Commonly Asked Linux Commands For Interviews

1. **echo:** Prints text to the console.

```
-----------------------------------------------------------------------------------------
echo "Hello, world!"
-----------------------------------------------------------------------------------------
```

2. **read:** Reads input from the user.

```
-----------------------------------------------------------------------------------------
read -p "Enter your name: " name

echo "Hello, $name"
-----------------------------------------------------------------------------------------
```

3. **if/else:** Checks a condition and executes different code based on whether the condition is true or false.

```
-----------------------------------------------------------------------------------------
if [[ $num -gt 10 ]]

then

   echo "The number is greater than 10"

else

   echo "The number is less than or equal to 10"

fi
-----------------------------------------------------------------------------------------
```

4. **for loop:** Repeats a block of code for a specified number of times.

```
------------------------------------------------------------------------------------------
for (( i=1; i<=10; i++ ))
do
   echo $i
done
------------------------------------------------------------------------------------------
```

5. **while loop:** Repeats a block of code while a condition is true.

```
------------------------------------------------------------------------------------------
i=1
while (( i<=10 ))
do
   echo $i
   i=$((i+1))
done
------------------------------------------------------------------------------------------
```

6. **case statement:** Executes different code based on the value of a variable.

```
------------------------------------------------------------------------------------------
case $fruit in
   apple)
      echo "It's an apple"
      ;;
   banana)
      echo "It's a banana"
```

```
        ;;
    *)
        echo "It's not an apple or a banana"
        ;;
esac
```
----

7. **function:** A block of code that can be called multiple times from within a script.

```
function add_numbers {
    sum=$((num1 + num2))
    echo "The sum of $num1 and $num2 is $sum"
}

num1=5
num2=10
add_numbers
```
----

8. **grep:** Searches for a pattern in a file.

```
grep "example" file.txt
```
----

9. **sed:** Searches for a pattern and replaces it with another pattern in a file.

The `sed` command (short for "stream editor") is a powerful text-processing tool available in Linux and other Unix-like operating systems. It is used to perform text transformations on an input stream (a file or input from a pipe) and output the result to the standard output.

The basic syntax of `sed` command is as follows:

```
---------------------------------------------------------------------------------------
sed OPTIONS... [SCRIPT] [INPUTFILE...]
---------------------------------------------------------------------------------------
```

Here, `OPTIONS` are the command line options that modify the behavior of the `sed` command. `SCRIPT` is the set of commands that `sed` will execute on the input file, and `INPUTFILE` is the name of the file on which `sed` will operate. If no input file is specified, `sed` will read input from the standard input (typically the keyboard).

Some common uses of the `sed` command include:

- Substituting one string for another in a file or stream of input.
- Selecting and printing specific lines from a file.
- Deleting lines from a file.
- Performing complex text transformations using regular expressions.

Here are some examples of `sed` commands in action:

- Substitute the word "apple" with "orange" in a file called `fruits.txt`:

```
  ------------------------------------------------------------------------------------------
--

  sed 's/apple/orange/g' fruits.txt

  ------------------------------------------------------------------------------------------
--
```

- Delete all lines that contain the word "banana" in a file called `fruits.txt`:

```
  ------------------------------------------------------------------------------------------
--

  sed '/banana/d' fruits.txt

  ------------------------------------------------------------------------------------------
--
```

- Print only the first 10 lines of a file called `sample.txt`:

```
  ------------------------------------------------------------------------------------------
--

  sed -n '1,10p' sample.txt

  ------------------------------------------------------------------------------------------
--
```

These are just a few examples of the many uses of `sed` command in Linux. It is a powerful tool for text processing and is widely used by system administrators and programmers.

10. **awk:** Processes text files and generates reports.

The `awk` command is a powerful text-processing tool that is available in most Unix-based operating systems, including Linux. It is used for processing and manipulating text files, and it supports pattern scanning and processing.

The basic syntax of the `awk` command is as follows:

```
------------------------------------------------------------------------------------------
awk 'pattern { action }' file
------------------------------------------------------------------------------------------
```

Here, `pattern` specifies a regular expression or a pattern to search for in the input file, and `action` is a set of commands to be executed when the pattern is matched. `file` is the input file that `awk` will process. If no input file is specified, `awk` will read input from the standard input (usually the keyboard).

Some common uses of the `awk` command include:

- Extracting specific fields or columns from a file.
- Searching for specific patterns or regular expressions in a file.
- Performing mathematical or string operations on the input data.
- Filtering or selecting specific lines from a file.

Here are some examples of `awk` commands in action:

- Print the first column of a file called `data.txt`:

```
  ------------------------------------------------------------------------------------------
--
  awk '{print $1}' data.txt
  ------------------------------------------------------------------------------------------
--
```

- Print all lines that contain the word "error" in a file called `log.txt`:

--

```
awk '/error/ {print}' log.txt
```

--

- Calculate the sum of the second column of a file called `sales.txt`:

--

```
awk '{sum+=$2} END {print sum}' sales.txt
```

--

These are just a few examples of the many uses of `awk` command in Linux shell scripting. It is a versatile tool for text processing and is widely used by system administrators and programmers.

11. **cut:** Extracts specific columns from a file.

```
cut -d"," -f1,3 file.txt
```

----

12. **sort:** Sorts lines of text in a file.

The `sort` command in Linux is a powerful tool for sorting text files, either in ascending or descending order, based on specified fields or columns.

The basic syntax of the `sort` command is as follows:

```
sort [OPTIONS] FILE
```

Here, `OPTIONS` are the command-line options that modify the behavior of the `sort` command. `FILE` is the name of the file that `sort` will process. If no input file is specified, `sort` will read input from the standard input (usually the keyboard).

Some common uses of the `sort` command include:

- Sorting lines in a file in alphabetical or numerical order.
- Sorting specific fields or columns in a file.
- Sorting a file in reverse order.

Here are some examples of `sort` commands in action:

- Sort a file called `names.txt` in alphabetical order:

```
  sort names.txt
```

- Sort a file called `sales.txt` in descending order based on the third column:

```
  ------------------------------------------------------------------------------------------
--

  sort -r -n -k 3 sales.txt

  ------------------------------------------------------------------------------------------
--
```

- Sort a file called `data.txt` based on the first field, and then the second field:

```
  ------------------------------------------------------------------------------------------
--

  sort -k 1,1 -k 2,2 data.txt

  ------------------------------------------------------------------------------------------
--
```

These are just a few examples of the many uses of `sort` command in Linux shell scripting. It is a versatile tool for sorting text files and is widely used by system administrators and programmers.

13. **uniq: Removes duplicate lines from a sorted file.**

```
--------------------------------------------------------------------------------------------

uniq file.txt

--------------------------------------------------------------------------------------------
```

14. **wc: Counts the number of lines, words, and characters in a file.**

```
--------------------------------------------------------------------------------------------

wc file.txt

--------------------------------------------------------------------------------------------
```

15. **tee:** Reads input from standard input and writes it to both standard output and one or more files.

```
echo "Hello, world!" | tee file.txt
```

Major Commands

1. `ls`: This command is used to list the contents of a directory in Linux. It is the most frequently used command and can be used to display files, directories, and other information about them. Here is an example:

```
ls
```

This will display a list of all the files and directories in the current directory.

2. `pwd`: This command is used to display the current working directory in Linux. Here is an example:

```
pwd
```

This will display the current working directory, for example: `/home/user/documents/`.

3. `cd`: This command is used to navigate through directories in Linux. You can use it to change the current directory to a specific directory. Here is an example:

```
cd /home/user/documents/
```

This will change the current directory to `/home/user/documents/`.

4. \`mkdir\`: This command is used to create a new directory in Linux. Here is an example:

```
-------------------------------------------------------------------------------------------
mkdir new_directory
-------------------------------------------------------------------------------------------
```

This will create a new directory called \`new_directory\`.

5. \`mv\`: This command is used to move or rename files or directories in Linux. Here are a couple of examples:

```
-------------------------------------------------------------------------------------------
mv old_directory new_directory
-------------------------------------------------------------------------------------------
```

This will move the directory \`old_directory\` to \`new_directory\`.

```
-------------------------------------------------------------------------------------------
mv file.txt new_name.txt
-------------------------------------------------------------------------------------------
```

This will rename the file \`file.txt\` to \`new_name.txt\`.

6. \`cp\`: This command is used to copy files or directories in Linux. Here is an example:

```
-------------------------------------------------------------------------------------------
cp file.txt copy_file.txt
-------------------------------------------------------------------------------------------
```

This will create a copy of the file `file.txt` called `copy_file.txt`.

7. `rm`: This command is used to delete files or directories in Linux. Here are a couple of examples:

```
---------------------------------------------------------------------------------------------
rm file.txt
---------------------------------------------------------------------------------------------
```

This will delete the file `file.txt`.

```
---------------------------------------------------------------------------------------------
rm -r directory/
---------------------------------------------------------------------------------------------
```

This will delete the directory `directory/` and all its contents.

8. `touch`: This command is used to create a new, empty file in Linux. Here is an example:

```
---------------------------------------------------------------------------------------------
touch new_file.txt
---------------------------------------------------------------------------------------------
```

This will create a new, empty file called `new_file.txt`.

9. `ln`: This command is used to create a symbolic link or shortcut to another file or directory. Here is an example:

```
-------------------------------------------------------------------------------------------
ln -s /path/to/source/file /path/to/link
-------------------------------------------------------------------------------------------
```

This will create a symbolic link called `link` to the file `file` located at `/path/to/source/`.

10. `cat`: This command is used to display the contents of a file on the terminal. Here is an example:

```
-------------------------------------------------------------------------------------------
cat file.txt
-------------------------------------------------------------------------------------------
```

This will display the contents of `file.txt` on the terminal.

3. `clear`: This command is used to clear the terminal display. Here is an example:

```
-------------------------------------------------------------------------------------------
clear
-------------------------------------------------------------------------------------------
```

This will clear the terminal display.

4. `echo`: This command is used to print any text that follows the command on the terminal. Here is an example:

```
-------------------------------------------------------------------------------------------
echo "Hello, World!"
-------------------------------------------------------------------------------------------
```

This will print the text "Hello, World!" on the terminal.

5. `less`: This command is used to display paged outputs in the terminal, usually for long texts or files. Here is an example:

```
less file.txt
```

This will display the contents of `file.txt` in a paged output on the terminal, allowing the user to navigate and search through the file.

6. `man`: This command is used to access manual pages for all Linux commands. Here is an example:

```
man ls
```

This will display the manual page for the `ls` command, providing information on how to use it and its various options.

7. `uname`: This command is used to get basic information about the operating system. Here is an example:

```
uname -a
```

This will display information about the system architecture, kernel version, and other details.

8. `whoami`: This command is used to display the current active username on the terminal. Here is an example:

```
-----------------------------------------------------------------------------------------------
whoami
-----------------------------------------------------------------------------------------------
```

This will display the active username on the terminal.

Sure, here are explanations and examples for each of the commands you requested:

1. `tar`: This command is used to extract and compress files in Linux. Here are a couple of examples:

```
-----------------------------------------------------------------------------------------------
tar -xvf archive.tar
-----------------------------------------------------------------------------------------------
```

This will extract the contents of the `archive.tar` file.

```
-----------------------------------------------------------------------------------------------
tar -cvf archive.tar file1.txt file2.txt
-----------------------------------------------------------------------------------------------
```

This will create a new archive called `archive.tar` that contains `file1.txt` and `file2.txt`.

2. `grep`: This command is used to search for a string within an output. Here is an example:

```
----------------------------------------------------------------------------------------------

grep "hello" file.txt

----------------------------------------------------------------------------------------------
```

This will search for the string "hello" within `file.txt` and return all lines that contain that string.

3. `head`: This command is used to return the specified number of lines from the top of a file. Here is an example:

```
----------------------------------------------------------------------------------------------

head -n 10 file.txt

----------------------------------------------------------------------------------------------
```

This will return the first 10 lines of `file.txt`.

4. `tail`: This command is used to return the specified number of lines from the bottom of a file. Here is an example:

```
----------------------------------------------------------------------------------------------

tail -n 5 file.txt

----------------------------------------------------------------------------------------------
```

This will return the last 5 lines of `file.txt`.

5. `diff`: This command is used to find the difference between two files. Here is an example:

```
-------------------------------------------------------------------------------------------

diff file1.txt file2.txt

-------------------------------------------------------------------------------------------
```

This will show the differences between `file1.txt` and `file2.txt`.

6. `cmp`: This command is used to check if two files are identical. Here is an example:

```
-------------------------------------------------------------------------------------------

cmp file1.txt file2.txt

-------------------------------------------------------------------------------------------
```

This will compare `file1.txt` and `file2.txt` and report whether they are identical or not.

7. `comm`: This command is used to combine the functionality of `diff` and `cmp`. Here is an example:

```
-------------------------------------------------------------------------------------------

comm file1.txt file2.txt

-------------------------------------------------------------------------------------------
```

This will compare `file1.txt` and `file2.txt` and show the lines that are common to both files, as well as the lines that are unique to each file.

Sure, here are explanations and examples for each of the commands you requested:

1. `sort`: This command is used to sort the content of a file while outputting. Here is an example:

```
--------------------------------------------------------------------------------------------

sort file.txt

--------------------------------------------------------------------------------------------
```

This will sort the contents of `file.txt` and output the sorted content to the terminal.

2. `export`: This command is used to export environment variables in Linux. Here is an example:

```
--------------------------------------------------------------------------------------------

export MY_VAR=my_value

--------------------------------------------------------------------------------------------
```

This will create a new environment variable called `MY_VAR` with a value of `my_value`.

3. `zip`: This command is used to zip files in Linux. Here is an example:

```
--------------------------------------------------------------------------------------------

zip archive.zip file1.txt file2.txt

--------------------------------------------------------------------------------------------
```

This will create a new zip archive called `archive.zip` that contains `file1.txt` and `file2.txt`.

4. `unzip`: This command is used to unzip files in Linux. Here is an example:

```
--------------------------------------------------------------------------------------------

unzip archive.zip
```

This will extract the contents of `archive.zip` into the current directory.

5. `ssh`: This command is used to establish a secure shell connection in Linux. Here is an example:

```
ssh user@server.com
```

This will establish a secure shell connection to the server at `server.com` using the `user` account.

6. `service`: This command is used to start and stop services in Linux. Here is an example:

```
service nginx start
```

This will start the nginx web server.

7. `ps`: This command is used to display active processes in Linux. Here is an example:

```
ps aux
```

This will display a list of all active processes on the system.

8. `kill` and `killall`: These commands are used to kill active processes by process ID or name. Here are a couple of examples:

```
kill 1234
```

This will kill the process with ID `1234`.

```
killall nginx
```

This will kill all processes with the name `nginx`.

Sure, here are explanations and examples for each of the commands you requested:

1. `df`: This command is used to display disk filesystem information in Linux. Here is an example:

```
df -h
```

This will display the filesystem information in a human-readable format.

2. `mount`: This command is used to mount file systems in Linux. Here is an example:

```
------------------------------------------------------------------------------------------
mount /dev/sdb1 /mnt/usb
------------------------------------------------------------------------------------------
```

This will mount the file system located at `/dev/sdb1` to the directory `/mnt/usb`.

3. `chmod`: This command is used to change file permissions in Linux. Here is an example:

```
------------------------------------------------------------------------------------------
chmod 755 file.txt
------------------------------------------------------------------------------------------
```

This will change the permissions of `file.txt` so that the owner can read, write, and execute the file, while other users can only read and execute it.

4. `chown`: This command is used for granting ownership of files or folders in Linux. Here is an example:

```
------------------------------------------------------------------------------------------
chown user:group file.txt
------------------------------------------------------------------------------------------
```

This will change the ownership of `file.txt` to the user `user` and the group `group`.

5. `ifconfig`: This command is used to display network interfaces and IP addresses in Linux. Here is an example:

```
------------------------------------------------------------------------------------------
ifconfig
```

```
------------------------------------------------------------------------------------------
```

This will display the network interfaces and their associated IP addresses.

6. `traceroute`: This command is used to trace all the network hops to reach the destination in Linux. Here is an example:

```
------------------------------------------------------------------------------------------
traceroute google.com
------------------------------------------------------------------------------------------
```

This will trace the network route to reach `google.com`.

7. `wget`: This command is used to directly download files from the internet in Linux. Here is an example:

```
------------------------------------------------------------------------------------------
wget https://example.com/file.txt
------------------------------------------------------------------------------------------
```

This will download the file located at `https://example.com/file.txt` to the current directory.

8. `ufw`: This command is used to configure and manage firewall settings in Linux. Here is an example:

```
------------------------------------------------------------------------------------------
ufw allow 80/tcp
------------------------------------------------------------------------------------------
```

This will allow incoming TCP traffic on port `80`.

Sure, here are explanations and examples for each of the commands you requested:

1. `iptables`: This command is used as a base firewall for all other firewall utilities to interface with in Linux. Here is an example:

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

This will allow incoming TCP traffic on port `80`.

2. `apt`, `pacman`, `yum`, `rpm`: These are package managers that are used to install and manage software packages on Linux. The specific package manager you use depends on your Linux distribution. Here is an example using `apt`:

```
sudo apt install firefox
```

This will install the Firefox web browser using the `apt` package manager.

3. `sudo`: This command is used to escalate privileges in Linux. Here is an example:

```
sudo apt update
```

This will update the package list using `apt` package manager with elevated privileges.

4. `cal`: This command is used to view a command-line calendar in Linux. Here is an example:

```
   -------------------------------------------------------------------------------------------

   cal

   -------------------------------------------------------------------------------------------
```

This will display the calendar for the current month.

5. `alias`: This command is used to create custom shortcuts for your regularly used commands in Linux. Here is an example:

```
   -------------------------------------------------------------------------------------------

   alias ll='ls -alF'

   -------------------------------------------------------------------------------------------
```

This will create an alias `ll` for the `ls -alF` command, which will display the detailed list of files and directories.

6. `dd`: This command is majorly used for creating bootable USB sticks in Linux. Here is an example:

```
   -------------------------------------------------------------------------------------------

   sudo dd if=ubuntu.iso of=/dev/sdb bs=4M status=progress

   -------------------------------------------------------------------------------------------
```

This will write the `ubuntu.iso` image to the USB stick located at `/dev/sdb` with a block size of `4M`.

7. `whereis`: This command is used to locate the binary, source, and manual pages for a command in Linux. Here is an example:

```
whereis ls
```

This will display the binary, source, and manual pages for the `ls` command.

8. `whatis`: This command is used to find what a command is used for in Linux. Here is an example:

```
whatis ls
```

This will display a brief description of what the `ls` command is used for.

9. `top`: This command is used to view active processes live with their system usage in Linux. Here is an example:

```
top
```

This will display the list of active processes along with their CPU, memory, and other system usage statistics.

10. `useradd` and `usermod`: These commands are used to add new users or change existing users' data in Linux. Here is an example:

```
sudo useradd -m john
```

This will create a new user named `john` with a home directory.

11. `passwd`: This command is used to create or update passwords for existing users in Linux. Here is an example:

```
sudo passwd john
```

This will set a new password for the user `john`.