LINUX SYSTEMS ADMINISTRATION · UBUNTU

Boot Process, Services and Processes

Presenter: _____

INTRODUCTION

What You Will Learn Today

By the end of this lesson you will be able to:

1. Describe the stages of the Linux boot process
2. Explain what systemd, units and targets are
3. Start, stop, enable and check services with systemctl
4. Read service logs with journalctl
5. Schedule tasks with cron and systemd timers
6. View running processes and stop them safely
7. Create, secure and test your own service

1. The Linux Boot Process

When you switch on the computer, five stages run in this order before you can log in.

Stage 1

UEFI

Checks the hardware and finds the bootloader on the disk

Stage 2

GRUB

Shows the boot menu and loads the kernel into memory

Stage 3

Kernel

The core of Linux. Takes control of the CPU, memory and devices

Stage 4

initramfs

A small temporary system that helps the kernel reach the main disk

Stage 5

systemd

The first program to run. It starts all the services

After Stage 5, the login screen appears.

1.1 Checking How Your System Booted

These commands show how long the boot took and what slowed it down.

Command	What it shows
`systemd-analyze`	The total time the last boot took
`systemd-analyze blame`	Every service, with how long each one took to start
`systemd-analyze blame | head`	Only the ten slowest services

Try it now: run systemd-analyze blame | head on your machine and note the slowest service.

2. What is systemd?

systemd is the program that starts and manages everything else on the system.

Definition	systemd is the init system and service manager used by Ubuntu.
PID 1	It is the first program the kernel starts, so its process ID (PID) is always 1.
Its job	It starts services at boot, stops them at shutdown, and restarts them if they crash.

```
ps -p 1 -o pid,comm        # shows that PID 1 is systemd
```

2.1 What is a Unit?

A unit is anything systemd manages. Each unit is described in a small text file.

Unit type	What it is	Example
`.service`	A program that runs in the background	nginx.service, ssh.service
`.target`	A group of units that make a system state	multi-user.target
`.timer`	A schedule that starts a service at set times	backup.timer
`.socket`	Starts a service when something connects to it	ssh.socket

The ending of the file name tells you what type of unit it is.

2.2 What is a Target?

A target is the state the system is trying to reach when it boots.

Target	Meaning
`multi-user.target`	A full server with networking, but no desktop. Normal for servers
`graphical.target`	Everything in multi-user, plus a desktop
`rescue.target`	A minimal system used to repair problems

```
systemctl get-default      # shows which target your system boots into
```

2.3 Managing Services with systemctl

systemctl is the command used to control services. Replace nginx with any service name.

Command	What it does
`sudo systemctl start nginx`	Starts the service now
`sudo systemctl stop nginx`	Stops the service now
`sudo systemctl restart nginx`	Stops it, then starts it again
`sudo systemctl enable nginx`	Makes it start automatically at every boot
`sudo systemctl disable nginx`	Stops it starting automatically at boot
`systemctl status nginx`	Shows whether it is running, and its recent messages

2.4 The Difference Between Start and Enable

This is the most common mistake beginners make with services.

`start`

Runs the service now.

After a reboot, it will NOT be running.

`enable`

Sets it to start at every boot.

It does NOT start it now.

```
sudo systemctl enable --now nginx      # does both: starts now AND at every boot
```

Rule: to keep a service running after a reboot, you must enable it.

2.5 Reading the Status of a Service

systemctl status shows four important pieces of information.

```
● nginx.service - A high performance web server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled)
     Active: active (running) since Mon 2026-09-21 08:14:02 EAT; 2h ago
   Main PID: 1284 (nginx)
```

Line	What it tells you
Loaded	Where the service file is, and whether it is enabled at boot
Active	running = working, inactive = stopped, failed = crashed
Main PID	The process ID of the running program
Last lines	The most recent messages. Errors usually appear here

SECTION 3: LOGS

3. Viewing Logs with journalctl

Every service writes messages to the system log. journalctl lets you read and search them.

Command	What it shows
`journalctl -u nginx`	All messages from the nginx service
`journalctl -u nginx -b`	Messages since the last boot only
`journalctl -f`	New messages as they happen (Ctrl+C to stop)
`journalctl -p err`	Error messages only
`journalctl --since "1 hour ago"`	Messages from the last hour

When a service fails: first run systemctl status, then read its log with journalctl -u.

4. Scheduling Tasks with cron

cron runs a command automatically at a time you set. Each job has five time fields and a command.

Minute	Hour	Day	Month	Weekday	then the command
0–59	0–23	1–31	1–12	0–7	

Example	Meaning
`0 2 * * *  backup.sh`	Every day at 2:00 a.m.
`*/15 * * * *  check.sh`	Every 15 minutes
`30 7 * * 1-5  report.sh`	At 7:30 a.m., Monday to Friday

```
crontab -e     # add or edit your jobs          crontab -l     # list your jobs
```

- * Matches every value, every minute
- , matches the list of values 0(sunday ,4(thur
- - matches the range of values 0-3
- / matches steps /0(every Sunday) /15 every after 15mins

4.1 Scheduling Tasks with systemd Timers

A timer does the same job as cron, using two files: a service (what to run) and a timer (when).

```
# /etc/systemd/system/backup.timer
[Timer]
OnCalendar=*-*-* 02:00:00
Persistent=true

[Install]
WantedBy=timers.target
```

Runs at 2 a.m. daily

```
OnCalendar=*-*-* 02:00:00
```

Catches up if the server was off

```
Persistent=true
```

See all timers

```
systemctl list-timers
```

Use cron for simple jobs. Use a timer when you need logs and missed jobs to run later.

5. What is a Process?

A process is a program that is running. Every process has a number called a PID.

State	Name	Meaning
R	Running	Using the CPU right now
S	Sleeping	Waiting for work. This is normal
D	Waiting on disk	Waiting for the disk. It cannot be stopped until the wait ends
T	Stopped	Paused
Z	Zombie	Finished, but not yet cleaned up by the program that started it

5.1 Viewing Running Processes

Use these tools to see which processes are running and how much they use.

Command	What it does
`top`	A live list of processes. Press P to sort by CPU, M by memory, q to quit
`ps aux`	A snapshot list of every process
`ps aux --sort=-%cpu | head`	The ten processes using the most CPU
`pgrep -a nginx`	Finds a process by its name and shows its PID

Try it now: open top, press P, and name the process using the most CPU.

5.2 Stopping a Process with Signals

A signal is a message sent to a process. Always try the gentle signal first.

Signal	Number	What it does
SIGTERM	15	Asks the process to stop. It can save its work first. Try this first
SIGKILL	9	Forces the process to stop at once. Unsaved work is lost. Use last
SIGHUP	1	Tells many services to reload their settings

```
kill 1284          # sends SIGTERM (gentle)
kill -9 1284       # sends SIGKILL (forced)
```

If the process is a service, stop it with systemctl stop instead of kill.

SECTION 6: LAB

6. Lab: Create a Monitoring Service

You will create a service that records the system's health every 10 seconds.

1. Step 1: Write a script that prints the system's health
2. Step 2: Create a service file so systemd runs the script
3. Step 3: Add settings that make the service secure
4. Step 4: Start the service and test that it works

Before you start: take a snapshot of your virtual machine.

SECTION 6: LAB

6.1 Step 1: Write the Script

Create the file /usr/local/bin/sys-monitor.sh with this content.

```
#!/bin/bash
while true; do
  LOAD=$(cut -d' ' -f1-3 /proc/loadavg)
  MEM=$(free -m | awk '/Mem:/{print $7}')
  DISK=$(df -h / | awk 'NR==2{print $5}')
  echo "Load: $LOAD | Free memory: ${MEM}MB | Disk used: $DISK"
  sleep 10
done
```

```
sudo chmod +x /usr/local/bin/sys-monitor.sh      # make the script executable
```

6.2 Step 2: Create the Service File

Create the file /etc/systemd/system/sys-monitor.service with this content.

```
[Unit]
Description=System health monitor
After=network.target

[Service]
ExecStart=/usr/local/bin/sys-monitor.sh

[Install]
WantedBy=multi-user.target
```

[Unit]

Describes the service

[Service]

Says which program to run

[Install]

Says it should start at boot

SECTION 6: LAB

6.3 Step 3: Make the Service Secure

Add these lines inside the [Service] section of the same file.

```
User=nobody
NoNewPrivileges=yes
ProtectSystem=strict
PrivateTmp=yes
Restart=on-failure
RestartSec=5
```

`User=nobody`
Runs the service as a user with almost no rights

`NoNewPrivileges=yes`
The service can never gain extra rights

`ProtectSystem=strict`
The service cannot change any system files

`PrivateTmp=yes`
The service gets its own private temporary folder

`Restart=on-failure`
If the service crashes, systemd starts it again

6.4 Step 4: Start and Test the Service

Run these commands in order. Check the result of each one before the next.

Command	What you should see
`sudo systemctl daemon-reload`	Nothing. systemd has read your new file
`sudo systemctl enable --now sys-monitor`	The service starts, and will start at boot
`systemctl status sys-monitor`	active (running), and a Main PID
`journalctl -u sys-monitor -f`	A new health line every 10 seconds
`sudo kill -9 <Main PID>`	The service crashes on purpose
`systemctl status sys-monitor`	Running again after 5 seconds, with a new PID

A new PID after the crash proves that Restart=on-failure is working.

SECTION 6: LAB

6.5 Troubleshooting Exercise

In pairs: make ONE of these changes on your partner's machine. They must find and fix it.

	The problem you create	How your partner finds it
A	Type a wrong path in ExecStart, then restart	systemctl status, then journalctl -u
B	Remove execute permission from the script	journalctl -u shows ‘permission denied’
C	Disable the service, then reboot	systemctl status shows inactive
D	Run yes > /dev/null & to use up the CPU	top, sorted by CPU with P

Always check in this order: status, then logs, then fix.

REVIEW

Review Questions

Answer these without looking at your notes.

1. What is the process ID of systemd, and why?
2. What is the difference between start and enable?
3. Which command shows the log of one service?
4. Which signal should you try first to stop a process, and why?
5. What does Restart=on-failure do?

Summary

The key points from today's lesson.

Boot	UEFI, GRUB, kernel, initramfs, then systemd starts everything
systemd	The first program (PID 1). It manages units, services and targets
systemctl	start = now. enable = at every boot. status = check it
journalctl	Read a service's log with journalctl -u <name>
Scheduling	cron for simple jobs; timers when you need logs and catch-up
Processes	Find them with top or ps. Stop with SIGTERM first, SIGKILL last