PROG8820- Linux Administration

Winter 2025

Week 6-Linux User administration & file permissions

Aladdin AlRadhi
Aalradhi@conestogac.on.ca

Learning Outcomes

- 6.1 Demonstrate viewing logs in Linux.
- 6.2 Analyze logs for specific events using command line utilities.
- 6.3 Transform raw daemon logs into useful debugging snippets.

LINUX TEXT EDITORS

Linux Editors

- Linux offers a vast array of text editors that are diverse in simplicity, extensive customization, and powerful coding capabilities.
- Each of the editors serves different needs and preferences.
- These editors can be classified from different perspectives such as use-case and appearances.
- However, Linux text editors are generally classified into two types:
  - CLI-based Text Editors
  - GUI-based Text Editors

Use of Text Editors in Linux

- Notes:
  - Text editors can be used for taking notes, email writing or journaling.
  - The lightweight text editor such as Nano provides a clutter-free environment for quick texting.
- Coding:
  - Programmers and web developers use different text editors for different functionalities related to coding such as code navigation, debugging support, extensive plugin support for frameworks and so on.

Use of Text Editors in Linux

- Writing Scripts:
  - Text editors are used for writing scripts such as Bash that help in task automation, controlling system processes, and performing various system administration functions.
- Configuring System Files:
  - Text editors help in editing configuration files that manage system configuration, services and applications.
- Automation:
  - Using text editors, users can create automation scripts that automatically schedule and execute tasks at regular intervals.

Basic Editors

- Nano:
  - This beginner-friendly editor comes pre-installed on many Linux systems, making it perfect for quick edits and simple tasks.
  - With its intuitive on-screen prompts and straightforward commands, it’s ideal for Linux newbies.
- Vim (vi):
  - Vim, or its predecessor vi, is the power user’s weapon of choice.
  - It offers high performance among text editors, offering unparalleled speed and customization.
  - However, it demands practice and mastery of its modal editing and extensive keyboard shortcuts.

Basic Editors

- Emacs:
  - This can handle a vast majority of tasks.
  - It’s more than just an editor; it’s a full-fledged environment for writing code, managing projects, and even sending emails.
  - With its extensibility through Emacs Lisp, it’s a favorite among users who crave complete control.
- Gedit:
  - The default text editor for the GNOME desktop environment.
  - It’s a good choice for those who prefer a graphical editor with essential features like syntax highlighting and plugin support.

Basic Editors

- Kate:
  - If you’re a KDE user, Kate is your go-to advanced text editor.
  - It seamlessly integrates with KDE and offers robust features like multi-document editing, an integrated terminal, and extensive customization options.

Simple GUI Based Text Editors

GNOME Text Editor	Default text editor for the GNOME desktop environment
Pluma	Powerful text editor for MATE
Mousepad	Simple text editor for the Xfce desktop environment
FeatherPad	Lightweight Qt plain-text editor
Nota	Browse, create, and edit text files
CorePad	Document editor for C Suite, a minimalistic application suite
gedit	Small and lightweight text editor for GNOME
xed	Small and lightweight text editor
Howl	General purpose editor
Airpad	Basic, generic graphical text editor

Programmers' Editors

- Sublime Text:
  - Known for its speed, elegance, and extensive customization options, Sublime Text is a favorite among many developers. Its sleek interface, powerful search functionality, and vast plugin ecosystem make it a versatile tool for a variety of programming languages.
- Geany:
  - If you're looking for a lightweight yet feature-rich editor, Geany is a great choice.
  - It offers built-in support for many programming languages, along with features like code folding, auto-completion, and symbol lists, making it a great option for beginners and experienced developers alike.

Programmers' Editors

- Visual Studio Code:
  - Developed by Microsoft, Visual Studio Code (VS Code) has quickly become one of the most popular code editors on the market.
  - It's open-source, highly extensible, and boasts a wide range of features, including debugging tools, integrated terminals, and Git integration.
  - VS Code's massive library of extensions allows you to tailor it to your specific development needs.

Nano

- Step 1: Open the terminal.
  - Command: Ctrl + Alt + T
- Step 2: Type ` nano filename.txt ` and press Enter.
- Step 3: Edit the text file as needed.
- Step 4:To save the file, press ` Ctrl + O `, then Enter.
- Step 5:To exit Nano, press ` Ctrl + X `.

Vim

- Step 1: Open the terminal.
  - Command: Ctrl + Alt + T
- Step 2: Type ` vim filename.txt ` and press Enter.
- Step 3: Press ` i ` to enter insert mode and start editing.
- Step 4: Press Esc to exit insert mode.
- Step 5: To save and exit, ` type :wq ` and press Enter.

Emacs

- First install Emacs by the writing the command in the terminal :-
  - Command: apt install e3
- Step 1: Open the terminal.
  - Command: Ctrl + Alt + T
- Step 2: Type ` emacs filename.txt ` and press Enter.
- Step 3: Edit the file directly
- Step 4: To save the file, press ` Ctrl + X ` followed by ` Ctrl + S `.
- Step 5: To exit Emacs, press ` Ctrl + X ` followed by ` Ctrl + C `.

Emacs

emacs myfile	Start emacs and edit myfile
CTRL-x i	Insert prompted for file at current position
CTRL-x s	Save all files
CTRL-x CTRL-w	Write to the file giving a new name when prompted
CTRL-x CTRL-s	Saves the current file
CTRL-x CTRL-c	Exit after being prompted to save any modified files

Vi

- Command mode
  - By default Vi, starts in command mode
- Insert mode
  - i mode is used to insert text and modify file
  - Type i to enter insert mode and esc to get out of it.
- Line mode
  - Type : to enter line mode and use operations such as exit vi or save file

Vi commands

vi myfile	Start the editor and edit myfile
:w	Write to the file
:w myfile	Write out to myfile
:w! file2	Overwrite file2
:x or :wq	Exit and write out modified file
:q	Quit
:q!	Quit even though modifications have not been saved

Cursor positions in vi

arrow keys	To move up, down, left and right
j or <ret>	To move one line down
k	To move one line up
h or Backspace	To move one character left
l or Space	To move one character right
0	To move to beginning of line
$	To move to end of line
w	To move to beginning of next word
:0 or 1G	To move to beginning of file
:n or nG	To move to line n
:$ or G	To move to last line in file
CTRL-F or Page Down	To move forward one page
CTRL-B or Page Up	To move backward one page
^l	To refresh and center screen

Gedit

- Within the GNOME desktop environment, Gedit is the default text editor.
- With plugin support and syntax highlighting, it's easy to use yet extremely effective.
- How to edit text files in Linux using Gedit:
  - Locate the text file by opening the file manager and opening it.
  - When you do a right-click on the file, choose "Open with Gedit."
  - Make changes to the file and save it.

Kate

- The KDE desktop environment includes Kate, a potent text editor.
- It has sophisticated features like syntax highlighting and enables multiple document editing.
- How to edit text files in Linux using Kate:
    - From the apps menu, launch Kate.
    - Drop the text file into the Kate window by dragging it there.
    - After making changes, click "Save" to close the file.

LINUX LOGS

What are Linux Logs?

- Linux logs are detailed records generated by a Linux system that provide information about system activities, application behavior, and user interactions.
- They serve as a valuable tool for troubleshooting, performance monitoring, and security analysis.
- Whether you're managing a personal device or a large server, understanding and utilizing Linux logs effectively is critical for maintaining system health and ensuring smooth operations.

A Linux log would typically include

- System Events:
  - to track kernel activities, boot processes, and hardware interactions.
- Application Behavior:
  - Monitor software performance to detect application-level issues.
- Security Activities:
  - authentication attempts, firewall activities, and threats.
- Network Traffic:
  - Analyze connectivity, Wi-Fi issues, and domain processing errors.

Linux Logs Needs

- Linux logs also track user activities and system commands.
- Using tools or commands, administrators can review a detailed history of actions, including Linux log all commands executed by users.
- This is particularly useful for auditing purposes.
- To improve security, Linux log authentication entries provide insights into login attempts and potential unauthorized access.
- Additionally, steam Linux logs can help gamers diagnose issues with the Steam client, while crontab Linux logs reveal the execution status of scheduled tasks.

Linux Log management

- For effective log management, learning how to read Linux logs is a crucial skill.
- Commands like cat, less, and tail allow users to view logs directly.
- For example, using tail Linux logs enables real-time monitoring of a log file's most recent entries, which is helpful when debugging ongoing issues.
- Advanced users often rely on Linux log analysis tools to process and interpret large volumes of data.
- These tools, such as Graylog or the ELK Stack, offer powerful Linux log aggregation and visualization features.

How can Linux Logs Help?

- Linux logs serve as a record of everything happening on your system.
- They detail system events, startup sequences, and shutdown processes.
- For instance, Linux log boot files capture all the information related to the system’s initialization, including loaded services and potential errors during startup.
- Similarly, Linux logs before reboot allow you to identify issues that might have led to a system failure or prompted a restart.
- By analyzing these logs, you can pinpoint recurring issues and address them proactively.

Troubleshooting Errors

- One of the most common uses of Linux logs is diagnosing and resolving system errors.
- Whether an application is crashing, or a service isn’t starting, the logs can provide detailed error messages and timestamps.
- For example, by accessing Linux logs as root, administrators can view privileged information that may not be accessible to standard users.
- These logs often include critical system files, such as /var/log/syslog or /var/log/messages, which are essential for understanding system-wide errors.
-

Troubleshooting Errors

- For less sensitive tasks, you can access Linux logs as a user.
- This approach is safer for regular analysis and avoids the risks associated with elevated privileges.
- Tools like less, cat, or graphical interfaces make it easy to explore log files for troubleshooting purposes.

Tracking User Activities and Commands

- Linux logs are vital for auditing and monitoring user activities.
- Using commands like history in conjunction with logging tools, you can examine Linux log all commands executed by users on the system.
- This feature is particularly useful for identifying unauthorized changes or tracking specific activities performed by users.
- For example, developers can review command logs to debug scripts, while security teams can audit activities for compliance purposes.

Enhancing Security

- Security monitoring is one of the most critical functions of Linux logs.
- Logs like /var/log/auth.log provide Linux log authentication details, including successful and failed login attempts.
- These logs can help detect potential brute-force attacks or unauthorized access to your system.
- By analyzing authentication logs, you can identify suspicious patterns and strengthen your security measures, such as implementing stricter password policies or enabling two-factor authentication.

Centralized Log Management with Aggregation

- For larger environments, managing individual log files across multiple servers can be overwhelming.
- Linux log aggregation tools like the ELK Stack (Elasticsearch, Logstash, Kibana) or Graylog enable centralized collection and analysis of logs.
- By aggregating logs from multiple sources, you can create a unified view of your system's behavior, making it easier to identify trends and anomalies.
- These tools often include visualization features, allowing you to monitor system performance through dashboards and alerts.

Proactive Maintenance

- Regularly analyzing Linux logs can help in proactive system maintenance.
- By monitoring system health and performance indicators, you can detect potential issues before they escalate.
- For instance, disk usage warnings in logs can alert you to free up space before running out of storage.
- Additionally, reviewing Linux logs before reboot can help prevent unnecessary downtime by identifying unresolved issues.

Practical Applications in Real Scenarios

- Debugging System Startups:
  - By reviewing Linux log boot files, you can identify services that failed to start and take corrective actions.
- Detecting Intrusions:
  - Analyzing Linux log authentication entries helps detect unauthorized login attempts and implement stronger security measures.

Practical Applications in Real Scenarios

- Tracking Changes:
    - Linux log all commands executed by administrators allow you to trace system modifications, making it easier to revert or document changes.
- Aggregating Logs for Clarity:
    - Using Linux log aggregation tools consolidates data from multiple servers, simplifying complex log analysis tasks.

Linux Log Files Location

- Almost all log files are located under /var/log/ directory and its sub-directories on Linux.
- You can change to this directory using the cd command.
- Of course, you need to be the root user to access log files on Linux or Unix-like operating systems.

Linux Log Files Location

- /var/log/syslog or /var/log/messages:
  - These files store general system activity logs and are crucial for troubleshooting common issues.
- /var/log/auth.log:
  - Contains authentication-related logs, including login attempts and SSH activities.
- /var/log/kern.log:
  - Records kernel messages, useful for debugging hardware or kernel issues.

Linux Log Files Location

- /var/log/boot.log:
    - Tracks the system boot process.
- /var/log/dmesg:
    - Provides kernel ring buffer messages, including hardware detection during boot.

Important Linux logs

System Logs:

- System Linux logs track the operating system activities and events, such as boot processes, kernel events, and system performance metrics.
- These logs are saved in:
  - /var/log/syslog: General system activity logs.
  - /var/log/kern.log: Kernel-related logs.

Important Linux logs

Application Logs:

- To monitor application behavior and performance running on the system, such as errors, warnings, and status updates from databases.
- These logs are saved in:
  - /var/log/apache2/: Apache web server logs.
  - /var/log/mysql/: MySQL database logs.

Important Linux logs

Security Linux Logs:

- To record authentication attempts across events and control potential security threats for example failed login attempts, user activity, and firewall logs.
- These logs are in:
  - /var/log/auth.log: Authentication logs.
  - /var/log/secure: Security-related logs (on Red Hat-based systems).

Important Linux logs

Debug Linux Logs:

- To provide detailed debugging information and make troubleshooting easy for complex issues, such as verbose application or system-level messages.
- These logs are saved in:
  - Location depends on the application being debugged (e.g., /var/log/debug).

Important Linux logs

Custom Linux Logs

- These custom logs are created for specific applications or server scripts, such as third-party applications.
- These logs are stored in:
  - Defined by the application or script: (e.g.,/var/log/custom_app.log).

Important Linux logs

Network Logs:

- Network Linux logs capture details about the network activity and connections, such as packet transfer, connectivity issues, and domain activities.
- These logs are in:
  - /var/log/messages: General network and system messages.
  - /var/log/ufw.log: Firewall (UFW) activity logs.

List of Linux Network Logs

1. **General Network Linux Logs**

- These logs record general network activity and messages.
- They are used to monitor network interfaces and diagnose general connectivity issues.
- These logs are saved in:
    - /var/log/messages (on Red Hat-based systems).
    - /var/log/syslog (on Debian-based systems).

List of Linux Network Logs

2. Firewall Linux Logs

- To capture traffic filtered by the firewall, identify blocked traffic, and debug firewall rules.
- These logs are in:
  - /var/log/ufw.log (for systems using UFW).
  - /var/log/iptables.log (if iptables logging is enabled).

List of Linux Network Logs

3. DHCP Linux Logs

- To track DHCP server and client interactions, analyze IP address assignments, and debug DHCP-related connectivity issues.
- These logs are saved in:
  - /var/log/syslog (on Debian-based systems).
  - /var/log/messages (on Red Hat-based systems).

List of Linux Network Logs

4. DNS Logs

- To log Domain Name System for queries and responses and troubleshoot domain resolution problems.
- These logs are in:
  - /var/log/named.log (for BIND DNS server).
  - Custom locations based on DNS software configuration.

List of Linux Network Logs

5. Network Manager Logs

- To record actions of the Network Manager service, track Wi-Fi connections, and diagnose interface-specific issues.
- The logs are saved in:
    - /var/log/syslog or /var/log/NetworkManager.log.

List of Linux Network Logs

6. WIFI Logs

- To capture the wireless network activities and debug Wi-Fi authentication and connection errors.
- Monitor signal strength and interface.
- These logs are saved in:
  - /var/log/wpa_supplicant.log (for WPA-related logs).

List of Linux Network Logs

7. SSH Logs

- To Record Secure Shell connections and activities, monitor remote login attempts, and identify potential unauthorized access.
- These logs are saved in:
  - /var/log/auth.log (on Debian-based systems).
  - /var/log/secure (on Red Hat-based systems).

Troubleshooting with Linux Logs

Issue	Key Logs to Check	Steps to Troubleshoot
System Issues	/var/log/syslog, /var/log/messages	– Use grep to search for errors or warnings.- Analyze recent entries for anomalies.
Wi-Fi Issues	/var/log/syslog, /var/log/wpa_supplicant.log, /var/log/dmesg	– Search for Wi-Fi-related entries using grep.- Check if the network interface is active using ifconfig or ip a.
Network Connectivity Problems	/var/log/syslog, /var/log/ufw.log, /var/log/iptables.log	– Verify the network interface status with ip link show.- Ping a known address (e.g., ping 8.8.8.8).- Review logs for blocked connections or packet drops.
Application Issues	Application-specific logs (e.g., /var/log/apache2/error.log)	– Search for error entries using grep.- Use log timestamps to correlate errors with recent changes.
Domain Join Issues	/var/log/messages, /var/log/syslog, /var/log/ssd/	– Review domain-related errors using grep.- Check time synchronization between the client and domain controller.

How to View Linux Logs?

1. **Using the cat Command**
   - To display the entire contents of a log file together, use the command:
     ```
     cat /var/log/syslog
     ```
2. **Using the tail command**
   - To display a few lines of log, use the command:
     ```
     tail -n 50 /var/log/syslog
     ```

   -
     ```
     tail -f /var/log/syslog
     ```

How to View Linux Logs?

3. **Using the less command**
   - To allow scrolling through a log file, use command:
     less /var/log/syslog
4. **Using the grep command**
   - To filter logs according to the relevant entries, use command:
     grep “error” /var/log/syslog

How to View Linux Logs?

5. Using journalctl command

- To provide access to logs managed by systemd, use command:
  - journalctl
  - journalctl -u nginx.service
  - journalctl –since “1 hour ago”
  - journalctl -f

How to View Linux Logs?

6. Viewing specific logs

- To view specific logs, use the following command:
- **System Logs:**
  - cat /var/log/syslog
  - cat /var/log/messages
- **Authentication Logs:**
  - cat /var/log/auth.log
  - cat /var/log/secure
- **Kernel Logs:** dmesg | less
- **Application Logs:** cat /var/log/apache2/error.log

journalctl command

- The journalctl command is a powerful utility for managing system logs stored by the systemd journal.
- It enables viewing, filtering, and exporting logs.
- Display all logs: journalctl
- View logs for a specific service: journalctl -u apache2.service
- Show logs from the current boot session: journalctl -b
- The journalctl command simplifies working with logs and provides advanced filtering options.

Check Error Logs in Linux

- Error logs are crucial for identifying and fixing issues in a Linux system.
- Depending on the source of the error, you can check different log files:
  - System Errors: Look in /var/log/syslog or /var/log/messages.
  - grep "error" /var/log/syslog
  - Authentication Errors: Check /var/log/auth.log.
  - Application Errors: Review application-specific logs, such as /var/log/apache2/error.log for Apache.
- Using tools like grep or journalctl with error-related keywords can quickly help locate issues.

View Live Logs in Linux Command Line

- To monitor logs in real-time, you can use the tail command with the -f option:
  - tail -f /var/log/syslog
- This command continuously displays new log entries as they are written.
- For more advanced live monitoring, tools like less +F or journalctl -f can be used.

/var/log/messages

- The /var/log/messages file is one of the most used logs in Linux.
- It contains general system activity messages, including service starts, shutdowns, and errors.
- It’s an essential log for troubleshooting system-wide issues.
- However, some distributions like Ubuntu replace /var/log/messages with /var/log/syslog for similar functionality.

Linux Network Logs

- Network logs provide valuable insights into network-related activities, such as connection attempts, packet filtering, and firewall rules.
- These logs are often found in /var/log/syslog or dedicated files like /var/log/iptables.log.
- For live monitoring, you can use:
  - sudo tail -f /var/log/syslog | grep "network"
- This allows real-time tracking of network events.

Best Practices for Managing Linux Logs

- Use tools like Rsyslog, Graylog, or ELK Stack to collect and analyze logs from multiple systems.
- Configure logrotate to prevent logs from using too much disk space.
- Restrict access with permission files.
- Use tail -f or tools like Logwatch to detect issues proactively.
- Use commands like grep, awk or tools like Logstash to focus on relevant log entries.
- Define log retention policies to meet regulatory requirements.

How to Read Linux Logs

Linux logs are structured to contain the same standard information as event logs generated by other technologies:

- **Timestamp**: date and time of event
- **Hostname**: machine generating the log
- **Service or Application Name**: service or application generating the log
- **Process ID (PID)**: process generating the log
- **Log Level**: severity or importance, like informational (INFO), warning, or error
- **Message Body**: event details

Use the following commands to Read Linux Logs

- **cat:** displays the log file’s complete contents, including message body
- **less**: displays one page of a file and supports searching the file, so you can more easily move between longer files and examine details
- **more**: displays the file contents one screen at a time but limited searchability and navigation
- **tail**: displays last ten entries of a file, often used for checking recent events
- **head**: displays the beginning of the file to quickly scan initial entries
- **grep**: search for patterns or keywords within a file to locate information when investigating or troubleshooting

Use the following commands to Read Linux Logs

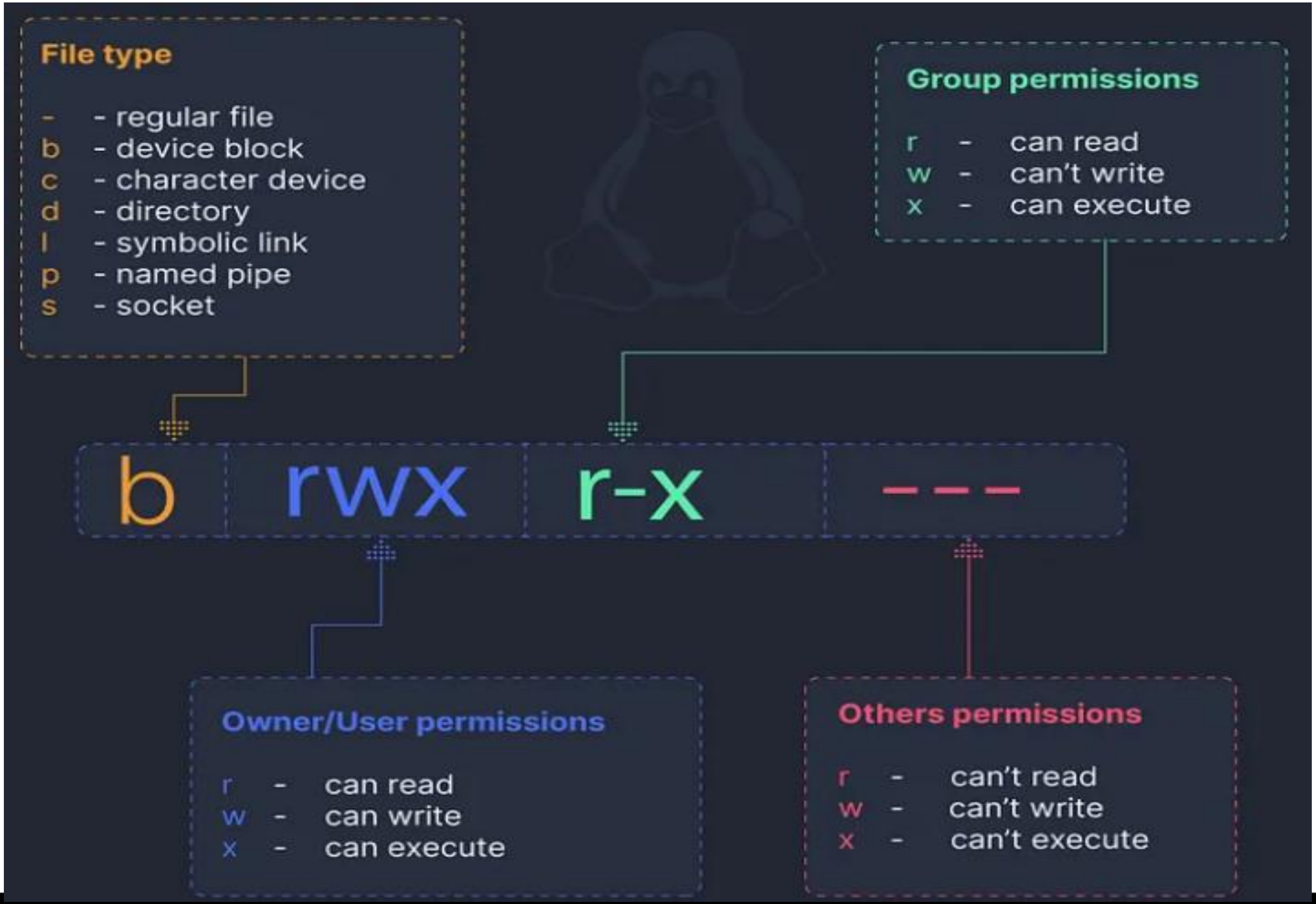
- **awk**: tool to extract information, filter using conditions, or perform operations
- **Sed**: stream editor for processing and modifying log file text, like only printing those containing a specific keyword
- **dmesg**: viewing system logs related to low-level hardware and kernel events, like driver problems
- **journalctl**: displays logs from systemd daemon with the ability to filter by service, date, or other criteria

Linux Centralized Log Analysis Tools

1. SigNoz
2. Splunk
3. Graylog
4. SumoLogic
5. Elasticsearch
6. Datadog
7. Logwatch
8. Logic Monitor
9. Sematext
10. SolarWinds

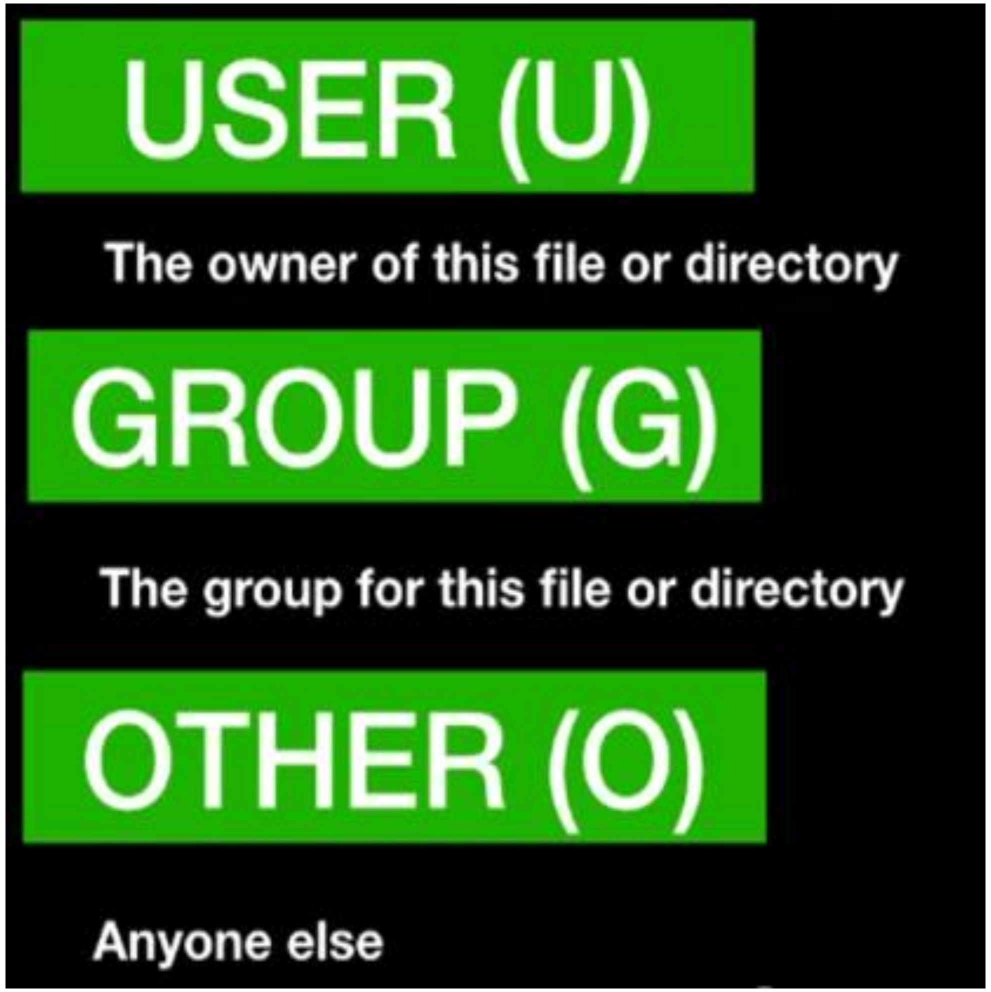
LINUX FILE PERMISSIONS ADMINISTRATION

Basic Linux File Permissions



3 types of users in Linux

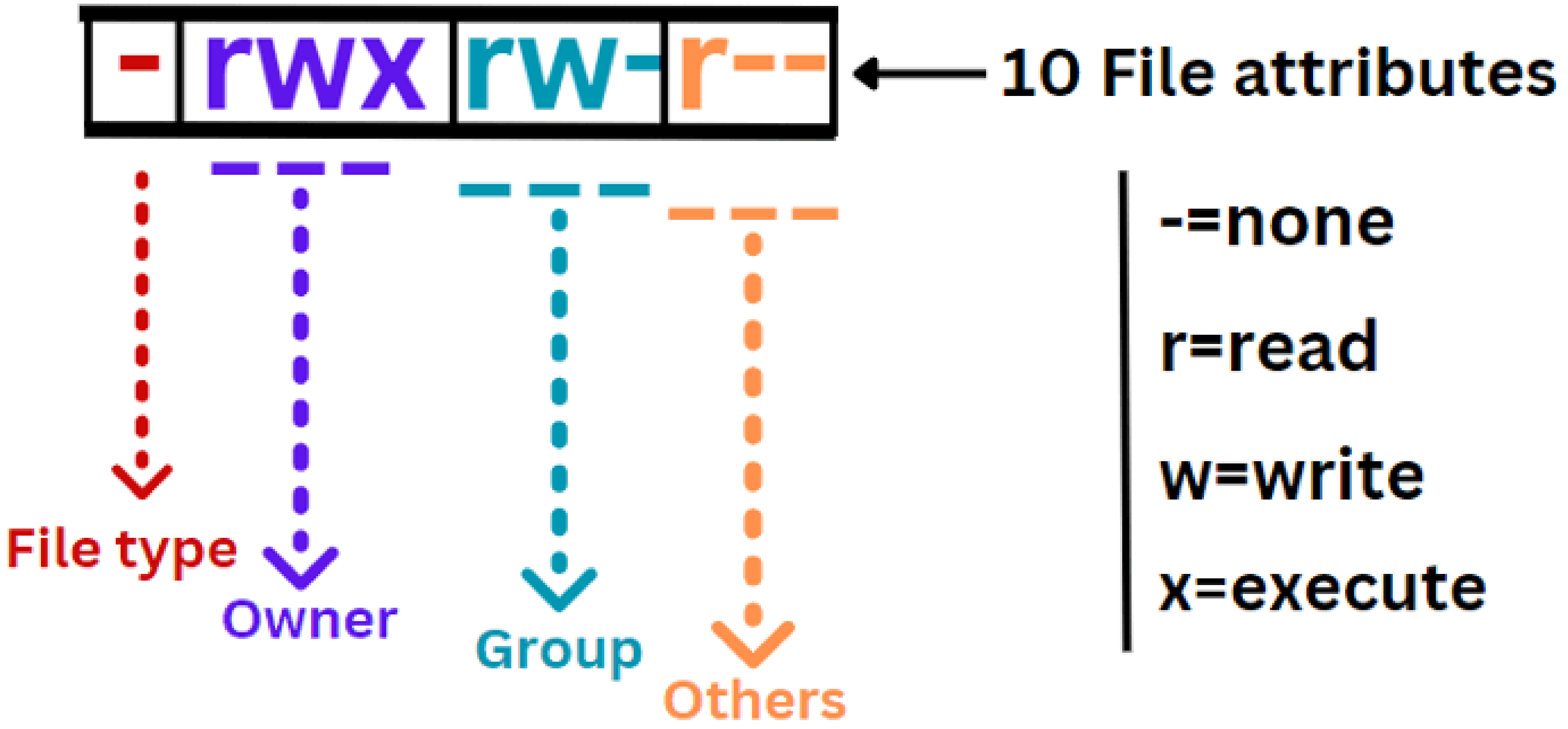
1. **Owner/User (u):**
   - The person who created the file or directory.
   - They have the most control over it.
2. **Group (g):**
   - A group of users who share access.
   - Permissions are often less than those of the owner but more than those of others.
3. **Others (o):**
   - All other users on the system.
   - Permissions for others are typically the most restrictive.


USER (U)
The owner of this file or directory
GROUP (G)
The group for this file or directory
OTHER (O)
Anyone else

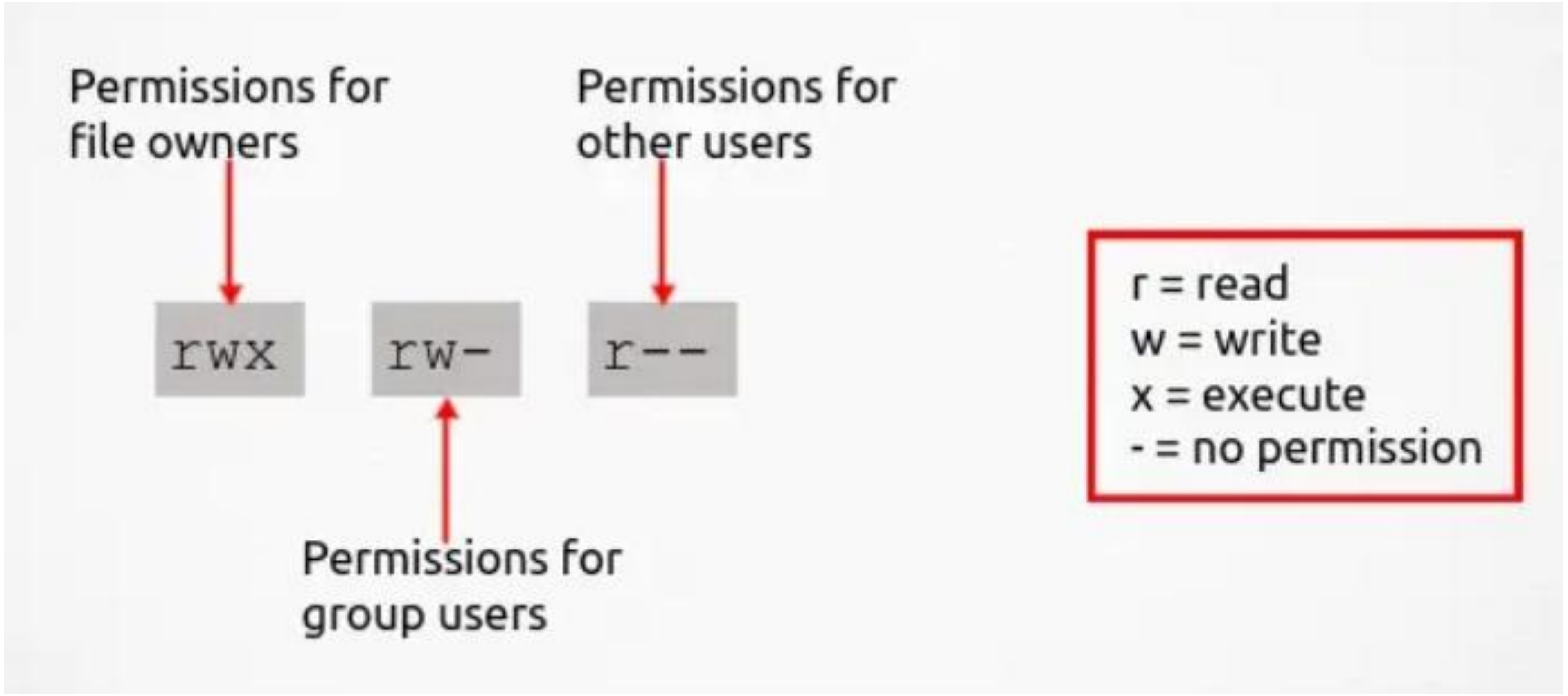
The 1st character represent the Linux File Type

- - indicates a regular file.
- d indicates a directory.
- l indicates a symbolic link.
- b: represent block devices
- c: represent character devices
- p: represent named pipes
- s represent sockets

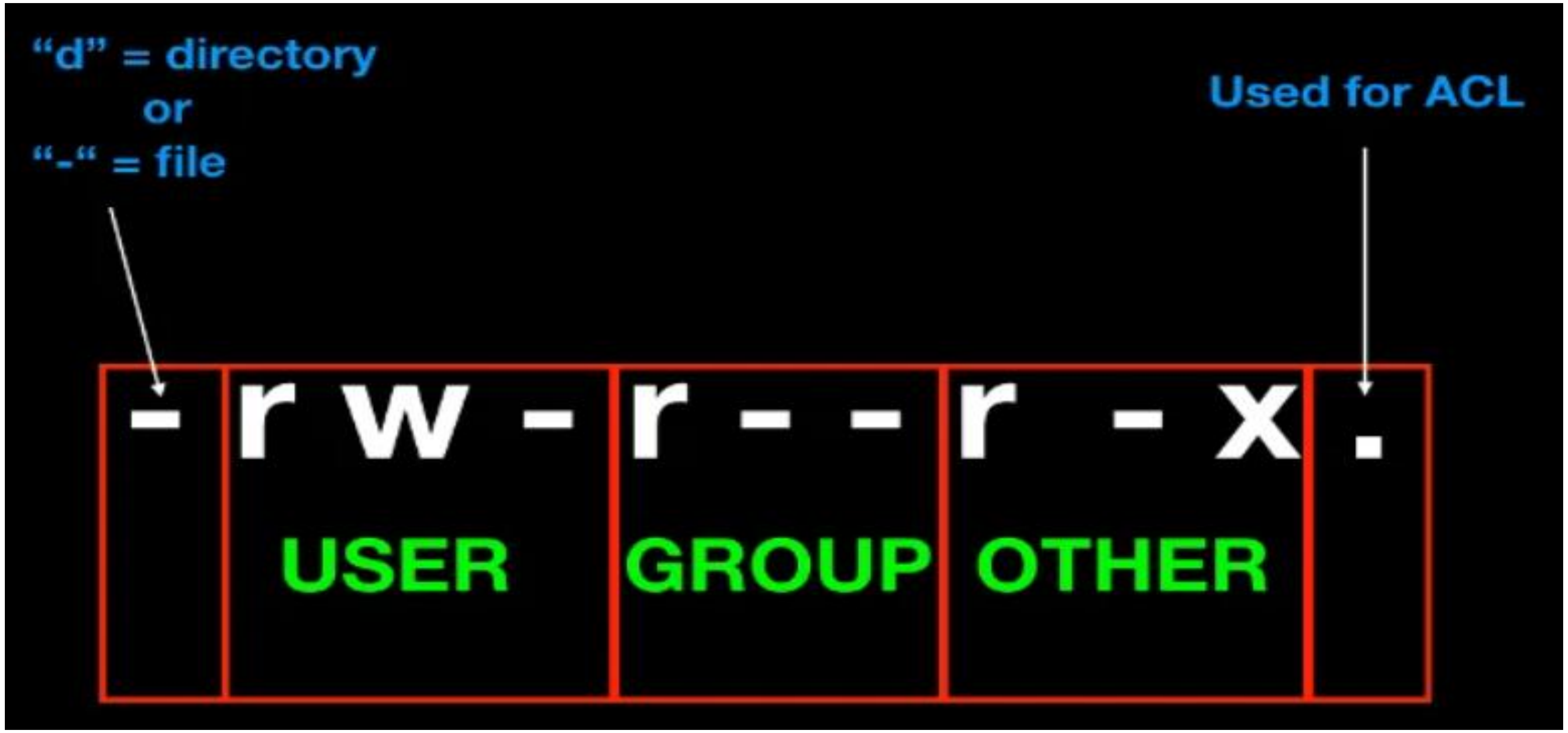
Linux File permission (symbolic)



Linux File permission (symbolic)

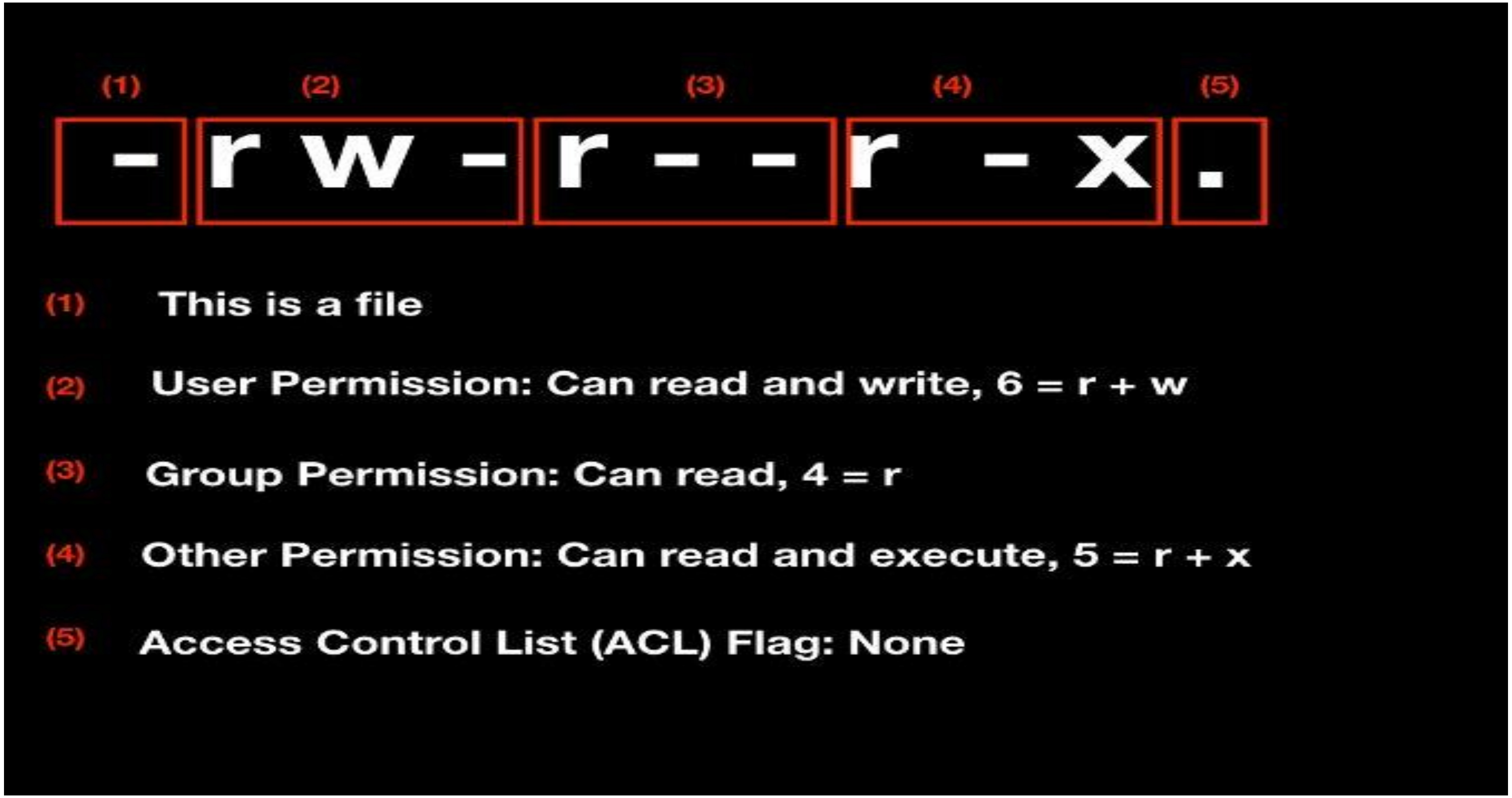

Permissions for file owners
Permissions for other users
rwx
rw-
r--
Permissions for group users
r = read
w = write
x = execute
- = no permission

Linux File permission (symbolic)



The last part is a flag for the **access-control list** (ACL), a list of permissions for an object.

Linux File permission (symbolic)



(1) This is a file

(2) User Permission: Can read and write, 6 = r + w

(3) Group Permission: Can read, 4 = r

(4) Other Permission: Can read and execute, 5 = r + x

(5) Access Control List (ACL) Flag: None

Linux File permission (symbolic)

Permission Type	Symbol
No permission	—
Execute	-x
Write	-w-
Write + Execute	-wx
Read	r-
Read + Execute	r-x
Read + Write	rw-
Read + Write + Execute	rwx

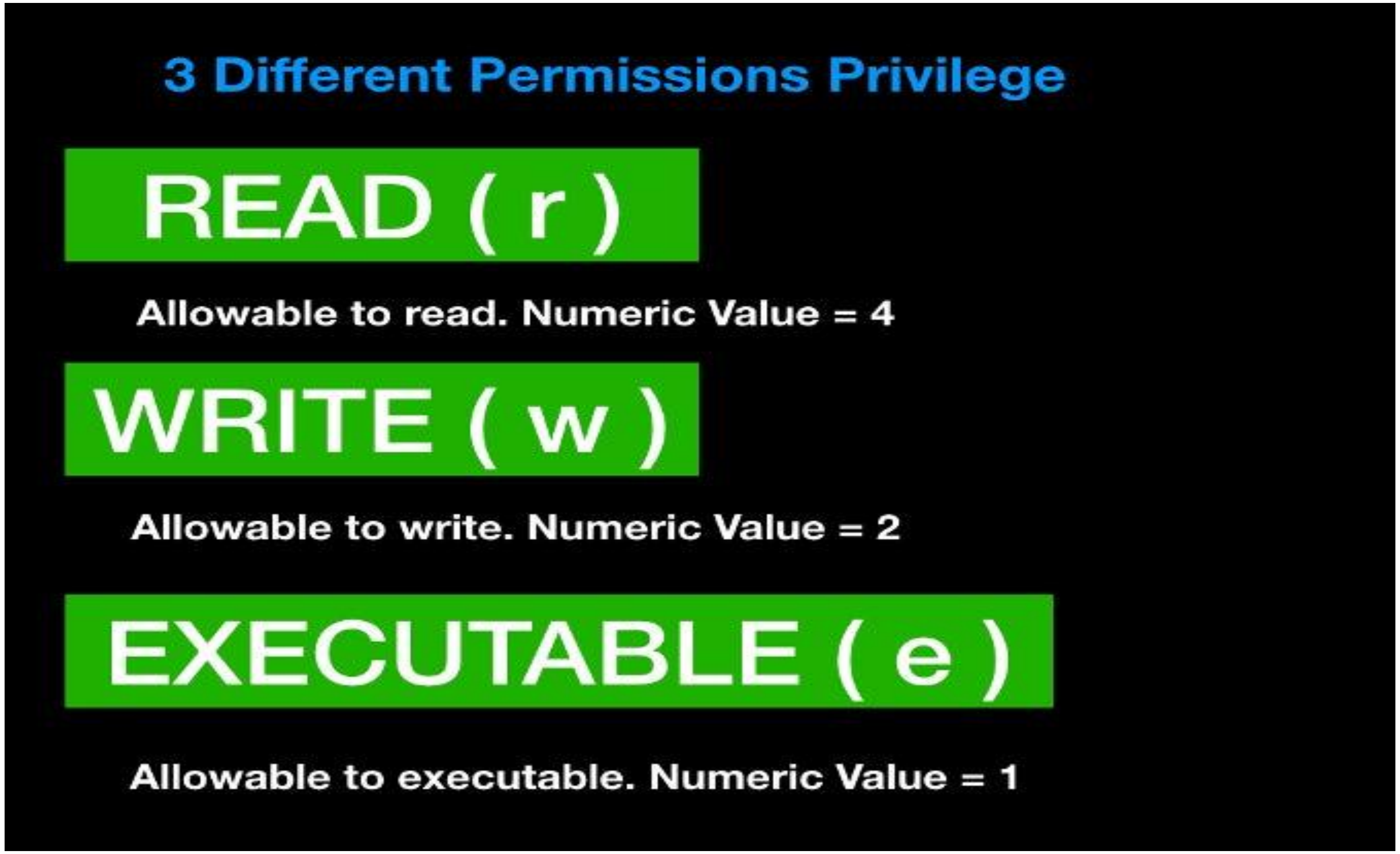
Linux File permission (Numeric)

Permission	Binary value	Octal value
—	000	0
–x	001	1
-w-	010	2
-wx	011	3
r–	100	4
r-x	101	5
rw-	110	6
rwx	111	7

Linux File permission (Numeric)

Permission Type	Number
No permission	0
Execute	1
Write	2
Write + Execute	3 (i.e. 2+1)
Read	4
Read + Execute	5 (i.e. 4+1)
Read + Write	6 (i.e. 4+2)
Read + Write + Execute	7 (i.e. 4+2+1)

Linux File permission (Numeric)



3 Different Permissions Privilege

READ (r)

Allowable to read. Numeric Value = 4

WRITE (w)

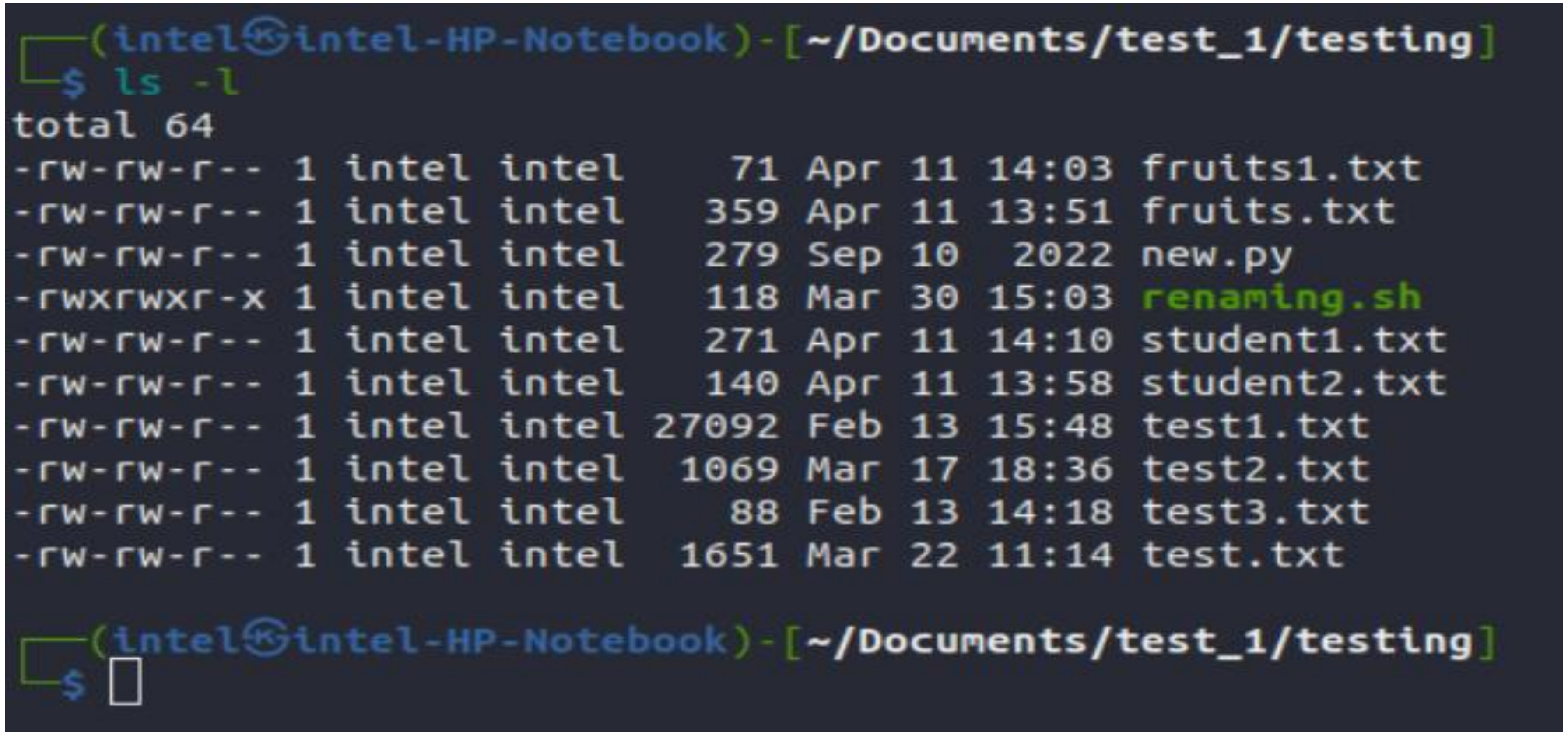
Allowable to write. Numeric Value = 2

EXECUTABLE (e)

Allowable to executable. Numeric Value = 1

How to View File Permissions in Linux

- Use this command: ls –l
- You will see a list of all files and directories in the current location.
- Example:

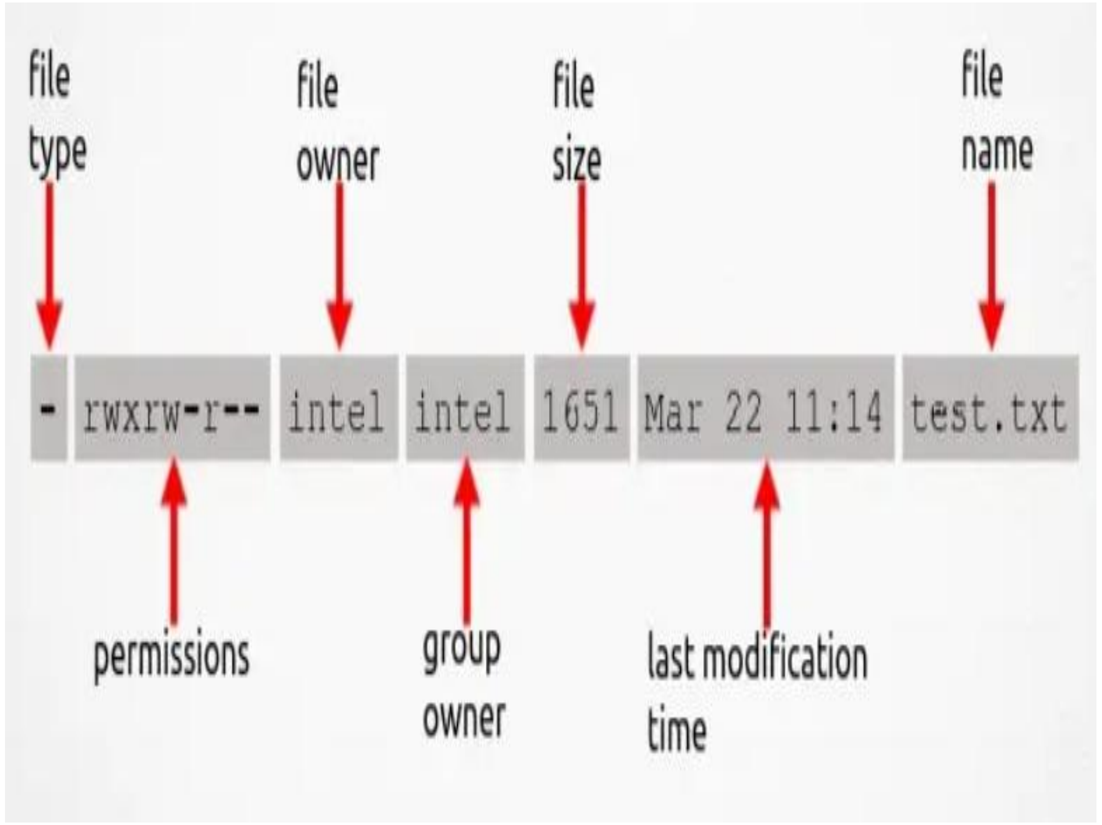


```
┌──(intel㉿intel-HP-Notebook)-[~/Documents/test_1/testing]
└─$ ls -l
total 64
-rw-rw-r-- 1 intel intel    71 Apr 11 14:03 fruits1.txt
-rw-rw-r-- 1 intel intel   359 Apr 11 13:51 fruits.txt
-rw-rw-r-- 1 intel intel   279 Sep 10  2022 new.py
-rwxrwxr-x 1 intel intel   118 Mar 30 15:03 renaming.sh
-rw-rw-r-- 1 intel intel   271 Apr 11 14:10 student1.txt
-rw-rw-r-- 1 intel intel   140 Apr 11 13:58 student2.txt
-rw-rw-r-- 1 intel intel 27092 Feb 13 15:48 test1.txt
-rw-rw-r-- 1 intel intel  1069 Mar 17 18:36 test2.txt
-rw-rw-r-- 1 intel intel    88 Feb 13 14:18 test3.txt
-rw-rw-r-- 1 intel intel  1651 Mar 22 11:14 test.txt

┌──(intel㉿intel-HP-Notebook)-[~/Documents/test_1/testing]
└─$ 
```

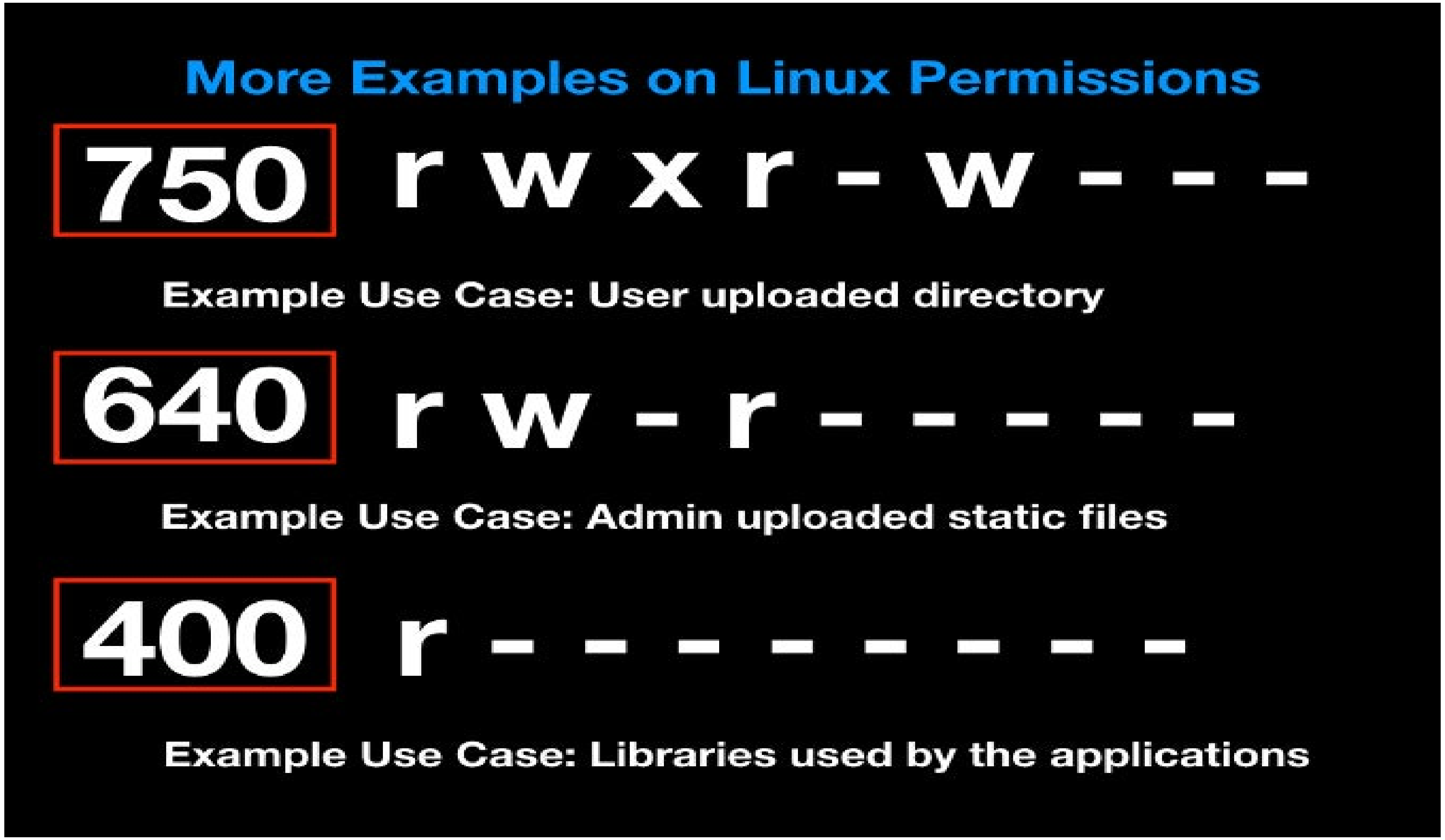
File Permissions in Linux Example

- The first character (-) indicates file type; ‘-‘ means the item is a file and ‘d‘ means a directory.
- The next 9 characters specify the permission set for the entry
- The next digit shows the number of links the file has. By default, the item will have 1.
- The next column shows the name of the file owner.
- The next column shows which group has access to the file.
- The next column shows the file size.
- The next column shows the latest modification time of the file.
- And the final column shows the name of the file/directory.



```
- rwxrw-r-- intel intel 1651 Mar 22 11:14 test.txt
```

File Permissions in Linux Example



More Examples on Linux Permissions

750 r w x r - w - - -

Example Use Case: User uploaded directory

640 r w - r - - - - -

Example Use Case: Admin uploaded static files

400 r - - - - - - - -

Example Use Case: Libraries used by the applications

Special file Permissions in Linux

- Special file permissions are used to grant or restrict access to files & directories.
- These permissions go beyond the basic read, write, and execute permissions that can be applied to files.
- There are 3 types of special file permissions:

1. Setuid: With the enablement of this permission on an executable file, it allows the user who executes the file to assume the permission of the owner of the file.
2. Setgid: When setgid is enabled on a directory, it forces all files and subdirectories created within that directory to inherit the group ownership of the parent directory.
3. Sticky bit: It restricts the deletion of files within that directory to only the owner of the file or the root user. This is used to ensure that important files are not accidentally deleted by other users.

Special File Permission: Set User ID (SUID)

- The SUID permission is a special type of permission that allows a user to execute a file with the permissions of the file owner rather than the user who launched it.
- This is particularly useful for executables that need to perform tasks that require higher privileges.

Special File Permission: Set User ID (SUID)

For example, consider the `passwd` command, which is used to change a user's password. This command needs to write to the `/etc/shadow` file, which is owned by root and not writable by regular users. By setting the SUID bit on the `passwd` command, users can change their passwords while the `passwd` command can update the `/etc/shadow` file.

```
-rwsr-xr-x 1 root root 68208 Feb 15  2021 /usr/bin/passwd
```

In the above example, the `s` in the user's execute field indicates that the SUID bit is set. When a user executes the `passwd` command, it runs with the permissions of the file's owner (in this case, `root`).

Special File Permission: Set Group ID (SGID)

- The SGID permission is like the SUID permission, but instead of the user, it affects the group.
- When the SGID bit is set on a directory, any files created within that directory will inherit the group ownership of the directory, not the primary group of the user who created the file.

Special File Permission: Set Group ID (SGID)

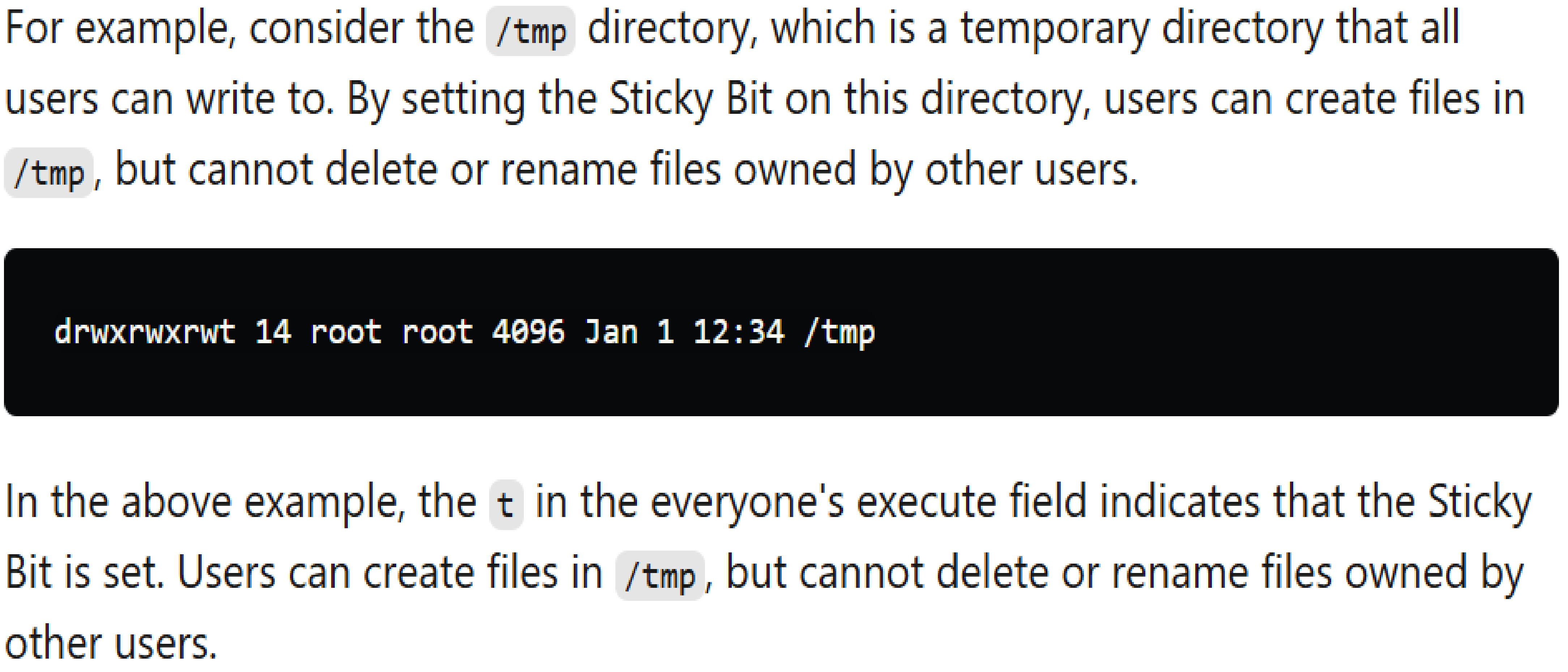
For example, consider a directory named `/data`, which is owned by the `staff` group. By setting the SGID bit on this directory, any files created within `/data` will be owned by the `staff` group.

```
drwxr-sr-x 2 root staff 4096 Jan 1 12:34 /data
```

In the above example, the `s` in the group's execute field indicates that the SGID bit is set. Any files created within `/data` will be owned by the `staff` group.

Special File Permission: Sticky Bit

- The Sticky Bit is a permission that is set on a directory and prevents a user from deleting or renaming files in that directory unless they are the owner of the file or the directory.
- This is particularly useful for directories like /tmp, which are world-writable but could cause issues if a user could delete or rename files they do not own.



Special File Permission: Sticky Bit

For example, consider the `/tmp` directory, which is a temporary directory that all users can write to. By setting the Sticky Bit on this directory, users can create files in `/tmp`, but cannot delete or rename files owned by other users.

```
drwxrwxrwt 14 root root 4096 Jan 1 12:34 /tmp
```

In the above example, the `t` in the everyone's execute field indicates that the Sticky Bit is set. Users can create files in `/tmp`, but cannot delete or rename files owned by other users.

Kind Reminders

- Class slides & materials are just startup for you to explore
- I count on your hard work
- I count on your deep dive
- I count on your commitment to learn
- I advice you to explore websites & LinkedIn for more to enrich your learning
- Be proactive rather than reactive
- Practice what you learn
- Stay Tuned. More to come.

DEEP DIVE

ALADDIN ALRADHI
AALRADHI@CONESTOGAC.ON.CA