Practical 1: Linux Scripting and File Permissions

Question 1: Understanding basic linux commands

Open a Linux Terminal (xterm or Konsole or gnome-console)
1.a Print the current date
1.b Read the manual pages and info pages for the commands, head, tail and touch.
hints: Note that the command "`man`" may use the system pager (less or more).
To quit the man page type q. To go to next page use space bar. You can also use the arrow keys usually.

1.c Explain the purpose of the following commands in 3 sentences or less each: head, tail, touch, more, less, man (use "man man" from the command line for this).
1.d If you wanted to print the first 3 lines of a file, and the last 5 lines of a file explain how would you do this using the commands, `head`, and `tail`. Print the first 3 lines and the last 3 lines of `/etc/passwd`.
1.e
Type "`ls`" to list your home directory contents.
type "`touch file.txt`" in your home directory
Now type `ls` again. Do you see the new file you just created?

Question 2: Composing basic linux commands

2.a Page through `/etc/passwd` (Hint: use "`less /etc/passwd`" or "`more /etc/passwd`" from the command line. Identify the name of some user (the first field on any line. Recall that each field is separated by the ":" character). Do you see your own username in this list? Why or why not?
(*Hint*: google `/etc/shadow` for some hints on alternate possibilities to `/etc/passwd`)
2.b The last field of each line is the default shell for a user. Use the "`cut`" command to print out all the shells being used by users. Use "`man`" to identify how to use cut for this purpose.
2.c. You will find that the output of "`cut`" has many duplicates. Try to remove these duplicates using the command "`uniq`". To do this you will have to pipe the output of cut to the command `uniq`.
2d. You may find that even after `uniq`, there are still some duplicates. This is because `uniq` requires its inputs to be sorted. (each Unix command does one job, and relies on others in the suite of tools to do other jobs. It is your job to compose them together properly). Now, to completely remove all duplicates, enhance the above pipeline, to sort the output of cut, before passing to uniq. The unix command for sorting a list of lines is, predictably, called `sort`. List all the unique shells you see now.
2e. Pick a file name which does not exist (e.g., */etc/password*), and try to list it ("ls <filename>")
Notice that an error message gets printed on screen. This is an **error** message and gets printed on the standard **error** stream, rather than on the standard **output** stream. How do we tell this?

Try to do the ls again, except, now, ***redirect*** the standard error to a file.
`ls /etc/password 2>~/logfile`
Read the contents of the file "logfile" created in your home directory
(hint: use "`cat ~/logfile`").

Now repeat the procedure with a file which exists:
`ls /etc/passwd 2>~/logfile`
Now read logfile.

Notice that there is nothing in the standard error stream now. But something did get printed on the standard output, which is also output to the terminal.
2f. You can also redirect standard output. Instead of using 2> symbol, you can use “1>” in the above command. use this to list both `/etc/password` and `/etc/passwd`.
write down the contents of the log file in each case.
(in the case of standard output, you may omit the 1, and use “>” instead of “1>”. try this and confirm it works).
2g. The error messages are for humans to read, When composing commands in a shell script, the author of a script is less interested in reading the message and more in doing different things depending on the output of a previous command.
2.g.1 Use the ll command to echo “sorry, could not find file” if a file cannot be listed. Redirect standard error to a logfile and standard output from ls to an “output.log” Try this to list /etc/passwd and /etc/password
hint: the format to follow would be “`ls <filename>” || echo “sorry,…`” [you need to add the output redirection syntax to this basic skeleton].
2.g.2 Before using a command such as cut, you may want to make sure that the file exists. Use the && command to ensure that the file exists, and then use the cut command to output the names of the shells of the users of your system.
hint: test that the file exists, before using the pipeline developed in 2.d

2.g.3 Develop a small shell script, using the ‘if’ construct. This script should output the contents of the file if it exists, and echo “sorry, cant find file” if the file does not exist. Test it on some arbitrary file that you create yourself, or on /etc/passwd and /etc/password.
hint: See 3.b, but only after you have tried this yourself.