

✓ Linux Tutorial

[+ Linux Tutorial](#)

[+ What is Linux](#)

[+ Advantages of Linux](#)

[+ Linux History](#)

[+ Linux Features](#)

[+ Linux Bash](#)

[+ Unix vs Linux](#)

[+ Linux vs Windows](#)

[+ Linux Distributions](#)

[+ Linux Licensing](#)

[+ Set Environment Variable](#)

[+ Linux Set Command](#)

[+ Linux export Command](#)

✓ Linux Directories

[+ Linux Directories](#)

[+ Linux Home Directory](#)

[+ Linux pwd](#)

[+ Linux cd](#)

[+ Linux Absolute Relative](#)

[+ Linux Path Completion](#)

[+ Linux ls](#)

[+ Linux mkdir](#)

[+ Linux rmdir](#)

[+ Rename Folder in Linux](#)

✓ Linux Commands

Advantages of Linux

← Prev

Next →

Linux is an open-source operating system like Windows and MacOS. It is not just limited to the operating system, but nowadays, it is also used as a platform to run **desktops, servers, and embedded systems**. It provides various distributions and variations as it is open source and has a modular design. The **kernel** is a core part of the Linux system.

Linux system is used to manage various services such as **process scheduling, application scheduling, basic peripheral devices, file system, and more**. Linux provides various advantages over other operating systems such as Windows and macOS. So, it is used in almost every field, from cars to home appliances and smartphones to servers (supercomputers).

In this section, we will see some major advantages of the Linux system. Further, we will see the advantages of Linux over other operating systems and will determine why it is better than other operating systems.

Why is Linux better than other operating systems?

There are many features of the Linux operating system that demonstrate that it is better than other operating systems. However, in some prospective other operating systems can be more useful than Linux.

Let's see the top 20 advantages of Linux OS.

Top 20 Advantages of Linux

Following are top 20 advantages of the Linux operating system:

+ New chat

❑ Advantages o  

❑ Linux Development

❑ Pull-Up/Down Resist

❑ Processor mode swit

—
Clear conversations

❑ Upgrade to Plus NEW

⚙ Settings

✉ Get help

[→ Log out

1. Cost-effective: Linux is free to download, install and use. This makes it a cost-effective option for individuals and businesses alike.
2. Customizability: Linux is highly customizable and can be modified to meet specific needs. This makes it an ideal choice for developers and IT professionals.
3. Stability and Reliability: Linux is known for its stability and reliability. It can run for months or even years without needing to be rebooted.
4. Security: Linux is inherently more secure than other operating systems due to its open-source nature. This means that vulnerabilities can be identified and addressed quickly.
5. Compatibility: Linux can run on a wide range of hardware, including older systems that may not be able to support newer operating systems.
6. Community Support: Linux has a large and supportive community of developers and users who are constantly working to improve the operating system.
7. Performance: Linux is known for its high performance and ability to handle large workloads efficiently.

Stop generating

Overall, Linux is a versatile and reliable operating system that offers a range of benefits for

Send a message...



...

+ New chat

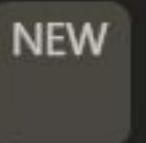
☐ Advantages o  

☐ Linux Development

☐ Pull-Up/Down Resist

☐ Processor mode swit

—
☐ Clear conversations

👤 Upgrade to 
Plus

⚙️ Settings

✉️ Get help

[→ Log out

1. Complexity: Linux can be more difficult to use than other operating systems, especially for beginners. This is due to its command-line interface and the need for technical knowledge to perform certain tasks.
2. Limited Software Support: Although Linux has a large selection of open-source software, some proprietary software applications are not available on Linux. This may be a concern for users who rely on specific software applications.
3. Hardware Compatibility: While Linux can run on a wide range of hardware, some devices may not have drivers or may require additional configuration to work properly.
4. Fragmentation: There are many different distributions of Linux, which can cause fragmentation in terms of software compatibility and user support.
5. Lack of Standardization: Unlike other operating systems that have strict standards for interface design and behavior, Linux does not have a universal standard. This can lead to inconsistency in user experience across different distributions.
6. Learning Curve: Linux can be more difficult to learn than other operating systems, especially for users who are used to a different operating system. This can make it harder for some users

Send a message...



+ New chat

□ Size vs. Length

□ Finding smallest/larg

□ I2C advantages

□ I2C Communication

Clear conversations

Upgrade to Plus NEW

Settings

Get help

[→ Log out

2. User Interface: Linux has a variety of user interfaces available, including command-line interfaces and graphical interfaces, while Windows primarily uses a graphical user interface.
3. Customization: Linux is highly customizable and can be modified to meet specific needs, while Windows is less customizable and more limited in terms of customization options.
4. Compatibility: Windows has wider software compatibility with more commercial applications, while Linux has more open-source software options and may require additional configuration to run some proprietary software.
5. Security: Linux is known for its strong security features, including built-in firewalls and permissions-based file systems, while Windows is more susceptible to malware attacks.
6. Hardware Support: Linux has better support for older hardware, while Windows tends to support newer hardware better.
7. Command Line: Linux makes greater use of the command line interface, which is more powerful and flexible than the Windows command line.
8. Community Support: Linux has a large and supportive community of developers and users who work to constantly improve the operating system, while Windows has a large user base and commercial support from Microsoft.

⟳ Regenerate response

Send a message...



- 1. Open-source: Linux is an open-source operating system, which means that its source code is freely available and can be modified by anyone.
- 2. Customizability: Linux is highly customizable and can be modified to meet specific needs. Users can choose from a variety of different desktop environments, window managers, and other software.
- 3. Security: Linux is known for its strong security features, including built-in firewalls, permissions-based file systems, and regular security updates.
- 4. Stability: Linux is stable and reliable, and can run for months or even years without needing to be rebooted.
- 5. Compatibility: Linux can run on a wide range of hardware, including older systems that may not be able to support newer operating systems.
- 6. Command-line Interface: Linux provides a powerful command-line interface that allows users to perform a wide range of tasks quickly and efficiently.
- 7. Software Management: Linux has a built-in package management system that makes it easy to install, update, and remove software.
- 8. Multi-user: Linux is designed to be a multi-user operating system, which means that multiple users can log in to the same system at the same time and use it simultaneously.

Send a message...



+ New chat



□ Size vs. Length

□ Finding smallest/larg

□ I2C advantages

□ I2C Communication

廻 Clear conversations

👤 Upgrade to Plus NEW

⚙️ Settings

↗️ Get help

[→] Log out

+ New chat

□ Size vs. Length

□ Finding smallest/larg

□ I2C advantages

□ I2C Communication

✉ Clear conversations

👤 Upgrade to Plus NEW

⚙ Settings

↗ Get help

[→] Log out

3. Security: Linux is known for its strong security features, including built-in firewalls, permissions-based file systems, and regular security updates.
4. Stability: Linux is stable and reliable, and can run for months or even years without needing to be rebooted.
5. Compatibility: Linux can run on a wide range of hardware, including older systems that may not be able to support newer operating systems.
6. Command-line Interface: Linux provides a powerful command-line interface that allows users to perform a wide range of tasks quickly and efficiently.
7. Software Management: Linux has a built-in package management system that makes it easy to install, update, and remove software.
8. Multi-user: Linux is designed to be a multi-user operating system, which means that multiple users can log in to the same system at the same time and use it simultaneously.
9. Virtualization: Linux provides powerful virtualization capabilities, allowing users to run multiple operating systems simultaneously on the same hardware.

Overall, Linux is a versatile and powerful operating system with a wide range of features that make it popular among developers, IT professionals, and casual users alike.

Send a message...

+ New chat

□ Size vs. Length

□ Finding smallest/larg

□ I2C advantages

□ I2C Communication

✉ Clear conversations

👤 Upgrade to Plus NEW

⚙ Settings

↗ Get help

[→ Log out

1. Bash stands for Bourne-Again SHell. It is an improved version of the original Unix shell, the Bourne shell (sh).
2. Bash is the default shell in most Linux distributions, and it is used to execute commands and run scripts.
3. Bash provides a powerful command-line interface that allows users to interact with the system and perform a wide range of tasks, including file management, process management, and system administration.
4. Bash supports command-line editing and history, which makes it easy to navigate and reuse commands.
5. Bash supports scripting, which allows users to automate tasks and create complex programs.
6. Bash provides a variety of built-in commands and utilities, and it also supports the use of external programs.
7. Bash is highly customizable, and users can configure their own aliases, functions, and environment variables.
8. Bash is open-source software, and its source code is freely available for modification and distribution.

⟳ Regenerate response

Send a message...

1. History: Unix was developed in the 1970s by AT&T Bell Labs, while Linux was developed in the 1990s by Linus Torvalds.
2. Licensing: Unix is a proprietary operating system, while Linux is an open-source operating system that is freely available and can be modified by anyone.
3. Architecture: Unix and Linux have similar architectures, with both using a monolithic kernel. However, Unix typically uses a System V-style init system, while Linux uses a variety of init systems, including System V, Upstart, and systemd.
4. Hardware Support: Unix is typically used on commercial hardware platforms, while Linux is used on a wide range of hardware, including desktops, servers, and embedded systems.
5. Command Line: Unix and Linux both use the command line interface, with Unix typically using the Bourne shell (sh) and Linux typically using the Bash shell.
6. Filesystems: Unix typically uses the UFS filesystem, while Linux uses a variety of filesystems, including ext4, Btrfs, and XFS.
7. User Interface: Unix typically uses a command-line interface, while Linux has a variety of graphical user interfaces available.
8. Availability: Unix is typically used in commercial environments and is less widely available than Linux, which is freely available and widely used in both commercial and non-commercial environments.

Regenerate response

+ New chat

☐ Size vs. Length

☐ Finding smallest/larg

☐ I2C advantages

☐ I2C Communication

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

Send a message...

Linux Bash - javatpoint

Advantages of Linux.

https://www.javatpoint.com/linux-bash

Home Linux C Java Python Shell Scripting SQL HTML CSS JavaScript XML Ajax Android Projects Interview Q

All the built-in command of the shell is available in Bash; moreover, it facilitates us with many other features. Some key features of Bash are as follows:

- **Shell Syntax:** The shell syntax contains **shell operations**, **quoting**, and **comments**. The shell operations are the basic operation of the shell. Quoting allows how to remove the special meaning from characters, and comments are meant to specify the comments.
- **Shell commands:** Shell commands are the types of commands that you can execute. These commands can be simple commands, pipelines, lists, compound commands, and more.
- **Shell Functions:** Shell functions are used to group commands by name. They are executed as traditional commands. When we use the name of a shell function, the list of commands associated with that is executed.
- **Shell parameters:** Basically, a parameter is an entity that stores value; it can be a name, number, or special character. The shell parameters specify how the shell store value. They can be a positional parameter or a special parameter. Positional parameters are the shell's command-line arguments, and the special parameters are denoted by a special character.
- **Shell Expansions:** Shell expansion is a technique that is used by Bash to expand the parameters. Expansion is performed on the command line after the input has been splitted into tokens.
- **Redirections:** It is a way to manage and control the input and output.
- **Command execution:** It decides how the system will react when we execute a command.
- **Shell Scripts:** It is a text file that has shell commands and executes them when it is used. Bash reads and executes the commands then exits.

↑ SCROLL TO TOP

In Linux, the "set" command is a built-in command that is used to set and modify shell options and environment variables. Here are some of the common uses of the "set" command:

1. View current settings: You can use "set" command without any options to view the current shell options and environment variables.
2. Set options: You can use the "set" command with options to modify the behavior of the shell. For example, "set -o noclobber" disables the overwriting of existing files with output redirection.
3. Set environment variables: You can use the "set" command to set environment variables that are used by the shell and other programs. For example, "set MYVAR=value" sets the environment variable MYVAR to the value "value".
4. Unset variables: You can use the "set" command to unset environment variables that are no longer needed. For example, "set -u MYVAR" unsets the variable MYVAR.
5. Set default shell options: You can use the "set" command with the "-o" option to set default shell options that will be used every time a new shell is started. For example, "set -o vi" sets the default shell to use the vi editor mode.
6. Set positional parameters: You can use the "set" command to set positional parameters that

⟳ Regenerate response

Send a message...

+ New chat

☐ Size vs. Length

☐ Finding smallest/larg

☐ I2C advantages

☐ I2C Communication

✉ Clear conversations

👤 Upgrade to NEW
Plus

⚙ Settings

↗ Get help

[→] Log out

+ New chat

□ Size vs. Length

□ Finding smallest/larg

□ I2C advantages

□ I2C Communication

✉ Clear conversations

👤 Upgrade to
Plus

NEW

⚙️ Settings

↗ Get help

[→ Log out

3. Set environment variables: You can use the "set" command to set environment variables that are used by the shell and other programs. For example, "set MYVAR=value" sets the environment variable MYVAR to the value "value".
4. Unset variables: You can use the "set" command to unset environment variables that are no longer needed. For example, "set -u MYVAR" unsets the variable MYVAR.
5. Set default shell options: You can use the "set" command with the "-o" option to set default shell options that will be used every time a new shell is started. For example, "set -o vi" sets the default shell to use the vi editor mode.
6. Set positional parameters: You can use the "set" command to set positional parameters that are used by shell scripts. For example, "set -- arg1 arg2 arg3" sets the positional parameters to "arg1", "arg2", and "arg3".

Overall, the "set" command is a powerful tool that can be used to modify the behavior of the shell and set environment variables. It is commonly used in shell scripts to automate tasks and configure the environment.

⟳ Regenerate response

Send a message...

- [+ What is Linux](#)
- [+ Advantages of Linux](#)
- [+ Linux History](#)
- [+ Linux Features](#)
- [+ Linux Bash](#)
- [+ Unix vs Linux](#)
- [+ Linux vs Windows](#)
- [+ Linux Distributions](#)
- [+ Linux Licensing](#)
- [+ Set Environment Variable](#)
- [+ Linux Set Command](#)
- [+ Linux export Command](#)

✓ Linux Directories

- [+ Linux Directories](#)
- [+ Linux Home Directory](#)
- [+ Linux pwd](#)
- [+ Linux cd](#)
- [+ Linux Absolute Relative](#)
- [+ Linux Path Completion](#)
- [+ Linux ls](#)
- [+ Linux mkdir](#)
- [+ Linux rmdir](#)
- [+ Rename Folder in Linux](#)

✓ Linux Commands

- [+ Linux commands List](#)
- [+ Commands with Example](#)

✓ Linux Users

Linux set command is used to set and unset certain flags or settings within the shell environment. These flags and settings determine the behavior of a defined script and help in executing the tasks without facing any issue. The values of shell attributes and parameters can be changed or displayed by using the set command.

Syntax:

```
set [options]
```

To know about how to set the environment variable in Linux, visit [Here](#).

Options: The supported options by the set command are as following:

- **-a:** It is used to mark variables that are modified or created for export.
- **-b:** It is used to notify of job termination immediately.
- **-e:** It is used to exit immediately if a command exits with a non-zero status.
- **-f:** It is used to disable the file name generation (globbing).
- **-h:** It is used to save the location of commands where they looked up.
- **-k:** It is used to place all assignment arguments in the environment variable of a command, except those that precede the command name.
- **-m:** It is used to enable Job control.
- **-n:** It is used to read commands.
- **-o:** It is used for option-name.

1. /: The root directory is the top-level directory in the file system hierarchy. All other directories are located beneath it.
2. /bin: This directory contains essential system binaries that are required for booting and basic system functionality, such as ls, cp, mv, and rm.
3. /etc: This directory contains system configuration files, including network settings, user accounts, and startup scripts.
4. /home: This directory contains the home directories for each user on the system. Each user has their own subdirectory within /home.
5. /lib: This directory contains shared libraries that are required by system binaries and other programs.
6. /mnt: This directory is used for temporarily mounting filesystems, such as removable storage devices or network shares.
7. /opt: This directory is used for installing additional software packages that are not part of the core system.
8. /usr: This directory contains user-related files, including system binaries, libraries, and documentation.
9. /var: This directory contains variable files, such as system logs, mail spools, and temporary

⟳ Regenerate response

Send a message...

+ New chat

☐ Size vs. Length

☐ Finding smallest/largest

☐ I2C advantages

☐ I2C Communication

✉ Clear conversations

👤 Upgrade to NEW
Plus

⚙ Settings

↗ Get help

[→ Log out

+ New chat

□ Size vs. Length

□ Finding smallest/larg

□ I2C advantages

□ I2C Communication

Clear conversations

Upgrade to
Plus NEW

⚙ Settings

↗ Get help

[→ Log out

system functionality, such as ls, cp, mv, and rm.

3. /etc: This directory contains system configuration files, including network settings, user accounts, and startup scripts.
4. /home: This directory contains the home directories for each user on the system. Each user has their own subdirectory within /home.
5. /lib: This directory contains shared libraries that are required by system binaries and other programs.
6. /mnt: This directory is used for temporarily mounting filesystems, such as removable storage devices or network shares.
7. /opt: This directory is used for installing additional software packages that are not part of the core system.
8. /usr: This directory contains user-related files, including system binaries, libraries, and documentation.
9. /var: This directory contains variable files, such as system logs, mail spools, and temporary files.

⟳ Regenerate response

These directories provide a standardized structure for the file system, which makes it easier to...

Send a message...



has their own subdirectory within /home.

+ New chat

□ Size vs. Length

□ Finding smallest/larg

□ I2C advantages

□ I2C Communication

✉ Clear conversations

👤 Upgrade to Plus NEW

⚙ Settings

↗ Get help

[→ Log out

5. /lib: This directory contains shared libraries that are required by system binaries and other programs.
6. /mnt: This directory is used for temporarily mounting filesystems, such as removable storage devices or network shares.
7. /opt: This directory is used for installing additional software packages that are not part of the core system.
8. /usr: This directory contains user-related files, including system binaries, libraries, and documentation.
9. /var: This directory contains variable files, such as system logs, mail spools, and temporary files.

These directories provide a standardized structure for the file system, which makes it easier to manage and navigate. They are typically created during the installation of the operating system, and can be modified or extended as needed.

⟳ Regenerate response

Send a message...



Linux current directory (cwd) — Mozilla Firefox

Linux Files - javatpoint X Linux current directory (cwd) +

https://chat.openai.com/chat

+ New chat

Linux current ✎ 🗑️

Advantages of Linux

Linux Development

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

Send a message...

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts

1. Regular files: These are the most common type of files in Linux. They contain data in various formats, such as text, images, audio, video, and program code.
2. Directories: Directories are used to organize files into a hierarchical structure. They can contain other directories and files, and provide a way to group related files together.
3. Symbolic links: Symbolic links, also known as soft links, are special files that point to other files or directories. They provide a way to create aliases for files or directories and are commonly used to create shortcuts or to reference files in different locations.
4. Device files: Device files are used to interact with hardware devices, such as hard disks, printers, and network interfaces. There are two types of device files: character devices and block devices.
5. Named pipes: Named pipes, also known as FIFOs, are used for interprocess communication. They provide a way for two processes to communicate with each other by reading and writing to a common pipe.

Linux current directory (cwd) — Mozilla Firefox

Linux Files - javatpoint X Linux current directory (cwd) +

https://chat.openai.com/chat

+ New chat

Linux current

Advantages of Linux

Linux Development

Clear conversations

Upgrade to Plus NEW

Settings

Get help

[→ Log out

Send a message...

In Linux, files are managed using a variety of commands and tools, such as ls (list files), cp (copy)

Regenerate response

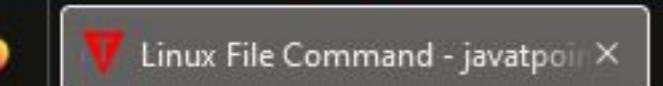
3. Symbolic links: Symbolic links, also known as soft links, are special files that point to other files or directories. They provide a way to create aliases for files or directories and are commonly used to create shortcuts or to reference files in different locations.

4. Device files: Device files are used to interact with hardware devices, such as hard disks, printers, and network interfaces. There are two types of device files: character devices and block devices.

5. Named pipes: Named pipes, also known as FIFOs, are used for interprocess communication. They provide a way for two processes to communicate with each other by reading and writing to a common pipe.

6. Sockets: Sockets are used for network communication between processes on different hosts. They provide a way for processes to exchange data over a network using a standard protocol, such as TCP or UDP.

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts



✓ 100+ Linux Commands

✓ Linux Interview

Note: File command tell us the file type with the help of a magic file that contains all the patterns to recognize a file type. Path of magic file is /usr/share/file/magic. For more information enter the command 'man 5 magic'.

Linux File Command Options

Option	Function
file -s	Used for special files.
file *	Used to list types of all the files.
file /directory name/*	Used to list types of all the files from mentioned directory.
file [range]*	It will list out all the files starting from the alphabet present within the given range.

Next Topic

Linux File Touch

Linux current directory (cwd) — Mozilla Firefox

Linux File Command - javatpoi X Linux current directory (cwd) +

https://chat.openai.com/chat

1. Set specific timestamp. You can also use the `touch` command to set the modification and access times of a file to a specific timestamp. To do this, use the `-t` option followed by a timestamp in the format `YYYYMMDDhhmm.ss`. For example:

bash

```
touch -t 2022011200.00 myfile.txt
```

Copy code

This will set the modification and access times of `myfile.txt` to January 1, 2022 at 12:00:00 PM.

By using the `touch` command, you can quickly create empty files or update the modification and access times of existing files.

Regenerate response

Send a message...

Get help Log out ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts



man 2 passwd

man Options

Commands	Function
man -aw	List all available sections of a command.
man -a	To view all man pages of a command.
sman -k (apropos)	Shows a list of results in man page containing a keyword match.
-f, whatis	It displays description from manual page if available.
whereis	Used to determine location of a man page

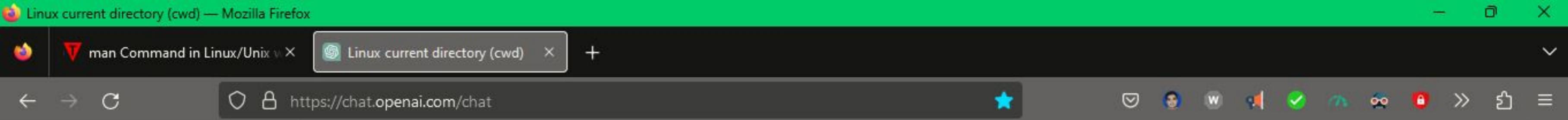
Next Topic

Linux-man-a

← Prev

Next →





+ New chat

Linux current

Advantages of Linux

Linux Development

Clear conversations

Upgrade to Plus

Settings

Get help

[→] Log out

This will display all available manual pages for the `ls` command, which might include pages in different sections of the manual, such as section 1 and section 1p.

The `-a` option is useful when you're not sure which section of the manual a particular topic or command is located in, or if you want to see all available documentation for a command or topic.

Keep in mind that not all commands or topics will have multiple manual pages available, so using the `-a` option may not always be necessary or helpful.

Regenerate response

Send a message...

We'll discuss about the following file contents given in the table:

Commands	Function
head	It displays the beginning of a file.
tail	It displays the last part of a file.
cat	This command is versatile and multi worker.
tac	Opposite of cat.
more	Command line displays contents in pager form that is either in more format.
less	Command line displays contents in pager form that is either in less format.

Next Topic

Open File in Linux

Next →



Linux File Contents - javatpoint X

Linux current directory (cwd) X +



https://www.javatpoint.com/linux-file-contents

[Home](#) [Linux](#) [C](#) [Java](#) [Python](#) [Shell Scripting](#) [SQL](#) [HTML](#) [CSS](#) [JavaScript](#) [XML](#) [Ajax](#) [Android](#) [Projects](#) [Interview Q](#)✓ [Introduction to Linux](#)✓ [Linux Directories](#)✓ [Linux Files](#)✓ [Linux Man Pages](#)✓ [Linux File Contents](#)↪ [Linux File Contents](#)↪ [Linux head](#)↪ [Linux tail](#)↪ [Linux cat](#)↪ [Linux tac](#)↪ [Linux more](#)↪ [Linux less](#)✓ [Linux File Hierarchy](#)✓ [12 Linux Filters](#)✓ [Linux I/O Redirection](#)✓ [Linux Unix Tools](#)✓ [Linux Regex](#)✓ [Linux Users](#)✓ [Linux File Security](#)

Linux File Contents Command

[← Prev](#)[Next →](#)

There are many commands which help to look at the contents of a file. Now we'll look at some of the commands like head, tac, cat, less & more and strings.

We'll discuss about the following file contents given in the table:

Commands	Function
head	It displays the beginning of a file.
tail	It displays the last part of a file.
cat	This command is versatile and multi worker.
tac	Opposite of cat.
more	Command line displays contents in pager form that is either in more format.
less	Command line displays contents in pager form that is either in less format.

Linux current directory (cwd) — Mozilla Firefox

Linux File Contents - javatpoint X Linux current directory (cwd) +

← → C https://chat.openai.com/chat ⭐

+ New chat

Linux current ✎

Advantages of Linux

Linux Development

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

Send a message... Regenerate response

head file1.txt file2.txt file3.txt

This will display the first 10 lines of each of the three files: `file1.txt`, `file2.txt`, and `file3.txt`.

By using the `head` command, you can quickly view the first few lines of a file without having to open the entire file. This can be useful when working with large files or when you only need to view the beginning of a file.

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts

continuously display new lines as they are added to the end of a file. This is useful for monitoring log files or other files that are being actively updated. For example:

bash

 Copy code

```
tail -f file.txt
```

This will display the last few lines of the `file.txt` file and continuously update the display as new lines are added to the file.

By using the `tail` command, you can quickly view the last few lines of a file without having to open the entire file. This can be useful when working with large files or when you only need to view the end of a file.

 Regenerate response

[Send a message...](#)



Track a file for changes

To track a file for changes, the '**-f**' option is used. Here, '**-f**' stands for the following. It is useful for monitoring log files. Execute the below command:

```
tail -f num.txt
```

The above command will monitor the file 'num.txt'. To exit from monitoring, press "CTRL+C" keys.
Consider the below output:

```
javatpoint@javatpoint-Inspiron-3542:~$ tail -f num.txt
6
7
8
9
10
11
12
13
14
15
```

Display multiple files

We can display multiple files from last at once by executing the tail command. To display the multiple files, provide the file names as input. It will display the last ten lines of specified files.

Linux current directory (cwd) — Mozilla Firefox

tail Command in Linux/Unix with Examples

Linux current directory (cwd)

<https://chat.openai.com/chat>

+ New chat

Linux current

Advantages of Linux

Linux Development

Clear conversations

Upgrade to Plus NEW

Settings

Get help

[→ Log out

tac file1.txt file2.txt > combined.txt

This will combine the contents of `file1.txt` and `file2.txt` into a new file called `combined.txt` in reverse order.

1. Display the contents of a file with line numbers in reverse order: You can use the ` -b` option along with the ` -n` option to display the contents of a file with line numbers in reverse order. For example:

bash

tac -b -n file.txt

Copy code

Regenerate response

Send a message...

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts

Home

Linux

C

Java

Python

Shell Scripting

SQL

HTML

CSS

JavaScript

XML

Ajax

Android

Projects

Interview Q

If you want to go to the last or backward page use '**b**' key.

Linux more options

Options	Function
more -num	Limits the line displayed per page.
more -d	Displays user message at right corner.
more -s	Squeeze blank lines.
more +/string name	It helps to find the string.
more +num	Used to display the content from a specific line.

Note: The 'more' command can't be used to display binary files.

Next Topic

Linux Less

Linux current directory (cwd) — Mozilla Firefox

Linux less Command - javatpoint X Linux current directory (cwd) +

https://chat.openai.com/chat

linux less command

+ New chat

Linux current

Advantages of Linux

Linux Development

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

In Linux, the `less` command is a pager similar to the `more` command that is used to display the contents of a file on the terminal one screen at a time. However, `less` provides more advanced features than `more`, such as the ability to scroll both forward and backward through a file, search for patterns in a file, and navigate to specific line numbers. Here are some common ways to use the `less` command:

1. Display the contents of a file one screen at a time: To display the contents of a file one screen at a time, simply type `less` followed by the name of the file. For example:

Copy code

Regenerate response

Send a message...

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts

Linux current directory (cwd) — Mozilla Firefox

Linux less Command - javatpoi X Linux current directory (cwd) +

← → C https://chat.openai.com/chat ⭐

+ New chat

□ Linux current

□ Advantages of Linux

□ Linux Development

Clear conversations

Upgrade to Plus NEW

⚙️ Settings

↗ Get help

[→ Log out

Send a message...

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts

1. Scrolling: `less` allows you to scroll both forward and backward through a file, while `more` only allows you to scroll forward.
2. Searching: `less` allows you to search for a pattern in a file using the `/` command, and it highlights all occurrences of the pattern. `more` also allows you to search for a pattern using the `/` command, but it only highlights the first occurrence of the pattern.
3. Navigation: `less` allows you to navigate to specific line numbers using the `:` command, while `more` does not have this feature.
4. Command prompt: `less` displays a command prompt at the bottom of the screen that allows you to enter various commands to navigate through the file, while `more` does not have this feature.

Overall, `less` is a more advanced and feature-rich pager than `more`, making it a popular choice for working with large files.

⟳ Regenerate response



Directory type	Types of files stored
Binary directories	Contains binary or compiled source code files, eg, /bin, /sbin, etc.
Configuration directories	Contains configuration files of the system, eg, /etc, /boot.
Data directories	Stores data files, eg, /home, /root, etc.
Memory directories	Stores device files which doesn't take up actual hard disk space, eg, /dev, /proc, /sys.
usr (Unix System Resources)	Contains sharable, read only data, eg, /usr/bin, /usr/lib, etc.
var (variable directory)	Contains larger size data, eg, /var/log, /var/cache, etc.
Non-standard directories	Directories which do not come under standard FHS, eg, lost+found, /run, etc.

Next Topic

Linux FHS Binary Directories

services running on the system. Some of the important configuration files in the `/etc` directory are:

1. `/etc/passwd`: This file contains user account information such as the username, user ID, home directory, and default shell.
 2. `/etc/group`: This file contains information about groups of users on the system.
 3. `/etc/fstab`: This file contains information about file systems that are mounted at boot time.
 4. `/etc/hosts`: This file contains information about the system's host names and IP addresses.
 5. `/etc/resolv.conf`: This file contains information about DNS name servers.
 6. `/etc/ssh/sshd_config`: This file contains configuration settings for the SSH server.
 7. `/etc/sudoers`: This file contains the rules that control which users can run commands as root using the `sudo` command.

Overall, the `/etc` directory is an important Regenerate respinsex system that contains

[Send a message...](#)

[←](#) [→](#) [C](#)<https://www.javatpoint.com/linux-fhs-data-directory>[Home](#)[Linux](#)[C](#)[Java](#)[Python](#)[Shell Scripting](#)[SQL](#)[HTML](#)[CSS](#)[JavaScript](#)[XML](#)[Ajax](#)[Android](#)[Projects](#)[Interview Q](#)

✓ Linux File Hierarchy

⊕ Linux File Hierarchy

Data directory contains following directories.

⊕ Binary Directories

⊕ Configuration Directories

⊕ Data Directories

⊕ Memory Directories

⊕ Usr Directory

⊕ var Directory

⊕ Non-standard directories

✓ 12 Linux Filters

✓ Linux I/O Redirection

✓ Linux Unix Tools

✓ Linux Regex

✓ Linux Users

✓ Linux File Security

✓ Linux Shell

✓ Linux Networking

✓ vi editor

✓ Linux System Admin

✓ 100+ Linux Commands

✓ Linux Interview

◦ /home

◦ /root

◦ /srv

◦ /media

◦ /mnt

◦ /tmp

/home

The '/home' directory stores users personnel files. After the '/home' there is a directory which is generally named at the user's name like we have '**/home/sssit**'. Inside this directory we have our sub-directories like Desktop, Downloads, Documents, pictures, etc.

Example:

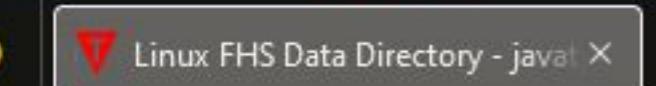
```
ls /home
```

```
ls /home/sssit
```

↑ SCROLL TO TOP



sssit@JavaPoint: ~



Linux FHS Data Directory - java X

Linux current directory (cwd) X +



https://www.javatpoint.com/linux-fhs-data-directory



Home

Linux

C

Java

Python

Shell Scripting

SQL

HTML

CSS

JavaScript

XML

Ajax

Android

Projects

Interview Q

For example, if you want to give the command `cd /sssit/Desktop` so instead of writing this you can also write "`~/Desktop`", both are same.

/root

The '/root' directory is the home directory of the root user.

Please note that '/root' directory is different from (/) root.

/srv

The term 'srv' is short for **service**. The '/srv' directory contains server specific data for services provided by the system like www, cvs, rysync, ftp, etc.

/media

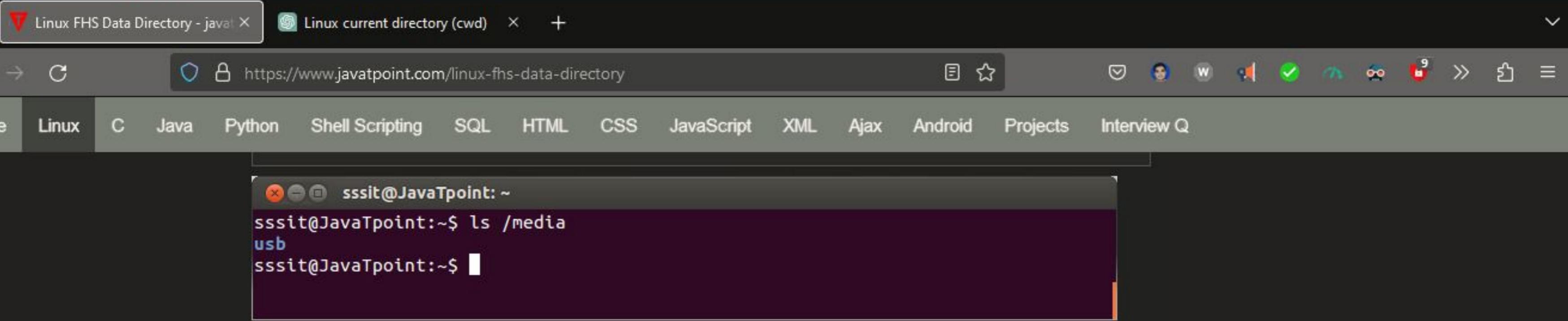
The '/media' directory acts as a mount point for removable media devices such as CD-Rom, floppy, USB devices, etc.

This is newly introduced directory and hence a system can run without this directory also.

Example:

```
ls /media
```

Linux FHS Data Directory - javatpoint — Mozilla Firefox



Linux FHS Data Directory - java X Linux current directory (cwd) +
https://www.javatpoint.com/linux-fhs-data-directory

Home Linux C Java Python Shell Scripting SQL HTML CSS JavaScript XML Ajax Android Projects Interview Q

```
sssit@JavaTpoint:~$ ls /media
usb
sssit@JavaTpoint:~$
```

Look at the below snapshot, command "**ls /media**" displays '/media' content.

/mnt

The term 'mnt' stands for **mount**. The '/mnt' directory should be empty and sysadmins can only mount temporary filesystems.

/tmp

The term 'tmp' stands for **temporary**. Data stored in '/tmp' is temporary and may use either disk space or RAM. When system is rebooted, files under this directory is automatically deleted. So it is advisable that never use '/tmp' to store important data.

[Next Topic](#)

[Linux Memory Directory](#)

Linux current directory (cwd) — Mozilla Firefox

Linux FHS Data Directory - java! X Linux current directory (cwd) +

https://chat.openai.com/chat

system is rebooted.

+ New chat

□ Linux current

□ Advantages of Linux

□ Linux Development

Clear conversations

Upgrade to Plus

Settings

Get help

[→ Log out

Send a message...

Regenerate response

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts

Linux current directory (cwd) — Mozilla Firefox

Linux FHS Memory Directory - X Linux current directory (cwd) +

https://chat.openai.com/chat

In Linux, the `/dev` directory is a special directory that contains device files, which are used to communicate with and access hardware devices connected to the system.

Device files in the `/dev` directory are used to access different types of devices, including hard drives, CD/DVD drives, USB devices, serial ports, and sound cards. Each device file is identified by a unique name that corresponds to the device it represents. For example, the device file `/dev/sda` represents the first hard drive on the system, while the device file `/dev/ttys0` represents the first serial port.

Device files in the `/dev` directory can be accessed like regular files, but they are not stored on disk. Instead, they are created dynamically by the kernel when the corresponding device is detected by the system. This allows programs to interact with hardware devices without needing to know the specific details of the device or its driver.

Regenerate response

Send a message...

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts

+ New chat

Linux current

Advantages of Linux

Linux Development

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

Linux current directory (cwd) — Mozilla Firefox

Linux FHS Memory Directory - X Linux current directory (cwd) +

← → C https://chat.openai.com/chat ⭐ ⓘ

represents the first serial port.

+ New chat

Linux current ✎

Advantages of Linux

Linux Development

Clear conversations

Upgrade to Plus NEW

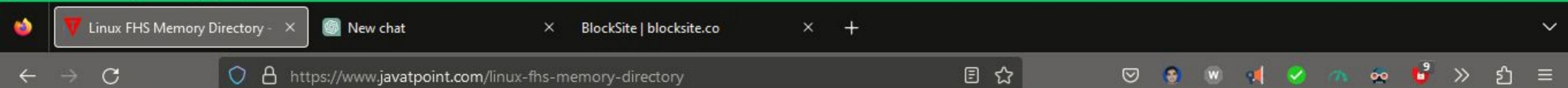
Settings

Get help

[→ Log out

Send a message... ↴ Regenerate response

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts



Home Linux C Java Python Shell Scripting SQL HTML CSS JavaScript XML Ajax Android Projects Interview Q

✓ Linux Interview back.

/proc

The term 'proc' is short for process. Same as '/dev', '/proc' also doesn't take up disk space. It contains process information. It is a pseudo filesystem that contains information about running processes. It also works as virtual filesystem containing text information about system resources.

- **/proc conversation with the kernel:** The '/proc' displays view of the kernel, what the kernel manages and it is a means to directly communicate with the kernel.

Example:

```
ls /proc
```

```
sssit@JavaTpoint:~$ ls /proc
1      1556  1691  2020  31   824      dma      pagetypeinfo
10     1557  1698  2050  314  829      dri      partitions
1108    1558  1738  2077  32   838     driver    sched_debug
1113    1561  1739  2083  33   867  execdomains  schedstat
1122    1563  1740  2099  355  907      fb      scsi
1139    1573  1742   21    41   912  filesystems  self
12      1579  1766   22    458  925      fs      slabinfo
1252   1581  1778   226   46   926  interrupts  softirqs
13      1582  1780   227   467  930      iomem  stat
```

Linux FHS unix system resources - javatpoint — Mozilla Firefox

Linux FHS unix system resource X New chat BlockSite | blocksite.co X +

https://www.javatpoint.com/linux-unix-system-resources

java point

Home Linux C Java Python Shell Scripting SQL HTML CSS JavaScript XML Ajax Android Projects Interview Q

- ✓ Introduction to Linux
- ✓ Linux Directories
- ✓ Linux Files
- ✓ Linux Man Pages
- ✓ Linux File Contents
- ✓ Linux File Hierarchy
- ↪ Linux File Hierarchy
- ↪ Binary Directories
- ↪ Configuration Directories
- ↪ Data Directories
- ↪ Memory Directories
- ↪ Usr Directory
- ↪ var Directory
- ↪ Non-standard directories
- ✓ 12 Linux Filters
- ✓ Linux I/O Redirection
- ✓ Linux Unix Tools
- ✓ Linux Regex
- ✓ Linux Users

Unix System Resources (/usr)

← Prev

Next →

Although it is pronounced as user but in actual it stands for **Unix System Resources**. It is also called secondary hierarchy as it contains binaries, libraries, documentation for all the user applications. It only contains shareable read-only data.

Example:

```
ls /usr
sssit@JavaPoint: ~
sssit@JavaPoint:~$ ls /usr
bin games include lib local sbin share src
sssit@JavaPoint:~$
```

Look the above snapshot, command "**ls /usr**" displays the '/usr' directories.

We'll explain some of the /usr sub-directories:

- ✓ Introduction to Linux
- ✓ Linux Directories
- ✓ Linux Files
- ✓ Linux Man Pages
- ✓ Linux File Contents
- ✓ Linux File Hierarchy
 - ↪ Linux File Hierarchy
 - ↪ Binary Directories
 - ↪ Configuration Directories
 - ↪ Data Directories
 - ↪ Memory Directories
 - ↪ Usr Directory
 - ↪ var Directory
 - ↪ Non-standard directories
- ✓ 12 Linux Filters
- ✓ Linux I/O Redirection
- ✓ Linux Unix Tools
- ✓ Linux Regex
- ✓ Linux Users
- ✓ Linux File Security
- ✓ Linux Shell
- ✓ Linux Networking
- ✓ vi editor

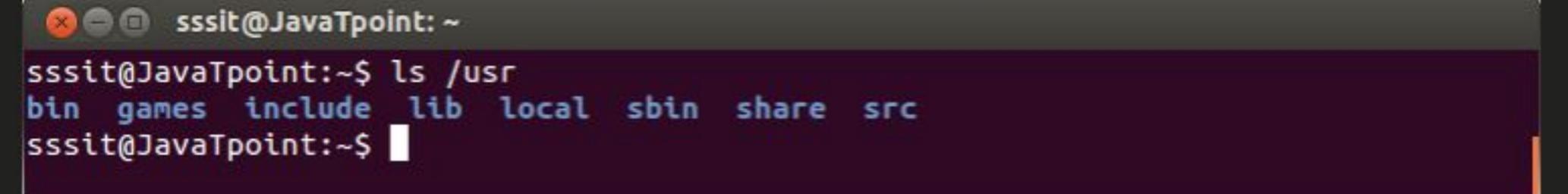
Unix System Resources (/usr)

[← Prev](#)[Next →](#)

Although it is pronounced as user but in actual it stands for **Unix System Resources**. It is also called secondary hierarchy as it contains binaries, libraries, documentation for all the user applications. It only contains shareable read-only data.

Example:

```
ls /usr
```



```
sssit@JavaPoint:~$ ls /usr
bin games include lib local sbin share src
sssit@JavaPoint:~$ █
```

Look the above snapshot, command "**ls /usr**" displays the '/usr' directories.

We'll explain some of the /usr sub-directories:

- /usr/bin
- /usr/include
- /usr/lib

+ New chat



Sure! `/usr/bin` is a directory in the Linux operating system (and other Unix-like systems) that stores executable files, or programs, that are available to all users of the system. This directory contains many of the standard Linux command-line utilities and tools that are used for various tasks, such as managing files and directories, networking, system administration, and more.



❑ /usr/bin lin li ✎ 🗑

Linux current directo

Advantages of Linux

Linux Development

 Clear conversations

Upgrade to Plus

NEW

Settings

 Get help

[→ Log out

Examples of programs that might be found in `/usr/bin` include `ls` (list files in a directory), `cp` (copy files), `mv` (move or rename files), `grep` (search for text in files), and `ssh` (secure shell client for remote login). In general, any program that is installed on the system and is intended to be run from the command line will be located in `/usr/bin` (or sometimes `/bin` or other directories within the system's `\$PATH`).

It's worth noting that some Linux distributions may organize their file systems differently, and some programs may be located in different directories depending on the specific distribution and version of Linux. However, `/usr/bin` is a very common location for executables in most Linux systems.

 Regenerate response

[Send a message...](#)

/usr/bin

The '/usr/bin' directory contains non-essential binary commands for all users. If you can't find a command in '/bin', search it in '/usr/bin'. It contains a lot of commands.

/usr/include

The '/usr/include' directory contains standard include files for C.

/usr/lib

The '/usr/lib' directory contains libraries that are not directly executed by the users. In other words, it contains binaries for the '/usr/bin' and '/usr/sbin'.

/usr/share

The '/usr/share' directory contains architecture independent (shared) data.

/usr/local

The '/usr/local' directory is used to install software locally. It means all the user programs that you'll install from source will be installed here.

/usr/src

The term 'src' is short for **source**. It is used to store source code like kernel source code with its header files.

/usr/include

The '/usr/include' directory contains standard include files for C.

/usr/lib

The '/usr/lib' directory contains libraries that are not directly executed by the users. In other words, it contains binaries for the '/usr/bin' and '/usr/sbin'.

/usr/share

The '/usr/share' directory contains architecture independent (shared) data.

/usr/local

The '/usr/local' directory is used to install software locally. It means all the user programs that you'll install from source will be installed here.

/usr/src

The term 'src' is short for **source**. It is used to store source code like kernel source code with its header files.

[Next Topic](#)[Variable Directory](#)

- ✓ [Introduction to Linux](#)
- ✓ [Linux Directories](#)
- ✓ [Linux Files](#)
- ✓ [Linux Man Pages](#)
- ✓ [Linux File Contents](#)
- ✓ [Linux File Hierarchy](#)
- ⊕ [Linux File Hierarchy](#)
- ⊕ [Binary Directories](#)
- ⊕ [Configuration Directories](#)
- ⊕ [Data Directories](#)
- ⊕ [Memory Directories](#)
- ⊕ [Usr Directory](#)
- ⊕ [var Directory](#)
- ⊕ [Non-standard directories](#)
- ✓ [12 Linux Filters](#)
- ✓ [Linux I/O Redirection](#)
- ✓ [Linux Unix Tools](#)
- ✓ [Linux Regex](#)
- ✓ [Linux Users](#)
- ✓ [Linux File Security](#)
- ✓ [Linux Shell](#)
- ✓ [Linux Networking](#)
- ✓ [vi editor](#)
- ✓ [Linux System Admin](#)
- ✓ [100+ Linux Commands](#)

[↑ SCROLL TO TOP](#)

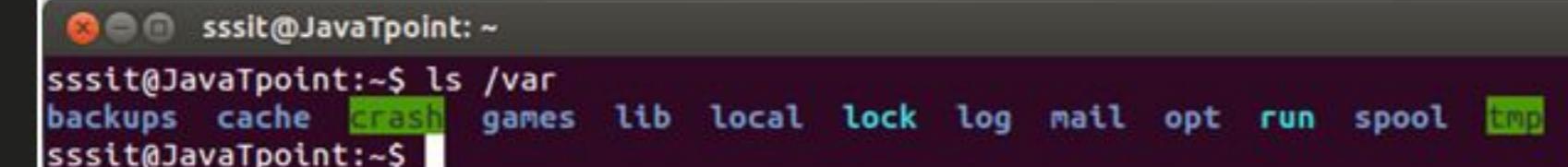
Variable Directory (/var)

[← Prev](#)[Next →](#)

The term 'var' is short for **variable**. Files that have an unexpected size and whose content is expected to change continuously (that's why it is named as variable) during normal operation of the system are stored here. For example, log files, spool files and cache files.

Example:

```
ls /var
```



```
sssit@JavaPoint:~$ ls /var
backups  cache  crash  games  lib  local  lock  log  mail  opt  run  spool  tmp
sssit@JavaPoint:~$
```

Look at the above snapshot, command "**ls /var**" displays '/var' content.

We'll explain some of the /var sub-directories here:

- [/var/log](#)
- [/var/cache](#)
- [/var/spool](#)
- [/var/lib](#)

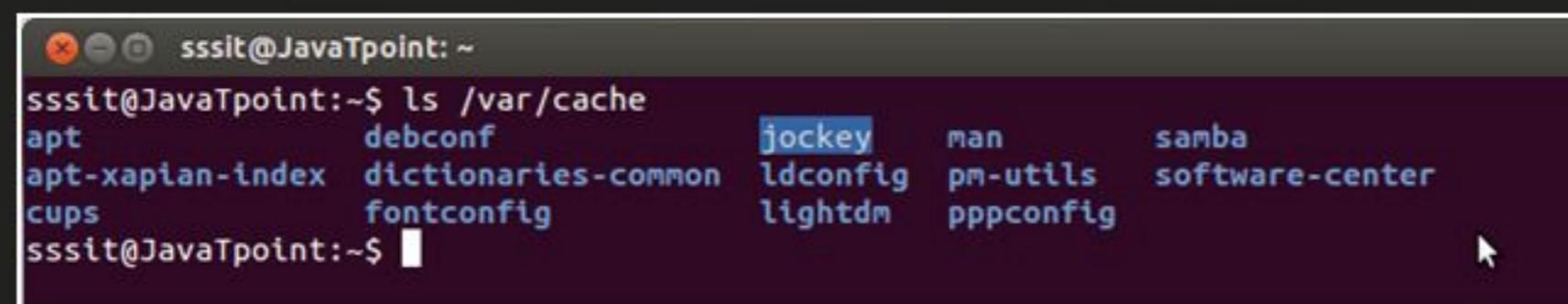
auth.log.1	dpkg.log	mail.log	upstart
auth.log.2.gz	dpkg.log.1	news	wtmp
auth.log.3.gz	faillog	pm-powersave.log	wtmp.1
boot	fontconfig.log	pm-powersave.log.1	Xorg.0.log

Look at the above snapshot, command "**ls /var/log**" displays the '/var/log' content.

/var/cache

The '/var/cache' directory stores application cache data. Cache data are locally generated by I/O or calculation. Cache must be able to regenerate or restore the data. These files can be deleted without any loss of data.

Example:



```
sssit@JavaTpoint:~$ ls /var/cache
apt          debconf      jockey    man        samba
apt-xapian-index dictionaries-common ldconfig  pm-utils   software-center
cups         fontconfig    lightdm   pppconfig
sssit@JavaTpoint:~$
```

ls /var/cache

Look at the above screenshot, command "**ls /var/cache**" displays the '/var/cache' content.

/var/spool

The '/var/spool' directory is used to spool the files waiting to be processed. For example, printing queues and mail queues.

+ New chat

□ /usr/bin lin li ↻ ⚡

□ Linux current directo

□ Advantages of Linux

□ Linux Development

—
□ Clear conversations

□ Upgrade to Plus NEW

⚙ Settings

↗ Get help

[→ Log out



In Linux (and other Unix-like systems), `/var/spool` is a directory that is used for storing various types of data that are waiting to be processed. The directory is typically used for storing temporary files that are created by various system daemons and services.



Some common examples of data that might be stored in `/var/spool` include:

- Printer queues: When you print a document, it is typically sent to a print queue, which holds the document until the printer is available to process it. The print queue is often located in `/var/spool`.
- Mail queues: When an email is sent, it is typically queued up in `/var/spool/mail` or a similar directory until the email server is able to process it and deliver it to the recipient.
- Cron jobs: Cron is a scheduler program that allows you to run tasks at specific times or intervals. When a cron job is scheduled to run, it is often stored in `/var/spool/cron`.
- News articles: In some systems, Usenet news articles are stored in `/var/spool/news`.
- Batch jobs: Some systems may use `/var/spool/batch` to store batch jobs that are waiting to be processed.

⟳ Regenerate response

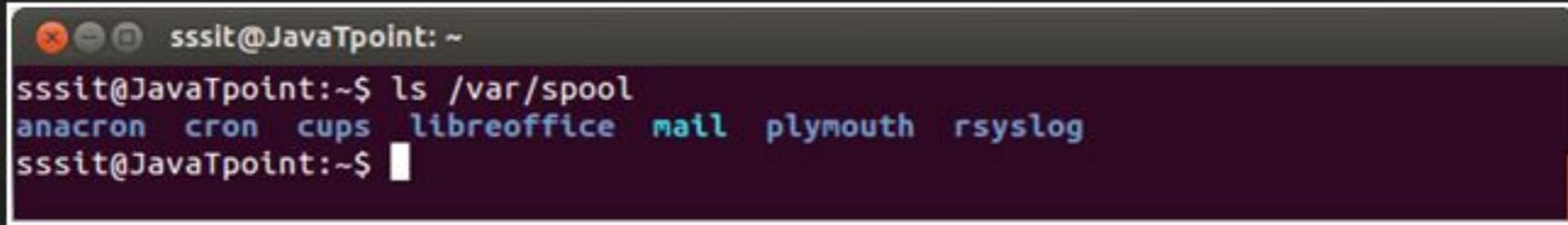
Send a message...



The '/var/spool' directory is used to spool the files waiting to be processed. For example, printing queues and mail queues.

Example:

```
ls /var/spool
```



```
sssit@JavaTpoint:~$ ls /var/spool
anacron cron cups libreoffice mail plymouth rsyslog
sssit@JavaTpoint:~$
```

Look at the above screenshot, command "**ls /var/spool**" displays the '/var/spool' content.

/var/lib

The '/var/lib' directory stores the files that contains state information like databases. File's data modifies as their respective programs run.

[Next Topic](#)[Non-Standard Directories](#)[← Prev](#)[Next →](#)

+ New chat

□ /usr/bin lin li ↻ ⚡

□ Linux current directo

□ Advantages of Linux

□ Linux Development

—
□ Clear conversations

□ Upgrade to Plus NEW

⚙ Settings

↗ Get help

[→ Log out

data that is specific to the system and its installed applications. This directory is often used for storing data files and databases that are required for the proper functioning of various system services and applications.

Some common examples of data that might be stored in `/var/lib` include:

- Package manager databases: Most Linux distributions use a package manager to manage the installation, removal, and updating of software packages. The package manager often stores its database of installed packages and dependencies in `/var/lib`.
- Application data: Many applications store their data files and settings in `/var/lib`. For example, the Firefox web browser stores its user profiles in `/var/lib/firefox`.
- Databases: Some databases, such as MySQL or PostgreSQL, may store their data files in `/var/lib`.
- System logs: Some Linux distributions store system log files in `/var/log`, while others may use `/var/lib` to store logs for specific services or applications.

Overall, `/var/lib` is a directory that is used for storing important data files and databases that

⟳ Regenerate response

Send a message...



Non-standard directories are as follows:

✓ Linux File Hierarchy

- ⊕ Linux File Hierarchy
- ⊕ Binary Directories
- ⊕ Configuration Directories
- ⊕ Data Directories
- ⊕ Memory Directories
- ⊕ Usr Directory
- ⊕ var Directory
- ⊕ Non-standard directories
- ✓ 12 Linux Filters
- ✓ Linux I/O Redirection
- ✓ Linux Unix Tools
- ✓ Linux Regex
- ✓ Linux Users
- ✓ Linux File Security
- ✓ Linux Shell
- ✓ Linux Networking
- ✓ vi editor
- ✓ Linux System Admin
- ✓ 100+ Linux Commands
- ✓ Linux Interview

- /cdrom

- /run

- /lost + found

/cdrom

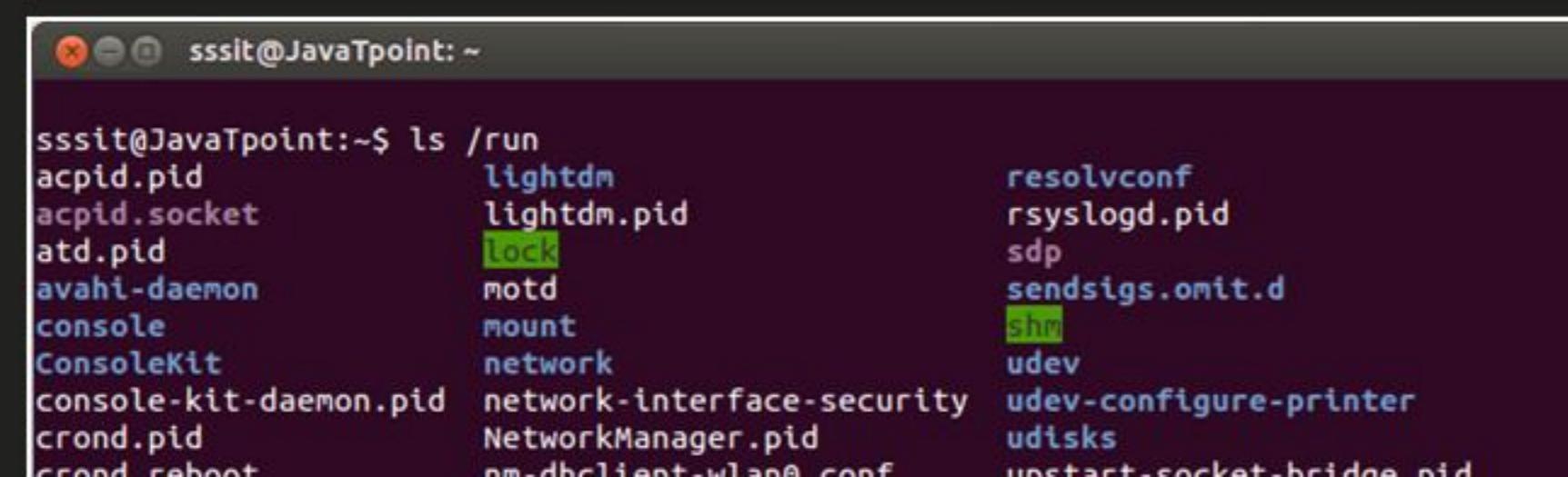
The '/cdrom' directory is not in the standard FHS but cdrom can be mounted on this directory. Ideally according to standard FHS cdrom should be mounted under '/media'.

/run

The '/run' directory stores run-time variable data. Run-time variable data means, data about the running system since last boot. For eg, running daemons.

Example:

```
ls /run
```



```
sssit@JavaPoint:~$ ls /run
acpid.pid          lightdm
acpid.socket       lightdm.pid
atd.pid            lock
avahi-daemon       motd
console            mount
ConsoleKit          network
console-kit-daemon.pid  network-interface-security
crond.pid          NetworkManager.pid
crond_reboot        pm-dbus-client-wlan0.conf
                           udev
                           udev-configure-printer
                           udisks
                           upstart-socket-bridge.pid
                           resolvconf
                           rsyslogd.pid
                           sdp
                           sendSIGS.omit.d
                           shm
                           udev
                           udev-configure-printer
                           udisks
                           upstart-socket-bridge.pid
```

✓ [Linux File Security](#)

system since last boot. For eg, running daemons.

✓ [Linux Shell](#)

✓ [Linux Networking](#)

✓ [vi editor](#)

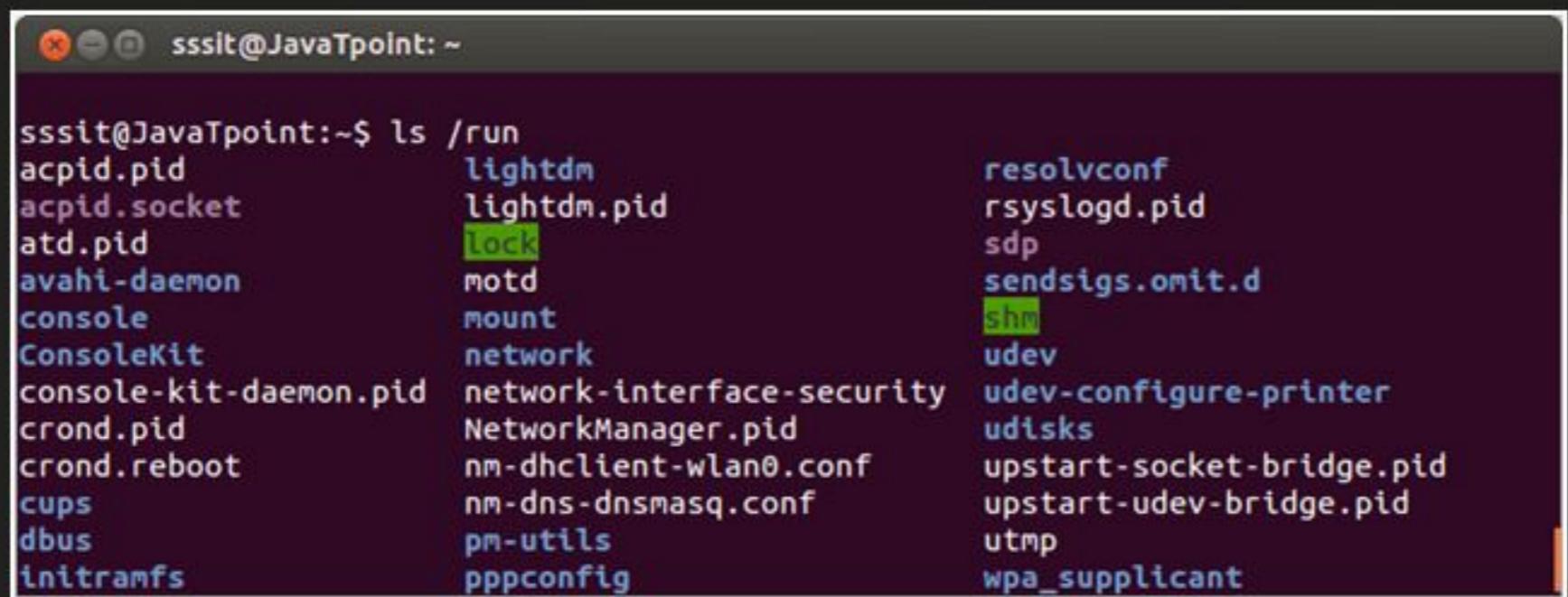
✓ [Linux System Admin](#)

✓ [100+ Linux Commands](#)

✓ [Linux Interview](#)

Example:

`ls /run`



```
sssit@JavaTpoint:~$ ls /run
acpid.pid          lightdm
acpid.socket       lightdm.pid
atd.pid            lock
avahi-daemon
console
ConsoleKit
console-kit-daemon.pid
crond.pid
crond.reboot
cups
dbus
initramfs          network
                   network-interface-security
                   NetworkManager.pid
                   nm-dhclient-wlan0.conf
                   nm-dns-dnsmasq.conf
                   pm-utils
                   pppconfig
                   resolvconf
                   rsyslogd.pid
                   sdp
                   sendsigs.omit.d
                   shm
                   udev
                   udev-configure-printer
                   udisks
                   upstart-socket-bridge.pid
                   upstart-udev-bridge.pid
                   utmp
                   wpa_supplicant
```

Look at the above snapshot, command "`ls /run`" displays content of '/run'.

/lost + found

During system crash or in any other situation when Linux file system checker (fsck) recovers lost data, that data is stored in this directory. Data may or may not be in a good condition.

[Next Topic](#)

[Linux Filters](#)

✓ [Introduction to Linux](#)

✓ [Linux Directories](#)

✓ [Linux Files](#)

✓ [Linux Man Pages](#)

✓ [Linux File Contents](#)

✓ [Linux File Hierarchy](#)

✓ [Linux Filters](#)

⊕ [Linux Filters](#)

⊕ [Linux cat](#)

⊕ [Linux cut](#)

⊕ [Linux grep](#)

⊕ [Linux comm](#)

⊕ [Linux sed](#)

⊕ [Linux tee](#)

⊕ [Linux tr](#)

⊕ [Linux uniq](#)

⊕ [Linux wc](#)

⊕ [Linux od](#)

⊕ [Linux sort](#)

⊕ [Linux gzip](#)

✓ [Linux I/O Redirection](#)

✓ [Linux Unix Tools](#)

✓ [Linux Regex](#)

✓ [Linux Users](#)

✓ [Linux File Security](#)

✓ [SCROLL TO TOP](#)

Linux Filters

← Prev

Next →

Linux Filter commands accept input data from **stdin** (standard input) and produce output on **stdout** (standard output). It transforms plain-text data into a meaningful way and can be used with pipes to perform higher operations.

These filters are very small programs that are designed for a specific function which can be used as building blocks.

Linux Filter Commands

1. [cat](#)

2. [cut](#)

3. [grep](#)

4. [comm](#)

5. [sed](#)

6. [tee](#)

7. [tr](#)

8. [uniq](#)

9. [wc](#)

10. [od](#)

11. [sort](#)

✓ [Introduction to Linux](#)

✓ [Linux Directories](#)

✓ [Linux Files](#)

✓ [Linux Man Pages](#)

✓ [Linux File Contents](#)

✓ [Linux File Hierarchy](#)

✓ [Linux Filters](#)

⊕ [Linux Filters](#)

⊕ [Linux cat](#)

⊕ [Linux cut](#)

⊕ [Linux grep](#)

⊕ [Linux comm](#)

⊕ [Linux sed](#)

⊕ [Linux tee](#)

⊕ [Linux tr](#)

⊕ [Linux uniq](#)

⊕ [Linux wc](#)

⊕ [Linux od](#)

⊕ [Linux sort](#)

⊕ [Linux gzip](#)

✓ [Linux I/O Redirection](#)

✓ [Linux Unix Tools](#)

✓ [Linux Regex](#)

✓ [Linux Users](#)

✓ [Linux File Security](#)

✓ [Linux Environment Variables](#)

Linux cut Command

← Prev

Next →

Linux cut command is useful for selecting a specific column of a file. It is used to cut a specific sections by byte position, character, and field and writes them to standard output. It cuts a line and extracts the text data. It is necessary to pass an argument with it; otherwise, it will throw an error message.

To cut a specific section, it is necessary to specify the delimiter. A delimiter will decide how the sections are separated in a text file. Delimiters can be a space (' '), a hyphen (-), a slash (/), or anything else. After '-f' option, the column number is mentioned.

Syntax:

```
cut OPTION... [FILE]...
```

Options:

The following command line options are used by the cut command to make it more specific:

-b, --bytes=LIST: It is used to cut a specific section by bytes.

-c, --characters=LIST: It is used to select the specified characters.

-d, --delimiter=DELIM: It is used to cut a specific section by a delimiter.

-f, --fields=LIST: It is used to select the specific fields. It also prints any line that does not contain any delimiter character, unless the -s option is specified.

↑ SCROLL TO TOP

- ✓ [Linux I/O Redirection](#)
- ✓ [Linux Unix Tools](#)
- ✓ [Linux Regex](#)
- ✓ [Linux Users](#)
- ✓ [Linux File Security](#)
- ✓ [Linux Shell](#)
- ✓ [Linux Networking](#)
- ✓ [vi editor](#)
- ✓ [Linux System Admin](#)
- ✓ [100+ Linux Commands](#)
- ✓ [Linux Interview](#)

-b, --bytes=LIST: It is used to cut a specific section by bytes.

-c, --characters=LIST: It is used to select the specified characters.

-d, --delimiter=DELIM: It is used to cut a specific section by a delimiter.

-f, --fields=LIST: It is used to select the specific fields. It also prints any line that does not contain any delimiter character, unless the **-s** option is specified.

-n: It is used to ignore any option.

--complement: It is used to complement the set of selected bytes, characters or fields

-s, --only-delimited: It is used to not print lines that do not have delimiters.

--output-delimiter=STRING: This option is specified to use a STRING as an output delimiter; The default is to use "input delimiter".

-z, --zero-terminated: It is used if line delimiter is NUL, not newline.

--help: It is used to display the help manual.

--version: It is used to display the version information.

Examples of the cut command

Let's see the following examples of the cut command:

- Cut by using Hyphen as delimiter
- Cut by using Space as delimiter
- Cut by byte position

from the above commands, the output will be trimmed from hyphen (-). Consider the below output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cat marks.txt
alex-50
alen-70
jon-75
carry-85
celena-90
justin-80
javatpoint@javatpoint-Inspiron-3542:~$ cut -d- -f2 marks.txt
50
70
75
85
90
80
javatpoint@javatpoint-Inspiron-3542:~$ cut -d- -f1 marks.txt
alex
alen
jon
carry
celena
justin
```

As we can see from the above output, our delimiter is the hyphen (-); hence we have used (-) after (-d).

Command "cut -d- -f1 marks.txt" displays column 1 and command "cut -d- -f2 marks.txt" displays column 2.

Using Space As Delimiter

If we want to use space as a delimiter, then we have to quote the space (' ') with the cut command. To cut the output by using space as delimiter, execute the command as follows:

Using Space As Delimiter

If we want to use space as a delimiter, then we have to quote the space (' ') with the cut command. To cut the output by using space as delimiter, execute the command as follows:

```
cut -d '' -f(columnNumber) <fileName>
```

Consider the following commands:

```
cut -d '' -f2 exm.txt
```

```
cut -d '' -f5 exm.txt
```

From the above commands, the output will be trimmed after space for the specified column. The above commands will produce the output as follows:

```
javatpoint@javatpoint-Inspiron-3542:~$ cat exm.txt
Apple is red.
mango is yellow.
your dress color is Red,
Red color suits on all.
javatpoint@javatpoint-Inspiron-3542:~$ cut -d '' -f2 exm.txt
is
is
dress
color
javatpoint@javatpoint-Inspiron-3542:~$ cut -d '' -f5 exm.txt
```

Red,
all.

Cut by byte

The '-b' option is used to cut a section of line by byte. To cut a file by its byte position, execute the command as follows:

```
cut -b <byte number> <file name>
```

Consider the below command:

```
cut -b 2 exm.txt
```

The above command will cut the line by a specified byte position. Consider the below output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cut -b 2 exm.txt
p
a
o
e
```

Cut by Character

The '-c' option is used to cut a specific section by character. However, these character arguments can be a number or a range of numbers, a list of comma-separated numbers, or any other character.

To cut by specified character, execute the command as follows:

```
cut -c < characters> <file name>
```

```
javatpoint@javatpoint-Inspiron-3542:~$ cut -b 2 exm.txt
P
a
o
e
```

Cut by Character

The '**-c**' option is used to cut a specific section by character. However, these character arguments can be a number or a range of numbers, a list of comma-separated numbers, or any other character.

To cut by specified character, execute the command as follows:

```
cut -c < characters > <file name>
```

Consider the below commands:

```
cut -c 1,6 exm.txt
cut -c 1-3 exm.txt
```

The above commands will cut the line by the specified characters. Consider the below output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cat exm.txt
Apple is red.
mango is yellow.
your dress color is Red,
Red color suits on all.
javatpoint@javatpoint-Inspiron-3542:~$ cut -c 1,6 exm.txt
A
m
```

line, and the second command is cutting the first to the third character from each line.

Cut by Complement Pattern

The '--complement' option is used to cut by the complement. This option is supported by the BSD version of the cut. To cut by the complement pattern, execute the command as follows:

```
cut --complement < complement pattern> <file name>
```

The compliment pattern can be a list of bytes, characters, or fields.

Consider the below command:

```
cut --complement -c 1 exm.txt
```

The above command will cut the file by the first character. Consider the below output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cut --complement -c 1 exm.txt
apple is red.
ango is yellow.
our dress color is Red,
ed color suits on all.
```

[Next Topic](#)[Linux grep](#)

+ New chat

Linux grep command

/usr/bin/linux

Linux current directory

Advantages of Linux

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

1. It allows you to search for a pattern in one or multiple files quickly and efficiently.
2. It supports regular expressions, which are powerful and flexible search patterns.
3. It can be used in combination with other Linux commands, such as `find`, `ls`, and `sed`, to perform complex operations on files.
4. It provides various options, such as case-insensitivity, word matching, and invert match, to customize your search.
5. It works in both interactive and non-interactive modes, making it suitable for use in shell scripts and automation.
6. It is a standard Linux command that is available on most Linux distributions, ensuring compatibility and portability.
7. It is a text-based command, making it easy to use and learn for both beginners and advanced users.

Overall, the `grep` command is a powerful and essential tool for text processing and file manipulation in Linux.

⟳ Regenerate response

Send a message...

+ New chat

□ Linux grep co

□ /usr/bin lin linux

□ Linux current directo

□ Advantages of Linux

—
Clear conversations○ Upgrade to **Plus**

⚙ Settings

✉ Get help

[→ Log out

```
grep pattern [file(s)]      # Search for the pattern in one or more files
grep pattern file1 file2    # Search for the pattern in file1 and file2

grep -i pattern [file(s)]    # Search case-insensitively for the pattern in one or more
grep -i hello myfile.txt    # Search for "hello" (case-insensitive) in myfile.txt

grep -v pattern [file(s)]    # Invert the match, show all lines that do not match the pa
grep -v hello myfile.txt    # Show all lines in myfile.txt that do not contain "hello"

grep -w pattern [file(s)]    # Search for the whole word (exact match) of the pattern in
grep -w hello myfile.txt    # Search for the whole word "hello" in myfile.txt

grep -r pattern [dir]        # Search for the pattern recursively in a directory and its
grep -r hello /home/user    # Search for "hello" in all files under /home/user

grep -n pattern [file(s)]    # Show the line number for each matching line in one or mor
grep -n hello myfile.txt    # Show the line number for each line in myfile.txt that cont
```

Regenerate response

Send a message...



```
grep -w pattern [file(s)]      # Search for the whole word (exact match) of the pattern in
grep -w hello myfile.txt       # Search for the whole word "hello" in myfile.txt

grep -r pattern [dir]           # Search for the pattern recursively in a directory and its
grep -r hello /home/user        # Search for "hello" in all files under /home/user

grep -n pattern [file(s)]       # Show the line number for each matching line in one or more files
grep -n hello myfile.txt       # Show the line number for each line in myfile.txt that contains "hello"

grep -c pattern [file(s)]       # Show only the count of matching lines in one or more files
grep -c hello myfile.txt       # Show the count of lines in myfile.txt that contain "hello"

grep -l pattern [file(s)]       # Show only the names of the files that contain the pattern
grep -l hello myfile.txt       # Show the name of the file(s) that contain "hello"

grep -h pattern [file(s)]       # Suppress the output of filenames, show only the matching lines
grep -h hello myfile.txt       # Show only the lines in myfile.txt that contain "hello"
```

⟳ Regenerate response

Send a message...

+ New chat

☐ Linux grep commands

☐ /usr/bin/linux

☐ Linux current directory

☐ Advantages of Linux

-

☒ Clear conversations

👤 Upgrade to **NEW**
Plus

⚙️ Settings

↗️ Get help

[→ Log out

+ New chat

Linux grep co ↗ ⚡

/usr/bin lin linux

Linux current directo

Advantages of Linux

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

```
grep -h pattern [file(s)] # Suppress the output of filenames, show only the matching  
grep -h hello myfile.txt # Show only the lines in myfile.txt that contain "hello"
```

Note that you can combine multiple options and arguments to create more complex search patterns. For example, you can search for a pattern recursively in a directory, ignoring case, and showing only the matching lines:

arduino

Copy code

```
grep -r -i -h "hello world" /home/user
```

This will search for the phrase "hello world" in all files under /home/user, ignoring case, and showing only the matching lines (not the filenames).

⟳ Regenerate response

Send a message...



- ✓ [Linux Files](#)
- ✓ [Linux Man Pages](#)
- ✓ [Linux File Contents](#)
- ✓ [Linux File Hierarchy](#)
- ✓ [Linux Filters](#)
 - ⊕ [Linux Filters](#)
 - ⊕ [Linux cat](#)
 - ⊕ [Linux cut](#)
 - ⊕ [Linux grep](#)
 - ⊕ [Linux comm](#)
 - ⊕ [Linux sed](#)
 - ⊕ [Linux tee](#)
 - ⊕ [Linux tr](#)
 - ⊕ [Linux uniq](#)
 - ⊕ [Linux wc](#)
 - ⊕ [Linux od](#)
 - ⊕ [Linux sort](#)
 - ⊕ [Linux gzip](#)
- ✓ [Linux I/O Redirection](#)
- ✓ [Linux Unix Tools](#)
- ✓ [Linux Regex](#)
- ✓ [Linux Users](#)
- ✓ [Linux File Security](#)
- ✓ [Linux Shell](#)
- ✓ [Linux Networking](#)
- ✓ [vi editor](#)

Examples

The 'grep' command stands for "**global regular expression print**". grep command filters the content of a file which makes our search easy.

It is a command-line utility to search plain-text data groups for lines that are the same as a regular expression. The name "**grep**" comes from the command, i.e., ed, which contains the same effect. Originally, grep was designed for the Unix operating system, but it became available for every Unix-like system later and a few others like OS 9.

The grep filter finds a file for a specific character pattern and shows every line that includes that pattern. Characters should be in quotation marks if they occur in the pattern parameter with a special meaning for the shell. Usually, we must enclose the whole pattern in one quotation mark if the pattern parameter is not a common string.

In an expression like [a-z], the minus sign (-) cmd describes a range based on the current collating order. A collating order may specify equivalent classes in character ranges for use. The grep command assumes stdin when no files are mentioned.

The pattern is called the regular expression that is found inside the file. The pattern is restricted regular expressions in the format of the egrep or ed command. The grep command applies a solid non-deterministic algorithm. It comes in handy when we need to filter large log files.

Brief History of grep

Before it was titled, grep was a confidential utility specified by **Ken Thompson** to find files for several patterns. Unknown to its existence, **Doug McIlroy** asked Thompson to specify such a function. Replying that he would overnight think of such a utility. Thompson made improvements and corrected bugs for

If you want to output a single column, you have to specify number of the columns which are not to be displayed.

Syntax:

comm -23 (To display first column)

comm -13 (To display second column)

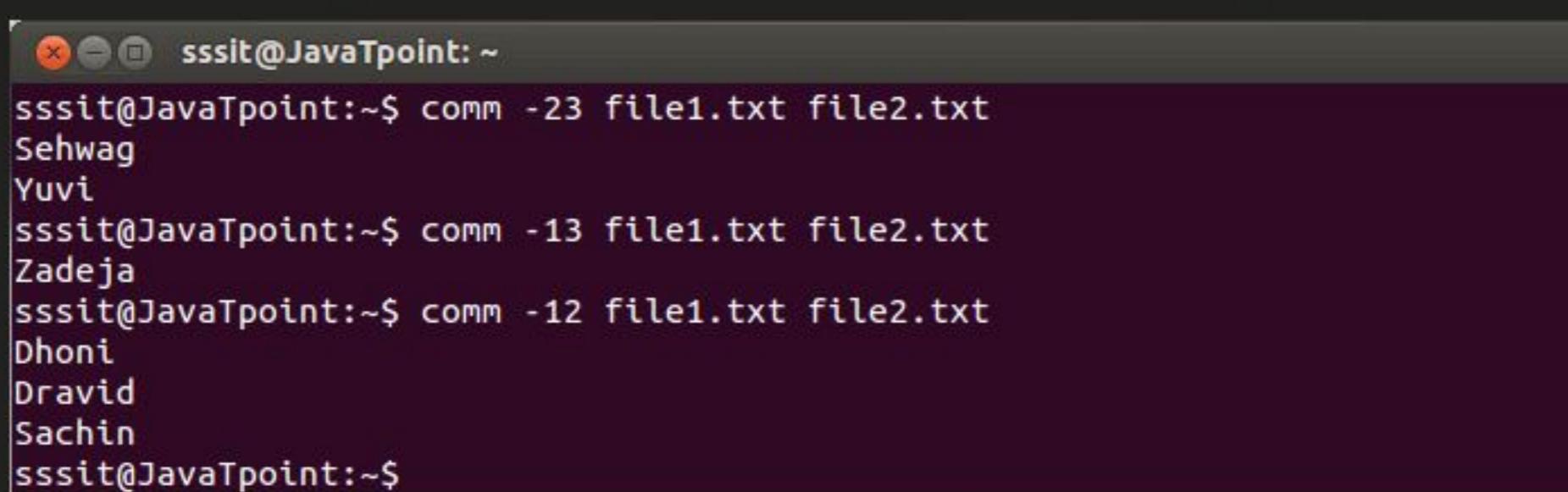
comm -12 (To display third column)

Example:

comm -23 file1.txt file2.txt

comm -13 file1.txt file2.txt

comm -12 file1.txt file2.txt



The screenshot shows a terminal window with a dark background and light-colored text. It displays three commands being run at a prompt: 'sssit@JavaPoint:~\$'. The first command, 'comm -23 file1.txt file2.txt', outputs 'Sehwag' and 'Yuvi'. The second command, 'comm -13 file1.txt file2.txt', outputs 'Zadeja'. The third command, 'comm -12 file1.txt file2.txt', outputs 'Dhoni', 'Dravid', and 'Sachin'. The terminal window has a standard OS X style with a title bar and close/minimize buttons.

```
sssit@JavaPoint:~$ comm -23 file1.txt file2.txt
Sehwag
Yuvi
sssit@JavaPoint:~$ comm -13 file1.txt file2.txt
Zadeja
sssit@JavaPoint:~$ comm -12 file1.txt file2.txt
Dhoni
Dravid
Sachin
sssit@JavaPoint:~$
```

Look at the above snapshot, column number which needs to be displayed are not mentioned in the 'comm' command.

[✓ Linux Files](#)[✓ Linux Man Pages](#)[✓ Linux File Contents](#)[✓ Linux File Hierarchy](#)[✓ Linux Filters](#)[↳ Linux Filters](#)[↳ Linux cat](#)[↳ Linux cut](#)[↳ Linux grep](#)[↳ Linux comm](#)[↳ Linux sed](#)[↳ Linux tee](#)[↳ Linux tr](#)[↳ Linux uniq](#)[↳ Linux wc](#)[↳ Linux od](#)[↳ Linux sort](#)[↳ Linux gzip](#)[✓ Linux I/O Redirection](#)[✓ Linux Unix Tools](#)[✓ Linux Regex](#)[✓ Linux Users](#)[✓ Linux File Security](#)[✓ Linux Shell](#)[✓ Linux Networking](#)[✓ vi editor](#)[↑ SCROLL TO TOP](#)

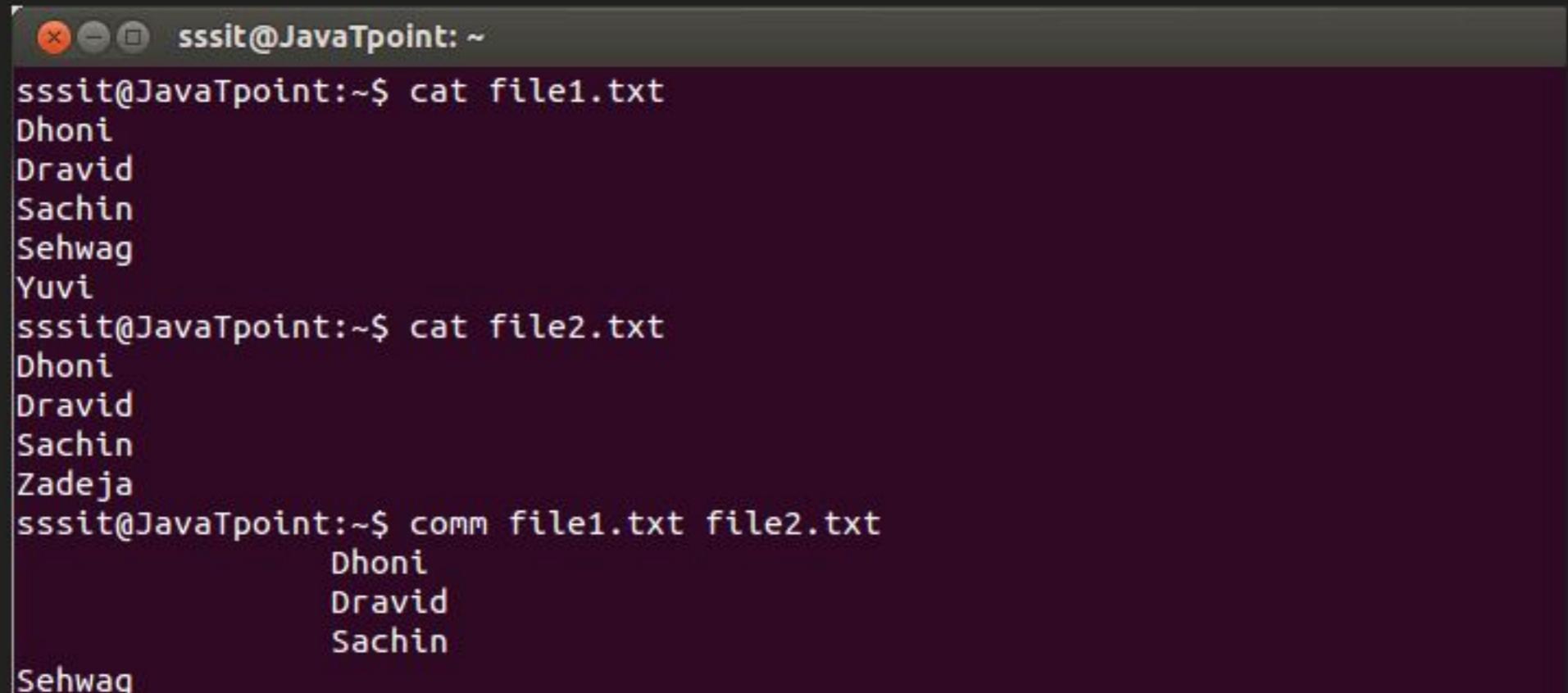
The 'comm' command compares two files or streams. By default, 'comm' will always display **three columns**. First column indicates non-matching items of first file, second column indicates non-matching items of second file, and third column indicates matching items of both the files. Both the files has to be in sorted order for 'comm' command to be executed.

Syntax:

```
comm <file1> <file2>
```

Example:

```
comm file1.txt file2.txt
```



sssit@JavaPoint:~\$ cat file1.txt

Dhoni
Dravid
Sachin
Sehwag
Yuv*i*

sssit@JavaPoint:~\$ cat file2.txt

Dhoni
Dravid
Sachin
Zadeja

sssit@JavaPoint:~\$ comm file1.txt file2.txt

Dhoni	
Dravid	
Sachin	
Sehwag	

+ New chat

New chat



Linux grep command

/usr/bin lin linux

Linux current directo

Clear conversations

Upgrade to
Plus

NEW

Settings

Get help

Log out



The "sed" command in Linux is a powerful text editor that can perform various text manipulation tasks. It stands for "stream editor" and works by processing a stream of input text and applying a set of commands to modify it. Here are some commonly used examples:

1. Search and replace:

The "s" command is used for search and replace operation. It can replace all occurrences of a word or pattern in a file or stream.

Example: `sed 's/old/new/g' filename`

2. Print lines:

The "p" command is used to print specific lines from a file or stream.

Example: `sed -n '10p' filename` - this will print only the 10th line of the file

3. Delete lines:

The "d" command is used to delete specific lines from a file or stream.

Example: `sed '2d' filename` - this will delete the second line from the file

There was an error generating a response

4. Append and insert lines:

The "a" command is used to append text after the current line, while "i" command is

Regenerate response

Linux sed command. — Mozilla Firefox

sed Command in Linux/Unix w X Linux sed command. BlockSite | blocksite.co X +

← → C https://chat.openai.com/chat ⭐

+ New chat

□ I2C Communication

□ UART for Serial Com

□ Cortex-M4 OS Supp

Clear conversations

Upgrade to Plus NEW

Settings

Get help

[→ Log out

Send a message... Regenerate response

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts

Some of the commonly used options with the `sed` command are as follows:

- `'-n'`: Suppresses the automatic printing of the pattern space.
- `'-i'`: Modifies the file in-place.
- `'-e'`: Specifies a script to execute.
- `'-f'`: Specifies a script file to execute.

Some commonly used editing commands with `sed` are:

- `'s'`: Substitutes one string for another.
- `'d'`: Deletes a line or range of lines.
- `'p'`: Prints the pattern space.
- `'a'`: Appends text after a line.

[←](#) [→](#) [C](#)<https://chat.openai.com/chat>[+ New chat](#)[I2C Communication](#)[UART for Serial Com](#)[Cortex-M4 OS Supp](#)[Clear conversations](#)[Upgrade to Plus](#)

NEW

[Settings](#)[Get help](#)[\[→\] Log out](#)

The `sed` command is a powerful utility in Linux that is used for streamlining text editing operations. `sed` stands for "Stream Editor" and it is used to perform basic text transformations on an input stream (a file or data input from a pipe).



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

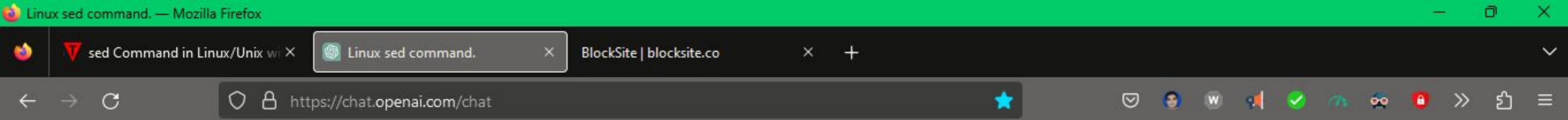
arduino

[Copy code](#)

```
sed options 'command' filename
```

Here, `options` are the command line options that you can use with the `sed` command, `command` is the editing command that you want to perform, and `filename` is the name of the

[Regenerate response](#)



Some commonly used editing commands with `sed` are:

- `s`: Substitutes one string for another.
- `d`: Deletes a line or range of lines.
- `p`: Prints the pattern space.
- `a`: Appends text after a line.
- `i`: Inserts text before a line.
- `c`: Changes the selected line(s) with new text.

For example, to substitute all occurrences of "foo" with "bar" in a file named `file.txt`, you can use the following command:

arduino

⟳ Regenerate response

Copy code

Send a message...



[→] Log out

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts

+ New chat

I2C Communication

UART for Serial Com

Cortex-M4 OS Supp

 Clear conversations

Upgrade to
Plus  NEW

Settings

 Get help

[→ Log out

arduino

 Copy code

```
sed 's/foo/bar/g' file.txt
```

This will replace all occurrences of "foo" with "bar" and print the modified output to the terminal.

If you want to modify the file in-place, you can use the `-i` option:

arduino

 Copy code

```
sed -i 's/foo/bar/g' file.txt
```

This will modify the file `file.txt` directly, replacing all occurrences of "foo" with "bar".

⟳ Regenerate response

Send a message...



← → C



https://chat.openai.com/chat



+ New chat

□ I2C Communication

□ UART for Serial Com

□ Cortex-M4 OS Supp

Clear conversations

Upgrade to Plus NEW

⚙ Settings

↗ Get help

[→ Log out

1. Substitute (s) command: This command is used to replace text in a file.

arduino

Copy code

```
sed 's/search/replace/g' filename
```

- `search`: The text to search for
- `replace`: The text to replace the search text with
- `g`: Global replace, replaces all occurrences in a line.

Example: To replace all occurrences of "apple" with "orange" in a file named `fruits.txt`, use the following command:

arduino

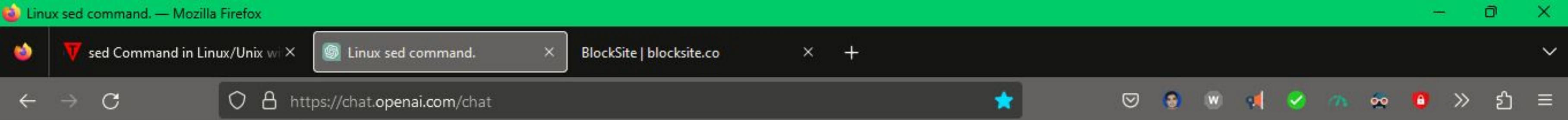
Copy code

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

Regenerate response

Send a message...





2. Delete (d) command: This command is used to delete lines from a file.

arduino

Copy code

sed 'd' filename

Example: To delete all lines containing the text "banana" from a file named `fruits.txt`, use the following command:

arduino

Copy code

sed '/banana/d' fruits.txt

3. Print (p) command: This command is used to print lines from a file.

Regenerate response

arduino

Copy code

Send a message...



[→] Log out

3. Print (p) command: This command is used to print lines from a file.

arduino

 Copy code

```
sed -n 'p' filename
```

+ New chat

I2C Communication

UART for Serial Comm

Cortex-M4 OS Support

Clear conversations

Upgrade to
Plus **NEW**

Settings

 Get help

[→ Log out

Example: To print all lines containing the text "orange" from a file named `fruits.txt`, use the following command:

arduino

 Copy code

```
sed -n '/orange/p' fruits.txt
```

4. Append (a) command: This command is used to append text after a line.

arduino

 Regenerate response

 Copy code

Send a message...

Linux sed command. — Mozilla Firefox

sed Command in Linux/Unix w X Linux sed command. BlockSite | blocksite.co X + https://chat.openai.com/chat ⭐ ↻ 📌 📁 🗑️ ✅ 🔍 🔒 ➡️ 📁

4. Append (a) command: This command is used to append text after a line.

arduino

sed 'a\text' filename

Copy code

• `text`: The text to append.

Example: To append the text "I love fruits." after each line in a file named `fruits.txt`, use the following command:

css

sed 'a\I love fruits.' fruits.txt

Copy code

5. Insert (i) command: This command is used to insert text before a line.

Regenerate response

arduino

Send a message...

Copy code

New chat

I2C Communication

UART for Serial Com

Cortex-M4 OS Supp

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts

5. Insert (i) command: This command is used to insert text before a line.

arduino

 Copy code

```
sed 'i\text' filename
```

- `text`: The text to insert.

Example: To insert the text "These are my favorite fruits:" before the first line in a file named `fruits.txt`, use the following command:

arduino

 Copy code

```
sed '1i\These are my favorite fruits:' fruits.txt
```

6. Change (c) command: This command is used to replace a line or range of lines with new text.

Send a message...

+ New chat

I2C Communication

UART for Serial Comm

Cortex-M4 OS Supp

 Clear conversations

Upgrade to Plus NEW

Settings

 Get help

[→ Log out

6. Change (c) command: This command is used to replace a line or range of lines with new text.

arduino

 Copy code

```
sed 'c\text' filename
```

- `text`: The text to replace the selected line(s) with.

Example: To replace the first line of a file named `fruits.txt` with the text "These are my favorite fruits:", use the following command:

arduino

 Copy code

```
sed '1c\These are my favorite fruits:' fruits.txt
```

 Regenerate response

[Send a message...](#)



Examples of sed Command

Let's see the following examples:

- Applying to the STDIN directory
- Global Replacement
- Removing a Line
- Using the Multiple sed Command
- Reading Commands From a File
- Replacing Characters
- Limiting the sed
- Inserting and Appending Text
- Modifying Lines
- Transformation of Characters
- Printing the Line Numbers

Applying to the STDIN directory

The sed command is not just limited to manipulate files; also, we can apply it to the STDIN directory.



How to use the tee command

The tee command is used to write a standard input to standard output and a file. It is used after a pipe. To write to standard output and a file, specify the tee command after a pipe and provide the file(s) name.

Consider the below command:

```
cat weeks.txt | tee newfile.txt
```

The above command will write the input of the 'weeks.txt' to 'newfile.txt.' Consider the below output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cat weeks.txt | tee newfile.txt
```

```
sunday  
monday  
tuesday  
wednesday  
thursday  
friday  
saturday
```

```
javatpoint@javatpoint-Inspiron-3542:~$ cat newfile.txt
```

```
sunday  
monday  
tuesday  
wednesday  
thursday  
friday  
saturday
```



From the above output, a file 'newfile.txt' is created using the tee command.

Write a file and append output

The '-a' option is used with the tee command to append the output and write it to a file. Consider the below command:

```
echo 'Saturday and Sunday are week offs' | tee -a newfile.txt
```

The above command will append the specified input to the 'newfile.txt'. Consider the below output:

```
javatpoint@javatpoint-Inspiron-3542:~$ echo 'saturday and sunday are week offs'  
| tee -a newfile.txt  
saturday and sunday are week offs  
javatpoint@javatpoint-Inspiron-3542:~$ cat newfile.txt  
sunday  
monday  
tuesday  
wednesday  
thursday  
friday  
saturday  
saturday and sunday are week offs
```

As from the above output, the specified input is appended to 'newfile.txt'.



The tee command allows us to write to multiple files. To write to the multiple files, specify the names of the files after the tee command as follows:

```
echo " add text" | tee file1.txt file2.txt file3.txt
```

The above command will create all the specified files. Consider the below output:

```
javatpoint@javatpoint-Inspiron-3542:~$ echo " add text" | tee file1.txt file2.tx  
t file3.txt  
add text  
javatpoint@javatpoint-Inspiron-3542:~$ cat file1.txt  
add text  
javatpoint@javatpoint-Inspiron-3542:~$ cat file2.txt  
add text  
javatpoint@javatpoint-Inspiron-3542:~$ cat file3.txt  
add text
```

Write to a privileged file

The tee command allows us to write to a file having sudo privilege. If we try to write a file owned by the root user will through the permission error. But, we can elevate the sudo permission by executing the tee command as follows:

```
echo "some text" | sudo tee -a <file name>
```

command as follows:

```
echo "some text" | sudo tee -a <file name>
```

Ignoring Interrupts

The '-i' option is used to ignore the interrupts. This is useful if we want to stop and exit the command gracefully during the execution. It is used as follows:

```
command | tee -i <filename>
```

Hide the output

To restrict the tee command not to write to the standard output, redirect it to "/dev/null". Execute the command as follows:

```
echo "Text" | tee newfile.txt >/dev/null
```

Consider the below output:

```
javatpoint@javatpoint-Inspiron-3542:~$ echo "Text" | tee newfile.txt >/dev/null
javatpoint@javatpoint-Inspiron-3542:~$ cat newfile.txt
Text
```

+ New chat

□ I2C Communication

□ UART for Serial Com

□ Cortex-M4 OS Supp

□ Systick timer in Cort

□ Cortex-M4 Interrupt

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

This command lists all processes and saves the output to two files named `processes.txt` and `log.txt`.

3. Appending output to a file:

bash

 Copy code

```
echo "New line" | tee -a output.txt
```

This command appends the text "New line" to the file `output.txt`.

4. Ignoring interrupts:

bash

 Copy code

```
ping google.com | tee -i output.txt
```

This command pings Google's server and saves the output to the file `output.txt`. If you interrupt the command using `ctrl+c`, the command will continue running instead of terminating.

 Regenerate response

5. Prompt-sensitive output:

Send a message...



- ✓ [Introduction to Linux](#)
- ✓ [Linux Directories](#)
- ✓ [Linux Files](#)
- ✓ [Linux Man Pages](#)
- ✓ [Linux File Contents](#)
- ✓ [Linux File Hierarchy](#)

[Linux Filters](#)

- ⊕ [Linux Filters](#)
- ⊕ [Linux cat](#)
- ⊕ [Linux cut](#)
- ⊕ [Linux grep](#)
- ⊕ [Linux comm](#)
- ⊕ [Linux sed](#)
- ⊕ [Linux tee](#)
- ⊕ [Linux tr](#)
- ⊕ [Linux uniq](#)
- ⊕ [Linux wc](#)
- ⊕ [Linux od](#)
- ⊕ [Linux sort](#)
- ⊕ [Linux gzip](#)

[Linux I/O Redirection](#)

[Linux Unix Tools](#)

[Linux Regex](#)

[Linux Users](#)

[Linux File Security](#)

↑ [SCROLL TO TOP](#)

Linux tr

[← Prev](#)[Next →](#)

The command 'tr' stands for '**translate**'. It is used to translate, like from lowercase to uppercase and vice versa or new lines into spaces.

Syntax:

```
command | tr <'old'> <'new'>
```

Change Case

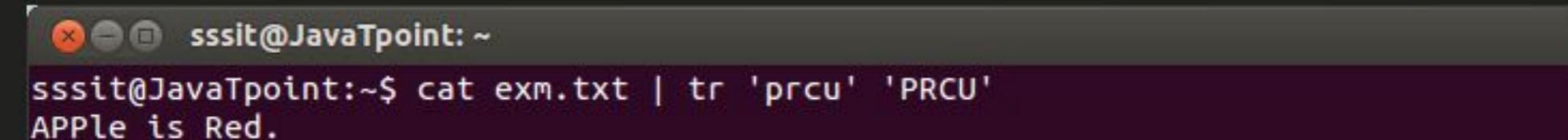
The 'tr' command can change case.

Syntax:

```
command | tr <'old'> <'new'>
```

Example:

```
cat exm.txt | tr 'prcu' 'PRCU'
```



```
sssit@JavaTpoint: ~
sssit@JavaTpoint:~$ cat exm.txt | tr 'prcu' 'PRCU'
PRCU is Red.
```

tr Options

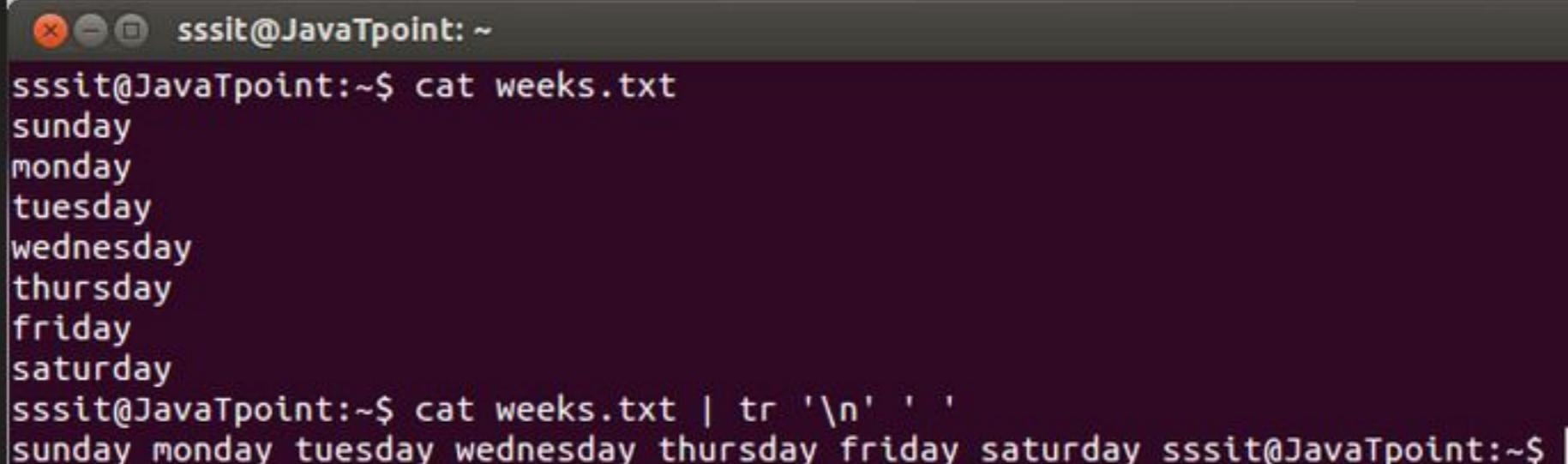
- **tr -s** : The 'tr -s' command squeezes the occurrence of multiple characters into one.

Syntax:

```
command | tr -s '<letter>'
```

Example:

```
cat jtp.txt | tr -s 'I'  
cat spaces.txt | tr -s ''
```



The screenshot shows a terminal window with a dark background and light-colored text. It displays two examples of the tr command. The first example shows the command "cat jtp.txt | tr -s 'I'" being run, which results in all occurrences of the letter 'I' being squeezed into one. The second example shows the command "cat spaces.txt | tr -s ''", which results in all spaces being squeezed into a single space. The terminal prompt is "sssit@JavaTpoint:~\$".

```
sssit@JavaTpoint:~$ cat weeks.txt  
sunday  
monday  
tuesday  
wednesday  
thursday  
friday  
saturday  
sssit@JavaTpoint:~$ cat weeks.txt | tr '\n' ''  
sunday monday tuesday wednesday thursday friday saturday sssit@JavaTpoint:~$
```

- ✓ [Introduction to Linux](#)
- ✓ [Linux Directories](#)
- ✓ [Linux Files](#)
- ✓ [Linux Man Pages](#)
- ✓ [Linux File Contents](#)
- ✓ [Linux File Hierarchy](#)
- ✓ [Linux Filters](#)

- ↪ [Linux Filters](#)
- ↪ [Linux cat](#)
- ↪ [Linux cut](#)
- ↪ [Linux grep](#)
- ↪ [Linux comm](#)
- ↪ [Linux sed](#)
- ↪ [Linux tee](#)
- ↪ [Linux tr](#)
- ↪ [Linux uniq](#)

- ↪ [Linux wc](#)
- ↪ [Linux od](#)
- ↪ [Linux sort](#)
- ↪ [Linux gzip](#)
- ✓ [Linux I/O Redirection](#)

- ✓ [Linux Unix Tools](#)
- ✓ [Linux Regex](#)
- ✓ [Linux Users](#)

Linux uniq Command

[← Prev](#)[Next →](#)

Linux uniq command is used to remove all the repeated lines from a file. Also, it can be used to display a count of any word, only repeated lines, ignore characters, and compare specific fields. It is one of the most frequently used commands in the Linux system. It is often used with the sort command because it compares adjacent characters. It discards all the identical lines and writes the output.

Syntax:

```
uniq [OPTION]... [INPUT [OUTPUT]]
```

Options:

Some useful command line options of the uniq command are as following:

-c, --count: it prefixes the lines by the number of occurrences.

-d, --repeated: it is used to print duplicate lines, one for each group.

-D: It is used to print all the duplicate lines.

--all-repeated[=METHOD]: It is quite similar to the '-D' option, the difference between both the options is that it allows separation of groups with an empty line.

-w, --check-chars=N: It is used to compare not more than N characters in lines.

--help: It is used to display help documentation.

--version: It is used to display the version information.

Examples of uniq Command

Let's see the following examples of the uniq command:

- Remove repeated lines
- count the number of occurrences of a word
- Display the repeated lines
- Display the unique lines
- Ignore characters in comparison
- Ignore fields in comparison

Remove repeated lines

To remove repeated lines from a file, execute the basic uniq command as follows:

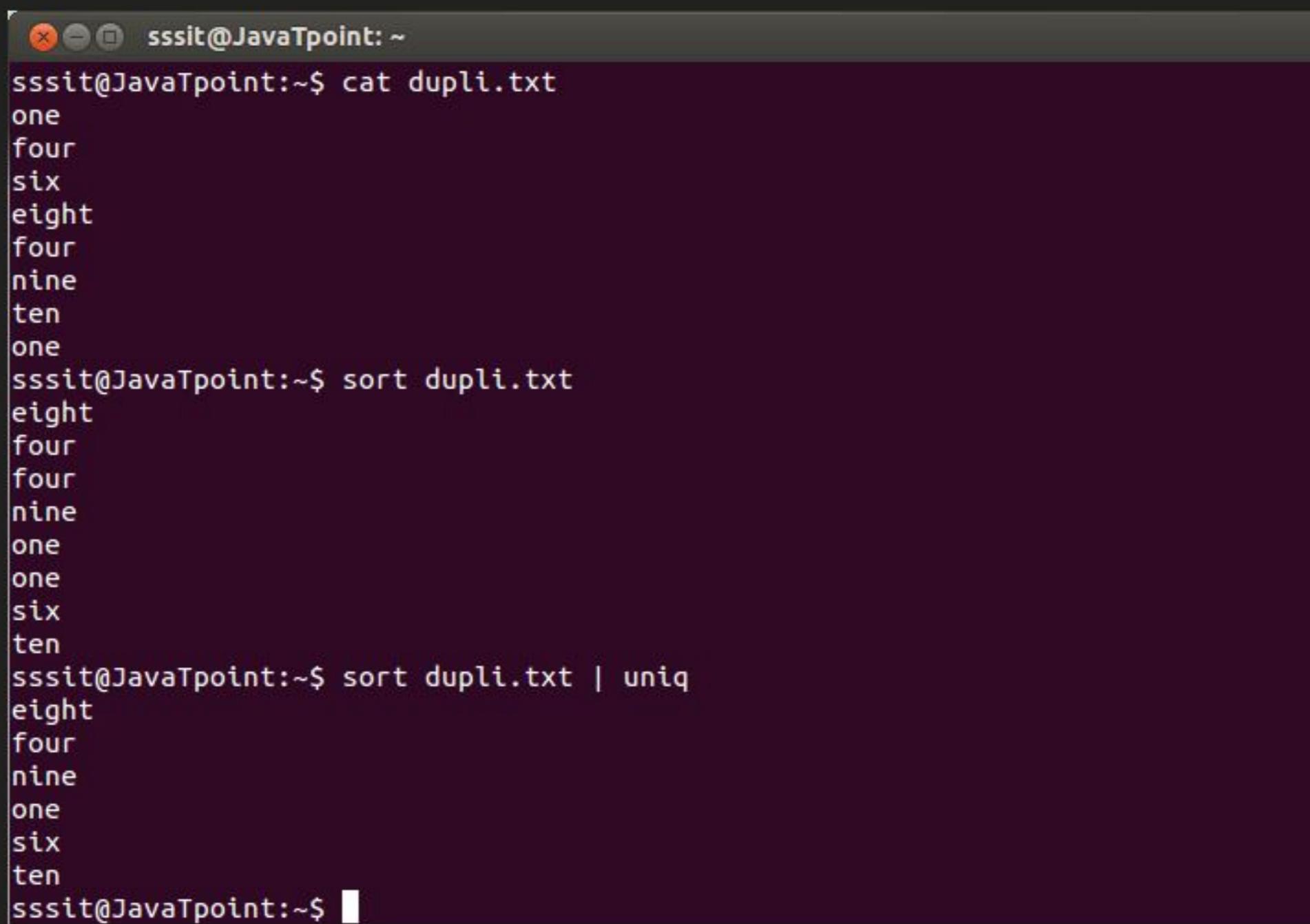
```
sort dupli.txt | uniq
```

The above command will remove the duplicate lines from the file 'dupli.txt.' Consider the below output:

↑ SCROLL TO TOP

```
sort dupli.txt | uniq
```

The above command will remove the duplicate lines from the file 'dupli.txt.' Consider the below output:



```
sssit@JavaTpoint:~$ cat dupli.txt
one
four
six
eight
four
nine
ten
one
sssit@JavaTpoint:~$ sort dupli.txt
eight
four
four
nine
one
one
six
ten
sssit@JavaTpoint:~$ sort dupli.txt | uniq
eight
four
nine
one
six
ten
sssit@JavaTpoint:~$
```

From the above output, the repeating words are ignored.

Count the number of occurrences of a word

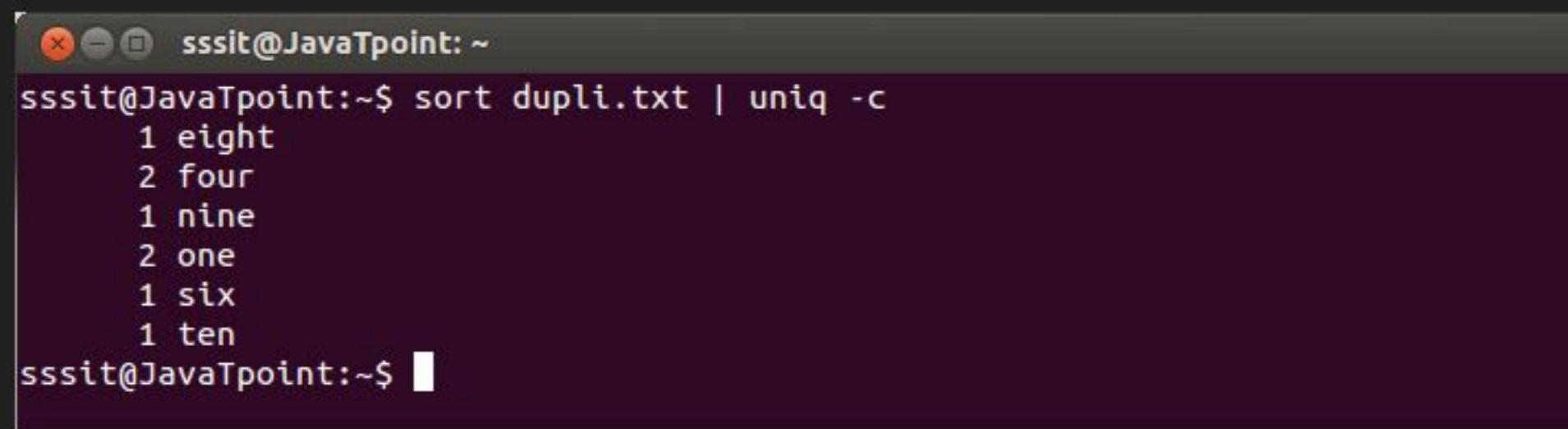
From the above output, the repeating words are ignored.

Count the number of occurrences of a word

We can count the number of occurrences of a word by using the `uniq` command. The '`-c`' option is used to count the word. Execute it as follows:

```
sort dupli.txt | uniq -c
```

The above command will count the words which come in 'dupli.txt'. Consider the below output:



```
sssit@JavaTpoint:~$ sort dupli.txt | uniq -c
 1 eight
 2 four
 1 nine
 2 one
 1 six
 1 ten
sssit@JavaTpoint:~$
```

From the above output, the command "`sort dupli.txt | uniq -c`" counts the number of times a word is repeating.

Display the repeated lines

The '`-d`' option is used to display only the repeated lines. It will only display the lines that will be more than once in a file and write the output to standard output. Consider the below command:

```
↑ ↵x  
1 ten  
sssit@JavaTpoint:~$ █
```

From the above output, the command "sort dupli.txt | uniq -c" counts the number of times a word is repeating.

Display the repeated lines

The '-d' option is used to display only the repeated lines. It will only display the lines that will be more than once in a file and write the output to standard output. Consider the below command:

```
sort dupli.txt | uniq -d
```

The above command will display only the repeated lines. Consider the below output:

```
javatpoint@javatpoint-Inspiron-3542:~$ sort dupli.txt | uniq -d  
four  
one  
two  
three  
five  
six  
seven  
eight  
nine  
ten
```

Display the unique lines

The '-u' option is used to display only the unique lines (which are not repeated). It will only display the lines that occur only once and write the result to standard output. Consider the below command:

```
sort dupli.txt | uniq -u
```

[+ Linux filters](#)[+ Linux cat](#)[+ Linux cut](#)[+ Linux grep](#)[+ Linux comm](#)[+ Linux sed](#)[+ Linux tee](#)[+ Linux tr](#)[+ Linux uniq](#)[+ Linux wc](#)[+ Linux od](#)[+ Linux sort](#)[+ Linux gzip](#)[**✓ Linux I/O Redirection**](#)[**✓ Linux Unix Tools**](#)[**✓ Linux Regex**](#)[**✓ Linux Users**](#)[**✓ Linux File Security**](#)[**✓ Linux Shell**](#)[**✓ Linux Networking**](#)[**✓ vi editor**](#)[**✓ Linux System Admin**](#)[**✓ 100+ Linux Commands**](#)[**✓ Linux Interview**](#)`wc [OPTION]... [FILE]...``wc [OPTION]... --files0-from=F`

Options:

Some useful command line options supported by the wc command are as following:

-c, --bytes: It is used to print the byte counts.

-m, --chars: It is used to print the character counts.

-l, --lines: It is used to print the newline counts.

--files0-from=F: It is used to read input from specified files.

-L, --max-line-length: It is used to print the maximum display width.

-w, --words: It is used to print the word counts.

--help: It is used to display the help manual.

--version: It is used to display the version information.

Examples of the wc Command

Let's see the following examples of the wc command:

- Display count information of a file

Examples of the wc Command

Let's see the following examples of the wc command:

- Display count information of a file
- Display count information of multiple files
- Display the number of lines in a file
- Display the number of characters in a file
- Display the number of bytes in a file
- Display the number of words in a file
- Count the number files in a directory
- Display the length of the longest line

Display count information of a file

To display the complete count information of a file, execute the command without any argument. It will display the number of lines, words, and bytes from the file. Execute the command as follows:

```
wc <file name>
```

Consider the below command:

```
wc exm.txt
```

- Count the number files in a directory
- Display the length of the longest line

Display count information of a file

To display the complete count information of a file, execute the command without any argument. It will display the number of lines, words, and bytes from the file. Execute the command as follows:

```
wc <file name>
```

Consider the below command:

```
wc exm.txt
```

The above command will display the number of lines, number of words, number of bytes, and file name from the file 'exm.txt'. Consider the below output:

```
javatpoint@javatpoint-Inspiron-3542:~$ cat exm.txt
Apple is Red.
mango is yellow.
your dress color is Red,
Red color suits on all.
javatpoint@javatpoint-Inspiron-3542:~$ wc exm.txt
 4 16 80 exm.txt
```

Display count information of multiple files

✓ [Introduction to Linux](#)

✓ [Linux Directories](#)

✓ [Linux Files](#)

✓ [Linux Man Pages](#)

✓ [Linux File Contents](#)

✓ [Linux File Hierarchy](#)

✓ [Linux Filters](#)

⊕ [Linux Filters](#)

⊕ [Linux cat](#)

⊕ [Linux cut](#)

⊕ [Linux grep](#)

⊕ [Linux comm](#)

⊕ [Linux sed](#)

⊕ [Linux tee](#)

⊕ [Linux tr](#)

⊕ [Linux uniq](#)

⊕ [Linux wc](#)

⊕ [Linux od](#)

⊕ [Linux sort](#)

⊕ [Linux gzip](#)

✓ [Linux I/O Redirection](#)

✓ [Linux Unix Tools](#)

✓ [Linux Regex](#)

✓ [Linux Users](#)

✓ [Linux File Security](#)

Linux od

[← Prev](#)[Next →](#)

The 'od' term stands for octal dump. It displays content of a file in different human-readable formats like hexadecimal, octal and ASCII characters.

Syntax:

`od -b <fileName>` (display files in octal format)

`od -t x1 <fileName>` (display files in hexadecimal bytes format)

`od -c <fileName>` (display files in ASCII (backslashed) character format)

Example:

`od -b format.txt`

`od -t x1 format.txt`

`od -c format.txt`

sssit@JavaPoint: ~

sssit@JavaPoint:~\$ cat format.txt

12345

abcde

sssit@JavaPoint:~\$ od -b format.txt

0000000 061 062 063 064 065 012 111 112 113 114 115 012

- [✓ Linux Files](#)
- [✓ Linux Man Pages](#)
- [✓ Linux File Contents](#)
- [✓ Linux File Hierarchy](#)
- [✓ Linux Filters](#)
- [↳ Linux Filters](#)
- [↳ Linux cat](#)
- [↳ Linux cut](#)
- [↳ Linux grep](#)
- [↳ Linux comm](#)
- [↳ Linux sed](#)
- [↳ Linux tee](#)
- [↳ Linux tr](#)
- [↳ Linux uniq](#)
- [↳ Linux wc](#)
- [↳ Linux od](#)
- [↳ Linux sort](#)
- [↳ Linux gzip](#)
- [✓ Linux I/O Redirection](#)
- [✓ Linux Unix Tools](#)
- [✓ Linux Regex](#)
- [✓ Linux Users](#)
- [✓ Linux File Security](#)
- [✓ Linux Shell](#)
- [✓ Linux Networking](#)
- [✓ vi editor](#)

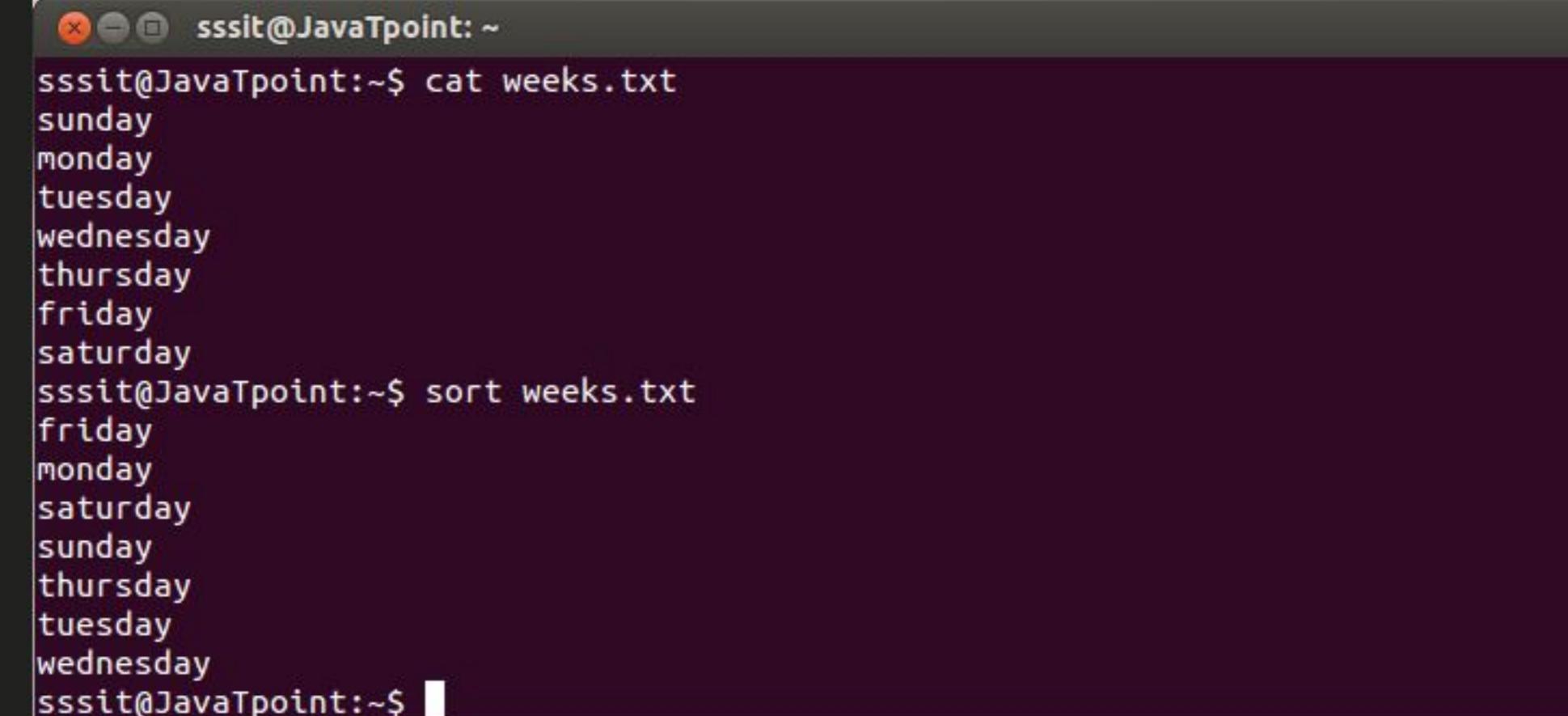
The 'sort' command sorts the file content in an alphabetical order.

Syntax:

```
sort <fileName>
```

Example:

```
sort weeks.txt
```



A screenshot of a terminal window titled "sssit@JavaTpoint: ~". The window shows the command "cat weeks.txt" followed by the days of the week: sunday, monday, tuesday, wednesday, thursday, friday, saturday. Below this, the command "sort weeks.txt" is run, resulting in the same days listed in alphabetical order: friday, monday, saturday, sunday, thursday, tuesday, wednesday.

```
sssit@JavaTpoint:~$ cat weeks.txt
sunday
monday
tuesday
wednesday
thursday
friday
saturday
sssit@JavaTpoint:~$ sort weeks.txt
friday
monday
saturday
sunday
thursday
tuesday
wednesday
sssit@JavaTpoint:~$
```

Look at the above snapshot, the 'sort' command has sorted the file 'weeks.txt' in alphabetical order.

+ New chat

□ I2C Communication

□ UART for Serial Com

□ Cortex-M4 OS Supp

□ Systick timer in Cort

□ Cortex-M4 Interrupt

Clear conversations

Upgrade to Plus NEW

Settings

Get help

[→ Log out



`gzip` is a command-line utility in Linux that is used to compress files. It is commonly used to reduce the size of large files, making it easier to transfer or store them. The compressed file has a **`.gz`** extension added to its name.

Here are some commonly used options with the **`gzip`** command:

- **`-c`**: Compress the input file to standard output and leave the original file unchanged.
- **`-d`**: Decompress a compressed file.
- **`-r`**: Recursively compress all files in a directory and its subdirectories.
- **`-k`**: Keep the original file and create a compressed file with a **`.gz`** extension.
- **`-v`**: Verbose output that displays the name and percentage reduction of each file being compressed.

Here are some examples of using the **`gzip`** command:

1. Compressing a file:

Stop generating

Copy code

Send a message...

+ New chat

□ I2C Communication

□ UART for Serial Com

□ Cortex-M4 OS Supp

□ Systick timer in Cort

□ Cortex-M4 Interrupt

Clear conversations

Upgrade to Plus NEW

Settings

Get help

[→ Log out

1. Compressing a file:

Copy code

```
gzip example.txt
```

This command compresses the file `example.txt` and creates a new compressed file called `example.txt.gz`.

2. Decompressing a file:

Copy code

```
gzip -d example.txt.gz
```

This command decompresses the compressed file `example.txt.gz` and creates a new file called `example.txt`.

3. Recursively compressing all files in a directory:

Stop generating

Copy code

Send a message...



..

+ New chat

□ I2C Communication

□ UART for Serial Com

□ Cortex-M4 OS Supp

□ Systick timer in Cort

□ Cortex-M4 Interrupt

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

```
gzip -d example.txt.gz
```

This command decompresses the compressed file `example.txt.gz` and creates a new file called `example.txt`.

3. Recursively compressing all files in a directory:

 Copy code

```
gzip -r myfolder
```

This command compresses all files in the directory `myfolder` and its subdirectories. The compressed files will have a `gz` extension.

4. Keeping the original file and creating a compressed file:

 Copy code

```
gzip -k example.txt
```

 Regenerate response

Send a message...



`gzip -r myfolder`

+ New chat

□ I2C Communication

□ UART for Serial Com

□ Cortex-M4 OS Supp

□ Systick timer in Cort

□ Cortex-M4 Interrupt

Clear conversations

Upgrade to Plus NEW

⚙ Settings

↗ Get help

[→] Log out

This command compresses all files in the directory `myfolder` and its subdirectories. The compressed files will have a `gz` extension.

4. Keeping the original file and creating a compressed file:

Copy code

`gzip -k example.txt`

This command compresses the file `example.txt` and creates a new compressed file called `example.txt.gz`. The original file `example.txt` is kept unchanged.

5. Verbose output:

Copy code

`gzip -v example.txt`

Regenerate response

This command compresses the file `example.txt` and displays a verbose output that shows

Send a message...



+ New chat

□ I2C Communication

□ UART for Serial Com

□ Cortex-M4 OS Supp

□ Systick timer in Cort

□ Cortex-M4 Interrupt

Clear conversations

Upgrade to Plus NEW

⚙️ Settings

↗ Get help

[→ Log out

This command compresses all files in the directory `myfolder` and its subdirectories. The compressed files will have a `gz` extension.

4. Keeping the original file and creating a compressed file:

Copy code

```
gzip -k example.txt
```

This command compresses the file `example.txt` and creates a new compressed file called `example.txt.gz`. The original file `example.txt` is kept unchanged.

5. Verbose output:

Copy code

```
gzip -v example.txt
```

This command compresses the file `example.txt` and displays a verbose output that shows the name and percentage reduction of the compressed file.

 Regenerate response

Send a message...



[+ New chat](#)[I2C Communication](#)[UART for Serial Com](#)[Cortex-M4 OS Supp](#)[Systick timer in Cort](#)[Cortex-M4 Interrupt](#)[Clear conversations](#)[Upgrade to Plus](#) NEW[Settings](#)[Get help](#)[Log out](#)

`awk` is a command-line tool used for text processing and data extraction in Linux. It is a powerful tool for working with structured text data, such as log files, CSV files, and other structured data formats.



The `awk` command processes each line of a file, and by default, separates fields based on whitespace. The syntax of the `awk` command is as follows:

arduino

[Copy code](#)

```
awk 'pattern {action}' input_file
```

Here's a brief explanation of the components of the syntax:

- `pattern`: This specifies the pattern to match in each line of the input file. If the pattern

[Regenerate response](#)

- + New chat
- I2C Communication
- UART for Serial Com
- Cortex-M4 OS Supp
- Systick timer in Cort
- Cortex-M4 Interrupt
- Clear conversations

Upgrade to **Plus** NEW

⚙️ Settings

↗️ Get help

[→] Log out

- `pattern`: This specifies the pattern to match in each line of the input file. If the pattern matches, the specified action is taken.
- `{action}`: This specifies the action to take when the pattern matches.
- `input_file`: This is the input file to process. If not specified, `awk` reads from standard input.

Here are some commonly used options with the `awk` command:

- `--F`: This specifies the field separator. The default field separator is whitespace.
- `--v`: This sets a variable to a value that can be used in the `awk` program.
- `--f`: This specifies the name of a file containing the `awk` program to run.

Here are some examples of using the `awk` command:

1. Print specific fields from a CSV file:

swift

⟳ Regenerate response

📋 Copy code

Send a message...

[+ Rename Folder in Linux](#)[✓ Linux Commands](#)[+ Linux commands List](#)[+ Commands with Example](#)[✓ Linux Users](#)[+ Linux Users](#)[+ Linux Create User](#)[+ Local su Commands](#)[+ Linux User Management](#)[+ User Password](#)[+ Local Groups](#)[+ Add User to Group](#)[+ Linux id Command](#)[✓ Linux Files](#)[+ Linux Files](#)[+ Linux File System](#)[+ Linux Create File](#)[+ Linux file](#)[+ Linux touch](#)[+ Linux rm](#)[+ Linux cp](#)[+ Linux mv](#)[+ Linux rename](#)[+ Linux install Command](#)[+ Linux read Command](#)[✓ Linux File Contents](#)

How is it named as AWK?

This command is named by using the first letter of the name of three people who wrote the original version of this command in 1977. Their names are **Alfred Aho, Peter Weinberger, and Brian Kernighan** and they were from **AT & T Bell Laboratories**.

Features of AWK command

Various features of the Awk command are as follows:

- It scans a file line by line.
- It splits a file into multiple fields.
- It compares the input text or a segment of a text file.
- It performs various actions on a file like searching a specified text and more.
- It formats the output lines.
- It performs arithmetic and string operations.
- It applies the conditions and loops on output.
- It transforms the files and data on a specified structure.
- It produces the format reports.

Syntax:

The Awk command is used as follows:

`awk options 'selection _criteria {action}' input-file > output-file`

- [↪ Linux Files](#)
- [↪ Linux File System](#)
- [↪ Linux Create File](#)
- [↪ Linux file](#)
- [↪ Linux touch](#)
- [↪ Linux rm](#)
- [↪ Linux cp](#)
- [↪ Linux mv](#)
- [↪ Linux rename](#)
- [↪ Linux install Command](#)
- [↪ Linux read Command](#)

- ✓ Linux File Contents**
- [↪ Linux File Contents](#)
 - [↪ Open File in Linux](#)
 - [↪ Linux Edit File](#)
 - [↪ Linux which command](#)
 - [↪ Linux head](#)
 - [↪ Linux tail](#)
 - [↪ Linux cat](#)
 - [↪ Linux tac](#)
 - [↪ Linux more](#)
 - [↪ Linux less](#)
 - [↪ Linux strings command](#)
 - [↪ Linux Print](#)

✓ Linux Process

- [↪ Kill Process Linux](#)
- [↪ Linux terminating](#)

[↑ SCROLL TO TOP](#)

- It performs various actions on a file like searching a specified text and more.
- It formats the output lines.
- It performs arithmetic and string operations.
- It applies the conditions and loops on output.
- It transforms the files and data on a specified structure.
- It produces the format reports.

Syntax:

The Awk command is used as follows:

```
awk options 'selection _criteria {action }' input-file > output-file
```

The options can be:

- **-f program files:** It reads the source code of the script written on the awk command
- **-F fs:** It is used as the input field separator.

AWK program structure

The AWK program is a collection of the pairs of pattern actions, specified as:

```
condition {action}
```

```
condition {action}
```

...

- [+ Linux dpkg command](#)
- [+ Latest Kernel Linux Version](#)
- [+ Linux Package Manager](#)
- [+ Linux proc](#)
- [+ Linux Task Manager](#)
- [+ Linux Signals](#)
- [+ Parrot OS vs. Kali Linux](#)
- [+ Ubuntu Features](#)
- [+ Linux ISO](#)
- [+ Putty Command in Linux](#)
- [+ Check OS Version in Linux](#)
- [+ Linux Change Timezone](#)
- [+ Linux Load Average](#)
- [+ Linux Firewall](#)
- [+ RPM Command in Linux](#)
- [+ Cron Command](#)
- [+ Linux Change Hostname](#)
- [+ Ubuntu Disable Firewall](#)
- [+ Latest Release of Ubuntu](#)
- [+ Ubuntu Logo](#)
- [+ Ubuntu MATE](#)
- [+ Ubuntu Nvidia Drivers](#)
- [+ Zoom Ubuntu](#)
- [+ Install tar.gz Ubuntu](#)
- [+ Ubuntu Screen Recorder](#)
- [+ Ubuntu Server](#)
- [+ Ubuntu Wallpaper](#)
- [+ xVideoServiceThief Ubuntu Linux Free](#)

Arya IT
Mike ECE
Helena ECE

Built-in variables in AWK command

Awk command supports many built-in variables, which include \$1, \$2, and so on, that break the file content into individual segments.

NR: It is used to show the current count of the lines. The awk command performs action once for each line. These lines are said as records.

NF: It is used to count the number of fields within the current database.

FS: It is used to create a field separator character to divide fields into the input lines.

OFS: It is used to store the output field separator. It separates the output fields.

ORS: It is used to store the output record separator. It separates the output records. It prints the content of the ORS command automatically.

Example4: Print the output and display the line number.

To display the line number in output, use the NR variable with the Awk command as follows:

```
awk '{print NR,$0}' student.txt
```

Consider the below output:

```
javatpoint@javatpoint-GB-BXBT-2807:~$ awk '{print NR,$0}' student.txt
```

1 Sam CS

+ New chat

□ I2C Communication

□ UART for Serial Com

□ Cortex-M4 OS Supp

□ Systick timer in Cort

□ Cortex-M4 Interrupt

Clear conversations

Upgrade to Plus NEW

⚙️ Settings

↗️ Get help

[→] Log out

sources files and the build process for each target. The Makefile contains rules that specify how to build each target, along with any dependencies that need to be built first.

Here are some commonly used options with the `make` command:

- `all`: This builds all targets specified in the Makefile.
- `--f`: This specifies the name of the Makefile to use.
- `--j`: This specifies the number of parallel jobs to run. For example, `--j4` runs up to 4 parallel jobs.
- `clean`: This removes all generated files and returns the build directory to its original state.
- `install`: This installs the built files to the system.
- `uninstall`: This removes the installed files from the system.

Here are some examples of using the `make` command:

1. Build all targets specified in the Makefile:

css

Copy code

⟳ Regenerate response

make all

Send a message...

+ New chat

□ I2C Communication

□ UART for Serial Com

□ Cortex-M4 OS Supp

□ Systick timer in Cort

□ Cortex-M4 Interrupt

Clear conversations

Upgrade to Plus NEW

Settings

Get help

[→ Log out

2. Build a specific target:

go

Copy code

make **target_name**

This command builds the specified target, where `target_name` is the name of the target specified in the Makefile.

3. Clean the build directory:

go

Copy code

make clean

This command removes all generated files and returns the build directory to its original state.

4. Install the built files to the system:

go

Copy code

make install

⟳ Regenerate response

Send a message...



[+ Linux File Contents](#)[+ Open File in Linux](#)[+ Linux Edit File](#)[+ Linux which command](#)[+ Linux head](#)[+ Linux tail](#)[+ Linux cat](#)[+ Linux tac](#)[+ Linux more](#)[+ Linux less](#)[+ Linux strings command](#)[+ Linux Print](#)

✓ Linux Process

[+ Kill Process Linux](#)[+ Linux terminating](#)[+ 1\) Linux kill](#)[+ 2\) Linux killall](#)[+ 3\) Linux pkill](#)[+ 4\) Linux xkill](#)[+ Nohup Command](#)[+ Linux at command](#)[+ Linux exit command](#)

✓ Linux File Hierarchy

[+ Linux File Hierarchy](#)[+ Binary Directories](#)[+ Configuration Directories](#)[+ Data Directories](#)[+ Memory Directories](#)[↑ SCROLL TO TOP](#)

follows:

- **-b, -m:** These options are used to ignore the compatibility for the different versions of the make command.
- **-B, --always-make:** These options are used to unconditionally make all targets.
- **-C dir, --directory=dir:** These options are used to change the directory before executing the makefile.
- **-d:** It is used to print the debugging information.
- **--debug[=FLAGS]:** It is used to print the debugging information along with normal processing. If we skip the flag, then it will show similar results as the '-d' option.
- **-e, --environment-overrides:** It is used to provide the variables taken from the environment precedence to makefile.
- **-f file, --file=file, --makefile=FILE:** It is used to use a file as a makefile.
- **-i, --ignore-errors:** The '-i' option is used to ignore all errors in commands.
- **-I dir, --include-dir=dir:** It is used to specify a directory to search for the specified makefile. If we specify the many '-I' options, it will search in the many directories in the order are specified.
- **-j [jobs], --jobs[=jobs]:** It is used to specify the number of jobs to run simultaneously. If we provide many '-j' option, the last one will be considered to execute. If we do not specify the number of jobs, it will not limit the jobs that can run simultaneously.
- **-k, --keep-going:** It is used to continue the program as much as possible after getting an error.
- **-l [load], --load-average[=load]:** It is used to specify that no new task should be started if other tasks are in the queue and the load average is at its minimum.
- **-n, --just-print, --dry-run, --recon:** It is used to display the command that would be run.
- **-o file, --old-file=file, --assume-old=file:** It is used to assure that the make will not remake the

[+ Linux uniq](#)[+ Linux wc](#)[+ Linux od](#)[+ Linux sort](#)[+ Linux gzip](#)

✓ AWK Command

[+ Linux AWK Command](#)

✓ Programming with Linux

[+ Linux make command](#)[+ GCC Linux](#)

✓ Linux I/O Redirection

[+ Linux I/O Redirection](#)[+ Linux Input Redirection](#)[+ Linux Output Redirection](#)[+ Linux Error Redirection](#)

✓ Linux Unix Tools

[+ Linux Unix Tools](#)[+ Linux Find](#)[+ Linux Locate](#)[+ Linux Date](#)[+ Linux Cal](#)[+ Linux Sleep](#)[+ Linux Time](#)[+ Linux zcat/zmore](#)[+ Linux bzip2 / bunzip2](#)[+ Linux bzcat / bzmore](#)[↑ SCROLL TO TOP](#)[Linux Regex](#)

- **-q, --question:** The '-q' option is used for the **Question mode**. It will not run any command or print anything. It will only return an exit status is zero if the specified target is already synced; otherwise, it will display a non-zero exit status.

- **-r, --no-builtin-rules:** It is used to eliminate the use of the built-in implicit rules.

- **-R, --no-builtin-variables:** It is useful if we do not want to define any built-in variable.

- **-s, --silent, --quiet:** These options are called **Silent operation**. It restricts to print the commands as they are executed.

- **-S, --no-keep-going, --stop:** It is used to cancel the effect of "-k, --keep-going" operation.

- **-t, --touch:** It is used to touch files instead of running their commands.

- **--trace:** It is used to trace the disposition of each target.

- **-v, --version:** It is used to print the installed version of the make utility. Further, it displays a list of authors, copyright, and some notice regarding the make utility.

- **-w, --print-directory:** It is used to track a print message that contains a working directory before and after other processing. It is useful to track down errors from the complicated structure of recursive make command.

- **--no-print-directory:** It is used to turn off the '-w' option.

- **-W file, --what-if=file, --new-file=file, --assume-new=file:** These options pretend that the target file has just been modified.

- **--warn-undefined-variables:** This option is used to warn that an undefined variable is referenced.

Let's understand some examples of the make command. We will see the basic use of makefile, and further, we will create some c++ programs and a makefile. We will perform some operations on them to better understand the concept.

+ New chat

□ I2C Communication

□ UART for Serial Com

□ Cortex-M4 OS Supp

□ Systick timer in Cort

□ Cortex-M4 Interrupt

Clear conversations

Upgrade to Plus NEW

⚙ Settings

↗ Get help

[→ Log out

```
# This is a comment

# The target is named "program"
program: main.o functions.o
    gcc -o program main.o functions.o

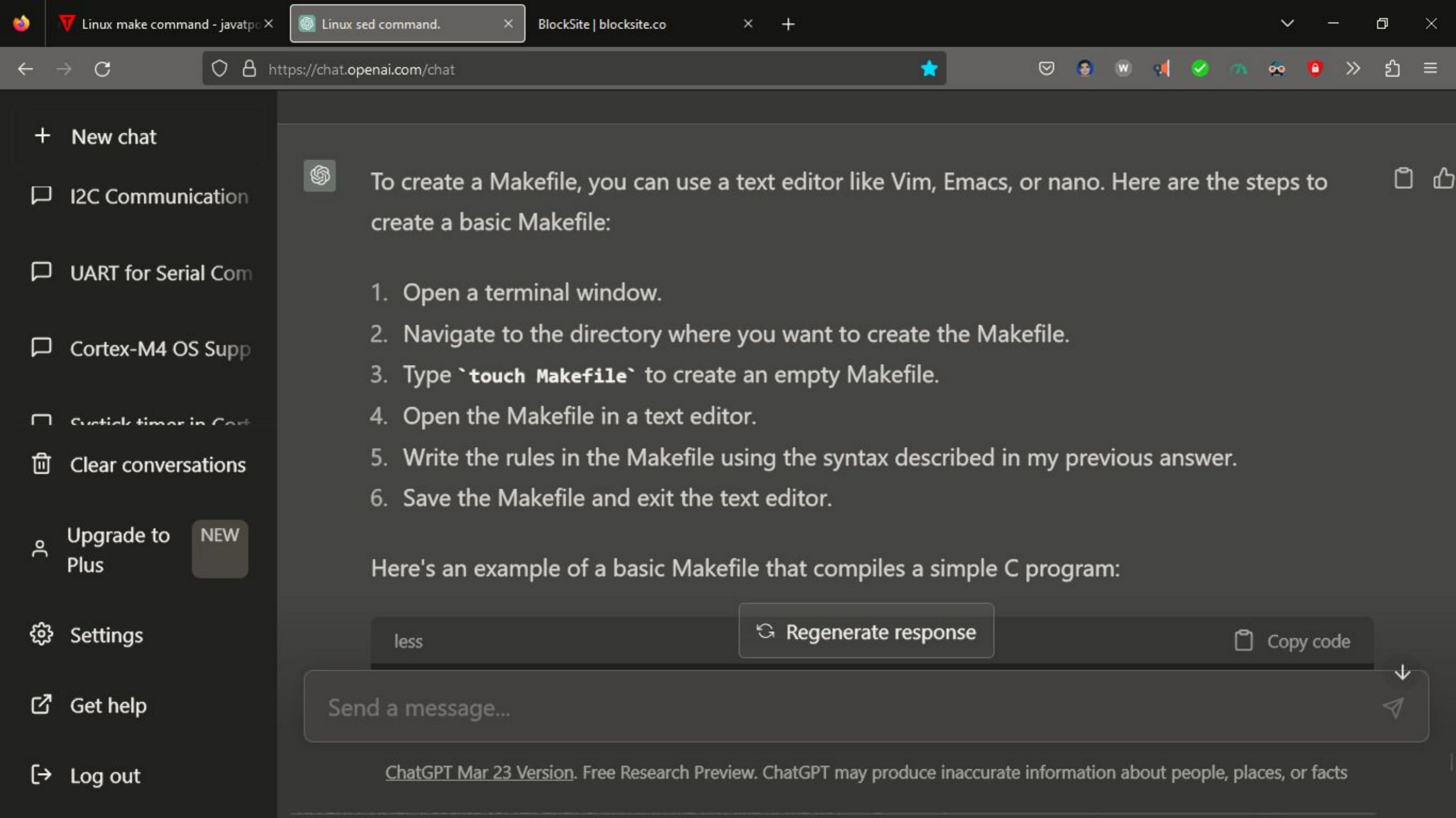
# The first dependency is "main.o"
main.o: main.c
    gcc -c main.c

# The second dependency is "functions.o"
functions.o: functions.c
    gcc -c functions.c

# The "clean" target removes all generated files
clean:
    rm -f program *.o
```

⟳ Regenerate response

Send a message...



- ↪ Linux ls
- ↪ Linux mkdir
- ↪ Linux rmdir
- ↪ Rename Folder in Linux

✓ Linux Commands

- ↪ Linux commands List
- ↪ Commands with Example

✓ Linux Users

- ↪ Linux Users
- ↪ Linux Create User
- ↪ Local su Commands
- ↪ Linux User Management
- ↪ User Password
- ↪ Local Groups
- ↪ Add User to Group
- ↪ Linux id Command

✓ Linux Files

- ↪ Linux Files
- ↪ Linux File System
- ↪ Linux Create File
- ↪ Linux file
- ↪ Linux touch
- ↪ Linux rm
- ↪ Linux cp
- ↪ Linux mv
- ↪ Linux rename

Components of GCC

GCC is a portable tool, and it can run on many operating systems. Also, it can be ported to **Windows** by using some tools such as **Cygwin**, **MinGW**, and **MinGW-W64**. As it is a key component of GNU toolchain, it contains the following components for developing applications and operating systems:

GNU Compiler Collection (GCC): It is a compiler suite for many languages, such as **C/C++** and **Objective-C/C++**.

GNU Make: It is an automation tool that is used to compile and build applications.

GNU Binutils: It is a suite that contains binary utility tools such as **linker** and **assembler**.

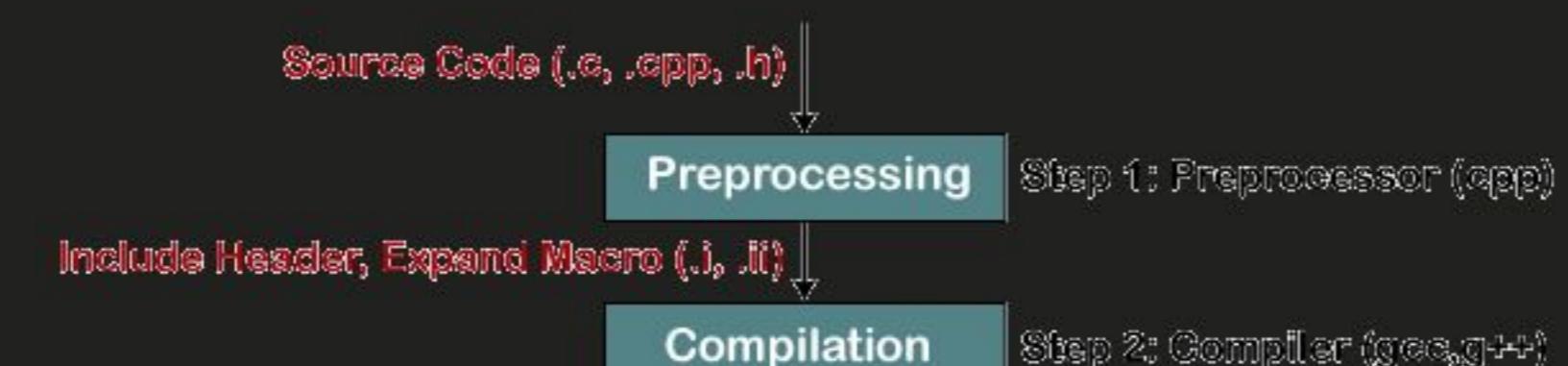
GNU Debugger (GDB). It is used to debug the applications.

GNU Autotools: It is a build system that contains some utility tools such as Autoconf, Autoheader, Automake, and Libtool.

GNU Bison: It is a parser generator as lex and yacc.

GCC Compiler Process

The GCC compiles and makes executable a program in the following four steps:



[+ Linux User Management](#)[+ User Password](#)[+ Local Groups](#)[+ Add User to Group](#)[+ Linux id Command](#)[✓ Linux Files](#)[+ Linux Files](#)[+ Linux File System](#)[+ Linux Create File](#)[+ Linux file](#)[+ Linux touch](#)[+ Linux rm](#)[+ Linux cp](#)[+ Linux mv](#)[+ Linux rename](#)[+ Linux install Command](#)[+ Linux read Command](#)[✓ Linux File Contents](#)[+ Linux File Contents](#)[+ Open File in Linux](#)[+ Linux Edit File](#)[+ Linux which command](#)[+ Linux head](#)[+ Linux tail](#)[+ Linux cat](#)[+ Linux tac](#)[+ Linux more](#)[+ -----](#)[↑ SCROLL TO TOP](#)

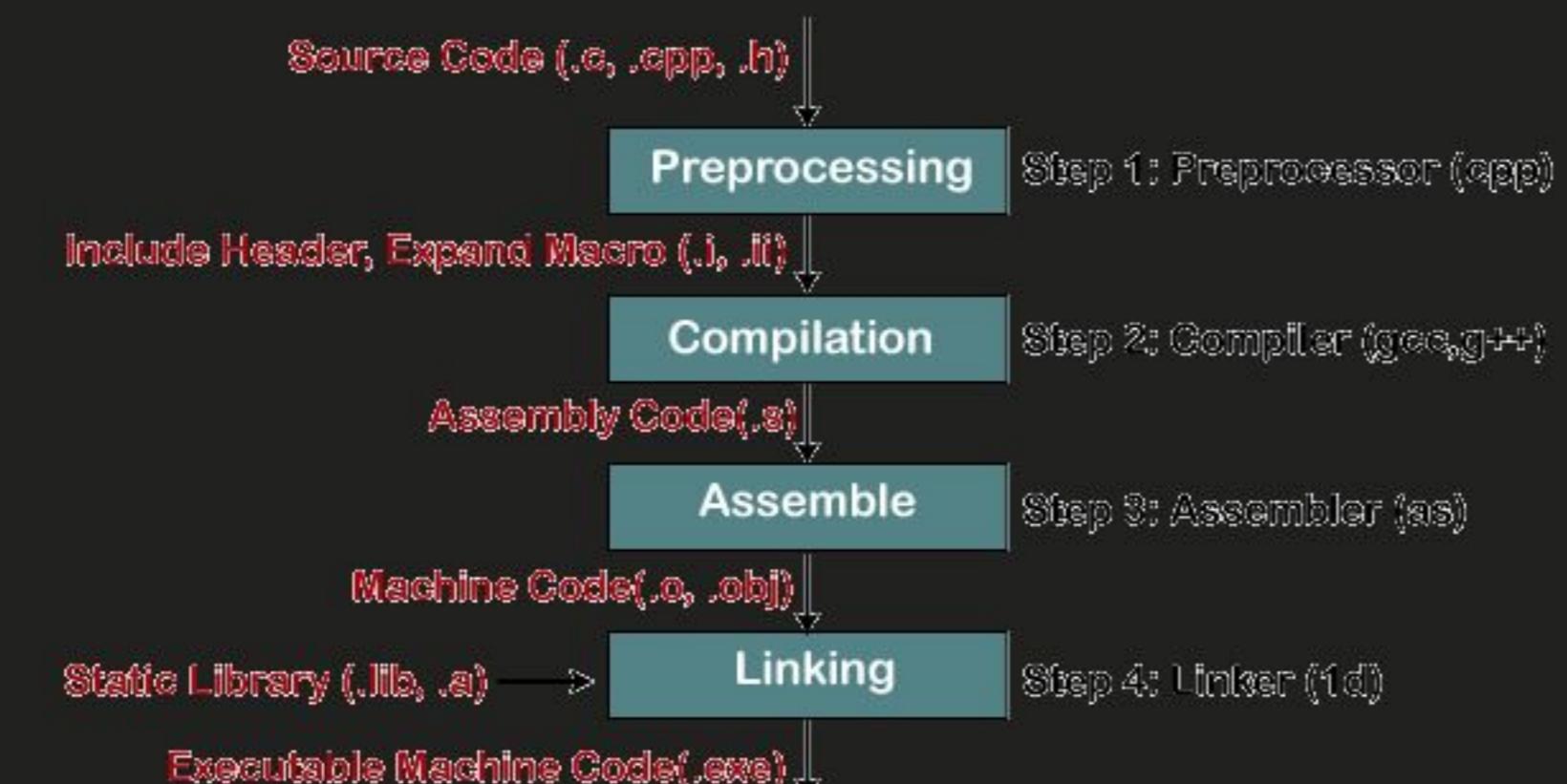
GNU Debugger (GDB). It is used to debug the applications.

GNU Autotools: It is a build system that contains some utility tools such as Autoconf, Autoheader, Automake, and Libtool.

GNU Bison: It is a parser generator as lex and yacc.

GCC Compiler Process

The GCC compiles and makes executable a program in the following four steps:



Pre-processing: In this step, the GNU preprocessor (cpp.exe) creates an intermediate file having the source code. The preprocessor includes the headers (#include) and expands the macros (#define). Consider the below command:

The output of the compilation phase is an object file, which contains the machine code and other information needed for linking.

+ New chat

Run C Program  

Linux sed command.

New chat

Linux grep command

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

1. Linking:

The linking phase is where the object files produced by the compiler are combined together into a single executable file.

- The linker is a program that combines object files into an executable file. It resolves external references between object files, such as function calls and variable references.
- The linker is typically invoked automatically by the C compiler, which passes it a list of object files to link together.
- The output of the linking phase is an executable file, which can be executed directly by the computer.

1. Running:

Finally, the executable file can be run  to execute the program.

Send a message...

- ✓ [Linux Files](#)
- ✓ [Linux Man Pages](#)
- ✓ [Linux File Contents](#)
- ✓ [Linux File Hierarchy](#)
- ✓ [Linux Filters](#)
- ✓ [Linux I/O Redirection](#)
- ✓ [Linux Unix Tools](#)
 - ⊕ [Linux Unix Tools](#)
 - ⊕ [Linux Find](#)
 - ⊕ [Linux Locate](#)
 - ⊕ [Linux Date](#)
 - ⊕ [Linux Cal](#)
 - ⊕ [Linux Sleep](#)
 - ⊕ [Linux Time](#)
 - ⊕ [Linux zcat/zmore](#)
 - ⊕ [Linux bzip2 / bunzip2](#)
 - ⊕ [Linux bzcat / bzmore](#)
 - ⊕ [Linux df](#)
- ✓ [Linux Regex](#)
- ✓ [Linux Users](#)
- ✓ [Linux File Security](#)
- ✓ [Linux Shell](#)
- ✓ [Linux Networking](#)
- ✓ [vi editor](#)
- ✓ [Linux System Admin](#)
- ✓ [100+ Linux Commands](#)

↑ SCROLL TO TOP

Look at the above snapshot, command "**zcat acb**" displays file 'acb' while cat command shows an error.

Note: The command **zmore** and **zless** works same for zipped files as more and less command works for unzipped files.

```
sssit@JavaTpoint:~$ ls
abc Desktop Disk1 Downloads Pictures Templates xyz
acb.gz df Documents Music Public Videos
sssit@JavaTpoint:~$ zcat acb
fg / 45
dg / 56
rl / 79
go / 69
sssit@JavaTpoint:~$ cat acb
cat: acb: No such file or directory
sssit@JavaTpoint:~$
```

- ✓ [Introduction to Linux](#)
- ✓ [Linux Directories](#)
- ✓ [Linux Files](#)
- ✓ [Linux Man Pages](#)
- ✓ [Linux File Contents](#)
- ✓ [Linux File Hierarchy](#)
- ✓ [Linux Filters](#)
- ✓ [Linux I/O Redirection](#)

✓ [Linux Unix Tools](#)

- ⊕ [Linux Unix Tools](#)
- ⊕ [Linux Find](#)
- ⊕ [Linux Locate](#)
- ⊕ [Linux Date](#)
- ⊕ [Linux Cal](#)
- ⊕ [Linux Sleep](#)
- ⊕ [Linux Time](#)
- ⊕ [Linux zcat/zmore](#)
- ⊕ [Linux bzip2 / bunzip2](#)
- ⊕ [Linux bzcat / bzmore](#)
- ⊕ [Linux df](#)

✓ [Linux Regex](#)

- ✓ [Linux Users](#)
- ✓ [Linux File Security](#)
- ✓ [Linux Shell](#)
- ✓ [Linux Networking](#)

df Command in Linux/Unix with Examples

[← Prev](#)[Next →](#)

Linux df command is used to display the **disk space used in the file system**. The 'df' stands for "**disk filesystem**." It defines the number of blocks used, the number of blocks available, and the directory where the file system is mounted.

Syntax:

```
df [OPTION]... [FILE]...
```

Options:

-a, --all: It is used to include pseudo, duplicate, remote file systems.

-B, --block-size=SIZE: It is used to scale sizes by SIZE before printing them, for example, the '-BM' option prints sizes in units of 1,048,576 bytes.

-h, --human-readable: It is used to display sizes in powers of 1024 (e.g., 1023M).

-H, --si: It is used to show sizes in powers of 1000 (e.g., 1.1G)

-i, --inodes: It is used to list inode information instead of block usage

-l, --local: It is used to limit the listing to local file systems.

--no-sync: It is used for not invoking sync before getting usage info (default).

[↑ SCROLL TO TOP](#)

[Linux Cat](#)[Linux Sleep](#)[Linux Time](#)[Linux zcat/zmore](#)[Linux bzip2 / bunzip2](#)[Linux bzcat / bzmore](#)[Linux df](#)[✓ Linux Regex](#)[✓ Linux Users](#)[✓ Linux File Security](#)[✓ Linux Shell](#)[✓ Linux Networking](#)[✓ vi editor](#)[✓ Linux System Admin](#)[✓ 100+ Linux Commands](#)[✓ Linux Interview](#)[✓ Misc](#)[⊕ Linux Create User](#)[⊕ Linux AWK Command](#)

-a, --all: It is used to include pseudo, duplicate, remote file systems.

-B, --block-size=SIZE: It is used to scale sizes by SIZE before printing them, for example, the '-BM' option prints sizes in units of 1,048,576 bytes.

-h, --human-readable: It is used to display sizes in powers of 1024 (e.g., 1023M).

-H, --si: It is used to show sizes in powers of 1000 (e.g., 1.1G)

-i, --inodes: It is used to list inode information instead of block usage

-l, --local: It is used to limit the listing to local file systems.

--no-sync: It is used for not invoking sync before getting usage info (default).

--output[=FIELD_LIST]: This option used if we want to use the output format defined by FIELD_LIST or print all fields if FIELD_LIST is omitted.

-P, --portability: It is used to use the POSIX output format.

--total: It is used to exclude all entries insignificant to available space, and produce a total.

-t, --type=TYPE: It is used to limit the listing to file systems of type TYPE.

-T, --print-type: It is used to display the file system type.

-x, --exclude-type=TYPE: It is used to limit the listing to file systems, not of type TYPE.

--help: It is used to display the help manual having brief information about the supported options.

--version: It is used to display the version information of the df command.

-x, --exclude-type=TYPE: It is used to limit the listing to file systems, not of type TYPE.

--help: It is used to display the help manual having brief information about the supported options.

--version: It is used to display the version information of the df command.

Examples of the df command

Let's see the following examples of the df command:

- Display the disk space usage
- Display the disk space usage in a human-readable form
- Display the file system type
- Display specific file system types
- Exclude the particular file system types
- Display available space and mount point for a folder

Display the Disk Space Usage

To display the disk space usage, execute the df command without any argument. It will show the disk space usage in a tabular form. The df command is useful for discovering the available free space on a system or file system. Execute the below command:

```
df
```

The above command will produce the output as follows:

\b	Matches empty string at the edge of a word.
\B	Matches empty string if it is not at the edge of a word.
\<	Matches empty string at the beginning of a word.
\>	Matches empty string at the end of a word.

Regex Versions

There are three versions of regular expressions syntax:

- BRE : Basic Regular Expressions
- ERE : Extended Regular Expressions
- PRCE: Perl Regular Expressions

Depending on tool or programs, one or more of these versions can be used.

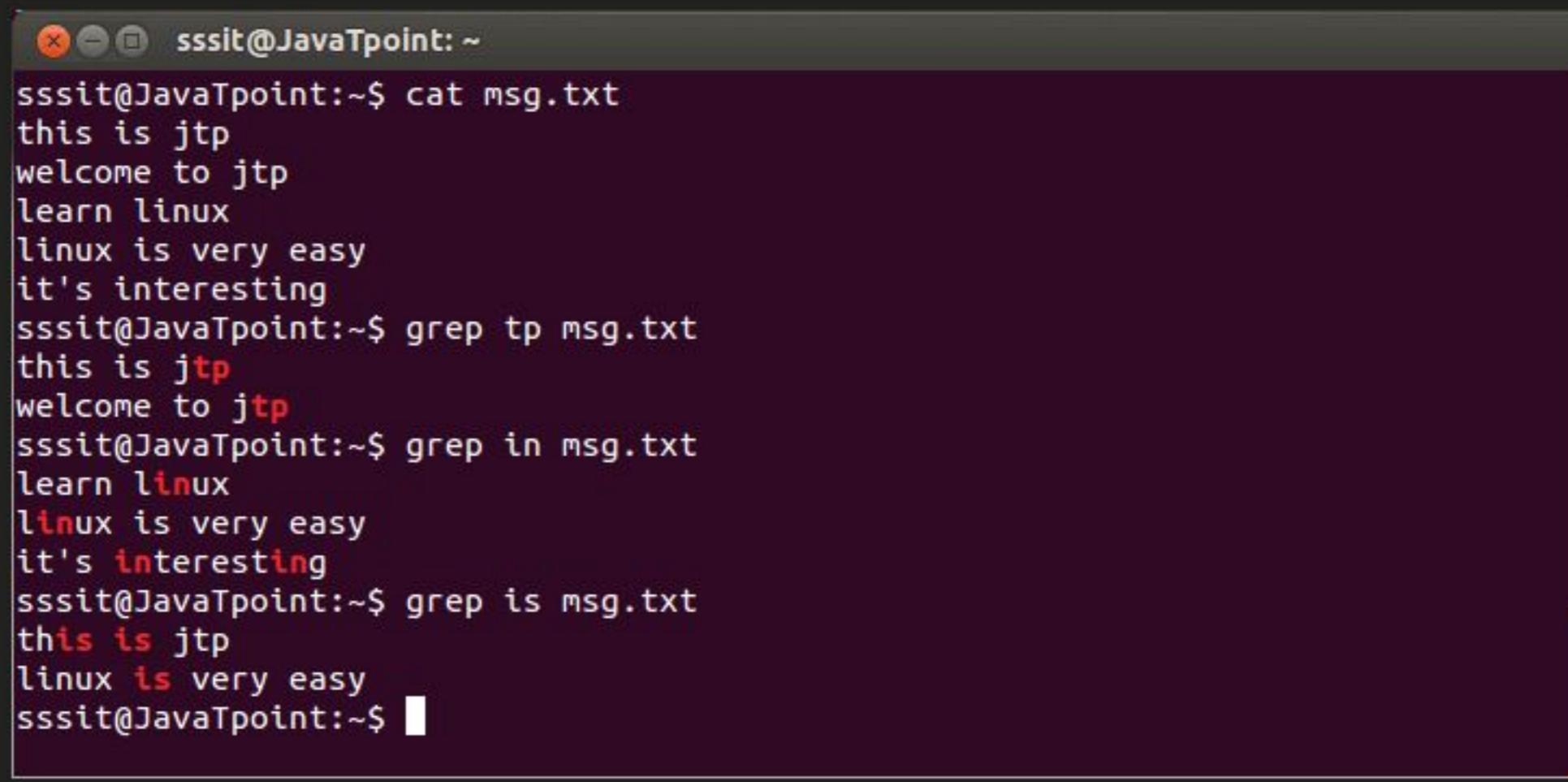
[Next Topic](#)[Linux grep Regular Expressions](#)[← Prev](#)[Next →](#)

Concatenating Characters

If a pattern is of concatenating characters then it has to be matched as it is, for the line to be displayed.

Example:

```
grep tp msg.txt  
grep in msg.txt  
grep is msg.txt
```

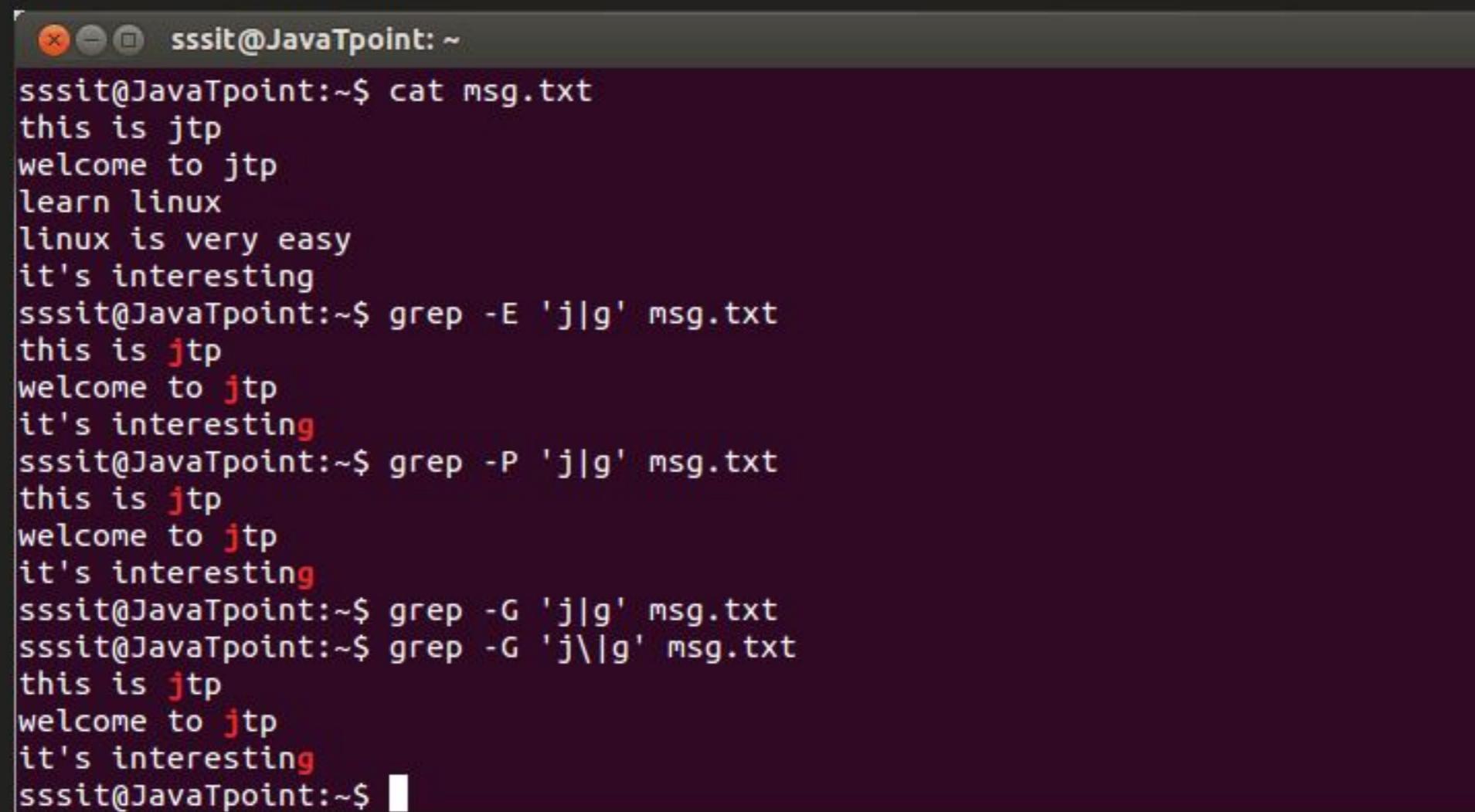


The screenshot shows a terminal window with a dark background and light-colored text. It displays the following command-line session:

```
sssit@JavaPoint: ~  
sssit@JavaPoint:~$ cat msg.txt  
this is jtp  
welcome to jtp  
learn linux  
linux is very easy  
it's interesting  
sssit@JavaPoint:~$ grep tp msg.txt  
this is jtp  
welcome to jtp  
sssit@JavaPoint:~$ grep in msg.txt  
learn linux  
linux is very easy  
it's interesting  
sssit@JavaPoint:~$ grep is msg.txt  
this is jtp  
linux is very easy  
sssit@JavaPoint:~$
```

Example:

```
grep -E 'j|g' msg.txt  
grep -P 'j|g' msg.txt  
grep -G 'j\|g' msg.txt
```



The screenshot shows a terminal window titled "sssit@JavaTpoint:~". It displays the following session:

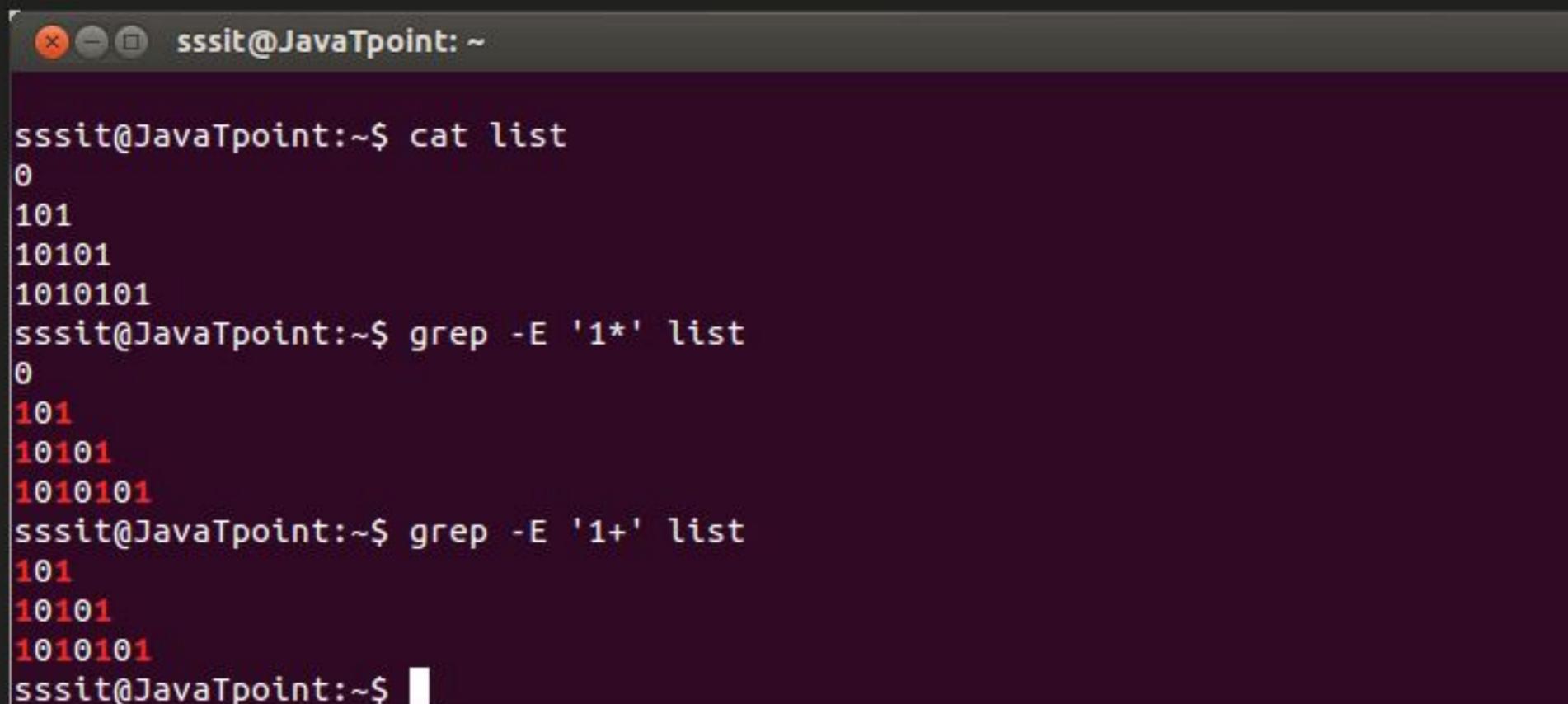
```
sssit@JavaTpoint:~$ cat msg.txt  
this is jtp  
welcome to jtp  
learn linux  
linux is very easy  
it's interesting  
sssit@JavaTpoint:~$ grep -E 'j|g' msg.txt  
this is jtp  
welcome to jtp  
it's interesting  
sssit@JavaTpoint:~$ grep -P 'j|g' msg.txt  
this is jtp  
welcome to jtp  
it's interesting  
sssit@JavaTpoint:~$ grep -G 'j|g' msg.txt  
sssit@JavaTpoint:~$ grep -G 'j\|g' msg.txt  
this is jtp  
welcome to jtp  
it's interesting  
sssit@JavaTpoint:~$
```

Look at the above snapshot, either pattern 'j' or 'g' should be matched to display the lines.

```
grep <option> <'pattern'*> <fileName>
```

Example:

```
grep -E '1*' list  
grep -E '1+' list
```



The screenshot shows a terminal window titled "sssit@JavaTpoint:~". It displays two commands and their outputs. The first command, "cat list", shows the file "list" containing the binary digits "0", "101", "10101", and "1010101". The second command, "grep -E '1*' list", highlights all occurrences of the pattern "1" in red, including the entire file content. The third command, "grep -E '1+' list", highlights only the first occurrence of the pattern "1" in red, which is the digit "101".

```
sssit@JavaTpoint:~$ cat list  
0  
101  
10101  
1010101  
sssit@JavaTpoint:~$ grep -E '1*' list  
0  
101  
10101  
1010101  
sssit@JavaTpoint:~$ grep -E '1+' list  
101  
10101  
1010101  
sssit@JavaTpoint:~$
```

Look at the above snapshot, * character displays zero or more times occurrence of pattern '1'. But + character displays one or more times occurrence.

+ New chat

Run C Program  

AB

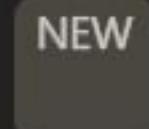
all regular expression list in linux

Linux sed command.

New chat

Linux grep command

Clear conversations

Upgrade to Plus 

Settings

Get help

Log out



Here is a list of some commonly used regular expressions in Linux:



1. Character classes:

- `'[abc]'`: matches any of the characters a, b, or c.
- `'[0-9]'`: matches any digit from 0 to 9.
- `'[a-z]'`: matches any lowercase letter from a to z.
- `'[A-Z]'`: matches any uppercase letter from A to Z.
- `'.'`: matches any character except a newline.

1. Anchors:

 Regenerate response

Send a message...



+ New chat

Run C Progra

Linux sed command.

New chat

Linux grep comman

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

1. Anchors:

- `^`: matches the beginning of a line.
- `\$`: matches the end of a line.

1. Quantifiers:

- `*`: matches zero or more occurrences of the preceding pattern.
- `+`: matches one or more occurrences of the preceding pattern.
- `?`: matches zero or one occurrence of the preceding pattern.
- `{n}`: matches exactly n occurrences of the preceding pattern.
- `{n,m}`: matches between n and m occurrences of the preceding pattern.

1. Grouping:

- `(pattern)`: groups the pattern **together**.
- `(?:pattern)`: groups the pattern **together**, but does not create a capture group.

Send a message...



+ New chat

Run C Progra ✎ ✖

Linux sed command.

New chat

Linux grep comman

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out



Here is a list of the most commonly used parameters in regular expressions in Linux:

1. Character classes:

- `'[characters]'`: matches any of the characters inside the brackets.
- `'^[characters]'`: matches any character that is not inside the brackets.
- `':[:class:]'`: matches any character belonging to the specified character class.

1. Anchors:

- `'^`*: matches the beginning of a line or string.
- `'\$`*: matches the end of a line or string.
- `'\b`*: matches a word boundary.
- `'\B`*: matches a non-word boundary.

1. Quantifiers:

Stop generating

Send a message...

..

+ New chat

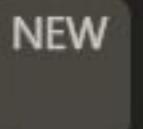
Run C Progra  

Linux sed command.

New chat

Linux grep comman

Clear conversations

Upgrade to Plus 

Settings

Get help

Log out

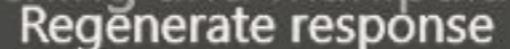
1. Escaping:

- `\\` escapes the next character to match it literally.

1. Special characters:

- `.`: matches any character except a newline.
- `\\s`: matches any whitespace character.
- `\\S`: matches any non-whitespace character.
- `\\d`: matches any digit character.
- `\\D`: matches any non-digit character.
- `\\w`: matches any word character (letters, digits, or underscore).
- `\\W`: matches any non-word character.

These are just a few of the parameters that can be used in regular expressions in Linux. Regular expressions are a powerful tool for searching and manipulating text, and understanding these parameters can make working with regular expressions much more effective.

 Regenerate response

Send a message...



1. Grouping and capture:

- `(`): groups and captures the pattern inside the parentheses.
- `(?:)`: groups the pattern inside the parentheses, but does not capture it.
- `(?<name>)`: names the group inside the parentheses.
- `\N`: matches the text captured by the Nth group.

1. Alternation:

- `|`: matches either the pattern on the left or the pattern on the right.

1. Escaping:

- `\\`: escapes the next character to match it literally.

1. Special characters:

- `.`: matches any character except a newline.

Send a message...



+ New chat

Run C Progra

Linux sed command

New chat

Linux grep command

Clear conversations

Upgrade to Plus

Settings

Get help

Log out

- ``\$``: matches the end of a line or string.
- ``\b``: matches a word boundary.
- ``\B``: matches a non-word boundary.

+ New chat

Run C Program  

Linux sed command

New chat

Linux grep command

Clear conversations

Upgrade to Plus 

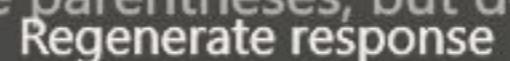
Settings

Get help

1. Quantifiers:

- ``*``: matches zero or more occurrences of the preceding pattern.
- ``+``: matches one or more occurrences of the preceding pattern.
- ``?``: matches zero or one occurrence of the preceding pattern.
- ``{n}``: matches exactly n occurrences of the preceding pattern.
- ``{n,}``: matches at least n occurrences of the preceding pattern.
- ``{n,m}``: matches between n and m occurrences of the preceding pattern.

1. Grouping and capture:

- ``()``: groups and captures the pattern inside the parentheses.
- ``(?:)`": groups the pattern inside the parentheses, but does not capture it. 
- ``(?<name>)``: names the group inside the parentheses.

Send a message...



Log out

+ New chat

chmod com ✎ ✖

Linux File Permission

Run C Program in Ba

Linux sed command.

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

The `chown` command is used in Linux to change the ownership of a file or directory. Here are the steps to use the `chown` command:

1. Open a terminal: To use the `chown` command, you need to open a terminal in Linux.
2. Identify the current owner of the file or directory: Before changing the ownership of a file or directory, you should identify the current owner of the file or directory. You can do this by using the `ls -l` command. For example, to view the ownership of a file named `example.txt`, you can use the following command:

bash

Copy code

```
ls -l example.txt
```

This command will display the file permissions, ownership, and other details of the file. The output will be something like this:

csharp

Copy code

```
-rw-r--r-- 1 user group 1234 Mar 10 12:34 example.txt
```

⟳ Regenerate response

Send a message...

csharp

 Copy code

+ New chat

 chmod com  

 Linux File Permission

 Run C Program in Ba

 Linux sed command.

 Clear conversations

 Upgrade to 
Plus

 Settings

 Get help

[→ Log out

-rw-r--r-- 1 user group 1234 Mar 10 12:34 example.txt

In this example, the user named `user` is the owner of the file, and the group named `group` is the group owner of the file.

3. Decide on the new owner and group: Decide on the new owner and group for the file or directory. You can use either a username or a numeric user ID (UID) to specify the user, and either a group name or a numeric group ID (GID) to specify the group.
4. Use the chown command to change ownership: Once you have identified the current owner and decided on the new owner and group, you can use the `chown` command to change the ownership of the file or directory. The basic syntax of the `chown` command is as follows:

bash

 Copy code

chown new_owner:new_group file_or_directory

For example, to change the ownership of the file `example.txt` to the user `new_user` and

Send a message...

the group `new_group`, you can use the following command:

+ New chat

chmod com ↕ ⚡

Linux File Permission

Run C Program in Ba

Linux sed command.

Clear conversations

Upgrade to Plus NEW

Settings

Get help

[→ Log out

bash

Copy code

```
chown new_user:new_group example.txt
```

To change the ownership of a directory and all its contents, you can use the `-R` option, which stands for "recursive". For example, to change the ownership of a directory named `mydir` and all its contents to the user `new_user` and the group `new_group`, you can use the following command:

bash

Copy code

```
chown -R new_user:new_group mydir
```

5. Verify the ownership: After using the `chown` command, you can verify that the ownership has been changed by using the `ls -l` command again. For example:

bash

Regenerate response

Copy code

Send a message...

been changed by using the `ls -l` command again. For example:

+ New chat

chmod com ↻ ⚡

Linux File Permission

Run C Program in Ba

Linux sed command.

Clear conversations

Upgrade to Plus NEW

Settings

Get help

[→ Log out

bash

ls -l example.txt

Copy code

The output should show the new owner and group for the file:

css

-rw-r--r-- 1 new_user new_group 1234 Mar 10 12:34 example.txt

Copy code

That's it! These are the basic steps to use the `chown` command to change the ownership of a file or directory in Linux.

all arguments to passing the chown command

Regenerate response

Send a message...

+ New chat

chmod com ↗ ⚡

Linux File Permission

Run C Program in Ba

Linux sed command.

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

Here are some key differences between hard links and soft links:

1. In terms of size, hard links and original files have the same size, while soft links are typically smaller than the original file, since they only contain the path to the original file.
2. Hard links cannot be created for directories, while soft links can be created for both files and directories.
3. Hard links cannot be created across different filesystems, while soft links can be created across different filesystems.
4. If you delete the original file, the hard link will still point to the file and can be used to access the file. However, if you delete the original file that a soft link points to, the soft link will be broken and will no longer work.

In general, hard links are useful when you want to create multiple names for a single file, while soft links are useful when you want to create a reference to another file or directory.

⟳ Regenerate response

Send a message...



links that can be created to a file.

+ New chat

chmod com  

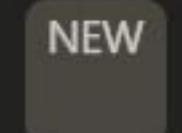
Linux File Permission

Run C Program in Ba

Linux sed command.

-

 Clear conversations

Upgrade to Plus 

 Settings

 Get help

[→ Log out

A hard link is a link that directly points to the inode of a file. In other words, it is an additional name for an existing file. When you create a hard link, both the original file and the hard link have the same inode number and the same contents. If you make changes to the original file or the hard link, both files will be updated, since they are essentially the same file. Hard links can only be created for files and not for directories, and they must be created on the same filesystem as the original file.

A soft link, on the other hand, is a link to the filename itself, rather than to the inode. It is a pointer to another filename or directory. When you create a soft link, a new file is created that contains the path to the original file or directory. If you make changes to the original file or directory, the soft link will still point to the original file, but the contents of the soft link file itself will not change. Soft links can be created for both files and directories, and they can be created across different filesystems.

Here are some key differences between hard links and soft links:

 Regenerate response

Send a message...



+ New chat

chmod com  

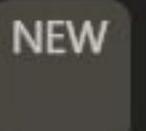
Linux File Permission

Run C Program in Ba

Linux sed command.

-

 Clear conversations

Upgrade to Plus 

 Settings

 Get help

[→] Log out

In a Unix-like operating system, an inode (short for "index node") is a data structure that stores information about a file or directory. Each file or directory on a Unix filesystem is represented by an inode, which contains metadata such as the file type, permissions, owner, size, and timestamps. The inode also contains a pointer to the location of the file's data on the disk.

When a file is created on a Unix filesystem, the operating system creates a new inode to represent the file and assigns it a unique inode number. The inode number is used to identify the file on the filesystem and is used by the operating system to locate the file's data.

The use of inodes allows Unix-like operating systems to efficiently manage file and directory data. Because each file or directory is represented by an inode, the operating system can quickly access the metadata for a file without having to search the entire filesystem. In addition, because the inode contains a pointer to the file's data location, the operating system can read or write the file's data directly without having to search for it.

 Regenerate response

Inode numbers are unique within a filesystem, which means that no two files or directories on

Send a message...

+ New chat

chmod com  

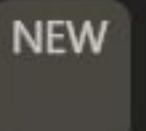
Linux File Permission

Run C Program in Ba

Linux sed command.

-

 Clear conversations

Upgrade to Plus 

 Settings

 Get help

[→ Log out

In a Unix-like operating system, an inode (short for "index node") is a data structure that stores information about a file or directory. Each file or directory on a Unix filesystem is represented by an inode, which contains metadata such as the file type, permissions, owner, size, and timestamps. The inode also contains a pointer to the location of the file's data on the disk.

When a file is created on a Unix filesystem, the operating system creates a new inode to represent the file and assigns it a unique inode number. The inode number is used to identify the file on the filesystem and is used by the operating system to locate the file's data.

The use of inodes allows Unix-like operating systems to efficiently manage file and directory data. Because each file or directory is represented by an inode, the operating system can quickly access the metadata for a file without having to search the entire filesystem. In addition, because the inode contains a pointer to the file's data location, the operating system can read or write the file's data directly without having to search for it.

 Regenerate response

Inode numbers are unique within a filesystem, which means that no two files or directories on

Send a message...

+ New chat

chmod com ↗ ⚡

Linux File Permission

Run C Program in Ba

Linux sed command.

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

Aliases in Linux shell commands are a way to create a shortcut for a command or set of commands. By creating an alias, you can save time by not having to type out a long command or remember a complex series of commands. To create an alias, you can use the `alias` command followed by the name of the alias and the command or set of commands that the alias represents. Here is an example:

bash

```
alias ll='ls -alh'
```

Copy code

This creates an alias called `ll` that will run the `ls -alh` command when typed into the terminal. Now, instead of typing out `ls -alh` every time, you can simply type `ll` and the command will be executed.

You can view a list of all current aliases by typing `alias` into the terminal. If you want to remove an alias, you can use the `unalias` command followed by the name of the alias. Here is

Send a message...

+ New chat

chmod com 📁 🗑

Linux File Permission

Run C Program in Ba

Linux sed command.

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

1. Options: Options are flags or switches that modify the behavior of a command. They usually start with a dash (`-`) or a double dash (`--`) and are followed by a single letter or a word. For example, the `ls` command has several options, such as `-l` to show the long listing format and `-a` to show hidden files.
2. Arguments: Arguments are additional information that a command needs to perform its function. For example, the `cp` command requires the names of the source file(s) and the destination directory or file.
3. Wildcards: Wildcards are special characters that can be used to represent groups of files or directories. The most common wildcards are the asterisk (`*`) and the question mark (`?`). For example, the command `rm *.txt` will remove all files in the current directory that have a `.txt` file extension.
4. Redirection: Redirection is used to direct the input or output of a command to a file or another command. For example, the `>` symbol is used to redirect the output of a command to a file, and the `|` symbol is used to pipe the output of one command to the input of another command.

⟳ Regenerate response

5 Environment Variables: Environment variables are special variables that contain information

Send a message...

+ New chat

chmod com 🚧 🗑

Linux File Permission

Run C Program in Ba

Linux sed command.

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

1. `;`: The semicolon is used to separate multiple commands on the same line. For example, `command1; command2` will execute `command1` and then `command2`.
2. `&&`: The double ampersand is used to execute the second command only if the first command succeeds (returns a zero exit status). For example, `command1 && command2` will execute `command2` only if `command1` succeeds.
3. `||`: The double pipe is used to execute the second command only if the first command fails (returns a non-zero exit status). For example, `command1 || command2` will execute `command2` only if `command1` fails.
4. `>`: The greater-than symbol is used to redirect the output of a command to a file. For example, `command > output.txt` will send the output of `command` to a file called `output.txt`.
5. `>>`: The double greater-than symbol is used to append the output of a command to a file. For example, `command >> output.txt` will append the output of `command` to the end of a file called `output.txt`.
6. `<`: The less-than symbol is used to redirect the input of a command from a file. For example, `command < input.txt` will read the input of `command` from a file called `input.txt`.

Send a message...

Regenerate response

+ New chat

chmod com ✎ ⚡

Linux File Permission

Run C Program in Ba

Linux sed command.

Clear conversations

Upgrade to Plus NEW

Settings

Get help

Log out

example, `command > output.txt` will send the output of `command` to a file called `output.txt`.

5. `>>`: The double greater-than symbol is used to append the output of a command to a file.

For example, `command >> output.txt` will append the output of `command` to the end of a file called `output.txt`.

6. `<`: The less-than symbol is used to redirect the input of a command from a file. For example, `command < input.txt` will read the input of `command` from a file called `input.txt`.

7. `|`: The vertical bar is used to pipe the output of one command as input to another command. For example, `command1 | command2` will send the output of `command1` as input to `command2`.

8. `&`: The ampersand is used to run a command in the background. For example, `command &` will run `command` in the background and return control to the shell.

9. `():` The parentheses are used to group commands together. For example, `(command1; command2) > output.txt` will group `command1` and `command2` together and send their output to a file called `output.txt`.

⟳ Regenerate response

Send a message...

+ New chat

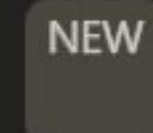
chmod com  

Linux File Permission

Run C Program in Ba

Linux sed command.

 Clear conversations

Upgrade to 
Plus

 Settings

 Get help

[→ Log out

1. `history`: Displays a list of all the commands you have executed in the current shell session. By default, the output shows the command number and the command itself.
2. `!n`: Executes the command with the specified number `n` from your command history. For example, `!10` will execute the 10th command in your history.
3. `!!`: Executes the previous command in your history. This can be useful if you need to run the same command again.
4. `!string`: Executes the most recent command in your history that starts with `string`. For example, `!ls` will execute the most recent `ls` command in your history.
5. `history n`: Displays the last `n` commands you have executed in your shell session. For example, `history 5` will display the last five commands you have executed.
6. `ctrl + r`: Allows you to search your command history for a specific command. Simply type `ctrl + r` and then start typing the command you want to search for. The shell will display the most recent command that matches your search.
7. `history -c`: Clears your command history.

 Regenerate response

Send a message...

+ New chat

ifconfig in Lin  

chmod command ty

Linux File Permissio

Run C Program in Ba

 Clear
conversations

 Upgrade to Plus NEW

 Settings

 Get help

[→ Log out



‘**ifconfig**’ is a command-line tool used in Linux and other Unix-like operating systems to configure and display information about network interfaces. It can be used to view and set network interface configuration parameters such as IP addresses, netmasks, and broadcast addresses.



To use ‘**ifconfig**’ in Linux, follow these steps:

1. Open a terminal window.
2. Type ‘**ifconfig**’ and press Enter. This will display information about all network interfaces currently configured on the system.
3. To view information about a specific interface, type ‘**ifconfig <interface_name>**’ and press Enter. For example, ‘**ifconfig eth0**’ will display information about the eth0 interface.
4. To set configuration parameters for an interface, use the appropriate options with the ‘**ifconfig**’ command. For example, to set the IP address for the eth0 interface to 192.168.1.100, type ‘**ifconfig eth0 192.168.1.100**’ and press Enter.

Send a message...



+ New chat

ifconfig in Lin ✎ ✖

chmod command ty

Linux File Permissio

Run C Program in Ba

Clear
conversations

Upgrade to
Plus NEW

Settings

Get help

Log out

The main differences between `ip` and `ifconfig` are:

1. Syntax: `ip` uses a more consistent and flexible syntax compared to `ifconfig`, which has different options and formats across different operating systems and distributions.
2. Functionality: `ip` provides a wider range of functionalities for configuring and managing network interfaces, such as creating virtual interfaces, changing link-layer addresses, managing routing tables, and more. `ifconfig` is more limited in its capabilities and is mainly used for basic network interface configuration.
3. Display: `ip` provides a more detailed and structured output than `ifconfig`, which can be easier to parse and use for scripting and automation.

Overall, while `ifconfig` is still widely used and supported on many Linux distributions, `ip` is becoming the preferred tool for network interface configuration and management due to its more advanced features and flexibility.

Send a message...



+ New chat

ifconfig in Lin ✎ ✖

chmod command ty

Linux File Permissio

Run C Program in Ba

Clear
conversations

Upgrade to
Plus NEW

Settings

Get help

Log out



'**truncate**' is a command-line utility in Linux and other Unix-like operating systems that is used to shrink or extend the size of a file to a specified size. This can be useful in various situations, such as freeing up disk space, reducing the size of log files, or creating empty files of a specific size.



The basic syntax of the '**truncate**' command is:

bash

Copy code

```
truncate [OPTION]... FILE...
```

Here, '**FILE**' is the name of the file to be truncated, and '**OPTION**' specifies the action to be taken. Some common options are:

- '**-s, --size=<SIZE>**': Set the file size to '**<SIZE>**' bytes. If '**<SIZE>**' is larger than the

Send a message...



bash

 Copy code

+ New chat

`truncate [OPTION]... FILE...`

 ifconfig in Lin  

 chmod command ty

 Linux File Permissio

 Run C Program in Ba

 Clear
conversations

 Upgrade to **Plus** 

 Settings

 Get help

[→ Log out

Here, `FILE` is the name of the file to be truncated, and `OPTION` specifies the action to be taken. Some common options are:

- `--s, --size=<SIZE>`: Set the file size to `<SIZE>` bytes. If `<SIZE>` is larger than the current file size, the file is extended with null bytes.
- `--c, --no-create`: Do not create a new file if the specified file does not exist.
- `--o, --io-blocks`: Set the file size in units of the filesystem's I/O block size.

For example, to truncate a file named `example.txt` to 1000 bytes, you can use the following command:

bash

 Copy code

`truncate -s 1000 example.txt`

Send a message...

- `--c, --no-create`: Do not create a new file if the specified file does not exist.
- `--o, --io-blocks`: Set the file size in units of the filesystem's I/O block size.

+ New chat

ifconfig in Lin ✎ ✖

chmod command ty

Linux File Permissio

Run C Program in Ba

Clear
conversations

Upgrade to
Plus NEW

Settings

Get help

[→ Log out

For example, to truncate a file named `example.txt` to 1000 bytes, you can use the following command:

bash

Copy code

```
truncate -s 1000 example.txt
```

If `example.txt` is currently larger than 1000 bytes, the extra bytes will be removed from the end of the file. If `example.txt` is currently smaller than 1000 bytes, it will be padded with null bytes to reach the specified size.

Note that `truncate` can only be used to shrink or extend regular files, and not directories or special files such as sockets or pipes. Also, be careful when using the `truncate` command, as it can potentially cause data loss if used incorrectly.

Send a message...

+ New chat

ifconfig in Lin ✎ ✖

chmod command ty

Linux File Permissio

Run C Program in Ba

Clear
conversations

Upgrade to
Plus NEW

Settings

Get help

Log out



'traceroute' is a command-line utility in Linux and other Unix-like operating systems that is used to trace the path that an Internet Protocol (IP) packet takes from the local host to a remote destination host. It does this by sending a series of packets with gradually increasing Time-To-Live (TTL) values, which causes the packets to traverse through a series of routers until they reach the destination host.



The basic syntax of the 'traceroute' command is:

css

Copy code

```
traceroute [OPTIONS] <HOSTNAME/IP ADDRESS>
```

Here, '**HOSTNAME/IP ADDRESS**' is the name or IP address of the destination host, and '**OPTIONS**' specifies various parameters to customize the traceroute behavior. Some common options are:

Send a message...



+ New chat

ifconfig in Lin  

chmod command ty

Linux File Permissio

Run C Program in Ba

 Clear conversations

Upgrade to Plus 

Settings

Get help

[→ Log out

The basic syntax of the `traceroute` command is:

css

 Copy code

```
traceroute [OPTIONS] <HOSTNAME/IP ADDRESS>
```

Here, `HOSTNAME/IP ADDRESS` is the name or IP address of the destination host, and `OPTIONS` specifies various parameters to customize the traceroute behavior. Some common options are:

- `'-n'`: Do not resolve hostnames to IP addresses.
- `'-m <MAX TTL>'`: Set the maximum TTL value to `<MAX TTL>` (default is 30).
- `'-p <PORT>'`: Use a specific destination port number for the packets (default is 33434).
- `'-q <NUM PACKETS>'`: Send `<NUM PACKETS>` packets for each TTL value (default is 3).
- `'-w <WAIT TIME>'`: Set the timeout for each packet in milliseconds (default is 5 seconds).

For example, to trace the path to the host `www.google.com`, you can use the following

Send a message...



+ New chat

ifconfig in Lin ✎ ✖

chmod command ty

Linux File Permissio

Run C Program in Ba

Clear
conversations

Upgrade to Plus NEW

Settings

Get help

[→ Log out



'**tracepath**' is a command-line utility in Linux and other Unix-like operating systems that is used to trace the path that an Internet Protocol (IP) packet takes from the local host to a remote destination host, similar to '**traceroute**'. However, '**tracepath**' uses different techniques to achieve its results, and provides a simpler and more concise output format.

The basic syntax of the '**tracepath**' command is:

css

Copy code

```
tracepath [OPTIONS] <HOSTNAME/IP ADDRESS>
```

Here, '**HOSTNAME/IP ADDRESS**' is the name or IP address of the destination host, and '**OPTIONS**' specifies various parameters to customize the tracepath behavior. Some common options are:

- '**-n**': Do not resolve hostnames to IP addresses.

Send a message...



+ New chat

☐ ifconfig in Lin ✎ ✖

☐ chmod command ty

☐ Linux File Permissio

☐ Run C Program in Ba

 Clear
conversations

 Upgrade to
Plus NEW

 Settings

 Get help

[→ Log out

Here, `HOSTNAME/IP ADDRESS` is the name or IP address of the destination host, and `OPTIONS` specifies various parameters to customize the tracepath behavior. Some common options are:

- `'-n'`: Do not resolve hostnames to IP addresses.
- `'-p <PORT>'`: Use a specific destination port number for the packets (default is 80 for TCP, 33434 for UDP).
- `'-m <MAX HOPS>'`: Set the maximum number of hops to `<MAX HOPS>` (default is 30).

For example, to trace the path to the host `www.google.com` using TCP packets, you can use the following command:

css

 Copy code

```
tracepath -n -p 80 www.google.com
```

Stop generating

Send a message...

+ New chat

ifconfig in Lin ✎ ✖

chmod command ty

Linux File Permissio

Run C Program in Ba

Clear
conversations

Upgrade to
Plus NEW

Settings

Get help

[→ Log out



‘ping’ is a command-line utility in Linux and other Unix-like operating systems that is used to test the connectivity and latency of a network host, typically using the Internet Control Message Protocol (ICMP). The ‘ping’ utility works by sending ICMP echo request packets to the destination host, and waiting for ICMP echo reply packets in response.



The basic syntax of the ‘ping’ command is:

css

Copy code

```
ping [OPTIONS] <HOSTNAME/IP ADDRESS>
```

Here, ‘**HOSTNAME/IP ADDRESS**’ is the name or IP address of the destination host, and ‘**OPTIONS**’ specifies various parameters to customize the ping behavior. Some common options are:

- ‘**-c <COUNT>**’: Send ‘**<COUNT>**’ packets and then stop (default is infinite).
- ‘**-i <INTERVAL>**’: Set the time interval between packets in seconds (default is 1 second).

Send a message...



[+ New chat](#)[ifconfig in Lin](#)[chmod command ty](#)[Linux File Permissio](#)[Run C Program in Ba](#)[Clear conversations](#)[Upgrade to Plus](#)[Settings](#)[Get help](#)

ping www.google.com

This will send a series of ICMP echo request packets to the destination host, and display the round-trip time (RTT) and other statistics for each packet. The output of `ping` shows the IP address of the destination host, the size of the ICMP packet payload, the number of packets sent and received, the percentage of packet loss, and the minimum, average, and maximum RTT.

Note that `ping` requires superuser privileges to run, since it sends and receives raw ICMP packets. Some firewalls or routers may also be configured to block or limit ICMP traffic, which can affect the accuracy and completeness of the `ping` results.

[\[→ Log out](#)

[+ New chat](#)[ifconfig in Lin](#)

I believe you may be referring to the `netstat` command, which is a command-line utility in Linux and other Unix-like operating systems that displays various information about network connections and network statistics.

[chmod command ty](#)[Linux File Permissio](#)[Run C Program in Ba](#)[Clear conversations](#)[Upgrade to Plus](#)[Settings](#)[Get help](#)[Log out](#)

The basic syntax of the `netstat` command is:

css

[Copy code](#)

```
netstat [OPTIONS]
```

Here, `OPTIONS` specifies various parameters to customize the `netstat` behavior. Some common options are:

- `-a`: Show all connections and listening ports.



The basic syntax of the `netstat` command is:

+ New chat

ifconfig in Lin ⌂ ⚡

chmod command ty

Linux File Permissio

Run C Program in Ba

Clear
conversations

Upgrade to
Plus NEW

Settings

Get help

[→ Log out

css

Copy code

`netstat [OPTIONS]`

Here, `OPTIONS` specifies various parameters to customize the `netstat` behavior. Some common options are:

- `'-a'`: Show all connections and listening ports.
- `'-n'`: Do not resolve hostnames to IP addresses.
- `'-t'`: Show only TCP connections and listening ports.
- `'-u'`: Show only UDP connections and listening ports.
- `'-l'`: Show only listening ports.
- `'-p'`: Show the process ID and name for each connection.

For example, to display all TCP connections and listening ports on the system, you can use the

Send a message...

+ New chat

AB

ss in linux

ifconfig in Lin ✎ ✖

chmod command ty

Linux File Permissio

Run C Program in Ba

Clear
conversations

Upgrade to
Plus NEW

Settings

Get help

[→ Log out

 `ss` is a command-line utility in Linux and other Unix-like operating systems that displays various information about network sockets, including active connections, listening sockets, and network statistics. `ss` is similar to the `netstat` command, but provides more detailed and flexible output.

The basic syntax of the `ss` command is:

css

 Copy code

ss [OPTIONS]

Here, `OPTIONS` specifies various parameters to customize the `ss` behavior. Some common options are:

Send a message...

+ New chat

ifconfig in Lin ✎ ✖

chmod command ty

Linux File Permissio

Run C Program in Ba

Clear
conversations

Upgrade to
Plus NEW

Settings

Get help

[→ Log out



'ss' is a command-line utility in Linux and other Unix-like operating systems that displays various information about network sockets, including active connections, listening sockets, and network statistics. 'ss' is similar to the 'netstat' command, but provides more detailed and flexible output.



The basic syntax of the 'ss' command is:

css

Copy code

ss [OPTIONS]

Here, 'OPTIONS' specifies various parameters to customize the 'ss' behavior. Some common options are:

- '-a': Show all connections and listening sockets.

Send a message...



[+ New chat](#)[ifconfig in Lin](#) [chmod command ty](#)[Linux File Permissio](#)[Run C Program in Ba](#)[Clear conversations](#)[Upgrade to Plus](#) NEW[Settings](#)[Get help](#)

ss [OPTIONS]

Here, `OPTIONS` specifies various parameters to customize the `ss` behavior. Some common options are:

- `'-a'`: Show all connections and listening sockets.
- `'-n'`: Do not resolve hostnames to IP addresses.
- `'-t'`: Show only TCP connections and listening sockets.
- `'-u'`: Show only UDP connections and listening sockets.
- `'-l'`: Show only listening sockets.
- `'-p'`: Show the process ID and name for each connection or socket.

For example, to display all TCP connections and listening sockets on the system, you can use the following command:

[\[→ Log out](#)

- + Deadlock Prevention
- + Deadlock Avoidance
- + Resource Allocation Graph
- + Detection using RAG
- + Detection and Recovery

✓ Memory Management

- + Introduction
- + Fixed Partitioning
- + Dynamic Partitioning
- + Compaction
- + Bit Map for Dynamic Partitioning
- + Linked List for Dynamic Partition
- + Partitioning Algorithms
- + GATE on Best Fit & First Fit

- + Need for Paging
- + Paging with Example
- + Binary Addresses
- + Physical & Logical Address
- + Page Table
- + Mapping from page table
- + Page Table Entry
- + Page Table Size
- + Finding Optimal Page Size
- + Virtual Memory
- + Look aside Buffer
- + GATE question on TLB

+ Demand Paging

+ Inverted Page Table

↑ SCROLL TO TOP ↓

2. User-level thread.

User-level thread

The operating system does not recognize the user-level thread. User threads can be easily implemented and it is implemented by the user. If a user performs a user-level thread blocking operation, the whole process is blocked. The kernel level thread does not know nothing about the user level thread. The kernel-level thread manages user-level threads as if they are single-threaded processes?examples: Java thread, POSIX threads, etc.

Advantages of User-level threads

1. The user threads can be easily implemented than the kernel thread.
2. User-level threads can be applied to such types of operating systems that do not support threads at the kernel-level.
3. It is faster and efficient.
4. Context switch time is shorter than the kernel-level threads.
5. It does not require modifications of the operating system.
6. User-level threads representation is very simple. The register, PC, stack, and mini thread control blocks are stored in the address space of the user-level process.
7. It is simple to create, switch, and synchronize threads without the intervention of the process.

Disadvantages of User-level threads

1. User-level threads lack coordination between the thread and the kernel.
2. If a thread causes a page fault, the entire process is blocked.

- + Finding Optimal Page Size
- + Virtual Memory
- + Look aside Buffer
- + GATE question on TLB
- + Demand Paging
- + Inverted Page Table
- + Page Replacement
- + Gate on LRU and FIFO
- + Numerical on LRU, FIFO
- + Beladys Anomaly
- + Segmentation
- + Paging VS Segmentation
- + Segmented Paging

- ✓ File Management
- + Attributes of the File
 - + Operations on the File
 - + File Access Methods
 - + Directory Structure
 - + Single level Directory
 - + Two level Directory
 - + Tree structured Directory
 - + Acyclic Graph Directories
 - + File System
 - + File System Structure
 - + Master Boot Record

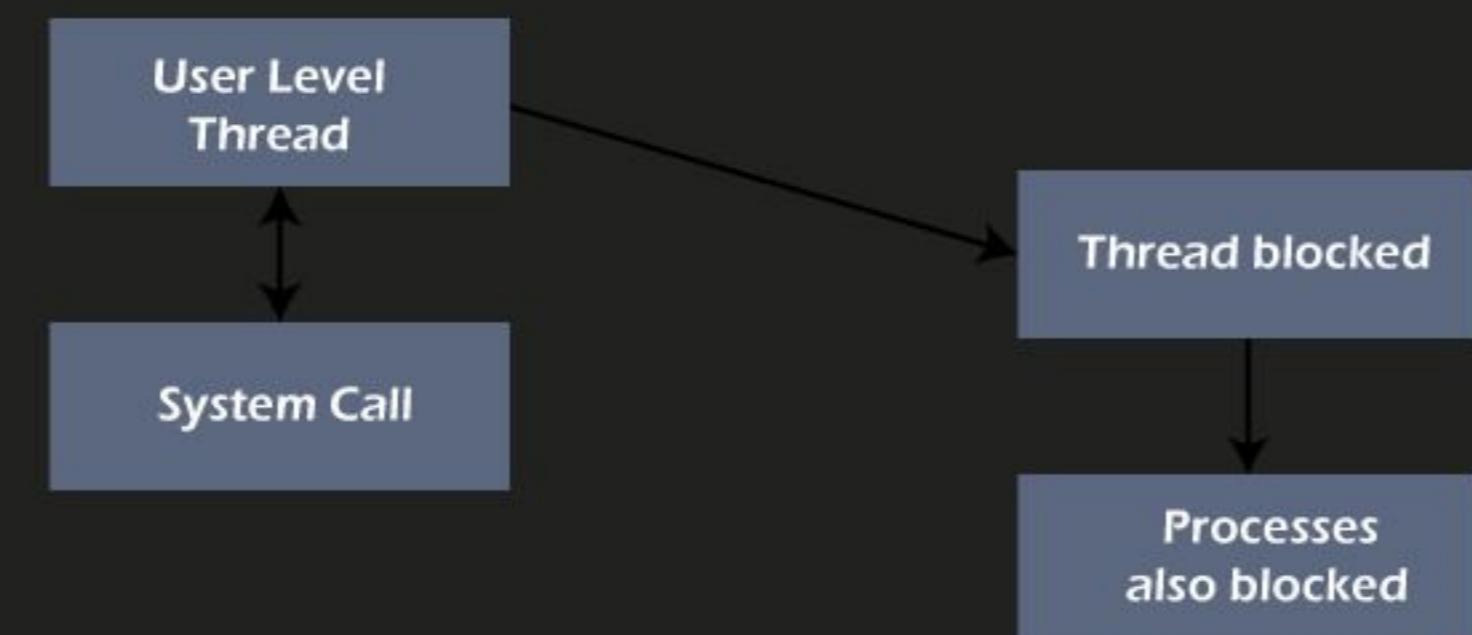
- + On Disk Data Structures
- + In memory Data structures
- + Directory Implementation

blocks are stored in the address space of the user-level process.

7. It is simple to create, switch, and synchronize threads without the intervention of the process.

Disadvantages of User-level threads

1. User-level threads lack coordination between the thread and the kernel.
2. If a thread causes a page fault, the entire process is blocked.



Kernel level thread

The kernel thread recognizes the operating system. There is a thread control block and process control block in the system for each thread and process in the kernel-level thread. The kernel-level thread is implemented by the operating system. The kernel knows about all the threads and manages them. The kernel-level thread offers a system call to create and manage the threads from user-space. The implementation of kernel threads is more difficult than the user thread. Context switch time is longer in

[+ Banker's Algorithm in OS](#)[+ What is the context switching in the operating system](#)[+ Internal vs. External Fragmentation](#)[+ Multiprocessing Operating system](#)[+ Multiprogramming vs Multitasking](#)[+ Network Operating System](#)[+ Latest Operating System](#)[+ System Calls in Operating System](#)[+ Multitasking Operating System](#)[+ Distributed Operating System](#)[+ Fragmentation in Operating System](#)[+ Multithreading Models in Operating system](#)[+ Operating system services](#)[+ Batch Operating System](#)[+ Embedded Operating System](#)[+ Time-Sharing vs Real-Time Operating System](#)[+ Multi-User Operating System](#)[+ Memory Management](#)[+ Monolithic Structure of Operating System](#)[+ MS-DOS Operating System](#)[+ Spooling in Operating System](#)[+ Network Operating System vs Distributed Operating System](#)[+ Operating system vs Application](#)[↑ SCROLL TO TOP](#)[+ Operating Systems](#)

Advantages of Kernel-level threads

1. The kernel-level thread is fully aware of all threads.
2. The scheduler may decide to spend more CPU time in the process of threads being large numerical.
3. The kernel-level thread is good for those applications that block the frequency.

Disadvantages of Kernel-level threads

1. The kernel thread manages and schedules all threads.
2. The implementation of kernel threads is difficult than the user thread.
3. The kernel-level thread is slower than user-level threads.

Components of Threads

Any thread has the following components.

1. Program counter
2. Register set
3. Stack space

Benefits of Threads

- **Enhanced throughput of the system:** When the process is split into many threads, and each thread is treated as a job, the number of jobs done in the unit time increases. That is why the throughput of the system also increases.

[software](#)[+ System Software vs Operating](#)[System](#)[+ Real-Time operating system](#)[+ Booting in Operating System](#)[+ Layered Structure of Operating System](#)[+ Multiprogramming vs. Time Shared Operating System](#)[+ Distributed File System](#)[+ Multiprogramming Operating System](#)[+ Server Operating System](#)[+ 32 bit vs 64 bit Operating System](#)[+ Single User Operating System](#)[+ Process vs Program](#)[+ Operating System Security](#)[+ Components of Operating System](#)[+ GUI Operating System](#)[+ Android Operating System](#)[+ Buffering in Operating System](#)[+ Device Driver in Operating System](#)[+ Best Operating System for Hack](#)[+ Clustered Operating System](#)[+ Android vs Windows OS](#)[+ Scheduler vs Dispatcher](#)[+ Deadlock vs Starvation](#)[+ Semaphore vs Monitor](#)[+ Boot Block and Bad Block in](#)[+ Time Operati](#)

- **Alternatives**

Also, the Fedora Project distributes many other versions with fewer use cases than discussed above, like minimal installation images and network installers. They are meant for expert users or special cases that wish to include custom installations or set up Fedora using scratch.

Fedora Server

Fedora Server is a very flexible and powerful OS. It keeps all your infrastructure and services under your control. Fedora operating system offers the latest data center technologies.

Advantages of Fedora Operating System

1. Fedora OS is a very reliable and stable operating system.
2. It enhances the security in this operating system.
3. It offers many graphical tools.
4. This operating system updates automatically.
5. This OS supports many file formats.
6. It also offers many education software.
7. It supports a large community.
8. It provides unique security features.

Disadvantages of Fedora Operating System

1. It requires a long time to set up.

- [Components of Operating System](#)
- [GUI Operating System](#)
- [Android Operating System](#)
- [Buffering in Operating System](#)
- [Device Driver in Operating System](#)
- [Best Operating System for Hack](#)
- [Clustered Operating System](#)
- [Android vs Windows OS](#)
- [Scheduler vs Dispatcher](#)
- [Deadlock vs Starvation](#)
- [Semaphore vs Monitor](#)
- [Boot Block and Bad Block in Operating System](#)
- [Hard and Soft Real-Time Operating System](#)
- [Concurrency in Operating System](#)
- [Microkernel in Operating System](#)
- [Protection in Operating System](#)
- [Benefits of Multithreading](#)
- [Host vs Guest Operating System](#)
- [Livelock in Operating System](#)
- [Dual Mode Operations in Operating System](#)
- [What is Access Token in Operating System](#)
- [Deadlock Detection in Distributed Systems](#)
- [CUI vs GUI](#)
- [Monolithic vs layered Operating](#)

2. It enhances the security in this operating system.
3. It offers many graphical tools.
4. This operating system updates automatically.
5. This OS supports many file formats.
6. It also offers many education software.
7. It supports a large community.
8. It provides unique security features.

Disadvantages of Fedora Operating System

1. It requires a long time to set up.
2. It requires additional software tools for the server.
3. It does not provide any standard model for multi-file objects.
4. Fedora has its own server, so we can't work on another server in real-time.

[Next Topic](#)[Uses of Operating System](#)[← Prev](#)[Next →](#)[↑ SCROLL TO TOP](#)

For Videos Join Our Youtube Channel: [Join Now](#)

[⊕ FCFS with overhead](#)[⊕ SJF Scheduling](#)[⊕ Burst Time Prediction](#)[⊕ SRTF scheduling](#)[⊕ SRTF GATE 2011 Example](#)[⊕ Round Robin Scheduling](#)[⊕ RR scheduling Example](#)[⊕ HRRN Scheduling](#)[⊕ HRNN Example](#)[⊕ Priority Scheduling](#)[⊕ Non Preemptive Priority](#)[⊕ Preemptive Priority](#)[⊕ SRTF:IO bound processes](#)

✓ Synchronization

[⊕ Introduction](#)[⊕ Critical Section Problem](#)[⊕ Lock Variable Mechanism](#)[⊕ TSL Mechanism](#)[⊕ Priority Inversion in TSL](#)[⊕ Turn Variable](#)[⊕ Interested Variable](#)[⊕ Paterson Solution](#)[⊕ Without Busy Waiting](#)[⊕ Sleep and Wake](#)[⊕ Semaphore Introduction](#)[⊕ Counting Semaphore](#)[⊕ Problem on counting semaphore](#)[⊕ Binary Semaphore](#)

operating system is commendable as it was developed in 1950 to handle storage tape. It acts as an interface. The features of operating system are given below.

- Error detection and handling
- Handling I/O operations
- Virtual Memory Multitasking
- Program Execution
- Allows disk access and file systems
- Memory management
- Protected and supervisor mode
- Security
- Resource allocation
- Easy to run
- Information and Resource Protection
- Manipulation of the file system

Characteristics of Operating System

Memory Management: The operating system manages memory. It has complete knowledge of primary memory; which part of the memory is used by which program. Whenever a program requests, it allocates memory.

Processor Management: It allocates the program to the processor (CPU) and also deallocates it when a program runs out of the CPU needs.

[+ without busy waiting](#)[+ Sleep and Wake](#)[+ Semaphore Introduction](#)[+ Counting Semaphore](#)[+ Problem on counting semaphore](#)[+ Binary Semaphore](#)[✓ Deadlocks](#)[+ Introduction](#)[+ strategies Handling](#)[+ Deadlock Prevention](#)[+ Deadlock Avoidance](#)[+ Resource Allocation Graph](#)[+ Detection using RAG](#)[+ Detection and Recovery](#)[✓ Memory Management](#)[+ Introduction](#)[+ Fixed Partitioning](#)[+ Dynamic Partitioning](#)[+ Compaction](#)[+ Bit Map for Dynamic Partitioning](#)[+ Linked List for Dynamic Partition](#)[+ Partitioning Algorithms](#)[+ GATE on Best Fit & First Fit](#)[+ Need for Paging](#)[+ Paging with Example](#)[+ Binary Addresses](#)[+ Physical & Logical Address](#)[+ Page Table](#)[↑ SCROLL TO TOP ↑](#)

Characteristics of Operating System

Memory Management: The operating system manages memory. It has complete knowledge of primary memory; which part of the memory is used by which program. Whenever a program requests, it allocates memory.

Processor Management: It allocates the program to the processor (CPU) and also deallocates it when a program runs out of the CPU needs.

Device Management: The operating system keeps the information about all devices. It is also called the I/O controller, and the operating system also decides which devices are used to which program, when, and for how long.

Security: It prevents unauthorized access to any program. It uses passwords and other technologies.

Reliability: It is very reliable because no any virus and harmful code can be detected in it.

File Management: It allocates and deallocates resources and decides which program to allocate resources.

Easy to use: It can be easily used as it also has a GUI interface.

Needs of Operating System

The following points indicate the need for the operating system:

- More than one program runs at a time in a computer, and all of them require your computer's CPU and memory. The operating system manages resources for all those programs. That is why the operating system is required.
- Multitasking is a very critical feature of the OS. With its help, we can run many programs simultaneously.