Salgado Berardo, Joao Antonio 100280950 A3 Jun/03/2017 Mark:__________

Chapter 2: 2.8. 2.11, 2.13, 2.14, 2.17, 2.19, 2.20 and 2.22

Chapter 2:

2.8 Yes. For example, in the command “ls -l -a dir1” we have four words.

The first word is the command itself and the others are arguments (3 arguments in this example).

Two arguments begin with “-” and are called options. The other argument is a directory that will be listed.

In the command cat < foo > bar, there are no arguments. We have 3 distinct commands with no arguments in them.

2.11 It would not work, because the -f option must be followed by a file name, it can’t have an option right after it.

If we use the command without the “-” symbol, it will work but it is not recommended.

2.13 The shell doesn’t care about whitespaces between arguments and options. You can use as many whitespaces you want, but the shell ensures that there is only one whitespace before running a command.

If you use double quotes with echo, it will print everything between “ “, including white spaces you type.

2.14 The secondary prompt looks like “>”. From the bash man page:

PS2 The value of this parameter is expanded as with PS1 and used as the secondary prompt string. The default is ``> ''

For example, we can create a file with the "cat" command. If we type use "cat > filename", the secondary prompt will appear, and you can type the contents of the file and press Ctrl+D to stop.

In this example, what you typed will be saved in a file named "filename".

2.17 I can have the same output with "man -k <arg>". With that option, man searches all man pages that contain the keyword in the NAME section. If we look at the man page for man, we have the following example:

man -k printf*: Search the short descriptions and manual page names for the keyword printf as regular expression. Print out any matches. Equivalent to* ***apropos printf****.*

2.19 There are certain characters that are interpreted by the shell, for example ", $ and \.

The backslash character (\) is used to "escape" those sequences of characters, so they are not interpreted by the shell and is passed to the command.

BSD systems doesn't recognize this, so we have to use the option -e in echo command to escape sequence.

Examples:

- \n is for line feed
- \\ is a backslash
- \t use for tab

A good example is for user input. We can use echo -e "enter input: \c". In that case, echo will place the cursor at the end of line, instead of new one.

2.20

Hexadecimal value: printf “Hex value of 255 is %x” 255

Octal value: printf “Hex value of 255 is %o” 255

2.22 I like this question. My answer will be a practical example, as I was doing this in my computer this week.

Suppose I have a shell script that displays the computer’s “system health”. By that, I mean some information about my computer, for example: CPU usage, memory usage, processes running and so on.

I can have a cron job scheduled to run this program, redirect its output to mailx command as the email’s body. I don’t need to interact with the program anymore.

With only a simple script, a cron job and a single-line command, I can get e-mails with my computer’s information without worrying. It is impossible to achieve that with a GUI program.