The Notebook of LINUX

Huy Bui

PL

THE PUBLISHER

Contents

Chapter 1

The Shell

1.1 Basic commands and tips

Table 1.1: Shell shortcuts

Shortcuts	**Explanation**
`Ctrl+Alt+F1-F6`	Start or switch to the first (to sixth) session
Ctrl + Alt + F7	Switch to GUI session
Shift + PgUp	Scroll up
Shift + PgDown	Scroll down
Ctrl + A	Jump to the beginning of the line
Ctrl + E	Jump to the end of the line
Ctrl + U	Clear from the cursor to the beginning of the line
Ctrl + K	Clear from the cursor to the end of the line
!string	Re-execute a recent command by matching the command name
!65	Re-execute a 65th command in the history list
Ctrl + R	Search the history list of commands for a pattern
Esc + .	Copy the last argument of the previous command to the current one
Ctrl + C	Stop the running command
Ctrl + Z	Suspend the running command

Command substitution: allows output of a command to replace itself. It occurs when a command is enclosed with a dollar sign and a pair of parentheses `$(command)`, or backticks `` `command` ``. The form with backticks cannot be nested inside another pair of backticks. In contrast, `$(command)` can nest multiple expansions inside each other.

```
admin $> echo Today is ‘date‘
admin $> cat $(fgrep -l ens32 /var/*)
admin $> echo Today is $(date)
```

Protecting argument from expansion: To ignore meta-character special meanings, we use *escaping* and *quoting*. The backslash \ is an escape character that protects the single following character from special interpretation. The single quotes ' ' and double quotes " " are used to enclose a string. The double

Table 1.2: Basic Files and Directory commands

Activity	**Command**	**Explain**
Copy a file	`cp file1 file2`	Copy `file1` to a desire location and change name to `file2`
Copy multiple files	`cp file1 file2 file3 dir`	Copy `file1`, `file2`, and `file3` to directory `dir`. Note that the last argument must be a directory.
Move a file	`mv file1 file2`	If `file2` resides in a different directory than `file1`, then `file1` is moved to the new location
Rename a file	`mv file1 file2`	If `file2` and `file1` are in the same directory, then `file1` is renamed as `file2`
Move multiple files	`mv file1 file2 file3 dir`	Move `file1`, `file2`, and `file3` to directory `dir`
Create directory	`mkdir dir`	Create directory `dir` under the current directory
Create directory path	`mkdir -p ~/dir1/dir2/dir3`	Create `dir1`,`dir2`, and `dir3` to match the specified directory structure `~/dir1/dir2/dir3`
Copy directories	`cp -r dir1 dir2 dir3 dir4`	Recursively copy `dir1`,`dir2`, and `dir3` to `dir4`
Move directories	`mv dir1 dir2 dir3 dir4`	Move `dir1`,`dir2`, and `dir3` to `dir4`
Remove directories	`rm -rf dir1 dir2 dir3`	Remove (without questioning) `dir1`,`dir2`, and `dir3`

Table 1.3: Common meta-characters and pattern classes

Characters	**Explanation**
`*`	Any string of zero or more characters
`?`	Any single character
`~`	The current user's home directory
`~kevin`	The current Kevin's home directory
`~+`	The current working directory
`~-`	The previous working directory
`[abc...]`	Any one character in the enclosed class
`[!abc...]`	Any one character *not* in the enclosed class
`[^abc...]`	Any one character *not* in the enclosed class
`[[:alpha:]]`	Any alphabetic character
`[[:alnum:]]`	Any alphabetic character or digit
`[[:lower:]]`	Any lower-case character
`[[:upper:]]`	Any upper-case character
`[[:punct:]]`	Printable character (not a space or alphanumeric)
`[[:digit:]]`	Any digit, 0 – 9
`[[:space:]]`	Any one whitespace character (space, tab, newline, carriage return, form feeds)
`{a,b,2,3}`	Any characters in the brace
`{1..8}`	Any single digit from 1 to 8
`{a..d}`	Any single characters from a to d (a, b, c, d)

quotes suppress shell expansions and meta-characters but still allows command and variable substitutions. The single quotes interpret all text literally, it suppress shell expansions and meta-characters as well as command and variable substitutions.

```
kevin$> echo "**hostname is $(host)**"
**hostname is rtracy**

kevin$> echo '**hostname is $(host)**'
**hostname is $(host)**
```

1.2 Configuration files

Shell configuration files are scripts that execute when a shell starts. There are two types of shells:

- Login shell: run when the system starts and is using only the CLI as the user interface.
- Non-login shell run when a shell session is opened from within the GUI.

Login shells execute only one of the following configuration files in order (means that if a file is found, all files followed it are abandoned):

1. `/etc/profile`
2. `~/.bash_profile`
3. `~/.bash_login`
4. `~/.profile`

Table 1.4: Shell configuration files

File	**Type**	**Explanation**
`/etc/profile.d/`	login	Configurations in this directory are applied for all users. Do not modify the system's files, instead create your own file (for example: `MyConf.sh`) and add custom configurations to it.
`~/.bash_profile`	login	Contains shell preferences for individual user.
`~/.profile`	login	Contains environment variables for individual users.
`~./bash_login`	login	Store commands executed when a user logs in.
`~./bash_logout`	login	Store commands executed when a user logs out.
`/etc/bashrc`	non-login	Contain shell preferences for all users.
`~/.bashrc`	non-login	Pass variables between shells.

Chapter 2

Getting help

2.1 The man command

A manual (man) page is a text-based help file for a specific command, service, or configuration file. These pages are stored in `/usr/man` or `/usr/share/man`. The `MANPATH` environment variable is used to identify location of man pages. Each man page contains information about a specific type of files or commands, which have become the *sections* listed below:

1. User commands (both executable and shell programs)
2. System calls
3. Library functions
4. Special files (such as device files)
5. File formats (configuration files and structures)
6. Games
7. Conventions, standards, miscellaneous (protocols, file systems)
8. System administration and privileged commands
9. Linux kernel API

To distinguish identical topic names in different sections, man page references include the section number in parentheses after the topic. For example, `passwd(1)` describes the command to change passwords, while `passwd(5)` explains the `/etc/passwd` file format for storing local user accounts.

To read specific man pages, user `man <topic>` command. For example, `man passwd` displays man pages about `passwd` command. To display the man page topic from a specific section, please include the section number argument, for example, `man 5 passwd` displays `passwd(5)` man pages. The command `man -k <string>` performs a keyword search, which displays a list of keyword-matching man pages with section numbers.
Each man page is broken into sections. Each section is designed to provide specific information about a command. The table 2.2 describes some of the more common sections that you will find in man pages.
The `-t` option of `man` command prepares a man page for printing in PostScript format (`.ps` extension). To view this PostScript file, use `evince` command:

```
$ man -t passwd > passwd.ps
$ evince passwd.ps
$ evince -w passwd.ps
$ evince -i 3  passwd.ps
$ lp passwd.ps -P -2 -3
```

While the normal `evince` mode allows full-screen viewing, the preview mode (`-w` option) provides quick browsing and printing preview. The `-i` option is used to specify the starting page. The `lp` command sends page 2 and 3 of this PostScript file to the default printer for real printing.

Table 2.1: Navigating man pages

Command	**Result**
`Spacebar`, `PgDown`	Scroll down one screen
`PgUp`, `b`	Scroll up one screen
`d`	Scroll down one half-screen
`u`	Scroll up one half-screen
Up/Down arrow	Scroll up/down one line
`/string`	Search forward for `string`
`n`	Repeat the previous search forward
`N`	Repeat the previous search backward
`g`	Go to the start of the man page
`G`	Go to the end of the man page
`q`	Exit man page and return to the command shell prompt

Table 2.2: Sections in each man page

Section	**Purpose**
NAME	Provides the name of the command and a very brief description.
SYNOPSIS DESCRIPTION	A brief summary of the command or function's interface. A summary of how the command line syntax of the program looks. Provides a more detailed description of the command.
OPTIONS	Lists the options for the command as well as a description of how they are used. Often this information will be found in the DESCRIPTION section and not in a separate OPTIONS section.
FILES	Lists the files that are associated with the command as well as a description of how they are used. These files may be used to configure the command's more advanced features. Often this information will be found in the DESCRIPTION section and not in a separate FILES section.
AUTHOR	The name of the person who created the man page and (sometimes) how to contact the person.
REPORTING BUGS	Provides details on how to report problems with the command.
COPYRIGHT	Provides basic copyright information.
SEE ALSO	Provides you with an idea of where you can find additional information. This also will often include other commands that are related to this command.

2.2 The pinfo command

GNU Project developed a different online documentation system, known as GNU info. Info documentations are structured as hyperlinked info nodes and more in-depth than man pages. Info nodes for a particular topic are browsed with the `pinfo <topic>` command (Table 2.3).

Table 2.3: Navigating info pages

Command	**Result**
`Spacebar`, `PgDown`	Scroll down one screen
`PgUp`, `b`	Scroll up one screen
`d`	Display the directory of the topics
`u`	Display the parent node
`HOME`	Display the top of a topic
Up/Down arrow	Scroll up/down one hyperlink
`ENTER`	Open topic at cursor location
`/string`	Search forward for `string`
`n`	Display the next node in the topic
`/` then `ENTER`	Repeat the previous search forward
`p`	Display the previous node in the topic
`q`	Exit man page and return to the command shell prompt

2.3 Reading documentation in /usr/share/doc

Bundling documentation with RPM packages are stored in the directory `/usr/share/doc/`. When the package is installed, files recognized as documentation are moved to `\usr/share/doc/<pkg-name>`. Use `less` or `gedit` to read text-based file. Use a browser to read a PDF- or HTML-based documentation.

```
[student@serverX doc]$ less vim-common-*/README.txt
[student@serverX doc]$ firefox grub2-common-2.02/grub.html
```

Some software provides its document as a separate package. For example, the `gnuplot` program has the extra `gnuplot-doc` package, which must be installed separately. Use `yum` to display only those packages that contain `-doc`, or `documentation`.

Note! The `kernel-doc` package has kernel, driver, tuning, and advanced configuration information.

2.4 Use redhat-support-tool

2.4.1 Introduction

Red Hat provides a text console interface to the subscription-based Red Hat Customer Portal. Internet access is required to have access to this service. This tool is text-based for use from any terminal or SSH connection; no graphical interface is provided. The `redhat-support-tool` command can be used as either interactive shell (by default) or invoked as individually executed commands. When first invoked, this tool requires Red Hat Access subscriber login information. To avoid repetitively supplying this account information, a user can store it in his/her home directory:

```
~/.redhat-support-tool/redhat-support-tool.conf
```

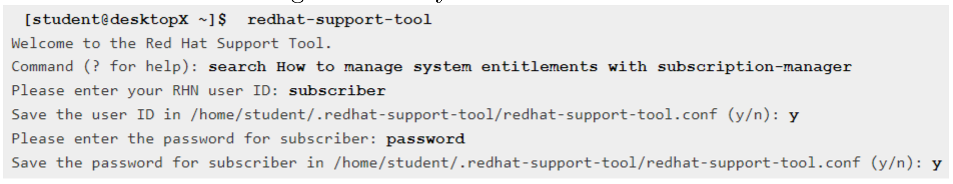
If a Red Hat account is shared by many people, the `--global` option can save account information to `/etc/` directory:

```
/etc/redhat-support-tool.conf
```

2.4.2 Red Hat Knowledgebase

The `redhat-support-tool` command allows subscribers to search and display the same Knowledgebase content seen when on the Red Hat Customer Portal. The Knowledgebase permits keyword searches (similar to to the `man` command). Users can enter error codes, syntax from log files, or any mix of keywords to produce a list of relevant solution documents.

Figure 2.1: Keyword search feature



```
 [student@desktopX ~]$  redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help): search How to manage system entitlements with subscription-manager
Please enter your RHN user ID: subscriber
Save the user ID in /home/student/.redhat-support-tool/redhat-support-tool.conf (y/n): y
Please enter the password for subscriber: password
Save the password for subscriber in /home/student/.redhat-support-tool/redhat-support-tool.conf (y/n): y
```

Red Hat subscribers can also locate online articles using Knowledgebase document ID:

```
$ redhat-support-tool kb 253273 | less
```

2.4.3 Bug report

Another benefit of the `redhat-support-tool` command is that users can gather relevant information for a bug report. In the bug report, the problem and its symptoms must be clarified. Background information (which product is affected, version, etc.) is also required. For kernel problems, Red Hat uses kernel crash dump core files to create and extract *backtrace*[1] to provide onsite diagnostic and open a support case. Therefore, system's **kdump** crash dump or a digital photo of the kernel backtrace display must be included in the SoS report.

Red Hat uses four severity levels to classify issues. The **Urgent** and **High** severity problem reports should be followed by a phone cal to the relevant local support center.

1. **Urgent:** A problem that severely impacts your use of the software in a production environment (e.g. loss of data, production systems are not functioning). The situation *halts* your business operations and *no procedural* workaround exists.

2. **High:** A problem where the software is *functioning* but your use in a production environment is *severely reduced*. The situation is causing a *high impact* to portions of your business operations and *no procedural* workaround exists.

3. **Medium:** A problem that involves *partial*, *non-critical* loss of use of the software. For production environments, there is a *medium-to-low* impact on your business, but your business *continues to function*, including a procedural workaround. For development environments, your business *migrates into production* or *no longer to continue*.

4. **Low:** A general usage question, reporting of a documentation error, or recommendation for a future product enhancement or modification. For production environments, there is a low-to-no impact on your business. For development environments, there is a medium-to-low impact but your business continues to function.

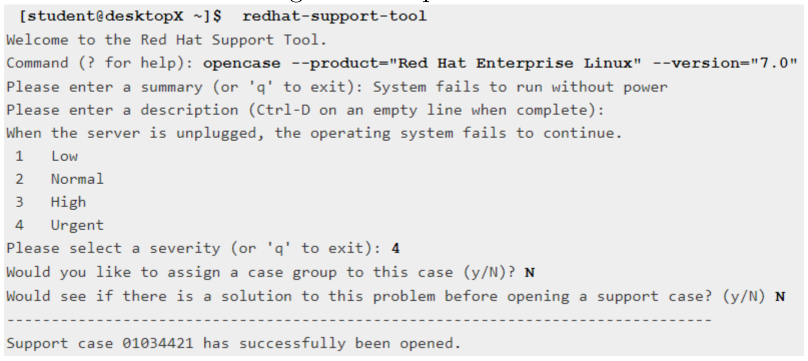
When support cases are opened, users may create a diagnostic reports using `sosreport` sub-command of `redhat-support-tool`. This command stores the report in `/var/tmp/` directory as an `xz` archive.

```
# sosreport
# cd /var/tmp/
# tar -xvJf sosreport-*.tar.xz
# cd sosreport-yourname.01034221-0164831865
```

Users can also upload and attache files to online cases. Case detail including *product*, *version*, *summary*, *description*, *severity*, and *case group*. In the picture 2.2, the `--product` and `--version` are specified using `opencase` command.

[1]a report of active stack frames at the point of the crash

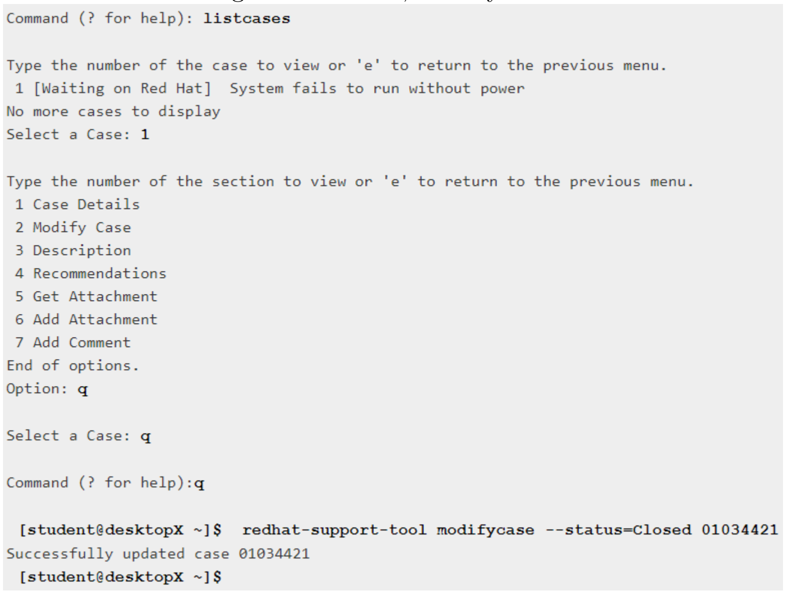
Figure 2.2: Open a case



```
 [student@desktopX ~]$  redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help): opencase --product="Red Hat Enterprise Linux" --version="7.0"
Please enter a summary (or 'q' to exit): System fails to run without power
Please enter a description (Ctrl-D on an empty line when complete):
When the server is unplugged, the operating system fails to continue.
 1   Low
 2   Normal
 3   High
 4   Urgent
Please select a severity (or 'q' to exit): 4
Would you like to assign a case group to this case (y/N)? N
Would see if there is a solution to this problem before opening a support case? (y/N) N
-------------------------------------------------------------------------------
Support case 01034421 has successfully been opened.
```

Including diagnostic information when a support case is first created contributes to quicker problem resolution. The `sosreport` command generates a compressed tar archive of diagnostic information gathered from the running system. If a current SoS report is not already prepared, an admin can generate and attach one later, using the `addattachment` command. Support cases can also be modified by the subscribers (picture 2.3).

Figure 2.3: View, Modify a case



```
Command (? for help): listcases

Type the number of the case to view or 'e' to return to the previous menu.
 1 [Waiting on Red Hat]  System fails to run without power
No more cases to display
Select a Case: 1

Type the number of the section to view or 'e' to return to the previous menu.
 1 Case Details
 2 Modify Case
 3 Description
 4 Recommendations
 5 Get Attachment
 6 Add Attachment
 7 Add Comment
End of options.
Option: q

Select a Case: q

Command (? for help):q

 [student@desktopX ~]$  redhat-support-tool modifycase --status=Closed 01034421
Successfully updated case 01034421
 [student@desktopX ~]$
```

The `redhat-support-tool` also support log file analysis. The tool's `analyze` command log files of many types. The analysis result can be parsed to recognize problem symptoms.

Chapter 3

Text files

3.1 Redirecting output

Table 3.1: Channels (File descriptor)

Number	**Channel**	**Description**	**Default connection**	**Usage**
0	`stdin`	Standard input	keyboard	read only
1	`stdout`	Standard output	terminal	write only
2	`stderr`	Standard error	terminal	write only
3+	`filename`	Other files	none	read and/or write only

Table 3.2: Output redirection operators

Usage	**Explanation**
`>file`	redirect `stdout` to overwrite the file
`>>file`	redirect `stdout` to append to the file
`2>file`	redirect `stderr` to append to the file
`2>/dev/null`	discard error messages by redirecting to `/dev/null`
`&>file`	redirect `stdout` and `stderr` to overwrite the same file
`&>>file`	redirect `stdout` and `stderr` to append to the same file

The order of redirection is important. The sequence `> file 2>&1` redirects redirect `stdout` to a file (`>file`), then redirect `stderr` (`2>`) to the same place that `stdout` is directed to (`&1`, 1 is the number of `stdout`). However, the sequence `2>&1 > file` redirects `stderr` (`2>`) to the place of `stdout` (`&1`), meaning the terminal window. It then redirects only the `stdout` to a file.

- Standard error can be redirected through a pipe, but the merging operator (`&>` or `&>>`) cannot be used. The following command is the correct way to redirect both `stdout` and `stderr` through a pipe.

```
admin $> find /etc -name passwd 2>&1 | less
```

- The `tee` command redirects `stdout` to the terminal window, and at the same time, passes it to some other program through a pipe. The following command prints the output of `ls -l` command on the terminal window as well as redirects `stdout` to a file.

```
admin $> ls -l | tee /tmp/output
```

- In the following example, `/dev/pts/0` is the device file of that represents the current terminal window. The command prints the `stdout` of `ls -l` command on the terminal window as well as sends it to a specified email.

```
admin $> ls -l | tee /dev/pts/0 | mail huy.bui@edu.xamk.fi
```

- Save the output to a file and discard error messages

```
admin $> find /etc -name passwd > /tmp/output 2> /dev/null
```

- Store output and error messages to the same file

```
admin $> find /etc -name passwd &> /tmp/output-errors
```

- Append output and error messages to an existing file

```
admin $> find /etc -name passwd >> /tmp/output-errors 2>&1
```

- Copy 10 lines from a log file and append them to another file

```
admin $> tail -n 10 /var/lo/dmseg >> /tmp/boot-messages
```

- Concentrate four files into one

```
admin $> cat file1 file2 file3 file4 > /tmp/four-in-one
```

- Save output and error messages to separate files

```
admin $> find /etc -name passwd > /tmp/output 2> /tmp/error
```

3.2 Vim editor

Vim is a highly configurable and efficient editor for practiced users, including such features as split screen editing, color formatting, and highlighting command syntax. When first opened, Vim starts in *command mode*, used for text manipulation (navigation, cut, paste, delete, etc.). Enter each of other modes with a single character keystrokes:

- An `i` keystroke enters *insert mode*, where all text typed becomes file content. Press `Esc` to return to command mode.
- A `v` keystroke enters *visual mode*, where multiple characters may be selected for text manipulation. Use `V` for multi-line selection and `Ctrl+v` for block selection. The same keystroke used to enter the visual mode (`v` , `Ctrl+v` , `V`) is used to exit.
- The `:` keystroke begins *extended command mode* for tasks like saving file, exiting Vim editor.

The following workflow presents the minimum keystrokes every Vim user must learn:

1. Open a file with `vim <filename>`
2. Use arrow keys to move the cursor
3. Edit file content in insert mode

   - Press `i` to enter insert mode, then edit the file content
   - Press `Esc` to return to command mode

4. Text manipulation in command mode
   - Press `u` to undo mistaken edits on the current line
   - Press `x` to delete a selection of text

5. Rearranging existing text
   - In command mode, press `v` to enter visual mode. Use arrow keys to position the cursor to the last character
   - Press `y` to yank (copy) the selection. Move the cursor to position the insert location. Press `p` to paste the selection

6. Save files or exit
   - Enter `:w` to write (save) the file and remain in command mode
   - Enter `:wq` to write the file and quit Vim
   - Enter `:q!` to quit Vim but don't save file

Chapter 4

Users and Groups

4.1 Introduction

Every process on the system runs as a particular user. Every file is owed by a particular user. Access to files and directories are restricted by a user. Each user has a name and UID.
There are two types of user: Standard user and System user. **Standard user** can log into the system and have an ID starting from 500 or 1000. **System users** are created by default during the Linux installation and are used by the system for specific roles. These user accounts cannot be used to log into the system. The **root** user account is created by default; it can be used to log into a system and perform any tasks.

Ranges of UID: root = 0, static system processes = 1 – 200, dynamic system processes = 201 – 999, regular users = 1000+.

The `su` command allows a user to switch to another user account. If a name is not specified, the *root* account is implied. The command `su <username>` starts non-login shell with the current environment settings, while `su - <username>` starts login shell and sets up the shell environment as if this were a clean login as that user.

Like users, each group has a name and an ID. There are two types of group: Primary group and Supplementary group. Every one user has exactly one **primary group**. This group automatically owns every new files or directories that the corresponding user created. It is created with the same name when a new user is created. **Supplementary group** membership ensures that particular users share access to files or directories.

4.2 Configuration files

The system uses `/etc/passwd` file to store information about local users. The format of `/etc/passwd` follows:

```
username:password:UID:GID:GECOS:/home/dir/:shell

admin:x:1000:1000:admin:/home/admin:/bin/bash
gdm:x:42:42::/var/lib/gdm:sbin/nologin
```

- `username` is either the name of a standard user or the name of a daemon
- An `x` in the `password` field indicates passwords are stored in the `/etc/shadow` file
- `UID` is the unique ID assigned to that user.
- `GID` is the user's primary group ID
- `GECOS` is arbitrary text indicating user's real name
- `/home/dir` is the location of the user's data and configuration files
- `shell` is a program that runs as the use logs in

The `/etc/skel` directory contains a set of configuration file templates that are copied into a new user's home directory when it is created. The `/etc/default/useradd` file contains default values used by the useradd utility when creating a user account.

The `/etc/shadow` file holds passwords and password expiration information for user accounts. Like the `/etc/passwd` file, each entry corresponds to a user account and each entry contains multiple fields, with each field separated by a colon. The following line is a sample entry in the /etc/shadow file:

```
pclark:$ab7Y56gu9bs:12567:0:99999:7:::
```

The fields within this line are as follows:

1. User account name.
2. Password. `$` preceding the password identifies the password as an encrypted entry. `!` or `!!` indicates that the account is locked and cannot be used to log in. `*` indicates a system account entry and cannot be used to log in.
3. Last change. The date of the most recent password change, measured in the number of days since 1 January 1970.
4. Minimum password age. The minimum number of days the user must wait before changing the password.
5. Maximum password age. The maximum number of days between password changes.
6. Password change warning. The number of days a user is warned before the password must be changed.
7. Grace logins. The number of days the user can log in without changing the password.
8. Disable time. The number of days since 1 January 1970, after which the account will be disabled.

Local groups' information is stored in `/etc/group`. The users of a supplementary group are listed at the last field of the group entry in this file. The following line is a sample entry in the /etc/group file:

```
sales:x:510:pclark,mmckay,hsamson
```

The /etc/gshadow file holds passwords for groups. The following line is a sample entry in the /etc/gshadow file:

```
sales:!:pclark:pclark,mmckay,hsamson
```

4.3 Superuser

To give standard user accounts the permissions to execute a limited set of commands as the root user, use the `sudo` command coupled with the `/etc/sudoers` file. Unlike other tools such as `su`, `sudo` requires users to enter their own password for authentication, not the password of the account they are trying to access. All commands executed using `sudo` are logged to `/var/log/secure` file. In Red Hat Enterprise Linux 7, all members of group `wheel` can use `sudo` to run commands as any users, including `root`.

The `/etc/sudoers` file can only be edited using the `visudo` command. Table 4.1 describes the sections used to configure the this file:

Table 4.1: Sections in sudoers file

Section	**Description**
`User_Alias`	Specify a set of users who are allowed to execute sudo command.
`Cmnd_Alias`	Specify a set of commands that users can execute using the sudo command.
`Hosts_Alias`	Specify a list of computers on which sudo users can perform commands.
`Runas_Alias`	Specify a username that is used when running commands with sudo. Usually this is just root.

Each of these aliases in table 4.1 are defined independently within the `/etc/sudoers` file. To grant users elevated access to the system, these aliases needs to be associated with each other to define exactly what will happen. The syntax is:

```
User_Alias Host_Alias = (user) Cmnd_Alias
```

Listing 1: Sudoers configuration

```
User_Alias INSTALLERS = jsmith, psimms
Cmnd_Alias INSTALL = /bin/rpm, /usr/bin/up2date, /user/bin/yum
Host_Alias FILESERVERS = fs1, fs2, fs3

INSTALLERS FILESERVERS = (root) INSTALL
%wheel FILESERVERS = (root) INSTALL
```

The 1st command adds the users `jsmith` and `psimms` to the `INSTALLERS` alias. The 2nd command assigns the `rpm`, `up2date`, and `yum` commands to the `INSTALL` alias. The 3rd command adds the three computers `fs1`, `fs2`, and `fs3` to the alias. The 4th entry specifies that all users listed in `INSTALLERS` alias are allowed to run the commands associated with `INSTALL` alias, and this can only be done on the computers listed in `FILESERVERS` alias.

We can specify a list of users using group instead of `User_Alias`. For example, the last command specifies all users in group `wheel` are allowed to run the commands in the `INSTALL` alias on the hosts contained in the `FILESERVERS` alias as the root user.

After editing `/etc/sudoers` file, reboot the system to apply changes.

4.4 User configuration

Some defaults, such as the range of UID and default password aging rules, are read from the `/etc/login.defs` file. Values in this file are only used when creating new users. Any changes on this file does not affect the existing users.

useradd	
`<username>`	Create a user with default settings. Note that this command does not set any valid password and the user cannot log in until the password is set.
`--help`	Display `useradd` commands and options.
usermod	
`-c <string>`	Add value to GECOS field.
`-d <home-dir>`	Specify new home directory location (use absolute path)
`-m <dir>`	Move user's home directory, must be used with `-d` option
`-G <group>`	Defines the secondary group membership.
`-a`	Used with `-G` option to append the user to supplementary group without removing the user from other groups
`-s <shell>`	Specify a login shell for user account
`-L <username>`	Lock user account
`-U <username>`	Unlock user account
userdel	
`<username>`	Delete the user from `/etc/passwd` file but leaves the home's directory
`-r <username>`	Delete the user and the user's home directory

Note! When a user is removed using `userdel` command without `-r` options, the remaining files in that user's home directory are owned by unassigned UID. Because the first free UID is assigned to any newly created user, the old user's UID will be reassigned to the new one, and therefore, the new user owns all files of the old user. This is a serious security issues.

passwd	
Without options	Assign or Change the current user's password.
`<username>`	Assign or Change password of a specified account.
`-S <username>`	Display status of the user account. `LK` indicates that the user account is locked; `PS` indicates that the user account has a password.
`-l`	Renames a user account.
id	
Without options	Display the current user information (UID, GID, etc.)
`<username>`	Display the specified user information (UID, GID, etc.)

chage	
`-l <user>`	Display user's aging information.
`-E`	Set expiration date
`-d`	set the date when the password was last changed
`-n <day>`	Sets the minimum number of days a password exists before it can be changed.
`-x <day>`	Sets the number of days before a user must change the password (password expiration time).
`-W <day>`	Sets the number of days before the password expires that the user is warned. `-I <day>` sets the number of days following the password expiration that the account will be locked.

4.5 Group configuration

4.6 Useful commands

- Find "unowned" files and directories

```
admin $> sudo find / -nouser -o -nogroup 2> /dev/null
```

- Creates the khart user account. Then modify account information of user `khart` change comment, assign to new supplementary group `comedian`, change shell to `/bin/tsch`

```
admin $> sudo useradd khart
admin $> sudo usermod -c "Kevin_Hart" -Ga comedian -s /bin/tsch khart
```

- Remove all supplementary groups of esmith

```
admin $> sudo usermod -G "" esmith
```

groupadd	
`<groupname>`	Create a group with default settings. A group must exist before a user can be added to the group.
`-g <num>`	Specify GID.
`-r <groupname>`	Create a system group.
`-n`	Change group name. For example, `groupmod -n sales2 sale` rename group `sale` to `sales2`.
Other group commands	
`groupdel <group>`	Delete a group. Check all file systems to ensure that no files remain owned by the group.
`gpasswd`	Assign or change group password.
`gpasswd -d <user> <group>`	Remove a user from a group.
`newgrp`	Log in to a group with password.
`groups <user>`	Display the primary and secondary group membership for the specified user account.

- Force user `esmith` to change password on next login

```
admin $> chage -d 0 esmith
```

- Set the date on which the user account `ravi` will no longer be accessible

```
admin $> chage -E 2018-02-16 ravi
```

Chapter 5

Permissions and Ownerships

5.1 Permission

5.1.1 Introduction

Permissions are identified with either the letter abbreviation (`r`, `w`, `x`, etc.), or the octal value that corresponds to the permission. In letter abbreviation: if a given permission has not been assigned, a dash (-) takes its place in the mode.

```
[user1@host1 ~]$ ls -l
-rw-rw-r--. 1 user1 user1 0 Feb 8 17:36 test.txt
[user1@host1 ~]$ ls -ld /home
drw-rw-r--. 1 user1 user1 0 Feb 8 17:36 /home
```

Effects of permissions on files and directories

Permission	**Effect on files**	**Effect on directories**
`r` (read)	Contents of the file can be read	Content of the directory (list of file names) can be shown
`w` (write)	Contents of the file can be changed	Any file in the directory can be created or deleted
`x` (execute)	Files can be executed as commands	Contents of the directory can be accessed

A user normally has to have both `read` and `execute` on a read-only directory, so that the content of the directory can be listed. If a user only has `read` permission on a directory, then except for the list of all files, no other information such as permissions, time stamps, can be accessed. If users only have `execute` permission on a directory, they cannot see the list of all file names in the directory, but they can access any file in the directory as long as they know the file names.

The `ls -l` command will expand the file listing including file's permissions and ownerships. The `ls -ld` command does the same for directories.

5.1.2 Configuration

The command `chmod` can change permissions of files and directories. The permission instruction can be issued in either symbolic form or numeric form.

```
chmod WhoWhatWhich file|directory
```

The symbolic method of change permissions uses letters to represent different groups of permissions: `u` for user, `g` for group, `other` for other, and `a` for all. In order to change permission, use three symbols: `+` to add permissions, `-` to remove permissions, and `=` to replace the entire set of permissions. For example, the following command removes write and read permission from group and other on `file.txt`

```
admin $> chmod go-rw file.txt
```

The following command execute permission to everyone on `file.sh`

```
admin $> chmod a+x file.sh
```

As for numeric method of changing permissions, each digit # represents an access level: user, group, and other. Each digit # is the sum of `r`=4, `w`=2, and `x`=1. Examine the permission `-rwxr-x---`. For the user, `rwx` is calculated as 4+2+1=7. For the group `r-x` is calculated as 4+0+1=5, for other is calculated as 0+0+0=0. Putting these three together, the numeric representation is 750.

```
chmod ### file|directory
```

The following command set permissions for `sampledir` file:

```
admin $> chmod 750 sampledir
```

5.2 Ownership

5.2.1 Basic configuration

When a file is created, its owner is the user who created it, and the owning group is the user's current group. The `chown` command can change these values to something else. The syntax of this command is shown as below:

```
chown <option> <ownership> <file>
```

Specifically, there are five ways to format `<ownership`:

- `user.group` or `user:group` – The user and group to own the file, separated by a colon (or dot), with no spaces in between.
- `user` – The name of the user to own the file. In this form, the colon (":") and the group is omitted. The owning group is not altered.
- `.group` or `:group` – The group to own the file. In this form, user is omitted, and the group must be preceded by a colon.
- `user:` or `user.` – If group is omitted, but a colon follows user, the owner is changed to user, and the owning group is changed to the login group of user.

For example, the following command would grant ownership of `file.txt` to user `visitor` and group `guests`.

```
admin $> chown visitor:guests file.txt
```

`chown` can be used with the `-R` option to recursively change the ownership of the entire directory tree. The following command would grant ownership of `foodir` along with all files and subdirectories within it to user `student`.

```
admin $> chown -R student ~/foodir/
```