Salgado Berardo, Joao Antonio 100280950 A10 Jul/06/2017 Mark:_____

Chapter 10: 1, 2, 6, 10, 11, 12, 13, 14, 16, 17, 30

Chapter 10:

1. Wild card is a single character, for example * or *?*. In fact, regular expressions and wild cards use some of the same characters (*, *?*, etc.), but some of their functionality differ. For example, the wild card *?* means any single character. In regular expression this is accomplished with a dot (.).

Wild cards are simpler and faster than regular expressions. For more simple patterns wild cards do a good job, but for more complex patterns, like a phone number or a zip code, regular expressions are generally better.

2. (i) Looks for the string *a* in the files named *b* and *c*.

(ii) This command would not work because it is not between double quotes. The shell will interpret the "<" symbol and will try to use HTML as standard input file for the *grep* command. It should be between double quotes or use *grep* \<HTML\>.

(iii) If the * is the first character in a regular expression, it is treated as the * itself. So, this expressions matches the preceding * (literally) zero or more times. In this case, it will match everything.

(iv) The dot means any single character, so it will look for a pattern that has 2 characters starting with *.

6. Yes, they are both equivalent. The *-v* option inverts the match, selecting lines that don't match.

The catch is the second caret in the first command. The grep command with *-v* option doesn't have it. If the second command didn't have the *-v* option, one command would be the inverse of the other.

10. The shell executes the commands inside the back-quote first. So, it first executes *grep*, looking for the string *fork* in all C source files. The output of is piped to *cut* command, that retrieves the first field delimited by a colon (:). The standard output of the *cut* command is piped to the *sort* command, that sorts using the *-u* (unique) option, to show the file name only one time.

The output of the commands inside the back-quote (list of unique file names that contains the string *fork* inside) is used for *ls* to list the files ordered by modification date.

To use two commands instead of four, we can use the *-l* option from *grep*, that prints the name of the files, instead of normal output.

$ *ls -t `grep -l fork *.c`*

11. $ *find . -perm `stat -c "%a" foo`*.

12. (i) jeff[er][er][iy][es]s*

(ii) hitch[ei]ng*

(iii) [hH][ei][ar][rd*]d*

(iv) di[cx][ko]*[sn]*[on]*

(v) [Mm][ac]g[he]ee*

(vi) wood[ch]*o*[cu]*[sk]*e*

13. (i) The first one searches for any digit between 0 and 9, or no digit at all, while the second command makes sure that at least one digit is found, followed by a digit or not.

(ii) Caret has some special meanings in regular expressions. At the beginning of the expression, it matches the pattern at the beginning of the line. If it is the first character inside double brackets, it negates the pattern inside []. Anywhere else it is treated as caret itself. So, the first one matches everything that doesn't begin with ^. The second one matches everything that begins with two carets (^^).

14. echo "You have" `cat /var/mail/romeo | grep -i "From: henry" | wc -l` e-mails from Henry!

echo "You have" `cat /var/mail/romeo | grep -i "urgent\|immediate" | wc -l` urgent/immediate e-mails!

16. This command retrieves the first three fields of the /etc/passwd file with the user-id 100. Even if we don't know the user name, we can obtain it from this command.
Suppose we have the user id 1005, we can use this command together with cut (cut -d: -f1) to retrieve the user name for that id.

17. $ grep -w "printf[^$]" file

30. PATH=`echo $PATH | sed -e "s/\/usr\/local\/bin//g"`