Salgado Berardo, Joao Antonio 100280950 A5 Jun/20/2017 Mark:_____

Chapter 4: 2, 4, 7, 8, 9, 10 and 14.

Chapter 6: 3, 5, 6, 7, 10, 11.

Chapter 4:

2. No. Owner of a file and group owner are distinct things, they can be different.

4. (i) chmod u+rwx,go-rwx foo

(ii) chmod 700 foo

Yes. foo's default permission would be set using umask settings.

7. (i) chmod u+x,g+w,o+wx file and chmod 777 file

(ii) chmod u-w,g-x,o-r file and chmod 440 file

(iii) chmod u-rw,g-x file and chmod 044 file

(iv) chmod u-rw,g-rx,o-r file and chmod 000 file

8. No. After using 'chmod -w' I was unable to create or delete files because of permissions. The command 'chmod -w' revokes writing privileges for everyone (including owner).

Yes, 'chmod -w .' is the same as 'chmod a-w foo'.

9. One reason could be lack of permission to read the file. Another reason is the directory's permission, the user must have read permission on the directory.

10. You can set the directory permissions to deny all other users from accessing you directories and files, so other users cannot list the contents nor access your directory. Using dot files would not work because anyone with access to the directory could use the command “ls -a”.

In summary, revoke read and execute permissions from the directory containing the files.

14. If umask shows the value (i) 000, (ii) 002, what implications do they have from the security viewpoint?

The default file and directory permissions in UNIX is 666 for regular files (read and write for all) and 777 for directories (read, write and execute for all).
The umask setting is subtracted from this default values to determine a user default permission when creating files and directories.

So, if umask’s value is 000, that means that nothing is subtracted, and the default permission is to allow all to read and write for files, and read, write and execute for directories. It is one of the worst things someone could do in terms of security.

With umask 002, the default permissions would be 664 (rw-rw---x) for files and 775 (rwxrwxr-x) for directories.

Chapter 6:

3. (i) [fF]ile[125]

(ii) quit.[coh]

(iii) [wW]atch.[hH][tT][mM][lL]

(iv) .*.swp

5. (i) rm -r .

(ii) begin and end with # rm [^#&@#]

(iii) rm [0-9][0-9][0-9]*

(iv) rm *.?

6. (i) [0-9]*?*[!zZ-aA]

(ii) !.*

(iii) ?*2004*?

7. (i) [A-z]????* files beginning with character between ASCII range (A-z), containing at least 5 characters

(ii) contains digits in the middle of the string

(iii) does not contain digits at the end

(iv) files with all extensions, except *.sh (all files with .sh extension)

10. What are the two consequences of using double quotes?

Single quote can be used to “turn off” metacharacters in bash. For example: rm ‘foo*’ removes the file names foo*. If we used double quotes, it would treat as a metacharacter. A good example using quotes is when a directory or files have spaces in the name.

Using double quotes doesn’t protect you from special characters like single quote does. So if you use backticks (``) inside double quotes, it will be treated as a command substitution.

Also, the special character $ is interpreted as a variable prefix.

11. The command wc < chap0[1-5] didn't work. The workaround was to use "cat file0[1-5] | wc".

You can use double quotes to remove the file chap0[1-5].