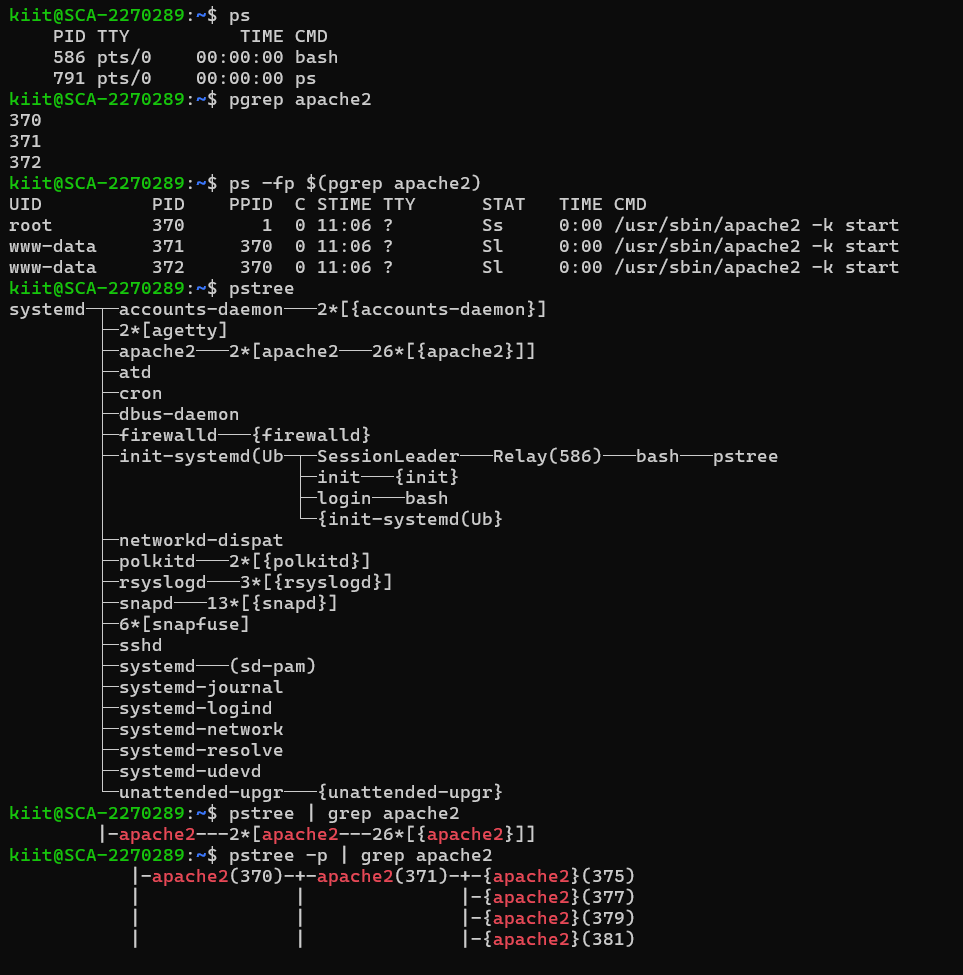
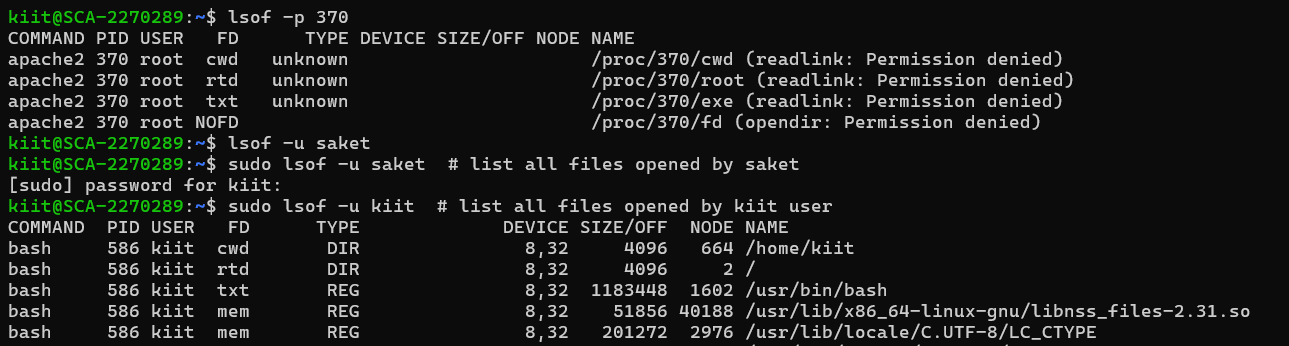
# Process Exploration and Identification

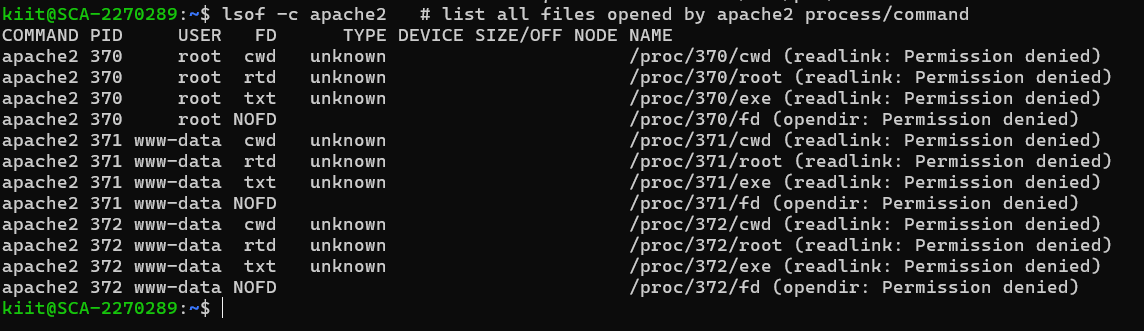
**Objective:** Understand how processes work in Linux, and how to identify and explore running processes.

#### Outcome:

This lab will help you become familiar with the tools and techniques used to explore and gather information about running processes in Linux

****

****

****

# Lab 2 : Process Control and Termination

#### Objective: Learn how to control, pause, resume, and terminate processes in Linux.

1. Send Signals to Processes:

# Use kill to send signals to processes. Try sending a SIGTERM and SIGKILL to terminate a process by PID.

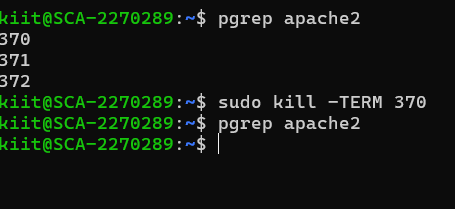
# Use kill -s STOP <PID> and kill -s CONT <PID> to stop and resume a process.

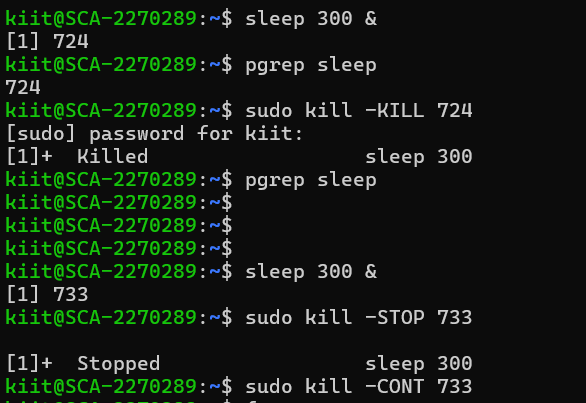
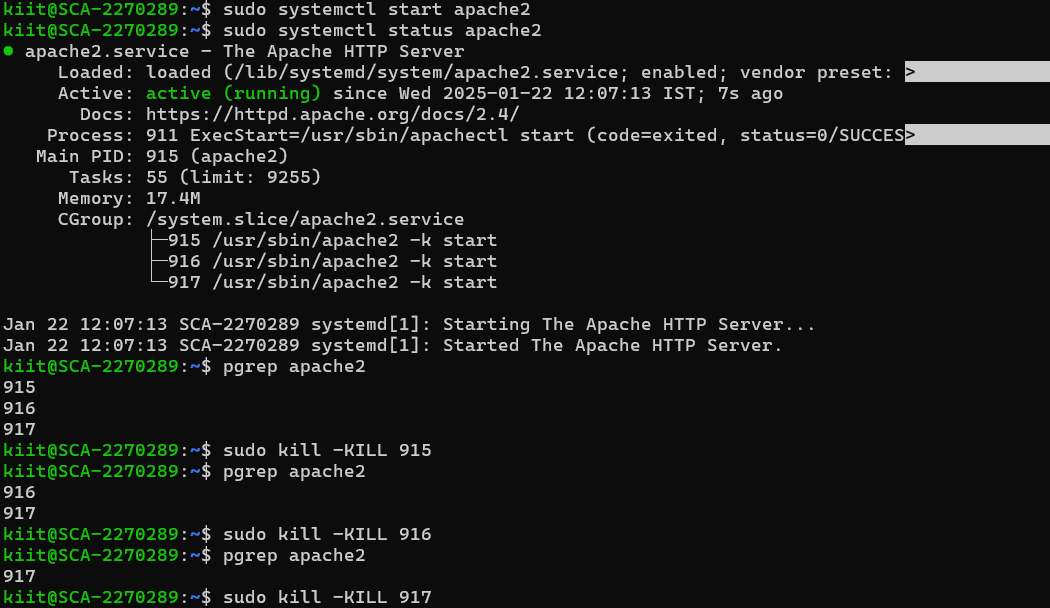
2. Send Custom Signals:

# Send a SIGINT signal to a running process (e.g., when running a program in the terminal, use Ctrl+C or kill -s SIGINT <PID>).

3. Test Process Termination:

# Start a process, for example, sleep 300, then find its PID and try to terminate it using kill or kill -9.





# Lab 3 : Managing Background and Foreground Processes

#### Objective:Learn how to run processes in the background and manage jobs effectively.

1. Run a Process in the Background:

# Start a process in the background using &, e.g., sleep 100 &.

# Use jobs to see a list of background jobs.

2. Bring a Process to the Foreground:

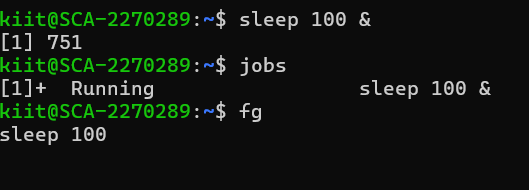
# Use the fg command to bring a background process to the foreground.

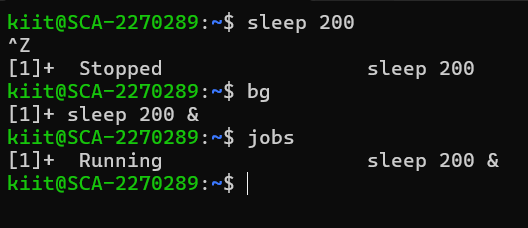
3. Pause and Resume a Process:

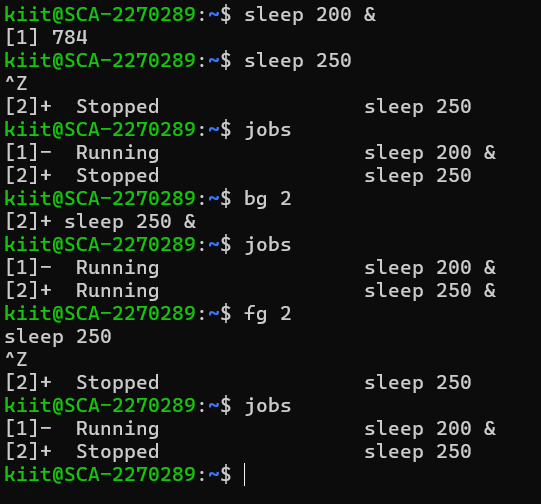
# Pause a background process using Ctrl+Z and resume it in the background with the bg command.

4. Control Multiple Jobs:

# Start multiple jobs in the background and manage them with jobs, fg, and bg.







# Lab 4 : Monitoring System Performance and Resource Usage

#### Objective: Learn how to monitor system resources and analyze processes consuming system resources.

1. Monitor CPU Usage:

# Use top or htop to monitor CPU usage in real-time.

# Look for processes consuming high CPU and analyze them.

2. Monitor Memory Usage:

# Use free or vmstat to check system memory usage.

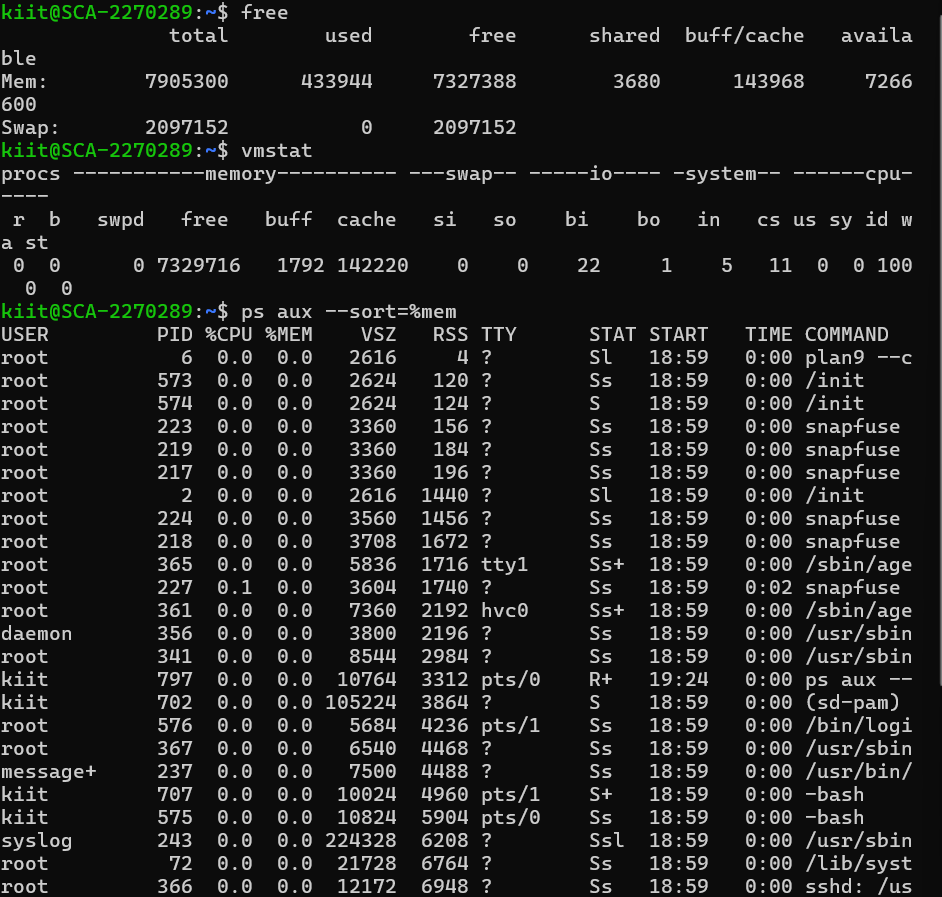
# Use ps aux --sort=-%mem to find processes using the most memory.

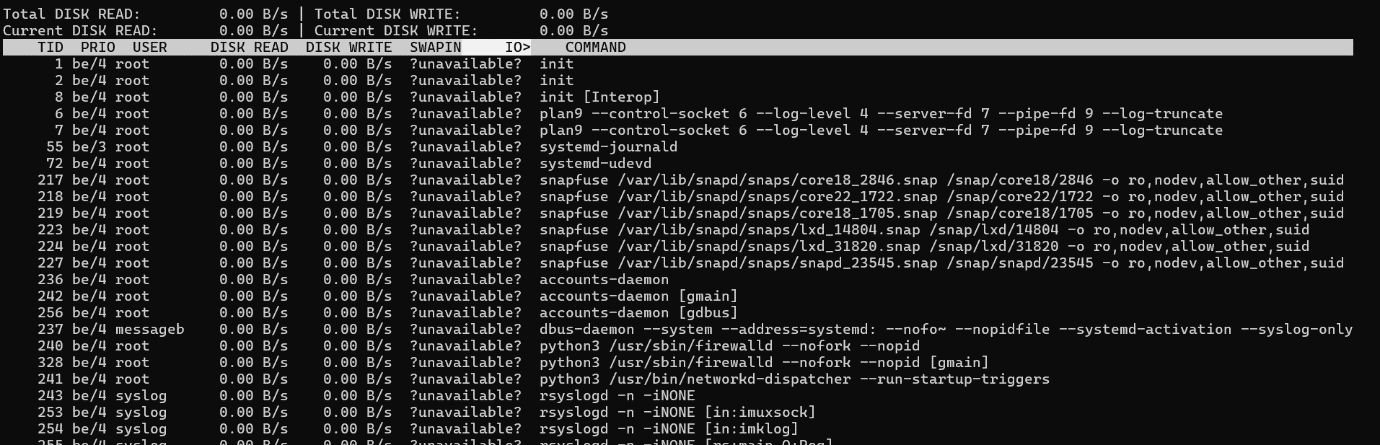
3. Disk Usage and I/O Monitoring:

# Use iotop or dstat to monitor real-time disk I/O usage by processes.

4. Check Process Limits:

# Use ulimit to check and modify user limits on processes (e.g., maximum number of open files).





# Lab 5 : Managing Daemons and Background Services

#### Objective:Learn how to manage background services and daemons in Linux.

1. Start and Stop Services:

# Use systemctl to start, stop, and restart system services (e.g., systemctl start apache2, systemctl stop nginx).

2. Enable/Disable Services on Boot:

# Use systemctl enable and systemctl disable to manage whether a service starts on boot.

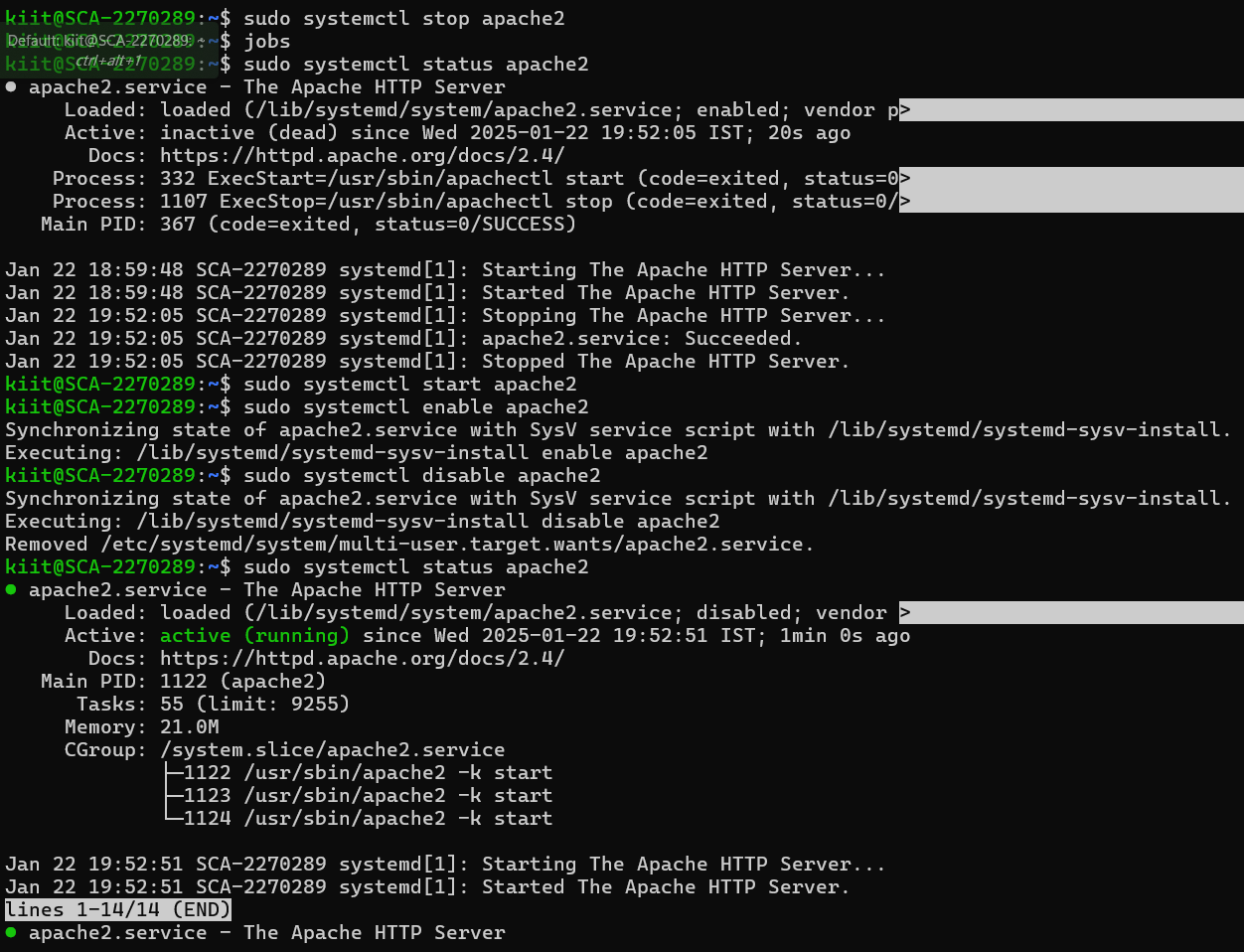
3. Check Service Status:

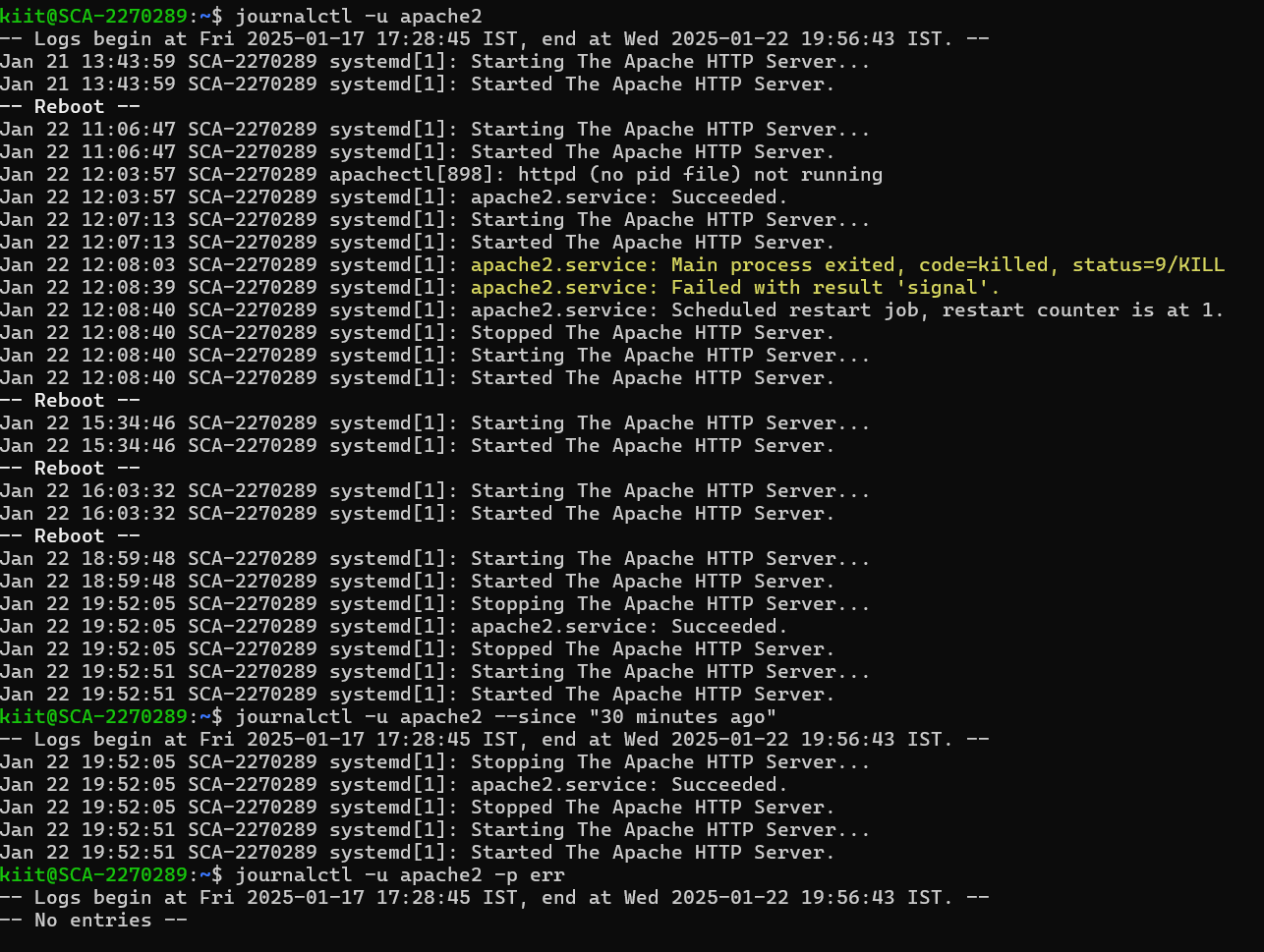
# Use systemctl status to check the status of a service (e.g., systemctl status apache2).

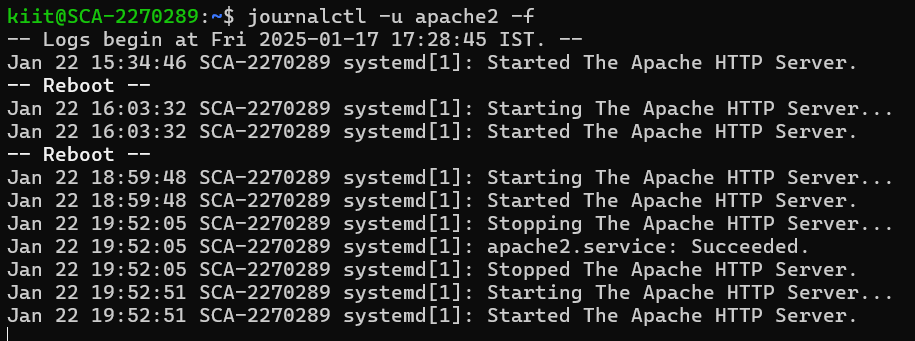
4. Managing Logs for Services:

# Use journalctl to check logs for systemd services.

# Filter logs for specific services or time periods to troubleshoot issues.







# Lab 6 : Process Scheduling and Prioritization

#### Objective: Learn how to control process priority and manage process scheduling.

1. Change Process Priority (Nice Value):

# Use nice to start a new process with a custom priority level (e.g., nice -n 10 command).

# Use renice to change the priority of an already running process by its PID (e.g., renice -n -5 <PID>).

2. Scheduling Processes:

# Use at to schedule a one-time task (e.g., at 09:00 to run a script).

# Use cron to schedule recurring tasks by adding entries to /etc/crontab or using crontab -e for user-specific jobs.

3. Monitor Process Execution Time:

# Use time to measure the execution time of a command or script.

