CS 251: Bash Scripting and Linux Shell

- Handed out: 8/14 Due: 8/17 11pm
- Please write (only if true) the honor code. If you used any source (person or thing) explicitly state it. You can find the honor code on the web page.

Overview

The goal of this assignment is to make you familiar with some of the tools available on a typical bash shell, and a bit about scripting using bash.

The goals in this assignment can be accomplished using multiple ways. The goal is to do the assignment using bash constructs, and commonly available tools in `coreutils` under Gnu-Linux. You are not expected to write the answer in C or C++ or other mechanisms.

Pretasks

Make sure you link your previous assignment on your `public_html` directory so that the instructor can find it during viva hours.

Start thinking about your Box2D based project (due after midsem).

Tasks:

1. Given a C filename as input argument, write a script to print the names of VALID header files that are included. The script will print the answer on the standard output (print only the header filename, e.g. `iostream.h`).

   Notes: Watch out for spaces in the syntax of included files. Header files may be included in comments and thus should not be printed (not a "valid" head file). A sample file `power.c` has been given. Compile C file using something like

   ```
   gcc power.c -lm
   ```

   [Marks: 10]

2. Write a script to print the count of all the files and the folders present in the current directory. Your count should also include the so-called hidden files. Recursion is not expected.

   Notes: This task can be easily accomplished by `find`. We don't want you to use this. On cheaper routers such as the in your hostel, and embedded system (such as the one you find on the TV screen in airplanes, the `find` command is not available, but `ls` is available.

   [Marks- 6]

3. Write a script system_spec.sh to print the frequency of the processor in your Linux system. After that, also print the physical and virtual address sizes in the system. Script should only output the values required and nothing apart from that.

   <frequency_of_the_processor>
   <physical_address_size>
   <virtual_address_size>

   Notes: Ask us on Piazza on where to look for this info.

   [Marks- 8]

4. Write a script that implements a flexible "file open" system. It will take as argument the file to be opened. And it will read a configuration file whose format is as given in the provided file `myopen.cfg`.

   Each line of `myopen.cfg` corresponds to the specification of which program to use to open a file of a particular extension. Running the script with argument ¡filename¿.¡extension¿ should search through the configuration file, find the program to use for opening the given file (by matching the extension), and then use the program to open the given file. Your script should exit with exit value of 1 if no argument is given. It should exit with exit value of 2 if the configuration file is not found. It should exit with exit value of 3 if no program can be found corresponding to the given extension. It should exit with exit value 4 if the program corresponding to given extension does not exist. The configuration file myopen.cfg can be assumed to be in the HOME directory.

   [Marks- 10]

5. Write a script `script\_compile.sh` to compile all the C programs present in current working directory and generate two output files named "good.txt" and "bad.txt". good.txt should contain names of all the files which were compiled without any error whereas bad.txt should contain those file names which generated any kind of warning or error during compilation.

   Keep in mind your script should NOT print anything on standard output in ANY case. [Marks- 8]

6. Imagine you are working on a program, and whenever you write changes to the hard disk with a "save" the file is stored with a random name so that all saved versions of your program are available at all the times.

   Now you want to differentiate between the versions of files by appending the date and time of last modification.

   Write a script that does this work for you. Assume you have the desired set of files in your current working directory

   example :- abc.txt will get renamed to abc.txt_month_day_hour:min

   month will be the numbers 01-12, day will be a number between 01 and 31, hour will be 01-24, and min will be rounded to the minute in 00-59

   [Marks- 12]

7. You have a file `population.csv` data) where column 1 is the year, and column 2 has the population in that particular year. You have to compute for each year the population increase in that particular year. Write a script that generates `population\_inc.csv` that has year as 1st column and the population increase from the previous year in the 2nd column. [Marks- 7]

Submission Guidelines:

1. When you submit, please document individual percentages such as Student 1: 80%, Student 2:100%, Student 3:10%. In this example, the second student will get full marks, i.e., 10/10 and the first student will receive 8/10.

2. Do include a `readme.txt` (telling me whatever you want to tell me including individual percentages). Do include group members (name, roll number), group number, honour code, citations etc. Make sure that numbers are in sorted ascending order.

3. The name of the submission folder should be **lab04_groupXY_final.tar.gz**

How We will Grade You

- Honor Code and package complete in all respects +2. **Incorrect or incomplete -2**.
- Marks corresponding to all the question is given along with the question itself.