Shell Programming - Points: 66 (4 of 10%)

Do this work on the Course Linux Server. Do not use ACADUNIX.

Perform the shell commands that do following steps using one or more shell command lines, in the exact order given below.

File Name:
: Record the successful shell lines you use for each step in a file named **`test1.sh`** in your home directory. Submit this file at the end of your test using number 31:
`datsubmit 31 test1.sh`

File Format:
: Do **not** use semicolons to put multiple commands on a line; use **separate** lines for each command or command pipeline.

Comments Required:
: Put a one-line comment containing the **step number** in front of the command lines you use to answer each step. The comment line must start with a "#" character, e.g.

`# Step 1`
[... one or more command lines for step 1 go here ...]
`# Step 2`
[... one or more command lines for step 2 go here ...]
`# Step 3`
[... repeat this format for all steps ...]

I will collect the results of your shell work from your home directory after the end of the lab period - do not delete it or modify it.

---

1. *[Points: 4]* Create a new, empty directory named **`test1`** in your home directory. Make this directory your current working directory. Do not change directories for the rest of this test. (Stay in the **`test1`** subdirectory for the rest of the test.) *(Remember to record the command(s) you use for each step in the file* **`test1.sh`** *in your home directory.)*
2. *[Points: 13]* Create a sub-directory named **`subdir`** and then, without changing directories, put the current date into an output file named **`dateout.txt`** under the **`subdir`** directory. Without changing directories, make a copy of that output file into a file named **`two.txt`** in the same directory. Without changing directories, rename the **`dateout.txt`** file to be a file named **`date.txt`** in the current directory. Finally, delete the **`subdir`** directory and everything inside it. *(Remember to record the command(s) you use for each step!)*
3. *[Points: 10]* Create three empty files named **`one`**, **`two`**, and **`three`**. Show a long listing (including the sizes in both blocks and bytes) of all pathnames in the current directory that are exactly three characters long. Delete all pathnames that are exactly five characters long.
4. *[Points: 10]* Write a single pipeline to select lines 17-26 from the Unix password file, sort them in reverse, and place the output in a file named **`password_reverse.txt`** in your current directory.
5. *[Points: 3]* Sort the file **`password_reverse.txt`** into the file

password_sorted.txt in the current directory.

6. *[Points: 6]* Write a single pipeline to select all lines from the Unix password file that contain the pattern **/bin/bash** and then count those lines. Display only the count of the lines (one number); do not display any other numbers. (Note: more than two dozen lines will be found.)
7. *[Points: 4]* Use an "invert match" option (RTFM) to select and display on your screen only lines that do *not* contain the pattern **nameserver** in the file **resolv.conf** under the directory **/etc** (find lines that do *not* contain the pattern). (Note: less than three lines will be output.)
8. *[Points: 5]* Without changing directories, write a single pipeline to count the number of non-hidden pathnames under the directory **/lib** that begin with the three-character string **lib** at the start. Display only the count of the lines (one number); do not display any other numbers. (Note: the count is larger than 70 and smaller than 90.)
9. *[Points: 5]* Without changing directories, display the file type information (what kind of data) of all the non-hidden pathnames under the **/bin** directory, and select only files whose type information contains the six-character string **script**. (Note: less than five files match this description.)
10. *[Points: 3]* Without changing directories, write a single command to display a long listing (showing the modify date) of the non-hidden pathnames under the directory **/lib** that contain the two-character string **ss** anywhere in the pathname. (Note: more than a dozen files match this description.)
11. *[Points: 3]* Use the **vim** editor to read in the Unix password file, globally change every occurrence of a digit zero into an upper-case letter **O**, and then write out the changes to a new file named **badpasswd.txt** in the current directory. (Note: there will be more than 175 substitutions.)

Record the commands you use to do the above actions.

Put a one-line comment containing the step number in front of the executable code lines in each step.

Submit your completed file using datsubmit when you are finished.

I will collect the results of your shell work from your home directory after the end of the lab period - do not delete it or modify it.

– *FIN* –