CS 460 Programming Assignment Two Shell Scripting

Due WEDNESDAY Feb 28, 2018, 11:59pm **This is a partner assignment**
35 points

Goal:
Build a small Bash shell script, find and use new Linux commands, and have fun.

To answer these questions, build a directory hierarchy as follows:

- PA2/
  - **PA2_PUNetID_PUNetID.sh** *this assignment!*
  - CS460_SysCalls_PUNetID *just one partner's executable from PA1*
  - LinuxTest/ *(fresh copy of LinuxTest.tar.gz's files)*
    - practice .sh files

Email the instructor only the file PA2_PUNetID_PUNetID.sh

Description:

Before displaying the answer to each question display the question number as:

```
echo
echo Question 1:
echo
```

The solution for each question must be general; if I add or remove files your output must update accordingly.

BONUS Write a function to output the question header above that takes the question number as a parameter.

Write a script that will:

1.Run your executable from Assignment One, CS460_SysCalls_PUNetID. The file opened by CS460_SysCalls_PUNetID must be the first command line argument to the script. The output of CS460_SysCalls_PUNetID must be saved in the filename given as the script's second command line argument. For example:

```
./PA2_PUNetID_PUNetID.sh /etc/passwd s.out
```

Must run:

```
strace CS460_SysCalls_PUNetID /etc/passwd > s.out 2>&1
```

Giving more to fewer command line arguments to the script needs to produce a nice error message and terminate.

2. Run your executable from Assignment One for each file that matches the pattern **`/usr/include/z*.h`** Save the output of each run in *filename*.out, where filename is the particular filename passed to your executable. For example:

```
CS460_SysCalls_PUNetID /usr/include/zz.h  > zz.h.out 2>&1
```

The list of files must be gathered dynamically from the file system.
HINT: Look at **`sed`** to produce the string zz.h

3. Dynamically determine which **`/usr/include/`**z*.h file produces the largest output file and display that filename name to stdout. If multiple files produce the same max size, output the last file you examine.

4. Print to the screen the number of /bin/bash processes that are currently running. Be sure to not include the processing that has /bin/bash as a command line argument! Hint: Look closely at grep!

5. Change directory to LinuxTest Display the current working directory to ensure you are in the correct location.

6. In a nice loop, invoke first.sh three times. Wait on the nth execution to finish before beginning the (n+1)th execution. Display the execution counter (1, 2, 3) immediately before invoking ./first.sh.

7. In a nice loop, invoke first.sh three times. Do **not** wait on the nth execution to finish before beginning the (n+1)th execution. Display the execution counter (1, 2, 3) immediately before invoking ./first.sh.

8. Return to the parent directory. Display the current working directory to ensure you are in the correct location.

9. List all the text files (*.txt) that exist in the current directory and all sub-directories.

10. Use the **`file`** command to determine which files from 9 are Unicode and which are ASCII. Write two commands, one to display just the ASCII files, one to display just the Unicode files. Write each command so that the only text printed to the screen is the filename (including the relative path to the file).

11. Using the two lists generated in 10, display the name and size of each file under the appropriate heading, ASCII or UNICODE. You may display more information than just the name and size.

12. Display a message that states which file type, ASCII or Unicode or neither, has more text files in the current directory and all sub-directories.

13. **BONUS**
In one command line string, determine the largest Unicode file and display that file's name and size to the screen.