

# SysAdmin Notes for RHCSA

Somenath Sinha

August 2017

# Contents

<b>1</b>	<b>Using Essential Tools</b>	<b>7</b>
1.1	Man Command . . . . .	7
1.2	Understanding Globbing and Wildcards . . . . .	8
1.3	Understanding Globbing and Wildcards . . . . .	8
1.4	Understanding I/O Redirection and Pipes . . . . .	9
1.4.1	I/O Redirection . . . . .	9
1.4.2	Piping . . . . .	9
1.5	Using I/O Redirection and Piping . . . . .	9
<b>2</b>	<b>Essential File Management Tools</b>	<b>11</b>
2.1	Understanding Linux File System Layout . . . . .	11
2.2	Finding Files . . . . .	11
2.3	Understanding Links . . . . .	12
2.4	Working with Links . . . . .	12
2.5	Working with tar . . . . .	13
<b>3</b>	<b>Working with Text Files</b>	<b>15</b>
3.1	Understanding Regular Expressions . . . . .	15
3.2	Using common text tools . . . . .	15
3.2.1	cat . . . . .	15
3.2.2	less . . . . .	15
3.2.3	Head and Tail . . . . .	16
3.2.4	cut . . . . .	16
3.2.5	sort . . . . .	16
3.2.6	tr . . . . .	17
3.3	grep . . . . .	17

3.3.1	wc . . . . .	18
3.3.2	grep -l . . . . .	18
3.3.3	grep -i . . . . .	18
3.3.4	grep -R . . . . .	19
3.3.5	grep -v . . . . .	19
3.4	sed and awk basics . . . . .	19
3.4.1	sed . . . . .	19
3.4.2	awk . . . . .	20
<b>4</b>	<b>Connecting to a RHEL Server</b>	<b>22</b>
4.1	Connecting to a Server with SSH . . . . .	22
4.2	RSA Key fingerprint and known hosts . . . . .	22
4.3	sshd_config . . . . .	22
4.4	Understanding SSH keys . . . . .	23
4.4.1	Client authentication without password . . . . .	23
4.5	Using SSH Keys . . . . .	24
4.5.1	Copying SSH keys . . . . .	24
4.5.2	Copying files to a server securely using SSH . . . . .	25
<b>5</b>	<b>Managing Users and Groups</b>	<b>26</b>
5.1	Understanding the need for Users . . . . .	26
5.2	User Properties . . . . .	26
5.2.1	Username . . . . .	26
5.2.2	UID . . . . .	27
5.2.3	GID . . . . .	27
5.2.4	GECOS or comment field . . . . .	27
5.2.5	Home Directory . . . . .	27
5.2.6	Default Shell . . . . .	27
5.3	Creating and Managing Users . . . . .	27
5.3.1	Adding users . . . . .	27
5.4	Understanding Group Membership . . . . .	28
5.5	Creating and Managing Groups . . . . .	28
5.5.1	groupadd . . . . .	28

5.5.2	Adding users to a group . . . . .	28
5.6	User and Group configuration files . . . . .	29
5.7	Managing Password properties . . . . .	29
5.7.1	passwd . . . . .	29
5.7.2	chage . . . . .	30
<b>6</b>	<b>Connecting to a LDAP Server</b>	<b>31</b>
6.1	Understanding LDAP . . . . .	31
6.1.1	/bin/login . . . . .	31
6.1.2	ldd . . . . .	31
6.1.3	PAM config file syntax . . . . .	32
6.2	Telling your server where to find the LDAP Server . . . . .	33
6.2.1	nscd . . . . .	33
6.2.2	nss-pam-ldapd . . . . .	33
6.2.3	pam_ldap . . . . .	33
6.2.4	authconfig-gtk . . . . .	34
6.2.5	Switching to an LDAP user . . . . .	34
6.3	Understanding Automount . . . . .	34
6.3.1	Server selection for auto-mounting . . . . .	35
6.3.2	Samba server's CIFS protocol to automount . . . . .	35
6.4	Configuring Automount . . . . .	36
6.4.1	NFS Server Automounting . . . . .	37
6.5	Configuring NFS and Automount . . . . .	37
6.5.1	yum search . . . . .	37
6.5.2	Creating an NFS Server . . . . .	37
6.5.3	Starting the NFS server . . . . .	38
6.5.4	Automounting NFS . . . . .	38
6.6	Modifying nslcd Configuration . . . . .	39
6.6.1	Naming Services LDAP Client Daemon . . . . .	39
6.6.2	/etc/nslcd.conf . . . . .	40
<b>7</b>	<b>Managing Permissions</b>	<b>41</b>
7.1	Understanding Ownership: Users, Groups and Others . . . . .	41

7.1.1	Permissions . . . . .	41
7.1.2	Ownership . . . . .	42
7.2	Changing file ownership . . . . .	42
7.2.1	chgrp . . . . .	42
7.2.2	chown . . . . .	42
7.3	Understanding Basic Permissions . . . . .	43
7.4	Managing Basic Permissions . . . . .	43
7.4.1	chmod . . . . .	43
7.5	Understanding Special Permissions . . . . .	44
7.6	Managing Special Permissions . . . . .	45
7.6.1	Finding a file with a particular set of permissions . . . . .	45
7.6.2	Setting Group ID for a directory . . . . .	46
7.6.3	Sticky Bit . . . . .	47
7.6.4	Lowercase 's' or 't' vs Uppercase in permissions . . . . .	48
7.7	Understanding ACLs . . . . .	48
7.7.1	Mount options . . . . .	48
7.7.2	Commands . . . . .	49
7.8	Managing ACLs . . . . .	49
7.8.1	history . . . . .	51
<b>8</b>	<b>Configuring Networking</b>	<b>52</b>
8.1	Understanding NIC Naming . . . . .	52
8.1.1	Network Device Naming Schemes . . . . .	52
8.2	Managing NIC Configuration with ip Command . . . . .	52
8.2.1	show commands . . . . .	53
8.2.2	ip addr add . . . . .	53
8.2.3	ip route add . . . . .	54
8.3	Storing Network Configuration persistently . . . . .	54
8.3.1	Hostname . . . . .	55
8.4	Understanding Network Manager . . . . .	55
8.5	Using Network Manager utilities (nmcli, nmtui) . . . . .	56
8.5.1	nmcli . . . . .	56

8.5.2	nmtui . . . . .	57
8.6	Understanding Routing and DNS . . . . .	57
8.6.1	Default route . . . . .	57
8.6.2	DNS . . . . .	58
8.7	Configuring Routing and DNS . . . . .	58
8.8	Understanding Network Analysis Tools . . . . .	59
8.9	Using Network Analysis Tools . . . . .	59
8.9.1	ping . . . . .	60
8.9.2	traceroute . . . . .	60
8.9.3	host . . . . .	60
8.9.4	dig . . . . .	61
8.9.5	Physical network problems . . . . .	62
<b>9</b>	<b>Managing Processes</b>	<b>65</b>
9.1	Understanding Jobs and Processes . . . . .	65
9.1.1	jobs . . . . .	65
9.2	Managing Shell Jobs . . . . .	65
9.3	Getting process information with ps . . . . .	67
9.3.1	Getting PID of a process . . . . .	68
9.3.2	Seeing Parent and Child process relation . . . . .	68
9.4	Understanding Memory Usage . . . . .	68
9.5	Understanding Performance Load . . . . .	69
9.5.1	uptime Command . . . . .	69
9.6	Monitoring System Activity with top . . . . .	70
9.7	Sending Signals to processes . . . . .	71
9.7.1	kill command . . . . .	71
9.8	Understanding Priorities and Niceness . . . . .	72
9.9	Changing Process Nice values . . . . .	72
9.9.1	Changing niceness from top . . . . .	72
9.9.2	Changing Niceness from command line . . . . .	72

## **Part I**

# **Installing RHEL Server**

# Chapter 1

## Using Essential Tools

### 1.1 Man Command

**man** followed by *keyword* yields the manual page of that command.

---

```
1 $ man ls
```

---

**man** followed by option **-k** (for keyword) and then followed by a *keyword* yields a list of all the commands containing that keyword and a brief description of that command.

---

```
1 $ man -k day
2 daylight (3)           - initialize time conversion information
3 dysize (3)            - get number of days for a given year
4 daylight (3p)         - set timezone conversion information
5 gettimeofday (2)      - get / set time
6 gettimeofday (3p)     - get the date and time
7 motd (5)              - message of the day
8 Net::Time (3pm)       - time and daytime network client interface
9 settimeofday (2)      - get / set time
10 Time::HiRes (3pm)     - High resolution alarm, sleep, gettimeofday, interval timers
```

---

The numbers next to the commands indicate which section of the man pages the command belongs to (based on their functionality). The actual section that the commands belong to can be determined by the use of

---

```
1 $ man man-pages
```

---

The relevant sections for SysAdmins are Section 1, 5 & 8. The sections are:



Section Number	Deals with	Description
1	Commands (Programs)	Those commands that can be executed by the user from within a shell.
2	System calls	Those functions which must be performed by the kernel.
3	Library calls	Most of the libc functions.
4	Special files (devices)	Files found in /dev.
5	File formats and conventions	The format for /etc/passwd and other human-readable files.
6	Games	
7	Overview, conventions, and miscellaneous	Overviews of various topics, conventions and protocols, character set standards, and miscellaneous other things.
8	System management commands	Commands like mount(8), many of which only root can execute.

To filter down the output of the **man -k** command, we can use **grep** to obtain only the relevant parts of the result on the basis of the appropriate section number in the man-pages.

This can be achieved using the pipe which feeds the output of the first command to the input of the second command.

```

1 $ man -k day | grep 3
2 daylight (3)          - initialize time conversion information
3 dysize (3)           - get number of days for a given year
4 daylight (3p)        - set timezone conversion information
5 gettimeofday (3p)    - get the date and time
6 Net::Time (3pm)      - time and daytime network client interface
7 Time::HiRes (3pm)    - High resolution alarm, sleep, gettimeofday, interval timers

```

## 1.2 Understanding Globbing and Wildcards

- \* - Indicates any string.
- ? - Indicates any single character.
- [...] - Indicates any character provided within brackets.
- [!...] - Indicates any character *NOT* provided within brackets.
- [a-f] - Indicates any character provided within the range of a to f.

## 1.3 Understanding Globbing and Wildcards

- \$ ls a\* - Lists all files and folders (including contents of each folder) that start with "a"
- \$ ls \*a\* - Lists all files and folders that contain the string "a".
- \$ ls -d a\* - Shows all files and folders that start with "a" but excludes the contents of each individual folder.
- \$ ls ??st\* - Lists all files and folders that have "st" as the 3<sup>rd</sup> and the 4<sup>th</sup> character in their name.
- \$ [a-f] - Indicates any character provided within the range of a to f.

## 1.4 Understanding I/O Redirection and Pipes

### 1.4.1 I/O Redirection

File Descriptors:

<b>STDIN</b>	-	0	-	Standard Input	-	Represents the "file" for the Standard Input Device (generally Keyboard).
<b>STDOUT</b>	-	1	-	Standard Output	-	Represents the "file" for the Standard Output Device (generally the Monitor).
<b>STDERR</b>	-	2	-	Standard Error	-	Represents the "file" for the Standard Output Device (also, generally the Monitor).

Redirection:

<b>STDIN</b>	-	<	-	Feeds the file to the right of the "<" as input to the command on the left.
<b>STDOUT</b>	-	>	-	Stores the output of the command to the left of the ">" to the file indicated on the right. <i>OVERWRITES</i> the mentioned file.
<b>STDOUT</b>	-	>>	-	Stores the output of the command to the left of the ">>" to the file indicated on the right. <i>APPENDS</i> the mentioned file.
<b>STDERR</b>	-	2 >	-	Redirects the errors from the command mentioned on the left to the file on the right. <i>OVERWRITES</i> the mentioned file.

---

```
1 $ mail -s hi root < .
2 $ ls > myFile
3 $ ls -lh >> myFile
4 $ grep hi * 2> /dev/tty6
```

---

1 - *mail* is a simple command used to send messages. The command expects the message to terminate with a ".", so we feed it directly to the command, instead of providing any input.

4 - The STDERR is redirected to tty6 (a virtual terminal connected to the host). Can also be diverted to a file if needed, such as an errorLog.

### 1.4.2 Piping

The command `$ ps aux` shows us the overview of all the running processes on the host. However, it's too long to view all at once. In such situations, or wherever we need to feed the output of the first command to the input of the second command, we use the pipe operator. The command would then be `$ ps aux | less`.

The difference in the usage of the piping and redirection operators is that Pipe is used to pass output to another program or utility, while Redirect is used to pass output to either a file or stream.

## 1.5 Using I/O Redirection and Piping

---

```
1 $ ps aux | awk '{print $2}'
2 $ ps aux | awk '{print $2}' | sort
3 $ ps aux | awk '{print $2}' | sort -n
```

---

---

The second column (\$2) of the `$ ps aux` command contains the Process ID (PID), and if we only want to filter the output such that only the PID is shown, we simply use the **awk** filtering utility.

If we want to sort the output of the command, we use the **sort** utility, but it generally sorts as a string. To sort the output as a number, we use the option **sort -n**.

If you expect lots of errors for a particular command, but want to discard all errors and only see the output when successful, then simply redirect the STDERR to `/dev/null`, which is a special device that discards all data written to it, i.e., a dustbin for data.

---

```
1 $ find / -name "*.rpm"
2 $ find / -name "*.rpm" 2> /dev/null
```

---

The first command shows all output including errors, but the second command discards all errors and shows the rest.

---

```
1 $ some_command > /dev/null 2> &1
```

---

The above code redirects STDOUT to `/dev/null` thus destroying the output, and also redirects the STDERR (2>) to STDOUT (&1). Essentially, it discards all output - useful when we don't need the output but only need the command to execute.

---

```
1 $ ls / > file_list.txt
2 $ sort < file_list.txt > file_list_sorted.txt
```

---

The above command stores the contents of the root directory in *file\_list.txt*. Then, the second command uses both input and output redirection! The input of the sort command is fed from *file\_list.txt* and the corresponding output sent to *file\_list\_sorted.txt*.

## Chapter 2

# Essential File Management Tools

### 2.1 Understanding Linux File System Layout

---

<b>root (/)</b>	-	Contains all other directories.
-----------------	---	---------------------------------

---

<b>/boot</b>	-	Contains everything the system needs to start up
<b>/usr</b>	-	Contains program files
<b>/etc</b>	-	Contains configuration files
<b>/home</b>	-	Contains a user's files
<b>/mnt</b>	-	Used to manually mount devices
<b>/media</b>	-	Devices like optical discs get auto-mounted on the media directory

---

Unlike other OSs, the linux files system is designed as such that multiple devices can be mounted on the same file system hierarchy. Thus, it's possible to mount devices remotely as well!

### 2.2 Finding Files

The **find** command is used to find a file within a folder and its subdirectories. When the starting point of the search is the root directory (/) then find will search the entire file system. While the utility is extremely thorough, this may cause delays due to remote devices on the network mounted on the file system.

---

```
1 $ find / -name "passwd"
```

---

If you're trying to find the location of a binary file, a better command would be **which** command, as it directly shows the location of the binary, but be careful as it only works with binaries.

---

```
1 $ which passwd
2 /usr/bin/passwd
```

---

Contrastingly, the command **whereis** not only gives us the locaiton of the binary, but the location of the complete environment of the binary!

---

```
1 $ whereis passwd
2 passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1.gz
   ↪ /usr/share/man/man5/passwd.5.gz
```

---

Another similar utility is called **locate** which shows all files that have the string provided to it in its name. Note, however, that **locate** operates on a database, that must be updated (especially after the creation of a new file) to show relevant results.

---

```
1 # touch sinha
2 # ls
3 sinha
4 # locate sinha
5 /usr/share/vim/vim74/keymap/sinhala-phonetic_utf-8.vim
6 /usr/share/vim/vim74/keymap/sinhala.vim
7 # updatedb
8 # locate sinha
9 /home/somu/Documents/sinha
10 /usr/share/vim/vim74/keymap/sinhala-phonetic_utf-8.vim
11 /usr/share/vim/vim74/keymap/sinhala.vim
```

---

## 2.3 Understanding Links

**inode** - An inode is a datastructure that describes a file system object such as a file or a directory, containing both the disc block locations as well as the attributes of the file system object. The inodes are identified by their inode number.

Consequently, for us to access the files/directories, we need to be able to provide a name to the inodes, which are called hardlinks. A file may have more than one hardlink. Note that each hardlink is simply a different name provided to the same inode. Thus, all hardlinks to the same file/directory have the same inode number. Hardlinks are one-directional only, i.e., the hardlink itself knows which inode it points to, but the inodes only know the total number of hardlinks that are associated with it, and not which exact ones are pointing to it. Since hardlinks point to some inode, they always need to stay on the same partition as the inode.

A symbolic link on the other hand, points to a hardlink instead of an inode. As such, it has a different inode number than the one that the hardlink points to. Thus, the hardlink and symbolic link can be on different partitions as well. It can even exist across servers. Whenever a hardlink is deleted, however, all the symbolic links pointing to it are rendered invalid.

## 2.4 Working with Links

The **ln** command is used to create both hardlinks and symbolic links. To create a symbolic link, we need only add the **-s** option. The **-i** option of the **ls** command shows us the inode number.

---

```
1 # ln /etc/hosts computers
2 # ls -il /etc/hosts computers
```

```

3  8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 computers
4  8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 /etc/hosts
5  # ln -s computers newcomputers
6  # ls -il /etc/hosts computers newcomputers
7  8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 computers
8  8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 /etc/hosts
9  27604468 lrwxrwxrwx. 1 root root   9 Sep  7 19:26 newcomputers -> computers
10 # rm -f computers
11 # ls -il /etc/hosts newcomputers
12 8388733 -rw-r--r--. 1 root root 158 Jun  7  2013 /etc/hosts
13 27604468 lrwxrwxrwx. 1 root root   9 Sep  7 19:26 newcomputers -> computers
14 # exit
15 exit
16 $ ln /etc/shadow mydata
17 ln: failed to create hard link 'mydata' => '/etc/shadow': Operation not permitted
18 $ ls -l /etc/shadow
19 -----. 1 root root 1375 Sep  5 21:04 /etc/shadow

```

---

When the hardlink *computers* to the inode associated with */etc/hosts* is deleted, the associated symbolic link of *newcomputers* becomes invalid.

Finally, RHEL 7 onwards, a user may only create a link to a file/directory that it at least has a read permission to. Thus, any user won't be able to create a link to */etc/shadow* as it has no permissions for anybody.

## 2.5 Working with tar

**tar** stands for Tape Archive. The command is most commonly used to make backups of files by storing them in archives. Some of the options of tar are:

- |           |                 |   |
|-----------|-----------------|---|
| <b>-c</b> | - create        | - typically has an extension of .tar                                  |
| <b>-t</b> | - show contents | - show contents of the archive.                                       |
| <b>-x</b> | - extract       |   |
| <b>-z</b> | - file          | - compress the archive using gzip. Typically has an extension of .tgz |
| <b>-v</b> | - verbose       | - tell us what the utility is doing.                                  |
| <b>-f</b> | - file          | - option to indicate the name of the archive file.                    |
| <b>-C</b> | - location      | - indicates where the archive is to be extracted.                     |

---

```

1 $ tar -cvf /root/etc.tar /etc

```

---

The above command creates the *etc.tar* archive in the */root* directory and puts the contents of */etc* in that archive. Note that the file *etc.tar* has a *.tar* extension only because we provided it, and not because Linux mandates it (unlike windows). Thus, sometimes we may run across tar archives that don't have an extension and are hard to detect. So, in that case we use the file command, which tells us the type of a particular file.

---

```

1 $ file /root/etc.tar
2 /root/etc.tar: POSIX tar archive (GNU)

```

---

Note that the *.tar* archive only puts all the files of the */etc* directory in the file *tar.etc*, but doesn't actually compress anything. To enable compression the *-z* option of the *tar* command must be used.

---

```
1 $ tar -czf /root/etc2.tgz /etc
```

---

Before extracting the contents of a tar file, we might want to see its contents, which can be done using the `-t` option of the `tar` command. *NOTE: Some older versions of `tar` may require the `-z` option to enable working with gzip archives, even when simply using the archive and not creating it.*

---

```
1 $ tar -tvf /root/etc2.tgz
```

---

To actually extract the archive, we use `-x` option. To indicate the location where we want the extracted files to reside, we include the `-C` option. If this option is not present then the files will be extracted in the present directory.

---

```
1 $ tar -xvf /root/etc2.tgz -C /tmp
```

---

To extract only one file from the archive, we can simply provide the name of the file at the very end.

---

```
1 $ tar -xvf /root/etc2.tgz -C / etc/wgetrc
```

---

*NOTE that in the above command, we use the relative path `etc/wgetrc` because of the fact that the archive stores a relative file path for easy extraction in any folder.*

## Chapter 3

# Working with Text Files

### 3.1 Understanding Regular Expressions

Character	Definition	Example	Result
^	Start of a string	^abc	abc, abcdef, abc123
\$	End of a string	abc\$	abc, blahabc, 456abc
.	Any character except newline	a.c	abc, aac, a2c
	Alteration	1 8	1,8
{...}	Explicit quantity of preceding character	ab{2}c	abbc
[...]	Explicit set of characters to match	a[bB]c	abc, aBc
(...)	Group of characters	(123){3}	123123123
*	Null or more of the preceding character	ab*c	ac, abc, abbbbc
+	One or more of the preceding character	ab+c	abc, abbbbc
?	Null or one of the preceding character	ab?c	ac, abc

PEARSON  
IT CERTIFICATION

livelessons®  
©2015 Pearson, Inc.

Figure 3.1: RegEx Cheat Sheet

### 3.2 Using common text tools

#### 3.2.1 cat

The `cat` command prints the entire content of a file on to the terminal.

#### 3.2.2 less

Sometimes the `cat` command is unsuitable, like in the case of extremely large files. In such cases, like the `/var/log/messages`, the default system log file, using `cat` won't work as the majority of the messages would scroll past fast. For such cases, `less` is a better utility. Search functionality is exactly the same as in the case of `vim`.



### 3.2.3 Head and Tail

#### Head

The `head` command by default shows us the first 10 lines of a text file. To see more or less lines, the `-n` option can be used.

---

```
1 $ head -n 20 file.txt
```

---

#### Tail

The `tail` command by default shows us the last 10 lines of a text file. To see more or less lines, the `-n` option can be used.

---

```
1 $ tail -n 5 file.txt
```

---

#### Combination of head and tail

The combination of these two commands can enable the viewing of text in between specific line numbers. The command below shows lines 16-20 of `file.txt`

---

```
1 $ head -n 20 file.txt | tail -n 5
```

---

### 3.2.4 cut

With the `cut` utility, we can print out a specific column from a text file. It assumes the columns are separated by Tabs. Which specific column is to be printed is set by using the `-f` option. For example, to only print the first column of a text file, we say:

---

```
1 $ cut -f 1 cities
```

---

To provide a different delimiter, such as ":" we use the `-d` option followed by the delimiter of our choice.

---

```
1 $ cut -f 1 -d : /etc/passwd
```

---

### 3.2.5 sort

This command sorts the input provided in the order of the ASCII table. That means numbers first, capital letters next and finally the lower case letters.

---

```
1 $ cut -f 1 -d : /etc/passwd | sort
```

---

To sort on the basis of a specific criteria:

- n - Sort on the basis of actual numerical value, instead of treating a number as a string.
- f - Sort in a case insensitive manner.

### 3.2.6 tr

The `tr` command replaces certain characters with certain other characters. Thus, it's frequently used in conjunction with pipes to modify the output of a command.

---

```
1 $ echo hello | tr a-z A-Z
2 HELLO
3 $ echo hello | tr [:lower:] [:upper:]
4 HELLO
```

---

## 3.3 grep

`grep` is a filtering utility that only prints those lines that contain a certain expression matching the pattern provided by a *RegEx*.

---

```
1 $ ps aux | grep tracker
2 somu      10450  0.0  0.4 469796  9000 ?        SNl  10:06   0:00
   ↪ /usr/libexec/tracker-miner-user-guides
3 somu      10465  0.0  0.6 536856 12012 ?        Sl   10:06   0:00
   ↪ /usr/libexec/tracker-store
4 somu      10611  0.0  0.7 779816 13108 ?        SNl  10:06   0:00
   ↪ /usr/libexec/tracker-extract
5 somu      10614  0.0  0.5 469800  9632 ?        SNl  10:06   0:00
   ↪ /usr/libexec/tracker-miner-apps
6 somu      10615  0.0  0.7 710160 13204 ?        SNl  10:06   0:00
   ↪ /usr/libexec/tracker-miner-fs
7 root      17396  0.0  0.0 112644   968 pts/0    R+   13:49   0:00 grep --color=auto
   ↪ tracker
```

---

Another use for `grep` is searching files. The syntax is `grep <filename-pattern> <search-directory>`.

To avoid errors notifying "is a directory", simply redirect errors to `/dev/null`.

---

```
1 $ grep lisa * 2> /dev/null
2 group:lisa:x:1001:
3 gshadow:lisa:::
4 passwd:lisa:x:1001:1001::/home/lisa:/bin/bash
5 passwd-:lisa:x:1001:1001::/home/lisa:/bin/bash
```

---

---

```

6  services:na-localise      5062/tcp                # Localisation access
7  services:na-localise      5062/udp                # Localisation access
8  shadow:lisa:$6$01/zSJkh$xjJNYNnj1rPs7Fq0hDwt8VucS0nLL82XrMYpmBnLF2DrzB2npFvCwxM9MJEHgCHCwvabCgEA17LK2aU0h9FIT/:1741
9  shadow-:lisa:password:17414:0:99999:7:::

```

---

### 3.3.1 wc

Counts the number of words, lines and characters.

- l** - Counts the number of lines
- w** - Counts the number of words
- m** - Counts the number of characters
- c** - Counts the number of bytes
- L** - Counts the length of the longest line

To see the number of matched lines using `grep`, simply use:

---

```

1  $ ps aux | wc
2  188      2344    19581

```

---

### 3.3.2 grep -l

`Grep` by default returns the name of the matching file followed by the matching lines. This output can be made more readable by `grep -l` which lists all the files in the directory that matches the criteria.

---

```

1  $ grep lisa * 2> /dev/null
2  group:lisa:x:1001:
3  passwd:lisa:x:1001:1001::/home/lisa:/bin/bash
4  services:na-localise      5062/tcp                # Localisation access
5  services:na-localise      5062/udp                # Localisation access
6  $ grep -l lisa * 2> /dev/null
7  group
8  passwd
9  services

```

---

### 3.3.3 grep -i

The `-i` flag turns the `grep` command case-insensitive!

---

```

1  $ grep lisa * 2> /dev/null
2  group:lisa:x:1001:
3  passwd:lisa:x:1001:1001::/home/lisa:/bin/bash
4  services:na-localise      5062/tcp                # Localisation access
5  services:na-localise      5062/udp                # Localisation access
6  $ grep -i lisa * 2> /dev/null
7  group:lisa:x:1001:

```

---

---

```

8 passwd:lisa:x:1001:1001:./home/lisa:/bin/bash
9 services:ltctcp          3487/tcp          # LISA TCP Transfer Channel
10 services:ltcudp         3487/udp          # LISA UDP Transfer Channel
11 services:na-localise    5062/tcp          # Localisation access
12 services:na-localise    5062/udp          # Localisation access

```

---

### 3.3.4 grep -R

The usage of the `-R` flag puts `grep` in recursive mode, where the utility searches for the file in each subfolder as well.

---

```

1 $ grep -iR lisa * 2> /dev/null
2 group:lisa:x:1001:
3 lvm/lvm.conf:      # If using external locking (type 2) and initialisation fails, with
4 passwd:lisa:x:1001:1001:./home/lisa:/bin/bash
5 Binary file pki/ca-trust/extracted/java/cacerts matches
6 Binary file pki/java/cacerts matches
7 ...

```

---

### 3.3.5 grep -v

`Grep` with a `-v` flag excludes the matching results. Here we can exclude the lines containing "Binary" using:

---

```

1 $ grep -iR lisa * 2> /dev/null | grep -v Binary
2 alternatives/jre_openjdk/lib/security/nss.cfg:handleStartupErrors =
3 ↪ ignoreMultipleInitialisation
4 alternatives/jre_openjdk_exports/lib/security/nss.cfg:handleStartupErrors =
5 ↪ ignoreMultipleInitialisation
6 brltty/fr-abrege.ctb:word civilisation      14-1236-16
7 brltty/fr-abrege.ctb:word civilisations     14-1236-16-234
8 brltty/fr-abrege.ctb:word généralisation    1245-1345-16
9 brltty/latex-access.ctb:  brailleTranslator.capitalisation = "6dot"
10 brltty/latex-access.ctb:
11 passwd:lisa:x:1001:1001:./home/lisa:/bin/bash
12 sane.d/canon_pp.conf:# Set a default initialisation mode for each port.  Valid modes are:
13 services:ltctcp          3487/tcp          # LISA TCP Transfer Channel
14 services:ltcudp         3487/udp          # LISA UDP Transfer Channel
15 ...

```

---

## 3.4 sed and awk basics

### 3.4.1 sed

`sed` is an old utility that's used to process text. Many of its functionalities can now be done using `grep` itself.

## sed q

To see the first two lines of a file using sed we use:

---

```
1 $ sed 2q /etc/passwd
2 root:x:0:0:root:/root:/bin/bash
3 bin:x:1:1:bin:/bin:/sbin/nologin
```

---

## sed -n

The `-n` flag makes sed print no output unless the `p` flag is also provided. Here, we use a Regular Expression `"root"` to match only a certain part of the text and then the `p` flag to print only if that criteria is matched.

---

```
1 $ sed -n /^root/p /etc/passwd
2 root:x:0:0:root:/root:/bin/bash
```

---

The above result could also be obtained with `grep "^root" /etc/passwd`.

## Substitution with sed

Sed can be used to substitute text within a file using the `s` parameter. It's used as :

---

```
1 $ cat names
2 Somu
3 Arpi
4 Neha
5 Santy
6 Debu
7 $ sed -i 's/Santy/Dickwad/g' names
8 $ cat names
9 Somu
10 Arpi
11 Neha
12 Dickwad
13 Debu
```

---

The `-i` flag asks the modifications to be made in place. Otherwise the output (changed text) would've simply been displayed to the screen and then need to be redirected to a file for storage.

## 3.4.2 awk

awk is another utility that is especially useful when working with text files. It excels at operations like cutting out information.

## Cutting out information using awk

For certain operations, the `awk` command is a lot more powerful than the `cut` utility. In the example below `cut` has a hard time recognizing the second field, while `awk` has no problem whatsoever!

---

```
1 $ ps aux | grep 'gdm'
2 root      1117  0.0  0.1 480248  4688 ?          Ssl  09:20   0:00 /usr/sbin/gdm
3 root      1333  0.8  1.2 328524 47220 tty1      Ssl+ 09:20   0:43 /usr/bin/X :0
   ↪ -background none -noreset -audit 4 -verbose -auth
   ↪ /run/gdm/auth-for-gdm-bgrBZH/database -seat seat0 -nolisten tcp vt1
4 root      1718  0.0  0.1 528944  5848 ?          Sl    09:20   0:00 gdm-session-worker
   ↪ [pam/gdm-password]
5 gdm       1737  0.0  0.1 458088  4152 ?          Sl    09:20   0:00 ibus-daemon --xim
   ↪ --panel disable
6 gdm       1741  0.0  0.1 373560  5424 ?          Sl    09:20   0:00 /usr/libexec/ibus-dconf
7 gdm       1745  0.0  0.2 438152  7772 ?          Sl    09:20   0:00 /usr/libexec/ibus-x11
   ↪ --kill-daemon
8 somu     12218  0.0  0.0 112664   972 pts/0    R+   10:47   0:00 grep --color=auto gdm
9 $ ps aux | grep 'gdm' | cut -f 2
10 root      1117  0.0  0.1 480248  4688 ?          Ssl  09:20   0:00 /usr/sbin/gdm
11 root      1333  0.8  1.2 328524 47220 tty1      Ssl+ 09:20   0:43 /usr/bin/X :0
   ↪ -background none -noreset -audit 4 -verbose -auth
   ↪ /run/gdm/auth-for-gdm-bgrBZH/database -seat seat0 -nolisten tcp vt1
12 root      1718  0.0  0.1 528944  5848 ?          Sl    09:20   0:00 gdm-session-worker
   ↪ [pam/gdm-password]
13 gdm       1737  0.0  0.1 458088  4152 ?          Sl    09:20   0:00 ibus-daemon --xim
   ↪ --panel disable
14 gdm       1741  0.0  0.1 373560  5424 ?          Sl    09:20   0:00 /usr/libexec/ibus-dconf
15 gdm       1745  0.0  0.2 438152  7772 ?          Sl    09:20   0:00 /usr/libexec/ibus-x11
   ↪ --kill-daemon
16 somu     12226  0.0  0.0 112664   968 pts/0    R+   10:48   0:00 grep --color=auto gdm
17 $ ps aux | grep 'gdm' | awk '{ print $2 }'
18 1117
19 1333
20 1718
21 1737
22 1741
23 1745
24 12248
```

---

## Chapter 4

# Connecting to a RHEL Server

### 4.1 Connecting to a Server with SSH

To connect to a server we use the `ssh` command. The Syntax is: `ssh <server-ip>`.

---

```
1 $ ssh 192.168.152.129
2 somu@192.168.152.129's password:
3 Last login: Mon Nov 13 12:37:26 2017 from 192.168.152.128
```

---

The default SSH port is **22**. To connect to SSH on a different port (common when server is exposed to the internet), is `ssh -p <port-number> <server-ip>`. Note that *if root login is disabled on the server*, we must also provide the username to login as. The syntax then becomes : `ssh -p <port-number> <username>@<server-ip>`.

---

```
1 $ ssh -p 2022 sander@ldap.rhatcertification.com
```

---

### 4.2 RSA Key fingerprint and known hosts

Upon each new connection the `ssh` daemon shows us the RSA key fingerprint of the host to verify if we're connecting to the right computer. If so, the host is added to a list of known hosts permanently, in `~/.ssh/known_hosts`.

When the key fingerprint of the server doesn't match the Key fingerprint on record, the system warns us from connecting! This may occur when the server has been reinstalled on the same IP address. Thus, the new key fingerprint won't match the old one. To fix this, simply remove the old entry from `~/.ssh/known_hosts`.

### 4.3 `sshd_config`

The details of the method of connection to a server is stored in the **`sshd_config`** file, located in `/etc/ssh/sshd_config`.

Some of the options are:

Option	Description	Default Value
Port	The port number which is used for SSH on that server	22
PermitRootLogin	Whether an user is allowed to login as <i>root</i> user via SSH	yes

If the root login on the server via SSH is disabled, it generally makes the server a little bit more secure!

## 4.4 Understanding SSH keys

To initiate the ssh connection, the **SSHD** service on the server is contacted by the client. In order to confirm it's identity, the server responds with it's own `/etc/ssh/ssh_host.pub` public key to the client. When the client's user has verified the key of the server, the public key fingerprint gets stored in the clients `~/.ssh/known_hosts` file. Finally, the user is asked for the password to log on to the server.

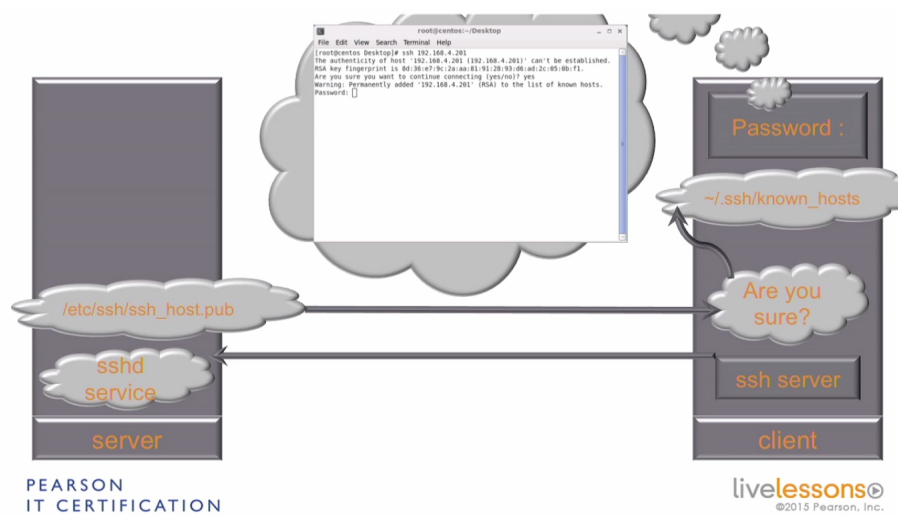


Figure 4.1: Server Authentication procedure

### 4.4.1 Client authentication without password

The client can also prove it's identity without a password by the use of a public key that it provides to the server. The private key of the user is stored in the home directory of the user `~/.ssh/id_rsa`. A packet encrypted with the private key is sent to the server which knows the user's public key. Some complex calculations based on this is performed on the authentication token sent from the client and if the identity is confirmed, then the user is logged in without needing a password.



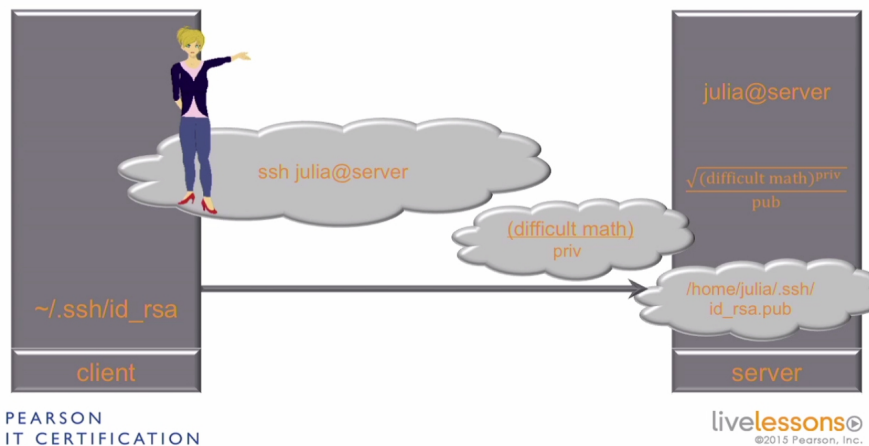


Figure 4.2: Public Key Client Authentication without password.

## 4.5 Using SSH Keys

SSH keys can be used to authenticate an user instead of a password. The private-public key pair can be generated using the `ssh-keygen` utility.

```

1  $ ssh-keygen
2  Generating public/private rsa key pair.
3  Enter file in which to save the key (/home/somu/.ssh/id_rsa):
4  Enter passphrase (empty for no passphrase):
5  Enter same passphrase again:
6  Your identification has been saved in /home/somu/.ssh/id_rsa.
7  Your public key has been saved in /home/somu/.ssh/id_rsa.pub.
8  The key fingerprint is:
9  SHA256:0C5uEHnAgJvzFEc0ulfl1YSyT/YF5Utl9lweTfPDac somu@vmPrime.somusysadmin.com
10 The key's randomart image is:
11 +---[RSA 2048]---+
12 | ..==.   ooo .E. |
13 | . .+++.oo+ + o.o|
14 | o.o+++.o..B . *o|
15 |+ ...===. o o  +|
16 | +. o +oS .      |
17 | .. o .          |
18 |   o             |
19 |   ,             |
20 |                 |
21 +---[SHA256]---+

```

### 4.5.1 Copying SSH keys

The SSH keys generated on the client now have to be copied to the server which requires authentication. This can be done using `ssh-copy-id`.

---

```
1 $ ssh-copy-id 192.168.152.129
2 /usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any
   ↳ that are already installed
3 /usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it
   ↳ is to install the new keys
4 somu@192.168.152.129's password:
5
6 Number of key(s) added: 1
7
8 Now try logging into the machine, with:  "ssh '192.168.152.129'"
9 and check to make sure that only the key(s) you wanted were added.
10
11 $ ssh 192.168.152.129
12 Last login: Wed Nov 15 11:59:03 2017 from 192.168.152.128
```

---

While previous versions of `ssh-copy-id` didn't support specifying a port number, RHEL 7 onwards this feature is supported.

---

```
1 $ ssh-copy-id -p 2022 sander@ldap.rhatcertification.com
```

---

In case the public key is not recognized, it is possible to specify the public key using the `ssh-copy-id -i <publicKeyFile.pub>` flag.

## 4.5.2 Copying files to a server securely using SSH

Files can be copied to a server using SSH connection using the `scp` (secure copy) utility.

---

```
1 $ scp -P 22 names 192.168.152.129:~
2 names                100%  28      8.1KB/s   00:00
```

---

**NOTE** that the directory on which the file has to be copied to on the server, (in this case the directory) has to be specified for the copy to be successful. Otherwise, `scp` just creates a local copy of the file with the name of the server as the filename.

Also, if the port has to be specified, the flag is `-P` which is in capital unlike `ssh` and `ssh-copy-id`.

## Chapter 5

# Managing Users and Groups

### 5.1 Understanding the need for Users

User accounts are not just to ensure that different people use resources with accountability and resource management. Several processes also have to execute with permissions given to them by their respective user accounts.

For example, the apache web server's processes and services execute under the permissions given to the apache account. This account doesn't have root privileges, which ensures that in case of security breaches of the apache user account, the culprit doesn't gain access to any critical resources that only an administrator or the root account should have access to.

### 5.2 User Properties

#### 5.2.1 Username

A typical user info in the `/etc/passwd` file consists of the login information of several users, each with the following details :- `somu:x:1000:1000:Somu:/home/somu:/bin/bash`. Here, Somu is the username, the `x` in the second field references that a password has been stored for that username in the `/etc/shadow` file. The file contains the (one-way) encrypted password as well as several password related information such as password expiration dates, etc. Since the `/etc/shadow` file is only readable by the root user, it minimizes the security risk. Generally, only real user accounts need a password and system users (accounts used by processes to execute) don't.

#### `/etc/shadow`

While the `/etc/shadow` file contains the password of an user in an encrypted format, if the user account is new and doesn't yet have any password assigned to it, then the entry for it in `/etc/shadow` looks like:

---

```
1 $ cat /etc/shadow | grep lisa
2 lisa:!!:17485:0:99999:7:::
```

---

The second entry (!) is where the encrypted password is usually stored. The double exclamation indicates that the *lisa* account hasn't set up a password yet.

### 5.2.2 UID

Each user on the system is setup with a unique UserID (UID). The root has a UID of 0, and normal users start with an UID of 1000 onwards. There are a total of 64,000 UIDs available for 2.4 kernels, and 4 billion for 2.6 kernels.

### 5.2.3 GID

On Linux, every user must be the member of at least one group, which is known as the **primary group** of the user, stored in the `/etc/passwd` file. On RHEL, that primary group has the same name as the username and the user is the only member of that group by default (i.e., private group). The list of Groups is stored in a file called `/etc/group`.

### 5.2.4 GECOS or comment field

This is a comment field that can contain anything the admin deems necessary. It generally contains information that makes the identification of each user easier.

### 5.2.5 Home Directory

The home directory refers to the location where the user is allowed to store files. For services, this folder is important because it defines the directory where the service can read and write files. While for regular users the home directory is typically inside `/home`, for services, they can be anywhere.

### 5.2.6 Default Shell

This is the shell (or command) that is executed on login of the user. The default value is `/bin/bash`.

## 5.3 Creating and Managing Users

### 5.3.1 Adding users

The default command for adding users on RHEL is `useradd`.

Option	Description
<b>-e</b>	Expiration date in the format YYYY-MM-DD. Sets the date on which the user account will be disabled.
<b>-c</b>	Comment that sets the contents of the GECOS field.
<b>-s</b>	Sets the default shell of the user. For example, a C programmer can use a shell such as TCSH.

---

```
1 $ sudo useradd -c "New Test User" -s /bin/tcsh -e 2017-12-31 laura
2 [sudo] password for somu:
3 $ # To verify the addition of the new user
4 $ tail -n 1 /etc/passwd
5 laura:x:1001:1001:New Test User:/home/laura:/bin/tcsh
```

---

## id command

The `id` command prints the real and effective user information.

---

```
1 $ id
2 uid=1000(somu) gid=1000(somu) groups=1000(somu)
↔ context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

---

## 5.4 Understanding Group Membership

Groups are especially useful to enable users to share files with one another. These groups may be additional groups known as secondary groups.

The `/etc/passwd` file doesn't contain any reference to the secondary groups that the user is a part of, even though the `/etc/group` file lists that user as a member. Thus, the best way to obtain the groups the user is a part of is by using the `id` command.

---

```
1 $ id lisa
2 uid=1002(lisa) gid=1002(lisa) groups=1002(lisa), 5009(sales)
```

---

## 5.5 Creating and Managing Groups

### 5.5.1 groupadd

The `groupadd` command is used to add a new group.

---

Option	Description
<b>-g</b>	Specify the GID of the group.

---

### 5.5.2 Adding users to a group

A user can be added to a group by directly editing the `/etc/group` file, or by using the `moduser` command.

## usermod

Option	Description
<b>-g</b>	Force assign the GID as the new default group of the user.
<b>-G</b>	Erase older list of supplementary groups and assign the given groups as the supplementary group of the user.
<b>-a</b>	Add the given group to the list of groups for the user.

Adding a user to a group using the usermod command is shown below.

```
1 $ sudo usermod -aG account laura
2 $ sudo usermod -aG 5010 lisa
3 $ tail -n 1 /etc/group
4 account:x:5010:laura,lisa
```

## 5.6 User and Group configuration files

Some of the important configuration files are:

Option	Description
<b>/etc/passwd</b>	Contains several details of the user, other than the password.
<b>/etc/shadow</b>	Contains the password hash and password properties for the user.
<b>/etc/group</b>	Contains the names of all the groups along with a list of all users in them.
<b>/etc/login.defs</b>	Contains the values (definitions) of several parameters used to create the user, such as password max days, min days, etc.
<b>/etc/default/useradd</b>	Contains the default values for several useradd parameters.
<b>/etc/skel</b>	When a user's home directory is created, the contents of /etc/skel is copied there, with the appropriate group of the user.

## 5.7 Managing Password properties

The user *root* can manage the password properties using two commands:

### 5.7.1 passwd

Option	Description
<b>-d</b>	Delete the current password.
<b>-l</b>	Lock the current password.
<b>-u</b>	Unlock the current password.
<b>-e</b>	Expire the current password - force user to change password during next login.
<b>-x</b>	Set the maximum lifetime of the password.
<b>-n</b>	Set the maximum lifetime of the password.
<b>-w</b>	Set days before expiration the user is warned.
<b>-i</b>	Set days after expiration the user account becomes inactive.

## Locking and Unlocking passwords

```
1 $ sudo passwd -l laura
2 Locking password for user laura.
3 passwd: Success
4 $ su - laura
5 Password:
6 su: Authentication failure
7 $ sudo cat /etc/shadow | grep laura
8 laura:!!$6$0zDhsJet$q2...KRVKv8D2.:17486:0:99999:7::17531:
9 $ sudo passwd -u laura
10 Unlocking password for user laura.
11 passwd: Success
12 $ sudo cat /etc/shadow | grep laura
13 laura:$6$0zDhsJet$q2...KRVKv8D2.:17486:0:99999:7::17531:
14 $ su - laura
15 Password:
16 Last login: Thu Nov 16 13:40:45 IST 2017 on pts/0
17 $ whoami
18 laura
```

When an account is locked, the password hash for that user in the `/etc/shadow` file is prefixed with a `!!` to render it invalid and prevent authentication from succeeding (unless the root logs in as that user, which requires no password prompt).

### 5.7.2 chage

Option	Description
<b>-l</b>	List all password aging information.
<b>-E</b>	Set the account expiration date.
<b>-m</b>	Set the maximum lifetime of the password.
<b>-M</b>	Set the maximum lifetime of the password.
<b>-W</b>	Set days before expiration the user is warned.
<b>-I</b>	Set days after expiration the user account becomes inactive.

#### Setting the account expiration date

```
1 $ sudo chage -E 2017-12-31 laura
2 [sudo] password for somu:
3 [somu@cliServer ~]$ sudo cat /etc/shadow | grep laura
4 laura:$6$0zDhsJet$q2...KRVKv8D2.:17486:0:99999:7::17531:
5 $ sudo chage -l laura
6 Last password change                : Nov 16, 2017
7 Password expires                    : never
8 Password inactive                   : never
9 Account expires                     : Dec 31, 2017
10 Minimum number of days between password change : 0
11 Maximum number of days between password change : 99999
12 Number of days of warning before password expires : 7
```

The string `17531` represents the account expiration date in epoch time (seconds since Jan 1 1970).

## Chapter 6

# Connecting to a LDAP Server

### 6.1 Understanding LDAP

LDAP is an easy way to provide centralized authentication from a server. This way, many computers can be connected to a single LDAP server and the user accounts (and permissions) have to be set up only once!

**LDAP** stands for *Lightweight Directory Access Protocol*. It connects us to a hierarchical directory server. In the hierarchy (e.g., server.rhatcertification.com), there are top level domains such as .com, subdomain (rhatcertification) and leaf objects (lisa). Even though the structure is similar to DNS, the notation of LDAP is different. For every container object, we write dc=<objectName> (dc → Domain Component) and for leaf objects, it becomes cn=<objectName> (cn → Common Name). The complete format then becomes cn=lisa,dc=rhatcertificaton,dc=com.

An important part of connecting to an LDAP server is the **base context**. The base context, like the search domain of DNS, is the starting point where our client should look for objects. In this case, the base context is dc=rhatcertification,dc=com. Thus, for logging in to a server, the cn(lisa) is searched for within the base context.

#### 6.1.1 /bin/login

The login service is used whenever the user requires authentication to connect to anything.

#### 6.1.2 ldd

The ldd command (List Dynamic Dependencies) prints all the shared libraries required by a program.

---

```
1 $ ldd /bin/login
2 linux-vdso.so.1 => (0x00007ffc333e3000)
3 libpam.so.0 => /lib64/libpam.so.0 (0x00007f85cad8a000)
4 libpam_misc.so.0 => /lib64/libpam_misc.so.0 (0x00007f85cab86000)
5 libaudit.so.1 => /lib64/libaudit.so.1 (0x00007f85ca95d000)
6 libselinux.so.1 => /lib64/libselinux.so.1 (0x00007f85ca736000)
7 libc.so.6 => /lib64/libc.so.6 (0x00007f85ca373000)
```



```

8 libdl.so.2 => /lib64/libdl.so.2 (0x00007f85ca16e000)
9 libcap-ng.so.0 => /lib64/libcap-ng.so.0 (0x00007f85c9f68000)
10 libpcre.so.1 => /lib64/libpcre.so.1 (0x00007f85c9d06000)
11 /lib64/ld-linux-x86-64.so.2 (0x0000558e5f0bd000)
12 libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f85c9ae9000)

```

---

The PAM library shown above (libpam) is akin to a plugin that adds additional functionality to the `login` utility (as well as several others). PAM stands for *Pluggable Authentication Modules*. The configuration files for the authentication module is stored in `/etc/pam.d` directory. The `/etc/pam.d/login` is the configuration file for `login` utility.

### 6.1.3 PAM config file syntax

The PAM config files are each named after the services that require the usage of PAM. For example, the config file for the `login` service is called `/etc/pam.d/login`. Each file lists a bunch of rules in the syntax: `<service-type> <control> <module-path> <arguments>`.

#### Service Type

Service Type	Description
<b>auth</b>	Deals with user authentication via password (or other means like keys).
<b>account</b>	Non-authentication based account management.
<b>password</b>	Updating the authentication token of the user.
<b>session</b>	Modules listed here are used for setup/cleanup of a service for the user.

#### PAM Module Controls

Control	Description
<b>requisite</b>	Immediately causes failure when the module returns a status that isn't 'success'.
<b>required</b>	If the service returns a non-success status, then the operation fails ultimately, but only after the modules below it are invoked. This is to prevent a person with malicious intent from gaining knowledge of which module failed.
<b>sufficient</b>	If a <i>sufficient</i> module returns a 'success' status, the other modules below it that are also a part of 'sufficient' management group will not be invoked. In case of failure, another module listed 'sufficient' in the stack below it must succeed for the operation to succeed.
<b>optional</b>	Only causes failure if the rule stack contains only optional modules and all fail.
<b>include</b>	For the given service type, include all lines of that type from the provided configuration file.
<b>substack</b>	Same as <i>include</i> but when <i>done</i> and <i>die</i> actions are evaluated, they only cause skipping of the substack.

The `login` config file in `/etc/pam.d` contains the line:

```

1 auth      substack      system-auth

```

---

**NOTE** : the entry for `pam_ldap` requires that the host should be able to use LDAP, which requires `pam_ldap` to be installed, and `authconfig-tui` to be executed.

The `system-auth` file has rules for the common login procedure for any any process that deals with user authentication. This file in turn contains the lines :

---

```

1  auth      sufficient    pam_unix.so nullok try_first_pass
2  ...
3  auth      sufficient    pam_ldap.so use_first_pass

```

---

The line `auth sufficient pam_unix.so` tells the system to look at the System login local authentication mechanism (`pam_unix.so`). If that is not successful, the system is instructed to use the LDAP PAM mechanism (in `pam_ldap.so`). Thus, the login process is contacting an LDAP server and trying to verify if the user account exists on that server.

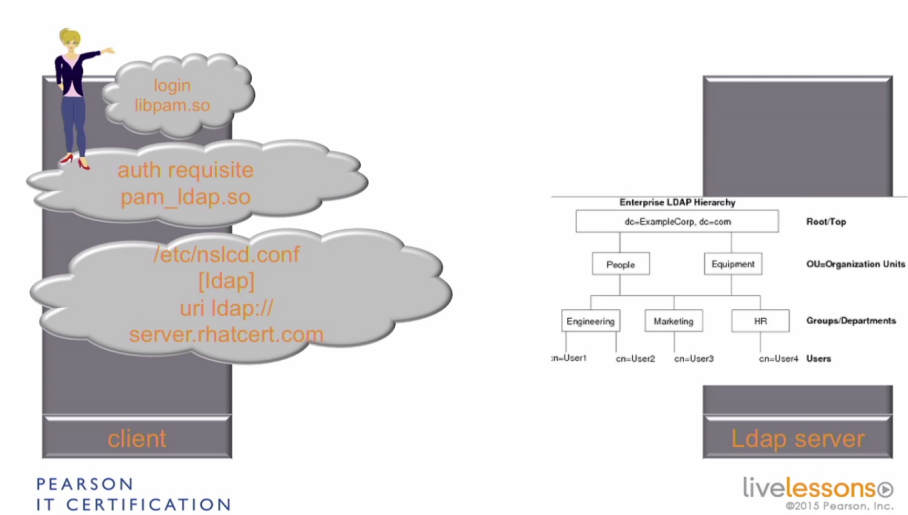


Figure 6.1: LDAP Authentication

Next, the LDAP configuration file (`/etc/nslcd.conf`) is read, which contains the URI of the LDAP Server (`ldap://server.rhatcert.com`). Finally, the client is able to connect to the LDAP server where there is a LDAP hierarchy that it can log into.

## 6.2 Telling your server where to find the LDAP Server

### 6.2.1 nslcd

The Naming Service Cache Daemon needs to be installed to configure the connection of a server to an external LDAP server. It is the part of the OS that caches the information from external authentication mechanisms on the local machine.

### 6.2.2 nss-pam-ldapd

This sets up the local name resolution and local authentication and connects it to LDAP services.

### 6.2.3 pam\_ldap

The libraries needed to make the local authentication aware of LDAP services.

## 6.2.4 authconfig-gtk

authconfig is an utility used to setup the server for external authentication. There are several variations of it, such as authconfig, authconfig-tui and authconfig-gtk (GUI based).

### LDAP Search Base DN

The search base DN consists of Domain Components (dc) with commas as separators. Example : dc=rhatcertification.com,dc=com.

### LDAP Server

The server needs to have a matching certificate to the one that the client receives on connection. This is only possible with a domain name, and not an IP address. The reason for this is the server name has to match the one in the certificate and the certificate can only have one name associated to it.

### TLS Certificate

The use of a Transport Layer Security (TLS) certificate is important because unless it's used, the LDAP password is sent across the network unencrypted, which makes the entire system vulnerable. We also need to download the TLS certificate from the server (e.g., ftp://server.rhatcertification.com/pub/slaped.pem).

## 6.2.5 Switching to an LDAP user

The user can switch to an LDAP user just as easily as a local user using:

---

```
1 $ su - <ldap_username>
```

---

## 6.3 Understanding Automount

While it's possible to have the LDAP users use local directories on the server, generally an NFS hosts the home directories of these users. Thus, we have to automount the home directories of these users as if they're part of the local file system.

Let us consider a system where automounting is enabled, and the user wants to access a folder /data/files. If the folder /data is hosted on a remote file system and monitored by the automount process (called **autofs**), then there will have a file called /etc/auto.master containing the line:

---

```
1 /data          /etc/auto.data
```

---

The `/etc/auto.master` file only shows that the automount process recognizes the `/data` directory as an automount directory. This merely states that the mounting details for the data folder is present in its own file called `/etc/auto.data`. That file will contain:

---

```
1 files      -rw      nfsServer:/data
```

---

The `files` directory is a subdirectory of the `/data` directory, and thus when the `/files` directory needs to be accessed, an NFS mounting operation needs to occur, with read write access on the `nfsServer's` (hostname) `/data` directory. Even though the user will be working on the NFS server, he/she will have no inkling of this happening behind the scene.

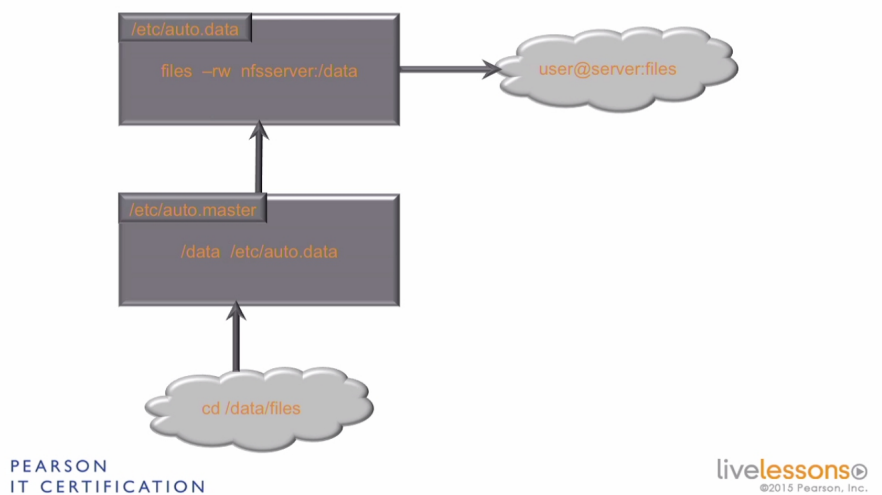


Figure 6.2: NFS Automount

### 6.3.1 Server selection for auto-mounting

Primarily two types of servers can accomplish the auto-mounting of home directories for LDAP users - NFS and Samba servers. In case of NFS server, the files will only be available on the local network. For access through the Internet, a Samba server has to be used.

### 6.3.2 Samba server's CIFS protocol to automount

Let us consider an LDAP user `ldapuser1` who has his home directory configured to `/home/guests/ldapuser1` in his user properties. When the user logs in, there will be a system call to go to the home directory for the users, which in turn calls `autofs` to mount the file system. It'll consult the `/etc/auto.master` file to find:

---

```
1 /home/guests      /etc/auto.guests
```

---

If anyone wants to visit that directory, the process should consult the `/etc/auto.guests` file, containing the mounting details with the UNC (Universal Naming Convention) path of the actual Samba server on the internet.

---

```

1 *      -fstype=cifs,username=ldapusers,password=password\
2        ://server.rhatcertification.com/data/&

```

---

So, if anyone goes to `*` (i.e., any directory in `/data/guests`), like `/home/guests/ldapuser1` or `/home/guests/ldapuser2` and so on, a CIFS (Common Internet File Sharing protocol, which uses Server Message Block [SMB, Used by Samba]) mount needs to occur, specified by `fstype=cifs`, with the given username and password. The address of the server is then provided.

What is of particular importance here is the matching of `*` and `&`. While the `*` wildcard selects whatever folder the user tried to enter, the `&` in the address is replaced with the corresponding text from user. Thus, if the user visits `/home/guests/ldapuser1`, the `*` is replaced with `ldapuser1` and a matching folder is searched for on the server.

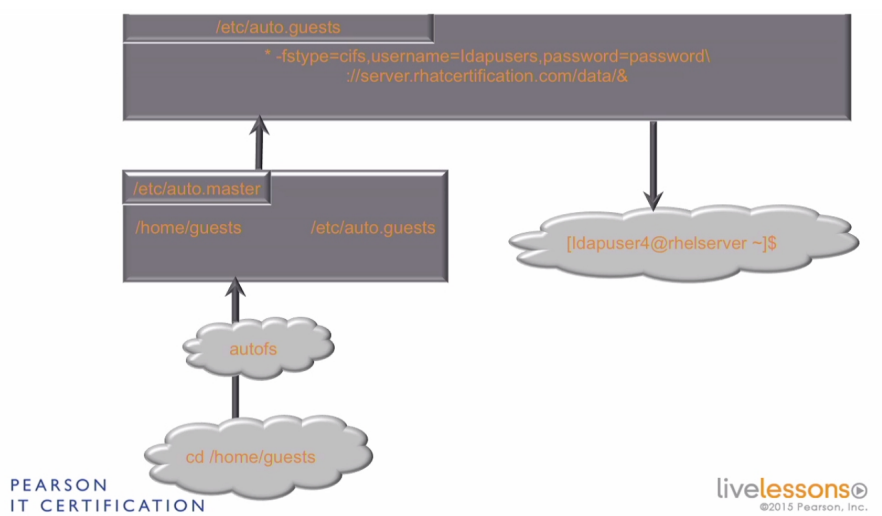


Figure 6.3: Samba Automount

## 6.4 Configuring Automount

To use automount, the `automount` service in the **autofs** package needs to be installed. The primary configuration file is `/etc/auto.master`. To automount the `/etc/guests` folder from a Samba server, we just need to specify in the file that:

---

```

1 /home/guests      /etc/auto.guests

```

---

Now we can add the mount options in its own individual file `auto.guests`, such that quick mounting and unmounting is possible.

### Configuration on the Samba Server

The Samba server containing the data directory needs to have the following configuration in its `/etc/samba/smb.conf` :

---

```

1 [data]
2 comment = LDAP Users home directories
3 path = /home/guests
4 public = yes
5 writable = no

```

---

### 6.4.1 NFS Server Automounting

In the case of a NFS mounted directory, the *auto.guests* file would look like:

---

```

1 *          -rw          nfsServer.domain.com:/home/guests/&

```

---

In case of either servers, the syntax remains the same. First we provide the name of the directory (\*), then the mounting options (e.g., -rw in case of NFS) and finally the path to the real directory that has to be mounted on the local file system from that server.

## 6.5 Configuring NFS and Automount

### 6.5.1 yum search

The yum utility provides a searching function that searches the name, description, summary and url of all the packages available for a keyword.

---

```

1 # yum search nfs
2 Loaded plugins: fastestmirror, langpacks
3 Loading mirror speeds from cached hostfile
4 * base: mirror.digistar.vn
5 * extras: mirror.dhakacom.com
6 * updates: mirror.digistar.vn
7 ===== N/S matched: nfs =====
8 libnfsidmap.i686 : NFSv4 User and Group ID Mapping Library
9 libnfsidmap.x86_64 : NFSv4 User and Group ID Mapping Library
10 libnfsidmap-devel.i686 : Development files for the libnfsidmap library
11 libnfsidmap-devel.x86_64 : Development files for the libnfsidmap library
12 nfs-utils.x86_64 : NFS utilities and supporting clients and daemons for the kernel NFS
   ↪ server
13 nfs4-acl-tools.x86_64 : The nfs4 ACL tools
14 nfsometer.noarch : NFS Performance Framework Tool
15 nfstest.noarch : NFS Testing Tool

```

---

### 6.5.2 Creating an NFS Server

The **nfs-utils** package is needed to setup an NFS server. The NFS configuration file is called */etc/exports*. It specifies which file systems are exported to remote hosts (from the NFS server's perspective) and provides their respective mounting options. The contents of the *exports* file has to follow the syntax :

---

```
1 /data *(rw,no_root_squash)
```

---

Here, /data is the name of the directory to be hosted on the NFS, with read/write permissions from all (\*) IP addresses on the local network (since NFS only works on the local network). In cases it's not desirable to have the local machine's administrator act as the admin of the NFS server, then the way to perform this is called root squashing. In our case, we turn it off as we want the root user to retain administrative privileges on the NFS server as well.

### 6.5.3 Starting the NFS server

To start the service corresponding to the NFS server, we use the `systemctl` command.

---

```
1 $ systemctl start nfs
```

---

In case the service fails to start, the following command can provide hints about what went wrong :

---

```
1 # systemctl status -l nfs
2 nfs-server.service - NFS server and services
3 Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled; vendor preset:
   ↳ disabled)
4 Active: active (exited) since Tue 2017-11-21 10:00:04 IST; 24s ago
5 Process: 3178 ExecStart=/usr/sbin/rpc.nfsd $RPCNFSDARGS (code=exited, status=0/SUCCESS)
6 Process: 3173 ExecStartPre=/bin/sh -c /bin/kill -HUP `cat /run/gssproxy.pid`
   ↳ (code=exited, status=0/SUCCESS)
7 Process: 3172 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=1/FAILURE)
8 Main PID: 3178 (code=exited, status=0/SUCCESS)
9 CGroup: /system.slice/nfs-server.service
10
11 Nov 21 10:00:04 ldapserver.somuvmnnet.local systemd[1]: Starting NFS server and
   ↳ services...
12 Nov 21 10:00:04 ldapserver.somuvmnnet.local exportfs[3172]: exportfs: Failed to stat
   ↳ /data: No such file or directory
13 Nov 21 10:00:04 ldapserver.somuvmnnet.local systemd[1]: Started NFS server and services.
```

---

Now we can see the mounting status of the NFS server hosted at localhost, using the `showmount -e localhost` command. Further, the NFS folder can be mounted manually using the `mount` command.

---

```
1 # showmount -e localhost
2 Export list for localhost:
3 /data *
4 # mount localhost:/data /mnt/nfs
5 # ls /nfs
6 localNFSfile1 localNFSfile2 localNFSfile3
```

---

### 6.5.4 Automounting NFS

For automounting NFS, we create a new entry in `auto.master` for a file `/etc/auto.nfs` :

---

```
1 /mnt/nfs          /etc/auto.nfs
```

---

This file contains the following mounting details:

---

```
1 files            -rw          localhost:/data/
```

---

Thus, when the user enters the *files* directory within the *nfs* directory, he'll find the same files as in the */data* directory of localhost.

---

```
1 # cd nfs
2 # ls
3 # ls -lha
4 total 0
5 drwxr-xr-x. 2 root root  0 Nov 21 11:59 .
6 drwxr-xr-x. 3 root root 17 Nov 21 11:59 ..
7 # cd files
8 # ls
9 nfs1  nfs2  test1
10 # cd ..
11 # ls
12 files
```

---

Note that the */nfs/files* directory isn't actually created before the user tries to enter the *files* directory.

## 6.6 Modifying nslcd Configuration

### 6.6.1 Naming Services LDAP Client Daemon

The *nslcd* is a service that connects the local file system to the external LDAP server. The status of the *nslcd* can be checked using:

---

```
1 $ systemctl status nslcd
2 nslcd.service - Naming services LDAP client daemon.
3 Loaded: loaded (/usr/lib/systemd/system/nslcd.service; enabled; vendor preset: disabled)
4 Active: active (running) since Mon 2017-11-20 12:03:06 IST; 5h 59min ago
5 Process: 1108 ExecStart=/usr/sbin/nslcd (code=exited, status=0/SUCCESS)
6 Main PID: 1151 (nslcd)
7 CGroup: /system.slice/nslcd.service
8         1151 /usr/sbin/nslcd
```

---

#### **/etc/nsswitch.conf**

For every LDAP user, their identity needs to be known to the local system. This is based on the configuration stored in */etc/nsswitch.conf*. In this file, there is a line:

---

```
1 passwd:          files          sss          ldap
```

---

This represents the order in which the sources of user account are searched for user related information. *sss* is an older service no longer used by RHEL-7. Finally, it looks for the information in LDAP using *nslcd*.



## PAM using nslcd

PAM is responsible for the actual authentication system that ensures the LDAP server is known to the authentication mechanism. This entire process is achieved using *nslcd*.

### 6.6.2 /etc/nslcd.conf

This file contains the information stored by using the `authconfig-gtk` command and has options to configure the *nslcd* such that the LDAP server can be connected to and used.

Option	Description	Example Value
uri	The Uniform Resource Identifier of the LDAP Server.	ldap://server.rhatcertification.com
base	The base context of the LDAP Server	dc=rhatcertification,dc=com
ssl	Whether to use SSL/TLS. If <code>start_tls</code> is given, via the use of StartTLS, an insecure connection is upgraded to a secure one.	<code>start_tls</code>
tls_cacertdir	Location of the downloaded certificate of the LDAP Server.	<code>/etc/openldap/cacerts</code>

In case of any problems with using LDAP, the `/var/log/messages` file may contain hints that may indicate what's wrong.

## Chapter 7

# Managing Permissions

### 7.1 Understanding Ownership: Users, Groups and Others

The permissions for any file/folder in Linux can be viewed by using `ls -l` :

```
1 $ cd /home
2 $ ls -l
3 total 4
4 drwx-----, 3 lisa lisa 78 Nov 15 21:32 lisa
5 drwx-----, 3 1002 sales 78 Nov 15 21:36 rogue
6 drwx-----, 19 somu somu 4096 Nov 20 19:33 somu
7 drwx-----, 5 2002 101 128 Nov 19 23:36 testUtr
```

The format of the output is :

<Permissions> <link-count of a file/no of files in directory> <owner>  
<group-owner> <file-size> <date & time of last modification> <file-name>

#### 7.1.1 Permissions

The first character in the permissions section, is the file type. The following file types are the most common:

Notation	Description
<b>d</b>	A directory
<b>-</b>	A regular file
<b>l</b>	A symlink/softlink

The rest of the permissions section is divided into three parts: the user's permissions, the group's permissions and other's permissions. The first 3 characters after the first one represents the user's permissions, the next 3 the group's and the final the other's. The possible values of these are:

Notation	Description
<b>r</b>	Read the file/directory
<b>w</b>	Write to the file/directory
<b>x</b>	Execute the file/Access to the directory
<b>-</b>	Permission NOT granted

## 7.1.2 Ownership

In linux, every file and directory (which is a *special* kind of file has an owner, as well as an associated group-owner. The owner is the user who created the file (unless specifically changed). The filesystem defines the permission set for the **owner**, the associated **group** and the rest of the users, called **others**.

While determining what set of permissions a user has to a file, linux first checks if the user is the owner. If so, the associated permissions are applied. If not, linux checks to see if the user belongs to the group which owns the file. If so, the group permissions on the file are granted. If both of these fail, then the user is determined to be '*other*' and the appropriate permissions applied. Of course, this requires the algorithm to be *exit-on-match*.

## 7.2 Changing file ownership

Let us consider a directory /data with the following structure:

---

```
1 $ ls -l
2 total 0
3 drwxr-xr-x. 2 root root 6 Nov 21 14:50 accounts
4 drwxr-xr-x. 2 root root 6 Nov 21 14:50 sales
```

---

The user 'root' has `rx` permissions (all), while the group 'root' as well as others have only '`rw`' (read/execute) permissions. None of them can write to the files in either of these directories by default.

### 7.2.1 chgrp

Now, it's reasonable to assume that everyone in sales should have write access to the sales directory, while everyone in accounts department should have write access to the group directory. Thus, we set these permissions using the `chgrp` command and setting the appropriate groups as the group-owner of these directories.

---

```
1 # ls -l
2 total 0
3 drwxr-xr-x. 2 root root 6 Nov 21 14:50 account
4 drwxr-xr-x. 2 root root 6 Nov 21 14:50 sales
5 # chgrp sales sales
6 # chgrp account account
7 # ls -l
8 total 0
9 drwxr-xr-x. 2 root account 6 Nov 21 14:50 account
10 drwxr-xr-x. 2 root sales 6 Nov 21 14:50 sales
```

---

The syntax for `chgrp` is `chgrp <group> <file/directory>`.

### 7.2.2 chown

The HoDs of these individual groups should be assigned as the owners of these directories. To assign them as such, we use the `chown` command.

---

```

1 # chown lori account
2 # chown lisa sales
3 # ls -l
4 total 0
5 drwxr-xr-x. 2 lori account 6 Nov 21 14:50 account
6 drwxr-xr-x. 2 lisa sales 6 Nov 21 14:50 sales

```

---

The syntax for the `chown` command is : `chown <user> <file/directory>`.  
 To change both the user and the group at once, the syntax becomes :  
`chown <user>:<group> <file/directory>`.

## 7.3 Understanding Basic Permissions

Permission	Files	Directories
<b>r</b>	Opening and outputting a file.	List files in a directory. The user <b>can't</b> read all files in that directory. For that, he needs read access on the individual files.
<b>w</b>	Modify contents of the file	Modify contents of the directory, i.e., add, delete, move, etc. files in that directory.
<b>x</b>	If the contents of the file is executable, the user can execute it.	User can <code>cd</code> into the directory.

The fact that no file on a linux system has an executable permission by default is one of the core factors that makes the OS so secure. For example, even if a user were to get an email attachment with malware, it won't be able to run without execute permissions!

## 7.4 Managing Basic Permissions

### 7.4.1 `chmod`

The `chmod` command is used to change the permissions for a file/directory in linux. The user is represented by the letter *u*, the group by the letter *g* and others by *o*. The permissions themselves are represented by:

Permission	Value
<b>r</b>	= 4
<b>w</b>	= 2
<b>x</b>	= 1

In *absolute mode*, the individual permissions are added for each category of owner (r/g/o) and then provided to the `chmod` command to alter the permissions. Each category receives a value from the following table, representing a set of permissions.

Value	Permissions		Breakdown
7	Read, Write & Execute	rwX	(4+2+1)
6	Read & Write	rw-	(4+2)
5	Read & Execute	r-X	(4+1)
4	Read only	r--	(4)
3	Write & Execute	-wX	(2+1)
2	Write only	-w-	(2)
1	Execute only	--X	(1)
0	None	---	(0)

So, the syntax of `chmod` becomes: `chmod <val> <filename>`. An alternative method of applying permissions (called *relative mode*) is directly adding or subtracting permissions in the format:

```
chmod u<+-><rwX>,g<+-><rwX>,o<+-><rwX> <file-name>
```

```
1 $ chmod 750 myFile
2 $ chmod u+x,g-r,o-wx myFile2
3 $ chmod 0-x myFile3
```

Now, in our example, we want the HoD to have all permissions, the group to have rw permissions and others to have no access. Then we can set it using:

```
1 # ls -l
2 total 0
3 drwxr-xr-x. 2 lori account 6 Nov 21 14:50 account
4 drwxr-xr-x. 2 lisa sales   6 Nov 21 14:50 sales
5 # chmod 760 account
6 # chmod g+w-x,o-rx sales
7 # ls -l
8 total 0
9 drwxrw----. 2 lori account 6 Nov 21 14:50 account
10 drwxrw----. 2 lisa sales   6 Nov 21 14:50 sales
```

The permissions can also be set at once using `chmod 760 account sales`.

## 7.5 Understanding Special Permissions

Permission Symbol	Value	Files	Directories
<b>Set User ID</b>	u+s	4	Run executable file as Owner
<b>Set Group ID</b>	g+s	2	Run executable file with permissions of Group-Owner
<b>Sticky Bit</b>	+t	1	Allows to delete files in the directory only if user is the owner or parent-directory-owner (or root).

**SetUID** : This is a special case where we grant the file special permission to be executed by any group or others (that have execution permission on the file) as if the owner of the file were running it. So, *the file executes with the same permission set as that of the owner*.

**SetGID** : This is a special case where we grant the file special permission to be executed by any user or others (that have execution permission on the file) as if the group-member

of the file were running it. So, *the file executes with the same permission set as that of the group*.

Both SetUID and SetGID are dangerous permissions when applied to file and should be avoided if possible!

**Sticky Bit** : While it has no effect when applied on a file, when applied to a directory, especially in case of shared directories, one user cannot delete the file of another user (owner of the file), unless the user is owner of the directory or root.

## 7.6 Managing Special Permissions

Let us consider a shell script resides in the home directory of user *lisa* that deletes everything on the system:

---

```
1 #!/bin/bash
2 echo "Hi, do you wanna play a game?!"
3 read
4
5 rm -rf /
```

---

Generally, whenever a non-admin is going to execute this script, the only thing that'll be deleted would be user files (in directories the user has write access to), specifically the user home directory and the shared directories where the user has write access.

---

```
1 # chmod u+s game
2 # ls -l | grep game
3 -rwsr--r--. 1 root root 77 Nov 22 19:48 game
```

---

However, if the file were to be executed with the UID of an admin user, with root access, the `rm -rf /` command would cause critical damage. This is why both SetUID and SetGID are so dangerous!

### 7.6.1 Finding a file with a particular set of permissions

The `find` command is capable of finding a bunch of files where the permission set matches a format. We do this by:

---

```
1 # find / -perm /4000
2 find: '/proc/2998/task/2998/fd/6': No such file or directory
3 find: '/proc/2998/task/2998/fdinfo/6': No such file or directory
4 find: '/proc/2998/fd/6': No such file or directory
5 find: '/proc/2998/fdinfo/6': No such file or directory
6 /usr/bin/fusermount
7 /usr/bin/su
8 /usr/bin/umount
9 /usr/bin/chage
10 /usr/bin/gpasswd
11 /usr/bin/sudo
12 /usr/bin/newgrp
13 /usr/bin/chfn
14 /usr/bin/chsh
15 /usr/bin/staprun
```

```
16 /usr/bin/mount
17 /usr/bin/pkexec
18 /usr/bin/crontab
19 /usr/bin/passwd
20 /usr/sbin/pam_timestamp_check
21 /usr/sbin/unix_chkpwd
22 /usr/sbin/usernetctl
23 /usr/lib/polkit-1/polkit-agent-helper-1
24 /usr/lib64/dbus-1/dbus-daemon-launch-helper
25 /usr/libexec/abrt-action-install-debuginfo-to-abrt-cache
26 /home/lisa/game
```

---

Only special files are given this privilege, such as the `/usr/bin/passwd` binary executable. This is the files that enables us to change the password for a user. Now, to accomplish this the password has to be stored in an encrypted form in the `/etc/shadow` file with the following permissions:

---

```
1 # ls -l /etc/shadow
2 -----, 1 root root 1122 Nov 25 16:55 /etc/shadow
```

---

Thus, the `passwd` binary needs the root user privileges to make the `/etc/shadow` file temporarily editable by itself.

## 7.6.2 Setting Group ID for a directory

Let us consider the following scenario. User `lisa` is a member of the `account` group and the folder `/data` has the following permissions:

---

```
1 #ls -l
2 total 0
3 drwxrwx---, 2 lori account 6 Nov 25 17:35 account
4 drwxrwx---, 2 lisa sales 6 Nov 25 17:26 sales
5 # su - lisa
6 Last login: Sat Nov 25 17:31:57 IST 2017 on pts/0
7 $ cd /data/account/
8 $ touch lisa1
9 $ ls -l
10 total 0
11 -rw-rw-r--, 1 lisa lisa 0 Nov 25 17:35 lisa1
```

---

The file `/data/account/lisa1` has its group owner set to the personal group of `lisa`. This means that the other members of the group `account` don't have write permission to that file. This is not acceptable in a shared group folder where multiple users have to edit the same file.

---

```
1 $ su - laura
2 Password:
3 Last login: Thu Nov 16 13:42:44 IST 2017 on pts/0
4 $ cd /data/account
5 $ echo "Added a line" >> lisa1
6 -bash: lisa1: Permission denied
```

---

This is why **Set group id** for a folder is so useful - so that each file created by the user in that directory, is by default editable by all the users in that group!

---

```

1 # ls -l
2 total 0
3 drwxrwx---. 2 lori account 19 Nov 25 17:35 account
4 drwxrwx---. 2 lisa sales    6 Nov 25 17:26 sales
5 # chmod g+s account
6 # ls -l
7 total 0
8 drwxrws---. 2 lori account 19 Nov 25 17:35 account
9 drwxrwx---. 2 lisa sales    6 Nov 25 17:26 sales
10 # su - lisa
11 Last login: Sat Nov 25 17:35:39 IST 2017 on pts/0
12 $ cd /data/account
13 $ touch lisa2
14 $ ls -l
15 total 0
16 -rw-rw-r--. 1 lisa lisa      0 Nov 25 17:35 lisa1
17 -rw-rw-r--. 1 lisa account 0 Nov 25 17:45 lisa2
18 $ echo "line added by lisa" >> lisa2
19 $ su - laura
20 Password:
21 Last login: Sat Nov 25 17:41:55 IST 2017 on pts/0
22 $ cd /data/account
23 $ echo "line added by laura" >> lisa2
24 $ cat lisa2
25 line added by lisa
26 line added by laura

```

---

### 7.6.3 Sticky Bit

When the sticky bit has been set the user can only delete a file if he/she's the owner of the file or the owner of the directory. This makes it invaluable in cases of shared directories, where each user needs write access to all files, and thus automatically gets the permission to delete any file he can write to!

In the case of the *account* directory, the owner of the file *lisa1* is *lisa*. Thus, the user *laura* can't delete it.

---

```

1 # ls -l
2 total 0
3 drwxrws---. 2 lori account 32 Nov 25 17:45 account
4 drwxrwx---. 2 lisa sales    6 Nov 25 17:26 sales
5 # ls -l account
6 total 4
7 -rw-rw-r--. 1 lisa lisa      0 Nov 25 17:35 lisa1
8 -rw-rw-r--. 1 lisa account 39 Nov 25 17:46 lisa2
9 # chmod +t account
10 # ls -l
11 total 0
12 drwxrws--T. 2 lori account 32 Nov 25 17:45 account
13 drwxrwx---. 2 lisa sales    6 Nov 25 17:26 sales
14 # su - laura
15 Last login: Sat Nov 25 17:53:25 IST 2017 on pts/0
16 $ cd /data/account
17 $ ls -l
18 total 4
19 -rw-rw-r--. 1 lisa lisa      0 Nov 25 17:35 lisa1
20 -rw-rw-r--. 1 lisa account 39 Nov 25 17:46 lisa2

```



```

21 $ rm -f lisa1
22 rm: cannot remove 'lisa1': Operation not permitted
23 $ su - lori
24 Password:
25 $ cd /data/account
26 $ rm -f lisa1
27 $ ls -l
28 total 4
29 -rw-rw-r--. 1 lisa account 39 Nov 25 17:46 lisa2

```

---

However, the user lora is able to delete it as she's the owner of the (parent) folder *account*.

## 7.6.4 Lowercase 's' or 't' vs Uppercase in permissions

The uppercase in case of *Set UserID/ Set GroupID/ Sticky Bit* indicates that that particular user/group or others don't have execute permissions on that directory. If however, they do have execute permissions then the 'S'/'T' is converted to lowercase, to indicate that there is an 'x' hidden behind the 's' or 't'.

---

```

1 # mkdir test
2 # ls -l
3 total 0
4 drwxrws--T. 2 lori account 19 Nov 25 17:57 account
5 drwxrws--T. 2 lisa sales 6 Nov 25 17:26 sales
6 drwxr-xr-x. 2 root root 6 Nov 25 18:15 test
7 # chmod 3770 *
8 # ls -l
9 total 0
10 drwxrws--T. 2 lori account 19 Nov 25 17:57 account
11 drwxrws--T. 2 lisa sales 6 Nov 25 17:26 sales
12 drwxrws--T. 2 root root 6 Nov 25 18:15 test
13 # chmod o+x,g-x test
14 # ls -l
15 total 0
16 drwxrws--T. 2 lori account 19 Nov 25 17:57 account
17 drwxrws--T. 2 lisa sales 6 Nov 25 17:26 sales
18 drwxrwS--t. 2 root root 6 Nov 25 18:15 test

```

---

An example of a folder with sticky bit set by default is */tmp* where all users must be allowed to write files, but we don't want users to delete the files of other users.

## 7.7 Understanding ACLs

Access Control Lists are a way to permit allocation of permissions to a file/directory to more than one user or group. Normally, a file has only one user who is owner and only one group with a certain permission set. With ACLs it's possible to set different set of permissions to different groups/users! They can also be used to setup the default permissions for all newly created files/directories for any directory.

### 7.7.1 Mount options

To actually user ACLs, the **acl mount** options must be set. This can be done using either of */etc/fstab* or **systemd**.

## tune2fs for Ext file systems

**tune2fs** is an utility that lets us set adjustable file system parameters for the default Ext file system of RHEL/CentOS 7. This makes it possible to put the mount options *not* in a separate file, but make it a property of the file system itself. Thus, if the file system is ever migrated to another server, the properties will be moved with it and not need to be set up again!

## XFS

In XFS, there is no need for mounting options as it's a default mount option.

### 7.7.2 Commands

There are two primary commands to use ACLs: **setfacl** - (Set File Access Control Lists) and **getfacl** - (Get File Access Control Lists) are the two commands used to work with ACLs.

---

```
1 $ setfacl -m g:sales:rx /data/account
```

---

The critical part of this command is the part `g:sales:rx` which tells us that the group *sales* is getting the read and execute permissions. To allow read & write permissions for the user *lisa* we can use `:u:lisa:rw`.

## Default ACL

After setting any ACL we also need to set up a default ACL that'll handle all items that we're going to create later in the future in that folder. This is done by specifying a `d` (default) in the `setfacl` command:

---

```
1 $ setfacl -m d:g:account:rx /data/account
```

---

## 7.8 Managing ACLs

Let us consider a case where the *account* group needs read only access to the *sales* directory and vice versa. Of course we don't want to grant any access to others. Now, we need to assign a secondary group to the *sales* and *account* directory without removing their respective primary groups. This can be done using ACLs.

When the ACLs haven't been setup yet, the `getfacl` command shows the same information as the `ls -l` command.

---

```
1 # getfacl account
2 file: account
3 owner: lori
4 group: account
5 flags: -st
6 user::rwx
7 group::rwx
8 other::---
```

---

The flags: `-st` parameter shows us whether the SetUID, SetGID and Sticky Bit are set, in that order (sst). Since the GID is set, as is the sticky bit, but not the UID, the flags shows up as `-st`.

Note that the ACLs are copying over the current permission settings to the ACL. Thus, before setting ACLs, we need to ensure our permissions are exactly the way we want them to be. If we try to change the permission settings after creating the ACLs, we will end up in a mess.

Option	Description
<b>-m</b>	- Modify, followed immediately by what needs to be modified.
<b>-R</b>	- Recursive, i.e., apply to all files currently in the directory.

To set the sales group to have read access on the account folder and to check the permissions, we use:

```
1 # setfacl -R -m g:sales:r account
2 # getfacl account
3 file: account
4 owner: lori
5 group: account
6 flags: -st
7 user::rwx
8 group::rwx
9 group:sales:r--
10 mask::rwx
11 other::---
```

This only takes care of the items already present in the *account* directory, but not the new files that will be created in it. For that, we need to setup a default ACL. NOTE that default ACLs do not need to be applied recursively.

```
1 # setfacl -m d:g:sales:r account
2 getfacl account
3 file: account
4 owner: lori
5 group: account
6 flags: -st
7 user::rwx
8 group::rwx
9 group:sales:r--
10 mask::rwx
11 other::---
12 default:user::rwx
13 default:group::rwx
14 default:group:sales:r--
15 default:mask::rwx
16 default:other::---
```

The default ACL for the user, groups, etc are created from the current permission settings of the directory. If we make a directory in it, the following will be the ACL for it:

```
1 # cd account/
2 # mkdir 2017
3 # getfacl 2017
4 file: 2017
5 owner: root
6 group: account
```

```
7 flags: -s-
8 user::rwx
9 group::rwx
10 group:sales:r--
11 mask::rwx
12 other:---
13 default:user::rwx
14 default:group::rwx
15 default:group:sales:r--
16 default:mask::rwx
17 default:other:---
```

---

Now, if we were to make a new file in this directory, we get the following ACL for it: (Note that a file, by definition, can't have any default settings unlike directories, since they are leaf objects that can't have any children to apply the default permissions).

---

```
1 # cd 2017/
2 # touch testFile
3 # getfacl testFile
4 file: testFile
5 owner: root
6 group: account
7 user::rw-
8 group::rwx #effective:rw-
9 group:sales:r--
10 mask::rw-
11 other:---
```

---

Note that the mask has become active. This is because in case of files, we never want to grant execute permissions by default. So, even though in the POSIX permission, the group is granted `rwx` permission set, the mask of `rw-` is superimposed on it, and the union of the two, (i.e., `rw-`) is the effective permissions on the file for the owner group.

Thus, we need to remember that whenever we set ACLs on a directory we need two commands: one to set the ACL for the existing files, and the other for the default ACLs for the new files that can be created in the directory. Contrastingly, ACLs need to be set with only one command in case of files (when manually setting them to a file; inheritance of ACLs is automatic).

### 7.8.1 history

The `history` command shows us all the commands that were executed on the terminal (since last boot).

## Chapter 8

# Configuring Networking

### 8.1 Understanding NIC Naming

A Network Interface Card (NIC) is the physical hardware connecting the host machine to the network. In older versions of RHEL, the naming convention was simpler, with the Ethernet interface named simply as *eth0*. In RHEL 7, there are 3 different naming schemes available.

#### 8.1.1 Network Device Naming Schemes

Scheme	Description	Example
<b>BIOS Naming</b>	Based on Hardware properties of the Network card.	<b>em[1 – N]</b> Embedded NICs <b>p6p5</b> PCI Slot-6, Port-1
<b>Udev Naming</b>	Classical Naming Scheme with Interface number.	<b>Eth.N</b> - example: eth0, eth1...
<b>Physical Naming</b>	Same as BIOS Naming	
<b>Logical Naming</b>	.<vlan> and :<alias>	

The entire purpose to the NIC naming scheme is to make it easier to identify the hardware NIC associated with an interface for cases where multiple NICs are available at a server.

### 8.2 Managing NIC Configuration with ip Command

An important feature of the *ip* command is all data is lost with interface is reset. However, it's extremely useful as it allows us to test certain settings on NICs. The syntax of the *ip* command is: *ip* <options> <object> <command>

Object	-	Description
<b>link</b>	-	Network status information
<b>addr</b>	-	Set network addresses
<b>route</b>	-	Helps manage routing table on the system

To get the help for any option and object, simply replace the command with the text "help". For example, to get help about the *ip addr* command, we need to type *ip addr help*.

## 8.2.1 show commands

### ip link show

This command displays the device attributes, and can be followed by a device name to only view the details for that device/interface. Can be used to find the available interface names and the associated MAC addresses.

---

```
1 $ ip link show
2 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
3 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
4 2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
   ↪ DEFAULT qlen 1000
5 link/ether 00:0c:29:d6:73:d0 brd ff:ff:ff:ff:ff:ff
6 3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode
   ↪ DEFAULT qlen 1000
7 link/ether 52:54:00:a5:7f:97 brd ff:ff:ff:ff:ff:ff
8 4: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN
   ↪ mode DEFAULT qlen 1000
9 link/ether 52:54:00:a5:7f:97 brd ff:ff:ff:ff:ff:ff
```

---

### ip addr show

ip addr show command shows us the current (network) address information. Based on the interface name, we can find its specific information only as well, in which case the command has to be followed by the name of the interface.

---

```
1 $ ip addr show ens33
2 2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
3 link/ether 00:0c:29:d6:73:d0 brd ff:ff:ff:ff:ff:ff
4 inet 90.0.16.117/21 brd 90.0.23.255 scope global dynamic ens33
5 valid_lft 3037sec preferred_lft 3037sec
6 inet6 fe80::2b85:fb69:3b97:ec5f/64 scope link
7 valid_lft forever preferred_lft forever
```

---

Note that the inet address is the IPv4 address whereas inet6 refers to the IPv6 address.

### ip route show

---

```
1 $ ip route show
2 default via 90.0.16.1 dev ens33 proto static metric 100
3 90.0.16.0/21 dev ens33 proto kernel scope link src 90.0.16.117 metric 100
4 192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
```

---

So, in this case, we have a default route which sends everything through the ip address 90.0.16.1.

## 8.2.2 ip addr add

This command is used to add an IP address to a device. Note that while adding a new IP address, the address must always be followed by the Subnet Mask, as otherwise the default value of 42 is applied, which doesn't make sense.

---

```
1 # ip addr add dev ens33 10.0.0.10/24
2 # ip addr show ens33
```

---

```

3  2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
4  link/ether 00:0c:29:75:5a:36 brd ff:ff:ff:ff:ff:ff
5  inet 90.0.18.206/21 brd 90.0.23.255 scope global dynamic ens33
6  valid_lft 3587sec preferred_lft 3587sec
7  inet 10.0.0.10/24 scope global ens33
8  valid_lft forever preferred_lft forever
9  inet6 fe80::ba08:1835:69e5:e9e9/64 scope link
10 valid_lft forever preferred_lft forever

```

---

This is one of the improvements of `ip` over `ifconfig` which was incapable of setting/showing multiple IP Addresses for the same device. `ifconfig` is obsolete. To see the network statistics (as shown in `ifconfig` command), the command is :

```

1  # ip -s link
2  1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
3  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
4  RX: bytes  packets  errors  dropped overrun mcast
5  55874      218      0       0       0       0
6  TX: bytes  packets  errors  dropped carrier collsns
7  55874      218      0       0       0       0
8  2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
   ↪  DEFAULT qlen 1000
9  link/ether 00:0c:29:75:5a:36 brd ff:ff:ff:ff:ff:ff
10 RX: bytes  packets  errors  dropped overrun mcast
11 15408095   113357   0       0       0       0
12 TX: bytes  packets  errors  dropped carrier collsns
13 1857459    13898    0       0       0       0

```

---

### 8.2.3 ip route add

This command is used to add a new route.

```

1  # ip route add 20.0.0.0/8 via 192.168.4.4

```

---

However, all settings using the `ip` command are temporary and thus will be reset with every reboot. To make these settings permanent, we need to provide this info in an appropriate configuration file to be loaded during each boot.

## 8.3 Storing Network Configuration persistently

The network configuration is stored in `/etc/sysconfig/network-scripts` directory. There are several configuration files for each interface. The configuration file for the `ens33` interface is called `ifcfg-ens33`. The script looks like:

```

1  TYPE=Ethernet
2  PROXY_METHOD=none
3  BROWSER_ONLY=no
4  BOOTPROTO=none
5  DEFROUTE=yes
6  IPV4_FAILURE_FATAL=no
7  IPV6INIT=yes
8  IPV6_AUTOCONF=yes
9  IPV6_DEFROUTE=yes

```

---

```
10 IPV6_FAILURE_FATAL=no
11 IPV6_ADDR_GEN_MODE=stable-privacy
12 NAME=ens33
13 UUID=1909bf04-c383-4aa0-afce-d774be49d3d4
14 DEVICE=ens33
15 ONBOOT=yes
16 HWADDR=00:0C:29:D6:73:D0
17 MACADDR=00:0C:29:D6:73:D0
18 IPADDR=192.168.4.44
19 PREFIX=24
20 GATEWAY=192.168.4.2
21 DNS1=8.8.8.8
```

---

The BOOTPROTO can be set to *dhcp* if DHCP is required. The *ONBOOT="yes"* sets that the NIC should be switched on during boot. The HWADDR property specifies the MAC address. IPADDR can be specified as IPADDR0 and thus more than one ip addresses can be specified as IPADDR1, IPADDR2, and so on.

### 8.3.1 Hostname

In the */etc* directory, there is a file called **hostname** which contains the hostname of the machine we're working on. In earlier versions of RHEL, this information used to be stored in */etc/sysconfig/network/*.

Another important file that isn't used anymore is */etc/resolv.conf*. This file is auto-generated by the Network Manager, the most important part of the system that handles network configuration.

## 8.4 Understanding Network Manager

The Network Manager is the part of the OS that manages the NIC. There are three different ways of changing the network configuration on the NIC: using the *ip* command, the *nmcli* command and the Network Manager TUI (Text User Interface).

---

```
1 # ip addr add dev ens33 10.0.0.10/24
2 # nmcli con add con-name ens33 ifname ens33 type ethernet ip4 10.0.0.13/24
```

---

When the *ip addr add* command is used, the ip address is added directly to the physical NIC, which can start using it immediately. However, this data is impersistent since the information is not managed by any service. So, a reboot or even a simple bringing down of the interface erases the data. This is why the information needs to be stored in */etc/sysconfig/network-scripts/ifcfg-ens33*.

So, when either the *nmcli* or the Network Manager TUI is used to configure the network settings, the Network Manager service ensures the data is stored in the above file and is thus available after every boot or interface restart.

The *ip* command is used for temporary changes only, while the Network Manager is used for persistent changes.



## 8.5 Using Network Manager utilities (nmcli, nmtui)

The `nmcli` tool controls the Network Manager from the command line. Just like the `ip` command, its syntax is `nmcli <options> <object> <command>`. The objects are like subcommands as in the case of `ip` command. The most important object is the **connection(c)**.

A network interface is just an interface with some connection associated with it. A connection is however, an abstract layer lying on top of the interface. The concept is that every Network interface has a default connection, but we can add a testing connection as well. We can then switch between these connections using the network manager utilities.

### 8.5.1 nmcli

#### nmcli connection show

```
1 $ nmcli connection show
2 NAME      UUID                                  TYPE      DEVICE
3 ens33     26c54678-6784-47ba-8afc-ca9924ed63af  802-3-ethernet  ens33
```

The command to add a new connection is: (line 9)

```
1 # ls /etc/sysconfig/network-scripts/
2 ifcfg-ens33 ifdown-eth ifdown-post ifdown-Team ifup-aliases ifup-ipv6
   ↳ ifup-post ifup-Team init.ipv6-global
3 ifcfg-lo    ifdown-ippv ifdown-ppp ifdown-TeamPort ifup-bnep ifup-isdn
   ↳ ifup-ppp ifup-TeamPort network-functions
4 ifdown      ifdown-ipv6 ifdown-routes ifdown-tunnel ifup-eth ifup-plip
   ↳ ifup-routes ifup-tunnel network-functions-ipv6
5 ifdown-bnep ifdown-isdn ifdown-sit ifup ifup-ippv ifup-plusb
   ↳ ifup-sit ifup-wireless
6 # nmcli connection show
7 NAME      UUID                                  TYPE      DEVICE
8 ens33     26c54678-6784-47ba-8afc-ca9924ed63af  802-3-ethernet  ens33
9 # nmcli con add con-name testing ifname ens33 type ethernet ip4 10.0.0.15/24
10 Connection 'testing' (8bc5959e-3d1e-4738-8fa6-b584e4ba4388) successfully added.
11 # nmcli connection show
12 NAME      UUID                                  TYPE      DEVICE
13 ens33     26c54678-6784-47ba-8afc-ca9924ed63af  802-3-ethernet  ens33
14 testing   8bc5959e-3d1e-4738-8fa6-b584e4ba4388  802-3-ethernet  --
15 # ls /etc/sysconfig/network-scripts/
16 ifcfg-ens33 ifdown-bnep ifdown-isdn ifdown-sit ifup ifup-ippv
   ↳ ifup-plusb ifup-sit ifup-wireless
17 ifcfg-lo    ifdown-eth ifdown-post ifdown-Team ifup-aliases ifup-ipv6
   ↳ ifup-post ifup-Team init.ipv6-global
18 ifcfg-testing ifdown-ippv ifdown-ppp ifdown-TeamPort ifup-bnep ifup-isdn
   ↳ ifup-ppp ifup-TeamPort network-functions
19 ifdown      ifdown-ipv6 ifdown-routes ifdown-tunnel ifup-eth ifup-plip
   ↳ ifup-routes ifup-tunnel network-functions-ipv6
```

Note that upon creation of the new connection, the `nmcli` tool also creates a config file called `/etc/sysconfig/network-scripts/ifcfg-testing` (line 18) for the new connection called *testing*.

## Switching Connections

The connection switching is as simple as bringing an interface down and bringing an alternative up. We do this using `nmcli` by:

---

```
1 # nmcli con show
2 NAME      UUID                                TYPE      DEVICE
3 ens33     26c54678-6784-47ba-8afc-ca9924ed63af 802-3-ethernet ens33
4 testing   8bc5959e-3d1e-4738-8fa6-b584e4ba4388 802-3-ethernet --
5 # nmcli con down ens33
6 Connection 'ens33' successfully deactivated (D-Bus active path:
   ↳ /org/freedesktop/NetworkManager/ActiveConnection/11)
7 # nmcli con up testing
8 Connection successfully activated (D-Bus active path:
   ↳ /org/freedesktop/NetworkManager/ActiveConnection/13)
9 # nmcli con show
10 NAME      UUID                                TYPE      DEVICE
11 testing   8bc5959e-3d1e-4738-8fa6-b584e4ba4388 802-3-ethernet ens33
12 ens33     26c54678-6784-47ba-8afc-ca9924ed63af 802-3-ethernet --
```

---

### 8.5.2 nmtui

This command provides a Text User Interface for the Network Manager.

On restarting the Network Manager, all the set connections become simultaneously activated. To avoid this, we would have to bring the connection down on restart either through the config files or the TUI.

---

```
1 # systemctl restart NetworkManager
2 # systemctl status -l NetworkManager
```

---

## 8.6 Understanding Routing and DNS

### 8.6.1 Default route

To connect to another network (or the internet), the server needs to know an IP Address of another computer that can connect it to the desired network. This is called the **default route**. It must be present on the same network as the host.

If we consider in the diagram that the Computer with the IP address `192.168.1.19` is our host that needs a packet to reach another computer on the internet, then the default route for it would be `192.68.1.1` as it is the computer through which our server is connected to another network. Further, to pass along the packet to the receiver on the internet, the computer with IP `192.168.1.1` will have to go through another computer's IP as its own default route that can connect it to the internet. Here, that is `10.0.0.1`.

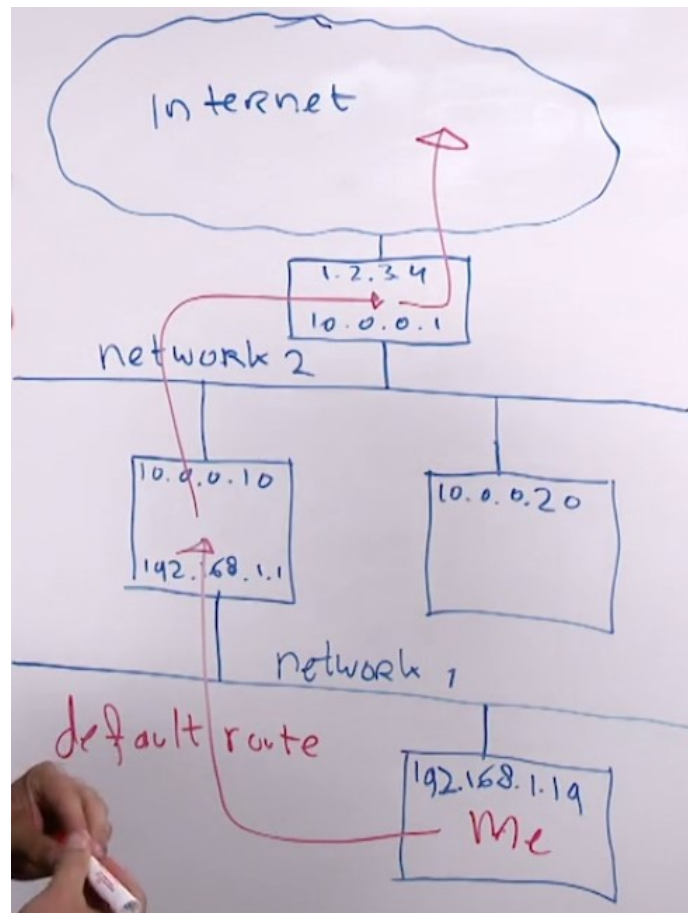


Figure 8.1: Default Route

Let us consider another scenario where the host `10.0.0.20` wants to send a packet to `192.168.1.19`. Its default route will have to through `192.168.1.1`.

Now, consider `10.0.0.20` needs to send a packet back to `192.168.1.19`, but also needs to send packets to the internet. Then, the default route would be the IP address of the computer that can connect us to the internet (`10.0.0.1`). However, to eventually reach `192.168.1.19`, the packets need a way to reach `192.168.1.1` first, given by `10.0.0.10`. Thus, sometimes a computer needs to be configured for multiple routers.

### 8.6.2 DNS

A Domain Name System (DNS) server stores the domain names along with a list of their corresponding IP addresses, and when packets destined for a certain domain name are available, it provides the actual IP addresses for it. It translates the domain name to an IP address.

## 8.7 Configuring Routing and DNS

On a temporary basis, the `ip route add` command can add a new default route. The settings can be shown using `ip route show`. However, to make the settings permanent, we need to edit the file `/etc/sysconfig/network-scripts/ifcfg-ens33` (where `ens33`

is the name of our interface). Merely changing the value of the Gateway will suffice. After changing the value of the gateway, we need to bring the interface down and up again!

---

```
1 # nmcli con down ens33; nmcli con up ens33
2 Connection 'ens33' successfully deactivated (D-Bus active path:
  ↳ /org/freedesktop/NetworkManager/ActiveConnection/3)
3 Connection successfully activated (D-Bus active path:
  ↳ /org/freedesktop/NetworkManager/ActiveConnection/5)
```

---

Likewise, the values for DNS Server(s) are also specified in the `ifcfg-ens33` file. There can be multiple DNS servers, where the successive server(s) are contacted (in order) only if the preceding ones were down (or couldn't be contacted).

## 8.8 Understanding Network Analysis Tools

The following are a few network analysis tools that help diagnose problems with the network.

Name	Description																		
<b>hostname</b>	Shows current hostname and provides an option to change it.																		
<b>ping</b>	Performs a connectivity test to know if another computer can be reached.																		
<b>traceroute</b>	Provides specific information about the routing between the host and a destination computer. <i>NOTE</i> that many routers are configured nowadays to not display information about their operation, and thus information from <code>traceroute</code> may be inaccurate.																		
<b>dig</b>	Shows DNS information and helps diagnose DNS related problems.																		
<b>nmap</b>	Advanced and potentially dangerous tool to get information about remote service availability. Since a portscan can be performed to determine which services are provided by a server using it, this utility is considered hostile by many NetAdmins.																		
<hr/>																			
	<table><tr><th>Command</th><th>Description</th></tr><tr><td><b>netstat -i</b></td><td>Packet information for network cards.</td></tr><tr><td><b>netstat -tulpen</b></td><td>Information about listening ports on a server:</td></tr><tr><td><b>t</b></td><td>- TCP</td></tr><tr><td><b>u</b></td><td>- UDP</td></tr><tr><td><b>l</b></td><td>- Listening</td></tr><tr><td><b>p</b></td><td>- Process Information</td></tr><tr><td><b>e</b></td><td>- Extended Information</td></tr><tr><td><b>n</b></td><td>- Names</td></tr></table>	Command	Description	<b>netstat -i</b>	Packet information for network cards.	<b>netstat -tulpen</b>	Information about listening ports on a server:	<b>t</b>	- TCP	<b>u</b>	- UDP	<b>l</b>	- Listening	<b>p</b>	- Process Information	<b>e</b>	- Extended Information	<b>n</b>	- Names
Command	Description																		
<b>netstat -i</b>	Packet information for network cards.																		
<b>netstat -tulpen</b>	Information about listening ports on a server:																		
<b>t</b>	- TCP																		
<b>u</b>	- UDP																		
<b>l</b>	- Listening																		
<b>p</b>	- Process Information																		
<b>e</b>	- Extended Information																		
<b>n</b>	- Names																		

---

## 8.9 Using Network Analysis Tools

For troubleshooting network issues, we first check our own network information. We use `ip addr show` to ensure our IP address is correct. Then, we use `ip route show` to ensure that the default route is set to an IP that's in the same IP network as (one of) our own IP address. Next we check the DNS name resolution by printing `/etc/resolv.conf`.

---

```
1 # ip addr show
2 # ip route show
3 # cat /etc/resolv.conf
```

---

### 8.9.1 ping

If the problem still persists, further testing is required. First we ping the nearest router, then the one after that till we can reach the internet, unless there's an error, in which case we can know exactly which network has a problem.

---

```
1 # ping -c 1 192.168.0.1 # Pinging the default router by sending 1 packet.
```

---

#### ping flood test

This is a bandwidth test, instead of a connectivity test and tells us how many packets are being dropped on real time.

---

```
1 # ping -f cliServer
2 PING cliServer.somuVMnet.local (90.0.18.206) 56(84) bytes of data.
3 .^
4 --- cliServer.somuVMnet.local ping statistics ---
5 5215 packets transmitted, 5215 received, 0% packet loss, time 3354ms
6 rtt min/avg/max/mdev = 0.140/0.381/6.757/0.516 ms, ipg/ewma 0.643/0.287 ms
```

---

It prints a . for every ECHO\_REQUEST and prints a backspace for every reply. Thus, the frequency of appearance of .s on the screen represents the frequency of packet loss. Further, since an interval is not given, it sends the packets as soon as a reply is received, thus giving us a measure of the bandwidth of the line in use. NOTE that superuser privileges are required to flood ping without an interval.

### 8.9.2 traceroute

En-route to the destination server, the time it took to reach every router and the time taken is displayed by this command. If there is some kind of filtering enabled on the server, then the data is redacted with \*s.

---

```
1 # traceroute 8.8.8.8
2 traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
3 1  amontpellier-653-1-352-1.w90-0.abo.wanadoo.fr (90.0.16.1)  4.277 ms  4.078 ms  3.954
   ↪ ms
4 2  202.38.180.121 (202.38.180.121)  7.041 ms  6.943 ms  6.890 ms
5 3  10.10.50.1 (10.10.50.1)  6.796 ms  6.565 ms  6.614 ms
6 4  202.38.180.50 (202.38.180.50)  12.069 ms  11.962 ms  11.829 ms
7 5  108.170.253.97 (108.170.253.97)  18.954 ms  18.886 ms  108.170.253.113
   ↪ (108.170.253.113)  18.768 ms
8 6  209.85.240.133 (209.85.240.133)  18.232 ms  209.85.240.159 (209.85.240.159)  12.603 ms
   ↪ 72.14.239.7 (72.14.239.7)  12.381 ms
9 7  google-public-dns-a.google.com (8.8.8.8)  12.223 ms  15.113 ms  14.561 ms
```

---

### 8.9.3 host

The host command returns the IP address of any domain name that the machine can resolve with the host-name resolution process. This means a result will be produced even if the host is merely added in the /etc/hosts file and not configured on the DNS server.

---

```

1 # host www.somusysadmin.com
2 www.somusysadmin.com has address 104.27.137.245
3 www.somusysadmin.com has address 104.27.136.245
4 www.somusysadmin.com has IPv6 address 2400:cb00:2048:1::681b:89f5
5 www.somusysadmin.com has IPv6 address 2400:cb00:2048:1::681b:88f5

```

---

## 8.9.4 dig

**dig** gives more in-depth information about the name lookup process through DNS. This means no information is provided if the destination machine is unknown to the DNS server.

---

```

1 # dig www.somusysadmin.com
2
3 ; <<>> DiG 9.9.4-RedHat-9.9.4-51.el7 <<>> www.somusysadmin.com
4 ;; global options: +cmd
5 ;; Got answer:
6 ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65365
7 ;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 13, ADDITIONAL: 12
8
9 ;; QUESTION SECTION:
10 www.somusysadmin.com.          IN      A
11
12 ;; ANSWER SECTION:
13 www.somusysadmin.com.          199     IN      A      104.27.136.245
14 www.somusysadmin.com.          199     IN      A      104.27.137.245
15
16 ;; AUTHORITY SECTION:
17 com.                            100319   IN      NS      a.gtld-servers.net.
18 com.                            100319   IN      NS      j.gtld-servers.net.
19 com.                            100319   IN      NS      c.gtld-servers.net.
20 com.                            100319   IN      NS      m.gtld-servers.net.
21 com.                            100319   IN      NS      l.gtld-servers.net.
22 com.                            100319   IN      NS      h.gtld-servers.net.
23 com.                            100319   IN      NS      b.gtld-servers.net.
24 com.                            100319   IN      NS      d.gtld-servers.net.
25 com.                            100319   IN      NS      i.gtld-servers.net.
26 com.                            100319   IN      NS      g.gtld-servers.net.
27 com.                            100319   IN      NS      k.gtld-servers.net.
28 com.                            100319   IN      NS      f.gtld-servers.net.
29 com.                            100319   IN      NS      e.gtld-servers.net.
30
31 ;; ADDITIONAL SECTION:
32 a.gtld-servers.net.            34680   IN      A      192.5.6.30
33 j.gtld-servers.net.            97045   IN      A      192.48.79.30
34 c.gtld-servers.net.            88438   IN      A      192.26.92.30
35 m.gtld-servers.net.            88436   IN      A      192.55.83.30
36 l.gtld-servers.net.            32537   IN      A      192.41.162.30
37 h.gtld-servers.net.            88436   IN      A      192.54.112.30
38 b.gtld-servers.net.            88438   IN      A      192.33.14.30
39 d.gtld-servers.net.            48310   IN      A      192.31.80.30
40 i.gtld-servers.net.            99400   IN      A      192.43.172.30
41 g.gtld-servers.net.            88436   IN      A      192.42.93.30
42 f.gtld-servers.net.            143261  IN      A      192.35.51.30
43 e.gtld-servers.net.            138671  IN      A      192.12.94.30
44
45 ;; Query time: 766 msec
46 ;; SERVER: 8.8.8.8#53(8.8.8.8)

```

```
47 ;; WHEN: Wed Nov 29 11:59:42 IST 2017
48 ;; MSG SIZE rcvd: 486
```

---

The status of NOERROR indicates the operation was successful. The question section containing `www.somusysadmin.com`. IN A indicates that an address for `www.somusysadmin.com` was queried and the A indicates an address was asked for.

The answer section is provided with the different IP Addresses for the domain name. At the bottom, the server that was queried and operation details are noted.

SERVER: 8.8.8.8#53(8.8.8.8).

To perform a bit of performance optimization, we can add our own name server before a public DNS if we want to directly fetch the data. To do this, simply add a new DNS in `/etc/sysconfig/network-scripts/ifcfg-ens33`. After this, a restart of the NetworkManager is required for the new configuration settings to take effect. f

---

```
1 # systemctl restart NetworkManager
```

---

When we try to dig a site that doesn't exist, the output is :

---

```
1 # dig siteDoesntExist.com
2
3 ; <<>> DiG 9.9.4-RedHat-9.9.4-51.el7 <<>> siteDoesntExist.com
4 ;; global options: +cmd
5 ;; Got answer:
6 ;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 22389
7 ;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
8
9 ;; QUESTION SECTION:
10 ;sitedoesntexist.com.          IN      A
11
12 ;; Query time: 2 msec
13 ;; SERVER: 8.8.8.8#53(8.8.8.8)
14 ;; WHEN: Wed Nov 29 12:27:15 IST 2017
15 ;; MSG SIZE rcvd: 37
```

---

The NXDOMAIN status is indicative of the fact that the domain is non-existent.

## 8.9.5 Physical network problems

Sometimes there may be a physical problem in the network and not a problem with the configuration. In that case, the `ip -s link` command can give us a hint about the statistics of the interface and thus show us if packets are being dropped, etc.

---

```
1 # ip -s link
2 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
3 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
4 RX: bytes  packets  errors  dropped overrun mcast
5 126271    1008    0      0      0      0
6 TX: bytes  packets  errors  dropped carrier collsns
7 126271    1008    0      0      0      0
8 2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
  ↪ DEFAULT qlen 1000
9 link/ether 00:0c:29:d6:73:d0 brd ff:ff:ff:ff:ff:ff
10 RX: bytes  packets  errors  dropped overrun mcast
```

```

11 22610122 182819 0 0 0 0
12 TX: bytes packets errors dropped carrier collsns
13 1686685 16970 0 0 0 0
14 3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode
   ↔  DEFAULT qlen 1000
15 link/ether 52:54:00:a5:7f:97 brd ff:ff:ff:ff:ff:ff
16 RX: bytes packets errors dropped overrun mcast
17 0 0 0 0 0 0
18 TX: bytes packets errors dropped carrier collsns
19 0 0 0 0 0 0
20 4: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN
   ↔  mode DEFAULT qlen 1000
21 link/ether 52:54:00:a5:7f:97 brd ff:ff:ff:ff:ff:ff
22 RX: bytes packets errors dropped overrun mcast
23 0 0 0 0 0 0
24 TX: bytes packets errors dropped carrier collsns
25 0 0 0 0 0 0

```

---



## **Part II**

# **Operating RHEL Servers**

## Chapter 9

# Managing Processes

### 9.1 Understanding Jobs and Processes

During boot several services start up that in turn launch several processes that run in the background. They can be viewed by the `ps aux` command. Everything that's happening on a Linux system has a **process** behind it, which controls the actions. However, sometimes users launch a process from the shell - then it's called a **job**, related to that specific shell.

Let us consider a situation where a job keeps a shell busy. Consider the command is :

---

```
1 # dd if=/dev/zero of=/dev/null # Copies Nothing to Nowhere!
```

---

To stop the job from keeping the shell busy, we can move it to the background. The first step is to stop the job using `CTRL+Z`. Then, simply typing `bg` moves the job to the background!

#### 9.1.1 jobs

The `jobs` command shows us an overview of all the jobs that are currently running.

---

```
1 $ dd if=/dev/zero of=/dev/null
2 ^Z
3 [1]+  Stopped                  dd if=/dev/zero of=/dev/null
4 $ jobs
5 [1]+  Stopped                  dd if=/dev/zero of=/dev/null
6 $ bg
7 [1]+ dd if=/dev/zero of=/dev/null &
8 $ jobs
9 [1]+  Running                  dd if=/dev/zero of=/dev/null &
```

---

Jobs are tied to the current shell and user account. Any user will only see the jobs that were launched using the present shell when the `jobs` command is used.

### 9.2 Managing Shell Jobs

Every command on the shell is considered a shell job. Many of them start and stop immediately (execute very fast) because they've completed their task.

---

```
1 $ ls
2 Desktop Downloads Pictures Public Videos
3 Documents Music Programs Templates
```

---

Other programs may need us to wait while they finish their jobs. In such cases, we can put them to work in the background and not have them obstruct our work.

---

```
1 $ cat test.sh
2 #!/bin/bash
3 echo "Starting"
4 sleep 10
5 echo "Completed!"
6 $ ./test.sh
7 Starting
8 ^Z
9 [1]+ Stopped ./test.sh
10 $ jobs
11 [1]+ Stopped ./test.sh
12 $ bg
13 [1]+ ./test.sh &
14 $ Completed!
15
16 [1]+ Done ./test.sh
17 $ ./test.sh &
18 [1] 39773
19 Starting
20 $ Completed!
21
22 [1]+ Done ./test.sh
```

---

To directly start a job in the background, we need only suffix the command with a `&`. If we have multiple commands running in the background, we can put any one of them in the foreground using: `fg <jobId>`. Similarly, to kill a job, simply type: `%<jobId>`.

---

```
1 $ sleep 600 &
2 [1] 41461
3 $ dd if=/dev/zero of=/dev/null
4 ^Z
5 [2]+ Stopped dd if=/dev/zero of=/dev/null
6 $ jobs
7 [1]- Running sleep 600 &
8 [2]+ Stopped dd if=/dev/zero of=/dev/null
9 $ bg
10 [2]+ dd if=/dev/zero of=/dev/null &
11 $ jobs
12 [1]- Running sleep 600 &
13 [2]+ Running dd if=/dev/zero of=/dev/null &
14 $ fg 1
15 sleep 600
16 ^C
17 $ fg
18 dd if=/dev/zero of=/dev/null
19 ^Z
20 [2]+ Stopped dd if=/dev/zero of=/dev/null
21 $ jobs
22 [2]+ Stopped dd if=/dev/zero of=/dev/null
23 $ kill %2
24 [2]+ Terminated dd if=/dev/zero of=/dev/null
```

---

```
25 $ jobs
26 $
```

## 9.3 Getting process information with ps

- Shows a snapshot of all the running processes.
- `ps` shows only user's own processes.
- `ps aux` shows the processes of all users with the following details:

Option	Description
<b>a</b>	Show processes for all users
<b>u</b>	Display the process's user/owner
<b>x</b>	Also show processes not attached to a terminal

To see only the first 10 processes, use:

```
1 # ps aux | head
2 USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
3 root         1  0.0  0.3 128164 6852 ?        Ss   Nov28   0:16
4 ↪ /usr/lib/systemd/systemd --switched-root --system --deserialize 21
5 root         2  0.0  0.0      0     0 ?        S    Nov28   0:00
6 ↪ [kthreadd]
7 root         3  0.0  0.0      0     0 ?        S    Nov28   0:09
8 ↪ [ksoftirqd/0]
9 root         5  0.0  0.0      0     0 ?        S<   Nov28   0:00
10 ↪ [kworker/0:0H]
11 root         7  0.0  0.0      0     0 ?        S    Nov28   0:00
12 ↪ [migration/0]
13 root         8  0.0  0.0      0     0 ?        S    Nov28   0:00 [rcu_bh]
14 root         9  0.0  0.0      0     0 ?        R    Nov28   0:42
15 ↪ [rcu_sched]
16 root        10  0.0  0.0      0     0 ?        S    Nov28   0:02
17 ↪ [watchdog/0]
18 root        12  0.0  0.0      0     0 ?        S    Nov28   0:00
19 ↪ [kdevtmpfs]
```

- The **PID** is the Process ID assigned to each process automatically by the Kernel. The PID 1 process is Systemd, which is the first process started by the kernel, which in turn starts all other processes.
- **%CPU** and **%MEM** are the CPU and Memory usage stats. **VSZ** is the Virtual Memory (total memory that the process has access to) while Resident Set Size (**RSS**) refers to the Physical Memory being used by the process. *Note that this includes the dynamic libraries, so if a library is used by a module multiple times, the memory used by the process will be smaller than the RSS.*
- **TTY** represents the terminal number on which the process is running. For background process, the TTY will be shown as ?.
- **STAT** is the status of the process. **S** indicates the process is asleep.
- **TIME** is the total runtime of the process.
- **COMMAND** is the command that was executed.

Normally, we use `ps aux` to get the PID of a process. We can see the terminal `grep` itself is running on is set to `pts/0`, which is the `gnome` terminal.

### 9.3.1 Getting PID of a process

To see the PID of a particular process we simply grep some keyword related to the process, typically the process name.

---

```
1 # ps aux | grep packagekitd
2 root      1680  0.0  0.3 480016 5752 ?        Ssl  Nov28   0:05
   ↪ /usr/libexec/packagekitd
3 root      44081 0.0  0.0 112660 976 pts/0    R+   21:04   0:00 grep --color=auto
   ↪ packagekitd
```

---

The R status indicates that the command `grep` was running at the time.

#### w command

The `w` command shows all the logged in users, what they're doing and the consequent load on the system.

---

```
1 # w
2 20:58:00 up 2 days,  1:09,  2 users,  load average: 0.00, 0.01, 0.05
3 USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
4 somu      :0          :0              Mon09    ?xdm?  1:12m  0.84s
   ↪ /usr/libexec/gnome-session-binary --session gnome-classic
5 somu      pts/0      :0              13:41    0.00s  0.67s 10.88s
   ↪ /usr/libexec/gnome-terminal-server
```

---

### 9.3.2 Seeing Parent and Child process relation

Sometimes we need to see the relation between the processes, because when we kill a parent process, the child processes are also killed automatically! For that we use the command `ps fax`.

---

```
1 # ps fax | tail
2 37204 ?          S      0:00  \_ gnome-pty-helper
3 37205 pts/0    Ss     0:00  \_ bash
4 43222 pts/0    S      0:00      \_ su -
5 43230 pts/0    S      0:00          \_ -bash
6 43291 pts/0    S      0:00              \_ su - somu
7 43292 pts/0    S      0:00                  \_ -bash
8 43465 pts/0    S      0:00                      \_ su
9 43471 pts/0    S      0:00                          \_ bash
10 44146 pts/0    R+     0:00                              \_ ps fax
11 44147 pts/0    D+     0:00                                  \_ [tail]
```

---

## 9.4 Understanding Memory Usage

To get an overview of the free memory available we use the command `free -m` where `-m` stands for Mega-Bytes. *Free* is the unused physical RAM. *Shared* refers to the memory used by shared libraries (shared by different programs). *buff/cache* is the combined Buffer and/or cache usage, where buffer is typically used when there's a large amount of data that needs to be committed to disk. *Available* is the amount of memory available for starting new applications without swapping.

---

```

1 # free -m
2 total      used      free      shared  buff/cache   available
3 Mem:       1823      930       198        18        694       656
4 Swap:      1907        0      1907

```

---

**Swap** contains the unused memory pages from the RAM that have been transferred to the Hard disk.

## 9.5 Understanding Performance Load

When a process is ready for execution, it's added to the runqueue, where it awaits evaluation by the scheduler to be assigned to a CPU. The amount of processes awaiting to be executed determines the performance load.

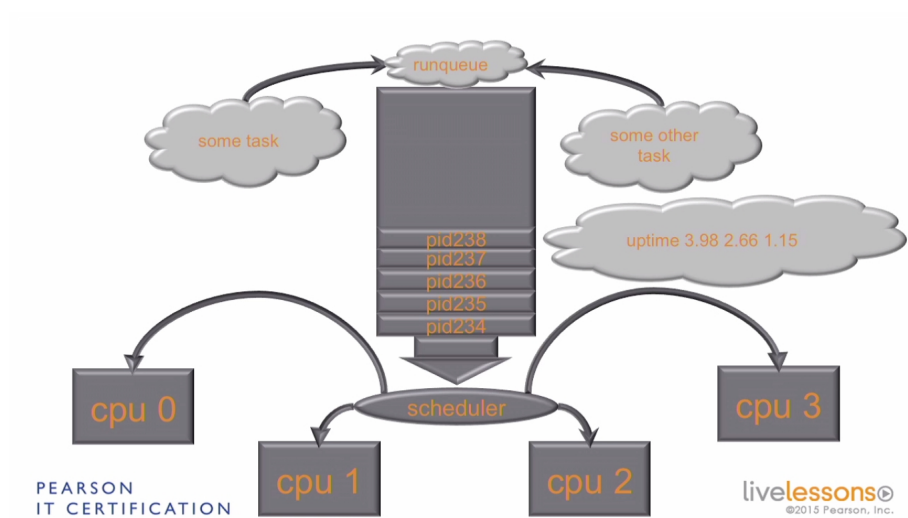


Figure 9.1: Performance Load

### 9.5.1 uptime Command

The `uptime` command shows us how long the system has been on, the number of users logged on and a snapshot of the current CPU demand for the last 1, 5 and 15 minutes, in that order. However, this number is represented for the total CPU resources being used, and not an average! Thus, for a single CPU system, uptime of 1 = 100% usage, while for a 4 CPU system, it means the CPU has been 75% idle.

The *System Load Average* is the number of processes in the runqueue that are either in runnable (ready to run/already running) state or in an interruptible state (e.g., waiting for I/O access).

---

```

1 # uptime
2 11:59:13 up 2 days, 6:02, 2 users, load average: 0.00, 0.01, 0.05

```

---

The `nproc` command shows us the number of CPU cores available to us.

---

```

1 # nproc
2 1

```

---

## 9.6 Monitoring System Activity with top

**top** shows us the top active processes on the system in real-time.

```
1 # top
2 top - 12:24:09 up 2 days, 6:27, 2 users, load average: 0.15, 0.07, 0.06
3 Tasks: 207 total, 2 running, 205 sleeping, 0 stopped, 0 zombie
4 %Cpu(s): 3.9 us, 0.7 sy, 0.0 ni, 95.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
5 KiB Mem : 1867024 total, 196468 free, 957244 used, 713312 buff/cache
6 KiB Swap: 1953788 total, 1953788 free, 0 used. 667488 avail Mem
7
8 PID USER      PR  NI   VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
9 2023 somu    20   0 1943348 246224 49292 S   5.3  13.2 61:53.62 gnome-shell
10 2250 somu    20   0 525584  12240  5984 S   2.3   0.7  0:00.61 tracker-store
11 1367 root      20   0 291800  35340 10312 S   1.3   1.9  2:53.91 X
12 48709 root      20   0 157716   2284  1536 R   0.7   0.1  0:00.06 top
13 764 root      20   0 305080   6140  4768 R   0.3   0.3  6:45.26 vmttoolsd
14 2270 somu    20   0 385944  19688 15236 S   0.3   1.1  6:02.21 vmttoolsd
15 2290 somu    39  19 637720  11280  8036 S   0.3   0.6  0:00.52 tracker-miner-f
16 48449 root      20   0      0      0      0 S   0.3   0.0  0:00.80 kworker/0:0
17 1 root       20   0 128164   6852  4084 S   0.0   0.4  0:18.42 systemd
18 2 root       20   0      0      0      0 S   0.0   0.0  0:00.22 kthreadd
19 3 root       20   0      0      0      0 S   0.0   0.0  0:11.45 ksoftirqd/0
20 5 root       0 -20      0      0      0 S   0.0   0.0  0:00.00 kworker/0:0H
21 7 root       rt    0      0      0      0 S   0.0   0.0  0:00.00 migration/0
22 8 root       20   0      0      0      0 S   0.0   0.0  0:00.00 rcu_bh
23 9 root       20   0      0      0      0 S   0.0   0.0  0:48.31 rcu_sched
24 10 root      rt    0      0      0      0 S   0.0   0.0  0:02.52 watchdog/0
25 12 root      20   0      0      0      0 S   0.0   0.0  0:00.00 kdevtmpfs
26 13 root       0 -20      0      0      0 S   0.0   0.0  0:00.00 netns
27 14 root      20   0      0      0      0 S   0.0   0.0  0:00.22 khungtaskd
28 15 root       0 -20      0      0      0 S   0.0   0.0  0:00.00 writeback
29 16 root       0 -20      0      0      0 S   0.0   0.0  0:00.00 kintegrityd
30 17 root       0 -20      0      0      0 S   0.0   0.0  0:00.00 bioset
31 18 root       0 -20      0      0      0 S   0.0   0.0  0:00.00 kblockd
32 19 root       0 -20      0      0      0 S   0.0   0.0  0:00.00 md
33 25 root      20   0      0      0      0 S   0.0   0.0  0:00.20 kswapd0
```

If we put 3 processes that all run `dd if=/dev/zero of=/dev/null` on the runqueue, and then execute the `top` command, and then press the `1` key, all the individual CPU loads are displayed. Since our VM has only one CPU, it shows up as *Cpu0*.

```
1 # top
2 ...
3 %Cpu0 : 27.6 us, 72.4 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
4 ...
5 PID USER      PR  NI   VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
6 48739 root      20   0 107948   612   516 R  32.9   0.0  0:04.59 dd
7 48737 root      20   0 107948   612   516 R  32.2   0.0  0:06.47 dd
8 48738 root      20   0 107948   608   516 R  32.2   0.0  0:04.94 dd
```

The first line of the output of the `top` command is the same as that of the `uptime` command. The `top` command also shows the Free and available memory in both physical RAM as well as Swap.

Now since there is only one cpu available, the 3 busiest processes, (the `dd` commands) have evenly distributed the CPU cycles among them (assigned by the scheduler), nearly 33% each (the rest is overheads and cycles used by other processes).

```
%Cpu(s): 26.5 us, 73.5 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

The CPU line shows the CPU Usage stats in percentages. Here, we see the CPU usage summary [Cpu(s)], the *us* stands for user-space, or processes started by the user. The *sy* shows us the system space CPU Usage, and is typically programs that directly deal with hardware. Since we've used the *dd* command that's directly copying data from one device to another, we have a high system space CPU usage.

The *id* stands for idle and shows us the percentage of time the system is idle. The *wa* shows the percentage of time the system is waiting for I/O operation completion, such as a slow disk or network.

## 9.7 Sending Signals to processes

Signals are mandatory instructions that the process can't ignore. Some of the most common Signals are **SIGTERM** and **SIGKILL**. The **SIGTERM** signal asks a process to cease its activity. If the signal doesn't work, i.e., the process doesn't obey it, then we send the **SIGKILL** signal, which terminates the process.

We can send these signals by the use of the *top* command. While *top* is running, we have to press the *k* key to initiate signal sending via *kill*. To choose the appropriate process, we enter the PID. The default signal is **SIGTERM** (a.k.a. Signal 15). The process is terminated immediately on sending the signal.

When we send Signal 9 (**SIGKILL**) the process doesn't have time to save or clean up its work, and thus the execution stops instantaneously! This means if a process is working on an open file, the **SIGKILL** signal can cause irreparable damage to it.

### 9.7.1 kill command

To send a signal directly from the command line, we use the *kill* command. To send a **SIGTERM** signal, we merely provide the PID. To send a **SIGKILL** signal, we have to provide a signal number of 9.

```
1 # ps aux | grep dd
2 ...
3 root      49575 50.3  0.0 107948   608 pts/0    R   13:25
4 [2]+  Running                  dd if=/dev/zero of=/dev/null &
5 # kill -9 49575
6 # jobs
7 [2]+  Killed                    dd if=/dev/zero of=/dev/null
```

To kill all processes matching a certain process using Signals uses:

```
1 # top
2 ...
3 PID USER      PR  NI   VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
4 49747 root        20   0 107948    608    516 R   22.8   0.0   0:05.81 dd
5 49748 root        20   0 107948    612    516 R   22.5   0.0   0:05.30 dd
6 49749 root        20   0 107948    612    516 R   22.5   0.0   0:04.91 dd
7 49746 root        20   0 107948    612    516 R   22.2   0.0   0:06.60 dd
8 ...
9 # jobs
10 [1]  Running                  dd if=/dev/zero of=/dev/null \&
```



```

11  [2]   Running          dd if=/dev/zero of=/dev/null \&
12  [3]-  Running          dd if=/dev/zero of=/dev/null \&
13  [4]+  Running          dd if=/dev/zero of=/dev/null \&
14  # killall dd
15  # jobs
16  [1]   Terminated      dd if=/dev/zero of=/dev/null
17  [2]   Terminated      dd if=/dev/zero of=/dev/null
18  [3]-  Terminated      dd if=/dev/zero of=/dev/null
19  [4]+  Terminated      dd if=/dev/zero of=/dev/null

```

---

## 9.8 Understanding Priorities and Niceness

Consider a hypothetical scenario where a bunch of people are standing in a queue to get movie tickets. There, everyone in the queue has the same priority. Now, if a lady comes along and skips the queue, we can fairly say she isn't nice. The same goes for processes in a Linux system.

The processes are born with the same priority, but their niceness can be adjusted. If the niceness is negative, the process is served first, and if positive, it lets the other processes (with lower niceness) be served before itself!

## 9.9 Changing Process Nice values

In the output of the `top` command, the *PR* column shows us the priority of each individual process. Normal processes are started with the same priority of **20**. There are certain real-time processes as well, that have a value of `rt` in their priority column.

In case we need to increase or decrease the priority of a process (e.g., higher priority to quickly complete a query, or free resources for other users by lowering priority) we adjust the niceness (*NI*) value of the process. This process is called *nicing* a process.

The niceness of a process can range from  $-20$  to  $+19$ , and the consequent value of priority is related as :

$$PR = 20 + NI$$

Thus, the priority of a process can range from 0 (most aggressive) to 39 (nicest). We can adjust the niceness in small increments instead of huge jumps (setting niceness to  $-20$ ) and see if that does our work. If not, we can change the niceness incrementally till our goals are met!

### 9.9.1 Chaning niceness from top

To renice a process from top, we can simply press `r`, then select the PID to renice and enter the new niceness value.

### 9.9.2 Changing Niceness from command line

We can also set the niceness from the command line, both while first running the process using the `nice` command, or change the nice value of a running process using the `renice` command.

---

```

1 # nice -n 10 dd if=/dev/zero of=/dev/null &
2 # top | head
3 top - 15:40:26 up 2 days,  9:44,  3 users,  load average: 1.10, 0.70, 0.71
4 Tasks: 215 total,   3 running, 212 sleeping,   0 stopped,   0 zombie
5 %Cpu(s):  9.5 us, 71.4 sy, 19.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
6 KiB Mem : 1867024 total,  233356 free,  919716 used,  713952 buff/cache
7 KiB Swap: 1953788 total, 1953788 free,      0 used.  704744 avail Mem
8
9 PID USER      PR  NI   VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
10 51465 root        30   10 107948    612    516 R  75.0   0.0    2:00.96 dd
11 1367 root        20    0 290560  34108 10412 S   5.0   1.8    3:22.26 X
12 2023 somu     20    0 1923992 226836 49300 S   5.0  12.1   66:10.12 gnome-shell
13 # renice -n -10 51465
14 51465 (process ID) old priority 10, new priority -10
15 # top | head
16 top - 15:42:18 up 2 days,  9:45,  3 users,  load average: 1.41, 0.92, 0.79
17 Tasks: 215 total,   2 running, 213 sleeping,   0 stopped,   0 zombie
18 %Cpu(s): 25.0 us, 75.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
19 KiB Mem : 1867024 total,  233440 free,  919632 used,  713952 buff/cache
20 KiB Swap: 1953788 total, 1953788 free,      0 used.  704828 avail Mem
21
22 PID USER      PR  NI   VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
23 51465 root        10  -10 107948    612    516 R  64.0   0.0    3:47.16 dd
24 1 root        20    0 128164    6852   4084 S   0.0   0.4    0:19.34 systemd
25 2 root        20    0      0      0      0 S   0.0   0.0    0:00.23 kthreadd

```

---

If however we find that we have to renice processes all the time, we might want to move some resource-hungry processes to other server(s).