Linux: System Administration

TEKsystems Global Services

WORKFORCE DEVELOPMENT

LOGISTICS

Class Hours:

- Instructor will set class start and end times.
- There will be regular breaks in class.

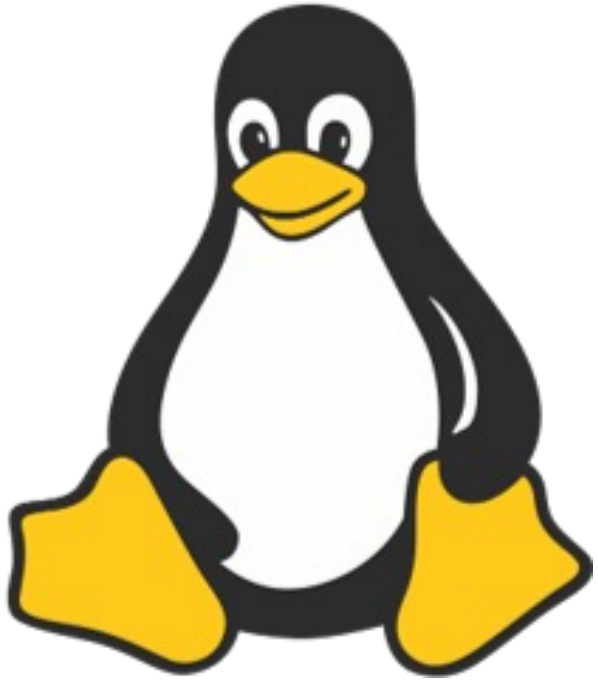
Telecommunication:

- Turn off or set electronic devices to silent (not vibrate)

Learning:

- Run the commands with the instructor as the slides are presented to you
- Ask questions and participate

Objectives



1. Identify and safely modify key Linux configuration files.
2. Automate system tasks using both legacy (`cron`) and modern (`systemd timers`) scheduling tools.
3. Understand and manage SELinux to maintain secure system access controls.

Linux: System Config Files

Configuration files are at the heart of Linux administration. They determine how services start, how users gain access, how logs are managed, and how the system itself behaves. Nearly every aspect of the operating system — from SSH access to kernel parameters — is controlled by editable text files.

For a Linux administrator, understanding these files is essential for maintaining stability and security. Knowing where they're located, what they control, and how to safely modify them enables you to customize systems, enforce policies, and troubleshoot issues effectively.

Linux: System Config Files

🧩 Common Configuration Files in Linux

Configuration files define how the system behaves and how users interact with it.

They control access, services, logging, network settings, and user environments — forming the backbone of Linux system administration.

- Most are found in **/etc/** — the central configuration directory
- User-specific configs live in your **home directory** (e.g., ~/.bashrc, ~/.ssh/config)
- Modular configurations are stored in .d directories for clarity and safety
  - Example: /etc/sudoers.d/, /etc/logrotate.d/, /etc/sysctl.d/, /etc/profile.d/

Linux: System Config Files

Common Config Files

Area	Main Config File	Modular Directory
Sudo privileges	`/etc/sudoers`	`/etc/sudoers.d/`
Log rotation	`/etc/logrotate.conf`	`/etc/logrotate.d/`
SSH access	`/etc/ssh/sshd_config`	*(SSH keys: `~/.ssh/authorized_keys`)*
Shell environment	`/etc/profile`, `~/.bashrc`, `~/.bash_profile`	`/etc/profile.d/`
Kernel tuning	`/etc/sysctl.conf`	`/etc/sysctl.d/`

💡 *Takeaway Linux configurations are organized, modular, and text-based — once you understand **where they live**, you can control almost every aspect of the system.*

Linux: System Config Files

🧠 Understanding the `sudoers` File

The **sudoers file** controls which users can execute commands with elevated privileges.
It's located at **`/etc/sudoers`**, but **you should never edit this file directly.**
A single typo can lock you out of administrative access.

🚨 Adding new files to /etc/sudoers.d is far less risky
than modifying /etc/sudoers

🔍 Inspect First — Safe & Read-Only

```
sudo less /etc/sudoers
sudo ls -la /etc/sudoers.d/
```

Linux: System Config Files

🧰 Best Practice: Use Drop-ins

Instead of changing /etc/sudoers, create separate files in /etc/sudoers.d/.

Edit safely using visduo and drop file:
sudo visudo -f /etc/sudoers.d/alice

Validate the file:
sudo visudo -c /etc/sudoers.d/alice

Linux: System Config Files

Example: Least Privilege Access

Allow Alice to check service status and logs (read-only)

```
# /etc/sudoers.d/alice
alice ALL=(ALL) NOPASSWD: /usr/bin/systemctl status *, /usr/bin/journalctl
```

Below is an example of full sudo access to all commands (unsafe)

```
rob ALL=(ALL:ALL) ALL
```

If rob has ALL command access, rob would be able to su to root

```
sudo su – root
# this would not work for Alice
```

Linux: System Config Files

Log Rotation with logrotate — Overview

System logs grow constantly as services run.
If unmanaged, they can fill the filesystem and lead to system failures.

The **logrotate** utility automatically handles this by rotating, compressing, and removing old log files.

It helps administrators:

- Prevent disk space exhaustion
- Maintain predictable log sizes
- Keep older logs archived and compressed

Logrotate is an essential background process for long-running systems that produce logs daily.

💡 *Log rotation ensures stability and reliability by keeping logs under control.*

Linux: System Config Files

⚙️ Using the logrotate Command

logrotate runs automatically on a schedule — typically through **cron** or a **systemd timer** — but it can also be triggered manually for testing or maintenance.

To inspect configuration files:

```
cat /etc/logrotate.conf
ls -la /etc/logrotate.d/
```

To run logrotate manually:

```
man logrotate
sudo logrotate -d /etc/logrotate.conf   # dry run (simulate)
sudo logrotate /etc/logrotate.conf      # perform rotation
```

💡 Knowing how to manually test log rotation helps verify service configurations without waiting for scheduled runs.

Linux: System Config Files

🔐 SSH Server Configuration

SSH (Secure Shell) allows secure remote access to Linux systems.
The main configuration file for the SSH server is:
less /etc/ssh/sshd_config

This file defines connection rules, authentication methods, and security controls.Administrators adjust settings here to harden remote access and prevent unauthorized logins.

Port 22
PermitRootLogin no
PasswordAuthentication yes
PubkeyAuthentication yes
MaxAuthTries 3
ClientAliveInterval 300

💡 *Do not make changes to this file in class.*
You would need to restart sshd if you did!

Linux: System Config Files

🏠 User-Level SSH Configuration

Each user can maintain their own SSH configuration and keys within their home directory. These files customize SSH behavior for client connections.

```
~/.ssh/config           # Personal SSH client settings
~/.ssh/id_rsa           # Private key (keep secure)
~/.ssh/id_rsa.pub       # Public key
~/.ssh/authorized_keys  # Keys allowed to log in
```

These settings define which hosts you connect to, authentication keys, and connection preferences — without affecting system-wide SSH behavior.

💡 *Keep private keys secure with correct permissions: chmod 600 ~/.ssh/id_rsa*

Linux: System Config Files

🧩 Shell Profiles and Environment Setup

Shell profile files execute when users log in or open terminals.
They define environment variables, PATH settings, and command aliases — shaping the user's working environment.

System-wide profiles affect **all users**, while user profiles only affect **the current account**.

Linux: System Config Files

🧩 Shell Profiles and Environment Setup

```
/etc/profile            # System-wide login settings
/etc/profile.d/ # drop folder for profiles

~/.bash_profile         # User-specific login shell
~/.bashrc               # User-specific interactive shell
~/.bash_logout          # Runs on logout
```

💡 Profiles help maintain consistency and automatically configure the environment at login.

Linux: System Config Files

🧠 Kernel Parameters and sysctl

The Linux kernel manages system resources — including memory, networking, process limits, and file handling.

Administrators can fine-tune these behaviors using **kernel parameters**, which adjust how the system performs and responds to workloads.

The sysctl utility provides a safe, runtime method for viewing and modifying these parameters **without rebooting**.

It's useful for performance tuning, troubleshooting, and enforcing limits in production environments.
Examples of what kernel parameters control:

- Maximum number of open files
- IP forwarding and TCP settings
- Process and PID limits
- Shared memory for applications like databases

💡 *Persistent tuning is done through configuration files rather than runtime changes.*

Linux: System Config Files

Using sysctl and Configuration Files

View common kernel parameters:
sysctl kernel.pid_max
sysctl fs.file-max
sysctl net.ipv4.ip_forward

List all parameters (very long output):
sysctl -a | head -n 20

View and edit configuration files:
cat /etc/sysctl.conf
ls -la /etc/sysctl.d/

The /etc/sysctl.conf file holds global kernel settings, while /etc/sysctl.d/ contains modular configuration files for specific tuning.

Lab 5.1 Common Config Files

Estimated Time: 30

POP QUIZ:

Where would I put settings to create a variable whenever anyone logs in?

A. ~/.bashrc
B. /etc/profile.conf
C. /etc/profile.d/
D. /.bin/

POP QUIZ:

Where would I put settings to create a variable whenever anyone logs in?

A. ~/.bashrc
B. /etc/profile.conf
C. /etc/profile.d/
D. /.bin/

"What does the .d mean?"

POP QUIZ:

What is the safe way to edit change sudo permissions?

A. Do not change sudo settings
B. visudo
C. vi /etc/sudoers.d/file
D. visudo -f /etc/sudoers.d/file

POP QUIZ:

What is the safe way to edit change sudo permissions?

A. Do not change sudo settings
B. visudo
C. vi /etc/sudoers.d/file
D. visudo -f /etc/sudoers.d/file

"What happens if you remove sudo on your own permissions after saving the sudo file"

Linux: Scheduling

Task scheduling is a core part of system administration. Every Linux system relies on automated tasks to perform routine maintenance—whether it's rotating logs, backing up data, or cleaning up temporary files. Automation ensures reliability, consistency, and efficiency by reducing the need for manual intervention and preventing overlooked maintenance.

For administrators, mastering scheduling tools is essential to keeping systems stable and predictable. From legacy cron jobs to modern systemd timers, these tools allow you to control when and how tasks run, track their execution, and recover gracefully from failures. Effective scheduling not only saves time but also helps maintain security, performance, and operational continuity across your infrastructure.

Linux: Scheduling

🕒 What Is Task Scheduling?

Task scheduling is the process of **automating commands or scripts** to run at specific times or intervals.

It allows the system to perform maintenance, monitoring, or updates **without human intervention**, ensuring consistency and reliability. These automations are great for House Keeping tools.

Avoid using them for business critical tasks like an update to application databases or client reports. Centralized solutions like control-m, autosys, jenkins, step functions, k8s-cronjobs

Common uses:

- Backups and log rotation
- Temporary file cleanup
- System updates or reports
- Monitoring and alerting scripts

Linux: Scheduling

⚙️ Two Main Scheduling Systems in Linux

1. Cron (Legacy Unix Scheduler)

- Simple, time-based syntax (minute hour day month weekday)
- Found on nearly all Linux systems
- Ideal for lightweight, recurring tasks
- Runs as a service (usually crond)

2. Systemd Timers (Modern Scheduler)

- Integrated with systemd for better logging and dependencies
- Can use calendar events or relative time intervals
- Preferred for modern environments and service automation

💡 *Both serve the same purpose — automating tasks — but systemd timers offer more control and visibility.*

Linux: Scheduling

🧩 Cron Syntax Overview

Cron jobs follow a **five-field format**, where each field represents a unit of time.
Each field can be a **number**, **range**, **list**, **asterisk (*)**, or **step value (*/n)**.

```
#  ┌──────────────────────── minute (0-59)
#  │   ┌──────────────────── hour (0-23)
#  │   │   ┌──────────────── day of month (1-31)
#  │   │   │   ┌──────────── month (1-12)
#  │   │   │   │   ┌──────── day of week (0-6, Sunday=0 or 7)
#  │   │   │   │   │
# * * * * * command_to_run
```

Use tools like https://crontab.guru/

💡 *Tip: Use crontab -e to create or edit your user's cron jobs.*
crontab –r deletes the crontab (the keys "e" and "r" are next to each other)

Linux: Scheduling

Cron Schedule Examples

Scenario	Cron Expression
Every minute	* * * * *
Every 15 minutes	*/15 * * * *
Every day at midnight	0 0 * * *
Every Friday at 5 PM	0 17 * * 5
Every Monday at 8 AM	0 8 * * 1
On the 1st of every month at 2 AM	0 2 1 * *
Every weekday at 9 AM	0 9 * * 1-5
Every Sunday at 3:30 AM	30 3 * * 0
Every 5th day of the month at noon	0 12 */5 * *
Once a year on January 1st at midnight	0 0 1 1 *

Linux: Scheduling

⏱️ Systemd Timer Syntax Basics

Systemd timers use a **calendar-based format** (like natural language) instead of numeric fields.
They define **when** a service runs — not *what* runs. The "what" goes in a matching .service file.

Two types of timers:

- **Monotonic timers** → Relative to system events (e.g., after boot)
- **Calendar timers** → Absolute times (like cron, but more readable)

```
# Examples of time directives
OnBootSec=5min        # 5 minutes after boot
OnUnitActiveSec=1h    # Every hour after last run
OnCalendar=daily      # Every day at midnight
OnCalendar=Mon *-*-* 09:00:00   # Every Monday at 9 AM
```

💡 *Think of systemd timers as cron jobs with superpowers — logs, dependencies, and retries built in.*

Linux: Scheduling

⚙️ Creating a Basic Systemd Timer

Systemd timers work **alongside service files** — the service defines *what* runs, the timer defines *when* it runs.

Assuming you have a .service file (e.g., hello.service) and .timer file (hello.timer) created.

Commands:
systemctl --user daemon-reload
systemctl --user enable --now hello.timer
systemctl --user list-timers

💡 *Timers run the matching service automatically (same base name).* hello.timer → runs hello.service

Linux: Scheduling

⚖️ Cron vs Systemd Timers — Quick Comparison

Feature	Cron	Systemd Timers
Syntax	Numeric fields	Natural language (calendar-based)
Logging	Mail / Syslog	Integrated with `journalctl`
Dependencies	None	Can depend on other units
Resource control	None	Supports cgroups & limits
Failure handling	Minimal	Retries & notifications
Boot/start control	Limited	`OnBootSec`, `OnStartupSec`
Best for	Simple, legacy systems	Modern services, automation

💡 *Use cron for quick scripts and backward compatibility. Use systemd timers for production and service reliability.*

Lab 5.2 Scheduling

Estimated Time: 30

POP QUIZ:

How often would 0 * */1 * 1-5 run?

A. Every minute once a day Monday thru Friday
B. On minute zero every hour Monday thru Friday
C. On minute zero every hour Tuesday thru Saturday
D. Every hour Monday thru Friday

POP QUIZ:

How often would 0 * */1 * 1-5 run?

A. Every minute once a day Monday thru Friday
B. On minute zero every hour Monday thru Friday
C. On minute zero every hour Tuesday thru Saturday
D. Every hour Monday thru Friday

"How many times a day does it run?
What if we replace */1 with 1?"

POP QUIZ:

What task below is a good use for an automation like cron of systemd timers?

A. An automation that cleans/terminates stuck processes
B. Daily client report via email
C. Synchronizing a remote database to your files
D. Updating a remote database

POP QUIZ:

What task below is a good use for an automation like cron of systemd timers?

"Pitfalls of using these tools for reports?"

A. An automation that cleans/terminates stuck processes
B. Daily client report via email
C. Synchronizing a remote database to your files
D. Updating a remote database

Linux: SELinux

Security-Enhanced Linux (SELinux) adds an extra layer of protection to Linux by enforcing strict, policy-based access controls. Instead of relying only on traditional permissions, SELinux defines what each process can access, helping contain threats and prevent unauthorized actions.

For system administrators, understanding SELinux is key to maintaining secure and reliable systems. By learning how to read contexts, manage modes, and interpret logs, admins can troubleshoot issues confidently while keeping strong, kernel-level security in place.

Linux: SELinux

🔒 What is SELinux?

Security-Enhanced Linux (SELinux) is a security framework built into the Linux kernel that enforces Mandatory Access Control (MAC) policies.

It defines what processes can access which files, ports, and resources, going beyond traditional user and group permissions.

Even if an attacker gains access to a service, SELinux limits what that process can do—containing potential damage and keeping the rest of the system secure.

💡 *SELinux protects the system from within by controlling interactions between programs and data at the most fundamental level.*

Linux: SELinux

🔒 How SELinux Controls Access

- Every file and process in Linux has a **security context**, stored as an extended attribute.
- When a service starts, SELinux assigns it a **process type** (domain) based on the label of its executable — not the file name.
- That process can only access files and directories with **types** allowed by its policy.
- If you need a process to reach another directory, simply **label that directory** with a compatible context (e.g., using semanage fcontext and restorecon).

💡 SELinux enforces access through labels — not permissions — letting you safely control what each service can see or do.

Linux: SELinux

🧭 Viewing SELinux Status with sestatus

The sestatus command provides a detailed summary of SELinux configuration and runtime state. It's the quickest way to verify if SELinux is active and how it's set to behave.

- **SELinux status** – Whether SELinux is enabled or disabled on the system.
- **Current mode** – The mode currently in effect (enforcing, permissive, or disabled).
- **Mode from config file** – The mode that will be applied **after the next reboot** (read from /etc/selinux/config).
- **Policy name** – The active policy type, usually targeted.

💡 If the **current mode** and **mode from config file** differ, the system will switch to the configured mode on the next reboot — so always check both before restarting.

```
[ec2-user@ip-172-31-1-87 ~]$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          permissive
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      33
```

Linux: SELinux

⚙️ SELinux Modes

SELinux can operate in one of three modes, controlling how strictly it enforces security policies:

- **Enforcing** – SELinux is fully active. Unauthorized actions are **blocked** and logged. This is the normal and recommended mode for production systems.
- **Permissive** – SELinux policies are loaded, but violations are **only logged**, not blocked. Useful for troubleshooting or testing policy changes.
- **Disabled** – SELinux is completely **turned off**. No access controls or logs are applied, and the system loses kernel-level protection.

💡 Use *permissive* mode temporarily for debugging, but keep *enforcing* mode enabled in production to maintain proper security boundaries.

Linux: SELinux

🧭 Changing SELinux Mode

You can switch SELinux modes temporarily or permanently:

Temporarily (no reboot):

```
sudo setenforce 0   # Switch to permissive
sudo setenforce 1   # Switch back to enforcing
# (Changes take effect immediately but reset after reboot.)
```

Permanently (after reboot):

```
# Edit /etc/selinux/config and set:
# SELINUX=enforcing
```

💡 Always confirm your change with sestatus — it shows both the current mode and what will load after the next reboot.

Linux: SELinux

Viewing SELinux Logs

When SELinux blocks an action, it records a **denial** event in the system logs. These messages help identify what was prevented and why.

Common ways to view them:

```
# View recent SELinux denials
sudo ausearch -m avc -ts recent

# Check SELinux messages in the system journal
sudo journalctl -t setroubleshoot --since "1 hour ago" --no-pager
```

💡 *Look for lines containing **"denied"** and the process name (e.g., comm="httpd"). These entries show exactly which access attempt SELinux prevented and are key to troubleshooting.*

Linux: SELinux

📌 Understanding SELinux Contexts

Every file and process in SELinux has a **security context**, which defines how they can interact. A context is written as:

user:role:type:level

- **user** – The SELinux user (e.g., system_u for system files)
- **role** – The role assigned to the object or process (e.g., object_r, system_r)
- **type** – The most important field; determines access rules between files and processes
- **level** – Used for multi-level security systems (often s0 on most systems)

💡 SELinux makes access decisions based on **type enforcement** — what one type (process) is allowed to do with another type (file).

Linux: SELinux

SELinux in Action: SSH Example

In the SSH directory, the key files and daemon share related SELinux types:

- SSH key files → sshd_key_t
- SSH daemon process → sshd_t

This matching relationship is intentional. SELinux policies explicitly allow sshd_t processes to read files labeled sshd_key_t, while all other processes are denied access if SELinux is enabled— **even if they run as root.**

```
[ec2-user@ip-172-31-1-87 ~]$ ls -Z /etc/ssh/*key*
system_u:object_r:sshd_key_t:s0 /etc/ssh/ssh_host_ecdsa_key
system_u:object_r:sshd_key_t:s0 /etc/ssh/ssh_host_ecdsa_key.pub
system_u:object_r:sshd_key_t:s0 /etc/ssh/ssh_host_ed25519_key
system_u:object_r:sshd_key_t:s0 /etc/ssh/ssh_host_ed25519_key.pub
[ec2-user@ip-172-31-1-87 ~]$ ps -eZ | grep sshd
system_u:system_r:sshd_t:s0-s0:c0.c1023 1572 ?   00:00:03 sshd
system_u:system_r:sshd_t:s0-s0:c0.c1023 454988 ? 00:00:00 sshd
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 455102 ? 00:00:00 sshd
```

Linux: SELinux

🛠️ Managing SELinux File Contexts

SELinux labels control how files can be accessed. In practice you'll either **view**, **define a persistent rule**, or **apply/temporarily change** a label.

```
# View a file/dir label
ls -Z /path/to/file

# Define a PERSISTENT rule for a path (policy database)
sudo semanage fcontext -a -t httpd_sys_content_t "/path(/.*)?"

# Apply the rule to the filesystem now (reads policy, fixes labels)
sudo restorecon -Rv /path

# TEMPORARY label change (does not survive relabels)
sudo chcon -t httpd_sys_content_t /path/to/file

# (Optional) List or remove fcontext rules
sudo semanage fcontext -l | grep /path
sudo semanage fcontext -d "/path(/.*)?"
```

💡 fcontext makes labeling **persistent**; restorecon **applies** those rules to files; chcon is **temporary** and will be overwritten by a relabel.

Linux: SELinux

🎯 Key Takeaways

- SELinux enforces fine-grained access control between processes and files.
- Most access issues are fixed by correcting file contexts (restorecon).
- Use *permissive* mode only for troubleshooting.
  Keep *enforcing* in production.
- Always check logs (ausearch, journalctl) before making changes.
- SELinux booleans can adjust service behavior (like allowing web apps network access) without changing core policy.

💡 SELinux isn't something to disable — it's a system-wide safety net that protects your services from each other and from compromise.

Lab 5.3 SELinux

Estimated Time: 30

POP QUIZ:

What is the attribute that every file has and controls what services can access it?

A. Security Context
B. File Permissions
C. Stat
D. Setuid

POP QUIZ:

What is the attribute that every file has and controls what services can access it?

A. Security Context
B. File Permissions
C. Stat
D. Setuid

"What is the difference between Security Context and File Permissions?"

POP QUIZ:

When examining the Security Context, which is the most relevant part?

A. Role
B. User
C. Type
D. Level

POP QUIZ:

When examining the Security Context, which is the most relevant part?

A. Role
B. User
C. Type
D. Level

"How does this related to a processes allowed type?"

Linux: Conclusion

🧭 The Role of a System Administrator

- A System Administrator ensures systems remain **secure, stable, and efficient**.
- The role requires **resourceful problem-solving** — knowing how to diagnose and correct issues under pressure.
- Good practices include **automation, documentation, and continuous learning** to maintain consistency across environments.
- Collaboration, attention to detail, and steady habits make administration predictable and dependable.

💡 Strong system administration comes from understanding how everything connects — and keeping that knowledge organized, tested, and reliable.