
Linux command line for you and me Documentation

Release 0.1

Kushal Das

Feb 23, 2022

Contents:

1	Shell commands	1
1.1	Terminal emulators	1
1.2	date command	1
1.3	cal command	2
1.4	whoami command	3
1.5	id command	3
1.6	pwd command	3
1.7	cd command	3
1.8	. directory and .. directory	4
1.9	ls command	4
1.10	mkdir command	4
1.11	rm command	5
1.12	Copying a file using cp command	5
1.13	Renaming or moving a file	5
1.14	tree command	5
1.15	wc command	6
1.16	echo command	6
1.17	Redirecting the command output	6
1.18	Using > to redirect output to a file	7
1.19	Moving around in the command line	8
1.20	man pages	8
1.21	Counting files in a directory	8
1.22	Editing longer commands	9
2	File system	11
2.1	FHS	11
3	Useful commands	13
3.1	Creating soft link to a file	13
3.2	Creating hard links	13
3.3	Extracting a tar file	14
3.4	Creating a tar file	14
3.5	Vim editor	15
3.6	:q to exit vim	15
3.7	Open a new file or edit an existing file	15
3.8	Different modes of vim	15
3.9	:w to save a file	17

3.10	:q! to quit without saving	17
3.11	Becoming root user	17
3.12	Using sudo command	17
3.13	!! trick	17
3.14	Environment variables	18
3.15	Setting up environment variable values	18
3.16	locate command	18
3.17	Finding date/time in different timezones	19
3.18	Bash history	19
3.19	Sort files by size	19
4	Users and Groups	21
4.1	Finding the owner of file	21
4.2	/etc/passwd file	21
4.3	Details about groups	23
4.4	wheel group	23
4.5	Becoming superuser	23
4.6	Adding a new user	23
4.7	Changing user passwords	24
4.8	Modifying existing user details	24
4.9	Deleting a user	24
4.10	Adding a new group	24
4.11	Adding new group to an user	24
5	File permissions	25
5.1	chmod command	25
5.2	PATH variable	26
5.3	~/.bash_profile file	26
5.4	.bashrc file	27
5.5	/etc/profile file	27
5.6	which command	27
5.7	Use <i>which</i> command to see how \$PATH variable works	28
5.8	she-bang or sha-bang in executable files	28
6	Processes in Linux	29
6.1	How to view all running processes?	29
6.2	How to find a particular process?	29
6.3	How to kill/stop a particular process?	30
6.4	Finding out list of open files	30
6.5	Signals	30
6.6	top command	31
6.7	Load average	31
6.8	htop tool	31
6.9	More about Linux processes	32
6.10	/proc directory	32
6.11	/proc/cpuinfo	33
6.12	/proc/cmdline	33
6.13	/proc/meminfo	33
6.14	/proc/uptime	34
6.15	/proc/sys/ & sysctl command	34
6.16	Enabling IP forward with sysctl	34
7	Linux Services	37
7.1	What is a service?	37
7.2	What is a daemon?	37

7.3	What is the init system?	37
7.4	Units in systemd	38
7.5	.service units in systemd	38
7.6	How to find all the systemd units in the system?	38
7.7	Working with a particular service	39
7.8	Enabling or disabling a service	40
7.9	Shutdown or reboot the system using systemctl	40
7.10	journalctl	40
7.11	Finding the logs of a service	40
7.12	To view only the last N entries	41
7.13	Continuous stream of logs	41
7.14	Listing of previous boots	42
7.15	Time-based log viewing	42
7.16	Total size of the journal logs	43
8	Package management	45
8.1	dnf command	45
8.2	Searching for a package	45
8.3	Finding more information about a package	46
8.4	Installing a package	46
8.5	To list the available updates	47
8.6	To list all security updates	47
8.7	Update the packages via dnf	48
8.8	Find out the services & applications need restart after update in Fedora/CentOS/RHEL	48
8.9	Automatic updates in dnf systems	50
8.10	apt command	50
8.11	apt update	50
8.12	Installing a package via apt	51
8.13	apt-cache search	51
8.14	Listing upgrades	51
8.15	Upgrading packages	52
8.16	Figuring out which services/processes need restart after package upgrade in Debian	52
8.17	Listing available security updates in Debian systems	53
8.18	Unattended upgrades in Debian systems	53
9	SELinux	55
9.1	SELinux Modes	56
9.2	getenforce	56
9.3	setenforce	56
9.4	Labels/Contexts	57
9.5	Checking contexts of files/directories or processes	57
9.6	SELinux booleans	58
10	File system mounting	59
10.1	Connecting USB drives to your system	60
10.2	Mounting a device	60
10.3	Unmounting	60
10.4	Encrypting drives with LUKS (for only Linux)	60
10.5	Encrypting drives for any OS using Veracrypt	61
11	Networking commands	63
11.1	Finding the IP address	63
11.2	ping command	63
11.3	Short note about DNS	64
11.4	/etc/hosts	64

11.5	/etc/resolv.conf	64
11.6	systemd-resolved controlled name resolution	64
11.7	resolvectl command	66
11.8	host command	67
11.9	dig command	67
11.10	ss command	70
11.11	traceroute command	70
11.12	tracpath command	71
11.13	Remote login to a computer using ssh tool	71
11.14	ssh key generation	72
11.15	ssh-copy-id	72
11.16	Stop and disable the sshd service	73
11.17	Disable password based login for ssh	73
11.18	How to find active (open) network connections from your computer?	73
11.19	To know about ports	73
12	Linux Firewall	75
12.1	Installation	75
12.2	Tables, chains and rules	75
12.3	filter table	76
12.4	nat table	76
12.5	iptables command	76
12.6	View the existing rules	77
12.7	Appending rules to INPUT chain	77
12.8	Flushing all rules	78
12.9	Example of a series of rules	78
12.10	Delete a rule based on rule number	79
12.11	Delete a rule directly	79
12.12	Saving the rules	79
12.13	A blog post from Major Hayden	80
12.14	Debugging firewall rules	80
13	Random things	81
13.1	w command	81
13.2	How long is the system running?	81
13.3	Finding CPU time of a command	81
13.4	dmesg command	82
13.5	Setting up cron jobs	82
13.6	Finding out details about previous logins or system reboots	83
14	Whats next?	85
15	Workbook	87
15.1	How to use this workbook?	87
15.2	copy paste	88
15.3	Find your user id	88
15.4	Creating softlinks	88
15.5	Basic vim usage	89
15.6	Adding a new user	89
15.7	Deleting an existing user	89
15.8	Finding the IP address of dgplug.org	89
15.9	Change the local timezone of the system	89
15.10	Add sudo access to an user	90
16	Advanced section	91

17 Containers	93
18 Team	95
19 Indices and tables	97
Index	99

In Linux the shell (or terminal) is the lifeline of the developer, and of any power user. Things which can be done on the GUI (by clicking on different buttons), can be done much more efficiently on the terminal by using commands. Maybe one can not remember all the commands, but with regular usage one can easily remember the most useful ones.

The following guide will introduce you to a minimal set of basic commands required to use your Linux computer efficiently.

1.1 Terminal emulators

The above is the screenshot of the Gnome terminal application. As you can see the command prompt contains the following information:

```
[username@hostname directoryname]
```

In our case the username is *babai*, hostname is *kdas-laptop*, and directory is mentioned as *~*. This *~* is a special character in our case. It means the home directory of the user. In our case the home directory path is */home/babai/*.

The Gnome terminal is one of many implementations of terminal emulators. Different Linux environments may come pre-installed with different terminals.

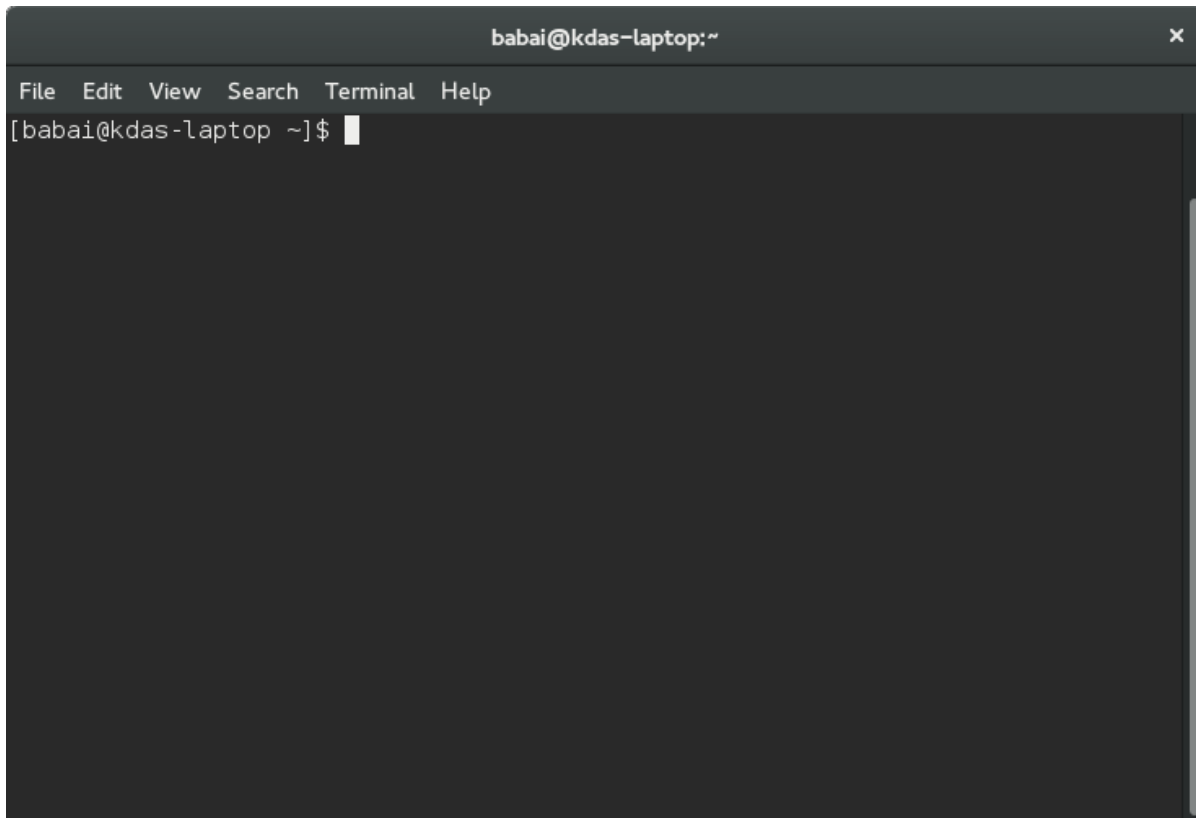
Read the articles on Wikipedia to learn about [computer terminals](#), [terminal emulators](#) and [shell](#).

1.2 date command

date command prints the current date time.

```
$ date
Sun Jun 25 10:13:44 IST 2017
```

In case you want to know the current date/time in UTC, use the following command. (I added this in 2018, so please do not get confused at the date.)



```
$ date -u
Mon May 21 01:43:47 UTC 2018
```

If you want to see yesterday's date, or a 10 days ago, you can even do that.

```
$ date --date="yesterday"
Fri Apr 9 07:09:01 PM IST 2021

$ date --date="10 days ago"
Wed Mar 31 07:09:06 PM IST 2021
```

1.3 cal command

cal command is used to display a calendar in your shell, by default it will display the current month.

```
$ cal
      June 2017
Su Mo Tu We Th Fr Sa
          1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30

$ cal 07 2017
```

(continues on next page)

(continued from previous page)

```
    July 2017
Su Mo Tu We Th Fr Sa
                1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

1.4 whoami command

whoami command will tell you which user account you are using in this system.

```
$ whoami
fedora
```

1.5 id command

id prints real user id, and various other details related to the account.

```
$ id
uid=1000(fedora) gid=1000(fedora) groups=1000(fedora),4(adm),10(wheel),190(systemd-
→journal) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

1.6 pwd command

pwd command, short for *print working directory*, will help you to find out the absolute path of the current directory. Let us see an example below:

```
[babai@kdas-laptop ~]$ pwd
/home/babai
```

1.7 cd command

The next command we will learn is *cd*, short for *change directory*. This command will help you to change your current directory. We will move to */tmp* directory in our example.

```
[babai@kdas-laptop ~]$ cd /tmp
[babai@kdas-laptop tmp]$ pwd
/tmp
[babai@kdas-laptop tmp]$ cd ~
[babai@kdas-laptop ~]$ pwd
/home/babai
```

Here you can see that first we moved to */tmp* directory, and then we moved back to the home directory by using *~* character.

1.8 . directory and .. directory

. and .. has special meaning in the Linux. . means the current directory and .. means the parent directory. We can use these in various situations for daily activities.

```
$ cd ..
```

The above command changes the current directory to the parent directory.

1.9 ls command

We use *ls* command to *list* the files and directories inside any given directory. If you use *ls* command without any argument, then it will work on the current directory. We will see few examples of the command below.

```
[babai@kdas-laptop ~]$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
[babai@kdas-laptop ~]$ ls /tmp/
cpython          systemd-private-759094c89c594c07a90156139ec4b969-colord.service-
↳hwUlhR
hogsuspend       systemd-private-759094c89c594c07a90156139ec4b969-rtkit-daemon.
↳service-AwylGa
hsperfddata_babai tracker-extract-files.1000
plugtmp          tracker-extract-files.1002
[babai@kdas-laptop ~]$ ls /
bin  cpython  etc  lib  lost+found  mnt  proc  run  srv  sysroot  usr
boot dev    home lib64 media      opt  root  sbin sys tmp  var
```

In the last two commands we provided a path as the argument to the *ls* command. / is a special directory, which represents root directory in Linux filesystem. You will learn more about that in the next chapter.

1.10 mkdir command

We can create new directories using *mkdir* command. For our example we will create a *code* directory inside our home directory.

```
[babai@kdas-laptop ~]$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
[babai@kdas-laptop ~]$ mkdir code
[babai@kdas-laptop ~]$ ls
code Desktop Documents Downloads Music Pictures Public Templates Videos
```

We can also create nested directories in a single command using the *-p* option.

```
[babai@kdas-laptop ~]$ mkdir -p dir1/dir2/dir3
[babai@kdas-laptop ~]$ ls dir1/ dir1/dir2/
dir1/:
dir2

dir1/dir2/:
dir3
```

1.11 rm command

rm command is used to *remove* a file, or directory. The *-r* option is being used to remove in a recursive way. With *-f* you *force* the removal, ignoring errors and never prompt. You can chain the flags, so instead of *rm -r -f* you can as well type *rm -rf*. But, always double check before you use *rm -rf* command, if you by mistake give this command in your home directory, or any other important directory, it will not ask to confirm, but it will delete everything there. So, please be careful *and read twice before pressing enter key*.

```
[babai@kdas-laptop ~]$ rm -rf dir1/dir2/dir3
[babai@kdas-laptop ~]$ ls dir1/ dir1/dir2/
dir1/:
dir2
dir1/dir2/:
```

1.12 Copying a file using cp command

We use the *cp* command to *copy* a file in the Linux shell. To copy a folder with its contents recursively use the *cp* command with the *-r* flag. We use the *cp file_to_copy new_location* format. In the example below, we are copying the *hello.txt* to *hello2.txt*.

```
$ cp hello.txt hello2.txt
$ ls -l
-rw-rw-r--. 1 fedora fedora 75 Jun 25 04:47 hello2.txt
-rw-rw-r--. 1 fedora fedora 75 Jun 25 04:33 hello.txt
```

In another example, I will copy the file *passwordauthno.png* from the Pictures directory in my home directory to the current directory.

```
$ cp ~/Pictures/passwordauthno.png .
```

In the following example, I will be copying the *images* directory (and everything inside it) from the *Downloads* directory under home to the */tmp/* directory.

```
$ cp -r ~/Downloads/images /tmp/
```

1.13 Renaming or moving a file

The *mv* command is used to rename or *move* a file or directory. In the following example, the file *hello.txt* is renamed to *nothello.txt*.

```
$ mv hello.txt nothello.txt
$ ls -l
-rw-rw-r--. 1 fedora fedora 75 Jun 25 04:33 nothello.txt
```

1.14 tree command

tree command prints the directory structure in a nice visual tree design way.

```
[babai@kdas-laptop ~]$ tree
.
├── code
├── Desktop
├── dir1
│   └── dir2
├── Documents
├── Downloads
├── Music
├── Pictures
│   └── terminal1.png
├── Public
├── Templates
└── Videos
```

1.15 wc command

wc, short for *word count*, is an useful command which can help us to count newlines, words and bytes of a file.

```
$ cat hello.txt
HI that is a file.
This is the second line.
And we also have a third line.
$ wc -l hello.txt
3 hello.txt
$ wc -w hello.txt
17 hello.txt
```

The *-l* flag finds the number of lines in a file, *-w* counts the number of words in the file.

1.16 echo command

echo command echoes any given string to the display.

```
$ echo "Hello"
Hello
```

1.17 Redirecting the command output

In Linux shells, we can redirect the command output to a file, or as input to another command. The pipe operator *|* is the most common way to do so. Using this we can now count the number of directories in the root (*/*) directory very easily.

```
$ ls /
bin boot dev etc home lib lib64 lost+found media mnt opt proc root run
↳sbin srv sys tmp usr var
$ ls / | wc -w
20
```

The *|* is known as pipe. To know more about this, watch [this video](#).

1.18 Using > to redirect output to a file

We can use > to redirect the output of one command to a file, if the file exists this will remove the old content and only keep the input. We can use >> to append to a file, means it will keep all the old content, and it will add the new input to the end of the file.

```
$ ls / > details.txt
$ cat details.txt
bin
boot
dev
etc
home
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
$ ls /usr/ > details.txt
$ cat details.txt
bin
games
include
lib
lib64
libexec
local
sbin
share
src
tmp
$ ls -l /tmp/ >> details.txt
$ cat details.txt
bin
games
include
lib
lib64
libexec
local
sbin
share
src
tmp
total 776
-rwxrwxr-x. 1 fedora fedora      34 Jun 24 07:56 helol.py
-rw-----. 1 fedora fedora 784756 Jun 23 10:49 tmp3lDEho
```

1.19 Moving around in the command line

There are key shortcuts available in Bash which will help you to move around faster. They are by the way very similar to the standard *emacs* keybindings, a number of key combinations that you will discover in many places and therefore are very handy to memorize and internalize. The following table is a good starting point.

Key combination	Action
Ctrl + A	Move to the beginning of the line
Ctrl + E	Move to the end of the line
Alt + B	Move to the previous word
Alt + F	Move to the next word
Ctrl + U	Cuts to the beginning of the line
Ctrl + K	Cuts to the end of the line
Ctrl + W	Cuts the previous word
Ctrl + P	Browse previously entered commands
Ctrl + R	Reverse search for previously entered commands
Ctrl + Y	Pastes the text in buffer

1.20 man pages

man shows the system's manual pages. This is the command we use to view the help document (manual page) for any command. The man pages are organized based on *sections*, and if the same command is found in many different sections, only the first one is shown.

The general syntax is *man section command*. Example **man 7 signal**.

You can know about different sections below. Press *q* to quit the program.

```
1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions eg /etc/passwd
6 Games
7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]
```

1.21 Counting files in a directory

Normally *ls* commands shows all the files and directories in multiple column. But if you pipe the output to any another command, then it prints one name in a line. We can combine that with *wc -l* to count the number of files in a directory.

```
ls | wc -l
73
```


1.22 Editing longer commands

If you are typing a long command or something multi-line, then you can type *Ctrl-x-e*, press Control button, and then *x* and then *e* key. This will open up a temporary editor using the *\$EDITOR*.

Now you know a few really basic, Linux commands. Before we can learn anything else, we should look into how files and directories are structured inside a Linux system.

2.1 FHS

```
$ ls /  
bin boot dev etc home lib lib64 lost+found mc media mnt opt output proc   
↳root run sbin srv sys tmp usr var
```

/ is the root directory of your file system. It's under this directory, that all the other files and directories reside. There's a [Filesystem Hierarchy Standard\(FHS\)](#), which talks about these different directories, and what kinds of files are located in which directory.

In this chapter, we will learn about a few more commands which we may have to use in daily life.

3.1 Creating soft link to a file

Soft link or symbolic links are a special kind of file, which actually point to some other file using either related or absolute paths. We can create soft links using **ln -s** command.

```
$ ln -s /etc/hostname name
$ ls -l
total 12
-rw-rw-r--. 1 fedora fedora  13 Jun 23 11:14 hello.txt
lrwxrwxrwx. 1 fedora fedora  13 Jun 23 12:32 name -> /etc/hostname
$ cat name
kushal-test.novalocal
```

In the above example, we created a soft link called *name* to the */etc/hostname* file. You can see details about the soft link files by using the **ls -l** command. You can create links to any directory in the same way.

If you remove the original file the soft link is pointing to, then the soft link will become useless, because it'll point to a file that does not exist. Soft links can also point to file which is in a different file system.

3.2 Creating hard links

```
$ echo "Hello World!" > hello.txt
$ ln hello.txt bye.txt
$ ls -l
total 16
-rw-rw-r--. 2 fedora fedora  13 Jun 23 11:14 bye.txt
-rw-rw-r--. 2 fedora fedora  13 Jun 23 11:14 hello.txt
lrwxrwxrwx. 1 fedora fedora  13 Jun 23 12:32 name -> /etc/hostname
```

(continues on next page)

(continued from previous page)

```
$ cat hello.txt
Hello World!
$ cat bye.txt
Hello World!
$ echo "1234" > hello.txt
$ cat bye.txt
1234
$ cat hello.txt
1234
$ rm hello.txt
$ cat bye.txt
1234
$ ls -l
total 12
-rw-rw-r--. 1 fedora fedora    5 Jun 23 12:39 bye.txt
lrwxrwxrwx. 1 fedora fedora   13 Jun 23 12:32 name -> /etc/hostname
```

If you look carefully, at the above example, we've created a hard link using the **ln** command. When we made a change to the original *hello.txt* file, that is also reflected in the *bye.txt* file.

But, because *bye.txt* is a hard link, even if I delete the *hello.txt*, the hard link still exists, and also has the original content.

3.3 Extracting a tar file

tar is a tool to create and extract archive files. Many times we will have to download and then extract tar files in our regular day to day work.

```
$ tar -xzvf files.tar.gz
hello.c
bye.txt
```

files.tar.gz file is compressed with gzip, if the file name ends with *.tar.bz2*, then it is compressed with bzip2.

```
$ tar -xjvf files.tar.bz2
hello.c
bye.txt
```

3.4 Creating a tar file

We can use the same **tar** command to create a tar file.

```
$ tar -czvf files.tar.gz hello.c bye.txt
hello.c
bye.txt
$ ls
bye.txt  files.tar.gz  hello.c
```


3.9 :w to save a file

In command mode, typing `:w` saves a file. If you want to save and quit the editor, then type either `:wq` or `:x`.

3.10 :q! to quit without saving

Typing `:q!`, when you are in command mode, will allow us quit without saving the current file.

Vim is a powerful editor, and we learned only a few, really basic steps in it. It will take a complete book, to explain different features of vim. But, the steps above are sufficient for our book's scope.

One major thing to remember about any text file, is keeping the newline character as the last line of the file. Because that is how the [POSIX standard](#) defines a line.

3.11 Becoming root user

root is the superuser. *root* has the power to make changes in various parts of a Linux system. That also means if you make any dangerous change (say deleting your user account) as *root* (by mistake), that can easily cause real damage.

The general rule is, when you need superuser power, use the **sudo** command to get work done, and use your normal user account for everything else. The **su -** command will help you become the *root* user; use this *extremely* carefully.

```
$ su -
Password:
#
```

Notice how the command prompt changed to `#` from `$`, `#` shows that you are using the *root* — another visible indication to think about every command you give as *root*. Press `Ctrl+d` to log out of the *root* account. (Or any account, for that matter.)

3.12 Using sudo command

Add the **sudo** command in front of any other command to execute them as *root*. For example:

```
$ less /var/log/secure
/var/log/secure: Permission denied
$ sudo less /var/log/secure
[sudo] password for fedora:
... long output
```

3.13 !! trick

There are times when you forgot to type *sudo* in the front of the command, you can use `!!` along with *sudo* to type that in faster.

```
$ less /var/log/secure
/var/log/secure: Permission denied
$ sudo !!
```

(continues on next page)

(continued from previous page)

```
[sudo] password for fedora:  
... long output
```

To know more about the ! based bash tricks, read [this blog post from Red Hat](#).

3.14 Environment variables

Environment variables are a way to pass data on to applications. We can set values of different variables, which any application can then access. There are various variables which decide how the shell will behave. To see all the variables, use the **printenv** command.

```
$ printenv  
... long output
```

You can execute the same command once as normal user, and once as *root*, and then check for the differences between the output. You will mostly see they are same, with some (or more) unique ones. That's because, variables are user specific.

3.15 Setting up environment variable values

We can use the **export** command to create a new environment variable or change an existing one. We use the **echo** command to print a particular environment variable's value.

```
$ export NAME="Kushal Das"  
$ echo $NAME  
Kushal Das  
$ export NAME="Babai Das"  
$ echo $NAME  
Babai Das
```

In our example we first created a new variable called *name*, and then we changed the value of the variable.

3.16 locate command

locate is a very useful tool to find files in the system. It's part of the **mlocate** package. For example, the following command will search all the files with *firewalld* in the name.

```
$ locate firewalld  
/etc/firewalld  
/etc/sysconfig/firewalld  
/etc/systemd/system/basic.target.wants/firewalld.service  
/home/kdas/.local/share/Zeal/Zeal/docsets/Ansible.docset/Contents/Resources/Documents/  
↪ docs.ansible.com/ansible/firewalld_module.html  
/home/kdas/Downloads/ansible-devel/lib/ansible/modules/system/firewalld.py  
/home/kdas/Downloads/ansible-fail-on-github-zipfile/lib/ansible/modules/system/  
↪ firewalld.py  
/home/kdas/code/git/ansible/lib/ansible/modules/system/firewalld.py  
... long output
```

You can update the search database by using **updatedb** command as root.

```
$ sudo updatedb
```

This may take some time as it will index all the files in your computer.

3.17 Finding date/time in different timezones

The `/usr/share/zoneinfo` directory contains all the different timezone files. We can use these file names to get current date/time in any timezone. For example, the following command will show the current date/time in *US/Pacific* timezone.

```
$ TZ=US/Pacific date
Sun May 20 18:45:54 PDT 2018
```

3.18 Bash history

Using **history** command you can check for any command you previously used in the shell, this output will not show you the commands from the current running shells. Only after you exit your shell, those commands will be written into `~/.bash_history` file, and history command tells us the details from there.

The environment variable **HISTFILESIZE** determines the number of commands stored in the file. By default, the history command does not show timestamps. You can have another environment variable to set the timestamp of every command. All commands from before setting the timestamp will show the same time for execution.

```
echo 'export HISTTIMEFORMAT="%d/%m/%y %T "' >> ~/.bashrc
source ~/.bashrc
...
...
history
```

3.19 Sort files by size

You can use **-S** or **--sort=size** option to the **ls** command.

```
ls -lSh
total 176K
-rw-r--r-- 1 kdas kdas 14K Aug 27 2018 networking.rst
-rw-r--r-- 1 kdas kdas 13K May 21 2018 services.rst
-rw-r--r-- 1 kdas kdas 13K Aug 30 2019 startingcommands.rst
-rw-r--r-- 1 kdas kdas 13K Jan 27 2019 processes.rst
-rw-r--r-- 1 kdas kdas 12K Sep 20 21:35 firewall.rst
...
...
```

You can reverse the sorting with passing **-r** option.

In this chapter we'll learn about user and group management on your system, and also about basic access control.

In Linux everything is associated to an user and a group. Based on these values, the system figures out, who can access what part of the system. That includes files, directories, network ports etc.

4.1 Finding the owner of file

We use the `ls -l` command to find the owner, and group of a file or directory.

```
[fedora@kushal-test ~]$ ls -l
total 12
-rw-rw-r--. 1 fedora fedora  13 Jun 23 11:14 hello.txt
-rw-rw-r--. 1 fedora fedora 5365 Jun  7 02:28 shellshare
[fedora@kushal-test ~]$ █
```

In the above example, fedora is the name of the owner and group both. The first value talks about who can access this file (we will learn about this in a while.)

4.2 /etc/passwd file

`/etc/passwd` contains all the users available in the system. This is a plain text file (this means you can view the information by using `cat` command.)

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

(continues on next page)

(continued from previous page)

```

daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-timesync:x:999:998:systemd Time Synchronization:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/ssh:/sbin/nologin
chrony:x:998:995:/:/var/lib/chrony:/sbin/nologin
systemd-coredump:x:994:994:systemd Core Dumper:/:/sbin/nologin
fedora:x:1000:1000:Fedora:/home/fedora:/bin/bash
polkitd:x:993:993:User for polkitd:/:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/
->null:/sbin/nologin

```

Each line has seven entries separated by `:`.

```
username:password:uid:gid:gecos:/home/dirname:shell
```

FIELD	MEANING
username	the username
password	the password of the user
uid	Numeric user id
gid	Numeric group id of user
gecos	arbitrary field
/home/dirname	Home directory of the user
shell	Which shell to use for the user

You'll see accounts with `/sbin/nologin` as their shell. These are generally accounts for various services, which are not supposed to be used by a normal human user; (which is why, no shell is needed.)

The actual user passwords are stored in an encrypted form in `/etc/shadow` file, with only the root user having access to this file.

```

$ ls -l /etc/shadow
-----. 1 root root 2213 Jun 22 15:20 /etc/shadow

```

If you want to know more about the current user, use the `id` command.

```
$ id
uid=1000(vagrant) gid=1000(vagrant) groups=1000(vagrant) context=unconfined_
↪u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

4.3 Details about groups

Group details are stored inside the `/etc/group` file. Each user has one primary group, and zero or more supplementary groups.

4.4 wheel group

If your user is part of the `wheel` group, then it has `sudo` access. If you remember the Fedora Installer, it actually gives you the option to mark a new user to be part of the wheel group during installation.

4.5 Becoming superuser

Have you noticed the silent command **sudo** in front of many commands in the lab before? We use that **sudo** command to become `root` user temporarily. The `root` user is also known as the superuser of the system, it has all the access power to change anything on the system. It is the administrator account of any Linux system.

Try the following command.

```
$ sudo id
```

Now, you will find the `id*` command worked as root instead of your regular user.

If you want to become `root` user for more than one command, then use the following command, and provide the `root` password to the input.

```
$ su -
```

Important: To be able to use **sudo** command, you must have your user mentioned in the `/etc/sudoers` file. The best way to edit the file is to use **visudo** command as root user.

Important: Read the man pages of `su` and `sudo` command.

4.6 Adding a new user

The **useradd** command adds a new user to the system. As you can well guess, this command has to execute as root, otherwise anyone can add random user accounts in the system. The following command adds a new user `babai` to the system.

```
$ sudo useradd babai
```

In Fedora, the initial user you create gets the uid 1000.

4.7 Changing user passwords

The **passwd** command helps to change any user password.

```
$ sudo passwd babai
Changing password for user babai.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

4.8 Modifying existing user details

The **usermod** command can help to modify an existing user. You can use the same command to lock user account in the system.

```
$ sudo usermod -L babai
$ su - babai
Password:
su: Authentication failure
$ sudo usermod -U babai
```

The last command in the above example unlocks the user account.

4.9 Deleting a user

We use the **userdel** command to delete a user from the system.

4.10 Adding a new group

The **groupadd** command adds a new group. You can also pass the group id as an option. In the following example we are adding a new group called *firejumpers*.

```
$ sudo groupadd -g 4001 firejumpers
```

4.11 Adding new group to an user

We can use **usermod** command to add any extra group to any of our system user. In the following example, we are adding *firejumpers* group to our *vagrant* user.

```
$ sudo usermod -aG firejumpers vagrant
```

Important: It is important to use **-a** flag to the *usermod* command. Without the **-a** flag *usermod* command will delete all the existing groups of the user. With *usermod -a* we append the user to the supplemental groups. And **-G** flag specifies the new list of supplementary GROUPS. Therefore with *usermod -aG* we append the new list of supplementary groups to the user's existing group/groups.

File permissions

Linux follows long Unix history, and has the same kinds of permission and ownership of files and directories. In this chapter, we will learn in detail about the same.

Let us look at the output of `ls -l` command.

```
$ ls -l
total 24
drwxrwxr-x. 2 fedora fedora 4096 Jun 24 08:00 dir1
-rw-rw-r--. 1 fedora fedora  174 Jun 23 13:26 files.tar.bz2
-rw-rw-r--. 1 fedora fedora  164 Jun 23 13:20 files.tar.gz
-rw-rw-r--. 1 fedora fedora   19 Jun 23 14:14 hello.txt
lrwxrwxrwx. 1 fedora fedora   13 Jun 23 12:32 name -> /etc/hostname
```

The first column contains the permission details of each file and directory. The permissions are displayed using groups of three values, *r* for read access, *w* for write access, and *x* for execute access. These 3 values are mentioned for owner, group, and other user accounts. The first - can be *d* for directories or *l* for links.

There's another way to calculate the same file permissions, using numbers.

Read	4
Write	2
Execute	1

This means, if you want to give read and write access only to the owner and group, you mention it like this “660”, where the first digit is for the owner, second digit is for the group, and the third digit is for the other users. We can use this format along with the `chmod` command to change permissions of any file or directory.

5.1 chmod command

`chmod` is the command which changes the file mode bits. Through `chmod` command one can alter the access permissions (i.e to permissions to read, write and execute) to file system objects (i.e files and directories). If we look at the command closely `chmod` is the abbreviation of change mode. A few examples are given below.

```
$ echo "hello" > myfile.txt
$ cat myfile.txt
hello
$ ls -l myfile.txt
-rw-rw-r--. 1 fedora fedora 6 Jun 25 03:42 myfile.txt
$ chmod 000 myfile.txt
$ ls -l myfile.txt
-----. 1 fedora fedora 6 Jun 25 03:42 myfile.txt
$ cat myfile.txt
cat: myfile.txt: Permission denied
$ chmod 600 myfile.txt
$ ls -l myfile.txt
-rw-----. 1 fedora fedora 6 Jun 25 03:42 myfile.txt
$ cat myfile.txt
hello
```

In the first line, we created a new file called *myfile.txt* using the *echo* command (we redirected the output of *echo* into the file). Using the *chmod 000 myfile.txt* command, we removed the read/write permissions of the file, and as you can see in the next line, even the owner of the file cannot read it. Setting the mode to *600* brings back read/write capability to the owner of that particular file.

The executable permission bit is required for directory access, and also for any file you want to execute.

5.2 PATH variable

The *PATH* is a shell variable. When we type a command in the bash shell, it searches for the command in the directories mentioned in the succeeding/sequential order, in the *PATH* variable. We can see the current *PATH* value using the *echo* command.

```
$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/fedora/.local/bin:/home/
↪fedora/bin
```

The different directories are separated by *.*. To search a particular command the shell will search in the following sequential order -

- */usr/local/bin*
- */usr/bin*
- */usr/local/sbin*
- */usr/sbin*
- */home/fedora/.local/bin*
- */home/fedora/bin*

You can see the */home/fedora/bin* directory is mentioned in the path. This means if we have that directory, and an executable file is in there, we can use it as a normal command in our shell. We will see an example of this, later in the book.

5.3 ~/.bash_profile file

~/.bash_profile is the configuration file for bash for the users who are allowed to login (via GUI or via ssh). On Fedora systems this file also read configuration from the *~/.bashrc* file.

```
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi
```

One can set environment variables, update **\$PATH** or any other important variables, or commands to execute after login using this file. But, remember to relogin or source the file (*source ~/.bash_profile*) after making the change.

5.4 .bashrc file

The *~/.bashrc* is a special configuration file for your bash terminal used for the users who can not login via the standard methods. These accounts will have *nologin* marked in the */etc/passwd* file. For example:

```
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
```

You can define or delete or update environment variables and add commands to execute when a new interactive shell opens up for the users who can not login.

For example, if want to add a new directory path to the **PATH** variable, then we can add the following line at the end of the *~/.bashrc* file.

```
export PATH=/mnt/myproject/bin:$PATH
```

After modifying the *.bashrc* file you will have to source it, or open a new tab in your terminal to see the change.

Important: To know more, read the man page of *bash* command.

5.5 /etc/profile file

This file is used to configure whenever a new login shell is created. This configures system wide, means if you add any variable here, that will be available for all users who can login to the system.

5.6 which command

We use the *which* command, to find the exact path of the executable being used by a command in our shell.

```
$ which chmod
/usr/bin/chmod
$ which tree
/usr/bin/which: no tree in (/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/
↪fedora/.local/bin:/home/fedora/bin)
```

The second example shows the output in case the *which* command cannot find the executable mentioned.

A process is a program (think about any Linux application) in a running state. It contains various details, like the memory space the program needs, a process id, the files opened by the process, etc.

6.1 How to view all running processes?

The following command shows all the processes from your computer.

```
$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0 215356 4984 ?        Ss   May29    0:28 /usr/lib/systemd/
↳systemd --system --deserialize 19
root         2  0.0  0.0     0     0 ?        S    May29    0:00 [kthreadd]
root         4  0.0  0.0     0     0 ?        S<   May29    0:00 [kworker/0:0H]
root         6  0.0  0.0     0     0 ?        S    May29    0:11 [ksoftirqd/0]
root         7  0.0  0.0     0     0 ?        S    May29    8:27 [rcu_sched]
... long output
```

You can see that the output also tells you under which user the process is running, what the actual command being used is, and the percentage of CPU and memory usage.

The *PID* column shows the process id; you can see that the *systemd* process has PID 1, which means it is the first process to start in the system.

6.2 How to find a particular process?

Let's say, I want to know the process id of the Firefox browser in my system. I can use the following command to find that information.

```
$ ps aux | grep firefox
kdas      26752 96.1  9.7 2770724 763436 ?        Sl   16:16   0:35 /usr/lib64/firefox/
↳firefox
kdas      26919  0.0  0.0 118520    980 pts/3    S+   16:17   0:00 grep --color=auto_
↳firefox
```

Here, we are first running the `ps` command, and then passing the output of that to the next command using the `|` character. In this case, as you see, `grep` is that second command. We can find and look for text using the `grep` tool. We will learn more about `grep` in the future.

6.3 How to kill/stop a particular process?

We can kill/stop any process using the `kill` command. We found out, in the last example, that the id of the Firefox process in my computer is `26752`, we can use that id to kill it.

```
$ kill 26752
```

If there is no error message, you'll find that Firefox has disappeared.

6.4 Finding out list of open files

`lsdf` command will show list of all open files. The man page has more details about the different command line options available.

6.5 Signals

Signals are a limited way to communicate to a process. You can think about them as notifications to a process, and depending on the signal handler in the code, the process does something with that signal. The `kill` command actually sends a signal to the given process id, the default signal is `TERM`, which says to terminate the process. To directly/forcibly kill a process, you can send the `KILL` signal.

```
$ kill -9 26752
```

Here `9` is number representation of the `KILL` signal. To know more about Linux signals, read the man page.

```
$ man 7 signal
```

`kill` command also has a `-l` flag, which prints all of the signal names, and numbers on the screen.

```
$ kill -l
1) SIGHUP      2) SIGINT      3) SIGQUIT     4) SIGILL      5) SIGTRAP
6) SIGABRT    7) SIGBUS     8) SIGFPE     9) SIGKILL    10) SIGUSR1
11) SIGSEGV   12) SIGUSR2   13) SIGPIPE   14) SIGALRM   15) SIGTERM
16) SIGSTKFLT 17) SIGCHLD  18) SIGCONT   19) SIGSTOP   20) SIGTSTP
21) SIGTTIN   22) SIGTTOU   23) SIGURG    24) SIGXCPU   25) SIGXFSZ
26) SIGVTALRM 27) SIGPROF  28) SIGWINCH  29) SIGIO     30) SIGPWR
31) SIGSYS    34) SIGRTMIN  35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3
38) SIGRTMIN+4          39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8
43) SIGRTMIN+9          44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14         49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
```

(continues on next page)

(continued from previous page)

```

53) SIGRTMAX-11      54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6      59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1      64) SIGRTMAX

```

6.6 top command

top is a very useful command while using a Linux system. It's a quick way to know about all the running processes in the system, and their related status about CPU and memory usage in general. To get out of **top**, press the key *q*.

```

top - 17:37:28 up 24 days, 11:52,  2 users,  load average: 0.57, 0.73, 0.75
Tasks: 372 total,  2 running, 370 sleeping,  0 stopped,  0 zombie
%Cpu(s): 11.6 us,  2.6 sy,  0.0 ni, 84.9 id,  0.1 wa,  0.3 hi,  0.5 si,  0.0 st
KiB Mem : 7858752 total, 1701052 free, 4444136 used, 1713564 buff/cache
KiB Swap: 3268604 total, 1558396 free, 1710208 used. 2431656 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
28300 kdas      20   0 1502016 287340 44396 R  25.0   3.7 290:56.60 chrome
2668  kdas       9  -11 2067292   9756   7164 S   6.2   0.1 166:06.48 pulseaudio
15122 kdas      20   0  771844  33104 11352 S   6.2   0.4  39:24.60 gnome-terminal-
24760 kdas      20   0 1945840 209128 76952 S   6.2   2.7   1:41.15 code
27526 kdas      20   0  156076   4268   3516 R   6.2   0.1   0:00.01 top
   1 root      20   0  215356   4880   3108 S   0.0   0.1   0:28.25 systemd
   2 root      20   0     0     0     0 S   0.0   0.0   0:00.66 kthreadd
   4 root       0  -20     0     0     0 S   0.0   0.0   0:00.00 kworker/0:0H
   6 root      20   0     0     0     0 S   0.0   0.0   0:11.79 ksoftirqd/0
   7 root      20   0     0     0     0 S   0.0   0.0   8:28.06 rcu_sched
... long output

```

By the way, feel free to press *l* and see if anything changes in the **top** command output.

6.7 Load average

If you look at the **top** output carefully, you will find load average mentioned. Actually, there are 3 numbers provided; these are the load averages of the system in the last one minute, 5 minutes ago, and 15 minutes ago.

```
load average: 0.57, 0.73, 0.75
```

In simple words, load average means the average time any process has to wait to get access to the CPU (or other resources), in idle state the load average is 0. This information is a quick way to learn about the system, if the system is slow to respond, just looking at the load-average, and then the rest of the **top** output should be a good starting point.

6.8 htop tool

htop is a modern version of the **top** tool. It has many more features, interactiveness being the biggest amongst them. **htop** does not come by default in most of the Linux installations, which means you will have to install it using the system's package management tool.

These are the ways to install it in Fedora and in Debian/Ubuntu

```
$ sudo dnf install htop -y
```

```
$ sudo apt-get install htop
```

The screenshot shows the htop terminal window on a system named 'babai@kdas-laptop'. The window title is 'babai@kdas-laptop:~'. The menu bar includes File, Edit, View, Search, Terminal, and Help. The top section displays system statistics: CPU usage (1.3%), memory usage (2.03G/15.5G), swap usage (204M/7.80G), tasks (149, 393 thr; 1 running), load average (0.98 1.31 1.45), and uptime (1 day, 04:45:42). Below the statistics is a table of running processes. The table has columns for PID, USER, PRI, NI, VIRT, RES, SHR, S, CPU%, MEM%, TIME+, and Command. The processes listed include htop (PID 9988), gnome-sh (PID 8997), trac (PID 6982), gnom (PID 9893), systemd (PID 1), auditd (PID 982, 981), audispd (PID 986, 983), sedispa (PID 985), and switch (PID 1030). The bottom of the window shows function key shortcuts: F1 Help, F2 Setup, F3 Search, F4 Filter, F5 Tree, F6 SortBy, F7 Nice, F8 Nice -, F9 Kill, and F10 Quit.

```
babai@kdas-laptop:~
File Edit View Search Terminal Help

 1 [ | | 1.3%] 5 [ | 0.7%]
 2 [ | 0.0%] 6 [ | 0.7%]
 3 [ | 0.7%] 7 [ | 0.7%]
 4 [ | 0.0%] 8 [ | 0.0%]
Mem [ | | | | | | 2.03G/15.5G] Tasks: 149, 393 thr; 1 running
Swp [ | 204M/7.80G] Load average: 0.98 1.31 1.45
Uptime: 1 day, 04:45:42

 PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
 9988 babai 20 0 124M 3504 2784 R 1.3 0.0 0:00.09 htop
 8997 babai 20 0 2397M 185M 76720 S 0.7 1.2 0:25.26 /usr/bin/gnome-sh
 6982 babai 20 0 595M 21936 7132 S 0.7 0.1 0:00.06 /usr/libexec/trac
 9893 babai 20 0 690M 37232 28512 S 0.7 0.2 0:00.39 /usr/libexec/gnom
 1 root 20 0 208M 4792 3076 S 0.0 0.0 0:02.73 /usr/lib/systemd/
 610 root 20 0 162M 35812 35240 S 0.0 0.2 0:05.98 /usr/lib/systemd/
 639 root 20 0 45700 980 596 S 0.0 0.0 0:00.39 /usr/lib/systemd/
 982 root 16 -4 55588 116 0 S 0.0 0.0 0:00.00 /sbin/auditd
 981 root 16 -4 55588 116 0 S 0.0 0.0 0:00.09 /sbin/auditd
 986 root 12 -8 84556 60 0 S 0.0 0.0 0:00.13 /sbin/audispd
 983 root 12 -8 84556 60 0 S 0.0 0.0 0:00.15 /sbin/audispd
 985 root 16 -4 43840 284 204 S 0.0 0.0 0:00.04 /usr/sbin/sedispa
 1030 root 20 0 254M 1140 1024 S 0.0 0.0 0:00.00 /usr/sbin/switche
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice - F8Nice + F9Kill F10Quit
```

To know more about htop, please read the man page.

```
$ man htop
```

6.9 More about Linux processes

You can learn more about Linux processes in the glibc manual. Use the *info* command to find out more.

```
$ info libc process
```

6.10 /proc directory

/proc is a special directory in our filesystem. This is a virtual filesystem which contains information about all the running processes, and information about the hardware present in the system. You will find that the files in the virtual filesystem are 0 in size.

Now we'll learn about a few files inside this directory.

6.11 /proc/cpuinfo

/proc/cpuinfo file has information about the CPU in your system. It includes the model number, and also the various flags available in that particular CPU model.

6.12 /proc/cmdline

/proc/cmdline file has all the parameters passed to the kernel at the bootup time. The following is a cloud-based virtual machine.

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-4.8.6-300.fc25.x86_64 root=UUID=9be70055-35f2-4a57-b120-
↪5a003dfdb504 ro no_timer_check console=tty1 console=ttyS0,115200n8 rhgb quiet_
↪console=ttyS1 LANG=en_US.UTF-8 initrd=/boot/initramfs-4.8.6-300.fc25.x86_64.img
```

6.13 /proc/meminfo

/proc/meminfo contains information related to the memory in the system. You can see the total amount RAM, the available memory and other values there.

```
$ cat /proc/meminfo
MemTotal:      4046820 kB
MemFree:       2960568 kB
MemAvailable:  3696216 kB
Buffers:       53756 kB
Cached:        830052 kB
SwapCached:    0 kB
Active:        347216 kB
Inactive:      575692 kB
Active(anon):  39388 kB
Inactive(anon): 196 kB
Active(file):  307828 kB
Inactive(file): 575496 kB
Unevictable:   0 kB
Mlocked:       0 kB
SwapTotal:     0 kB
SwapFree:      0 kB
Dirty:         4 kB
Writeback:     0 kB
AnonPages:     39120 kB
Mapped:        42032 kB
Shmem:         488 kB
Slab:          141692 kB
SReclaimable: 114996 kB
SUnreclaim:   26696 kB
KernelStack:  1360 kB
PageTables:    2700 kB
NFS_Unstable:  0 kB
Bounce:        0 kB
WritebackTmp:  0 kB
CommitLimit:  2023408 kB
Committed_AS: 127752 kB
```

(continues on next page)

(continued from previous page)

```
VmallocTotal: 34359738367 kB
VmallocUsed: 0 kB
VmallocChunk: 0 kB
HardwareCorrupted: 0 kB
AnonHugePages: 0 kB
ShmemHugePages: 0 kB
ShmemPmdMapped: 0 kB
CmaTotal: 0 kB
CmaFree: 0 kB
HugePages_Total: 0
HugePages_Free: 0
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize: 2048 kB
DirectMap4k: 61296 kB
DirectMap2M: 4132864 kB
DirectMap1G: 2097152 kB
```

6.14 /proc/uptime

```
$ cat /proc/uptime
52820.32 104802.84
```

The first value in this file shows the number of seconds the system is up. The second value is the total number of idle seconds for each CPU, so for the modern systems, this value can be more than the first value.

6.15 /proc/sys/ & sysctl command

This directory is a special one for system administrators. This not only provides information, but also allows you to quickly change (enable/disable) different kernel features.

We use the **sysctl** command to view or edit the values for */proc/sys/*. If you want to see all the different settings, use the following command.

```
$ sudo sysctl -a
[sudo] password for kdas:
abi.vsyscall32 = 1
crypto.fips_enabled = 0
debug.exception-trace = 1
debug.kprobes-optimization = 1
dev.cdrom.autoclose = 1
dev.cdrom.autoeject = 0
dev.cdrom.check_media = 0
dev.cdrom.debug = 0
dev.cdrom.info = CD-ROM information, Id: cdrom.c 3.20 2003/12/17
... long output
```

6.16 Enabling IP forward with sysctl

To enable IP forwarding to the VM(s), use the following command.

```
$ sudo sysctl -w net.ipv4.ip_forward=1
```

To check the current value, use the following command.

```
$ sysctl net.ipv4.ip_forward
```

You can see the same value in the `/proc/sys/net/ipv4/ip_forward` file too.

```
$ cat /proc/sys/net/ipv4/ip_forward  
1
```

To make the change permanent, write the following in the `/etc/sysctl.conf` file.

```
net.ipv4.ip_forward = 1
```

Then, enable the changes using the following command.

```
$ sudo sysctl -p /etc/sysctl.conf
```


This is also a chapter related to the *systemd* tool.

7.1 What is a service?

A service is a process or application which is running in the background, either doing some predefined task or waiting for some event. If you remember our process chapter, we learned about *systemd* for the first time there. It is the first process to run in our system; it then starts all the required processes and services. To know about how the system boots up, read the **bootup** man page. [Click here](#) to read it online.

```
$ man bootup
```

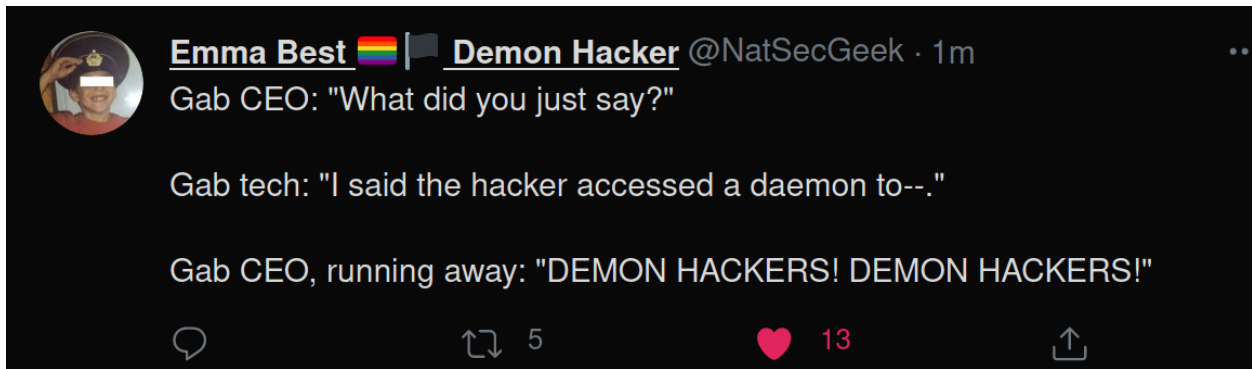
7.2 What is a daemon?

Daemon is the actual term for those long-running background processes. A service actually consists of one or more daemons.

Make sure that you don't get confused between Daemons and Demons :) Here is a gem from Internet:

7.3 What is the init system?

If you look at Unix/Linux history, you will find the first process which starts up, is also known as *init process*. This process used to start other processes by using the rc files from */etc/rc.d* directory. In the modern Linux systems, **systemd** has replaced the init system.



7.4 Units in systemd

Units are a standardized way for the `systemd` to manage various parts of a system. There are different kinds of units, `.service` is for system services, `.path` for path based ones. There is also `.socket` which are socket based `systemd` units. There are various other types, we can learn about those later.

7.5 `.service` units in `systemd`

These are service units, which explains how to manage a particular service in the system. In our daily life, we generally only have to work with these unit files.

7.6 How to find all the `systemd` units in the system?

```
$ systemctl
... long output
-.mount                                loaded active  ─
↔mounted /
boot.mount                              loaded active  ─
↔mounted /boot
dev-hugepages.mount                    loaded active  ─
↔mounted Huge Pages File System
dev-mqueue.mount                       loaded active  ─
↔mounted POSIX Message Queue File System
home.mount                              loaded active  ─
↔mounted /home
proc-fs-nfsd.mount                     loaded active  ─
↔mounted NFSD configuration filesystem
run-user-1000-doc.mount                 loaded active  ─
↔mounted /run/user/1000/doc
run-user-1000-gvfs.mount                loaded active  ─
↔mounted /run/user/1000/gvfs
run-user-1000.mount                     loaded active  ─
↔mounted /run/user/1000
run-user-42.mount                       loaded active  ─
... long output
```

In the output of the `systemctl` command, you should be able to see all the different kinds of units in the system. If you want to see only the service units, then use the following command.

```
$ systemctl --type=service
```

7.7 Working with a particular service

Let us take the *sshd.service* as an example. The service controls the sshd daemon, which allows us to remotely login to a system using the **ssh** command.

To know the current status of the service, I execute the following command.

```
$ sudo systemctl status sshd
sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled; vendor preset:
↳enabled)
  Active: inactive (dead)
    Docs: man:sshd(8)
          man:sshd_config(5)

Jun 19 12:07:29 kdas-laptop sshd[19533]: Accepted password for kdas from 192.168.1.
↳101 port 61361 ssh2
Jun 20 17:57:53 kdas-laptop sshd[30291]: Connection closed by 192.168.1.101 port
↳63345 [preauth]
Jun 20 17:58:02 kdas-laptop sshd[30293]: Accepted password for kdas from 192.168.1.
↳101 port 63351 ssh2
Jun 20 18:32:11 kdas-laptop sshd[31990]: Connection closed by 192.168.1.101 port
↳64352 [preauth]
Jun 20 18:32:17 kdas-laptop sshd[32039]: Accepted password for kdas from 192.168.1.
↳101 port 64355 ssh2
Jun 20 18:45:57 kdas-laptop sshd[32700]: Accepted password for kdas from 192.168.1.
↳101 port 64824 ssh2
Jun 21 08:44:39 kdas-laptop sshd[15733]: Accepted password for kdas from 192.168.1.
↳101 port 51574 ssh2
Jun 22 18:17:24 kdas-laptop systemd[1]: Stopping OpenSSH server daemon...
Jun 22 18:17:24 kdas-laptop sshd[20932]: Received signal 15; terminating.
Jun 22 18:17:24 kdas-laptop systemd[1]: Stopped OpenSSH server daemon.
```

To start the service, I'll use the following command, and then I can use the *status* argument to the **systemctl** to check the service status once again.

```
$ sudo systemctl start sshd
$ sudo systemctl status sshd
sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled; vendor preset:
↳enabled)
  Active: active (running) since Thu 2017-06-22 18:19:28 IST; 1s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
Main PID: 3673 (sshd)
  Tasks: 1 (limit: 4915)
  CGroup: /system.slice/sshd.service
          └─3673 /usr/sbin/sshd -D

Jun 22 18:19:28 kdas-laptop systemd[1]: Starting OpenSSH server daemon...
Jun 22 18:19:28 kdas-laptop sshd[3673]: Server listening on 0.0.0.0 port 22.
Jun 22 18:19:28 kdas-laptop sshd[3673]: Server listening on :: port 22.
Jun 22 18:19:28 kdas-laptop systemd[1]: Started OpenSSH server daemon.
```

In the same way, we can use either the *stop* or *restart* arguments to the **systemctl** command.

7.8 Enabling or disabling a service

Even if you start a service, you'll find that after you reboot the computer, the service did not start at the time of boot up. To do so, you will have to enable the service, or to stop a service from starting at boot, you will have to disable the service.

```
$ sudo systemctl enable sshd.service
Created symlink /etc/systemd/system/multi-user.target.wants/sshd.service → /usr/lib/
↳systemd/system/sshd.service.
$ sudo systemctl disable sshd.service
Removed /etc/systemd/system/multi-user.target.wants/sshd.service.
```

7.9 Shutdown or reboot the system using systemctl

We can also reboot or shutdown the system using the **systemctl** command.

```
$ sudo systemctl reboot
$ sudo systemctl shutdown
```

7.10 journalctl

systemd runs the **systemd-journald.service**, which stores logs in the journal from the different services maintained by *systemd*. We use *journalctl* command to read these log entries from the journal. If you execute the command without any arguments, it will show you all the log entries starting from the oldest in the journal. One needs to be *root* to be able to use the *journalctl* command. Remember that *systemd-journald* stores all the logs in binary format, means you can not just *less* the files and read them.

If you want any normal user to execute *journalctl* command, then add them into **systemd-journal** group.

7.11 Finding the logs of a service

We can use the **journalctl** command to find the log of a given service. The general format is *journalctl -u service-name*". Like below is the log for **sshd* service.

```
$ sudo journalctl -u sshd
-- Logs begin at Thu 2017-06-22 14:16:45 UTC, end at Fri 2017-06-23 05:21:29 UTC. --
Jun 22 14:17:39 kushal-test.novalocal systemd[1]: Starting OpenSSH server daemon...
Jun 22 14:17:39 kushal-test.novalocal systemd[1]: sshd.service: PID file /var/run/
↳sshd.pid not readable (yet?) after start: No such file or directory
Jun 22 14:17:39 kushal-test.novalocal sshd[827]: Server listening on 0.0.0.0 port 22.
Jun 22 14:17:39 kushal-test.novalocal sshd[827]: Server listening on :: port 22.
Jun 22 14:17:39 kushal-test.novalocal systemd[1]: Started OpenSSH server daemon.
Jun 22 14:22:08 kushal-test.novalocal sshd[863]: Accepted publickey for fedora from
↳103.249.881.17 port 56124 ssh2: RSA SHA256:lvn4rIzmfB14PBQwh4k9C
Jun 22 14:29:24 kushal-test.novalocal systemd[1]: Stopping OpenSSH server daemon...
Jun 22 14:29:24 kushal-test.novalocal sshd[827]: Received signal 15; terminating.
```

(continues on next page)

(continued from previous page)

```

Jun 22 14:29:24 kushal-test.novalocal systemd[1]: Stopped OpenSSH server daemon.
Jun 22 14:29:24 kushal-test.novalocal systemd[1]: Starting OpenSSH server daemon...
Jun 22 14:29:24 kushal-test.novalocal sshd[2164]: Server listening on 0.0.0.0 port 22.
Jun 22 14:29:24 kushal-test.novalocal sshd[2164]: Server listening on :: port 22.
Jun 22 14:29:24 kushal-test.novalocal systemd[1]: Started OpenSSH server daemon.
Jun 22 14:54:26 kushal-test.novalocal sshd[13522]: Invalid user  from 139.162.122.110_
↳port 51012
Jun 22 14:54:26 kushal-test.novalocal sshd[13522]: input_userauth_request: invalid_
↳user [preauth]
Jun 22 14:54:26 kushal-test.novalocal sshd[13522]: Failed none  for invalid user  from_
↳139.162.122.110 port 51012 ssh2
Jun 22 14:54:26 kushal-test.novalocal sshd[13522]: Connection closed by 139.162.122.
↳110 port 51012 [preauth]
Jun 22 15:15:29 kushal-test.novalocal sshd[13541]: Did not receive identification_
↳string from 5.153.62.226 port 48677

```

7.12 To view only the last N entries

You can use the `-n` argument to the `journalctl` command to view only the last N number of entries. For example, to view the last 10 entries.

```
# journalctl -n 10
```

7.13 Continuous stream of logs

In case you want to monitor the logs of any service, that is keep reading the logs in real time, you can use `-f` flag with the `journalctl` command.

```

$ sudo journalctl -f -u sshd
-- Logs begin at Thu 2017-06-22 14:16:45 UTC. --
Jun 23 03:39:09 kushal-test.novalocal sshd[14095]: Did not receive identification_
↳string from 158.85.81.118 port 10000
Jun 23 04:13:32 kushal-test.novalocal sshd[14109]: Received disconnect from 221.194.
↳47.242 port 55028:11: [preauth]
Jun 23 04:13:32 kushal-test.novalocal sshd[14109]: Disconnected from 221.194.47.242_
↳port 55028 [preauth]
Jun 23 04:33:59 kushal-test.novalocal sshd[14115]: Received disconnect from 59.45.175.
↳64 port 36248:11: [preauth]
Jun 23 04:36:53 kushal-test.novalocal sshd[14121]: Did not receive identification_
↳string from 82.193.122.22 port 58769
Jun 23 04:42:01 kushal-test.novalocal sshd[14123]: Received disconnect from 221.194.
↳47.233 port 51797:11: [preauth]
Jun 23 04:42:01 kushal-test.novalocal sshd[14123]: Disconnected from 221.194.47.233_
↳port 51797 [preauth]
Jun 23 04:51:46 kushal-test.novalocal sshd[14130]: Did not receive identification_
↳string from 191.253.13.227 port 4668
Jun 23 05:05:16 kushal-test.novalocal sshd[14189]: Received disconnect from 59.45.175.
↳88 port 33737:11: [preauth]
Jun 23 05:05:16 kushal-test.novalocal sshd[14189]: Disconnected from 59.45.175.88_
↳port 33737 [preauth]

```

I can see that someone was trying to break into this VM by trying random ports :)

7.14 Listing of previous boots

In systems like Fedora, **journalctl** by default keeps history from past boots. To know about all available boot history, type the following command.

```
$ sudo journalctl --list-boots
[sudo] password for fedora:
-112 7a88e13a76434a1199f82ad90441ae7f Tue 2014-12-09 03:41:08 IST--Tue 2014-12-09
↳03:41:08 IST
-111 b86086ed59b84b228e74f91ab08a66b3 Sun 2015-06-28 23:54:26 IST--Sun 2015-07-12
↳07:27:48 IST
-110 71d3f6024f514653bfd2574243d096d1 Sun 2016-06-05 01:51:05 IST--Sun 2016-06-05
↳01:51:16 IST
-109 b7721878a5144d009418cf269b5eea71 Fri 2016-08-19 19:47:57 IST--Sat 2016-08-20
↳01:16:07 IST
-108 6102102fc7804379b888d83cea66838b Sat 2016-08-20 01:21:36 IST--Sun 2016-08-21
↳00:05:38 IST
... long output
```

To know about any particular boot log, you can use the hash along with **-b** flag to the **journalctl** command.

```
$ sudo journalctl -b 7a88e13a76434a1199f82ad90441ae7f
-- Logs begin at Tue 2014-12-09 03:41:08 IST, end at Sat 2017-06-24 13:40:49 IST. --
Dec 09 03:41:08 localhost.localdomain systemd[1344]: Stopping Default.
Dec 09 03:41:08 localhost.localdomain systemd[1344]: Stopped target Default.
Dec 09 03:41:08 localhost.localdomain systemd[1344]: Starting Shutdown.
Dec 09 03:41:08 localhost.localdomain systemd[1344]: Reached target Shutdown.
Dec 09 03:41:08 localhost.localdomain systemd[1344]: Starting Exit the Session..
```

7.15 Time-based log viewing

We can also use **journalctl** to view logs for a certain time period. For example, if we want to see all the logs since yesterday, we can use the following command.

```
$ sudo journalctl --since yesterday
[sudo] password for fedora:
-- Logs begin at Tue 2014-12-09 03:41:08 IST, end at Sat 2017-06-24 15:21:54 IST. --
Jun 23 00:00:00 kushal-test.novalocal /usr/libexec/gdm-x-session[28622]: (evolution-
↳alarm-notify:11609): evolution-alarm-notify-WARNING **: alarm.c:253: Reques
Jun 23 00:01:01 kushal-test.novalocal CROND[22327]: (root) CMD (run-parts /etc/cron.
↳hourly)
... long output
```

You can also use date time following **YYYY-MM-DD HH:MM:SS** format.

```
$ sudo journalctl --since "2015-11-10 14:00:00"
-- Logs begin at Tue 2014-12-09 03:41:08 IST, end at Sat 2017-06-24 15:25:30 IST. --
Jun 05 01:51:05 kushal-test.novalocal systemd[5674]: Reached target Timers.
Jun 05 01:51:05 kushal-test.novalocal systemd[5674]: Reached target Paths.
Jun 05 01:51:05 kushal-test.novalocal systemd[5674]: Starting D-Bus User Message Bus.
↳Socket.
Jun 05 01:51:05 kushal-test.novalocal systemd[5674]: Listening on D-Bus User Message.
↳Bus Socket.
Jun 05 01:51:05 kushal-test.novalocal systemd[5674]: Reached target Sockets.
```

(continues on next page)

(continued from previous page)

```
Jun 05 01:51:05 kushal-test.novalocal systemd[5674]: Reached target Basic System.  
Jun 05 01:51:05 kushal-test.novalocal systemd[5674]: Reached target Default.
```

7.16 Total size of the journal logs

Use the `-disk-usage` flag to find out total amount entries and archive can be stored. The following is the output from an Ubuntu Focal (20.04) system.

```
# journalctl --disk-usage  
Archived and active journals take up 56.0M in the file system.
```

Package management




In the Free and Open Source Software world, most software is released in source code format by developers. This means that generally, if you want to install a piece of software, you will find the source code on the website of the project. As a user, you will have to find and install all the other bits of software, that this particular piece depends on (the *dependencies*) and then install the software. To solve this *painful* issue, all Linux distributions have something called a *package management system*. Volunteers (mostly) all across the world help make binary software packages out of source code released by the developers, in such a way that users of the Linux distribution can easily install, update or remove that software.

It's generally recommended, we use the package management system that comes with the distribution, to install software for the users. If you are really sure about what you're doing in the system, you can install from the source files too; but that can be dangerous.

8.1 dnf command

dnf is the package management system in Fedora. The actual packages come in the *rpm* format. *dnf* helps you search, install or uninstall any package from the Fedora package repositories. You can also use the same command to update packages in your system.

8.2 Searching for a package

```
$ dnf search pss
Fedora 25 - x86_64                                     34 MB/s |  
↳50 MB      00:01
Fedora 25 - x86_64 - Updates                          41 MB/s |  
↳23 MB      00:00
Last metadata expiration check: 0:00:07 ago on Sun Jun 25 04:14:22 2017.
===== N/S Matched: pss 
↳=====
pss.noarch : A power-tool for searching inside source code files
pssh.noarch : Parallel SSH tools
```

First the tool, downloads all the latest package information from the repository, and then gives us the result.

8.3 Finding more information about a package

dnf info gives us more information about any given package.

```
$ dnf info pss
Last metadata expiration check: 0:04:59 ago on Sun Jun 25 04:14:22 2017.
Available Packages
Name       : pss
Arch      : noarch
Epoch    : 0
Version   : 1.40
Release   : 6.fc25
Size      : 58 k
Repo      : fedora
Summary   : A power-tool for searching inside source code files
URL       : https://github.com/eliben/pss
License   : Public Domain
Description: pss is a power-tool for searching inside source code files.
           : pss searches recursively within a directory tree, knows which
           : extensions and file names to search and which to ignore, automatically
           : skips directories you wouldn't want to search in (for example .svn or .
           ↪git),
           : colors its output in a helpful way, and does much more.
```

8.4 Installing a package

The *dnf install* command helps us install any given package. We can pass more than one package name as the argument.

```
$ sudo dnf install pss wget
Last metadata expiration check: 0:37:13 ago on Sun Jun 25 03:44:07 2017.
Package wget-1.18-3.fc25.x86_64 is already installed, skipping.
Dependencies resolved.
=====
Package                Arch          Version      ↪
↪                      Repository   Size
=====
Installing:
pss                    noarch       1.40-6.fc25  ↪
↪                      fedora       58 k
Transaction Summary
=====
Install 1 Package

Total download size: 58 k
Installed size: 196 k
Is this ok [y/N]: y
Downloading Packages:
pss-1.40-6.fc25.noarch.rpm
↪                      969 kB/s | 58 kB    00:00
-----
↪-----
```

(continues on next page)

(continued from previous page)

```

Total
↳                               118 kB/s | 58 kB      00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Installing   : pss-1.40-6.fc25.noarch
↳
  Verifying    : pss-1.40-6.fc25.noarch
↳
↳
Installed:
  pss.noarch 1.40-6.fc25
Complete!

```

8.5 To list the available updates

The following command shows all the available updates for your system.

```

# dnf list updates

Last metadata expiration check: 0:52:28 ago on Fri 09 Apr 2021 08:51:39 PM IST.
Available Upgrades
fedora-gpg-keys.noarch           33-4          updates
fedora-repos.noarch             33-4          updates
fedora-repos-modular.noarch     33-4          updates

```

8.6 To list all security updates

`dnf` can also tell you about all the updates which are marked as security updates.

```

# dnf updateinfo list --security
Last metadata expiration check: 2:06:38 ago on Sun 25 Jul 2021 03:44:47 AM UTC.
FEDORA-2021-83fdddca0f Moderate/Sec.  curl-7.76.1-7.fc34.x86_64
FEDORA-2021-08cdb4dc34 Important/Sec.  dhcp-client-12:4.4.2-11.b1.fc34.x86_64
FEDORA-2021-08cdb4dc34 Important/Sec.  dhcp-common-12:4.4.2-11.b1.fc34.noarch
FEDORA-2021-e14e86e40e Moderate/Sec.  glibc-2.33-20.fc34.x86_64
FEDORA-2021-e14e86e40e Moderate/Sec.  glibc-common-2.33-20.fc34.x86_64
FEDORA-2021-e14e86e40e Moderate/Sec.  glibc-doc-2.33-20.fc34.noarch
FEDORA-2021-e14e86e40e Moderate/Sec.  glibc-langpack-en-2.33-20.fc34.x86_64
FEDORA-2021-07dc0b3eb1 Critical/Sec.  kernel-core-5.13.4-200.fc34.x86_64
FEDORA-2021-8b25e4642f Low/Sec.      krb5-libs-1.19.1-14.fc34.x86_64
FEDORA-2021-83fdddca0f Moderate/Sec.  libcurl-7.76.1-7.fc34.x86_64
FEDORA-2021-31fdc84207 Moderate/Sec.  libcrypt-1.9.3-3.fc34.x86_64
FEDORA-2021-2443b22fa0 Moderate/Sec.  linux-firmware-20210716-121.fc34.noarch
FEDORA-2021-2443b22fa0 Moderate/Sec.  linux-firmware-whence-20210716-121.fc34.noarch
FEDORA-2021-d1fc0b9d32 Moderate/Sec.  nettle-3.7.3-1.fc34.x86_64
FEDORA-2021-0ec5a8a74b Important/Sec.  polkit-libs-0.117-3.fc34.1.x86_64
FEDORA-2021-a6bde7ab18 Moderate/Sec.  python3-urllib3-1.25.10-5.fc34.noarch

```

8.7 Update the packages via dnf

Use **dnf update** command to install all the available updates. You can also pass the **-y** flag to it.

8.8 Find out the services & applications need restart after update in Fedora/CentOS/RHEL

The **dnf-utils** package contains a special command, **needs-restarting**. After you do a *dnf update*, when different libraries get updated, there may be running processes/services which needs restart. One way of doing this is restarting the system, but that may not be the right choice (may be you are running critical services) all the time. So, you can find out which ones you should restart.

Below is the output from a Fedora 34 desktop system.

```
# needs-restarting
1 : /usr/lib/systemd/systemd --system --deserialize 62
1616 : /usr/lib/systemd/systemd-resolved
1617 : /sbin/auditd
1638 : /usr/sbin/ModemManager
1639 : avahi-daemon: running [linux-2.local]
1640 : /usr/libexec/bluetooth/bluetoothd
1641 : /usr/libexec/bolttd
1642 : /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid
1643 : /usr/local/bin/ivpn-service
1646 : /usr/sbin/mcelog --daemon --foreground
1650 : /usr/sbin/rngd -f
1651 : /usr/libexec/rtkit-daemon
1657 : /usr/libexec/switcheroo-control
1659 : /usr/lib/systemd/systemd-machined
1662 : /usr/libexec/udisks2/udisksd
1664 : /usr/libexec/upowerd
1669 : avahi-daemon: chroot helper
1688 : /usr/bin/dbus-broker-launch --scope system --audit
1695 : /usr/sbin/abrt-d -d -s
1699 : /usr/sbin/chronyd
1723 : /usr/bin/abrt-dump-journal-core -D -T -f -e
1724 : /usr/bin/abrt-dump-journal-oops -fxtD
1725 : /usr/bin/abrt-dump-journal-xorg -fxtD
1728 : /usr/lib/polkit-1/polkitd --no-debug
1744 : /usr/libexec/accounts-daemon
1745 : /usr/lib/systemd/systemd-logind
1776 : /usr/sbin/NetworkManager --no-daemon
1788 : /usr/sbin/cupsd -l
1877 : /usr/sbin/pcscd --foreground --auto-exit
1898 : /usr/sbin/atd -f
1899 : /usr/sbin/crond -n
1900 : /usr/sbin/gdm
1920 : /usr/libexec/uresourced
2034 : /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --
↳ leasefile-ro --dhcp-script=/usr/libexec/libvirt_leaseshelper
2035 : /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --
↳ leasefile-ro --dhcp-script=/usr/libexec/libvirt_leaseshelper
2162 : /usr/sbin/wpa_supplicant -c /etc/wpa_supplicant/wpa_supplicant.conf -u -s
2164 : /usr/libexec/packagekitd
```

(continues on next page)

(continued from previous page)

```

2393 : /usr/libexec/colord
2714 : /usr/sbin/abrt-dbus -t133
2730 : gdm-session-worker [pam/gdm-password]
2750 : /usr/lib/systemd/systemd --user
2756 : (sd-pam)
2780 : /usr/libexec/gdm-x-session --run-script /usr/bin/gnome-session
2784 : /usr/libexec/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority -
↳nolisten tcp -background none -noreset -keeptty -novtswitch -verbose 3
2838 : /usr/bin/dbus-broker-launch --scope user
2843 : /usr/libexec/gnome-session-binary
2907 : /usr/libexec/at-spi-bus-launcher
2912 : /usr/bin/dbus-broker-launch --config-file=/usr/share/defaults/at-spi2/
↳accessibility.conf --scope user
2923 : /usr/libexec/gnome-session-ctl --monitor
2924 : /usr/libexec/uresourced --user
2926 : /usr/libexec/gnome-session-binary --systemd-service --session=gnome
2953 : /usr/bin/gnome-shell
2978 : ibus-daemon --panel disable --xim
2989 : /usr/libexec/ibus-extension-gtk3
3004 : /usr/libexec/ibus-x11 --kill-daemon
3015 : /usr/libexec/at-spi2-registryd --use-gnome-session
3023 : /usr/libexec/evolution-source-registry
3030 : /usr/libexec/goa-daemon
3031 : /usr/bin/pipewire
3032 : /usr/bin/pipewire-pulse
3038 : /usr/bin/pipewire-media-session
3044 : /usr/libexec/evolution-calendar-factory
3075 : /usr/libexec/gvfs-udisks2-volume-monitor
3092 : /usr/libexec/gvfs-mtp-volume-monitor
3116 : /usr/libexec/gvfs-gphoto2-volume-monitor
3130 : /usr/bin/gjs /usr/share/gnome-shell/org.gnome.Shell.Notifications
3147 : /usr/libexec/gsd-color
3150 : /usr/libexec/gsd-datetime
3153 : /usr/libexec/gsd-keyboard
3155 : /usr/libexec/gsd-media-keys
3157 : /usr/libexec/gsd-power
3160 : /usr/libexec/gsd-print-notifications
3161 : /usr/libexec/gsd-rfkill
3164 : /usr/libexec/gsd-sharing
3169 : /usr/libexec/gsd-sound
3175 : /usr/libexec/gsd-wacom
3176 : /usr/libexec/gsd-xsettings
3213 : /usr/libexec/evolution-data-server/evolution-alarm-notify
3219 : /usr/bin/gnome-software --gapplication-service
3272 : /usr/bin/abrt-applet --gapplication-service
3273 : /usr/bin/gjs /usr/share/gnome-shell/org.gnome.ScreenSaver
3366 : /usr/libexec/gsd-printer
3524 : /usr/libexec/fwupd/fwupd
3611 : /usr/bin/gnome-calendar --gapplication-service
3623 : /usr/libexec/gnome-terminal-server
3682 : /usr/libexec/bluetooth/obexd
3701 : /usr/libexec/tracker-miner-fs-3
3734 : bash
3833 : /usr/libexec/gvfsd-metadata
4345 : gpg-agent --homedir /var/cache/PackageKit/34/metadata/tor-34-x86_64.tmp/gpgdir_
↳--use-standard-socket --daemon
4357 : gpg-agent --homedir /var/cache/PackageKit/34/metadata/rpmsfusion-free-updates-
↳34-x86_64.tmp/gpgdir --use-standard-socket --daemon

```

(continues on next page)

(continued from previous page)

```

4382 : gpg-agent --homedir /var/cache/PackageKit/34/metadata/code-34-x86_64.tmp/
↳gpgdir --use-standard-socket --daemon
4483 : gpg-agent --homedir /var/cache/PackageKit/34/metadata/rpmsfusion-nonfree-
↳updates-34-x86_64.tmp/gpgdir --use-standard-socket --daemon
4495 : gpg-agent --homedir /var/cache/PackageKit/34/metadata/updates-modular-34-x86_
↳64.tmp/gpgdir --use-standard-socket --daemon
4529 : gpg-agent --homedir /var/cache/PackageKit/34/metadata/copr:copr.
↳fedorainfracloud.org:dawid:xcape-34-x86_64.tmp/gpgdir --use-standard-socket --daemon
4543 : gpg-agent --homedir /var/cache/PackageKit/34/metadata/rpmsfusion-nonfree-34-x86_
↳64.tmp/gpgdir --use-standard-socket --daemon
4556 : gpg-agent --homedir /var/cache/PackageKit/34/metadata/rpmsfusion-free-34-x86_64.
↳tmp/gpgdir --use-standard-socket --daemon
4568 : gpg-agent --homedir /var/cache/PackageKit/34/metadata/fedora-34-x86_64.tmp/
↳gpgdir --use-standard-socket --daemon
4581 : gpg-agent --homedir /var/cache/PackageKit/34/metadata/updates-34-x86_64.tmp/
↳gpgdir --use-standard-socket --daemon
4610 : gpg-agent --homedir /var/cache/PackageKit/34/metadata/google-chrome-34-x86_64.
↳tmp/gpgdir --use-standard-socket --daemon
4714 : /usr/libexec/flatpak-system-helper

```

8.9 Automatic updates in dnf systems

We can use **dnf-automatic** package to enable automatic installation of the updates. After you install the package, updated the configuration file `/etc/dnf/automatic.conf` to mark **apply_updates = yes**, by default it is set as *no*.

After that you can enable the timer, so that the packages get automatic updates installed.

```

$ sudo systemctl enable --now dnf-automatic.timer
Created symlink /etc/systemd/system/timers.target.wants/dnf-automatic.timer → /usr/
↳lib/systemd/system/dnf-automatic.timer.

```

In case if you want to only download the available updates, but not install them, you can enable the following timer.

```

$ sudo systemctl enable --now dnf-automatic-download.timer
Created symlink /etc/systemd/system/timers.target.wants/dnf-automatic-download.timer_
↳→ /usr/lib/systemd/system/dnf-automatic-download.timer.

```

In the configuration file, if you set **upgrade_type = security**, then the tool will only install security updates.

8.10 apt command

apt is the package management system for the *Debian* Linux distribution. As Ubuntu is downstream of the *Debian* distribution, it also uses the same package management system.

8.11 apt update

```

# apt update
... long output

```

The **apt update** command is used to update all the package information for the Debian repositories.

8.12 Installing a package via apt

`apt install packagename` is the command used to install any given package from the repository.

```
# apt install htop
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  lsof strace
The following NEW packages will be installed:
  htop
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 92.8 kB of archives.
After this operation, 230 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian buster/main amd64 htop amd64 2.2.0-1+b1 [92.8 kB]
Fetched 92.8 kB in 1s (113 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package htop.
(Reading database ... 6677 files and directories currently installed.)
Preparing to unpack ../htop_2.2.0-1+b1_amd64.deb ...
Unpacking htop (2.2.0-1+b1) ...
Setting up htop (2.2.0-1+b1) ...
```

8.13 apt-cache search

After you updated the cache, you can search for any package. Say, we want to search the package *neomutt*.

```
# apt-cache search neomutt
neomutt - command line mail reader based on Mutt, with added features
```

To know the exact policy (from where it will installed/upgrade or which version etc), you can use the following command.

```
# apt-cache policy libudev1
libudev1:
  Installed: 241-7~deb10u7
  Candidate: 241-7~deb10u8
  Version table:
     241-7~deb10u8 500
                500 http://security.debian.org/debian-security buster/updates/main amd64_
↳ Packages
   *** 241-7~deb10u7 500
                500 http://deb.debian.org/debian buster/main amd64 Packages
                100 /var/lib/dpkg/status
```

8.14 Listing upgrades

You can use `apt list --upgradable` to list all the packages that have updates in the repositories.

```
# apt list --upgradable
Listing... Done
```

(continues on next page)

(continued from previous page)

```
libsystemd0/stable 241-7~deb10u8 amd64 [upgradable from: 241-7~deb10u7]
libudev1/stable 241-7~deb10u8 amd64 [upgradable from: 241-7~deb10u7]
```

8.15 Upgrading packages

Use *apt dist-upgrade* to upgrade all the packages to the latest from the repositories.

```
# apt dist-upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  libsystemd0 libudev1
2 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 483 kB of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://security.debian.org/debian-security buster/updates/main amd64 libsystemd0 amd64 241-7~deb10u8 [331 kB]
Get:2 http://security.debian.org/debian-security buster/updates/main amd64 libudev1 amd64 241-7~deb10u8 [151 kB]
Fetched 483 kB in 1s (379 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
(Reading database ... 6677 files and directories currently installed.)
Preparing to unpack ../libsystemd0_241-7~deb10u8_amd64.deb ...
Unpacking libsystemd0:amd64 (241-7~deb10u8) over (241-7~deb10u7) ...
Setting up libsystemd0:amd64 (241-7~deb10u8) ...
(Reading database ... 6677 files and directories currently installed.)
Preparing to unpack ../libudev1_241-7~deb10u8_amd64.deb ...
Unpacking libudev1:amd64 (241-7~deb10u8) over (241-7~deb10u7) ...
Setting up libudev1:amd64 (241-7~deb10u8) ...
Processing triggers for libc-bin (2.28-10) ...
```

8.16 Figuring out which services/processes need restart after package upgrade in Debian

Just like Fedora systems, you can find the similar information in Debian too. You will need the **needrestart** package. The following is the output from the *needrestart* command.

```

Service restarts being deferred:
systemctl restart ModemManager.service
systemctl restart NetworkManager.service
/etc/needrestart/restart.d/dbus.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service
systemctl restart wpa_supplicant.service

No containers need to be restarted.

User sessions running outdated binaries:
kdas @ session #1: login[580]
kdas @ user manager service: systemd[1035]
root@debian:~# _

```

8.17 Listing available security updates in Debian systems

We can use the Debian Security Analyzer, **debsecan** tool for this. You have to install it via *apt* first. In the following example, we are checking system (running Debian Buster) against the available updates for security updates.

```

# apt install debsecan
# debsecan --suite buster --format packages --only-fixed
apache2-bin
firefox-esr
libnss-myhostname
libnss-systemd
libpam-systemd
libsystemd0
libudev1
linux-libc-dev
systemd
systemd-sysv
udev

```

8.18 Unattended upgrades in Debian systems

We can also setup the Debian systems for automatic upgrades. But, first install the **unattended-upgrades** and reconfigure it to download and apply the updates.

```

$ sudo apt install unattended-upgrades apt-listchanges && sudo dpkg-reconfigure -plow ↵
↳ unattended-upgrades

Creating config file /etc/apt/apt.conf.d/20auto-upgrades with new version

```

The configuration screen will look like below.

```
Package configuration
```

Configuring unattended-upgrades

Applying updates on a frequent basis is an important part of keeping systems secure. By default, updates need to be applied manually using package management tools. Alternatively, you can choose to have this system automatically download and install important updates.

Automatically download and install stable updates?

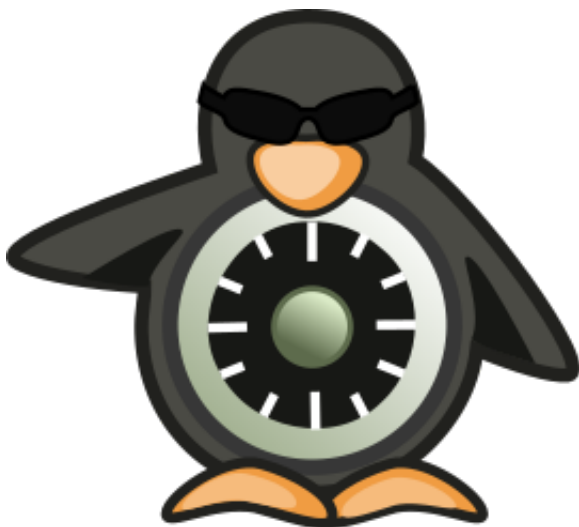
<Yes>

<No>

You can do a dry run afterward. By default *unattended-upgrade* will only install the **security** updates.

```
$ sudo unattended-upgrades --dry-run
/usr/bin/dpkg --status-fd 11 --no-triggers --unpack --auto-deconfigure --recursive /
↳tmp/apt-dpkg-install-xKmNdN
/usr/bin/dpkg --status-fd 11 --configure --pending
/usr/bin/dpkg --status-fd 11 --no-triggers --unpack --auto-deconfigure /var/cache/apt/
↳archives/libimobiledevice6_1.2.1~git20181030.92c5462-2+deb10u1_amd64.deb
```

For more details on this topic, please read the [Debian wiki page](#) on this topic.



Security-Enhanced Linux (SELinux) is a Linux kernel security module that provides a way to have access control security policies. This also allows a way to have Mandatory access control (MAC), according to the [Wikipedia](#):

In computer security, mandatory access control (MAC) refers to a type of access control by which the operating system constrains the ability of a subject or initiator to access or generally perform some sort of operation on an object or target.

The first version of SELinux was released in the year 2000 by NSA, and in 2003 it became part of the stable kernel. It was introduced in the Fedora Core 2, but by default it was disabled. From Fedora Core 3 it was enabled in the system.

For the rest of the chapter, you will need a Fedora/CentOS/RHEL installation.

9.1 SELinux Modes

There are 3 different modes.

- enforcing
- permissive
- disabled

By default your system will come with *enforcing* mode. In this mode the policies will be enforced in the system, and this should be used in every production system. In the *permissive* mode the policies will not be enforced but any denial is logged. The *disabled* mode completely disable the SELinux.

9.2 getenforce

The *getenforce* command will tell you the current SELinux mode.

```
$ getenforce
Enforcing
```

9.3 setenforce

Using *setenforce* command you can change the mode till the system reboots.

```
# setenforce
usage: setenforce [ Enforcing | Permissive | 1 | 0 ]
# setenforce Permissive
# getenforce
Permissive
# setenforce 1
# getenforce
Enforcing
```

Warning: Never disable SELinux on production systems, if required you can put them into permissive mode, so that you can get the denial logs, and create proper policies from those logs. Also check [this website](#) before further reading.

To change the label permanently, we modify the */etc/selinux/config* file.

```
$ sudo cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these three values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are
↳protected.
```

(continues on next page)

(continued from previous page)

```
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Change the value of *SELINUX* in the above mention file and then reboot the system to verify the change.

9.4 Labels/Contexts

Every process and object in the system has a corresponding label or context. This label defines which all processes can access which all objects. They have the following format:

```
user:role:type:range
```

The Fedora and other distributions use the *type* to define access control, the *range* is optional.

9.5 Checking contexts of files/directories or processes

You can use the *-Z* flag along with standard *ls* or *ps* command to see the SELinux context.

For example if you execute **ls -lZ** in your home directory.

```
$ ls -lZ
total 0
drwxr-xr-x. 11 vagrant vagrant unconfined_u:object_r:user_home_t:s0 222 Mar 21 05:38 ↵
↳lymworkbook
drwxrwxr-x.  3 vagrant vagrant unconfined_u:object_r:user_home_t:s0  21 Mar 29 11:55 ↵
↳Video
```

You can see the *unconfined_u:object_r:user_home_t:s0* and if you execute the same command against */tmp* then you will see the following:

```
$ ls -lZ /tmp
total 4
-rw-rw-r--. 1 vagrant vagrant unconfined_u:object_r:user_tmp_t:s0  0 Apr  2 03:18 ↵
↳example.txt
drwx-----. 3 root      root      system_u:object_r:tmp_t:s0                17 Mar 29 16:59 ↵
↳systemd-private-2aad7f8cd577426094e46ae7f4da1426-chronyd.service-gFq0Yn
-rwx--x--x. 1 vagrant vagrant unconfined_u:object_r:user_tmp_t:s0 205 Mar 21 05:17 ↵
↳vagrant-shell
```

The *type* context for temporary directory is *tmp_t* and when the user created those files under */tmp*, the context is *user_tmp_t*, for the user home directory it is *user_home_t*. The labels get matched against defined SELinux rules. The file's label stays in the extended attribute in the file system.

Now, let us execute the *ps* command with the *Z* flag.

```
$ ps auZ
LABEL                                USER      PID %CPU %MEM    VSZ   RSS TTY      STAT ↵
↳START  TIME COMMAND
system_u:system_r:getty_t:s0-s0:c0.c1023 root  776  0.0  0.3 15668 1812 ttyS0    Ss+  ↵
↳Mar29  0:00 /sbin/agetty -o -p -- \u --keep-baud 115200,38400,9600 ttyS0
system_u:system_r:getty_t:s0-s0:c0.c1023 root  777  0.0  0.3 13100 1696 tty1     Ss+  ↵
↳Mar29  0:00 /sbin/agetty -o -p -- \u --noclear tty1 linux
```

(continues on next page)

(continued from previous page)

```
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 vagrant 5373 0.0 0.8 27192_
↪4308 pts/0 Ss Mar31 0:00 -bash
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 vagrant 29048 0.0 0.7 57184_
↪3824 pts/0 R+ 03:21 0:00 ps auZ
```

Here you can see how different processes have different kind of *type* contexts. All *type* contexts generally ends with **_t**.

9.6 SELinux booleans

SELinux booleans are the rules which can be turned on or off. You can see all values (or a specific one) by using *getsebool* command.

```
$ getsebool -a
abrt_anon_write --> off
abrt_handle_event --> off
abrt_upload_watch_anon_write --> on
antivirus_can_scan_system --> off
antivirus_use_jit --> off
auditadm_exec_content --> on
authlogin_nsswitch_use_ldap --> off
authlogin_radius --> off
authlogin_yubikey --> off
awstats_purge_apache_log_files --> off
boinc_execmem --> on
cdrecord_read_content --> off
cluster_can_network_connect --> off
...
```

File system mounting

In this chapter, we'll learn how to mount file systems. If you type *mount* in the shell, it will tell you about various file systems, and how are they mounted (as a directory) in the system.

```
$ mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime,seclabel)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
devtmpfs on /dev type devtmpfs (rw,nosuid,seclabel,size=2012852k,nr_inodes=503213,
↪mode=755)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,seclabel)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,seclabel,gid=5,mode=620,
↪ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,seclabel,mode=755)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,seclabel,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,
↪release_agent=/usr/lib/systemd/systemd-cgroups-agent,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime,seclabel)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_
↪event)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,
↪cpuacct)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,
↪relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
configfs on /sys/kernel/config type configfs (rw,relatime)
/dev/vda1 on / type ext4 (rw,relatime,seclabel,data=ordered)
selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=23,pgrp=1,timeout=0,
↪minproto=5,maxproto=5,direct,pipe_ino=11175)
```

(continues on next page)

(continued from previous page)

```
mqueue on /dev/mqueue type mqueue (rw,relatime,seclabel)
debugfs on /sys/kernel/debug type debugfs (rw,relatime,seclabel)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,seclabel)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,seclabel,size=404680k,
->mode=700,uid=1000,gid=1000)
```

If you look carefully at the output above, you'll find that `/dev/vda1` is mounted as `root/` in the system. This is actually the primary hard drive in this system. The device can be different based on the system.

- `/dev/vd*` For virtual machines
- `/dev/sd*` For physical machines

The number at the end of the device name is the partition number.

10.1 Connecting USB drives to your system

If you connect vfat partitioned USB drives (the normal pendrives), they will auto mount under the `/run/media/username/` directory. But, for NTFS based drives, you will have to install the driver to mount those partitions.

```
$ sudo dnf install ntfs-3g -y
```

10.2 Mounting a device

We can use the `mount` command to mount a file system on an existing directory. The syntax to do that is, `mount device /path/to/mount/at`.

```
$ sudo mount /dev/sdb1 /mnt
```

In the example above, we mounted `/dev/sdb1` on the `/mnt` directory.

10.3 Unmounting

We use the `umount` command on a given directory to unmount the file system.

Do not remove any drive from the system before unmounting them. Just to be on the safe side, you can execute the `sync` command, which will write any existing cache to the drives. That will make sure that your chances of losing data is marginal.

10.4 Encrypting drives with LUKS (for only Linux)

Follow [this link](#) to learn about how to encrypt your drives with LUKS. This is a simple way to make sure that even if you lose your USB drive, the data inside can still be safe (relatively).

10.5 Encrypting drives for any OS using Veracrypt

VeraCrypt is an open source volume management tool compatible with macOS, Windows, and Linux systems.

Here is an excellent guide from [Freedom of the Press Foundation](#) on how to use it.

Networking commands

In this chapter, we will learn about a few basic networking commands, which will help us in our daily Linux usage.

11.1 Finding the IP address

The *ip* command can be used to find the IP address of the system.

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default_
↳qlen 1
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
   valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
   valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1454 qdisc fq_codel state UP group_
↳default qlen 1000
   link/ether fa:16:3e:3c:ed:27 brd ff:ff:ff:ff:ff:ff
   inet 172.10.95.253/20 brd 172.10.111.255 scope global dynamic eth0
   valid_lft 57021sec preferred_lft 57021sec
   inet6 fe80::f816:3eff:fe3c:ed27/64 scope link
   valid_lft forever preferred_lft forever
```

Here *lo* is a special device which points to the same system (also known as *localhost*). The IP *127.0.0.1* always points to the the *localhost*. *eth0* is our ethernet device which connects to the network.

11.2 ping command

ping is simple way to find if you are connected to Internet or not. We can also ping any particular computer to find if the computer is connected to the network or not. Press *Ctrl+c* to stop the loop.

```
$ ping google.com
PING google.com (216.58.201.142) 56(84) bytes of data.
64 bytes from mad06s25-in-f142.1e100.net (216.58.201.142): icmp_seq=1 ttl=44 time=157.
↪ms
64 bytes from mad06s25-in-f142.1e100.net (216.58.201.142): icmp_seq=2 ttl=44 time=156.
↪ms
64 bytes from mad06s25-in-f142.1e100.net (216.58.201.142): icmp_seq=3 ttl=44 time=156.
↪ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 156.373/156.811/157.566/0.704 ms
```

11.3 Short note about DNS

DNS or Domain Name System is a decentralized naming system for systems which are connected to Internet (can be for private networks too). This is the way a computer knows, which other computer to connect to, when we type `google.com` in our browser, or in the ping command. There are servers known as dns servers, and for every domain name it needs to find, the client system generally connects to these dns servers, and finds out the IP address of the computer at that domain name.

11.4 `/etc/hosts`

The system looks at this file first for any name resolution. If it can not find the DNS entry, then the system looks at the `/etc/resolv.conf`, and connects to the DNS server.

You can update `/etc/hosts` file to add a domain to any particular IP address. Say, you want to be able to reach a server at IP address `x.x.x.x` with the name `datastore`, so you add an entry like the following to the file.

```
x.x.x.x    datastore
```

11.5 `/etc/resolv.conf`

`/etc/resolv.conf` is the configuration file which contains the DNS server address to use for DNS queries.

```
$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 1.1.1.1
nameserver 8.8.8.8
```

Here you can see that `1.1.1.1` & `8.8.8.8` are two DNS servers are being in this machine. The `1.1.1.1` is the DNS server from Cloudflare, and `8.8.8.8` is the DNS server hosted by Google.

11.6 `systemd-resolved` controlled name resolution

In most of the modern systems you will find the `/etc/resolv.conf` looks a bit different and actually a symbolic link. The example below is from *Ubuntu 20.04*.


```

$ ls -l /etc/resolv.conf
lrwxrwxrwx 1 root root 39 Jul 31 2020 /etc/resolv.conf -> ../run/systemd/resolve/
↳ stub-resolv.conf

$ cat /etc/resolv.conf
# This file is managed by man:systemd-resolved(8). Do not edit.
#
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "resolvectl status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs must not access this file directly, but only through the
# symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a different way,
# replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.

nameserver 127.0.0.53
options edns0 trust-ad
search localdomain

```

That strange IP address, **127.0.0.53** is a special one managed by **systemd-resolved** service, where it listens for DNS queries. By default it picks up the DNS server addresses provided by the *DHCP* service, in case you want to manually set that up, you can configure them at the */etc/systemd/resolved.conf* file. Here we are setting *1.1.1.1* as the primary DNS server, *8.8.8.8* as the fallback server, and also enabling *DNS over TLS*.

```

# This file is part of systemd.
#
# systemd is free software; you can redistribute it and/or modify it
# under the terms of the GNU Lesser General Public License as published by
# the Free Software Foundation; either version 2.1 of the License, or
# (at your option) any later version.
#
# Entries in this file show the compile time defaults.
# You can change settings by editing this file.
# Defaults can be restored by simply deleting this file.
#
# See resolved.conf(5) for details

[Resolve]
DNS= 1.1.1.1
FallbackDNS= 8.8.8.8
#Domains=
#LLMNR=no
#MulticastDNS=no
#DNSSEC=no
DNSOverTLS=yes
#Cache=no-negative
#DNSStubListener=yes
#ReadEtcHosts=yes

```

You can learn about all the settings from the man page, *man resolved.conf*.

If want to go back to the simple */etc/resolv.conf* file, you can remove the symlink and put in a new file at

/etc/resolv.conf.

11.7 resolvectl command

The *resolvectl* command helps us to query via the *systemd-resolved* service. To check the current settings, use the *status* flag.

```
$ resolvectl status
Global
    LLMNR setting: no
MulticastDNS setting: no
    DNSOverTLS setting: yes
    DNSSEC setting: no
    DNSSEC supported: no
    DNS Servers: 1.1.1.1
Fallback DNS Servers: 8.8.8.8
    DNSSEC NTA: 10.in-addr.arpa
                16.172.in-addr.arpa
                168.192.in-addr.arpa
                17.172.in-addr.arpa
                18.172.in-addr.arpa
                19.172.in-addr.arpa
                20.172.in-addr.arpa
                21.172.in-addr.arpa
                22.172.in-addr.arpa
                23.172.in-addr.arpa
                24.172.in-addr.arpa
                25.172.in-addr.arpa
                26.172.in-addr.arpa
                27.172.in-addr.arpa
                28.172.in-addr.arpa
                29.172.in-addr.arpa
                30.172.in-addr.arpa
                31.172.in-addr.arpa
                corp
                d.f.ip6.arpa
                home
                internal
                intranet
                lan
                local
                private
                test

Link 2 (ens33)
    Current Scopes: DNS
DefaultRoute setting: yes
    LLMNR setting: yes
MulticastDNS setting: no
    DNSOverTLS setting: yes
    DNSSEC setting: no
    DNSSEC supported: no
    DNS Servers: 192.168.195.1
    DNS Domain: ~.
                localdomain
```

To query the IP address of a domain:

```

resolvectl query fedoraproject.org
fedoraproject.org: 140.211.169.196      -- link: ens33
                   140.211.169.206      -- link: ens33
                   152.19.134.198       -- link: ens33
                   38.145.60.21         -- link: ens33
                   8.43.85.67          -- link: ens33
                   152.19.134.142       -- link: ens33
                   209.132.190.2        -- link: ens33
                   38.145.60.20         -- link: ens33
                   67.219.144.68        -- link: ens33
                   8.43.85.73          -- link: ens33

-- Information acquired via protocol DNS in 1.4ms.
-- Data is authenticated: no

```

To view the *TXT* record:

```

$ resolvectl query -t TXT fedoraproject.org
fedoraproject.org IN TXT "v=spf1 a:mailers.fedoraproject.org ip4:38.145.60.11
↪ip4:38.145.60.12 ?all" -- link: ens33

-- Information acquired via protocol DNS in 289.7ms.
-- Data is authenticated: no

```

resolvectl command can do many more things. Please have a look at the man page for more examples.

11.8 host command

The **host** command is a simple DNS lookup utility. It will show you the IP address of any given hostname.

```

$ host www.example.com
www.example.com has address 93.184.216.34
www.example.com has IPv6 address 2606:2800:220:1:248:1893:25c8:1946

```

11.9 dig command

dig command can tell us DNS records, MX details (used to send emails) and other information for a given domain name.

```

$ dig kushaldas.in

; <<>> DiG 9.10.4-P8-RedHat-9.10.4-5.P8.fc25 <<>> kushaldas.in
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50750
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;kushaldas.in.                IN      A

```

(continues on next page)

(continued from previous page)

```
;; ANSWER SECTION:
kushaldas.in.          5528    IN      A       208.113.152.208

;; Query time: 66 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sun Jun 25 11:37:00 IST 2017
;; MSG SIZE rcvd: 57
```

If you want to print only the IP address in the output, you can use `+short` as argument.

```
$ dig +short kushaldas.in
208.113.152.208
```

You can also specify any particular type of record from the DNS server. For example, if I want to get the `TXT` records, I can do the following command.

```
$ dig +short kushaldas.in TXT
"google-site-verification=DppUk-OfBLT-5PkbSR9VM2Uht3eXaksthROvS-L9iKY"
"kolab-verify=35f0040cd1ebb20fb7f0b3fade0e1c8e"
```

You can use any of the following options instead of `TXT`: `A|MX|TXT|CNAME|NS`.

If you want to specify a DNS server to use, you can do that with the address specified at the end of the command along with a `@` sign.

```
$ dig rtnpro.com @208.67.222.222

; <<>> DiG 9.10.4-P8-RedHat-9.10.4-5.P8.fc25 <<>> rtnpro.com @208.67.222.222
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27312
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;rtnpro.com.                IN      A

;; AUTHORITY SECTION:
rtnpro.com.                 3600    IN      SOA     dns1.bigrock.in. rtnpro.gmail.com.
↪2017021401 7200 7200 172800 38400

;; Query time: 899 msec
;; SERVER: 208.67.222.222#53(208.67.222.222)
;; WHEN: Sun Jun 25 11:40:01 IST 2017
;; MSG SIZE rcvd: 106
```

If you want learn about the full path of the DNS query, you can find that out by `+trace` flag. For the following example where we are trying to find the IP address for `anweshadas.in`, `dig` first connects to the root name servers to find the correct DNS resolvers for the `.in` servers, then they tell `dig` to go those servers (in this case Dreamhost name servers) which contain the exact IP address for the domain.

```
$ dig +trace anweshadas.in @8.8.8.8

; <<>> DiG 9.11.5-P4-5.1+deb10u5-Debian <<>> +trace anweshadas.in @8.8.8.8
;; global options: +cmd
```

(continues on next page)

(continued from previous page)

```

.          47041  IN      NS      m.root-servers.net.
.          47041  IN      NS      b.root-servers.net.
.          47041  IN      NS      c.root-servers.net.
.          47041  IN      NS      d.root-servers.net.
.          47041  IN      NS      e.root-servers.net.
.          47041  IN      NS      f.root-servers.net.
.          47041  IN      NS      g.root-servers.net.
.          47041  IN      NS      h.root-servers.net.
.          47041  IN      NS      a.root-servers.net.
.          47041  IN      NS      i.root-servers.net.
.          47041  IN      NS      j.root-servers.net.
.          47041  IN      NS      k.root-servers.net.
.          47041  IN      NS      l.root-servers.net.
.          47041  IN      RRSIG   NS 8 0 518400 20210730050000_
↪20210717040000 26838 . MFT2Q71k1LZVfXyH2qKwLoS7a7j5aSvdlp4SrIptZXP0ydjav7y5sLv/_
↪Yz76Ki+3PU0G3SagwbC61bdi6sNV5DiBpxIzny8Mavx23P6XKsbetFr1_
↪RgkwlzyGJmd0kLA4ydgjrzRh2hhvQkBDWtzBpVLUo7tDmwodE/zi/RUA CMofG9YIkgxSX0/
↪5qUUKXhijHocYXQU++x7RbFqTxJBW8Fn6GDDTtg1Z_
↪pTT0UYpmMX5NHiRlneYb6ChHGQLfbQ1kBblxulSb46dJBKaXT3wr3/_
↪SXUXQCZ+ADCsolK+LhGeQtByqBEXryjuT/U2WK8mqcTAs/dlbToRwrH5 nxizXg==
;; Received 525 bytes from 8.8.8.8#53(8.8.8.8) in 1 ms

in.        172800  IN      NS      ns1.registry.in.
in.        172800  IN      NS      ns2.registry.in.
in.        172800  IN      NS      ns3.registry.in.
in.        172800  IN      NS      ns4.registry.in.
in.        172800  IN      NS      ns5.registry.in.
in.        172800  IN      NS      ns6.registry.in.
in.        86400   IN      DS      54739 8 1_
↪2B5CA455A0E65769FF9DF9E75EC40EE1EC1CDCA9
in.        86400   IN      DS      54739 8 2_
↪9F122CFD6604AE6DEDA0FE09F27BE340A318F06AFAC11714A73409D4 3136472C
in.        86400   IN      RRSIG   DS 8 1 86400 20210730170000_
↪20210717160000 26838 . i6toEqveLqwB/W4Z/77bfGyFyYJRepGi8uYoQ0jEZM1I95qsqMctdV_
↪cr3foafFJKaCkvH2eAfIUrHH8GMn/t91VDrHwikLisoaPaSahgoQAOPm C1R/
↪VdCAxkVwE+07Ir6ROt+qXn5jse4gnB+nezI4Q+rakearp8D9AaxJ ubWnAMfHOqKBLDMGNrm6/
↪XRk6HA43nrMIUKNCFbhpKo5gkvy+S768uQu_
↪ySRdLTUxN0ELO9Qv7fBqQxamRyZ1N5LKTpjKKNKYwnihOVIWvktqt4p7_
↪xoJL56z0XE9HhhI807GOBcpLBeaRKZXA8GKU77pm91DLHSuG4epF3zd X9Vayw==
;; Received 794 bytes from 193.0.14.129#53(k.root-servers.net) in 1 ms

aneshadas.in.          86400   IN      NS      ns3.dreamhost.com.
aneshadas.in.          86400   IN      NS      ns2.dreamhost.com.
aneshadas.in.          86400   IN      NS      ns1.dreamhost.com.
bo801o0uciino3vfr38lrljcrv2ucohi.in. 1800 IN NSEC3 1 1 1 00763C64_
↪BO9UQ54VB22M3J37NR3N6GRC6J4RVUTV NS SOA RRSIG DNSKEY NSEC3PARAM
bo801o0uciino3vfr38lrljcrv2ucohi.in. 1800 IN RRSIG NSEC3 8 2 1800 20210813034112_
↪20210714030559 65169 in. gZ3NODrbaP6/GV1McvgHTD4wn9w2w5CCqjoI+JyjRpNVweGuDex5A/ls_
↪OznLptg/nmmJlx3835suy9I79h0tOjDjWNXxLQ9scKctYZJsfqIdnRS9 QP5egjVJnZ3zOLN010/hQa/
↪gIhKCSqYpLCWLS1RoFn3B5uvF96VopKU YYfjXfbqYcJyx4T8oZi72xFUChr/yi/dVkhBMOOvwLCJrg==
8679tah9aq7s760bquasj6clf332vb3e.in. 1800 IN NSEC3 1 1 1 00763C64_
↪869E0HEFFMPE89PM1VHLGQHH72K7IGRM NS DS RRSIG
8679tah9aq7s760bquasj6clf332vb3e.in. 1800 IN RRSIG NSEC3 8 2 1800 20210815211611_
↪20210716202452 65169 in. GahrJsYIVpR5+eaykB/DuaIPSuEr+rX3DxR7yR3mMN/7pVSEkbZ/Fw4I_
↪q5NZLKhBp2WKq0aiwxYX4+VfKjdLyQLKeGoXYeFpwpS6KiCf8gLjDJwO_
↪Hx1PCgg5CnUEYw+iXd3GGx46Z1yHHbuSna4YbVBEHvcmo/3oc3ubiMf_
↪VKTcuR+upzBQmLzNid6LB9qZBpFfe6GPCy/sMMaCKB00wQ==

```

(continues on next page)

(continued from previous page)

```
; Received 664 bytes from 37.209.196.12#53(ns3.registry.in) in 2 ms
anweshadas.in.          14400   IN      A       159.89.209.77
; Received 58 bytes from 162.159.26.14#53(ns1.dreamhost.com) in 160 ms
```

11.10 ss command

ss command shows us socket statistics from the system. This command replaces the older **netstat** command. Read the man page of the command to know more about the different arguments we can pass at the command line.

```
$ ss -putn
Netid State      Recv-Q Send-Q           Local Address:Port      Peer Address:Port
↪
tcp    ESTAB      0      0           192.168.1.101:51496     162.125.34.129:443
↪                                     users: ("dropbox",
↪pid=28797,fd=80))
tcp    ESTAB      0      0           192.168.1.101:47864     74.125.200.189:443
↪                                     users: ("chrome",
↪pid=22112,fd=385))
tcp    ESTAB      0      0           192.168.1.101:59524     209.12.123.55:22
↪                                     users: ("ssh",
... long output
```

You can also learn various other statistics about sockets, for example, if you want to know all the sockets from your computer connected to any machine at port *443*, you can use the following command.

```
$ ss -pt dst :443
```

11.11 traceroute command

The **traceroute** command is used to show the full route of a network packet from the system to any given host.

```
$ traceroute www.rtnpro.com
traceroute to www.rtnpro.com (146.185.181.157), 30 hops max, 60 byte packets
 1 gateway (192.168.1.1)  1.434 ms  1.920 ms  1.891 ms
 2 45.113.248.3 (45.113.248.3)  7.478 ms  10.335 ms  10.343 ms
 3 45.113.248.1 (45.113.248.1)  10.319 ms  10.293 ms  10.274 ms
 4 121.244.26.1.static-pune.vsnl.net.in (121.244.26.1)  26.938 ms  26.608 ms  27.165
↪ms
 5 172.31.183.162 (172.31.183.162)  9.883 ms  10.133 ms  10.122 ms
 6 172.31.19.201 (172.31.19.201)  10.591 ms  172.29.250.33 (172.29.250.33)  6.894 ms
↪172.31.19.201 (172.31.19.201)  8.203 ms
 7 ix-ae-0-4.tcore1.MLV-Mumbai.as6453.net (180.87.38.5)  9.378 ms  8.886 ms  9.240 ms
 8 if-ae-9-5.tcore1.WYN-Marseille.as6453.net (80.231.217.77)  159.550 ms  if-ae-5-2.
↪tcore1.WYN-Marseille.as6453.net (180.87.38.126)  159.614 ms  if-ae-9-5.tcore1.WYN-
↪Marseille.as6453.net (80.231.217.77)  159.506 ms
 9 if-ae-8-1600.tcore1.PYE-Paris.as6453.net (80.231.217.6)  159.392 ms  159.474 ms
↪159.405 ms
10 if-ae-15-2.tcore1.AV2-Amsterdam.as6453.net (195.219.194.145)  159.327 ms  158.355
↪ms  122.520 ms
```

(continues on next page)

(continued from previous page)

```

11 195.219.194.26 (195.219.194.26) 133.216 ms 134.168 ms 134.683 ms
12 138.197.250.29 (138.197.250.29) 192.236 ms 192.125 ms 138.197.250.23 (138.197.
↪250.23) 192.083 ms
13 * 146.185.181.157 (146.185.181.157) 191.831 ms 191.861 ms

```

11.12 tracepath command

The **tracepath** command traces a path to a network host discovering MTU along the path. This is a modern replacement of the *traceroute* command, and also does not need superuser privileges to execute.

```

$ tracepath www.rtnpro.com
1?: [LOCALHOST]                pmtu 1500
1: gateway                      0.950ms
1: gateway                      0.715ms
2: gateway                      0.689ms pmtu 1492
2: 45.113.248.3                 3.564ms
3: 45.113.248.1                 4.639ms
4: 121.244.26.1.static-pune.vsnl.net.in 4.132ms
5: 172.31.183.162               4.733ms asymm 7
6: 172.29.250.33               12.524ms asymm 7
7: ix-ae-0-4.tcore1.MLV-Mumbai.as6453.net 7.208ms asymm 8
8: if-ae-5-2.tcore1.WYN-Marseille.as6453.net 125.727ms asymm 12
9: if-ae-8-1600.tcore1.PYE-Paris.as6453.net 128.893ms asymm 11
10: if-ae-15-2.tcore1.AV2-Amsterdam.as6453.net 126.019ms asymm 9
11: 195.219.194.26              136.373ms asymm 10
12: 138.197.250.27              130.198ms
13: 146.185.181.157             131.040ms reached
Resume: pmtu 1492 hops 13 back 13

```

11.13 Remote login to a computer using ssh tool

We use the **ssh** command to login to remote computers. The remote computer must have the **sshd** service running, and should also allow clients to connect to this service. Let's try to connect to localhost itself. Remember to start the **sshd** service before this step.

```

$ ssh kdas@localhost
kdas@localhost's password:
Last login: Wed Jun 21 08:44:40 2017 from 192.168.1.101
$

```

As you can see, the command syntax is **ssh** followed by **user@hostname**. If your remote system's user name is same as your current one, then you can omit the username and just use the hostname (IP address or domain name).

```

$ ssh localhost
kdas@localhost's password:
$

```

11.14 ssh key generation

ssh keys are used in the daily life of a Linux user or developer. In simple terms, it helps us to securely login to other computers. In the following example, we will create a new key for our user.

```
$ ssh-keygen -t rsa -b 4096 -C "kushaldas@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/fedora/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/fedora/.ssh/id_rsa.
Your public key has been saved in /home/fedora/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:O6Rxir7lpFBQsBnvs+NJRU8Ih01ffVBvLTE8s5TpxLQ kushaldas@gmail.com
The key's randomart image is:
+---[RSA 4096]-----+
|  O.O+O  ...*=O |
|  *.O.O  .  . @=. |
|  + . O O   =E++ |
|  o . o     oo |
|    + o S      |
|    . = * .    |
|    . = = o    |
|    = B .      |
|    *..        |
+-----[SHA256]-----+
```

As you can see in the output, the key has been saved in the `~/.ssh` directory. You can also find out that these files are only readable by the owner.

```
$ ls -l .ssh
total 12
-rw-----. 1 fedora fedora 3326 Jun 25 06:25 id_rsa
-rw-r--r--. 1 fedora fedora  745 Jun 25 06:25 id_rsa.pub
```

Each key has two parts. The `id_rsa.pub` is the public key and `id_rsa` is the private part of the key. One can safely upload or use the public key anywhere. But the private key, should be kept in a safe manner, because if people get access to your private key, they can also access all of your information from any system using that key.

In other words, do not give the private key to anyone, or do not randomly copy the `.ssh` directory to a USB drive and then forget about it.

11.15 ssh-copy-id

`ssh-copy-id` command copies the keys to a given remote system. After this step we can use the ssh key to login to the box directly, instead of the usual username / password method.

```
$ ssh-copy-id fedora@209.12.123.55
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out
↳any that are already installed
/usr/bin/ssh-copy-id: INFO: 2 key(s) remain to be installed -- if you are prompted
↳now it is to install the new keys

fedora@209.12.123.55's password:
```

(continues on next page)

(continued from previous page)

```
Number of key(s) added: 2
```

```
Now try logging into the machine, with: "ssh 'fedora@209.12.123.55'"
and check to make sure that only the key(s) you wanted were added.
```

11.16 Stop and disable the sshd service

If you don't need ssh access to your computer (say, your laptop), you should always stop and disable the *sshd* service in the computer.

11.17 Disable password based login for ssh

Remember, this step can be **dangerous**. Unless you're really, really sure that you can access a computer by either logging in physically or using your ssh key (and you have a backup of the key somewhere), you should not do this step.

By disabling password based login in the *sshd* service, you make sure that only people with the right private key can login to the computer. This helps greatly when people try to break into the system by guessing the password. This is also really helpful in case your computer is connected to some network, and you still need to access it over ssh.

We will use vim to open the */etc/ssh/sshd_config* file, which is the configuration file for *sshd* service.

```
$ sudo vim /etc/ssh/sshd_config
```

Search for the term *PasswordAuthentication*, and change the value to no. Below I have added a new line to do the same. You can also understand, that the lines starting with # are comments in this configuration file. This configuration will disable password based authentication for the *sshd* service. You should remember to restart the *sshd* service after this step for the change to take place.

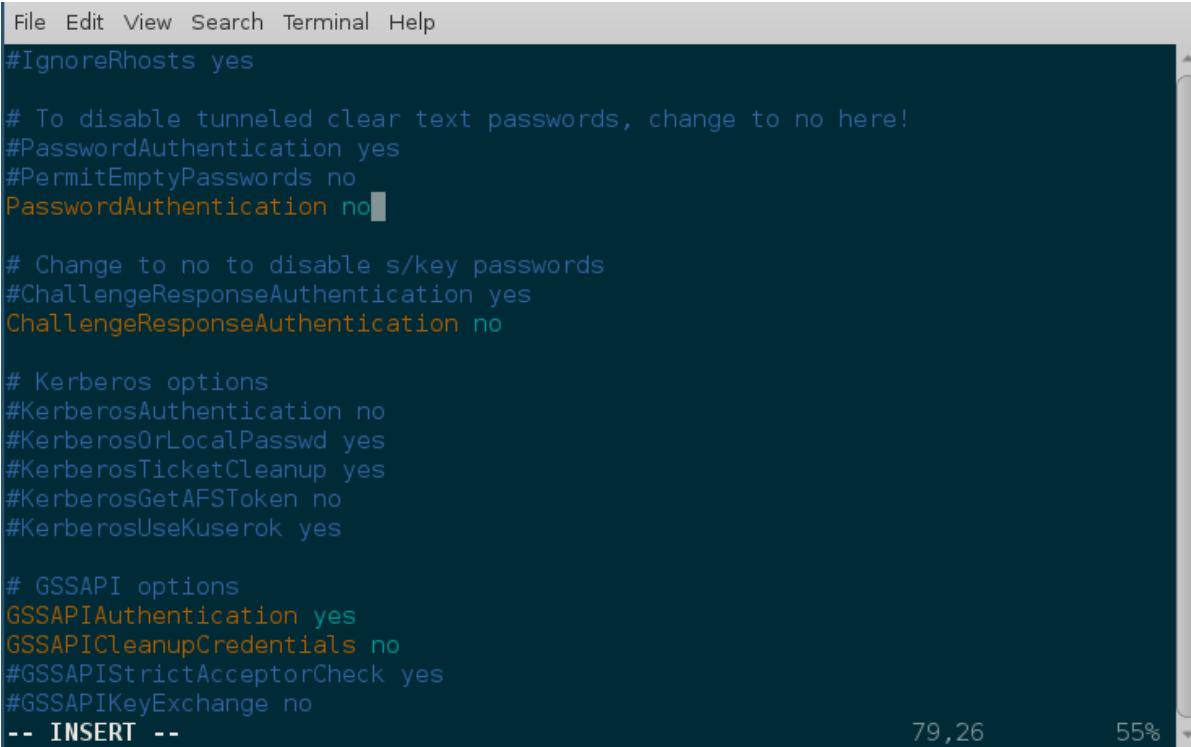
11.18 How to find active (open) network connections from your computer?

```
$ sudo lsof -i -n -P
```

The *lsof* command shows open files, using *-i* we are asking to list of all Internet and x.25 (HP-UX) network files. To know more, read the man page of the *lsof* command.

11.19 To know about ports

If you want to know more about popular ports used by the applications/protocols, you should look into */etc/services* file on your computer.



```
File Edit View Search Terminal Help
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication no

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no
#KerberosUseKuserok yes

# GSSAPI options
GSSAPIAuthentication yes
GSSAPICleanupCredentials no
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no
-- INSERT --
79,26 55%
```



Note: This chapter is an ongoing work.

A firewall is a network security system, which can monitor and control network packets coming in and going out from a system based on pre-defined rules.

In this chapter, we will learn about **iptables** command and how can we use the same to create and manage the system's firewall. The **netfilter** subsystem in Linux Kernel handles the actual packet filtering in the network level.

12.1 Installation

On CentOS

```
yum install iptables-services
```

On Debian systems

```
apt install iptables-persistent
```

12.2 Tables, chains and rules

There is a table based system which in turn uses chains of rules for the firewall. Each table has a defined set of chains, and the rules get into the get chain one after another.

When a network packet reaches the related table, and the related chain inside of the table, the rules gets matched from top to bottom. If the packet matches then the *target* of the rule gets executed. Each chain also has a default policy, if no rule matches, then, the default poilicy gets applied on the packet. We will learn more about these in details.

iptables has 5 built in chains.

- **INPUT** for all packets incoming to the system
- **OUTPUT** for all packets going out from the system
- **FORWARD** for the routed packets, this is when the system works as a router
- **PREROUTING** for port forwarding
- **POSTROUTING** for Source Network Address Translation (SNAT), this applies to all packets leaving the system

12.3 filter table

filter is the default table of iptables. It has 3 default chains.

- INPUT
- OUTPUT
- FORWARD

12.4 nat table

nat table is a special table for SNAT and DNAT (port forwarding). It has the following chains.

- PREROUTING
- POSTROUTING
- OUTPUT

There are two other different tables, **mangle** and **raw**.

12.5 iptables command

The following table will be helpful in remembering different arguments to **iptables** command.

↪-----+	-----+	-----+	-----+	-----+
↪ Table	Command	Chain	Matches	↪
↪ Target/Jump				
↪-----+	-----+	-----+	-----+	-----+
↪ filter (default)	-A (append)	INPUT	-p protocol	↪
↪ ACCEPT				
↪ nat	-I (insert)	OUTPUT	-s source_ip	↪
↪ DROP				
↪ mangle	-D (delete)	FORWARD	-d destination_ip	↪
↪ LOG				
↪ raw	-R (replace)	PREROUTING	--sport source_port	↪
↪ REJECT				

(continues on next page)

(continued from previous page)

↪ DNAT		-F (flush)		POSTROUTING		--dport destination_ip		↵
↪ SNAT		-L (list)		USER_DEFINED_CHAINS		-i incoming		↵
↪ LIMIT		-S (show)				-o outgoing		↵
↪ RETURN		-Z (zero)				-m mac		↵
↪ MASQUERADE		-N				-m time		↵
↪		-X				-m quota		↵
↪						-m limit		↵
↪						-m recent		↵
+-----+								
↪	-----+							

12.6 View the existing rules

```
# iptables -nvL --line-numbers
Chain INPUT (policy ACCEPT 82 packets, 4756 bytes)
num  pkts bytes target     prot opt in     out     source      destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source      destination
Chain OUTPUT (policy ACCEPT 42 packets, 3192 bytes)
num  pkts bytes target     prot opt in     out     source      destination
```

The above command shows the default table **filter** and all chains and rules inside of it. You can notice that each of the chains has a default policy **ACCEPT**. It means if no rules match (in this case no rules are defined), it will accept those packets.

12.7 Appending rules to INPUT chain

We can test an initial rule to **drop** all incoming *icmp* packets to the system. The following rule will append the rule to the **INPUT** chain.

Note: *ping* command uses *icmp* packets. So, the following command will block *ping* into the system.

```
iptables -A INPUT -p icmp -j DROP
```

Now, if you try to ping the system from any computer, you will not get any response.

12.8 Flushing all rules

```
iptables -F
```

The above command will help to flush (remove) all the rules from the default table. You can actually use `-t TABLE_NAME` argument to flush any particular table.

12.9 Example of a series of rules

Here is a list of rules to allow traffic to port 22 (ssh) and port 80 and 443 (http and https).

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -m state --state NEW --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -j ACCEPT
iptables -A INPUT -j REJECT
iptables -A FORWARD -j REJECT
```

The first rules allows all incoming traffic on the *loopback* device. The second line allows packets related to an already established connection, or the cases where a packet is trying to reconnect. The last 3rd last line allows all outgoing packets, and the last 2 lines reject everything else which does not match the rules. If you want to view all the rules.

```
# iptables -nvL --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source         destination
 1      0    0 ACCEPT     all  --  lo     *       0.0.0.0/0     0.0.0.0/0
 2     122 9641 ACCEPT     all  --  *      *       0.0.0.0/0     0.0.0.0/0
↳      state RELATED,ESTABLISHED
 3      1    52 ACCEPT     tcp  --  *      *       0.0.0.0/0     0.0.0.0/0
↳      state NEW tcp dpt:22
 4      0    0 ACCEPT     tcp  --  *      *       0.0.0.0/0     0.0.0.0/0
↳      tcp dpt:80
 5      0    0 ACCEPT     tcp  --  *      *       0.0.0.0/0     0.0.0.0/0
↳      tcp dpt:443
 6     22 2044 REJECT     all  --  *      *       0.0.0.0/0     0.0.0.0/0
↳      reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source         destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source         destination
 1     104 12085 ACCEPT     all  --  *      *       0.0.0.0/0     0.0.0.0/0
```

The `--line-numbers` argument shows the number of the each rule. We can use these line numbers to delete any rule.

Note: For a desktop or laptop, you may want to drop all incoming connections, that will help in cases where someone in the local network may try to attack/scan your system.

12.10 Delete a rule based on rule number

Let us delete the rule number 4, which allows traffic to port 80.

```
# iptables -D INPUT 4
# iptables -nvL --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source         destination
1    4    376 ACCEPT     all  --  lo     *       0.0.0.0/0     0.0.0.0/0
2    221 15445 ACCEPT     all  --  *      *       0.0.0.0/0     0.0.0.0/0
↳    state RELATED,ESTABLISHED
3    1    52  ACCEPT     tcp  --  *      *       0.0.0.0/0     0.0.0.0/0
↳    state NEW tcp dpt:22
4    0    0  ACCEPT     tcp  --  *      *       0.0.0.0/0     0.0.0.0/0
↳    tcp dpt:443
5    22  2044 REJECT     all  --  *      *       0.0.0.0/0     0.0.0.0/0
↳    reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source         destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source         destination
1    166 17248 ACCEPT     all  --  *      *       0.0.0.0/0     0.0.0.0/0
```

12.11 Delete a rule directly

If you know the rule properly, you can also delete it based on the rule directly.

```
# iptables -D INPUT -p tcp --dport 443 -j ACCEPT
# iptables -nvL --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source         destination
1    4    376 ACCEPT     all  --  lo     *       0.0.0.0/0     0.0.0.0/0
2    344 22417 ACCEPT     all  --  *      *       0.0.0.0/0     0.0.0.0/0
↳    state RELATED,ESTABLISHED
3    1    52  ACCEPT     tcp  --  *      *       0.0.0.0/0     0.0.0.0/0
↳    state NEW tcp dpt:22
4    22  2044 REJECT     all  --  *      *       0.0.0.0/0     0.0.0.0/0
↳    reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source         destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source         destination
1    234 22564 ACCEPT     all  --  *      *       0.0.0.0/0     0.0.0.0/0
```

12.12 Saving the rules

Any change made via **iptables** command stays on memory. To save it (so that it autoreloads in reboot), use the following command.

For Debian.

```
# netfilter-persistent save
```

For CentOS 7+

```
# systemctl stop firewalld && systemctl disable firewalld
# iptables-save > /etc/sysconfig/iptables
# systemctl enable iptables
Created symlink from /etc/systemd/system/basic.target.wants/iptables.service to /usr/
↳ lib/systemd/system/iptables.service.
# systemctl start iptables
```

The first line stops and then disables the **firewalld** service, it is a newer type of frontend for the same *netfilter* subsystem of the kernel.

12.13 A blog post from Major Hayden

Now, you should read the following [blog post](#) from Major Hayden best practices.

12.14 Debugging firewall rules

In case you want to debug the rules, and want to see which packet matches which rule in the chain, you can add these two following rules. After that, do **tail -f /var/log/kern.log** to see the messages. Remember to use the proper IP address and port number.

```
# iptables -t raw -A PREROUTING -p tcp --destination YOUR_IP/24 --dport PORT_NUMBER -
↳ j TRACE
# iptables -t raw -A OUTPUT -p tcp --destination YOUR_IP/24 --dport PORT_NUMBER -j_
↳ TRACE
```


I have yet to figure out where to put this information, which is why they are here, in the random chapter. These will be moved to different chapters in the future.

13.1 w command

The **w** command shows all the users, logged in to the computer. If you pass the *-f* flag, it toggles information about where each user is logged in from.

```
$ w
17:22:41 up 24 days, 11:37,  2 users,  load average: 0.56, 0.50, 0.59
USER      TTY      LOGIN@  IDLE   JCPU   PCPU WHAT
kdas     tty2      31May17 22days 3:07m  3:16  i3 -a --restart /run/user/1000/i3/
↪restart-state.28641
```

13.2 How long is the system running?

We have the **uptime** command which gives us information about how long the system is running. You can figure out the last time the system turned off or rebooted at a glance. For my laptop, it was 24 days ago.

```
$ uptime
17:31:30 up 24 days, 11:46,  2 users,  load average: 0.76, 0.98, 0.81
```

13.3 Finding CPU time of a command

The **time** command will help you to find the CPU time spent for any command. The following example will tell us how much time `du -sh` took to calculate the disk usage.

```
$ time du -sh
5.5G      .

real      0m1.026s
user      0m0.235s
sys       0m0.783s
```

13.4 dmesg command

The **dmesg** command prints out messages from the kernel buffer. Using this tool we can learn about the messages and information from the kernel drivers during and after the boot up process. This can be very handy when troubleshooting; for e.g. when the machine fails to boot or a certain piece of hardware does not function correctly.

13.5 Setting up cron jobs

One can schedule tasks using cron jobs. You can mention a certain time when a given task will be executed. In latest Fedora/CentOS, we use **crontab** package, in other systems we have **cron** or **anacron** package.

To view any existing jobs

```
crontab -l
```

To add a new cronjob or edit a previous one, use the command

```
crontab -e.
```

Format of a crontab file

```
* * * * * /path/to/command
+ + + + +
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | +----->   Day of the week (0-7)
| | | |
| | | +----->   Month of the year (1-12)
| | |
| | +----->   Day of the month (1-31)
| |
| +----->   Hour (0-23)
|
+----->   Minute (0,59)
```

Say we want to execute a shell script everyday at midnight.

```
0 0 * * * /usr/bin/myscript.sh
```

Another example can be executing the same script in every 15 minutes in every hour.

```
0,15,30,45 * * * * /usr/bin/myscript.sh
```

13.6 Finding out details about previous logins or system reboots

last command will give you the details about all the previous logins and shutdown/reboots. This command actually checks */var/tmp/wtmp* file for the logs.

The */var/log/btmp* file stores all the bad login details, and */var/log/utmp* file stores the details of the currently logged in users (**who** command reads this file).

You can read the *btmp* file using **last** command.

```
last -f /var/log/btmp
```

To know more, you can read the man page of *wtmp*.

CHAPTER 14

Whats next?

After you are familiar with the commands in this book, we would suggest you to learn shell scripting.

Start from <https://www.shellscript.sh> and then you can read the [beginners bash guide](#).

The [Lym Workbook](#) is an effort to create a small lab environment for the students to learn various commands from book. It uses [Vagrant](#) to create a new Virtual Machine (VM) on which you can practice commands, try out different solutions and solve these problem without having to worry about messing up your computer. We will slowly add more problems to it.

15.1 How to use this workbook?

15.1.1 Installation

You will need latest [Vagrant](#) for this. Install Vagrant following the steps from the website. On Windows you can use [VirtualBox](#) along with Vagrant.

15.1.2 Getting the code

Checkout latest workbook code from github.

```
git clone https://github.com/kushaldas/lymworkbook
cd lymworkbook
vagrant up
vagrant ssh workbook
```

The *vagrant up* command will create two vms.

Note: In case you managed to delete some configuration inside of the VM(s), you can very easily start from scratch. *vagrant destroy* will remove both the VMs, and *vagrant up* again will get them back. You can also destroy one particular VM, *vagrant destroy workbook*.

15.1.3 Setting up environment

Each problem here has a unique “string” which can be used to setup this new machine’s environment according to that problem’s specification, e.g., for setting up environment for a problem with string *unicorn*, you should do:

```
sudo lymsetup unicorn
```

After running the commands and making the changes mentioned in the problem statement, you should run the following to verify that the changes are as per the problem requirements:

```
sudo lymverify unicorn
```

15.2 copy paste

ID string: *copypaste*

- Create a directory called *work* in your home directory.
- Copy the file */tmp/problem1/work/files/hello.txt* into this newly created directory.
- Remember to remove the */tmp/problem1/work/files/hello.txt* file afterwards.
- Create a file named */tmp/chapter1/allusers* and add all of the directory names under your home directory into that file.

Note: To setup the problem environment, run:

```
sudo lymsetup copypaste
```

After performing the steps, verify the result using:

```
sudo lymverify copypaste
```

15.3 Find your user id

Find your user id and write it down in a file */tmp/myuserid.txt*.

15.4 Creating softlinks

To setup the problem environment:

```
sudo lymsetup softlinks
```

Create a softlink called *docs* in your home directory which will point to */usr/share/doc/* directory. Also create another softlink called *memory* to the */proc/meminfo* file.

To verify:

```
sudo lymverify softlinks
```

15.5 Basic vim usage

To setup the problem environment:

```
sudo lymsetup basicvim
```

Read the file at */etc/os-release* and write the value of `ID_LIKE` (without the double quotes) in a file at */tmp/id_like.txt*.

To verify:

```
sudo lymverify basicvim
```

15.6 Adding a new user

To setup the problem environment:

```
sudo lymsetup newuser
```

Add a new user called fatima to the system.

To verify:

```
sudo lymverify newuser
```

15.7 Deleting an existing user

To setup the problem environment, remember to add the user first from the previous problem.

```
sudo lymsetup deleteuser
```

Remove the fatima user from the system.

To verify:

```
sudo lymverify deleteuser
```

15.8 Finding the IP address of dgplug.org

Find the IP address of dgplug.org and save it to */tmp/ip_dgplug.txt* file.

To verify:

```
sudo lymverify findip
```

15.9 Change the local timezone of the system

Change the timezone of the system to the same of San Francisco, USA.

To verify:

```
sudo lymverify timezonechange
```

15.10 Add sudo access to an user

Grant administrative(sudo) privileges to an existing normal user account “lym”. Remember to create the user first.

To verify:

```
sudo lymverify assignsudo
```

CHAPTER 16

Advanced section

From this chapter onwards, we will learn more about different tools which people use for various. Most of these can be used for both personal use cases and also inside of big companies (depending on the situation).

To start, watch this [talk on failure](#).

CHAPTER 17

Containers

For now, just watch [this talk](#) from amazing Alice Goldfuss.

CHAPTER 18

Team

- Jason Braganza (Editor in command)
- Kushal Das (Adds typos in every form)

CHAPTER 19

Indices and tables

- `genindex`
- `search`

Symbols

/etc/profile, 27

A

apt, 50

B

bash_profile, 26

bashrc, 27

bootup, 37

C

chmod, 25

cmdline, 33

cpuinfo, 32

D

daemon, 37

dig, 67

dmesg, 82

dnf, 45

dnf install, 46

dns, 64

E

Environment variable, 18

export, 18

F

fhs, 11

File permission, 25

G

getenforce, 56

groupadd, 24

H

host, 67

I

id, 22

ip, 63

iptables, 76

J

journalctl, 40

K

kill, 30

L

last, 82

load average, 31

locate, 18

lsuf, 30

M

meminfo, 33

mount, 59, 60

N

NTFS, 60

P

passwd, 23

PATH, 26

ping, 63

proc filesystem, 32

ps, 29

R

resolvectl, 66

S

services, 38

setenforce, 56

signal, 30

ss, 70

ssh, 71
ssh-copy-id, 72
su, 17
sudo, 17, 23
sysctl, 34
systemctl, 38
systemd-resolved, 64

T

tar, 14
time, 81
timezones, 19
top, 31
tracepath, 71
traceroute, 70

U

umount, 60
updatedb, 18
uptime, 81
Use `*which*` command to see how `$PATH`
variable works, 27
useradd, 23
userdel, 24
usermod, 24

V

vim, 14

W

w, 81
which, 27