Hands On LINUX 🙌🐧

Hands On LINUX 🙌🐧

Introduction

What is Linux?

Linux is the kernel of various distributions (like Debian, Ubuntu, Red Hat) combined with GNU software. It serves as a free and open-source alternative to Unix 2.

Linux Distributions

A Linux distribution is a complete package of the Linux operating system. Major distributions include:

- **Ubuntu**: Popular for desktop use and comes in various editions (desktop, server, core) 3.
- **Debian**: A foundational distribution for many others, including Ubuntu.
- **Kali Linux**: Primarily used in cybersecurity and penetration testing.
- **Red Hat**: A business-oriented distribution that requires payment for support

Basics

File Links

Hard Links

A hard link allows multiple filenames to point to the same **inode** (index node).

❗ An inode is **a data structure that keeps track of all the files and directories within a Linux or UNIX-based filesystem**. So, every file and directory in a filesystem is allocated an inode, which is identified by an integer known as “inode number”. These unique identifiers store metadata about each file and directory.

❔ Inodes store metadata such as:
File type, File size, Owner ID, Group ID, Read write and execute, permissions, Last access time, Last change time, Last modification time

Use `ln [target] [link]` to create a hard link.

❔ For example, `ln /home/steve/pictures/house.jpg /home/mary/pictures/house.jpg`.

Soft Links (Symbolic Links)

A soft link points to a file path rather than an inode. Create a soft link using `ln -s [target] [link]`.

For example, `ln -s /home/steve/pictures/house.jpg /home/mary/pictures/house.jpg`.

Commands

Info about file and fs: `stat [path]`
Info about file inode: `ls -i [path]`
Info about directory inode: `ls -idl [path]`
Disk info (disk free) with inode: `df -i [disk]`

Basic commands

Packet management

Advanced Package Tool: `apt`
Install packages from configured resources: `apt update`
Install newer version of installed packages: `apt upgrade`
Install another package: `apt install`
Remove a package: `apt remove`

Directory

- Print working directory: `pwd`
- Change directory: `cd`
- List directory contents: `ls -a (all) -l (detail) -h (human readable)`
- Create directory: `mkdir -p (with parent or child)`
- Remove directory: `rmdir -p (non-empty directory) -r (recursive)`

File

- Type of file: `file`
- Create a file: `touch`
- Remove a file: `rm -i (interactive)`
- View first lines of file content: `head -n 5 filename (default 10)`
- View end lines of file content: `tail -n 5 filename (default 10)`
- Prints entire content and concatenates multiple files: `cat`

Operators

Redirection (>, >>):

```
echo "Hello, World!" > file.txt  # Writes "Hello, World!" to file.txt
echo "Another line" >> file.txt # Appends "Another line" to file.txt
cat > filename  # creates a file using cat
```

Piping (|):

```
ls -l | grep "txt" # Lists files with "txt" in their names
```

Background Execution (&):

```
./script.sh & # Runs script.sh in the background
```

Logical AND (&&):

```
mkdir newdir && cd newdir # Creates newdir and enters it only if mkdir is
successful
```

Logical OR (||):

```
mkdir newdir || echo "Failed to create directory" # Echoes a message if
mkdir fails
```

Command Substitution ($()):

```
echo "Today is $(date)" # Inserts the current date into the echo command
```

Environment Variables ($VAR):

```
export MY_VAR="Hello"
echo $MY_VAR # Outputs: Hello
```

Conditional Expressions ([[ ]]):

```
if [[ -f "file.txt" ]]; then echo "File exists"; fi # Checks if file.txt
exists
```

Loops (for, while):

```
for i in {1..5}; do echo "Number $i"; done # Loops from 1 to 5 and prints
each number

while read line; do echo "$line"; done < file.txt # Reads and prints each
line from file.txt
```

Functions:

```
function greet() {
    echo "Hello, $1!"
}
greet "Alice"  # Outputs: Hello, Alice!
```

Copy

Copy files: `cp -r (recursive)`

Move

Move a file: `mv file.txt /path/to/directory/file3.txt`

Processes

Show processes: `ps -A (All)`
Real-time processes info: `top` or `htop`

Disk

Fs disk info (disk free): `df`
Format disk or fixed disk: `fdisk`

Network

Ip config: ifconfig or ip

File Permissions

User info is stored in `/etc/passwd` and `/etc/group`.

Understanding Permissions

Each file has an owner and a group. Permissions are displayed using `ls -l` and consist of **read (r), write (w), and execute (x) permissions**.

Changing Permissions

Use the `chmod` command to change permissions. For example, `chmod u+w house.jpg` adds write permission for the user.

Setting Permissions

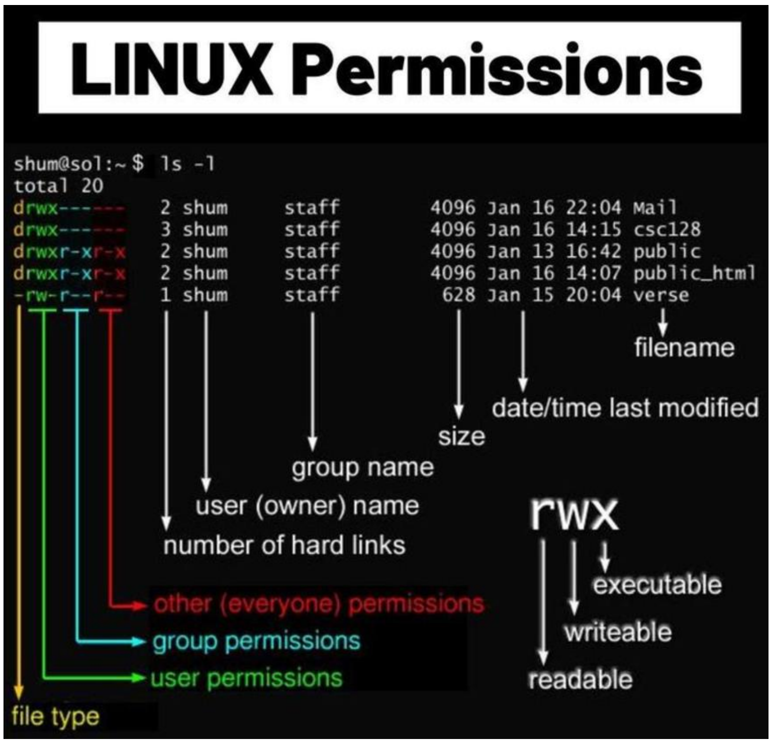
Permissions can be set exactly using `chmod u=rwx,g=rw,o=r house.jpg`.

❗ Permissions are evaluated in order: owner, group, and others. For example, if a user is the owner, only the owner's permissions apply

File Permissions

Permissions in Linux can be represented numerically. For example, rwx (read, write, execute) corresponds to 7, r-x (read, execute) corresponds to 5, and so on. Full permissions can be set using the command `chmod`.

7	rwx	111
6	rw-	110
5	r-x	101
4	r--	100
3	-wx	011
2	-w-	010
1	--x	001
0	---	000



```
shum@sol:~$ ls -l
total 20
drwx------  2 shum    staff       4096 Jan 16 22:04 Mail
drwx------  3 shum    staff       4096 Jan 16 14:15 csc128
drwxr-xr-x  2 shum    staff       4096 Jan 13 16:42 public
drwxr-xr-x  2 shum    staff       4096 Jan 16 14:07 public_html
-rw-r--r--  1 shum    staff        628 Jan 15 20:04 verse
```

File Ownership

Changing Ownership

Use `chown [user] [file]` to change the owner of a file.
For example, `sudo chown mary house.jpg` changes the owner to Mary.

File Types

Identifying File Types

The first character in the ls -l output indicates the file type (e.g., d for directory, - for regular file)

Pagers

Less

More feature-rich. To open a file, type less filename.
Use arrow keys to navigate, / to search, and q to quit.

Search in Less

Type /search_term to find text.
Use n for the next instance and N for the previous. To ignore case, use -i.

More

Simpler than less. Use space to scroll down and q to quit

Manual

The man command is crucial for **accessing manuals for other commands**, providing information on usage and options.

Example: `man man` provides the manual for the man command itself

Short description on a specific command: `whatis [command]`
Command that contains a specific string: `apropos [command]`

Nano Text Editor

Simpler text editor, use `nano filename`.

Vim Text Editor

Opening Vim

Use vim filename to start. **It has different modes, primarily insert and command mode**.

Basic Commands

- **Insert Mode**: Press `i` to enter insert mode and start typing.
- **Search**: Use `/search_term` in command mode. For case-insensitive search, use `\c`.
- **Line Navigation**: Type `:line_number` to jump to a specific line.
- **Copy-Cut/Paste**: Use `yy` to copy a line, `p` to paste, `dd` to cut line.
- **Saving and Quitting**: Use `:wq` to save and exit, or `:q!` to quit without saving.

Searching Text with Grep

- **Basic Grep Usage**: `grep 'pattern' filename` searches for lines matching the pattern.
- **Case Insensitivity**: Use -i to ignore case.
- **Recursive Search**: Use -r to search through directories.
- **Inverting Matches**: Use -v to find lines that do not match the pattern.

Regular Expressions in Grep

- **Basic Operators**: Use `^` for start of line and `$` for end of line.
- **Matching Specific Patterns**: Use `.` to match any character and `*` to match zero or more occurrences.
- **Character Classes**: Use `[abc]` to match any character in the set.
- **Grouping and Repetition**: Use `parentheses` for grouping and `+` for one or more occurrences.

Advanced Regular Expressions

- **Extended Regular Expressions**: Use -E with grep to enable extended regex features, **avoiding backslashes**.
- **Negating Sets**: Use `[^abc]` to match any character not in the set.

File System Structure

1. **/**: **The root directory**, the top-level directory of the file system.
2. **/bin**: Contains **essential binary executables** for basic commands.
3. **/boot**: Holds **bootloader files** and the kernel used to start the system.
4. **/dev**: Includes **device files that represent hardware components**.
5. **/etc**: Contains **system configuration** files and scripts.
6. **/home**: Stores **personal directories for users**.
7. **/lib**: Holds **shared library** files used by essential system programs.
8. **/media**: **Temporary mount** points for removable media like **CDs and USB** drives.
9. **/mnt**: **Temporary mount** points for **file systems**.
10. **/opt**: **Optional software packages** and add-on application files.
11. **/proc**: **Virtual filesystem providing process** and kernel information.
12. **/root**: **Home directory for the root user**.
13. **/run**: Stores data related to system **run-time processes**.
14. **/sbin**: Contains **system binaries** essential for booting and system repair.
15. **/srv**: Data for services provided by the system (e.g., **web servers**).
16. **/sys**: Virtual filesystem providing system information and **hardware access**.
17. **/tmp**: **Temporary files** created by system and users.
18. **/usr**: **User binaries**, documentation, libraries, and source code.
19. **/var**: **Variable files** such as logs, databases, and caches that are frequently updated.

User Management in Linux

Creating User Accounts

- The command to add a new user is `useradd`.
    - For example, `useradd John` creates a user named John, **along with a corresponding group and home directory at /home/john.**

Setting Passwords and Deleting Accounts

- To set a password for a user, use the `passwd` command.
- To delete a user, `userdel John` removes the account, **but the home directory can be retained or deleted with additional options.**

Modifying User Accounts

- The `usermod` command is used to change user details, such as home directory or username.
    - For example, to change John's home directory, use `usermod -d /new/home/directory John`.

Managing Groups

- Users can belong to multiple groups.
- To add a user to a group, use `gpasswd -a John developers`.

Resource Limits

- Resource limits can be set for users to prevent any single user from monopolizing system resources. This is configured in the `/etc/security/limits.conf` file.

Privileges and Sudo Access

1. Users can gain administrative privileges using `sudo`, which allows them to execute commands as the root user. To grant a user sudo access, add them to the **wheel group**.
2. The `/etc/sudoers` file defines specific permissions for users, and it should be edited using the `visudo` command to prevent syntax errors.

Access to the Root Account

- Users can switch to the root account using `su -` if they know the root password. If the root account is locked, they can still use sudo if they have the necessary permissions

Development Tools and Practices

HashiCorp Vault

Used to manage secrets and sensitive access in an infrastructure.

Lynis and OpenSCAP

Lynis and OpenSCAP are recommended audit tools for verifying server security.

Traefik and HAProxy

Traefik and HAProxy are popular choices to expose and manage web services.

Terraform and Ansible

Terraform and Ansible are predominant tools for provisioning and managing configurations on VMs.

Shell

2 main shells:

Bash

The .bashrc file is essential for customizing the Bash environment, including the definition of aliases and environment variables.

FZF and bash-it can be useful tools.
*Source: https://youtu.be/tB-AgxzBmH8

WSL

Feature	**WSL1**	**WSL2**
System Call Handling	Translates Linux system calls to Windows system calls	Full Linux kernel in a lightweight VM
Performance	Limited due to translation layer	Improved, full system call compatibility
Kernel	Windows NT kernel	Full Linux kernel
Resource Utilization	Lower efficiency due to compatibility layer	Higher efficiency with direct kernel access
Compatibility	Limited (no full kernel support)	Full compatibility with Linux applications

Summary of WSL

WSL allows you to run a Linux environment directly on Windows without the overhead of a traditional virtual machine.

WSL1

- **Mechanism**: Uses a compatibility layer to translate Linux system calls into Windows system calls.
- **Performance**: Limited compared to WSL2 due to the translation layer.
- **Kernel**: No full Linux kernel, uses Windows NT kernel.

WSL2

- **Mechanism**: Runs a full Linux kernel inside a lightweight virtual machine.
- **Performance**: Improved performance and full system call compatibility with Linux.
- **Kernel**: Full Linux kernel, providing better compatibility and performance.

Basic Setup

Install WSL | Microsoft Learn

Sudo

Administration privileges can be configured from the `/etc/sudoers` file:

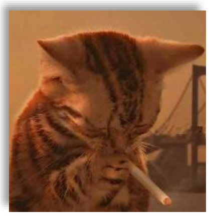
Term	Description
Host Alias	Allows defining aliases for groups of hosts.
User Alias	Allows creating aliases for groups of users.
Cmnd Alias	Defines aliases for groups of commands.
Defaults	Specifies default settings for sudo.
User Privilege Specification	Defines privileges for individual users or groups.
Members of Group	Grants privileges to members of specific groups.
NOPASSWD	Allows specific users to execute certain commands without a password.
Runas Alias	Allows defining aliases for users under which commands can be executed.

Made by :
Matthias Brat

Socials:

github.com/matthiasbrat

stackoverflow.com/users/17921879/matthias-b



matthiasbrat.dev