

## 8 Mysterious Uses of (!) Operator in Linux Commands

Avishek | Last Updated: July 14, 2023 | Read Time: 6 mins | [Linux Commands](#) | [14 Comments](#)

The `!` symbol or operator in Linux can be used as a Logical Negation operator as well as to [fetch commands from history](#) with tweaks or to [run previously executed commands](#) with modification.

Most of the following [Linux commands](#) usage with `!` symbol can vary between different shells. While the examples I provided are commonly used in the bash shell, [some other Linux shells](#) may have different implementations or may not support certain uses of the `!` symbol at all.

Let's dive into the amazing and mysterious uses of `!` symbol or operator in Linux commands.

### Table of Contents

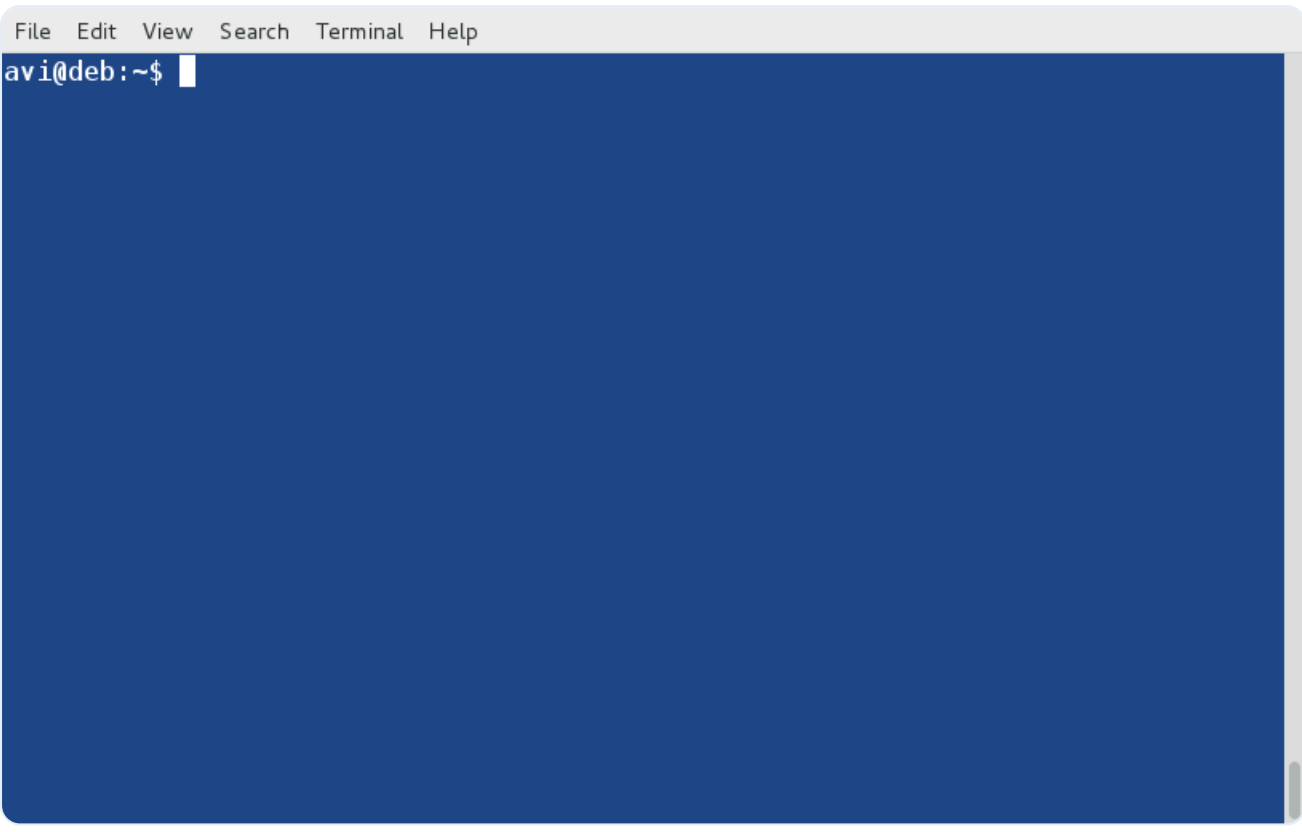


1. Run a Command from History Using Command Numbers
2. Run Previously Executed Commands in Linux
3. Pass Previous Command's Arguments to New Command
4. How to Handle Two or More Arguments in Commands
5. Run Last Commands Based on Specific Keywords
6. Repeat Last Executed Command in Linux
7. Remove Files Except One File Using `!` Operator
8. Check If a Directory Exists in Linux

### 1. Run a Command from History Using Command Numbers

You might not be aware of the fact that you can run a command from your [history command](#) (already/earlier executed commands). To get started first, find the command number by running the 'history' command.

```
$ history
```

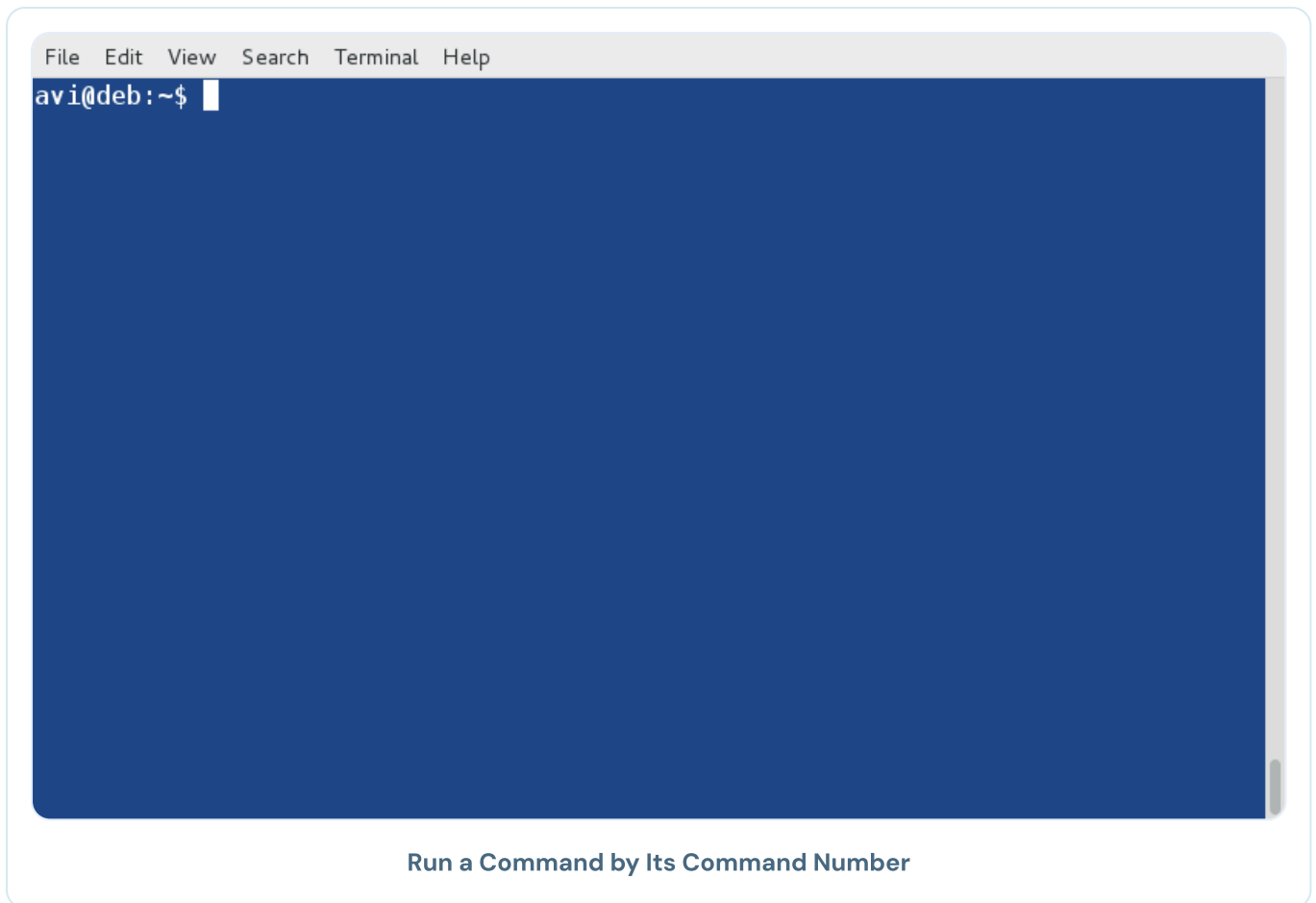


```
File Edit View Search Terminal Help
avi@deb:~$
```

Find the Last Executed Commands in Linux

To run a command from history by its command number, you can use the `!` symbol followed by the command number as shown.

```
$ !1551
```



When you execute the above command, it will run the [top command](#) at line number 1551 from your history.

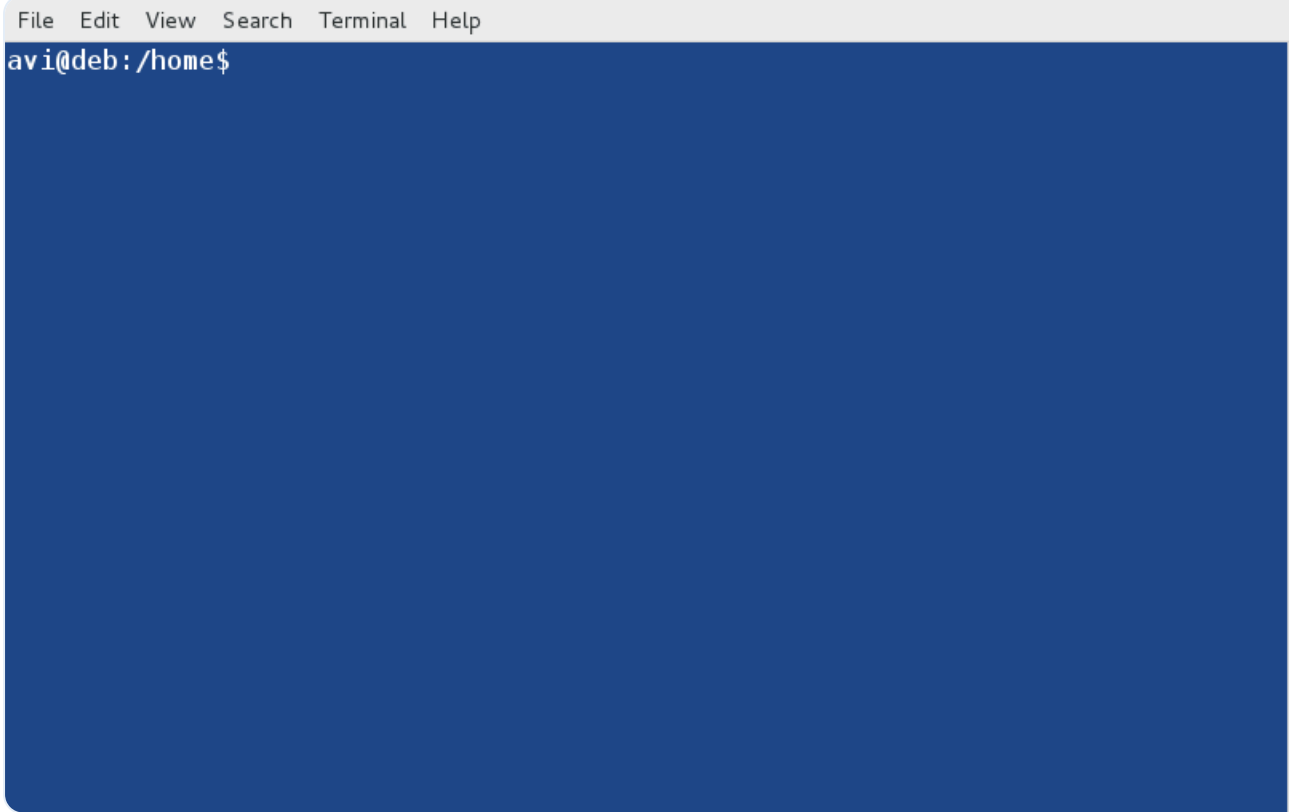
Please note that the actual command number may vary depending on your command history. You can use the history command to view the list of commands along with their line numbers.

## 2. Run Previously Executed Commands in Linux

You may run those commands which you have run previously by their running sequence the last run command will be represented as -1, the second last as -2, the seventh last as -7, and so on.

For example, if you want to rerun the command that was executed six, eight or tenth commands ago, you can use the `!-n`, where `n` is the number of commands back you want to reference.

```
$ history  
$ !-6  
$ !-8  
$ !-10
```



File Edit View Search Terminal Help

avi@deb:/home\$

Re-run a Command in Linux

### 3. Pass Previous Command's Arguments to New Command

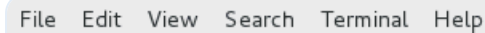
I need to list the content of the directory '/home/\$USER/Binary/firefox' so I fired.

```
$ ls /home/$USER/Binary/firefox
```

Then I realized that I should have fired 'ls -l' to see which file is executable there. So should I type the whole command again? No, I don't need it. I just need to carry the last argument to this new command as:

```
$ ls -l !$
```

Here `!$` will carry arguments passed in the last command to this new command.

A terminal window with a menu bar (File, Edit, View, Search, Terminal, Help) and a dark blue background. The prompt is 'avi@deb:/home\$' followed by a cursor.

Pass Arguments of the Last Executed Command to the New Command

## 4. How to Handle Two or More Arguments in Commands

Let's say I created a text file 1.txt on the Desktop.

```
$ touch /home/avi/Desktop/1.txt
```

and then copy it to '/home/avi/Downloads' using the complete path on either side with the [cp command](#).

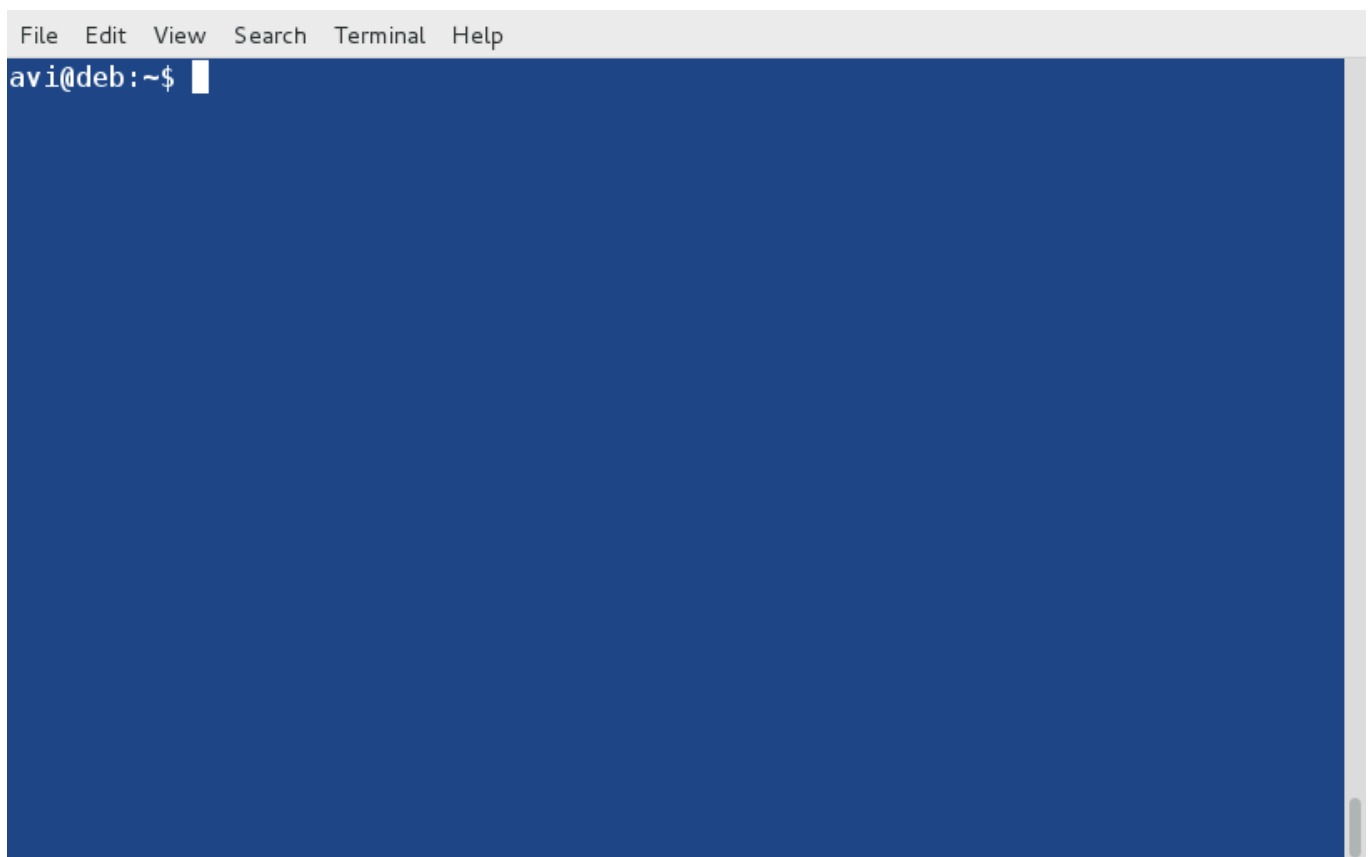
```
$ cp /home/avi/Desktop/1.txt /home/avi/downloads
```

Now we have passed two arguments with the cp command. First is '/home/avi/Desktop/1.txt' and the second is '/home/avi/Downloads', let's handle them differently, just execute `echo [arguments]` to print both arguments differently.

```
$ echo "1st Argument is : !^"  
$ echo "2nd Argument is : !cp:2"
```

Note 1st argument can be printed as `"!^"` and the rest of the arguments can be printed by executing `"![Name_of_Command]:[Number_of_argument]"`.

In the above example, the first command was 'cp' and 2nd argument was needed to print. Hence `"!cp:2"`, if any command says xyz is run with 5 arguments and you need to get the 4th argument, you may use `"!xyz:4"`, and use it as you like. All the arguments can be accessed by `"!*"`.



## 5. Run Last Commands Based on Specific Keywords

We can execute the last executed commands on the basis of keywords. We can understand it as follows:

```
$ ls /home > /dev/null [Command 1]  
$ ls -l /home/avi/Desktop > /dev/null [Command 2]
```

```
$ ls -la /home/avi/Downloads > /dev/null [Command 3]
```

```
$ ls -lA /usr/bin > /dev/null [Command 4]
```

Here we have used the [ls command](#) but with different switches and for different folders. Moreover, we have sent to the [output of each command](#) to '/dev/null' to keep the console clean.

Now execute the last run commands on the basis of keywords.

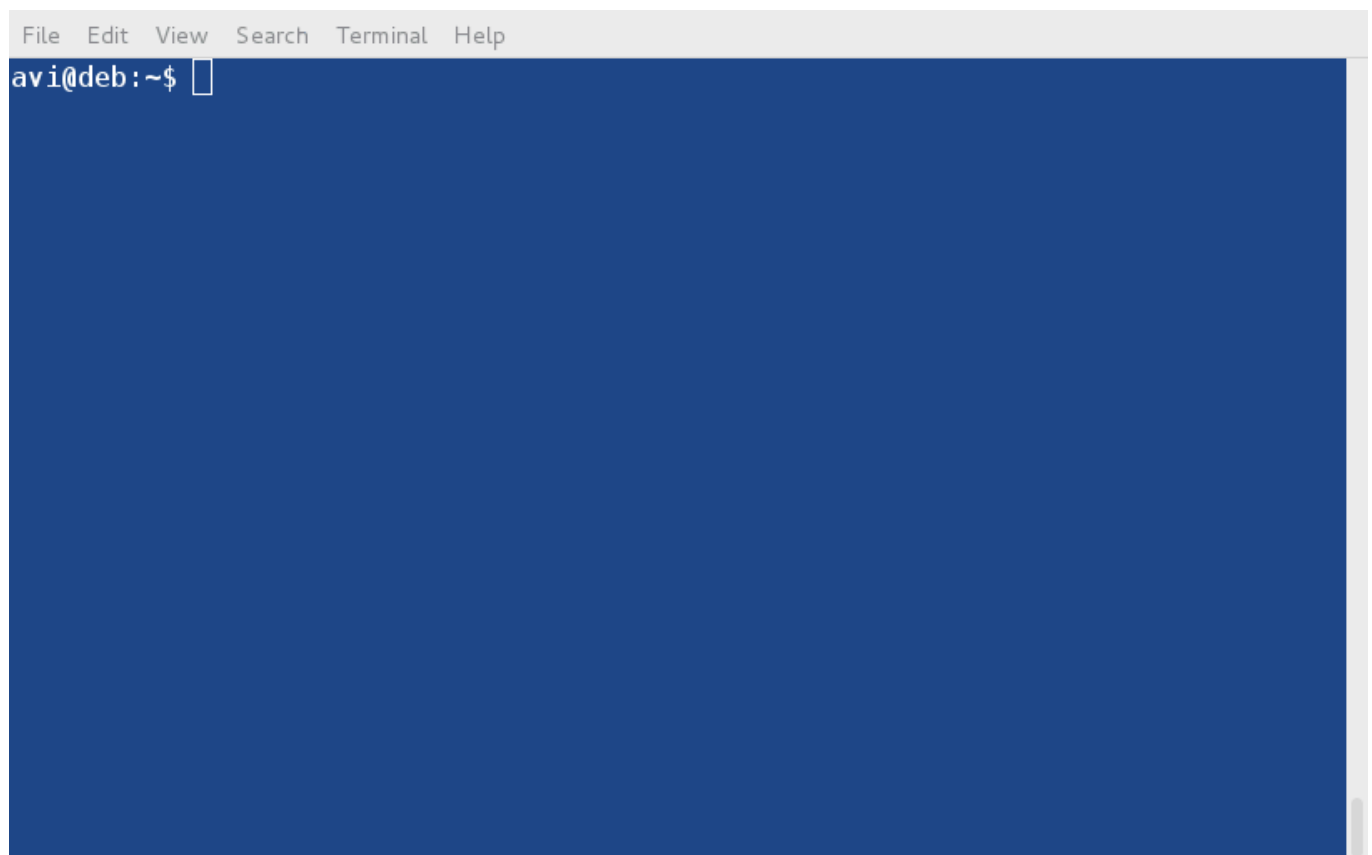
```
$ ! ls [Command 1]
```

```
$ ! ls -l [Command 2]
```

```
$ ! ls -la [Command 3]
```

```
$ ! ls -lA [Command 4]
```

Check the output and you will be astonished that you are running already executed commands just by [ls](#) keywords.



## 6. Repeat Last Executed Command in Linux

You can run/alter your last executed command using `(!!)` operator, which is a shorthand notation that allows you to refer to the previous command executed in the command line.

For example, last day I run a one-liner script to [find out the IP address of the Linux machine](#).

```
$ ip addr show | grep inet | grep -v 'inet6' | grep -v '127.0.0.1' | awk '{p
```

Then suddenly I figured out that I need to redirect the output of the above script to a file `ip.txt`, so what should I do? Should I retype the whole command again and [redirect the output to a file](#)? Well, an easy solution is to use `UP` navigation key and add `'> ip.txt'` to redirect the output to a file.

```
$ ip addr show | grep inet | grep -v 'inet6' | grep -v '127.0.0.1' | awk '{p
```

Thanks to the Life Savior `UP` navigation key here. Now consider the below condition, the next time I run the below one-liner script.

```
$ ifconfig | grep "inet addr:" | awk '{print $2}' | grep -v '127.0.0.1' | c
```

As soon as I run the script, the bash prompt returned an error with the message “bash: [ifconfig](#): command not found”, it was not difficult for me to guess I run this command as a user where it should be run as root.

So what's the solution? it is difficult to log in to root and then type the whole command again! Also (`UP` Navigation Key) in the last example didn't come to the rescue here. So? we need to call `“!!”` without quotes, which will call the last command for that user.

```
$ su -c “!!” root
```



Here `su` is the switch user which is the root, `-c` is to run the specified command as the user and the most important part `!!` will be replaced by command and the last run command will be substituted here. Yeah! You need to provide a root password.



```
File Edit View Search Terminal Help
avi@deb:~$ ifconfig | grep "inet addr:" | awk '{print $2}' | grep -v '127.0.0.1'
| cut -f2 -d:

```

Repeat Last Executed Command in Linux

## 7. Remove Files Except One File Using '!' Operator

In Linux, the `!` operator (known as the “bang” operator) is used for history expansion that allows you to refer to previous commands in your command history and perform various operations with them.

To remove all files from a directory except for a specific file (`important_file.txt`), you can use the [rm command](#) with `!` operator as shown.

```
$ rm !(important_file.txt)
```

To remove all the file types from the folder except the one the extension of which is `.pdf`.

```
$ $ rm !(*.pdf)
```

## 8. Check If a Directory Exists in Linux

Here we will use `['! -d']` to validate if the directory exists or is not followed by Logical AND Operator `(&&)` to print that directory does not exist and Logical OR Operator `(||)` to print the directory is present.

Logic is, when the output of `[ ! -d /home/avi/Tecmint ]` is 0, it will execute what lies beyond Logical AND else it will go to Logical OR `(||)` and execute what lies beyond Logical OR.

```
$ [ ! -d /home/avi/Tecmint ] && printf '\nno such /home/avi/Tecmint directory'
```

Similar to the above condition, but here if the desired directory doesn't exist it will exit the command.

```
$ [ ! -d /home/avi/Tecmint ] && exit
```

A general implementation in scripting language where if the desired directory does not exist, it will create one.

```
[ ! -d /home/avi/Tecmint ] && mkdir /home/avi/Tecmint
```

That's all for now. If you know or come across any other use of `['!']` which is worth knowing, you may like to provide us with your suggestion in the feedback. Keep connected!

Hey TecMint readers,

Exciting news! Every month, our top blog commenters will have the chance to win fantastic rewards, like free Linux eBooks such as RHCE, RHCSA, LFCS, Learn Linux, and Awk, each worth \$20!

Learn [more about the contest](#) and stand a chance to win by [sharing your thoughts below!](#)



PREVIOUS ARTICLE:

**[IPTraf-ng – A Console-Based Network Monitoring Tool](#)**

NEXT ARTICLE:

## How to Upgrade Debian 11 to Debian 12 (Bookworm) via CLI

### Avishek

A Passionate GNU/Linux Enthusiast and Software Developer with over a decade in the field of Linux and Open Source technologies.

*Each tutorial at TecMint is created by a team of experienced Linux system administrators so that it meets our high-quality standards.*

Join the [TecMint Weekly Newsletter](#) (More Than 156,129 Linux Enthusiasts Have Subscribed)

Was this article helpful? Please [add a comment](#) or [buy me a coffee](#) to show your appreciation.

### Related Posts

```
tecmin@tecmin ~/testing $ find . -type f \( -name "*.txt" -o -  
name "*.sh" -o -name "*.c" \)  
./emails.txt  
./script-1.sh  
./header.c  
./examples.txt  
./script.sh  
./expenses.txt
```

## Find Multiple Filenames (File Extensions) Using 'find' Command in Linux

How to Search Files by Name or Extension Using find Command



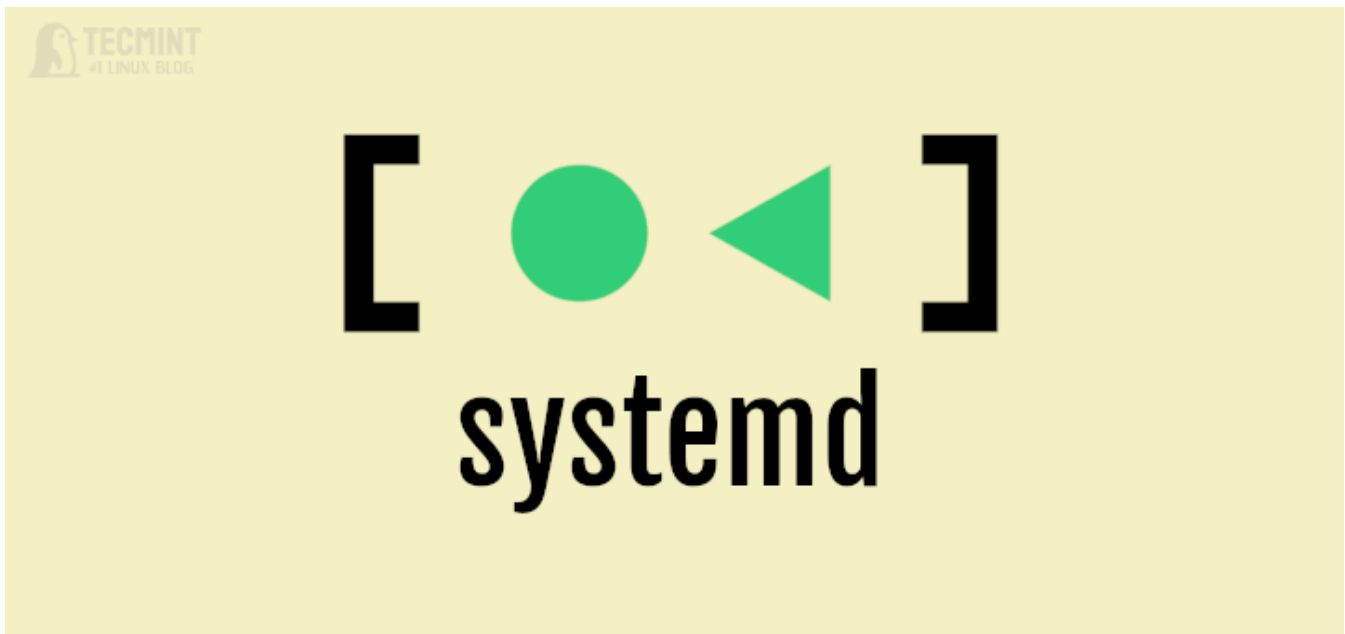
10 Lesser Known Linux Commands – Part 2



## 11 Lesser Known Useful Linux Commands



## 26 Security Hardening Tips for Modern Linux Servers



## How to Remove Systemd Services on Linux

A terminal window with a dark background and light green text. The window title is 'TecMint.com'. The prompt is 'ravi@TecMint:~/glibc-2.39/build\$'. The user has entered the command './configure --prefix=/usr/local/glibc-2.39'. The output shows various checks for build system type, compiler (gcc), and linker (g++), all passing. It also shows checks for various architectures and the presence of grep. The terminal output is as follows:

```
ravi@TecMint:~/glibc-2.39/build$
ravi@TecMint:~/glibc-2.39/build$ ./configure --prefix=/usr/local/glibc-2.39
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking for g++... g++
checking whether the compiler supports GNU C++... yes
checking whether g++ accepts -g... yes
checking for g++ option to enable C++11 features... none needed
checking whether g++ can link programs... yes
checking for sysdeps preconfigure fragments... aarch64 alpha arc arm csky hppa i386 loong
arch m68k microblaze checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
mips nios2 orlk powerpc riscv s390 sh checking for grep that handles long lines and -e...
(cached) /usr/bin/grep
```

## How to Install and Run Multiple glibc Libraries in Linux

 **14 Comments**

[Leave a Reply](#)

sedlav

May 14, 2021 at 7:35 pm

If you are using BASH in order to expose the use of `!`, then you must, at least, clarify that BASH is part of the GNU project not of Linux Kernel, please don't spread misconception.

[Reply](#)

**Amit Kumar Tripathi**

June 12, 2018 at 4:07 pm

Simply Awesome

[Reply](#)

**Edgar Allen**

May 19, 2015 at 10:30 pm

You might also mention `!?`

It finds the last command with its' string argument. For example, if...

```
1013 grep tornado /usr/share/dict/words
```

```
1014 grep hurricane /usr/share/dict/words
```

```
1015 wc -l /usr/share/dict/words
```

are all in the history then `!?!torn` will grep for tornado again where `!torn` would search in vain for a command starting with torn.

And ``wc !?!torn?:2`` works to select argument two from the command containing tornado and run ``wc`` on it.



[Reply](#)**Avishek Kumar**

May 20, 2015 at 11:25 am

Dear Edgar,

Your Suggestion is taken into account.

Thanks for your feedback.

[Reply](#)**Stephen**

May 19, 2015 at 6:07 pm

I didn't see a mention of historical context in the article, so I'll give some here in the comments. This form of history command substitution originated with the C Shell (csh), created by Bill Joy for the BSD flavor of UNIX back in the late 70's. It was later carried into tcsh, and bash (Bourne-Again SHell).

Personally, I've always preferred the C-shell history substitution mechanism, and never really took to the fc command (that I first encountered in the Korn shell).

[Reply](#)**Avishek Kumar**

May 20, 2015 at 11:27 am

Dear Stephen,

Thanks for the wonderful piece of Information. Keep connected and keep us

aware of such context.

[Reply](#)

**suzy**

May 16, 2015 at 11:45 am

4th command. You can access it much simpler. There are actually regular expressions:

^ is at the begging expression

\$ is at the end expression

:number any number parameter

Example:

touch a.txt b.txt c.txt

echo !^ -> display first parameter

echo !:1 -> also display first parameter

echo !:2 -> display second parameter

echo !:3 -> display third parameter

echo !\$ -> display last (in our case 3th) parameter

echo !\* -> display all parameters

[Reply](#)



**Avishek Kumar**

May 18, 2015 at 11:30 pm

And we did the same

echo "1st Argument is : !^"

\$ echo "2nd Argument is : !cp:2"

echo followed by 1st argument and 2nd argument is just to make the tutorial and command understandable. I have used '!^'. '!command:number' as you are suggesting.

I would like to know if you mean something different?

[Reply](#)

**Tomasz Wiszkowski**

May 16, 2015 at 10:50 am

I think (5) works differently than you pointed out, and redirection to devnull hides it, but ZSh still prints the command.

When you invoke "! ls...", it always picks the last ls command you executed, just appends your switches at the end (after /dev/null).

One extra cool thing is the !# operator, which picks arguments from current line. Particularly good if you need to retype long path names you already typed in current line. Just say, for example

```
cp /some/long/path/to/file.abc !#:1
```

And press tab. It's going to replace last argument with entire path and file name.

[Reply](#)



**Avishek Kumar**

May 18, 2015 at 11:37 pm

Tomasz,

For your first part of feedback: It doesn't pick the last command executed and just to prove this we have used 4 different switches for same command. (\$ ! ls \$ ! ls -l \$ ! ls -la \$ ! ls -IA ). Now you may check it by entering the keywords in any order and in each case it will output the same result.

As far as it is not working in ZSH as expected, i have already mentioned that it i have tested it on BASH and most of these won't work in other shell.

For the second part, what you mentioned is a HASH TAG in Linux Command Line and we have included it in one of our article. You may like to read it here:

<https://www.tecmint.com/linux-commandline-chat-server-and-remove-unwanted-packages/>

[Reply](#)

**Manu**

May 16, 2015 at 8:55 am

Nice Article

[Reply](#)



**Avishek Kumar**

May 18, 2015 at 11:30 pm

Welcome Manu, Keep Connected!

[Reply](#)

**Tachyon**

May 16, 2015 at 4:40 am

Great post. Thanks, I've reshared it to G+

One note, "su" stands for "switch user" not "Suitable User". Minor note.

At least that's still representative of what it actually does, unlike when most people refer to it as "Super User", thinking it only grants root user status, which is utterly incorrect.

[Reply](#)

Admin

**Ravi Saive**

May 18, 2015 at 11:08 am

@Tachyon,

thanks corrected in the write up..

[Reply](#)

## Got Something to Say? Join the Discussion...

*Thank you for taking the time to share your thoughts with us. We appreciate your decision to leave a comment and value your contribution to the discussion. It's important to note that we moderate all comments in accordance with our [comment policy](#) to ensure a respectful and constructive conversation.*

*Rest assured that your email address will remain private and will not be published or shared with anyone. We prioritize the privacy and security of our users.*

☐ Save my name, email, and website in this browser for the next time I comment.

Post Comment

## Do You Enjoy My Blog?

Support from readers like YOU keeps this blog running. Buying me a cup of coffee is a simple and affordable way to show your appreciation and help keep the posts coming!

Buy Me a Coffee

## Linux Commands and Tools

### How to Use Udev for Device Detection and Management in Linux

[How to Permanently Disable Swap Partition in Linux](#)

[Pscp – Transfer/Copy Files to Multiple Linux Servers Using Single Shell](#)

[How to Fix “Command ‘pip3’ not found” Error in Linux](#)

[How to Monitor Linux Users Activity with psacct or acct Tools](#)

[How to Disable Shutdown and Reboot Commands in Linux](#)

## Linux Server Monitoring Tools

[Dool – All-in-One Linux Server Performance Monitoring Tool](#)

[Hegemon – A Modular System Monitoring Tool for Linux](#)

[Nmon – Monitor Linux System and Network Performance](#)

[Glances – An Advanced Real Time System Monitoring Tool for Linux](#)

[Will ‘Htop’ Replace Default ‘Top’ Monitoring Tool in Linux?](#)

[How to Use ‘fsck’ to Repair Linux File System Errors](#)

## Learn Linux Tricks & Tips

[How to Find Out List of All Open Ports in Linux](#)

[6 Useful Tools to Remember Linux Commands Forever](#)

[How to Find Number of Files in a Directory and Subdirectories](#)

[How to Increase Disk Inode Number in Linux](#)

[How to Create Hard and Symbolic Links in Linux](#)

[Progress – Show Percentage of Copied Data for \(cp, mv, dd, tar\) Commands](#)

## Best Linux Tools

[Top 5 Open Source Collaboration Platforms for Linux in 2024](#)

**8 Best Command-Line/Terminal Email Clients for Linux**

**13 Most Used Microsoft Office Alternatives for Linux**

**8 Best IRC Clients for Linux in 2024**

**10 Best Flowchart and Diagramming Software for Linux**

**10 Best API Gateways and Management Tools in 2024**

Tecmint: Linux Howtos, Tutorials & Guides © 2024. All Rights Reserved.

The material in this site cannot be republished either online or offline, without our permission.

Hosting Sponsored by : [Linode Cloud Hosting](#)