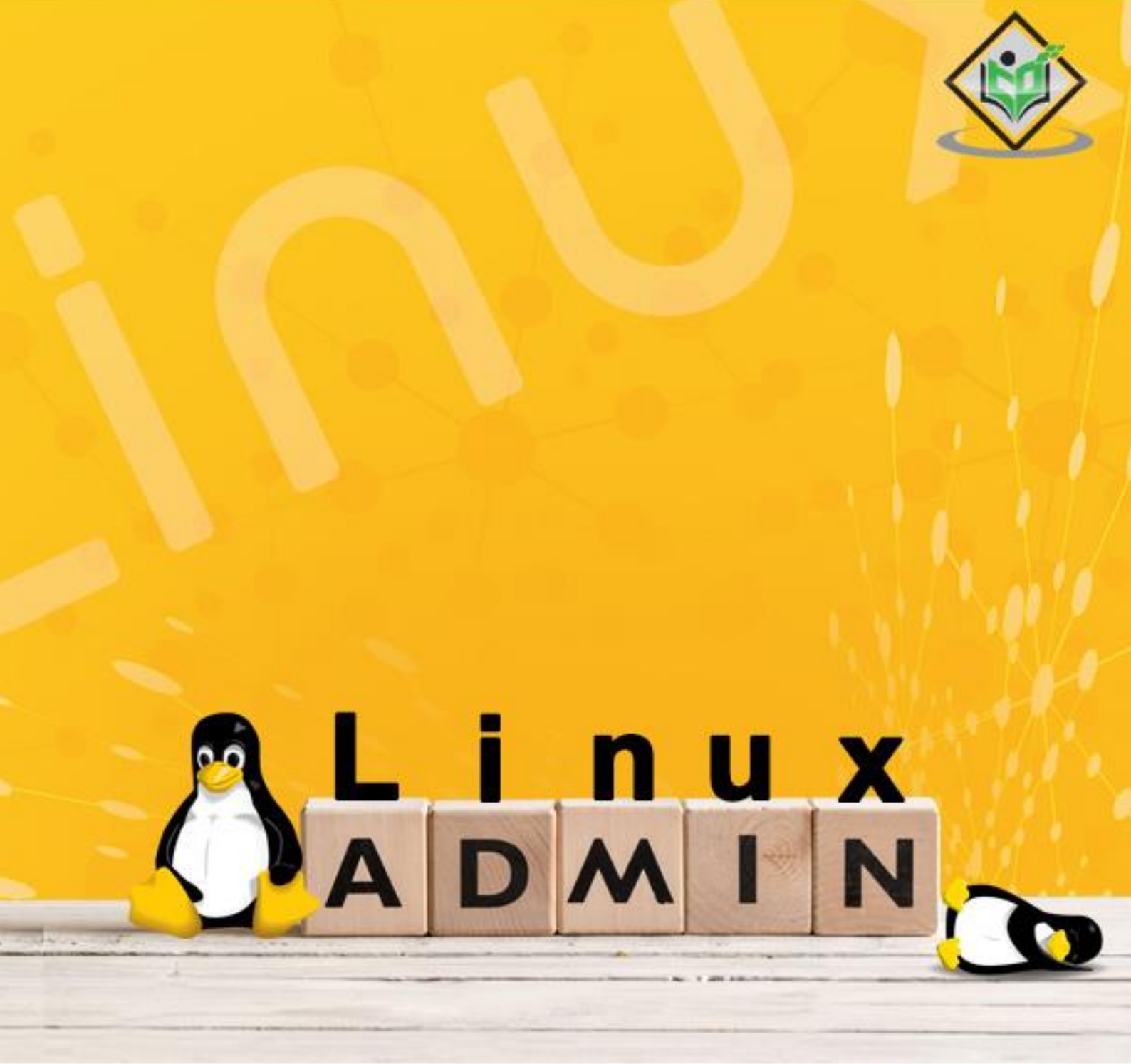

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Linux was designed based on the Unix philosophy of "small, precise tools chained together simplifying larger tasks". Linux, at its root, does not have large single-purpose applications for one specific use a lot of the time. Instead, there are hundreds of basic utilities that when combined offer great power to accomplish big tasks with efficiency.

Unique amongst business class Linux distributions, CentOS stays true to the open-source nature that Linux was founded on. This tutorial gives a complete understanding on Linux Admin and explains how to use it for benefit.

Audience

This tutorial has been prepared for beginners to help them understand the fundamentals of Linux Admin. It will specifically be useful for Linux administration professionals. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to the next levels.

Prerequisites

Before you go ahead with this tutorial, we assume that you have a basic knowledge of Linux and Administration fundamentals.

Copyright & Disclaimer

Table of Contents

1. Linux Admin – CentOS Overview

Unique among business class Linux distributions, CentOS stays true to the open-source nature that Linux was founded on. The first Linux kernel was developed by a college student at the University of Helsinki (Linus Torvalds) and combined with the GNU utilities founded and promoted by Richard Stallman. CentOS has a proven, open-source licensing that can power today's business world.

CentOS has quickly become one of the most prolific server platforms in the world. Any Linux Administrator, when seeking employment, is bound to come across the words: "CentOS Linux Experience Preferred". From startups to *Fortune 10* tech titans, CentOS has placed itself amongst the higher echelons of server operating systems worldwide.

What makes CentOS stand out from other Linux distributions is a great combination of:

- Open source licensing
- Dedicated user-base of Linux professionals
- Good hardware support
- Rock-solid stability and reliability
- Focus on security and updates
- Strict adherence to software packaging standards needed in a corporate environment

Before starting the lessons, we assume that the readers have a basic knowledge of Linux and Administration fundamentals such as:

- What is the root use?
- The power of the root user
- Basic concept of security groups and users
- Experience using a Linux terminal emulator
- Fundamental networking concepts
- Fundamental understanding of interpreted programming languages (Perl, Python, Ruby)
- Networking protocols such as HTTP, LDAP, FTP, IMAP, SMTP
- Cores that compose a computer operating system: file system, drivers, and the kernel

2. Linux Admin – Basic CentOS Linux Commands

Before learning the tools of a CentOS Linux Administrator, it is important to note the philosophy behind the Linux administration command line.

Linux was designed based on the Unix philosophy of "small, precise tools chained together simplifying larger tasks". Linux, at its root, does not have large single-purpose applications for one specific use a lot of the time. Instead, there are hundreds of basic utilities that when combined offer great power to accomplish big tasks with efficiency.

Examples of the Linux Philosophy

For example, if an administrator wants a listing of all the current users on a system, the following chained commands can be used to get a list of all system users. On execution of the command, the users are on the system are listed in an alphabetical order.

```
[root@centosLocal centos]# cut /etc/passwrd -d":" -f1 | sort
abrt
adm
avahi
bin
centos
chrony
colord
daemon
dbus
```

It is easy to export this list into a text file using the following command.

```
[root@localhost /]# cut /etc/passwd -d ":" -f1 > system_users.txt
[root@localhost /]# cat ./system_users.txt | sort | wc –l
40
[root@localhost /]#
```

It is also possible to compare the user list with an export at a later date.

```
[root@centosLocal centos]#  cut /etc/passwd -d ":" -f1 > system_users002.txt &&
cat system_users002.txt | sort | wc -l
41
[root@centosLocal centos]# diff ./system_users.txt ./system_users002.txt
evilBackdoor
[root@centosLocal centos]#
```

A new user, "evilBackdoor", has been added to the system.

With this approach of small tools chained to accomplish bigger tasks, it is simpler to make a script performing these commands, than automatically email results at regular time intervals.

Basic Commands every Linux Administrator should be proficient in are:

- vim
- grep
- more
- less
- tail
- head
- wc
- sort
- uniq
- tee
- cat
- cut
- sed
- tr
- paste

In the Linux world, Administrators use **filtering** commands every day to parse logs, filter command output, and perform actions with interactive shell scripts. As mentioned, the power of these commands come in their ability to modify one another through a process called **piping**.

The following command shows how many words begin with the letter **a** from the CentOS main user dictionary.

```
[root@centosLocal ~]# egrep '^a.*$' /usr/share/dict/words | wc -l
 25192
[root@centosLocal ~]#
```

Using the vi / vim Text Editor

vim represents a newer, improved version of the **vi** text editor for Linux. vim is installed by default on CentOS 7, the most recent version of CentOS. However, some older and minimal base installs will only include the original vi by default.

The biggest difference between vi and vim are advanced ease-of-use features such as moving the cursor with the arrow keys. Where vim will allow the user to navigate a text file with the arrow keys, vi is restricted to using the "h", "j", "k", "l" keys, listed as follows.

vi text document navigation:

Key	Action
j	Move down one line
k	Move up one line
l	Move to the left on character
h	Move to the right one character

Using vim the same actions can be accomplished with the arrow keys on a standard English (and other common language) based qwerty, keyboard layout. Similarly, vi will often not interpret the numeric keypad on as well.

Mostly, these days, vi will be symlinked to vim. If you ever find it frustrating your arrow keys are doing things unexpected when pressed, try using your package manager to install vim.

vim uses the concept of modes when manipulating and opening files. The two modes we will focus on are:

- **normal**: This is the mode vim uses when a file is first opened, and allows for entering commands.
- **insert**: The insert mode is used to actually edit text in a file.

Let's open a file in vim. We will use the CentOS default dictionary located at /usr/share/dict:

```
[root@localhost rdc]# cp /usr/share/dict/words
```

What you see is the text file opened in normalmode. Now practice navigating the document using the arrow keys. Also, try using the h,j,k and lkeys to navigate the document.

Vim expects us to send commands for file operations. To enable line number, use the colon key: *shift+:*. Your cursor will now appear at the bottom of the document. Type "set nu" and then hit enter.

```
:set nu
```

Now, we will always know where in the file we are. This is also a necessity when programming in vim. Yes! vim has the best syntax highlighting and can be used for making Ruby, Perl, Python, Bash, PHP, and other scripts.

Following table lists the most common commands in normal mode.

Command	Action
G	Go to the end of the file
gg	Go to the beginning of the file

x	Delete the selected character
u	Undo the last modifications
Enter	Jump forward by lines
dd	Delete the entire line
?	Search for a string
/	Proceed to the next search occurrence

Please try the following tasks in vim, to become familiar with it.

- Search for the string "test", then first 5 occurrences
- Move to the beginning of the document after finding the first 5 occurrences of "test"
- Go to line 100 using enter
- Delete the entire word using "x"
- Undo the deletions using "u"
- Delete the entire line using "dd"
- Reconstruct the line using "u"

We will pretend that we made edits on a critical file and want to be sure **not** to save any unintended changes. Hit the **shift+:** and type: **q!**. This will exit vim, discarding any changes made.

Now, we want to actually edit a file in vim: at the console type: vim myfile.txt

We are now looking at a blank text buffer in vim. Let's write something: say - hit "i".

vim is now in insert mode, allowing us to make edits to a file just like in Notepad. Type a few paragraphs in your buffer, whatever you want. Later, use the following steps to save the file:

- **Step 1**: Press the escape key
- **Step 2**: Press shift+:
- **Step 3**: type w myfile.txt:w and hit Enter
- **Step 4**: Press *shift+:*
- **Step 5**: Type q! and hit Enter

We have just created a text-file named, myfile.txt and saved it:

```
[root@localhost]# cat myfile.txt
 this is my txt file.
[root@localhost]#
```

Linux File Input/Output Redirection

The pipe character **"|"**, will take an output from the first command, passing it to the next command. This is known as Standard Output or **stdout**. The other common Linux redirector is Standard Input or **stdin**.

Following are two examples; first using the **cat** command putting the file contents to **stdout**. Second using **cat** to read a file with the **standardinput** redirector outputting its contents.

STDOUT

```
[root@centosLocal centos]# cat output.txt
Hello,
I am coming from Standard output or STDOUT.
[root@centosLocal centos]#
```

STDIN

```
[root@centosLocal centos]# cat < stdin.txt
Hello,
I am being read form Standard input, STDIN.
[root@centosLocal centos]#
```

Now, let's "pipe" the stdout of cat to another command.

```
[root@centosLocal centos]# cat output.txt | wc -l
2
[root@centosLocal centos]#
```

Above, we passed *cat's*stdout to wc for processing the pipe character. wc then processed the output from cat printing the line count of output.txt to the terminal. Think of the pipe character as a "pipe" passing output from one command, to be processed by the next command.

Following are the key concepts to remember when dealing with command redirection.

Number	File Descriptor	Character
0	standard input	<
1	standard output	>
2	standard error	
	append stdout	>>
	assign redirection	&
	pipe stdout	|

Grep Command

grep is commonly used by administrators to:

- Find files with a specific text string
- Search for a text string in logs
- Filter command out, focusing on a particular string

Following is a list of common switches used with grep.

Switch	Action
-E *	Interpret pattern as a regular expression
-G *	Interpret pattern as a basic regular expression
-c	Suppress normal output, only show the number of matches
-l	List files with matches
-n	Prefix each
-m	Stop reading after the number of matching lines
-o	Print only the matching parts of matching lines, per line (useful with pattern matches)
-v	Invert matches, showing non-matches
-i	Case insensitive search
-r	Use grep recursively

Search for errors **X** Server errors in **Xorg** logs:

```
[root@centosLocal log]# grep error ./Xorg*.log
    ./Xorg.0.log:   (WW) warning, (EE) error, (NI) not implemented, (??) unknown.
    ./Xorg.1.log:   (WW) warning, (EE) error, (NI) not implemented, (??) unknown.
    ./Xorg.9.log:   (WW) warning, (EE) error, (NI) not implemented, (??) unknown.
[root@centosLocal log]#
```

Check for possible RDP attacks on an imported Windows Server firewall log.

```
[root@centosLocal Documents]# grep 3389 ./pfirewall.log | grep " 146." | wc -l
 326
[root@centosLocal Documents]#
```

As seen in the above example, we had 326 Remote Desktop login attempts from IPv4 class A range in less than 24 hours. The offending IP Address has been hidden for privacy reasons. These were all from the same IPv4 address. Quick as that, we have tangible evidence to block some IPv4 ranges in firewalls.

grep can be a fairly complex command. However, a Linux administrator needs to get a firm grasp on. In an average day, a Linux System Admin can use a dozen variations of grep.

more and less Command

Both **more** and **less** commands allow pagination of large text files. When perusing large files, it is not always possible to use **grep** unless we know an exact string to search. So we would want to use either more or less.

Typically, **less** is the preferred choice, as it allows both forward and backward perusal of paginated text. However, less may not be available on default installations of older Linux distributions and even some modern Unix operating systems.

```
[root@centosLocal Documents]# grep "192.168" ./pfirewall.log | more
2016-01-07 15:36:34 DROP UDP 192.168.0.1 255.255.255.255 68 67 328 - - - - -  -  RECEIVE
2016-01-07 15:36:38 DROP UDP 192.168.0.21 255.255.255.255 68 67 328 - - - - -- - RECEIVE
2016-01-07 15:36:45 DROP ICMP 192.168.0.24 224.0.0.1 - - -- - - - - -- - - - -   RECEIVE
2016-01-07 15:37:07 DROP UDP 192.168.0.21 255.255.255.255 68 67 328 - - - -  - - RECEIVE
2016-01-07 15:37:52 DROP UDP 192.168.0.78 255.255.255.255 68 67 328 - - - -  - - RECEIVE
2016-01-07 15:37:52 ALLOW UDP 192.168.0.78 255.255.255.255 67 68 0 - - - - -- -  RECEIVE
2016-01-07 15:37:53 ALLOW UDP 192.168.0.78 224.0.0.252 51571 5355 0 - - - -  - - RECEIVE
```

Usually less is preferred, because less really offers more than more.

```
2016-01-07 15:43:53 DROP UDP 192.168.1.73 255.255.255.255 68 67 328 - - - - - -  RECEIVE
2016-01-07 15:44:17 ALLOW UDP 192.168.1.18 224.0.0.252 54526 5355 0 - - - - - -  RECEIVE
2016-01-07 15:44:23 DROP UDP 192.168.1.57 255.255.255.255 68 67 328 - - - - - -  RECEIVE
2016-01-07 15:44:33 DROP UDP 192.168.1.88 255.255.255.255 68 67 328 - - - - - -  RECEIVE
2016-01-07 15:44:33 ALLOW UDP 192.168.1.4 255.255.255.255 67 68 0 - - - - - - -  RECEIVE
2016-01-07 15:44:41 DROP UDP 192.168.1.126 255.255.255.255 68 67 328 - - - - - - RECEIVE
2016-01-07 15:44:43 DROP UDP 192.168.1.112 255.255.255.255 68 67 328 - - - - - - RECEIVE
2016-01-07 15:44:56 DROP ICMP 192.168.1.240 224.0.0.1 - - 36 - - - - 9 0 -       RECEIVE
2016-01-07 15:45:57 ALLOW UDP 192.168.1.47 192.168.1.255 138 138 0 - - - - - -      SEND
2016-01-07 15:49:13 DROP ICMP 192.168.1.241 224.0.0.1 - - 36 - - - - 9 0 -       RECEIVE
2016-01-07 15:49:38 DROP UDP 192.168.1.68 255.255.255.255 68 67 328 - - - - - -  RECEIVE
2016-01-07 15:49:38 ALLOW UDP 192.168.1.4 255.255.255.255 67 68 0 - - - - - -    RECEIVE
2016-01-07 15:49:39 DROP UDP 192.168.1.93 255.255.255.255 68 67 328 - - - - -    RECEIVE
:
```

As shown above, when invoked less opens into a new buffer separate from the shell prompt. When trying less, it sometimes may give an error as follows:

```
bash: less: command not found...
```

Either use more or install less from the source of the package manager. But less should be included on all modern Linux Distributions and even ported to Unix platforms. Some will even **symlink** more to less.

tail Command

tail will output (stdout) the last part of a text file. Most useful when perusing long text files and we only need to see the current updates.

Switch	Action
-c	Output last denoted in kilobytes
-n	Output n number of lines from eof
-f	Follow - output data as the file grows
-q	No headers, only file content

A useful switch option for tail is **-f**. The -f switch is really useful for real-time troubleshooting of logs. A good example is when a user has issues with remote login. Using the -f option, piping the output of VPN logs into grep, then the filtering user id is allowed to watch login attempts from the troubled user. It turned out auth credentials were garbled.

Upon further inspection, they had changed the encryption protocol of the RAS client. Something that could have taken a long time to figure out by asking questions and walking a user through client config. settings step by step.

Using tail with the -f switch to watch wpa supplicant logs as wifi is connected to a new AP.

```
[root@centosLocal Documents]# tail -f /var/log/wpa_supplicant.log-20161222

wlp0su: CTRL-EVENT-DISCONNECTED bssid=ee:ee:ee:12:34:56 reason=3
locally_generated=1

dbus: wpas_dbus_bss_signal_prop_changed: Unknown Property value 7

wlp0su: SME: Trying to authenticate with ff:ff:ff:78:90:12 (SSID='WiFi12345'
freq=5180 MHz)

wlp0su: Trying to associate with ff:ff:ff:78:90:12 (SSID='WiFi12345' freq=5180 MHz)

wlp0su: Associated with ff:ff:ff:78:90:12

wlp0su: WPA: Key negotiation completed with ff:ff:ff:78:90:12 [PTK=CCMP GTK=CCMP]

wlp0su: CTRL-EVENT-CONNECTED - Connection to ff:ff:ff:78:90:12 completed [id=0
id_str=]
```

When a Wifi connection fails, it could assist in troubleshooting issues in real-time.

head Command

head is a basic opposite of tail in relation to what part of the file operations are performed on. By default, **head** will read the first 10 lines of a file.

head offers similar options as tail:

Switch	Action
-c	Output last denoted in kilobytes
-n	Output n number of lines from eof
-q	No headers only file content

Note: Head offers no **-f** option, since the files are appended from the bottom.

head is useful for reading descriptions of configuration files. When making such a file, it is a good idea to use the first 10 lines effectively.

```
[root@centosLocal centos]# head /etc/sudoers
## Sudoers allows particular users to run various commands as
## the root user, without needing the root password.
##
## Examples are provided at the bottom of the file for collections
## of related commands, which can then be delegated out to particular
## users or groups.
##
## This file must be edited with the 'visudo' command.

## Host Aliases
[root@centosLocal centos]#
```

wc Command

wc is useful for counting occurrences in a file. It helps print newline, word, ad byte count from each file. Most useful is when combined with grep to show matches for a certain pattern.

Switch	Action
-c	Bytes
-m	Character count
-l	Line count
-L	Length of the longest line

We can see our system has 5 users with a group id of 0. Then upon further inspection only the root user has shell access.

```
[root@centosLocal centos]# cat /etc/passwd | cut -d":" -f4 | grep "^0" |  wc -l
5
[root@centosLocal centos]# cat /etc/passwd | cut -d":" -f4,5,6,7 | grep "^0"
0:root:/root:/bin/bash
0:sync:/sbin:/bin/sync
0:shutdown:/sbin:/sbin/shutdown
0:halt:/sbin:/sbin/halt
0:operator:/root:/sbin/nologin
[root@centosLocal centos]#
```

sort Command

sort has several optimizations for sorting based on datatypes. Theis command writes sorted concatenation of all files to standard output. However, be weary, complex sort operations on large files of a few GigaBytes can impede the system performance.

When running a production server with limited CPU and/or memory availability, it is recommended to offload these larger files to a workstation for sorting operations during peak business hours.

Switch	Action
-b	Ignore leading blank lines
-d	Dictionary order, consider only blanks and alphanumeric characters
-f	Ignore case, folding lower and upper characters
-g	General numeric sort
-M	Month sort
-h	Human readable numeric sort 1KB, 1MB, 1GB
-R	Random sort
-m	Merge already sorted files

Feel free to copy the tabular text below and follow along with our sort examples. Be sure each column is separated with a tab character.

first name	last name	office
Ted	Daniel	101
Jenny	Colon	608

Dana	Maxwell	602
Marian	Little	903
Bobbie	Chapman	403
Nicolas	Singleton	203
Dale	Barton	901
Aaron	Dennis	305
Santos	Andrews	504
Jacqueline	Neal	102
Billy	Crawford	301
Rosa	Summers	405
Kellie	Curtis	903
Matt	Davis	305
Gina	Carr	902
Francisco	Gilbert	101
Sidney	Mack	901
Heidi	Simmons	204
Cristina	Torres	206
Sonya	Weaver	403
Donald	Evans	403
Gwendolyn	Chambers	108
Antonia	Lucas	901
Blanche	Hayes	603
Carrie	Todd	201
Terence	Anderson	501
Joan	Parsons	102
Rose	Fisher	304
Malcolm	Matthews	702

Using **sort** in its most basic, default form:

```
[root@centosLocal centos]# sort ./Documents/names.txt
Aaron   Dennis  305
Antonia Lucas   901
Billy   Crawford    301
Blanche Hayes   603
Bobbie  Chapman  403
Carrie  Todd    201
Cristina    Torres  206
Dale    Barton 901
Dana    Maxwell     602
Donald  Evans   403
Francisco   Gilbert   101
Gina    Carr    902
Gwendolyn   Chambers    108
Heidi   Simmons   204
Jacqueline  Neal    102
Jenny   Colon   608
Joan    Parsons    102
Kellie  Curtis  903
Malcolm  Matthews   702
Marian  Little  903
Matt    Davis   305
Nicolas Singleton   203
Rosa    Summers    405
Rose    Fisher  304
Santos  Andrews  504
Sidney  Mack    901
Sonya   Weaver  403
Ted Daniel  101
Terence Anderson    501
[root@centosLocal centos]#
```

Sometimes, we will want to sort files on another column, other than the first column. A sort can be applied to other columns with the **-t** and **-k** switches.

```
-t : define a file delimiter

 -k : key count to sort by (think of this as a column specified from the
delimiter.

 -n : sort in numeric order
```

Note: In some examples, we have used cat piped into grep. This was to demonstrate the concepts of piping commands. Outputting cat into grep can increase the system load hundreds of times-over with large files, while adding complex sorting. This will make veteran Linux administrators cringe.

Now that we have a good idea of how the pipe character works, this poor practice will be avoided in the chapters to follow. The key to keeping the system resources low with commands like sort, is learning to use them efficiently.

```
[root@centosLocal centos]# sort -t '    ' -k 3n ./Documents/names.txt
Ted Daniel 101
Francisco   Gilbert 101
Jacqueline  Neal    102
Joan    Parsons 102
Gwendolyn   Chambers    108
Carrie  Todd    201
Nicolas Singleton   203
Heidi   Simmons 204
Cristina    Torres  206
Billy   Crawford    301
Rose    Fisher  304
Aaron   Dennis  305
Matt    Davis   305
Bobbie  Chapman 403
Donald  Evans   403
Sonya   Weaver  403
Rosa    Summers 405
Terence Anderson    501
Santos  Andrews 504
Dana    Maxwell 602
Blanche Hayes   603
Jenny   Colon   608
Malcolm  Matthews   702
Antonia Lucas   901
```

```
Dale    Barton  901
Sidney  Mack    901
Gina    Carr    902
Kellie  Curtis  903
Marian  Little  903
[root@centosLocal centos]#
```

Now we have our list sorted by office number. The astute reader will notice something out of the ordinary after the **-t** switch; single quotes separated by what appears to be a few spaces. This was actually a literal Tab character sent to the shell. A literal Tab can be sent to the BASH shell using the key combination of***: control+Tab+v***.

Most shells will interpret the Tab key as a command. For example, auto-completion in BASH. The shell needs an escape sequence to recognize a literal Tab character. This is one reason why Tabs are not the best choice for delimiters with Linux. Generally speaking, it is best to avoid both spaces and tabs, as they can cause issues when scripting a shell.

Let us fix our names.txt file.

```
[root@centosLocal centos]# sed -i 's/\t/:/g' ./Documents/names.txt &&
cat ./Documents/names.txt
Ted:Daniel:101
Jenny:Colon:608
Dana:Maxwell:602
Marian:Little:903
Bobbie:Chapman:403
Nicolas:Singleton:203
Dale:Barton:901
Aaron:Dennis:305
Santos:Andrews:504
Jacqueline:Neal:102
Billy:Crawford:301
Rosa:Summers:405
Kellie:Curtis:903:
Matt:Davis:305
Gina:Carr:902
Francisco:Gilbert:101
Sidney:Mack:901
Heidi:Simmons:204
Cristina:Torres:206
Sonya:Weaver:403
Donald:Evans:403
```

```
Gwendolyn:Chambers:108
Antonia:Lucas:901
Blanche:Hayes:603
Carrie:Todd:201
Terence:Anderson:501
Joan:Parsons:102
Rose:Fisher:304
Malcolm: Matthews:702
[root@centosLocal centos]#
```

Now, it will be much easier to work with the text file. If someone demands it be returned to Tab delimited for another application (this is common), we can accomplish that task easily as:

```
sed -i 's/:/\t/g' ./Documents/names.txt
```

Common end-user applications will work good with Tabs as a delimiter (An Accountant does not want to see a colon separating data columns, while working on Spreadsheets.). So learning to transform characters back and forth is a good practice; it comes up often.

Note: Office uses word-processors and spreadsheets with a Graphical User Interface, running on Windows. Hence, it is common for Linux Administrators to get good at completing transformation actions, accommodating end office-users (most times, our boss will be an end user).

Introduced was a command called **sed**. sed is a stream editor and can be used as a non-interactive text editor for manipulating streams of text and files. We will learn more about sed later. However, keep in mind for now, using sed, we avoided a need to pipe several filter commands when changing our text file. Thus, making the most efficient use of the tools at hand.

We also introduced a Bash shell operator: *&&*. *&&* will run the second command only if the first command completes with a successful status of "0".

```
[root@centosLocal centos]# ls /noDir &&  echo "You cannot see me"
ls: cannot access /noDir: No such file or directory
[root@centosLocal centos]# ls /noDir ;  echo "You cannot see me"
ls: cannot access /noDir: No such file or directory
You cannot see me
[root@centosLocal centos]# ls /noDir ;  echo "You cannot see me"
```

In the above code, note the difference between *&&* and *;*? The first will only run the second command when the first has completed successfully, while *;* simply chains the commands. More on this when we get to scripting shell commands.

uniq Command

Following are the common switches used with **uniq**. This command reports or omits repeated lines.

Switch	Action
-c	Prefix lines by the number of occurrence
-i	Ignore case
-u	Only print unique lines
-w	Check chars, compare no more than *n* chars
-s	Skip chars, avoid comparing the first two N characters
-f	Skip fields, avoid comparing first N fields
-D	Print all duplicate line groups

We have briefly used uniq in a few examples prior. The **uniq** command allows us to filter the lines of files based on matches. For example, say we got a second employee named Matt Davis in Sales. Three days later, Accounting needs new estimates for Sales Participation Awards for next quarter. We can check the employee list using the following command.

```
[root@centosLocal centos]# cat ./Documents/names.txt | wc -l
30
[root@centosLocal centos]#
```

We tell them 30 people in Sales for the annual participation awards. There might be a good chance Accounting will notice a discrepancy: they only needed 29 unique award plaques produced. Let's try again:

```
[root@centosLocal Documents]# cut -d ":" -f 1,2 ./names.txt | sort | uniq | wc
-l
29
[root@centosLocal Documents]#
```

Now we have enough information to give Accounting an accurate number of unique Participation Awards for the Sales Department (they will not need to pay to have two unique plaques made. Just duplicate a second for "Matt Davis").

Note: When looking for unique lines, we always want to use sort, piping its output to uniq. If non-uniq entries are not inline sequence, they will not be seen as duplicate lines.

To quickly generate a report letting us know how many sales people share an office:

```
[root@centosLocal Documents]# sort -t":" -k3 ./names.txt | cut -d ":" -f3  |
uniq -c | sort -n
  1 100
```

```
  1 108
  1 201
  1 203
  1 204
  1 205
  1 206
  1 301
  1 304
  1 404
  1 405
  1 501
  1 504
  1 602
  1 603
  1 608
  1 702
  1 902
  2 101
  2 102
  2 305
  2 901
  2 903
  3 403
[root@centosLocal Documents]#
```

tee Command

tee is a simple command, letting an administrator write command output and view a file at the same time. This simple command can save time over first writing stdout to a file, then viewing the file contents.

Following are the common Switches used with tee.

Command	Action
-a	Append to files instead of clobber file
-i	Ignore interrupts (for advanced use in scripting mostly)

Without **tee** to both view and write files and directories in /etc, where each begins with the letter "a".

```
[root@centosLocal Documents]# ls -d /etc/a*
/etc/abrt     /etc/aliases.db    /etc/anacrontab   /etc/at-spi2
/etc/autofs.conf
/etc/auto.master.d  /etc/auto.smb  /etc/adjtime  /etc/alsa
/etc/asound.conf
/etc/audisp         /etc/autofs_ldap_auth.conf  /etc/auto.misc  /etc/avahi
/etc/aliases  /etc/alternatives  /etc/at.deny   /etc/audit   /etc/auto.master
/etc/auto.net
[root@centosLocal Documents]# ls -d /etc/a* > ./etc_report_a.txt
[root@centosLocal Documents]# cat ./etc_report_a.txt
/etc/abrt
/etc/adjtime
/etc/aliases
/etc/aliases.db
/etc/alsa
/etc/alternatives
/etc/anacrontab
/etc/asound.conf
/etc/at.deny
/etc/at-spi2
/etc/audisp
/etc/audit
/etc/autofs.conf
/etc/autofs_ldap_auth.conf
/etc/auto.master
/etc/auto.master.d
/etc/auto.misc
/etc/auto.net
/etc/auto.smb
/etc/avahi
[root@centosLocal Documents]#
```

This small task is much more efficient with the tee command.

```
[root@centosLocal Documents]# ls -d /etc/a* | tee ./etc_report_a.txt
/etc/abrt
/etc/adjtime
/etc/aliases
/etc/aliases.db
/etc/alsa
/etc/alternatives
/etc/anacrontab
/etc/asound.conf
/etc/at.deny
/etc/at-spi2
/etc/audisp
/etc/audit
/etc/autofs.conf
/etc/autofs_ldap_auth.conf
/etc/auto.master
/etc/auto.master.d
/etc/auto.misc
/etc/auto.net
/etc/auto.smb
/etc/avahi
[root@centosLocal Documents]#
```

cat Command

The **cat** command is used to concatenate files and print to standard output. Formerly, we have demonstrated both uses and abuses with the cat command. cat servers the following distinct purposes:

- Show files contents
- Write contents of one file to another file
- Combine multiple files into a single file
- Support special features: adding line numbers, showing special characters, eliminating blank lines

Switch	Action

-b	Number non-blank lines
-E	Show line ends
-T	Show tabs
-s	Squeeze blank, suppress repeated empty lines

As noted previously, when using utilities such as *grep*, *sort*, and *uniq* we want to avoid piping output from *cat* if possible. We did this for simple demonstration of piping commands earlier. However, knowing when to perform an operation with a utility like *grep* is what separates Linux Administrators from Linux end-users.

Bad Habit

```
[root@centosLocal centos]# cat /etc/passwd | sort -t: -k1  | grep ":0"
halt:x:7:0:halt:/sbin:/sbin/halt
operator:x:11:0:operator:/root:/sbin/nologin
root:x:0:0:root:/root:/bin/bash
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
sync:x:5:0:sync:/sbin:/bin/sync
[root@centosLocal centos]#
```

Good Habit

```
[root@centosLocal centos]# grep ":0" /etc/passwd | sort -t: -k 1
halt:x:7:0:halt:/sbin:/sbin/halt
operator:x:11:0:operator:/root:/sbin/nologin
root:x:0:0:root:/root:/bin/bash
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
sync:x:5:0:sync:/sbin:/bin/sync
[root@centosLocal centos]#
```

Note: piping cat to secondary commands like sort or grep should only be done when it is needed.

One common use of *cat* is when dealing with Windows formatted line breaks. Both Linux and Windows by internal design, use a different control code to represent End Of Line (EOL):

```
* Linux line break is always a Line Feed: LF or depicted as "\n".
* Windows is Carriage Return followed by a Line Feed: CR LF or depicted as
"\r\n".
* Macintosh, in all moderne releases of OS X and now macOS, has adopted the
Linux/Unix standard of LF or "\n"
```

So, let's say we open our file in a GUI text-editor like gedit or are experiencing random issues while applying filtering commands. Text appears on a single line, or filtering commands do not operate as expected.

Especially, when the text file was downloaded off the Internet, we want to check line breaks. Following is a sample output from cat showing EOL characters.

```
[root@centosLocal centos]# cat -E  ./Desktop/WinNames.txt
 $ed:Daniel:101
 $enny:Colon:608
 $ana:Maxwell:602
 $arian:Little:903
 $obbie:Chapman:403
 $icolas:Singleton:203
 $ale:Barton:901
```

Notice the preceding "**$**" on each line? Linux is reading the CR "\n", breaking the file. Then translating a Carriage Return over the first character of each file.

When viewed without the -E switch, the file looks fine:

```
[root@centosLocal centos]# cat  ./Desktop/WinNames.txt
 Ted:Daniel:101
 Jenny:Colon:608
 Dana:Maxwell:602
 Marian:Little:903
 Bobbie:Chapman:403
 Nicolas:Singleton:203
 Dale:Barton:901
```

Luckily, with Linux filtering commands this is an easy fix:

```
[root@centosLocal centos]# sed -i 's/\r$//g' ./Desktop/WinNames.txt
[root@centosLocal centos]# cat -E ./Desktop/WinNames.txt
Ted:Daniel:101$
Jenny:Colon:608$
Dana:Maxwell:602$
```

Note*:* When viewed with the -E switch, all Linux line breaks will end in $.

End of ebook preview

If you liked what you saw…

Buy it from our store @ **https://store.tutorialspoint.com**