

Chapter 2

1. *Why do we refer to the user's environment as a shell?* It can be "filled in" by the user by supplying such definitions as aliases, variables, functions.
2. *What is the history of the Bash shell?* The earliest Unix shell was known as the Bourne-shell. With Linux, the Bash shell was developed which added features found in csh.
3. *Given the following prompt, answer the questions below.*

```
[zappaf@frenchie /etc]#
```

 - a. *Who is logged in?* zappaf
 - b. *Is the user logged in as themselves or root?* root (because of the #)
 - c. *What is the current working directory?* /etc
 - d. *What is the hostname?* frenchie
4. *What command would you use to determine your username?* whoami
5. *What command would you use to find out your current working directory?* pwd
6. *What command would you use to change your password?* passwd
7. *What is the difference between the arch command and the uname command?* uname tells you your operating system (e.g., Linux) while arch tells you your processor platform (e.g., i386).
8. *Assume that foxr is logged in to a Linux computer on console and user zappaf is logged into the computer remotely from IP address 1.2.3.4. What would the command who show?*

```
foxr    tty1    2013-11-04 14:23 (:0)
zappaf  pts/1   2013-11-04 15:31 (1.2.3.4)
```
9. *From your Bash shell, you type bash. After working for a while, you type exit. What happens?* From within bash, you opened another bash shell. Typing exit leaves that new bash shell and returns you to your earlier bash shell.
10. *What is the difference between ls -l and ls -L?* ls -l performs a long listing. ls -L performs an ordinary listing (ls) but where soft links are followed so that the file being pointed to is listed rather than the soft link's name itself.
11. *What is the difference between ls -a and ls -A?* both list all files including "hidden" or dot files, but -a also lists . (current directory) and .. (parent directory) while -A does not.
12. *You want to perform a recursive listing using ls. What option(s) would you specify?* ls -R.
13. *You type ls and you receive 3 columns worth of file names. You want to see the output as a single listing. What can you do?* Type ls -1 (the number 1).
14. *What does the synopsis portion of a man page tell you? If you are not sure, try man ls, man mount and man ip as examples and see what you get.* The synopsis tells you the form or forms of syntax allowable for this command.
15. *You want to re-execute the last instruction you entered in your Bash session. Describe three ways to do this.* Type !, type control+p <enter>, type !# where # is the number in the history list of the last instruction entered, or type !c where c is the first letter of the command you last typed in.
16. *Given the history list from figure 2.3, what command is recalled with each of the following?*

- a. `!! history (#12)`
 - b. `!5 ls -l`
 - c. `!ls ls -l | grep root`
 - d. `!cd cd ~`
 - e. `!c cat script1`
17. What does `history -c` do? It clears your history list.
18. Define a variable to store your first initial, your middle initial and your last name. Include proper spaces and punctuation. `NAME="Richard K. Fox"`
19. Assume the variable `AGE` stores 21. What is the output for each of the following?
- a. `echo AGE` `AGE`
 - b. `echo $AGE` `21`
 - c. `echo $age` (nothing outputs, age is not a defined variable)
 - d. `echo \ $AGE` `$AGE` (the `$` is treated literally, not as the character to output a variable's value)
 - e. `echo $AGE\!` `21!`
20. Provide an instruction to alter your prompt so that it displays the following information:
(time : instruction number) [username] prompt character as in
(10:04 am : 31) [foxr] \$
`PS1="(\t : \!) [\u] $"`
21. Write the command to define as an alias `...` to move you up 2 directories. `alias ...="cd ../.."`
22. Write the command to define as an alias `stuff` to perform a long listing of your home directory. `alias stuff="ls -l ~"`
23. Match the following command line editing function with the keystroke that provides it
- a. Move the cursor to the end of the line iii. `c+e`
 - b. Move back one character v. `c+b`
 - c. Move forward one word iv. `m+f`
 - d. Delete one character vi. `c+d`
 - e. Retrieve the last instruction entered vii. `c+p`
 - f. Undo the last keystroke ii. `c+-`
 - g. Delete all characters from here to the end of the line i. `c+k`
24. Provide an instruction to redirect the output of `ls -l` to the file `~listing.txt`. `ls -l > ~/listing.txt`
25. How does the operator `>>` differ from the operator `>`? The `>>` appends the output to an existing file while `>` will overwrite an existing file. If the file does not exist, they both create it.
26. Explain why `wc < somefile` does not require the `<` operator. Because by default, `<` expects input to come from a file, so `wc somefile` does the same thing.
27. What is the difference between `cat file1.txt file2.txt file3.txt` and `cat file1.txt file2.txt > file3.txt`? In the former case, all three files' contents are output to the terminal window while in the latter case the first two files' contents are concatenated together and stored in `file3.txt`.
28. Provide an example of why you might use `<<` as a form of redirection. To create a new file using `cat` with text typed from the keyboard.

29. *The following instruction does not do what we expect in that we want to output the word count for the items listed in the current directory. What does it do and how can you fix it?*

```
ls > wc
```

This instruction sends the ls output to be stored in the file wc. To fix this, we want to change > to | as in ls | wc.

30. *Name 3 compiled languages.* C, C++, Java, FORTRAN, COBOL, Pascal, Ada, Modula-2, Algol to name a few.
31. *Name 3 interpreted languages.* Perl, PHP, Ruby, Python, Lisp to name a few.
32. *Why is Bash an interpreted environment rather than a compiled one?* It would make little sense to make an operating system's interface compiled as you would have to pre-write your entire session (program) and compile it before running the instructions. Instead, you want the flexibility to see the results of the previous instruction before entering the next.
33. *Explain the role of the following characters as used in Bash.*
- a. ~ - user home directory
 - b. ! – recall instruction from history list
 - c. -- used to specify options in most Linux instructions
 - d. * - wild card character