LINUX For DevOps Engineers

Linux is an open-source operating system kernel that serves as the foundation for various Linux-based operating systems (distributions).
Linux is known for its stability, security, and flexibility. It is based on the Unix operating system and follows the Unix philosophy of small, modular, and reusable components. Linux is highly customizable and can be tailored to suit various needs, ranging from desktop computers and servers to embedded systems and supercomputers.

Linux distributions, such as Ubuntu, Fedora, Debian, CentOS, and many others, take the Linux kernel and combine it with additional software packages, utilities, and graphical interfaces to create complete operating systems that are ready to be used by end-users.

Linux offers a command-line interface (CLI) that allows users to interact with the system through text commands, as well as graphical user interfaces (GUIs) that provide a more user-friendly experience. It supports a wide range of applications, software development tools, and server services, making it suitable for various purposes, including general computing, web servers, cloud infrastructure, networking, and more.

Linux Folder Structure

The folder structure in Ubuntu Linux follows the Filesystem Hierarchy Standard (FHS), which is a standard for organizing the files and directories on a Unix-like operating system. Here is an overview of the main directories you will typically find in Ubuntu:

- **/bin**: Contains essential command-line executable files (binaries) that are available to all users.

- **/boot**: Contains files related to the boot process, including the Linux kernel, initial ramdisk (initrd), and bootloader configuration.

- **/dev**: Contains device files that represent and allow access to various hardware devices on the system.

- **/etc**: Contains system-wide configuration files for various applications and services.

- **/home**: The home directories for individual users. Each user typically has a subdirectory here to store their personal files and settings.

- **/lib** and **/lib64**: These directories contain shared libraries needed by the system and applications. The "lib64" directory is present on 64-bit systems.

- **/media**: Mount point for removable media devices such as USB drives or optical discs.

- **/mnt**: A general-purpose mount point for temporarily mounting filesystems.

- **/opt**: Contains optional software packages installed on the system. Applications installed here are often self-contained in their own directories.

- **/proc**: A virtual filesystem that provides information about processes and system status. It is used by many system utilities to obtain runtime information.

- **/root**: The home directory for the root user, the administrative superuser.

- **/run**: A temporary filesystem that contains runtime data for various system services. It is cleared on each reboot.

- **/sbin**: Contains system binaries (executable files) that are primarily used by the root user for system administration tasks.

- **/srv**: Contains data for services provided by the system.

- **/sys**: A virtual filesystem that exposes kernel-related information and configuration.

- **/tmp**: A directory for temporary files created by applications and users. Its contents are typically cleared on each reboot.

- **/usr**: Contains user-related programs, libraries, documentation, and shared resources. It has subdirectories such as /usr/bin for user binaries, /usr/lib for libraries, and /usr/share for shared data.

- **/var**: Contains variable data that changes during the system's operation, such as logs, databases, and spool files.

This is a high-level overview of the Ubuntu Linux folder structure. Each directory serves a specific purpose in organizing the system's files and resources.

Important Linux Commands for DevOps Engineers

This repository provides a list of important Linux commands that are frequently used by DevOps engineers. Each command is explained with a brief description and examples of usage.

1. `ls`

```
- Description: Lists files and directories in the current directory.
- Usage:
  - `ls`: Lists files and directories in the current directory.
  - `ls -l`: Lists files and directories in long format.
  - `ls -a`: Lists all files and directories, including hidden ones.

### 2. `cd`

- Description: Changes the current directory.
- Usage:
  - `cd /path/to/directory`: Changes the current directory to the specified
path.
  - `cd ..`: Moves up to the parent directory.
  - `cd ~`: Moves to the home directory.

### 3. `mkdir`

- Description: Creates a new directory.
- Usage:
  - `mkdir directory_name`: Creates a new directory with the specified name.
  - `mkdir -p path/to/directory`: Creates nested directories if they don't
exist.

### 4. `rm`

- Description: Removes files and directories.
- Usage:
  - `rm file_name`: Removes the specified file.
  - `rm -r directory_name`: Removes the specified directory and its contents
recursively.
  - `rm -f file_name`: Forces removal of the specified file without prompting.

### 5. `cp`

- Description: Copies files and directories.
- Usage:
  - `cp source_file destination_file`: Copies the source file to the
destination.
  - `cp -r source_directory destination_directory`: Copies the source
directory to the destination recursively.

### 6. `mv`

- Description: Moves or renames files and directories.
- Usage:
  - `mv source_file destination_file`: Moves the source file to the
destination or renames it.
```

  - `mv source_directory destination_directory`: Moves the source directory to the destination or renames it.

7. `grep`

- Description: Searches for a specific pattern in files or output.
- Usage:
  - `grep pattern file_name`: Searches for the specified pattern in the given file.
  - `grep -r pattern directory`: Searches for the pattern recursively in the specified directory.
  - `command | grep pattern`: Filters the output of a command and searches for the pattern.

8. `ps`

- Description: Lists running processes.
- Usage:
  - `ps`: Lists the running processes for the current user.
  - `ps -ef`: Lists all running processes.
  - `ps -eaf`: Lists all running processes with full details.

9. `top`

- Description: Displays real-time system information and the list of processes.
- Usage:
  - `top`: Displays real-time system information, CPU usage, memory usage, and the list of processes.
  - Press `q` to exit the `top` command.

10. `tail`

- Description: Outputs the last part of a file.
- Usage:
  - `tail file_name`: Displays the last 10 lines of the specified file.
  - `tail -n N file_name`: Displays the last N lines of the specified file.
  - `tail -f file_name`: Continuously outputs new lines appended to the file.

Here are the top 50 networking commands in Linux:

1. `ifconfig`: Displays or configures network interfaces.
   Example: `ifconfig eth0`

2. `ip`: Configures and displays network interfaces, routing tables, and more.
   Example: `ip address show`

3. `ping`: Sends ICMP echo requests to a specified network host.

```
   Example: `ping google.com`

4. `traceroute`: Displays the route packets take to reach a destination host.
   Example: `traceroute google.com`

5. `nslookup`: Queries DNS servers for DNS-related information.
   Example: `nslookup google.com`

6. `dig`: DNS lookup utility for querying DNS servers.
   Example: `dig google.com`

7. `host`: Performs DNS lookups.
   Example: `host google.com`

8. `netstat`: Displays network connections, routing tables, and network
statistics.
   Example: `netstat -tun`

9. `ss`: Provides detailed socket statistics.
   Example: `ss -tun`

10. `route`: Configures and displays routing table information.
    Example: `route -n`

11. `arp`: Manipulates or displays the ARP cache.
    Example: `arp -a`

12. `iptables`: Manages firewall rules.
    Example: `iptables -L`

13. `tcpdump`: Captures network traffic.
    Example: `tcpdump -i eth0`

14. `ifup`: Brings a network interface up.
    Example: `ifup eth0`

15. `ifdown`: Brings a network interface down.
    Example: `ifdown eth0`

16. `ethtool`: Displays or changes Ethernet device settings.
    Example: `ethtool eth0`

17. `hostname`: Displays or sets the system's hostname.
    Example: `hostname`

18. `ssh`: Connects to a remote server using the SSH protocol.
    Example: `ssh user@hostname`
```

19. `scp`: Copies files between hosts using SSH.
    Example: `scp file.txt user@hostname:/path/to/destination`

20. `rsync`: Syncs files and directories between different locations.
    Example: `rsync -avz source/ user@hostname:/path/to/destination`

21. `nc`: Reads and writes data across network connections.
    Example: `nc -l 8080`

22. `wget`: Downloads files from the web.
    Example: `wget http://example.com/file.txt`

23. `curl`: Transfers data to or from a server.
    Example: `curl http://example.com`

24. `nmap`: Scans ports and discovers network services.
    Example: `nmap -p 1-1000 hostname`

25. `telnet`: Establishes a telnet connection to a remote host.
    Example: `telnet hostname`

26. `ifstat`: Displays network interface statistics.
    Example: `ifstat`

27. `mtr`: Combines ping and traceroute functionality.
    Example: `mtr google.com`

28. `route add`: Adds a new route to the routing table.
    Example: `route add -net 192.168.0.0/24 gw 192.168.1.1`

29. `route delete`: Deletes a route from the routing table.
    Example: `route delete default gw 192.168.1.1`

30. `ifconfig up`: Enables a network interface.
    Example: `ifconfig eth0 up`

31. `ifconfig down`: Disables a network interface.
    Example: `ifconfig eth0 down`

32. `ip link`: Manages network interfaces.
    Example: `ip link show`

33. `ip route`: Manages routing tables.
    Example: `ip route show`

34. `ip neigh`: Manages ARP cache.
    Example: `ip neigh show`

35. `ip addr`: Manages IP addresses and interfaces.
    Example: `ip addr show`

36. `ip link set`: Modifies network interface properties.
    Example: `ip link set eth0 mtu 1500`

37. `ip route add`: Adds a new route to the routing table.
    Example: `ip route add 192.168.0.0/24 via 192.168.1.1 dev eth0`

38. `ip route delete`: Deletes a route from the routing table.
    Example: `ip route delete 192.168.0.0/24`

39. `ip addr add`: Adds an IP address to an interface.
    Example: `ip addr add 192.168.0.1/24 dev eth0`

40. `ip addr delete`: Deletes an IP address from an interface.
    Example: `ip addr delete 192.168.0.1/24 dev eth0`

41. `ip tunnel add`: Creates a tunnel interface.
    Example: `ip tunnel add mytunnel mode gre remote 203.0.113.1 local 198.51.100.1`

42. `ip tunnel delete`: Deletes a tunnel interface.
    Example: `ip tunnel delete mytunnel`

43. `ip link set promisc on`: Puts a network interface into promiscuous mode.
    Example: `ip link set eth0 promisc on`

44. `ip link set promisc off`: Disables promiscuous mode on a network interface.
    Example: `ip link set eth0 promisc off`

45. `ip link set mtu`: Sets the Maximum Transmission Unit (MTU) of a network interface.
    Example: `ip link set eth0 mtu 1500`

46. `iptables -A INPUT`: Appends a rule to the INPUT chain of the firewall.
    Example: `iptables -A INPUT -s 192.168.0.0/24 -j ACCEPT`

47. `iptables -D INPUT`: Deletes a rule from the INPUT chain of the firewall.
    Example: `iptables -D INPUT -s 192.168.0.0/24 -j ACCEPT`

48. `iptables -P`: Sets the default policy for a chain in the firewall.
    Example: `iptables -P INPUT DROP`

49. `iptables -F`: Flushes all rules from a chain in the firewall.
    Example: `iptables -F INPUT`

50. `iptables-save`: Saves the current firewall rules to a file.
    Example: `iptables-save > firewall.rules`

These commands cover a wide range of networking tasks and can be helpful for managing and troubleshooting network connections in Linux.

These commands provide essential functionality for network configuration, troubleshooting, and analysis in Linux. You can refer to the respective man pages or online documentation for each command to learn more about their options and usage.

In Linux, you can change permissions and ownership of files and directories using the `chmod`, `chown`, and `chgrp` commands. Here's how you can use these commands:

Changing Permissions

`chmod`

The `chmod` command is used to change permissions of files and directories. It supports two modes of operation: symbolic mode and octal mode.

1. Symbolic Mode:
   - Syntax: `chmod [options] [permissions] file(s)`

   - Examples:
     - Grant read and write permissions to the owner: `chmod u+rw file.txt`
     - Revoke execute permission from the group: `chmod g-x script.sh`
     - Add read and execute permissions to others: `chmod o+rx program`
     - Combined permissions: `chmod u=rw,go=r file.txt`

2. Octal Mode:
   - Syntax: `chmod [options] [mode] file(s)`

   - Examples:
     - Set read, write, and execute permissions for owner, group, and others: `chmod 755 script.sh`
     - Restrict permissions to the owner only: `chmod 700 private.txt`
     - Grant full permissions to everyone: `chmod 777 public_dir`

Changing Ownership

`chown`

The `chown` command is used to change the ownership of files and directories.

- Syntax: `chown [options] owner:group file(s)`

- Examples:
  - Change the owner and group of a file: `chown john:users file.txt`
  - Recursively change ownership for a directory and its contents: `chown -R alice:staff project_dir`

`chgrp`

The `chgrp` command is used to change the group ownership of files and directories.

- Syntax: `chgrp [options] group file(s)`

- Examples:
  - Change the group of a file: `chgrp developers script.sh`
  - Recursively change group ownership for a directory and its contents: `chgrp -R team project_dir`

In Linux, you can change permissions using the `chmod` command. The permissions determine who can read, write, or execute a file. Here's how you can use `chmod` to change permissions:

Syntax:
```
chmod [options] permissions file(s)
```

Options:
- `-R` or `--recursive`: Change permissions recursively for directories and their contents.

Permissions:
- `u` (user/owner): The owner of the file.
- `g` (group): The group associated with the file.
- `o` (others): Users who are neither the owner nor part of the group.
- `a` (all/everyone): All users, including the owner, group, and others.

Each permission has three possible settings:
- `r` (read): Permission to read the file or view directory contents.
- `w` (write): Permission to modify or delete the file or add/remove files in a directory.
- `x` (execute): Permission to execute a file or access a directory.

Examples:

1. Grant read and write permissions to the owner:
```
chmod u+rw file.txt
```

2. Revoke execute permission from the group:
```
chmod g-x script.sh
```

3. Add read and execute permissions to others:
```
chmod o+rx program
```

4. Set read and write permissions for the owner, and read-only permissions for the group and others:
```
chmod u=rw,g=r,o=r file.txt
```

5. Change permissions recursively for a directory and its contents:
```
chmod -R u+w project/
```

Top 100 LINUX COMMANDS

Sure! Here is a list of 100 commonly used Linux commands with examples:

1. ls: List files and directories in the current directory.
   Example: `ls`

2. cd: Change directory.
   Example: `cd /path/to/directory`

3. pwd: Print the current working directory.
   Example: `pwd`

4. mkdir: Create a new directory.
   Example: `mkdir directory_name`

5. rmdir: Remove an empty directory.
   Example: `rmdir directory_name`

6. cp: Copy files and directories.
   Example: `cp file.txt destination_folder/`

7. mv: Move or rename files and directories.
   Example: `mv file.txt new_file.txt` or `mv file.txt destination_folder/`

8. rm: Remove files and directories.

   Example: `rm file.txt` or `rm -r directory/` (recursively)

9. cat: Concatenate and display file content.
   Example: `cat file.txt`

10. less: View file content page by page.
    Example: `less file.txt`

11. head: Display the first few lines of a file.
    Example: `head file.txt`

12. tail: Display the last few lines of a file.
    Example: `tail file.txt`

13. grep: Search for a pattern in files.
    Example: `grep "pattern" file.txt`

14. find: Search for files and directories.
    Example: `find /path/to/search -name "filename"`

15. chmod: Change file permissions.
    Example: `chmod 755 file.txt` (gives read, write, execute permissions to the owner, read and execute permissions to the group and others)

16. chown: Change file ownership.
    Example: `chown user:group file.txt`

17. tar: Archive files and directories.
    Example: `tar -cvf archive.tar file.txt` (create an archive)

18. gzip: Compress files.
    Example: `gzip file.txt` (creates file.txt.gz)

19. gunzip: Decompress files compressed with gzip.
    Example: `gunzip file.txt.gz`

20. wget: Download files from the web.
    Example: `wget http://example.com/file.txt`

21. curl: Transfer data from or to a server.
    Example: `curl http://example.com`

22. ssh: Connect to a remote server securely.
    Example: `ssh user@remote_server`

23. scp: Copy files between local and remote machines.
    Example: `scp file.txt user@remote_server:/path/to/destination`

24. ping: Send ICMP echo requests to a network host.
    Example: `ping example.com`

25. ifconfig: Display or configure network interfaces.
    Example: `ifconfig`

26. netstat: Network statistics.
    Example: `netstat -an`

27. ip: Show or manipulate routing, devices, policy routing, and tunnels.
    Example: `ip addr show`

28. route: Show or manipulate the IP routing table.
    Example: `route -n`

29. whois: Retrieve WHOIS information for a domain.
    Example: `whois example.com`

30. uname: Print system information.
    Example: `uname -a`

31. ps: Display running processes.
    Example: `ps aux`

32. top: Monitor system processes in real-time.
    Example: `top`

33. kill: Terminate processes by ID or name.
    Example: `kill 1234` or `killall process_name`

34. df: Report file system disk space usage.
    Example: `df -h`

35. du: Estimate file and directory space usage.
    Example: `du -sh directory`

36. mount: Mount a file system or device.
    Example: `mount /dev/sda1 /mnt`

37. umount: Unmount a mounted file system.
    Example: `umount /mnt`

38. ln: Create hard or symbolic links.
    Example: `ln -s /path/to/file link_name`

39. echo: Print a message.
    Example: `echo "Hello, world!"`

40. date: Display the current date and time.
    Example: `date`

41. history: View command history.
    Example: `history`

42. tar: Archive files and directories.
    Example: `tar -cvf archive.tar file.txt` (create an archive)

43. unzip: Extract compressed files in a ZIP format.
    Example: `unzip file.zip`

44. file: Determine file type.
    Example: `file file.txt`

45. awk: Pattern scanning and processing language.
    Example: `awk '{print $1}' file.txt`

46. sed: Stream editor for filtering and transforming text.
    Example: `sed 's/foo/bar/' file.txt`

47. wc: Count lines, words, and characters in a file.
    Example: `wc -l file.txt`

48. sort: Sort lines of text.
    Example: `sort file.txt`

49. diff: Compare files line by line.
    Example: `diff file1.txt file2.txt`

50. grep: Search for a pattern in files.
    Example: `grep "pattern" file.txt`

51. tr: Translate or delete characters.
    Example: `tr 'a-z' 'A-Z' file.txt`

52. cut: Remove sections from lines of files.
    Example: `cut -d ',' -f 1 file.txt`

53. du: Estimate file and directory space usage.
    Example: `du -sh directory`

54. scp: Copy files between local and remote machines.
    Example: `scp file.txt user@remote_server:/path/to/destination`

55. ssh: Connect to a remote server securely.
    Example: `ssh user@remote_server`

56. man: Display the manual page for a command.
    Example: `man ls`

57. info: View command information and documentation.
    Example: `info command_name`

58. apt-get: Package handling utility for Debian-based systems.
    Example: `apt-get install package_name`

59. yum: Package manager for RPM-based systems.
    Example: `yum install package_name`

60. systemctl: Control the systemd system and service manager.
    Example: `systemctl start service_name`

61. service: Run a System V init script.
    Example: `service service_name start`

62. chown: Change file ownership.
    Example: `chown user:group file.txt`

63. chmod: Change file permissions.
    Example: `chmod 755 file.txt`

64. ln: Create hard or symbolic links.
    Example: `ln -s /path/to/file link_name`

65. tee: Redirect output to multiple files or commands.
    Example: `command | tee file.txt`

66. fg: Bring a background process to the foreground.
    Example: `fg %1`

67. bg: Send a process to the background.
    Example: `bg %1`

68. alias: Create an alias for a command.
    Example: `alias l='ls -l'`

69. source: Execute commands from a file in the current shell.
    Example: `source script.sh`

70. echo: Print a message.
    Example: `echo "Hello, world!"`

71. export: Set environment variables.
    Example: `

```
export VARIABLE_NAME=value`

72. env: Display the current environment variables.
    Example: `env`

73. sleep: Delay for a specified amount of time.
    Example: `sleep 5` (sleep for 5 seconds)

74. su: Switch user or become superuser.
    Example: `su username` or `su -` (switch to root user)

75. sudo: Execute a command as the superuser.
    Example: `sudo command_name`

76. passwd: Change user password.
    Example: `passwd username`

77. useradd: Create a new user.
    Example: `useradd username`

78. usermod: Modify user account settings.
    Example: `usermod -aG group_name username`

79. groupadd: Create a new group.
    Example: `groupadd group_name`

80. groupmod: Modify group settings.
    Example: `groupmod -n new_group_name old_group_name`

81. crontab: Schedule commands to run at specific times.
    Example: `crontab -e` (edit cron jobs)

82. at: Execute commands at a specified time.
    Example: `at 10:00AM` (enter commands and press Ctrl+D)

83. shutdown: Shutdown or restart the system.
    Example: `shutdown now` (shutdown immediately)

84. reboot: Reboot the system.
    Example: `reboot`

85. ifconfig: Display or configure network interfaces.
    Example: `ifconfig`

86. netstat: Network statistics.
    Example: `netstat -an`

87. ip: Show or manipulate routing, devices, policy routing, and tunnels.
```

```
    Example: `ip addr show`

88. route: Show or manipulate the IP routing table.
    Example: `route -n`

89. iptables: Administration tool for IPv4 packet filtering and NAT.
    Example: `iptables -L` (list firewall rules)

90. adduser: Interactive tool for adding new users.
    Example: `adduser username`

91. deluser: Remove a user account and associated files.
    Example: `deluser username`

92. passwd: Change user password.
    Example: `passwd username`

93. crontab: Schedule commands to run at specific times.
    Example: `crontab -e` (edit cron jobs)

94. history: View command history.
    Example: `history`

95. file: Determine file type.
    Example: `file file.txt`

96. hostname: Print or set the system's hostname.
    Example: `hostname`

97. uptime: Display the system's uptime.
    Example: `uptime`

98. dmesg: Print or control the kernel ring buffer.
    Example: `dmesg`

99. free: Display amount of free and used memory.
    Example: `free -h`

100. htop: Interactive process viewer and system monitor.
     Example: `htop`

Please note that the examples provided assume a general usage and may require
modifications based on your specific system and file paths.

These commands cover a wide range of tasks and utilities in Linux. Feel free
to explore their options and further usage by referring to their respective
man pages or online documentation.
```