⚙️ Linux Process Management

📘 Scenario Overview

During routine operations at LabEx, the main application server began experiencing severe performance degradation. With senior administrators unavailable, I was tasked with diagnosing the issue by inspecting running processes, identifying resource-intensive scripts, terminating non-critical offenders, and ensuring critical services remained operational.

This lab simulates **real-world live server troubleshooting**, where process awareness and decisive action are essential.

🎯 Objectives

- Inspect all running system processes
- Monitor CPU usage in real time
- Identify critical vs. non-critical processes
- Terminate a misbehaving process safely
- Run long-running jobs in the background without session dependency

---

📋 Step 1: List All Active Processes

Purpose

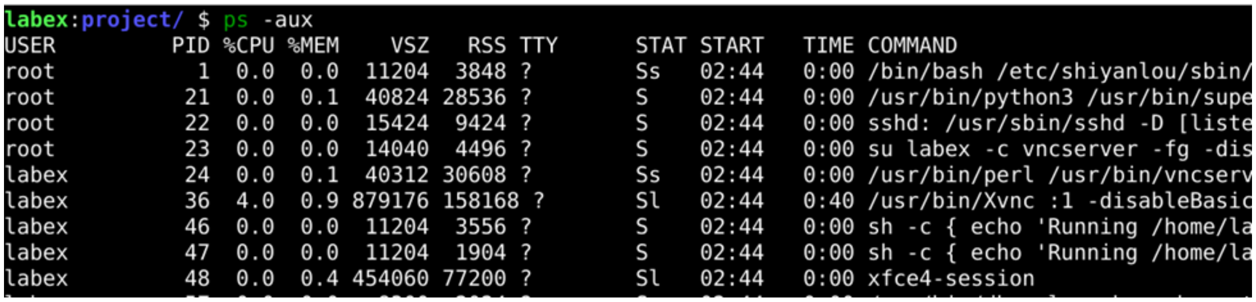
Obtain a complete snapshot of every running process on the system.

Command Used

```
ps aux
```

Why This Matters

- Displays processes for **all users**
- Shows CPU and memory usage
- Reveals full command paths for investigation



```
labex:project/ $ ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0  11204  3848 ?        Ss   02:44   0:00 /bin/bash /etc/shiyanlou/sbin/
root          21  0.0  0.1  40824 28536 ?        S    02:44   0:00 /usr/bin/python3 /usr/bin/supe
root          22  0.0  0.0  15424  9424 ?        S    02:44   0:00 sshd: /usr/sbin/sshd -D [liste
root          23  0.0  0.0  14040  4496 ?        S    02:44   0:00 su labex -c vncserver -fg -dis
labex         24  0.0  0.1  40312 30608 ?        Ss   02:44   0:00 /usr/bin/perl /usr/bin/vncserv
labex         36  4.0  0.9 879176 158168 ?       Sl   02:44   0:40 /usr/bin/Xvnc :1 -disableBasic
labex         46  0.0  0.0  11204  3556 ?        S    02:44   0:00 sh -c { echo 'Running /home/la
labex         47  0.0  0.0  11204  1904 ?        S    02:44   0:00 sh -c { echo 'Running /home/la
labex         48  0.0  0.4 454060 77200 ?        Sl   02:44   0:00 xfce4-session
```

📊 Step 2: Monitor Processes in Real Time

Purpose

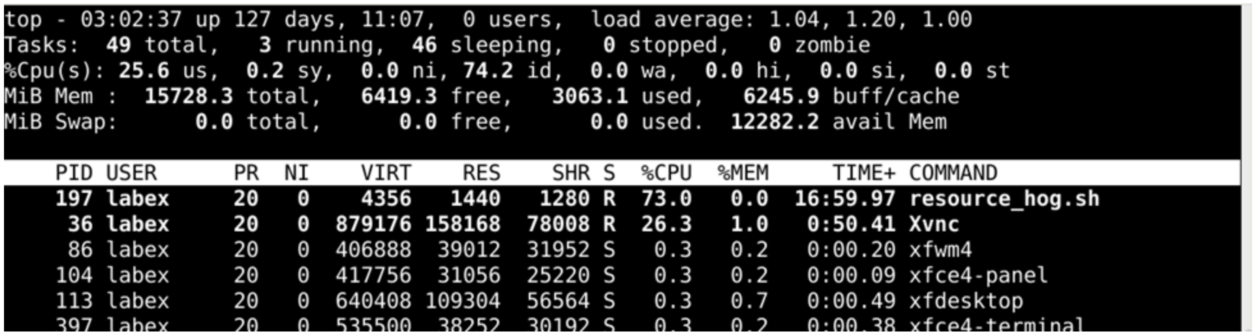
Identify which process is actively consuming system resources.

Command Used

```
top
```

Findings

- Processes sorted by CPU usage by default
- Identified **resource_hog.sh** as the top CPU consumer
- Exited `top` after confirmation



```
top - 03:02:37 up 127 days, 11:07,  0 users,  load average: 1.04, 1.20, 1.00
Tasks:  49 total,   3 running,  46 sleeping,   0 stopped,   0 zombie
%Cpu(s): 25.6 us,  0.2 sy,  0.0 ni, 74.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  15728.3 total,   6419.3 free,   3063.1 used,   6245.9 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.  12282.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    197 labex     20   0    4356   1440   1280 R  73.0   0.0  16:59.97 resource_hog.sh
     36 labex     20   0  879176 158168  78008 R  26.3   1.0   0:50.41 Xvnc
     86 labex     20   0  406888  39012  31952 S   0.3   0.2   0:00.20 xfwm4
    104 labex     20   0  417756  31056  25220 S   0.3   0.2   0:00.09 xfce4-panel
    113 labex     20   0  640408 109304  56564 S   0.3   0.7   0:00.49 xfdesktop
    397 labex     20   0  535500  38252  30192 S   0.3   0.2   0:00.38 xfce4-terminal
```

🔍 Step 3: Identify Critical Services

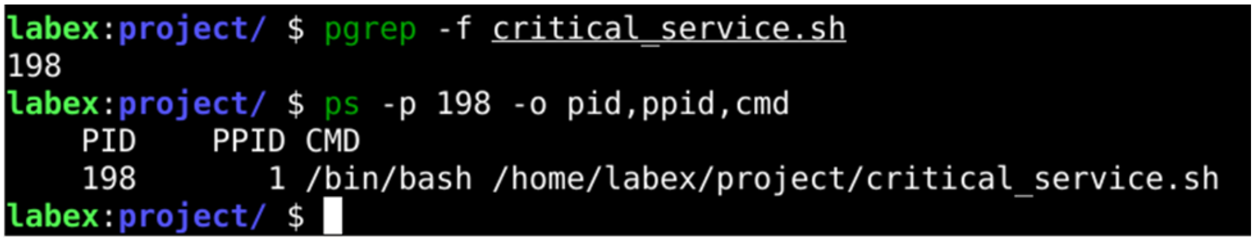
Before terminating any process, critical services were verified.

Command Used

```
pgrep critical_service.sh
```

Result

- Successfully returned a PID
- Confirmed `critical_service.sh` was running normally
- Ensured no disruption to essential services



```
labex:project/ $ pgrep -f critical_service.sh
198
labex:project/ $ ps -p 198 -o pid,ppid,cmd
    PID    PPID CMD
    198       1 /bin/bash /home/labex/project/critical_service.sh
labex:project/ $ 
```

🛑 Step 4: Terminate the Misbehaving Process

Target

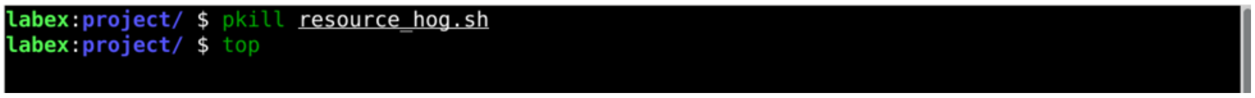
- `resource_hog.sh`

Command Used

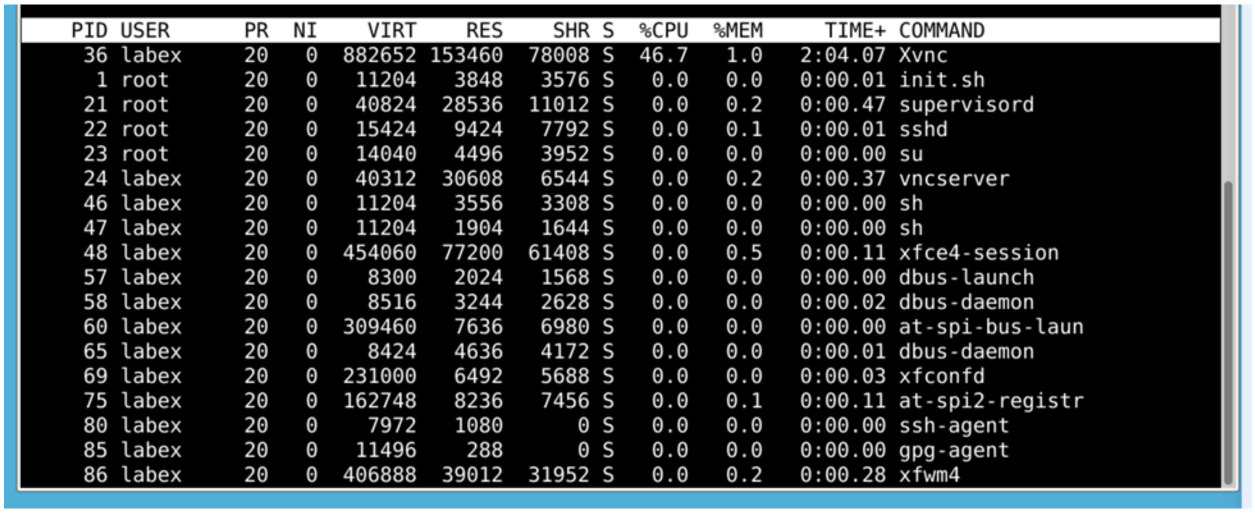
```
pkill resource_hog.sh
```

Outcome

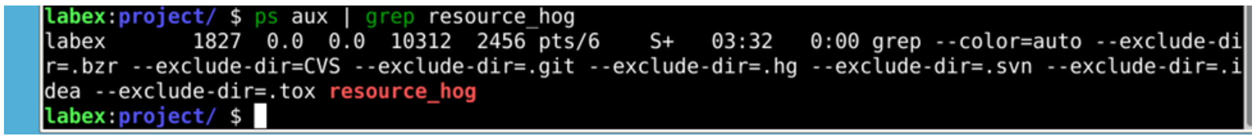
- Process terminated by name (no PID required)
- System performance stabilized
- Critical services unaffected



```
labex:project/ $ pkill resource_hog.sh
labex:project/ $ top
```



```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   36 labex     20   0  882652 153460  78008 S  46.7   1.0   2:04.07 Xvnc
    1 root      20   0   11204   3848   3576 S   0.0   0.0   0:00.01 init.sh
   21 root      20   0   40824  28536  11012 S   0.0   0.2   0:00.47 supervisord
   22 root      20   0   15424   9424   7792 S   0.0   0.1   0:00.01 sshd
   23 root      20   0   14040   4496   3952 S   0.0   0.0   0:00.00 su
   24 labex     20   0   40312  30608   6544 S   0.0   0.2   0:00.37 vncserver
   46 labex     20   0   11204   3556   3308 S   0.0   0.0   0:00.00 sh
   47 labex     20   0   11204   1904   1644 S   0.0   0.0   0:00.00 sh
   48 labex     20   0  454060  77200  61408 S   0.0   0.5   0:00.11 xfce4-session
   57 labex     20   0    8300   2024   1568 S   0.0   0.0   0:00.00 dbus-launch
   58 labex     20   0    8516   3244   2628 S   0.0   0.0   0:00.02 dbus-daemon
   60 labex     20   0  309460   7636   6980 S   0.0   0.0   0:00.00 at-spi-bus-laun
   65 labex     20   0    8424   4636   4172 S   0.0   0.0   0:00.01 dbus-daemon
   69 labex     20   0  231000   6492   5688 S   0.0   0.0   0:00.03 xfconfd
   75 labex     20   0  162748   8236   7456 S   0.0   0.1   0:00.11 at-spi2-registr
   80 labex     20   0    7972   1080      0 S   0.0   0.0   0:00.00 ssh-agent
   85 labex     20   0   11496    288      0 S   0.0   0.0   0:00.00 gpg-agent
   86 labex     20   0  406888  39012  31952 S   0.0   0.2   0:00.28 xfwm4
```



```
labex:project/ $ ps aux | grep resource_hog
labex       1827  0.0  0.0  10312  2456 pts/6    S+   03:32   0:00 grep --color=auto --exclude-di
r=.bzr --exclude-dir=CVS --exclude-dir=.git --exclude-dir=.hg --exclude-dir=.svn --exclude-dir=.i
dea --exclude-dir=.tox resource_hog
labex:project/ $ 
```

🔁 Step 5: Run a Long-Running Script in the Background

A developer requested execution of a long-running script that must persist after logout.

Requirements

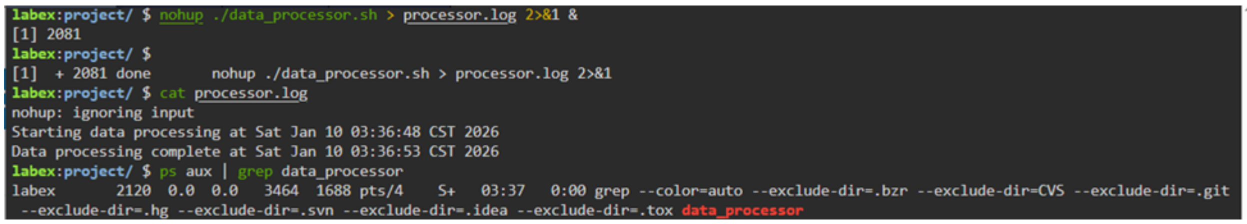
- Run from `~/project`
- Immune to terminal hangups
- Log all output

Command Used

```
cd ~/project
nohup ./data_processor.sh > processor.log 2>&1 &
```

Explanation

- `nohup` → prevents termination on logout
- `&` → runs process in background
- `> processor.log 2>&1` → captures stdout and stderr



```
labex:project/ $ nohup ./data_processor.sh > processor.log 2>&1 &
[1] 2081
labex:project/ $
[1]  + 2081 done       nohup ./data_processor.sh > processor.log 2>&1
labex:project/ $ cat processor.log
nohup: ignoring input
Starting data processing at Sat Jan 10 03:36:48 CST 2026
Data processing complete at Sat Jan 10 03:36:53 CST 2026
labex:project/ $ ps aux | grep data_processor
labex       2120  0.0  0.0   3464  1688 pts/4    S+   03:37   0:00 grep --color=auto --exclude-dir=.bzr --exclude-dir=CVS --exclude-dir=.git
 --exclude-dir=.hg --exclude-dir=.svn --exclude-dir=.idea --exclude-dir=.tox data_processor
```

🧠 Skills Demonstrated

- Process inspection (`ps`)
- Real-time monitoring (`top`)
- Process identification (`pgrep`)
- Safe process termination (`pkill`)
- Background job management
- Output redirection & job control
- Live server troubleshooting under pressure