The Notebook of LINUX

Huy Bui

THE PUBLISHER

Contents

List of Tables

Chapter 1

The Shell

1.1 Basic commands and tips

Table 1.1: Shell shortcuts

Shortcuts	**Explanation**
Ctrl + Alt + F1 – 6	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.

```
$ echo Today is `date +%A`
$ echo Today is $(date +%A)
$ cat $(fgrep -l ens32 /var/*)
```

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 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.

Table 1.2: File 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.

Pattern	**Matches**
`*`	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)

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

$ 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`

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

Files and Directories

2.1 Managing files

Chapter 3

Getting help

3.1 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 3.2 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 3.1: 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 3.2: Sections in each man page

Section	**Purpose**
NAME	Provides the name of the command and a very brief description.
SYNOPSIS	A brief summary of the command or function's interface. A summary of how the command line syntax of the program looks.
DESCRIPTION	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.

3.2 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 3.3).

Table 3.3: 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

3.3 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.

3.4 Use redhat-support-tool

3.4.1 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
```

3.4.2 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 3.1: 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
```

3.4.3 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 3.2, the `--product` and `--version` are specified using `opencase` command.

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

Figure 3.2: 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 3.3).

Figure 3.3: 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 4

Text files

4.1 Redirecting output

Table 4.1: 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 4.2: 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 such as `>` or `>>` cannot be used. The following command is the correct way to redirect both `stdout` and `stderr` through a pipe.

```
root $> 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.

```
root $> 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.

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

- Save the output to a file and discard error messages

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

- Store output and error messages to the same file

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

- Append output and error messages to an existing file

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

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

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

- Concentrate four files into one

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

- Save output and error messages to separate files

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

4.2 Edit text files