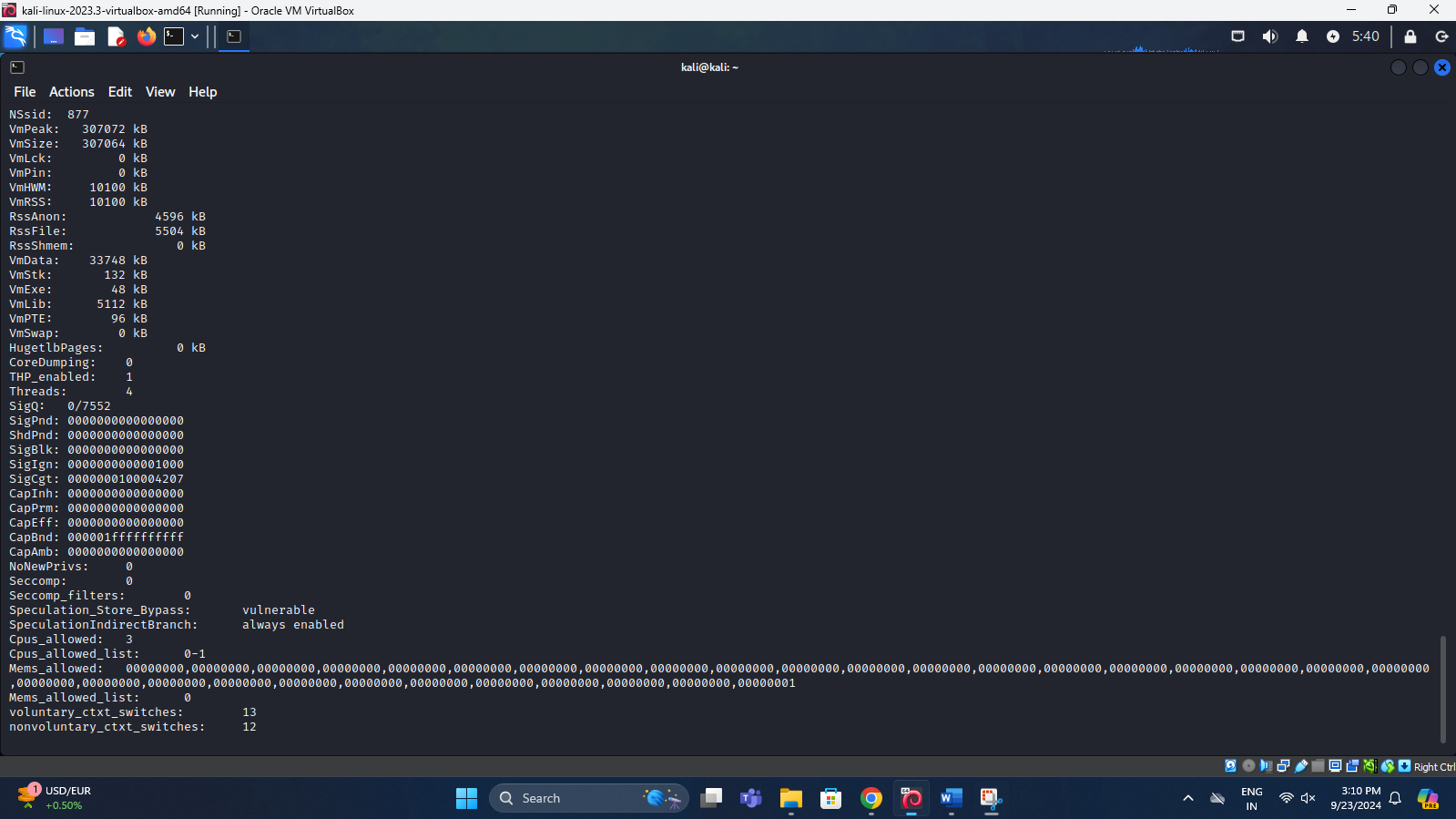
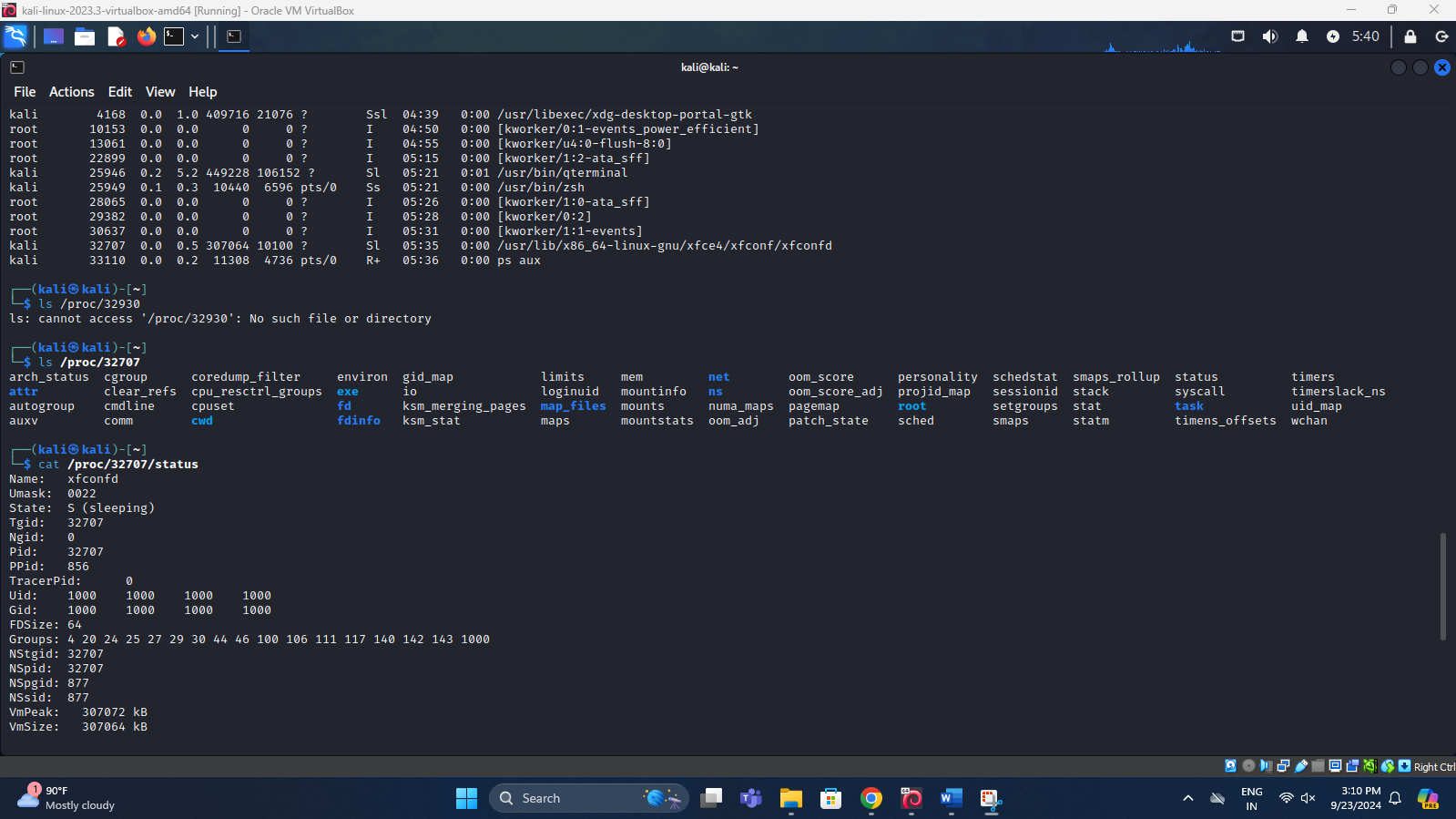
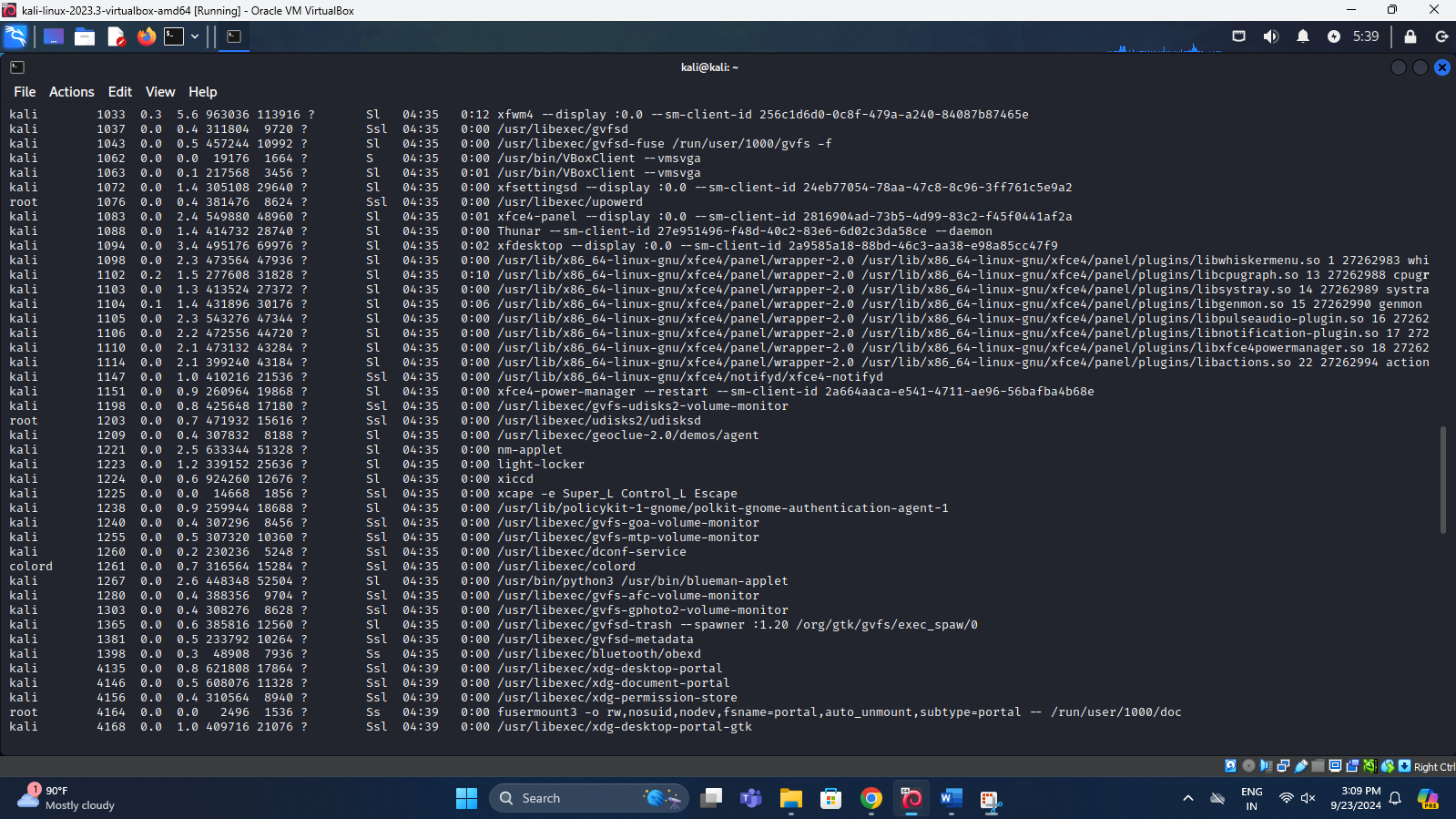
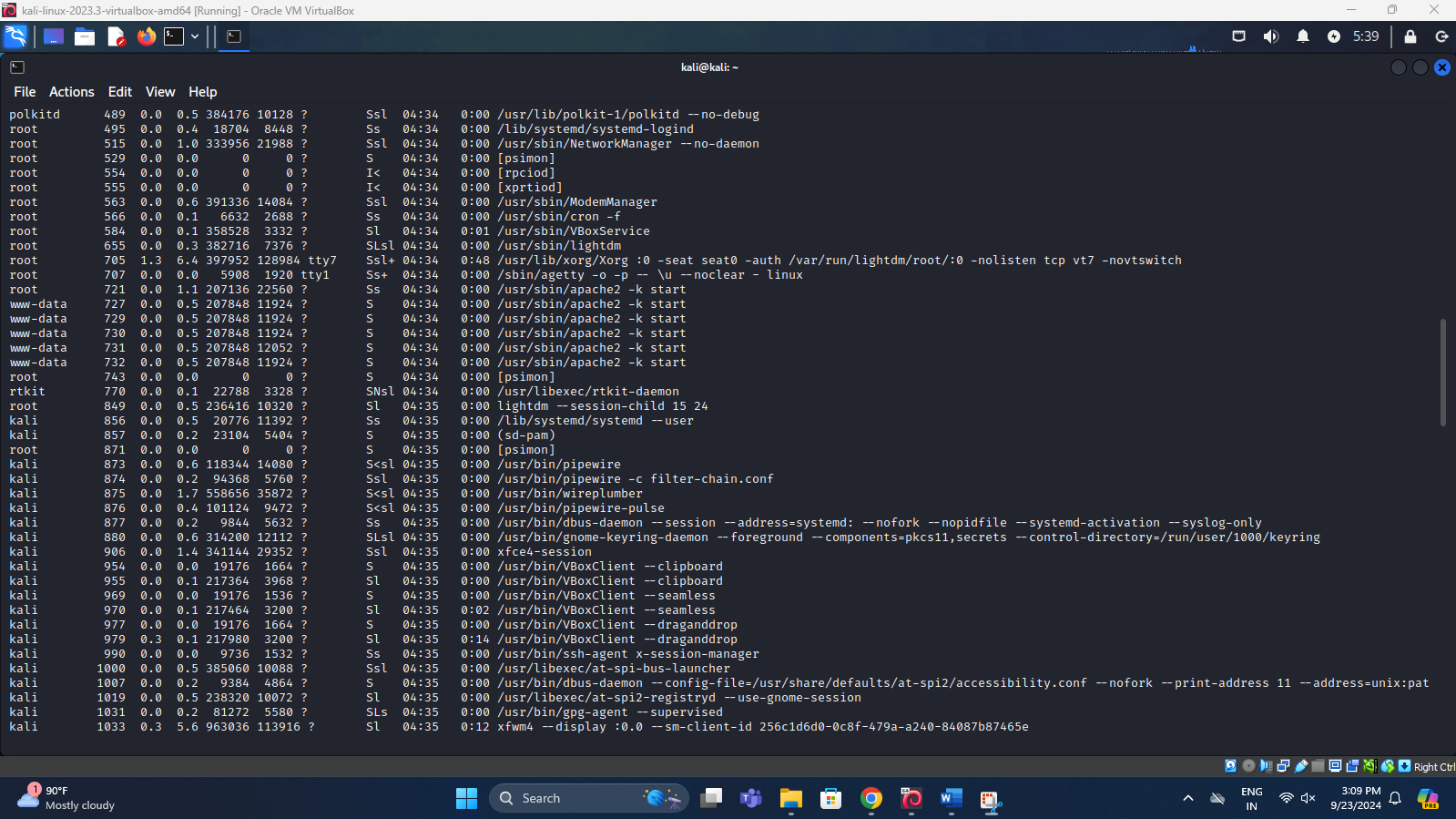
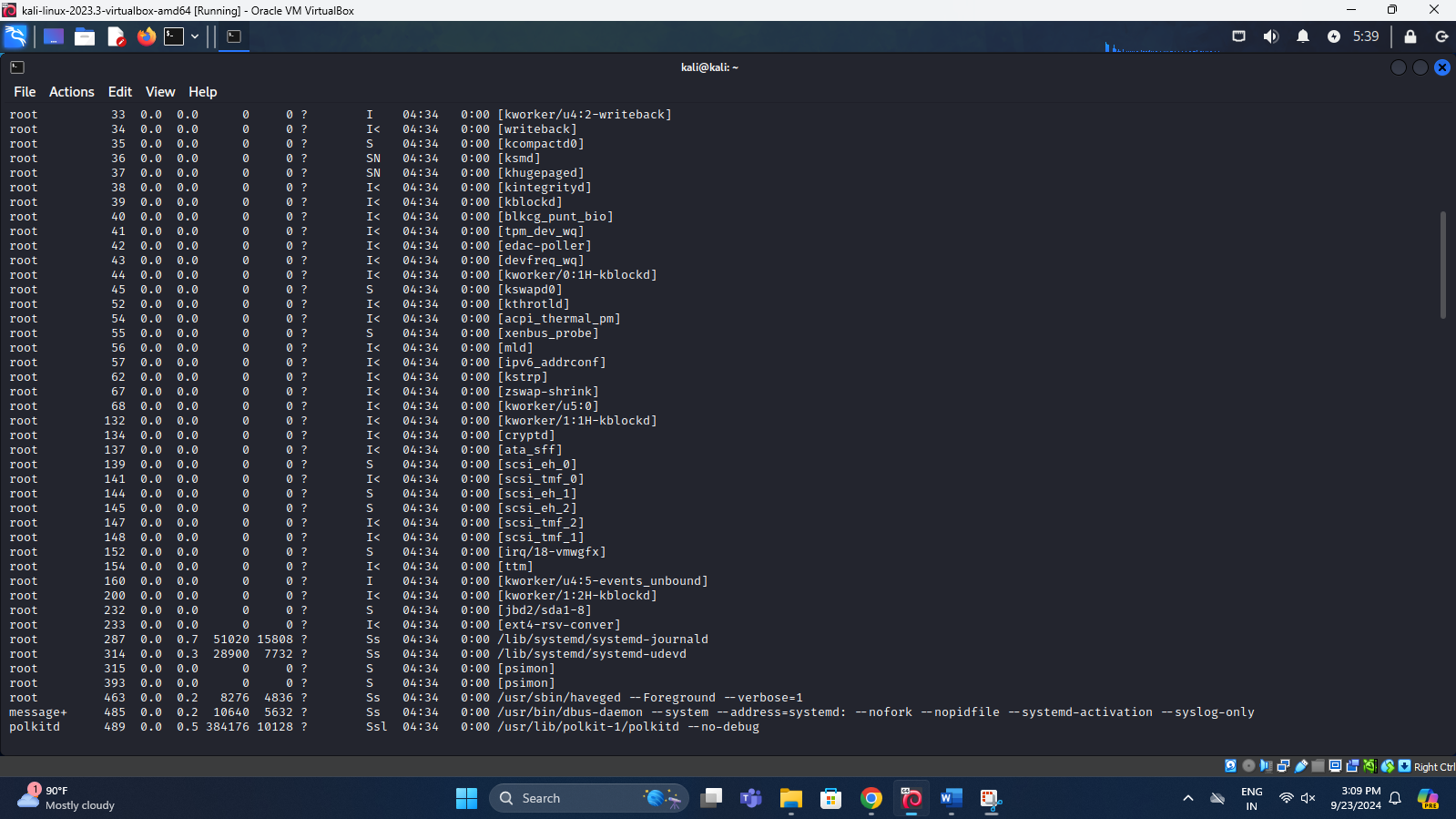
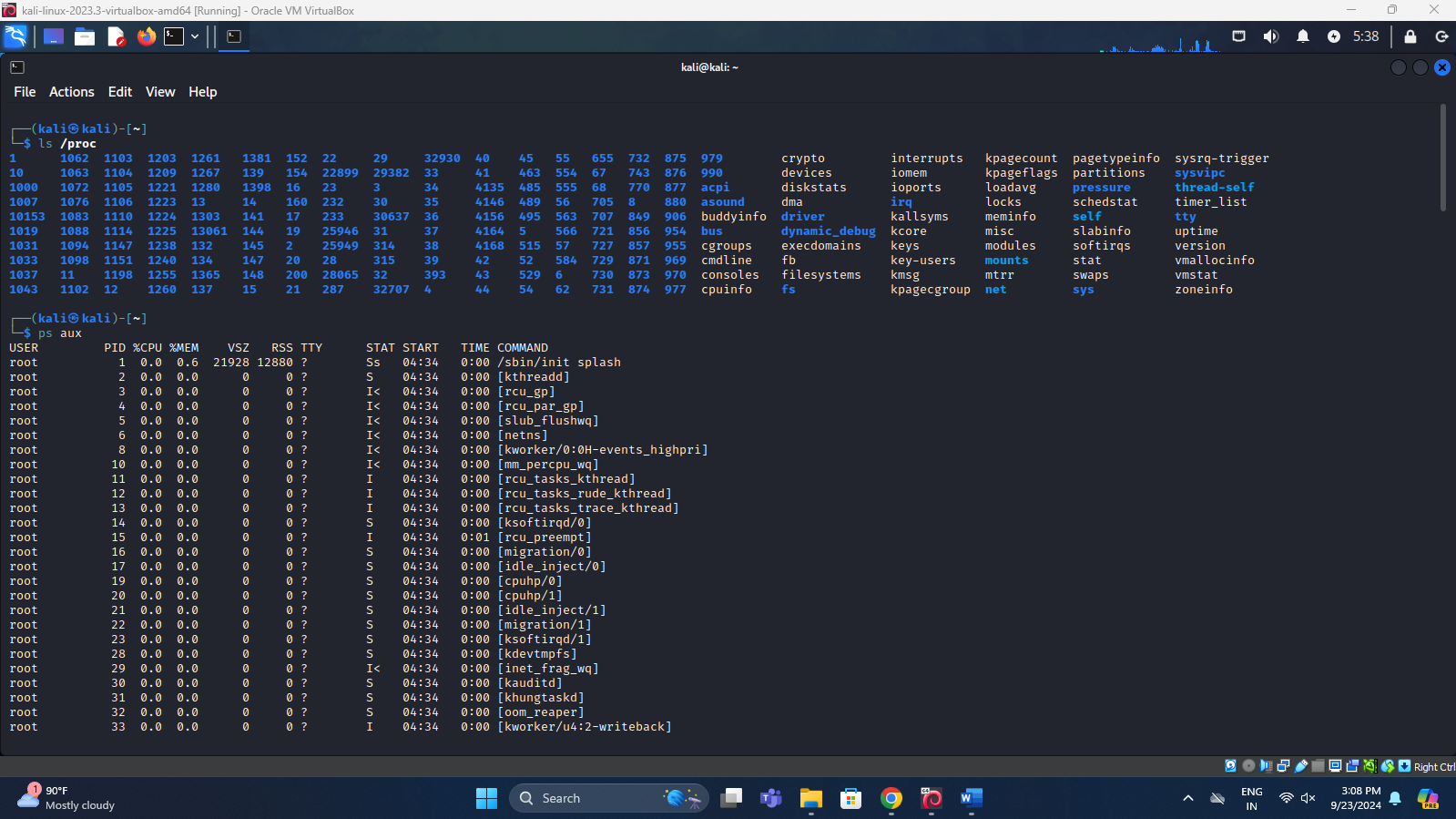
**Experiment (PART-B):**

1. Find the PID of a process using ps aux or top.

ps aux

1. Navigate to /proc/PID and explore files such as cmdline, status, and fd.

Ls /proc/32707

1. Use cat /proc/PID/status to examine the process’s status information.
2. Cat /proc/32707/stat

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

#### 3. Managing System Services with systemctl

**System Services:** **System services** in Linux (and other operating systems) refer to background processes that provide essential functionalities to the system or its users. These services run continuously, typically from the time the system starts until it is shut down, without direct interaction from users. They handle key tasks like network connectivity, device management, file sharing, and security.

**Characteristics of System Services:**

* **Daemon Processes**: Most system services are daemons, which means they run in the background and are usually started at boot time. These daemons typically have names ending with a "d", such as sshd (Secure Shell Daemon), or httpd (HTTP Daemon).
* **Managed by init systems**: Services are started, stopped, and managed by **init systems** like **systemd**, or **Upstart**.
* **Run at Specific Run Levels**: Services run at different run levels (or targets in systemd). For example, some services are only enabled when the system is in multi-user mode or graphical mode.

**Common Types of System Services:**

* **Network Services**:
  + sshd: Manages SSH connections, allowing remote login and command execution.
  + network-manager: Manages network connections like Wi-Fi, Ethernet, etc.
  + firewalld: Manages firewall rules and network security.
* **Web and Database Services**:
  + **nginx** / **apache2**: Web servers that serve web pages to clients.
* **System Monitoring and Scheduling**:
  + **crond**: Manages scheduled tasks, like running backups or system maintenance at specified times.
  + **rsyslogd**: Handles system log management.

**Service States:**

* **Active (Running)**: The service is up and running.
* **Inactive (Stopped)**: The service is not running but is not in an error state.
* **Failed**: The service encountered an issue and could not start or is in a failed state.

**Commands:**

1. **Check the status of a service:**

systemctl status service\_name - Displays the current status of the specified service.

1. **Start a service:**

sudo systemctl start service\_name - Starts the specified service.

1. **Stop a service:**

sudo systemctl stop service\_name - Stops the specified service.

1. **Restart a service:**

sudo systemctl restart service\_name - Restarts the specified service.

1. **Enable a service at boot:**

sudo systemctl enable service\_name - Configures the service to start automatically at boot.

1. **Disable a service:**

sudo systemctl disable service\_name - Prevents the service from starting automatically at boot.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Experiment (PART-C):**

The SSH service refers to the Secure Shell (SSH) protocol and its associated server process, which allows secure, encrypted communication between a client and a server over an unsecured network. SSH is primarily used to remotely log into another computer (often a server) and execute commands as if you were physically present on that machine.

SSH server is a service (daemon) that runs on the remote machine (server). It listens for incoming SSH connection requests on a designated port (default: **22**) and authenticates users trying to connect.

In Linux, the SSH server is typically managed by a process called sshd (Secure Shell Daemon).

1. Install ssh [use sudo apt install openssh-server]

sudo apt install openssh-server

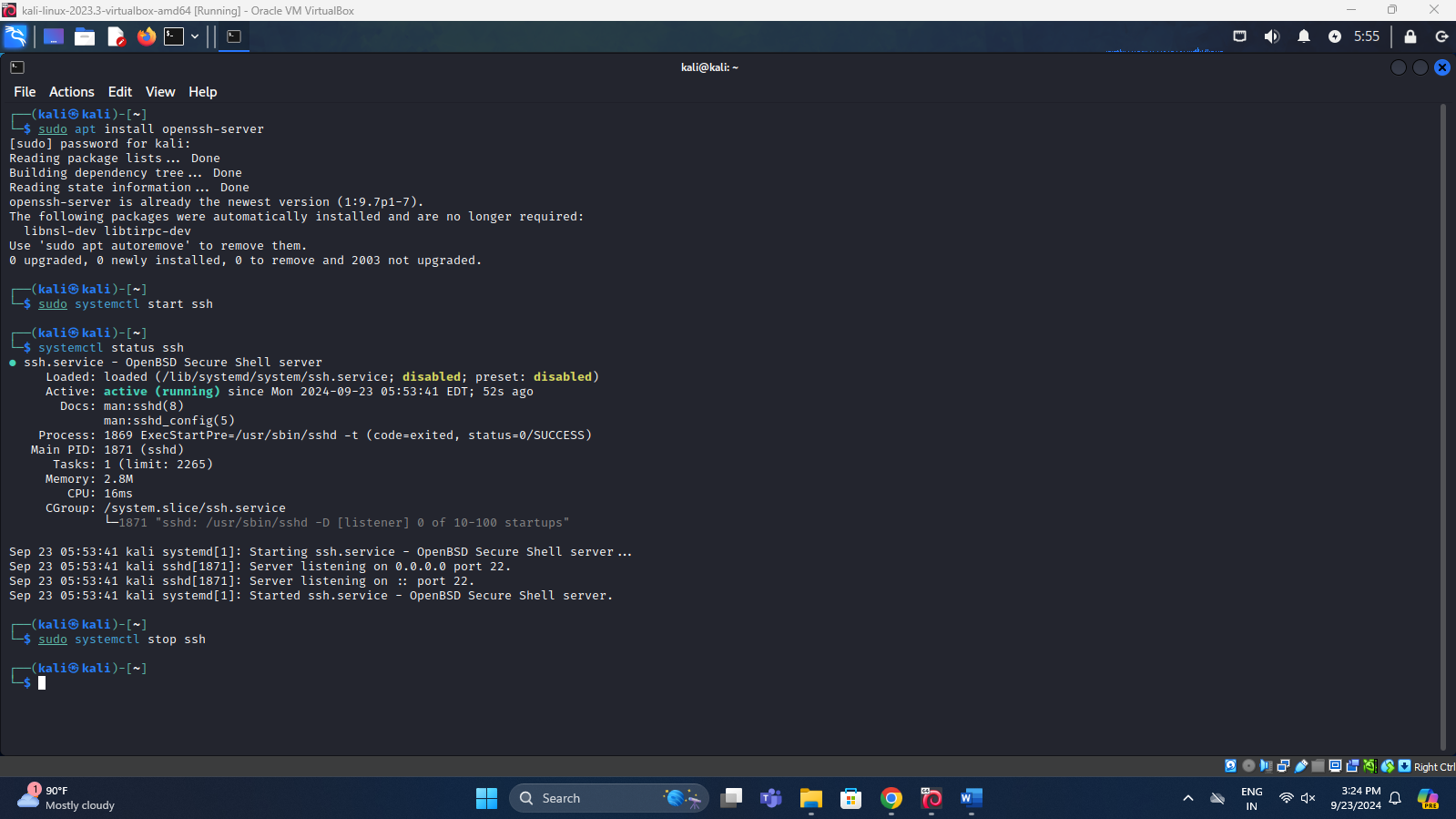
1. Start the sshd service

sudo systemctl start ssh

1. Check the status of sshd service

sudo systemctl status ssh

1. Stop it

sudo systemctl stop ssh

1. Restart it

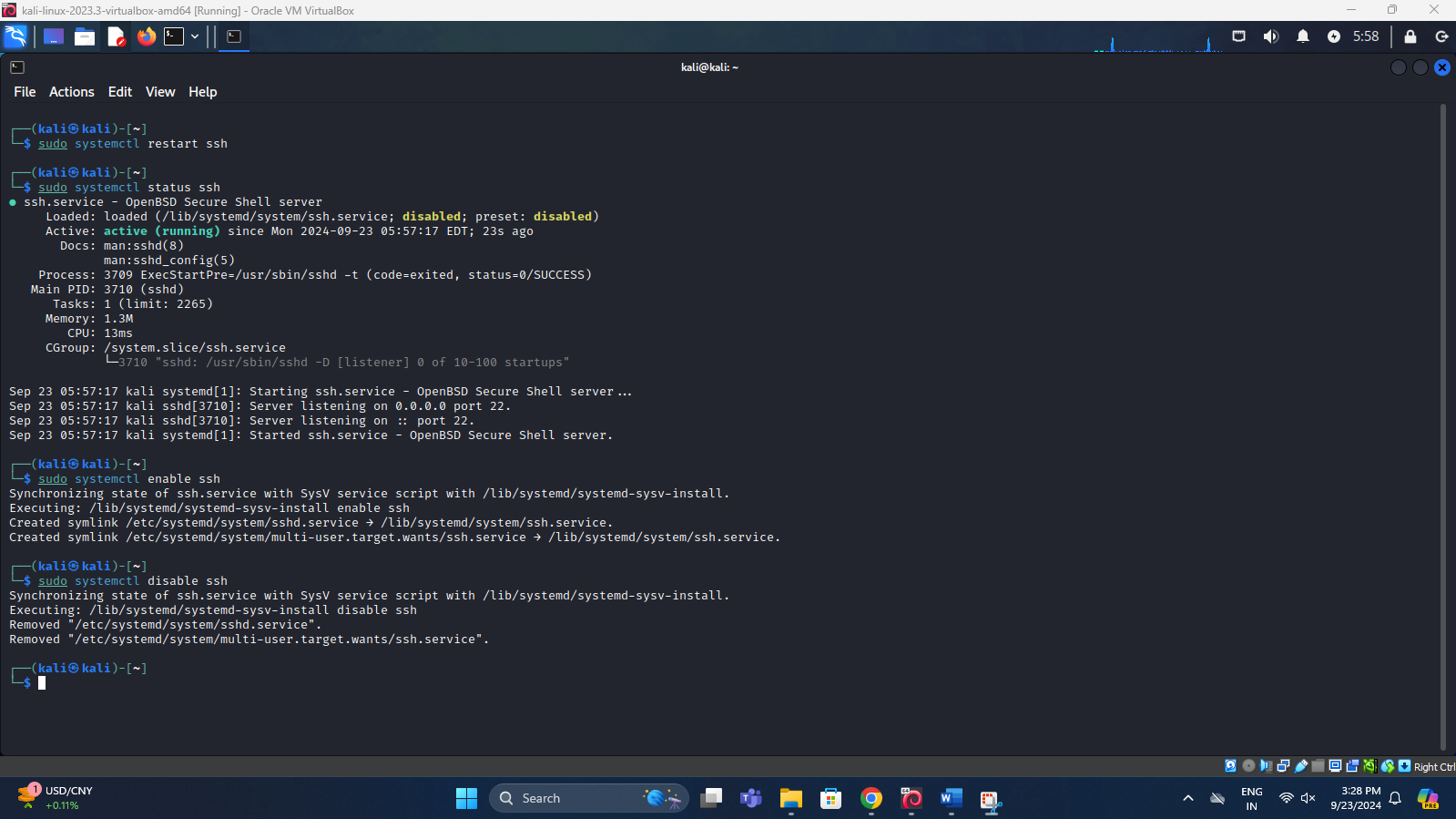
sudo systemctl restart ssh

1. Enable the service ssh at boot time.

sudo systemctl enable ssh

1. Disable the service.

sudo systemctl disable ssh



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