CS507 Computing Foundation for Computational Science HW2

Shell Programming

Min Long

Instruction

Due time: Due: 9/28/2022, 23:59
Submission command: `submit minlong CS507 HW2`

Instruction to electronic submission: http://cs.boisestate.edu/~cs221/SubmissionProcedure.html

- The written assignment can be done in a pure text format (*.txt, for example, problem1.txt) on Onyx.
- The programming assignment should be presented with source codes.
- Each problem should have its own working directory, such as HW2/prob1, HW2/prob2 ... For example, the following table shows the structure of HW1 from the user "student1" and how to submit HW to us through Onyx

```
[student1@onyx:HW1]$ ls -l
drwxr-x---. 2 student1 Students  13 Sep 19  2021 prob1
drwxr-xr-x. 3 student1 Students  14 Sep 19  2021 prob2
[student1@onyx:HW1]$ cd prob1/
[minlong@onyx:prob1]$ ls
-rw-r-----. 1 student1 Students 943 Sep 19  2021 problem1.txt
$ cd ..
[student1@onyx:HW1]$ pwd
/home/student1/CS507/HW1
[student1@onyx:HW1]$ submit minlong CS507 HW1
```

Listing 1: A sample structure of homework and submission procedure.

- Your source codes (if any) must compile and run on Onyx.
- Documentation is important and proper comments are expected in your source code.
  - comments giving description of: purpose, parameters, and return value if applicable
  - other comments where clarification of source code is needed
  - proper and consistent indentation
  - proper structure and modularity

Don't ask us or your classmates directly for solutions (it happened); just try as much as possible. Be patient and enjoy coding!

Programming Problems

1. (10 pts) Write a shell script to check to see if the file "/home/myname/CS507/HW1/readme.txt" exists or not. Next, check to see if you can write (i.e., have "w" permission or not) to the file or not. Hint, you may need to use test construct ([or [[) and flags (e.g, -f, -d, ...) (see Page 37 in Lec02).

2. (20 pts) Create 5 txt files (a.txt, b.txt, c.txt, e.txt, f.txt) in your working directory.

   (a) Write a Shell script `count1.sh` without using any logic control structures to count the total number of .txt files in the current directory, and print out the number.

   (b) Write a Shell script `count2.sh` using for-loop to count the number of .txt files in the current directory and print out the number.

   (c) Write a Shell script `count3.sh` to print the name of files and the total number of files in the current directory.

   (d) Write the shell script that renames all txt files to begin with today's date in the following format: YYYY-MM-DD. For example, rename "a.txt" to "2022–09–21-a.txt". (Hint: try "date +%F command", %F is a flag showing the required format).

3. (20 pts)

   (a) Use for-loop to write a shell script `sum1.sh` that takes an unspecified number of command line arguments of integers and finds their sum. That is, the for loop should read all input arguments and count all arguments and add them together.

   (b) Use a different style of for-loop to write a shell script `sum2.sh` to compute the summation between 2 numbers, for example, $\sum_{i=1}^{100} i$. Obviously, the above method doesn't work, since it's not easy to list the entire numbers as augments.

   (c) Use a while-loop to write a shell script `sum3.sh` to compute $\sum_{i=1}^{100} i$.

   (d) Write a script that executes the command "bash sum4.sh". If the command return a 0 exit status, report "command succeeded" and exit the test construct with a 0 exit status. If the command returns a non-zero exit status, report "Command failed" and exit with a 1 exit status.

```
$ bash sum1.sh 1 10 20
31
$ bash sum2.sh 1 100
5050
$ bash sum3.sh 1 100
5050
$ bash exit-status.sh
bash: sum4.sh: No such file or directory
Attn: Command failed
```

Listing 2: Sample output.

4. (10 pts) The default rm command will not confirm before it deletes any regular files. Write a short script called saferm.sh, such that it will make a copy before deleting a single file, by do the following:

   (a) (3 pts) Take one and only one argument at the command line. That is, if the number of arguments is more than 1, print out a warning and use shell command exit to exit the script.

(b) (3 pts) Create a directory "removed_files" in the current folder if it is not already created. Otherwise print out a warning and continue.
(c) (2 pts) Copy the file indicated by the first argument to this "removed_files" folder.
(d) (2 pts) Remove this file in the current working directory.

5. (10pts) Let's do a guess game. $RANDOM holds a random integer in shell. Let's use it to generate a smaller random number in a range [0, 50] (Hint: use modulus %). Write a shell script which can randomly set up a number as answer and read input from user (Hint: use `read` command). When user's guess number doesn't match the answer, it will prompt the user to input another guess till they match.

```
$ echo $RANDOM
24459


$ bash numbergame.sh
The magic number is between 0 and 50.
Make your guess:28
28 is too low
The magic number is between 29 and 50.
Make your guess:30
30 is too low
The magic number is between 31 and 50.
Make your guess:50
50 is too high
The magic number is between 31 and 50.
Make your guess:40
40 is too low
The magic number is between 41 and 50.
Make your guess:45
45 is too low
The magic number is between 46 and 50.
Make your guess:48
48 is too high
The magic number is between 46 and 47.
Make your guess:47
47 is too high
The magic number is between 46 and 47.
Make your guess:46
You got it in 8 guessess!
```

Listing 3: Sample output.