In-Chapter Exercise Solutions

Chapter 0 Basic Commands

1. **la -ls**(there is no **la** command in Linux), **more -q file1**(there is no **-q** option to the more command), **lsblk-a**(there has to be a space between the **lsblk** command and its option **-a**).

2. There has to be a space character separating those components.

3. A single command is just one operation or instruction given to the terminal. It performs a specific action and then returns output or completes the task.

When typing multiple commands, you can combine them in various ways, each serving different purposes. The execution of subsequent commands can depend on the outcome of the previous one, or they can be run sequentially regardless of the outcome. Here are the primary methods to combine multiple commands:

a. Sequential Execution (;):

Behavior: Executes commands in sequence, one after the other, regardless of the success or failure of the previous command.

Example: cd /var/log; ls will change the directory to /var/log and then list its contents, even if the cd command fails.

b. Logical AND (&&):

Behavior: Executes the next command only if the previous command succeeds (exits with status 0).

Example: mkdir new\_folder && cd new\_folder will only change the directory to new\_folder if the folder creation is successful.

c. Logical OR (||):

Behavior: Executes the next command only if the previous command fails (exits with a non-zero status).

Example: gcc program.c || echo "Compilation failed" will only display "Compilation failed" if the compilation does not succeed.

d. Pipelines (|):

Behavior: Uses the output of the previous command as the input to the next command.

Example: ls | grep "test" will list files and then filter this list to only show files containing "test".

e. Command Grouping (() and {}):

Behavior: Groups commands together to be executed as if they were a single command. Parentheses run the commands in a subshell, while curly braces run them in the current shell context.

Examples:

(cd /tmp; echo "This runs in a subshell") changes directory and echoes a message in a subshell.

{ cd /tmp; echo "This runs in the current shell"; } does the same but in the current shell context.

4. Before you execute the command, you should know exactly what it’s supposed to accomplish, and then after execution, you can examine the results to see if they match what you expected to happen. Sometimes that’s very difficult, or impossible, to determine.

5. No answer required.

6. The general form is- **cat file1 file2 > output\_file**

7. No answer required.

8. No answer required.

9. No answer required.

10. The **whatis** and **man -k** commands are both used to search for information about commands in Unix-like operating systems, such as Raspberry Pi OS, but they serve different purposes and operate in slightly different ways:

**whatis**

Purpose: Displays a one-line description of a command.

Operation: It searches for the command in the manpage database and returns a brief summary of the command's functionality. This summary is taken directly from the NAME section of the manpage.

Example: Running whatis passwd would typically return something like "passwd - update user's authentication tokens(s)", which is a concise description of what the passwd command does.

**man -k**

Purpose: Performs a keyword search across all manpage descriptions.

Operation: It looks for the specified keyword in the descriptions of all manpages and returns a list of all matches. This can include the command in question but also any other commands and programs that mention the keyword.

Example: Running **man -k passwd** might return entries for the **passwd** command itself as well as other related commands and configuration files, such as chpasswd, passwd(5) (for the passwd file format), and possibly others, each accompanied by a brief description.

11. See ICE 10.

12. On a Raspberry Pi 400 running the latest OS at the time of the writing of this book-

Of course bash is available, and is the default shell.

bob@raspberrypi:~ $ **whereis ksh**

ksh: /usr/bin/ksh /usr/share/man/man1/ksh.1.gz

bob@raspberrypi:~ $ **whereis sh**

sh: /usr/bin/sh /usr/share/man/man1/sh.1posix.gz /usr/share/man/man1/sh.1.gz

bob@raspberrypi:~ $ **whereis csh**

csh: /usr/bin/csh /usr/share/man/man1/csh.1.gz

bob@raspberrypi:~ $ **whereis zsh**

zsh: /usr/share/zsh

13. On our Raspberry Pi 400-

bob@raspberrypi:~ $ **who --all**

system boot 1969-12-31 16:00

bob + tty7 2023-10-04 14:53 old 774 (:0)

run-level 5 2023-10-04 14:53

bob - tty1 2023-10-04 14:53 old 1818

pts/0 2023-10-05 11:08 4781 id=ts/0 term=0 exit=0

14. On our Raspberry Pi 400-

bob@raspberrypi:~ $ **hostname -I**

192.168.1.2

bob@raspberrypi:~ $ **ip addr**

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

2: eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc mq state UP group default qlen 1000

link/ether dc:a6:32:ee:c6:6b brd ff:ff:ff:ff:ff:ff

inet 192.168.1.2/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0

valid\_lft 49347sec preferred\_lft 38547sec

inet6 fe80::78d9:c72e:75e2:82c/64 scope link

valid\_lft forever preferred\_lft forever

3: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000

link/ether dc:a6:32:ee:c6:6c brd ff:ff:ff:ff:ff:ff