Study Questions: Set No. 1

Introduction to UNIX

Sunday September 22, 2013

Covering:

Topic 01: *Introduction to Operating Systems;*

Topic 02: *Unix Basics;*

Topic 03: *Unix Editors;*

Topic 04: *Unix Files and Directories;*

Topic 05: *Unix Input-Output Redirection*

1. What are the two main kernel categories?
2. Explain the main differences between the two main kernel categories.
3. What is the difference between copyright and copyleft?
4. When was the first Unix operating system developed?
5. When was the first Linux operating system developed?
6. Where did the word "Unix" come from?
7. Where did the word "Linux" come from?
8. Who developed the first Unix operating system?
9. Who developed the first Linux operating system?
10. Who were the first to develop the first C compiler?
11. Who were the first to re-write the Unix operating system in C?
12. What will **cat foo foo foo** display?
13. Assuming that **bar** is a directory, explain what the command **rm -r bar** does. How is the command different from **rmdir bar**?
14. The command **rmdir c.progs** failed. State three possible reasons for this failure.
15. The command **rmdir bar** fails with a message saying that the directory is not empty. On running **ls bar,** no files are displayed. Why did the **rmdir** command fail?
16. Explain the difference between the commands **cd ~smart** and **cd ~/smart**.
17. Explain the difference between **cd $HOME** and **pushd $HOME**.
18. Explain what the following commands do: (i) **cd** (ii) **cd $HOME** (iii) **cd ~** (iv) **cd cd**
19. List 5 different ways to change the current working directory to your home directory.
20. How many characters can a Unix filename be?

21. Which character(s) can't be used in a Unix filename?

22. Why are we discouraged from having a filename beginning with a hyphen?

23. Can the files **note** and **Note** coexist in the same directory? Why?

24. Explain what the following commands do: (i) **rm *** (ii) **rm -i *** (iii) **rm -r *** (iv) **rm rm**

25. How does the Unix *shell* treat the * when used as an argument to a command like **echo ***?

26. What is the difference between **ls** and **ls *** Unix commands?

27. Does **ls *.*** match filenames that begin with a dot?

28. How do you remove only the hidden files in your current working directory? Does **rm *** remove these files as well?

29. Explain what the commands **ls .*** and **ls *.** display. Does it make any difference if the **-d** option is added?

30. Explain when **mv *.bat *.bit** will work without producing any error.

31. What does the command **ls -d [3-h]*** mean?

32. When will the command **cd *** work?

33. How do you remove all ordinary files in the current directory that (i) are hidden; (ii) begin and end with *; (iii) have numerals as the first three characters; (iv) have single-character extensions?

34. Match the filenames **chapa**, **chapb**, **chapc**, **chapx**, **chapy**, and **chapz** with a wild-card expression.

35. Devise wild-card patterns to match the following filenames: (i) **fool**, **foo2** and **foo5**; (ii) **quit.c**, **quit.o** and **quit.h**; (iii) all filenames that begin with a dot and end with **.swp**.

36. Explain what these wild-card patterns match: (i) **[A-z]????*** ; (ii) ***[0-9]***

37. A directory **bar** contains a number of files including one named **-foo**. How do you remove this **-foo** file?

38. You have a file named * and a directory named **My Documents** in the current directory. How do you remove them with a single command using (i) escaping, (ii) quotation?

39. What will happen when you execute the following Unix commands: (i) **mkdir file name**, (ii) **mkdir 'file name'** , (iii) **mkdir "file name"**

40. What will happen when you execute the following Unix commands: (i) **mkdir filename?**, (ii) **mkdir 'filename?'** , (iii) **mkdir "filename?"**

41. What difference do you notice when executing the commands **echo "$SHELL"** and **echo '$SHELL'**?

42. How do you display the *inode* number of a file?

43. If **ls -i** shows two filenames with the same *inode* number, what does that indicate?

44. What is the difference between executing **ln file1 file2** and **ln –s file1 file2** Unix commands? How will each command affect the *inode* of **file2**?

45. When you invoke **ls -l foo**, the access time of **foo** changes. True or false? Explain your answer.

46. Show the octal representation of these permissions: (i) **rwxr-xrw-** (ii) **rw-r-----** (iii) **--x-w-r--**

47. What will the permissions string look like for these octal values? (i) **567** (ii) **623** (iii) **421**

48. Use the *numeric* permission method to change the permission of a file in the current working directory called **abc** to give the *owner user* read and write permissions, while giving *group users* and *others* read permission only.

49. Use the *symbolic* permission method to change the permission of a file in the current working directory called **abc** to give the *owner user* read and write permissions, while giving *group users* and *others* read permission only.

50. Explain and give example showing the difference between *numeric absolute, symbolic absolute,* and *symbolic relative* permission methods.

51. How can you combine *absolute* and *relative* permission methods together? Give an example.

52. How do you ensure that all ordinary files that you have created have **rw-rw----** as the default permissions?

53. What do the **r**, **w**, and **x** letters mean in a regular file permission string?

54. What do the **r**, **w**, and **x** letters mean in a file directory permission string?

55. How do you assign all permissions of a file to the owner and remove all permissions from others using (i) symbolic relative assignment (ii) symbolic absolute assignment (iii) numeric absolute?

56. Assuming that a file's current permission is **rw-r-xr--**, specify the **chmod** argument(s) that you should use to change the file permission to (i) **rwxrwxrwx** (ii) **r--r-----** (iii) **---r--r--** (iv) ---------, using both relative and absolute methods of assigning permissions.

57. What do you do to ensure that no one is able to see the names of your files?

58. The command **cd bar** failed where **bar** is a directory. How can that happen?

59. How do you split a long command sequence into multiple lines?

60. When will **wc** < **chap0[1-5]** work?

61. Is **>foo <bar sort** a legitimate Unix command, and what does it appear to do?

62. What happens when you execute (i) **cat > foo** if **foo** contains data, (ii) **who >> foo** if **foo** doesn't exist**?**

63. When executing the following two Unix commands: **sort filename** and **sort < filename** you get the exact same result, if **filename** already exists. However, if **filename** does not exist, you get two different error messages. Explain why.

64. How do the commands **wc foo** and **wc < foo** differ?

65. You want to concatenate two files, **foo1** and **foo2**, but also insert some text after **foo1** (but before **foo2**) from the terminal. How will you do this?

66. How do you redirect each of the standard output and standard error to two different files in the Bourne-shell and C-shell? Give an example.

67. How do you redirect each of the standard output and standard error to the same file in the Bourne-shell and C-shell? Give an example.

68. How do you redirect the standard input from a file in the Bourne-shell and C-shell? Give an example.

69. What will you get if you execute **ls 1> out.file** in Bourne-shell and in C-shell? Explain.

70. Display lines 30 through 40 of the file **poem** on your screen. Assume that the file **poem** has more than 40 lines.

71. Print the 5^{th} line in a file called **input_file**. Hint: think of **head** and **tail** commands.

72. How do you select from a file: (i) lines 5 to 10, (ii) second-to-last line?

73. How does **head** display its output when used with multiple filenames?

74. What is the file **/dev/null** used for?

75. How do you find out the number of (i) users logged in (ii) directories in your home directory tree?

76. Consider the Unix command **ls -lrt | tee > newfile;** Why does this command not produce any output on the screen? Give the command that uses **tee** correctly (save the output to **newfile**).

77. In a Bourne-shell, what do the following commands mean? Are they equivalent? Explain.
    cat 1> letter 2> save < memo
    <memo >letter 2> save cat

78. Print the names of all files in the current working directory to the screen in upper case (i.e., translate all lower case characters to upper case). *Hidden* files should be included.

79. How do you convert a content of a file to uppercase letters?

80. How do you convert a content of a file to lowercase letters?

81. How do you convert, using one Unix command, each lowercase letter in a file to an uppercase letter and at the same time each uppercase letter in the same file to a lowercase letter?

82. A program, **a.out**, continuously writes to a file. How do you run the program so that you can monitor the growth of this file from the same terminal?

83. What is the difference, if any, between **sort filename | uniq** and **sort –u filename**

84. If you did not have the **wc** command on your system, how do you use **grep** command to count the number of users currently using the system (not necessarily unique users).

85. Assume that **filename** is a text file in your current working directory. Is there any difference in the output of the following two Unix commands: **cat filename | sort | uniq** and **cat filename filename | sort | uniq**; Justify and explain your answer.

86. Assume that **filename** is a text file in your current working directory. Is there any difference in the output of the following two Unix commands: **cat filename | sort | uniq** and **cat filename | uniq | sort**; Justify and explain your answer.