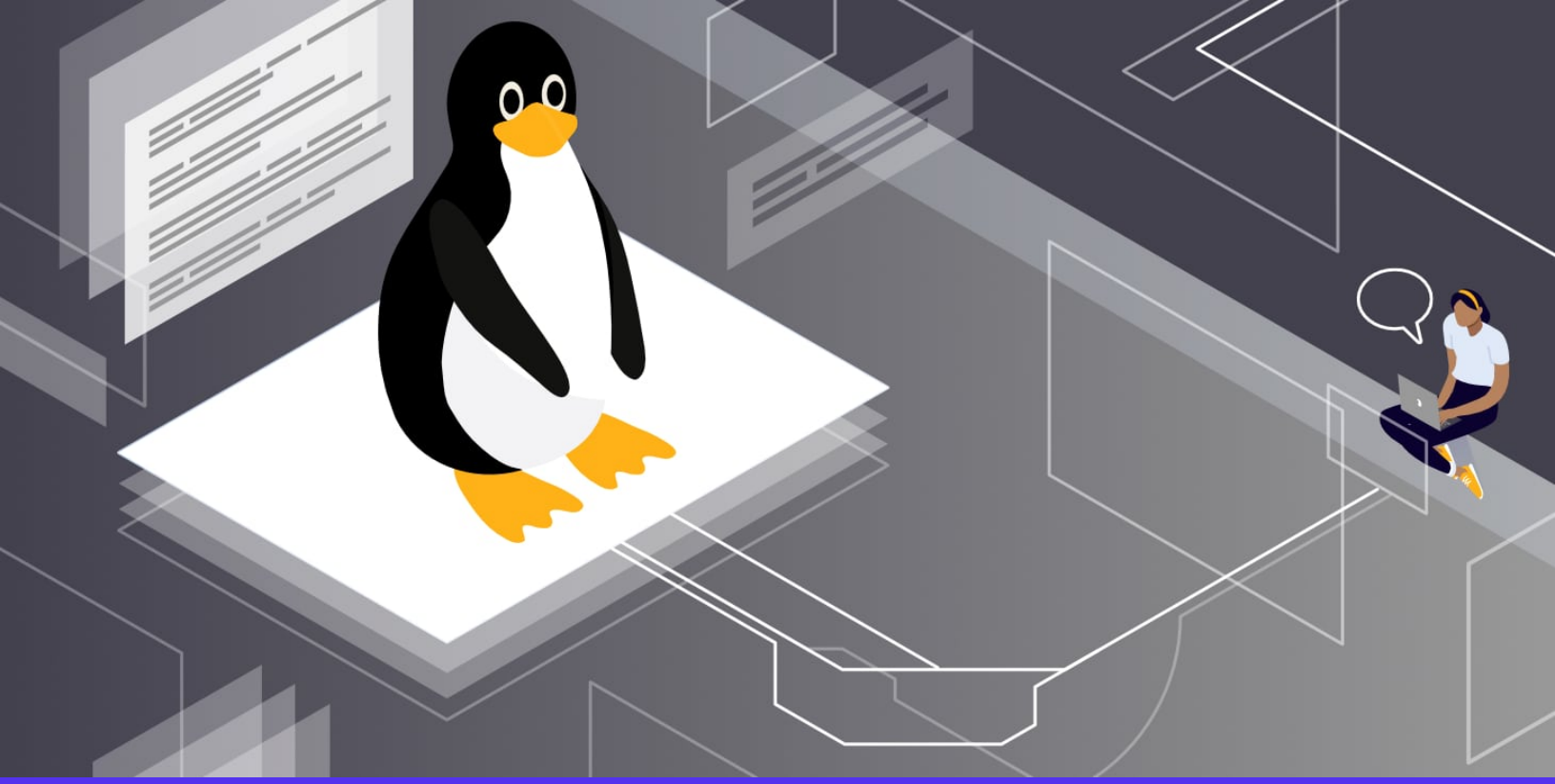


The 40 Most-Used Linux Commands You Should Know

KINSTA

As of writing this, Linux has a worldwide market share of 2.68% on desktops, but over 90% of all cloud infrastructure and hosting services run in this operating system. For this reason alone, it is crucial to be familiar with popular Linux commands.

According to a StackOverflow survey, Linux is the most-used operating system by professional developers, with an impressive 55.9% of the market share. It isn't just a coincidence. Linux is free and open-source, has better security than its competitors, and boasts a powerful command line that makes developers and power users more effective. You also have access to a powerful package manager and a bunch of development tools like DevKinsta.

Whether you're an experienced Sysadmin or a Linux newcomer, you can take advantage of this guide.

Let's begin!

What Is a Linux Command?

A Linux command is a program or utility that runs on the command line. A command line is an interface that accepts lines of text and processes them into instructions for your computer.

Any graphical user interface (GUI) is just an abstraction of command-line programs. For example, when you close a window by clicking on the “X,” there’s a command running behind that action.

A **flag** is a way we can pass options to the command you run. Most Linux commands have a help page that we can call with the flag `-h`. Most of the time, flags are optional.

An **argument** or parameter is the **input** we give to a command so it can run properly. In most cases, the argument is a file path, but it can be anything you type in the terminal.

You can invoke flags using hyphens (`-`) and double hyphens (`--`), while argument execution depends on the order in which you pass them to the function.

The 40 Most-Used Linux Commands

Before jumping into the most-used Linux commands, make sure to fire up a **terminal**. In most Linux distributions, you would use `Ctrl + Alt + T` to do so. If this isn’t working, search in your application panel for “terminal.”

— The Linux terminal emulator.

Now let's dive into the 40 most-used Linux commands. Many of these have multiple options you can string to them, so make sure to check out the commands' manual.

1. `ls` Command

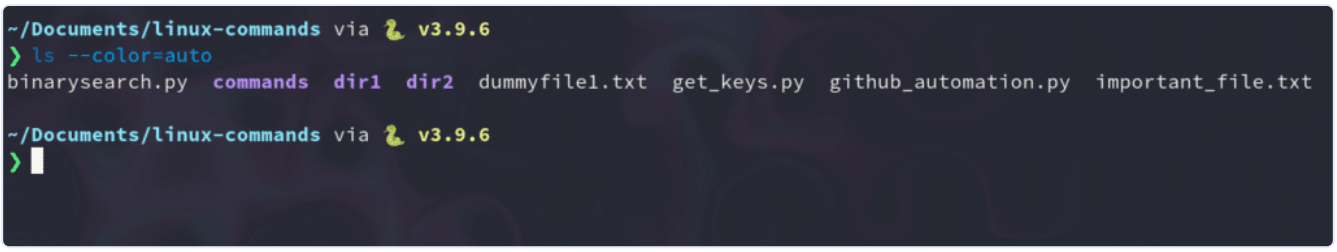
`ls` is probably the first command every Linux user typed in their terminal. It allows you to list the contents of the directory you want (the current directory by default), including files and other nested directories.

```
ls
```

It has many options, so it might be good to get some help by using the `--help` flag. This flag returns all the flags you can use with ls.

For example, to colorize the output of the `ls` command, you can use the following:

```
ls --color=auto
```



— The colorized ls command.

Now the `ls` command output is colorized, and you can appreciate the difference between a directory and a file.

But typing `ls` with the color flag would be inefficient; that's why we use the `alias` command.

2. `alias` Command

The `alias` command lets you define temporary aliases in your shell session. When creating an alias, you instruct your shell to replace a word with a series of commands.

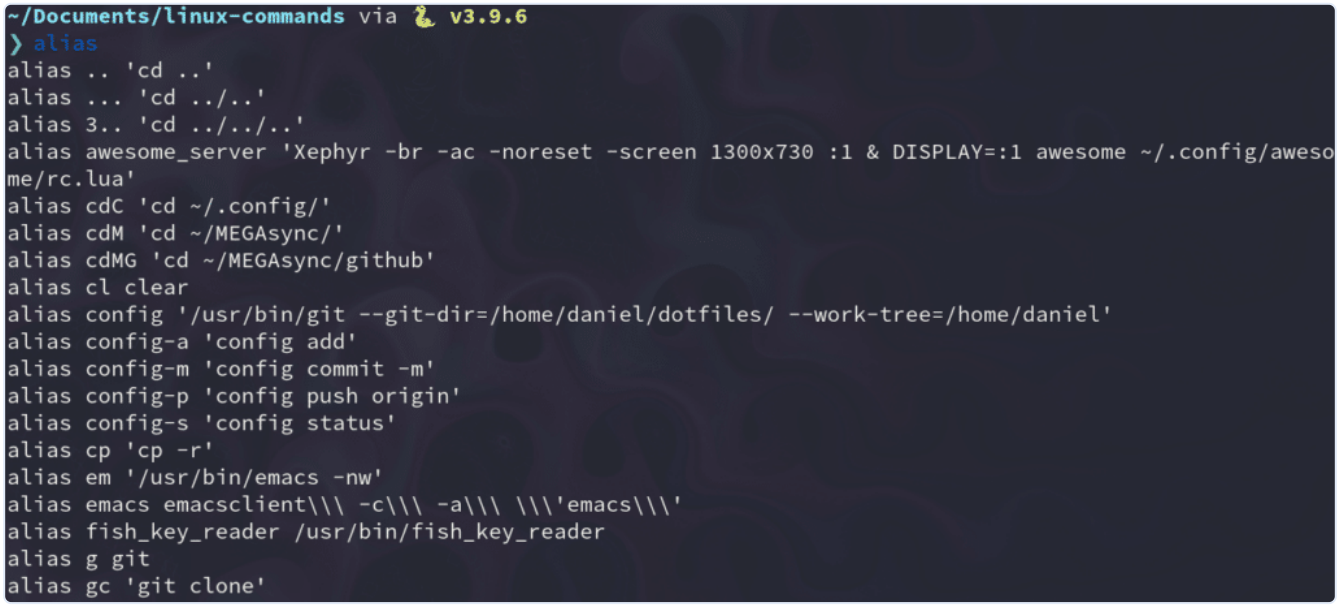
For example, to set `ls` to have color without typing the `--color` flag every time, you would use:

```
alias ls="ls --color=auto"
```

As you can see, the `alias` command takes one key-value pair parameter: `alias NAME="VALUE"`. Note that the value must be inside quotes.

If you want to list all the aliases you have in your shell session, you can run the `alias` command without argument.

```
alias
```



— The alias command.

3. `unalias` Command

As the name suggests, the `unalias` command aims to remove an `alias` from the already defined aliases. To remove the previous `ls` alias, you can use:

```
unalias ls
```

4. `pwd` Command

The `pwd` command stands for “print working directory,” and it outputs the absolute path of the directory you’re in. For example, if your username is “john” and you’re in your Documents directory, its absolute path would be: `/home/john/Documents`.

To use it, simply type pwd in the terminal:

```
pwd

# My result: /home/kinsta/Documents/linux-commands
```

5. `cd` Command

The `cd` command is highly popular, along with `ls`. It refers to “**c**hange **d**irectory” and, as its name suggests, switches you to the directory you’re trying to access.

For instance, if you’re inside your Documents directory and you’re trying to access one of its subfolders called **Videos**, you can enter it by typing:

```
cd Videos
```

You can also supply the absolute path of the folder:

```
cd /home/kinsta/Documents/Videos
```

There are some tricks with the `cd` command that can save you a lot of time when playing around with it:

1) Go to the home folder

```
cd
```

2) Move a level up

```
cd ..
```

3) Return to the previous directory

```
cd -
```

6. `cp` Command

It's so easy to copy files and folders directly in the Linux terminal that sometimes it can replace conventional file managers.

To use the `cp` command, just type it along with the source and destination files:

```
cp file_to_copy.txt new_file.txt
```

You can also copy entire directories by using the recursive flag:

```
cp -r dir_to_copy/ new_copy_dir/
```

Remember that in Linux, folders end with a forward slash (`/`).

7. `rm` Command

Now that you know how to copy files, it'll be helpful to know how to remove them.

You can use the `rm` command to remove files and directories. Be careful while using it, though, because it’s very difficult (yet not impossible) to recover files deleted this way.

To delete a regular file, you’d type:

```
rm file_to_copy.txt
```

If you want to delete an empty directory, you can use the recursive (`-r`) flag:

```
rm -r dir_to_remove/
```

On the other hand, to remove a directory with content inside of it, you need to use the force (-f) and recursive flags:

```
rm -rf dir_with_content_to_remove/
```

Info

Be careful with this — you can erase a whole day of work by misusing these two flags!

8. `mv` Command

You use the `mv` command to move (or rename) files and directories through your file system.

To use this command, you'd type its name with the source and destination files:

```
mv source_file destination_folder/

mv command_list.txt commands/
```

To utilize absolute paths, you'd use:

```
mv /home/kinsta/BestMoviesOfAllTime ./
```

…where `./` is the directory you're currently in.

You also can use `mv` to rename files while keeping them in the same directory:

```
mv old_file.txt new_named_file.txt
```

9. `mkdir` Command

To create folders in the shell, you use the `mkdir` command. Just specify the new folder's name, ensure it doesn't exist, and you're ready to go.

For example, to make a directory to keep all of your images, just type:

```
mkdir images/
```

To create subdirectories with a simple command, use the parent (`-p`) flag:

```
mkdir -p movies/2004/
```

10. `man` Command

Another essential Linux command is `man`. It displays the manual page of any other command (as long as it has one).

To see the manual page of the `mkdir` command, type:

```
man mkdir
```

You could even refer to the `man` manual page:

```
man man
```

```
MAN(1)                    Manual pager utils                    MAN(1)

NAME
       man - an interface to the system reference manuals

SYNOPSIS
       man [man options] [[section] page ...] ...
       man -k [apropos options] regexp ...
       man -K [man options] [section] term ...
       man -f [whatis options] page ...
       man -l [man options] file ...
       man -w|-W [man options] page ...

DESCRIPTION
       man  is  the  system's manual pager.  Each page argument
       given to man is normally the name of a program,  utility
       or  function.   The  manual page associated with each of
       these arguments is then found and displayed.  A section,
       if  provided,  will direct man to look only in that sec-
       tion of the manual.  The default action is to search  in
       all  of  the  available sections following a pre-defined
       order (see DEFAULTS), and to show only  the  first  page
       found, even if page exists in several sections.

       The  table below shows the section numbers of the manual
       followed by the types of pages they contain.

       1   Executable programs or shell commands
       2   System calls (functions provided by the kernel)
       3   Library calls (functions within program libraries)
       4   Special files (usually found in /dev)
       5   File formats and conventions, e.g. /etc/passwd
       6   Games
 Manual page man(1) line 1 (press h for help or q to quit)
```

— The manual page of “man.”

11. `touch` Command

The `touch` command allows you to update the access and modification times of the specified files.

For example, I have an old file that was last modified on April 12th:

```
~/Documents/linux-commands via 🐍 v3.9.6
❯ ls -lah
Permissions Size User   Date Modified Name
drwxr-xr-x     - daniel  8 ago 15:11  .
drwxr-xr-x     - daniel  8 ago 00:27  ..
drwxr-xr-x     - daniel  8 ago 00:34  commands
drwxr-xr-x     - daniel  7 ago 00:45  dir1
drwxr-xr-x     - daniel  7 ago 00:45  dir2
drwxr-xr-x     - daniel  8 ago 00:10  dir_to_copy
drwxr-xr-x     - daniel  8 ago 00:12  new_dir
.rw-r--r--     0 daniel  8 ago 00:38  BestMoviesOfAllTime
.rw-r--r--     0 daniel  7 ago 00:44  binarysearch.py
.rw-r--r--     0 daniel  7 ago 00:43  dummyfile1.txt
.rw-r--r--     0 daniel  8 ago 00:18  file_to_delete.txt
.rw-r--r--     0 daniel  7 ago 00:44  get_keys.py
.rw-r--r--     0 daniel  7 ago 00:44  github_automation.py
.rw-r--r--     0 daniel  7 ago 00:44  important_file.txt
.rw-r--r--     0 daniel  8 ago 00:04  new_file.txt
.rw-r--r--     0 daniel 12 abr 20:45  old_file
```

— Old date.

To change its modification date to the current time, we need to use the `-m` flag:

```
touch -m old_file
```

Now the date matches today's date (which at the time of writing was August 8th).

```
~/Documents/linux-commands via 🐍 v3.9.6
❯ ls -lah
Permissions Size User   Date Modified Name
drwxr-xr-x     - daniel  8 ago 15:11  .
drwxr-xr-x     - daniel  8 ago 00:27  ..
drwxr-xr-x     - daniel  8 ago 00:34  commands
drwxr-xr-x     - daniel  7 ago 00:45  dir1
drwxr-xr-x     - daniel  7 ago 00:45  dir2
drwxr-xr-x     - daniel  8 ago 00:10  dir_to_copy
drwxr-xr-x     - daniel  8 ago 00:12  new_dir
.rw-r--r--     0 daniel  8 ago 00:38  BestMoviesOfAllTime
.rw-r--r--     0 daniel  7 ago 00:44  binarysearch.py
.rw-r--r--     0 daniel  7 ago 00:43  dummyfile1.txt
.rw-r--r--     0 daniel  8 ago 00:18  file_to_delete.txt
.rw-r--r--     0 daniel  7 ago 00:44  get_keys.py
.rw-r--r--     0 daniel  7 ago 00:44  github_automation.py
.rw-r--r--     0 daniel  7 ago 00:44  important_file.txt
.rw-r--r--     0 daniel  8 ago 00:04  new_file.txt
.rw-r--r--     0 daniel  8 ago 16:30  old_file
```

— New date

Nonetheless, most of the time, you won't use `touch` to modify file dates, but rather to create new empty files:

```
touch new_file_name
```

12. `chmod` Command

The `chmod` command lets you change the mode of a file (permissions) quickly. It has a lot of options available with it.

The basic permissions a file can have are:

- r (read)
- w (write)
- x (execute)

One of the most common use cases for `chmod` is to make a file executable by the user. To do this, type `chmod` and the flag `+x`, followed by the file you want to modify permissions on:

```
chmod +x script
```

You use this to make scripts executable, allowing you to run them directly by using the `./` notation.

13. `./` Command

Maybe the `./` notation isn't a command itself, but it's worth mentioning in this list. It lets your shell run an executable file with any interpreter installed in your system directly from the terminal. No more double-clicking a file in a graphical file manager!

For instance, with this command, you can run a Python script or a program only available in .run format, like XAMPP. When running an executable, make sure it has executable (x) permissions, which you can modify with the `chmod` command.

Here's a simple Python script and how we would run it with the `./` notation:

```
#! /usr/bin/python3

# filename: script

for i in range(20):

print(f"This is a cool script {i}")
```

Here’s how we’d convert the script into an executable and run it:

```
chmod +x script

./script
```

14. `exit` Command

The `exit` command does exactly what its name suggests: With it, you can end a shell session and, in most cases, automatically close the terminal you’re using:

```
exit
```

15. sudo Command

This command stands for “superuser do,” and it lets you act as a superuser or root user while you’re running a specific command. It’s how Linux protects itself and prevents users from accidentally modifying the machine’s filesystem or installing inappropriate packages.

Sudo is commonly used to install software or to edit files outside the user’s home directory:

```
sudo apt install gimp

sudo cd /root/
```

It’ll ask you for the administrator’s password before running the command you typed after it.

16. `shutdown` Command

As you may guess, the `shutdown` command lets you power off your machine. However, it also can be used to halt and reboot it.

To power off your computer immediately (the default is one minute), type:

```
shutdown now
```

You can also schedule to turn off your system in a 24-hour format:

```
shutdown 20:40
```

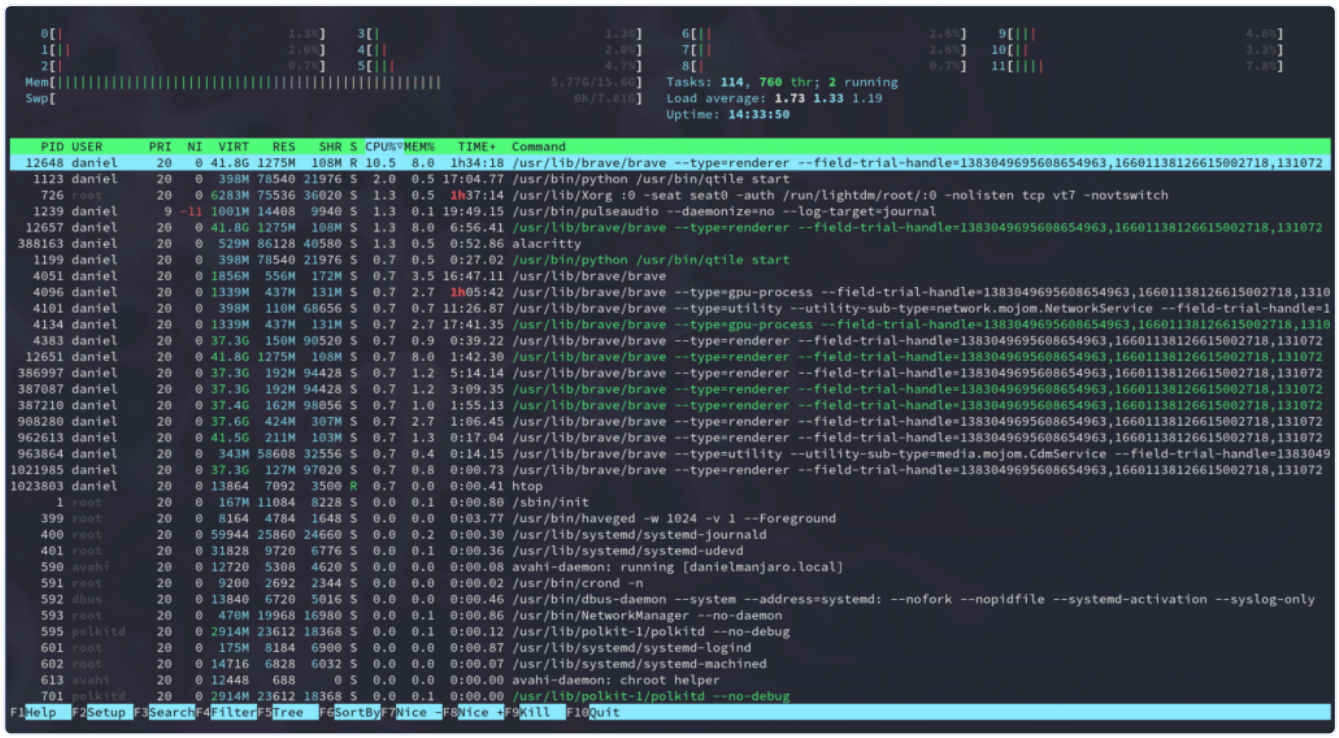
To cancel a previous `shutdown` call, you can use the `-c` flag:

```
shutdown -c
```

17. `htop` Command

`htop` is an interactive process viewer that lets you manage your machine's resources directly from the terminal. In most cases, it isn't installed d by default, so make sure to read more about it on its download page.

```
htop
```



— The “htop” interface.

18. `unzip` Command

The unzip command allows you to extract the content of a **.zip** file from the terminal. Once again, this package may not be installed by default, so make sure you install it with your package manager.

Here, we’re unpacking a .zip file full of images:

```
unzip images.zip
```

19. `apt`, `yum`, `pacman` commands

No matter which Linux distribution you're using, it's likely that you use package managers to install, update, and remove the software you use every day.

You can access these package managers through the command line, and you'd use one or another depending on the distro your machine is running.

The following examples will install GIMP, a free and open source software usually available in most package managers:

Debian-based (Ubuntu, Linux Mint)

```
sudo apt install gimp
```

Red Hat-based (Fedora, CentOS)

```
sudo yum install gimp
```

Arch-based (Manjaro, Arco Linux)

```
sudo pacman -S gimp
```

20. echo Command

The `echo` command displays defined text in the terminal — it's that simple:

```
echo "Cool message"
```

— The echo command

Its primary usage is to print environmental variables inside those messages:

```
echo "Hey $USER"

# Hey kinsta
```

21. `cat` Command

`Cat`, short for "concatenate," lets you create, view, and concatenate files directly from the terminal. It's mainly used to preview a file without opening a graphical text editor:

```
cat long_text_file.txt
```

— The cat command.

22. ps Command

With `ps`, you can take a look at the processes your current shell session is running. It prints useful information about the programs you're running, like process ID, TTY (TeleTYpewriter), time, and command name.

```
ps
```

```
~
❯ ps
    PID TTY          TIME CMD
 533494 pts/2    00:00:00 fish
 539315 pts/2    00:00:00 ps
```

— The ps command.

In case you want something more interactive, you can use `htop`.

23. `kill` Command

It's annoying when a program is unresponsive, and you can't close it by any means. Fortunately, the `kill` command solves this kind of problem.

Simply put, `kill` sends a TERM or kill signal to a process that terminates it.

You can kill processes by entering either the PID (processes ID) or the program's binary name:

```
kill 533494

kill firefox
```

Be careful with this command — with `kill`, you run the risk of accidentally deleting the work you've been doing.

24. `ping` Command

`ping` is the most popular networking terminal utility used to test network connectivity. `ping` has a ton of options, but in most cases, you'll use it to request a domain or IP address:

```
ping google.com

ping 8.8.8.8
```

25. `vim` Command

`vim` is a free and open source terminal text editor that's in used since the '90s. It lets you edit plain text files using efficient keybindings.

Some people consider it difficult to use — exiting Vim is one of the most-viewed StackOverflow questions — but once you get used to it, it becomes your best ally in the command line.

To fire up Vim, just type:

```
vim
```

— The vim text editor.

26. `history` Command

If you’re struggling to remember a command, `history` comes in handy. This command displays an enumerated list with the commands you’ve used in the past:

```
history
```

```
  256  man type
  257  kill firefox
  258  cat old_file
  259  ping google.com
  260  ping 8.8.8.8
  261  ping -c 8 google.com
  262  ps
  263  cd
  264  ls
  265  history
[daniel@danielmanjaro ~]$
```

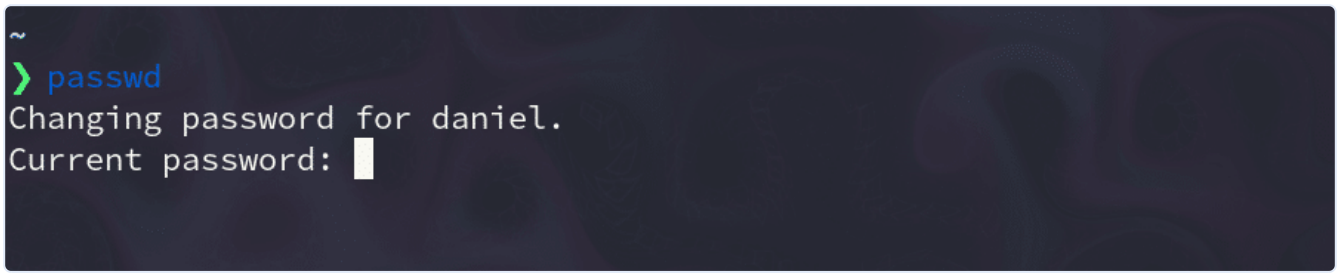
— The history command.

27. `passwd` Command

`passwd` allows you to change the passwords of user accounts. First, it prompts you to enter your current password, then asks you for a new password and confirmation.

It's similar to any other change of password you've seen elsewhere, but in this case, it's directly in your terminal:

```
passwd
```



— The passwd command

Be careful while using it — you don't want to mess up your user password!

28. `which` Command

The `which` command outputs the full path of shell commands. If it can't recognize the given command, it'll throw an error.

For example, we can use this to check the binary path for Python and the Brave web browser:

```
which python

# /usr/bin/python

which brave

# /usr/bin/brave
```

29. `shred` Command

If you ever wanted a file to be almost impossible to recover, `shred` can help you with this task. This command overrides the contents of a file repeatedly, and as a result, the given file becomes extremely difficult to recover.

Here’s a file with little content in it:

```
~/Documents/linux-commands via 🐍 v3.9.6
❯ cat file_to_shred.txt
A testing file, :))
```

— File to shred.

Now, let’s have `shred` do its thing by typing the following command:

```
shred file_to_shred.txt
```

— Overwritten content.

If you want to delete the file right away, you can use the `-u` flag:

```
shred -u file_to_shred.txt
```

30. `less` Command

`less` (opposite of more) is a program that lets you inspect files backward and forward:

```
less large_text_file.txt
```

```
Hey, you should be using less more!
Vim is the best editor, check it out.
You can start with any programming language, but do it with Python.

Hey, you should be using less more!
Vim is the best editor, check it out.
You can start with any programming language, but do it with Python.


Hey, you should be using less more!
Vim is the best editor, check it out.
You can start with any programming language, but do it with Python.


Hey, you should be using less more!
Vim is the best editor, check it out.
You can start with any programming language, but do it with Python.


Hey, you should be using less more!
Vim is the best editor, check it out.
You can start with any programming language, but do it with Python.
~
~
~
~
~
~
~
~
(END)
```

— The less command.

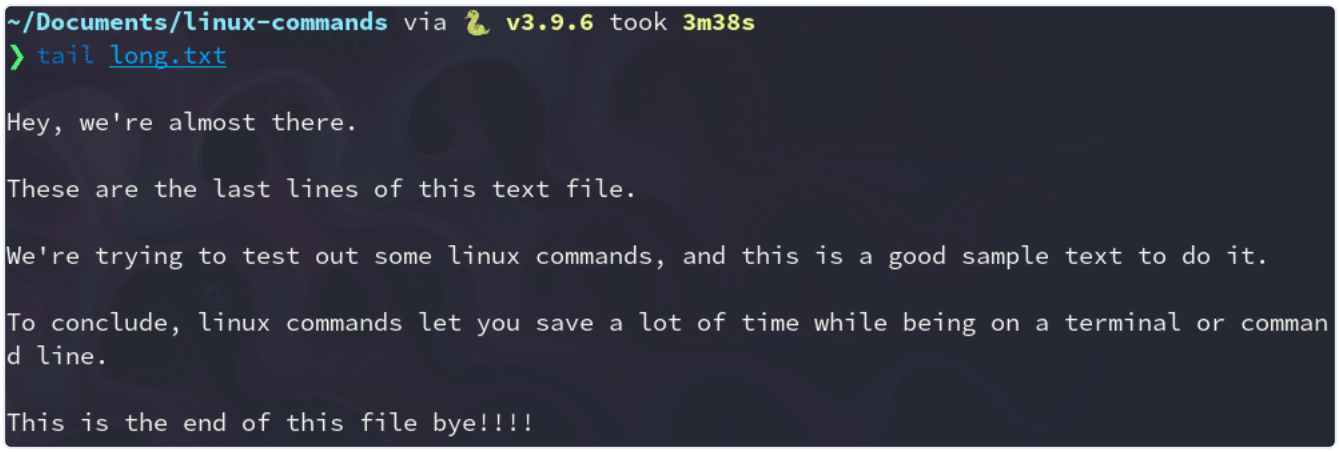
The neat thing about `less` is that it includes more and `vim` commands in its interface. If you need something more interactive than `cat`, `less` is a good option.

31. `tail` Command

Similar to `cat`, `tail` prints the contents of a file with one major caveat: It only outputs the last lines. By default, it prints the last 10 lines, but you can modify that number with `-n`.

For example, to print the last lines of a large text file, you'd use:

```
tail long.txt
```



— The tail command.

To view only the last four lines:

```
tail -n 4 long.txt
```

```
~/Documents/linux-commands via 🐍 v3.9.6
❯ tail -n 4 long.txt

To conclude, linux commands let you save a lot of time while being on a terminal or comman
d line.

This is the end of this file bye!!!!
```

— tail four lines.

32. head Command

This one is complementary to the `tail` command. `head` outputs the first 10 lines of a text file, but you can set any number of lines you want to display with the `-n` flag:

```
head long.txt

head -n 5 long.txt
```

```
~/Documents/linux-commands via 🐍 v3.9.6
❯ head long.txt
Beggining of this large file!

Here goes a ton of content.
Here goes a ton of content.
Here goes a ton of content.
Here goes a ton of content.
Here goes a ton of content.
Here goes a ton of content.
Here goes a ton of content.
Here goes a ton of content.

~/Documents/linux-commands via 🐍 v3.9.6
❯ head -n 5 long.txt
Beggining of this large file!

Here goes a ton of content.
Here goes a ton of content.
Here goes a ton of content.
```

— The head command.

33. `grep` Command

Grep is one of the most powerful utilities for working with text files. It searches for lines that match a regular expression and print them:

```
grep "linux" long.txt
```

```
~/Documents/linux-commands via 🐍 v3.9.6
❯ grep "linux" long.txt
We're trying to test out some linux commands, and this is a good sample text to do it.
To conclude, linux commands let you save a lot of time while being on a terminal or comman
d line.
```

— The grep command.

You can count the number of times the pattern repeats by using the `-c` flag:

```
grep -c "linux" long.txt

# 2
```

34. `whoami` Command

The `whoami` command (short for “who am i”) displays the username currently in use:

```
whoami

# kinsta
```

You would get the same result by using `echo` and the environmental variable $USER:

```
echo $USER

# kinsta
```

35. `whatis` Command

`whatis` prints a single-line description of any other command, making it a helpful reference:

```
whatis python

# python (1) - an interpreted, interactive, object-oriented programming la

whatis whatis

# whatis (1) - display one-line manual page descriptions
```

36. `wc` Command

Wc stands for “word count,” and as the name suggests, it returns the number of words in a text file:

```
wc long.txt

# 37 207 1000 long.txt
```

Let’s breakdown the output of this command:

- 37 lines
- 207 words
- 1000 byte-size
- The name of the file (long.txt)

If you only need the number of words, use the `-w` flag:

```
wc -w long.txt

207 long.txt
```

37. uname Command

`uname`(short for “Unix name”) prints the operative system information, which comes in handy when you know your current Linux version.

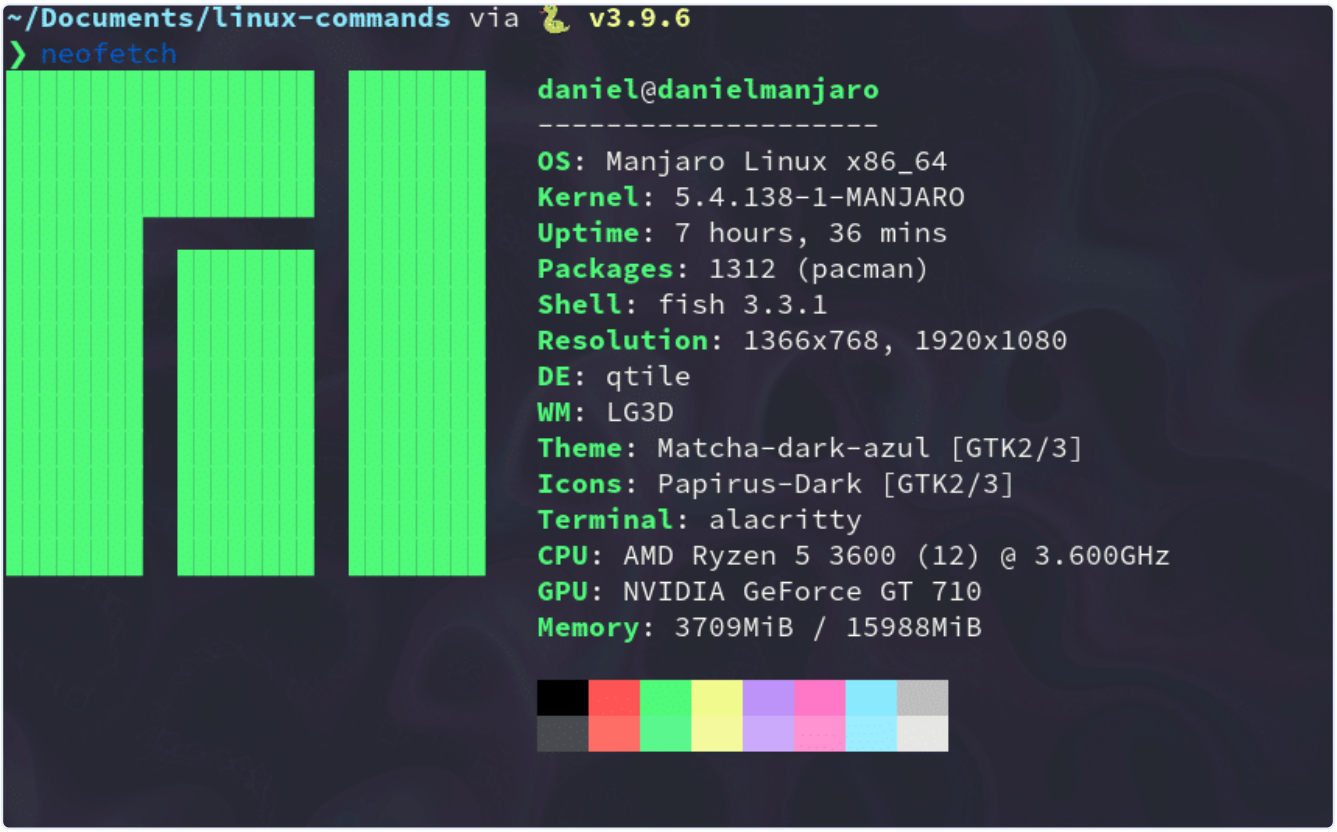
Most of the time, you’ll be using the `-a` (–all) flag, since the default output isn’t that useful:

```
uname

# Linux

uname -a

# Linux kinstamanjaro 5.4.138-1-MANJARO #1 SMP PREEMPT Thu Aug 5 12:15:21
```

38. neofetch Command

Neofetch is a CLI (command-line interface) tool that displays information about your system — like kernel version, shell, and hardware — next to an ASCII logo of your Linux distro:

```
neofetch
```



— The neofetch command.

In most machines, this command isn't available by default, so make sure to install it with your package manager first.

39. `find` Command

The `find` command searches for files in a directory hierarchy based on a regex expression. To use it, follow the syntax below:

```
find [flags] [path] -name [expression]
```

To search for a file named **long.txt** in the current directory, enter this:

```
find ./ -name "long.txt" # ./long.txt
```

To search for files that end with a **.py** (Python) extension, you can use the following command:

```
find ./ -type f -name "*.py" ./get_keys.py ./github_automation.py ./binar
```

40. `wget` Command

`wget` (World Wide Web get) is a utility to retrieve content from the internet. It has one of the largest collections of flags out there.

Here's how you would download a Python file from a GitHub repo:

```
wget https://raw.githubusercontent.com/DaniDiazTech/Object-Oriented-Progra
```

Linux Commands Cheat Sheet

Whenever you want a quick reference, just review the below table:

Command	Usage
`ls`	Lists the content of a directory
`alias`	Define or display aliases
`unalias`	Remove `alias` definitions
`pwd`	Prints the working directory
`cd`	Changes directory
`cp`	Copies files and directories
`rm`	Remove files and directories
`mv`	Moves (renames) files and directories
`mkdir`	Creates directories
`man`	Displays manual page of other commands
`touch`	Creates empty files
`chmod`	Changes file permissions
`./`	Runs an executable
`exit`	Exits the current shell session
`sudo`	Executes commands as superuser
`shutdown`	Shutdowns your machine
`htop`	Displays processes and resources information
`unzip`	Extracts compressed ZIP files
`apt`, `yum`, `pacman`	Package managers
`echo`	Displays lines of text
`cat`	Prints file contents
`ps`	Reports shell processes status
`kill`	Terminates programs
`ping`	Tests network connectivity
`vim`	Efficient text editing
`history`	Shows a list of previous commands
`passwd`	Changes user password
`which`	Returns the full binary path of a program
`shred`	Overwrites a file to hide its contents

Command	Usage
`less`	Inspects files interactively
`tail`	Displays last lines of a file
`head`	Displays first lines of a file
`grep`	Prints lines that match patterns
`whoami`	Outputs username
`whatis`	Shows single-line descriptions
`wc`	Word count files
`uname`	Displays OS information
`neofetch`	Displays OS and hardware information
`find`	Searches for files that follow a pattern
`wget`	Retrieves files from the internet

Summary

It can take some time to learn Linux, but once you master some of its tools, it becomes your best ally, and you won't regret choosing it as your daily driver.

One of the remarkable things about Linux is that even if you're an experienced user, you'll never stop learning to be more productive using it.

There are a lot more helpful Linux commands. If we've left something out, please share your favorite Linux commands in the comments below!

Linux Commands FAQ

What Is the Basic Command of Linux?

There's actually a series of basic commands that are perfect for anyone who is getting started with Linux:

- **pwd** (Prints the working directory)
- **cat** (Prints file contents)
- **cp** (Copies files and directories)
- **mv** (Moves and renames files and directories)
- **rm** (Remove files and directories)
- **touch** (Creates empty files)
- **mkdir** (Creates directories)

How Many Commands Does Linux Have?

There are thousands of commands (and new ones are being written daily). But don't worry: there's no need to remember any of them. You can always search for them online.

Can You Teach Yourself Linux?

It's possible. You can find solid resources online to help you get started. But if you feel in need of a hand, here are some well-recommended courses:

- Linux Mastery
- The Linux Command Line Bootcamp
- Learn The Linux Command Line (free)