SysAdmin Notes for RHCSA

Somenath Sinha

August 2017

Chapter 1

Using Essential Tools

1.1 Man Command

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

```
$ man ls
```

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

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

The numbers next to the commands indicate which seciton 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

```
$ 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.

```
$ man -k day | grep 3
daylight (3)         - initialize time conversion information
dysize (3)           - get number of days for a given year
daylight (3p)        - set timezone conversion information
gettimeofday (3p)    - get the date and time
Net::Time (3pm)      - time and daytime network client interface
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^{rd} and the 4^{th} 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:

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

Redirection:

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

```
$ find / -name "*.rpm"
$ 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.

```
$ 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.

```
$ ls / > file_list.txt
$ 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

root (/)	-	Contains all other directories.
/boot	-	Contains everything the system needs to start up
/usr	-	Contains program files
/etc	-	Contains configuration files
/home	-	Contains a user's files
/mnt	-	Used to manually mount devices
/media	-	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.

```
$ 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.

```
$ which passwd
/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!

```
$ whereis passwd
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.

```
# touch sinha
# ls
sinha
# locate sinha
/usr/share/vim/vim74/keymap/sinhala-phonetic_utf-8.vim
/usr/share/vim/vim74/keymap/sinhala.vim
# updatedb
# locate sinha
/home/somu/Documents/sinha
/usr/share/vim/vim74/keymap/sinhala-phonetic_utf-8.vim
/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 tot he same inode. Ths, 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.

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

```
8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 computers
8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 /etc/hosts
# ln -s computers newcomputers
# ls -il /etc/hosts computers newcomputers
8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 computers
8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 /etc/hosts
27604468 lrwxrwxrwx. 1 root root   9 Sep  7 19:26 newcomputers -> computers
# rm -f computers
# ls -il /etc/hosts newcomputers
8388733 -rw-r--r--. 1 root root 158 Jun  7  2013 /etc/hosts
27604468 lrwxrwxrwx. 1 root root   9 Sep  7 19:26 newcomputers -> computers
# exit
exit
$ ln /etc/shadow mydata
ln: failed to create hard link ‘mydata’ => ‘/etc/shadow’: Operation not permitted
$ ls -l /etc/shadow
----------. 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:

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

```
$ 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.

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

Note that the .tar archive only puts all the files of teh `/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.

```
$ 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.*

```
$ 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.

```
$ 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.

```
$ 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, abbbbbc
+	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

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.

```
$ 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.

```
$ 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`

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

3.2.4 cut

With the `cut` utility, we can print out a specific column form 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:

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

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

```
$ 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, captial letters next and finally the lower case letters.

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

To sort on the basis of a specific criteria:

- **-n** - Sort on the basis of actual numberical 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.

```
$ echo hello | tr a-z A-Z
HELLO
$ echo hello | tr [:lower:] [:upper:]
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*.

```
$ ps aux | grep tracker
somu      10450  0.0  0.4 469796  9000 ?        SNl  10:06   0:00
↪  /usr/libexec/tracker-miner-user-guides
somu      10465  0.0  0.6 536856 12012 ?        Sl   10:06   0:00
↪  /usr/libexec/tracker-store
somu      10611  0.0  0.7 779816 13108 ?        SNl  10:06   0:00
↪  /usr/libexec/tracker-extract
somu      10614  0.0  0.5 469800  9632 ?        SNl  10:06   0:00
↪  /usr/libexec/tracker-miner-apps
somu      10615  0.0  0.7 710160 13204 ?        SNl  10:06   0:00
↪  /usr/libexec/tracker-miner-fs
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`.

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

```
services:na-localise     5062/tcp                # Localisation access
services:na-localise     5062/udp                # Localisation access
shadow:lisa:$6$0l/zSJkh$xjJNYNnj1rPs7Fq0hDWt8VucS0nLL82XrMYpmBnLF2DrzB2npFvCwxM9MJEHgCHCwvabCgEA17LK2aU0h9FIT/:1741
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:

```
$ ps aux | wc
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.

```
$ grep lisa * 2> /dev/null
group:lisa:x:1001:
passwd:lisa:x:1001:1001::/home/lisa:/bin/bash
services:na-localise     5062/tcp                # Localisation access
services:na-localise     5062/udp                # Localisation access
$ grep -l lisa * 2> /dev/null
group
passwd
services
```

3.3.3 grep -i

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

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

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

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

3.3.5 grep -v

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

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

3.4 sed and awk basics

3.4.1 sed

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

sed q

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

```
$ sed 2q /etc/passwd
root:x:0:0:root:/root:/bin/bash
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.

```
$ sed -n /^root/p /etc/passwd
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 :

```
$ cat names
Somu
Arpi
Neha
Santy
Debu
$ sed -i 's/Santy/Dickwad/g' names
$ cat names
Somu
Arpi
Neha
Dickwad
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!

```
$ ps aux | grep 'gdm'
root       1117  0.0  0.1 480248  4688 ?        Ssl  09:20   0:00 /usr/sbin/gdm
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
root       1718  0.0  0.1 528944  5848 ?        Sl   09:20   0:00 gdm-session-worker
↪  [pam/gdm-password]
gdm        1737  0.0  0.1 458088  4152 ?        Sl   09:20   0:00 ibus-daemon --xim
↪  --panel disable
gdm        1741  0.0  0.1 373560  5424 ?        Sl   09:20   0:00 /usr/libexec/ibus-dconf
gdm        1745  0.0  0.2 438152  7772 ?        Sl   09:20   0:00 /usr/libexec/ibus-x11
↪  --kill-daemon
somu      12218  0.0  0.0 112664   972 pts/0    R+   10:47   0:00 grep --color=auto gdm
$ ps aux | grep 'gdm' | cut -f 2
root       1117  0.0  0.1 480248  4688 ?        Ssl  09:20   0:00 /usr/sbin/gdm
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
root       1718  0.0  0.1 528944  5848 ?        Sl   09:20   0:00 gdm-session-worker
↪  [pam/gdm-password]
gdm        1737  0.0  0.1 458088  4152 ?        Sl   09:20   0:00 ibus-daemon --xim
↪  --panel disable
gdm        1741  0.0  0.1 373560  5424 ?        Sl   09:20   0:00 /usr/libexec/ibus-dconf
gdm        1745  0.0  0.2 438152  7772 ?        Sl   09:20   0:00 /usr/libexec/ibus-x11
↪  --kill-daemon
somu      12226  0.0  0.0 112664   968 pts/0    R+   10:48   0:00 grep --color=auto gdm
$ ps aux | grep 'gdm' | awk '{ print $2 }'
1117
1333
1718
1737
1741
1745
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>`.

```
$ ssh 192.168.152.129
somu@192.168.152.129's password:
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>`.

```
$ 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 *root* 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.

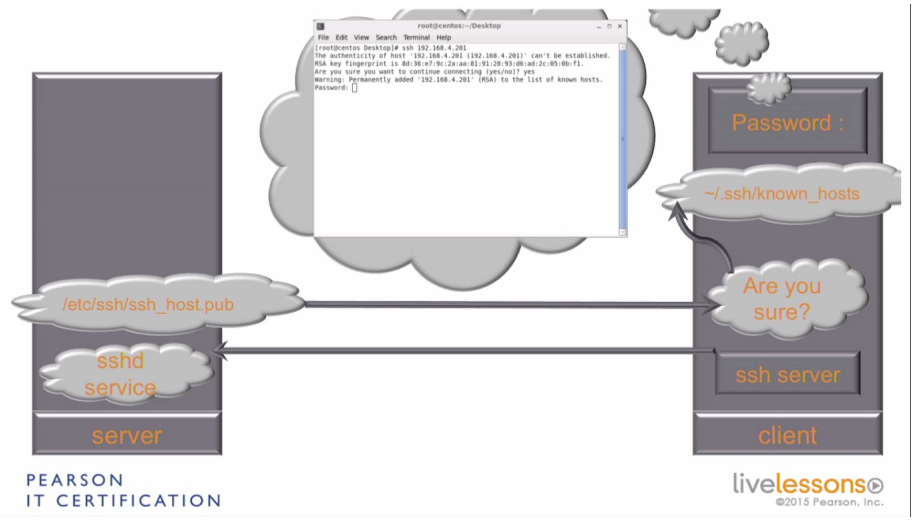

Figure 4.1: Server Authentication procedure

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.

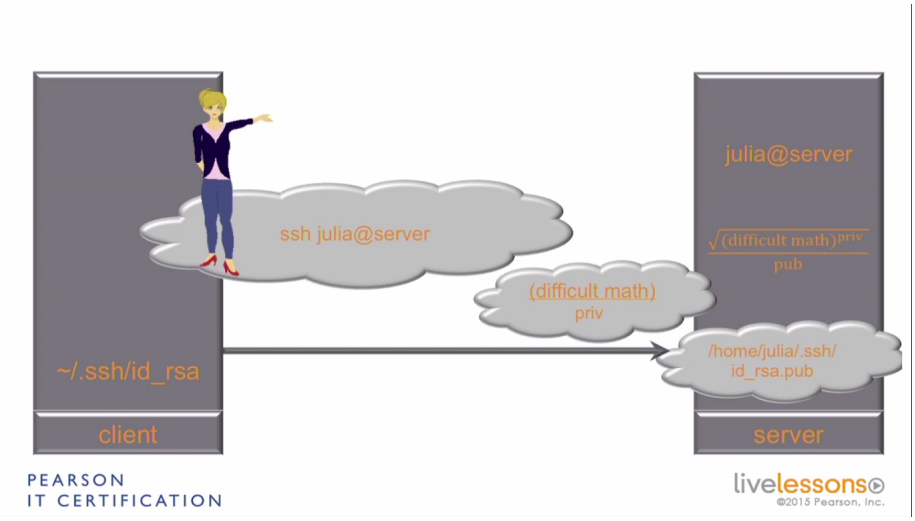

Figure 4.2: Public Key Client Authentication without password.

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.

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/somu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/somu/.ssh/id_rsa.
Your public key has been saved in /home/somu/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:0C5uEHnAgJvzFEc0ulfL1YSygT/YF5UtL9lweTfPDAc somu@vmPrime.somusysadmin.com
The key's randomart image is:
+---[RSA 2048]----+
| ..==.   ooo .E. |
|. ..++o.oo+ + o.o|
| o.oo+++o..B . *o|
|+ ...===. o o   +|
| +. o +oS  .     |
|  .. o .         |
|      o          |
|     .           |
|                 |
+----[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`.

```
$ ssh-copy-id 192.168.152.129
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any
↪  that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it
↪  is to install the new keys
somu@192.168.152.129's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '192.168.152.129'"
and check to make sure that only the key(s) you wanted were added.

$ ssh 192.168.152.129
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.

```
$ 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.

```
$ scp -P 22 names 192.168.152.129:~
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
-e	Expiration date in the format YYYY-MM-DD. Sets the date on which the user account will be disabled.
-c	Comment that sets the contents of the GECOS field.
-s	Sets the default shell of the user. For example, a C programmer can use a shell such as TCSH.

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

id command

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

```
$ id
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.|

```
$ id lisa
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
-g	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
-g	Force assign the GID as the new default group of the user.
-G	Erase older list of supplementary groups and assign the given groups as the supplementary group of he user.
-a	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.

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

5.6 User and Group configuration files

Some of the important configuration files are:

Option	Description
/etc/passwd	Contains several details of the user, other than the password.
/etc/shadow	Contains the password hash and password properties for the user.
/etc/group	Contains the names of all the groups along with a list of all users in them.
/etc/login.defs	Contains the values (definitions) of several parameters used to create the user, such as password max days, min days, etc.
/etc/default/useradd	Contains the default values for several useradd parameters.
/etc/skel	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
-d	Delete the current password.
-l	Lock the current password.
-u	Unlock the current password.
-e	Expire the current password - force user to change password during next login.
-x	Set the maximum lifetime of the password.
-n	Set the maximum lifetime of the password.
-w	Set days before expiration the user is warned.
-i	Set days after expiration the user account becomes inactive.

Locking and Unlocking passwords

```
$ sudo passwd -l laura
Locking password for user laura.
passwd: Success
$ su - laura
Password:
su: Authentication failure
$ sudo cat /etc/shadow | grep laura
laura:!!$6$0zDhsJet$q2...KRVKv8D2.:17486:0:99999:7::17531:
$ sudo passwd -u laura
Unlocking password for user laura.
passwd: Success
$ sudo cat /etc/shadow | grep laura
laura:$6$0zDhsJet$q2...KRVKv8D2.:17486:0:99999:7::17531:
$ su - laura
Password:
Last login: Thu Nov 16 13:40:45 IST 2017 on pts/0
$ whoami
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
-l	List all password aging information.
-E	Set the account expiration date.
-m	Set the maximum lifetime of the password.
-M	Set the maximum lifetime of the password.
-W	Set days before expiration the user is warned.
-I	Set days after expiration the user account becomes inactive.

Setting the account expiration date

```
$ sudo chage -E 2017-12-31 laura
[sudo] password for somu:
[somu@cliServer ~]$ sudo cat /etc/shadow | grep laura
laura:$6$0zDhsJet$q2...KRVKv8D2.:17486:0:99999:7::17531:
$ sudo chage -l laura
Last password change                                    : Nov 16, 2017
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : Dec 31, 2017
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
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* $\rightarrow$ Domain Component) and for leaf objects, it becomes `cn=<objectName>` (*cn* $\rightarrow$ 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.

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

```
libdl.so.2 => /lib64/libdl.so.2 (0x00007f85ca16e000)
libcap-ng.so.0 => /lib64/libcap-ng.so.0 (0x00007f85c9f68000)
libpcre.so.1 => /lib64/libpcre.so.1 (0x00007f85c9d06000)
/lib64/ld-linux-x86-64.so.2 (0x0000558e5f0bd000)
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 *Plugable 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> <arugments>`.

Service Type

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

PAM Module Controls

Control	Description
requisite	Immediately causes failure when the module returns a status that isn't 'success'.
required	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.
sufficient	If a *sufficient* 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.
optional	Only causes failure if the rule stack contains only optional modules and all fail.
include	For the given service type, include all lines of that type from the provided configuration file.
substack	Same as *include* but when *done* and *die* actions are evaluated, they only cause skipping of the substack.

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

```
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 :

```
auth        sufficient    pam_unix.so nullok try_first_pass
...
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.

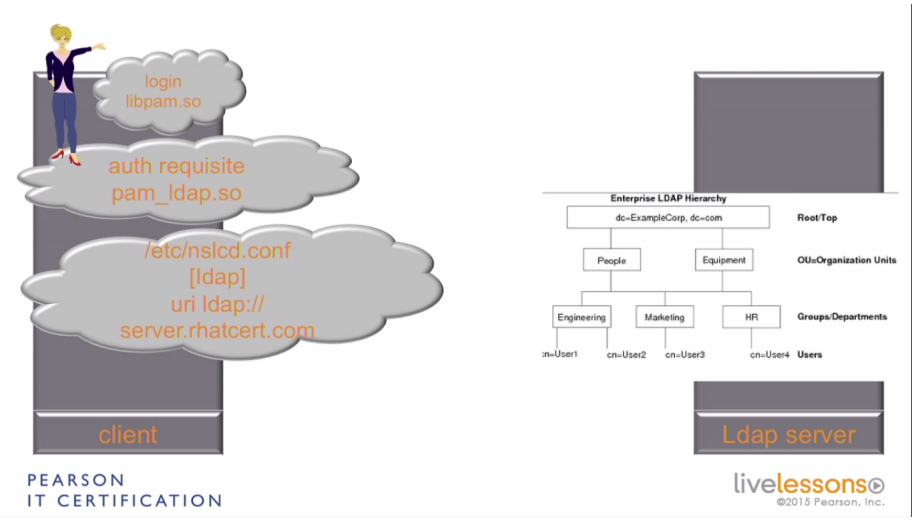

Figure 6.1: LDAP Authentication

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 nscd

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/slapd.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:

```
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.

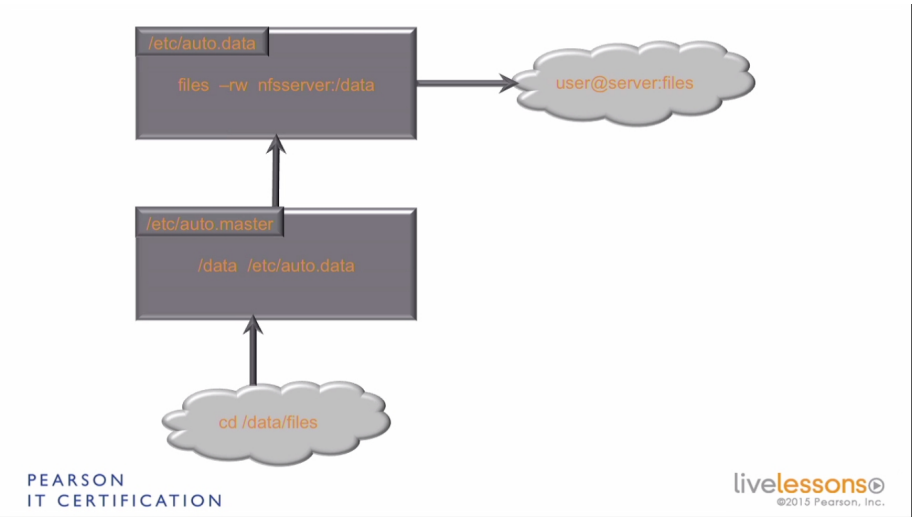

Figure 6.2: NFS Automount

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:

```
/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.

```
*         -fstype=cifs,username=ldapusers,password=password\
                  ://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.

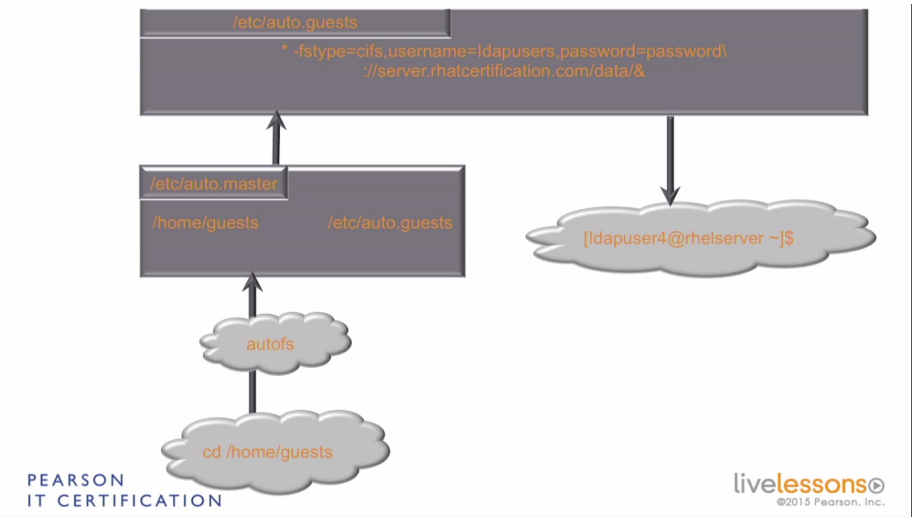

Figure 6.3: Samba Automount

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:

```
/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` :

```
[data]
comment = LDAP Users' home directories
path = /home/guests
public = yes
writable = no
```

6.4.1 NFS Server Automounting

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

```
*        -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.

```
# yum search nfs
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
* base: mirror.digistar.vn
* extras: mirror.dhakacom.com
* updates: mirror.digistar.vn
================================== N/S matched: nfs ==================================
libnfsidmap.i686 : NFSv4 User and Group ID Mapping Library
libnfsidmap.x86_64 : NFSv4 User and Group ID Mapping Library
libnfsidmap-devel.i686 : Development files for the libnfsidmap library
libnfsidmap-devel.x86_64 : Development files for the libnfsidmap library
nfs-utils.x86_64 : NFS utilities and supporting clients and daemons for the kernel NFS
↪  server
nfs4-acl-tools.x86_64 : The nfs4 ACL tools
nfsometer.noarch : NFS Performance Framework Tool
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 :

```
/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.

```
$ systemctl start nfs
```

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

```
# systemctl status -l nfs
nfs-server.service - NFS server and services
Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled; vendor preset:
↪  disabled)
Active: active (exited) since Tue 2017-11-21 10:00:04 IST; 24s ago
Process: 3178 ExecStart=/usr/sbin/rpc.nfsd $RPCNFSDARGS (code=exited, status=0/SUCCESS)
Process: 3173 ExecStartPre=/bin/sh -c /bin/kill -HUP `cat /run/gssproxy.pid`
↪  (code=exited, status=0/SUCCESS)
Process: 3172 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=1/FAILURE)
Main PID: 3178 (code=exited, status=0/SUCCESS)
CGroup: /system.slice/nfs-server.service

Nov 21 10:00:04 ldapserver.somuvmnet.local systemd[1]: Starting NFS server and
↪  services...
Nov 21 10:00:04 ldapserver.somuvmnet.local exportfs[3172]: exportfs: Failed to stat
↪  /data: No such file or directory
Nov 21 10:00:04 ldapserver.somuvmnet.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.

```
# showmount -e localhost
Export list for localhost:
/data *
# mount localhost:/data /mnt/nfs
# ls /nfs
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` :

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

This file contains the following mounting details:

```
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.

```
# cd nfs
# ls
# ls -lha
total 0
drwxr-xr-x. 2 root root  0 Nov 21 11:59 .
drwxr-xr-x. 3 root root 17 Nov 21 11:59 ..
# cd files
# ls
nfs1  nfs2  test1
# cd ..
# ls
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:

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

```
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 start_tls is given, via the use of StartTLS, an insecure connection is upgraded to a secure one.	start_tls
tls_cacertdir	Location of the downloaded certificate of the LDAP Server.	`/etc/openldap/cacerts`

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` :

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

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
d	A directory
-	A regular file
l	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
r	Read the file/directory
w	Write to the file/directory
x	Execute the file/Access to the directory
-	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:

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

The user 'root' has `rwx` 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.

```
# ls -l
total 0
drwxr-xr-x. 2 root root 6 Nov 21 14:50 account
drwxr-xr-x. 2 root root 6 Nov 21 14:50 sales
# chgrp sales sales
# chgrp account account
# ls -l
total 0
drwxr-xr-x. 2 root account 6 Nov 21 14:50 account
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.

```
# chown lori account
# chown lisa sales
# ls -l
total 0
drwxr-xr-x. 2 lori account 6 Nov 21 14:50 account
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**
r	Opening and outputting a file.	List files in a directory. The user **can't** read all files in that directory. For that, he needs read access on the individual files.
w	Modify contents of the file	Modify contents of the directory, i.e., add, delete, move, etc. files in that directory.
x	If the contents of the file is executable, the user can execute it.	User can `cd` 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 th 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**
r	=	4
w	=	2
x	=	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>
```

```
$ chmod 750 myFile
$ chmod u+x,g-r,o-wx myFile2
$ 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:

```
# ls -l
total 0
drwxr-xr-x. 2 lori account 6 Nov 21 14:50 account
drwxr-xr-x. 2 lisa sales   6 Nov 21 14:50 sales
# chmod 760 account
# chmod g+w-x,o-rx sales
# ls -l
total 0
drwxrw----. 2 lori account 6 Nov 21 14:50 account
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**
Set User ID	`u+s`	4	Run executable file as Owner	—
Set Group ID	`g+s`	2	Run executable file with permissions of Group-Owner	Inherit group ownership to all newly created items in the folder.
Sticky Bit	`+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:

```
#!/bin/bash
echo "Hi, do you wanna play a game?!"
read

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.

```
# chmod u+s game
# ls -l | grep game
-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:

```
# find / -perm /4000
find: ‘/proc/2998/task/2998/fd/6’: No such file or directory
find: ‘/proc/2998/task/2998/fdinfo/6’: No such file or directory
find: ‘/proc/2998/fd/6’: No such file or directory
find: ‘/proc/2998/fdinfo/6’: No such file or directory
/usr/bin/fusermount
/usr/bin/su
/usr/bin/umount
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/staprun
```

```
/usr/bin/mount
/usr/bin/pkexec
/usr/bin/crontab
/usr/bin/passwd
/usr/sbin/pam_timestamp_check
/usr/sbin/unix_chkpwd
/usr/sbin/usernetctl
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/lib64/dbus-1/dbus-daemon-launch-helper
/usr/libexec/abrt-action-install-debuginfo-to-abrt-cache
/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:

```
# ls -l /etc/shadow
----------. 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:

```
#ls -l
total 0
drwxrwx---. 2 lori account 6 Nov 25 17:35 account
drwxrwx---. 2 lisa sales   6 Nov 25 17:26 sales
# su - lisa
Last login: Sat Nov 25 17:31:57 IST 2017 on pts/0
$ cd /data/account/
$ touch lisa1
$ ls -l
total 0
-rw-rw-r--. 1 lisa lisa 0 Nov 25 17:35 lisa1
```

The file */data/account/lisa1* has it's 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.

```
$ su - laura
Password:
Last login: Thu Nov 16 13:42:44 IST 2017 on pts/0
$ cd /data/account
$ echo "Added a line" >> lisa1
-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!

```
# ls -l
total 0
drwxrwx---. 2 lori account 19 Nov 25 17:35 account
drwxrwx---. 2 lisa sales    6 Nov 25 17:26 sales
# chmod g+s account
# ls -l
total 0
drwxrws---. 2 lori account 19 Nov 25 17:35 account
drwxrwx---. 2 lisa sales    6 Nov 25 17:26 sales
# su - lisa
Last login: Sat Nov 25 17:35:39 IST 2017 on pts/0
$ cd /data/account
$ touch lisa2
$ ls -l
total 0
-rw-rw-r--. 1 lisa lisa    0 Nov 25 17:35 lisa1
-rw-rw-r--. 1 lisa account 0 Nov 25 17:45 lisa2
$ echo "line added by lisa" >> lisa2
$ su - laura
Password:
Last login: Sat Nov 25 17:41:55 IST 2017 on pts/0
$ cd /data/account
$ echo "line added by laura" >> lisa2
$ cat lisa2
line added by lisa
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.

```
# ls -l
total 0
drwxrws---. 2 lori account 32 Nov 25 17:45 account
drwxrwx---. 2 lisa sales    6 Nov 25 17:26 sales
# ls -l account
total 4
-rw-rw-r--. 1 lisa lisa     0 Nov 25 17:35 lisa1
-rw-rw-r--. 1 lisa account 39 Nov 25 17:46 lisa2
# chmod +t account
# ls -l
total 0
drwxrws--T. 2 lori account 32 Nov 25 17:45 account
drwxrwx---. 2 lisa sales    6 Nov 25 17:26 sales
# su - laura
Last login: Sat Nov 25 17:53:25 IST 2017 on pts/0
$ cd /data/account
$ ls -l
total 4
-rw-rw-r--. 1 lisa lisa     0 Nov 25 17:35 lisa1
-rw-rw-r--. 1 lisa account 39 Nov 25 17:46 lisa2
```

```
$ rm -f lisa1
rm: cannot remove ‘lisa1’: Operation not permitted
$ su - lori
Password:
$ cd /data/account
$ rm -f lisa1
$ ls -l
total 4
-rw-rw-r--. 1 lisa account 39 Nov 25 17:46 lisa2
```

However, the user laura 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'.

```
# mkdir test
# ls -l
total 0
drwxrws--T. 2 lori account 19 Nov 25 17:57 account
drwxrws--T. 2 lisa sales    6 Nov 25 17:26 sales
drwxr-xr-x. 2 root root     6 Nov 25 18:15 test
# chmod 3770 *
# ls -l
total 0
drwxrws--T. 2 lori account 19 Nov 25 17:57 account
drwxrws--T. 2 lisa sales    6 Nov 25 17:26 sales
drwxrws--T. 2 root root     6 Nov 25 18:15 test
# chmod o+x,g-x test
# ls -l
total 0
drwxrws--T. 2 lori account 19 Nov 25 17:57 account
drwxrws--T. 2 lisa sales    6 Nov 25 17:26 sales
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 seperate 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.

```
$ 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:

```
$ setfact -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.

```
# getfacl account
file: account
owner: lori
group: account
flags: -st
user::rwx
group::rwx
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
-m	-	Modify, followed immediately by what needs to be modified.
-R	-	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:

```
# setfacl -R -m g:sales:r account
# getfacl account
file: account
owner: lori
group: account
flags: -st
user::rwx
group::rwx
group:sales:r--
mask::rwx
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 no need to be applied recursively.

```
# setfacl -m d:g:sales:r account
getfacl account
file: account
owner: lori
group: account
flags: -st
user::rwx
group::rwx
group:sales:r--
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:sales:r--
default:mask::rwx
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:

```
# cd account/
# mkdir 2017
# getfacl 2017
file: 2017
owner: root
group: account
```

```
flags: -s-
user::rwx
group::rwx
group:sales:r--
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:sales:r--
default:mask::rwx
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).

```
# cd 2017/
# touch testFile
# getfacl testFile
file: testFile
owner: root
group: account
user::rw-
group::rwx                      #effective:rw-
group:sales:r--
mask::rw-
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).