CS214 Advanced UNIX

Due: February 28, 23:59

Assignment 1

The purpose of this assignment is for you to better understand how to use UNIX tools in shell scripts. At the end of this assignment you will be more familiar with:

- pipelining, how it works and how to use it to simplify your scripts
- input/output redirection
- how the shell interprets special characters
- the use of regular expressions
- how to pass arguments to a script and use them inside the script
- shell variables.

Of course, we will cover these topics in more detail in the later sections and assignments; here we just look at the basics.

One good way of writing scripts is to start small and build the script from pieces: first we break the problem into small subproblems, we write the commands to solve each subproblem, test on some examples, and then use pipelining to solve the initial task. As the last step, we try to optimize the code.

Since this is the first assignment, I will start by mentioning that it is very good practice to comment your code; first for yourself when you look over the code in a couple of weeks, and then for everybody who will try to use your scripts. I suggest to use the following header for all your scripts:

```
#!/bin/bash/
#Created [here write the date] by [name]
#Script that [what your script does]
#Usage: [how to run your script; for example:  myscript.sh  -option  -argument  ]
```

Problem 1: Words statistics

For psychologists, patterns in our speech reveal important information about our background, education, and personality. For example, if during a conversation you use the word "competition" far more frequent than words like "day dreaming", chances are that you are a hard-working student in a competitive university, unless you are an aspiring novelist...

Assume for the moment that you are working on analyzing speech patterns. Your data comes from interviews and articles, and is stored as text files (for example, look at **interview1.txt**, **article1.txt** on the class web page). From data you want to extract information about which words are more often used, as well as which words are less frequently used. You store this relevant information in a file called **wordstat.txt**.

How should you proceed? One idea is to read each data file one at a time, and perform a number of text processing operations; after all, you do not want an exclamation sign to be considered as a word, and you do not make a distinction between "Competition" (as written at the beginning of a sentence) and "competition". Follow these steps:

1. Write a command that reads from a data file and removes punctuation. (Hint: try the **tr** command.)

2. Write a command that reads from a data file and changes all upper letters to lower case. (Again, you might want to look at the **tr** command.)

3. Write a command that reads from a data file and removes all tabbing.

4. Write a command that reads from a data file and deletes all empty lines. (Hint: try **sed**.)

5. Write a command that takes as input a file containing a word on each line, sorts the file alphabetically, and removes duplicates.

6. Finally, write a script called **ws.sh** that takes as argument a data file, and outputs in **wordstat.txt** the words in the data file, in decreasing order of their frequency. Each line should contain two columns: number of occurences, followed by word; for example:

   ```
   10 competition
   6  athletics
   2  bagels
   ```

   Hint: use **pipelining**. For example, you can first remove punctuation, then pipeline the result, line by line, to a process that converts upper letters to lower case etc. Hint: read carefully the man entry for **sort**.

7. Test your script on a sample data file; ideally, you should test each of the commands you write on (first small, then larger) data files. What happens if you test your script on a new data file? Depending on how you wrote the code, you might have lost the results in **wordstat.txt** for the first data file! Think how to update **wordstat.txt**; in other words, if "bagels" appears 2 times in your first file, and 4 times in the second file, it should appear 6 times in **wordstat.txt**, and it might not be less frequent that "athletics". Call the improved script **wsc.sh** ("c" from cumulative).

Problem 2: User data

Write a short script called **mydata.sh** that outputs on the screen information about your account and your activity. You would like to see on the screen the following:

- a greeting that contains your user name
- your working directory
- your login shell
- your host computer
- the display number
- the full path
- the process you are currently running
- how many process you are currently running
- the jobs that are currently executing
- how many times you have logged in recently (hint: check the **who** command).

Say for example that your user name is **john**; to check your user name, simply type

```
$ echo $USER.
```

One possible result of executing **mydata.sh** is:

```
Welcome john!
Working directory: /amd/panda/b/john/cs214
Loggin shell:  /bin/csh
Host:  turing
Display: 128.84.99.7:0.0
```

```
Path: usr/local/gnu/bin:/usr/local/graphics/bin:.
Processes:
 F S   UID   PID  PPID  C PRI NI     ADDR     SZ    WCHAN TTY      TIME CMD
 8 S   998 23491 23485  0  99 20        ?    309        ? pts/4    0:01 bash
 8 S   873  8015  7996  0  81 20        ?   2656        ? pts/1   30:16 pine
 8 O     0  6703 23491  0  91 20        ?    240          pts/4    0:00 ps
 8 S   998    80 23491  0  41 20        ?    931        ? pts/4    0:02 emacs
 8 S   998 23492 23491  0  41 20        ?    934        ? pts/4    0:03 emacs
5 processes
Jobs:
[1]-  Running                 emacs week1.txt &
john has logged in
    1
```

Recall that by typing

```
$ printenv | less
```

you can see the variables in your environment and their values. For example, **HOME** is a variable that contains the path to your home directory. Recall also that to see the value of a variable **var** you can type

```
$ echo $(var)
```

The first dollar sign is the bash prompt, the second dollar sign is a special character! Recall also that you can store the result of a command into a variable.

What to turn in

For problem 1, please turn in the following:

- a script file **misc.sh** that contains each of the first 5 commands; I will then comment all but one command and test it, and do this for all 5 commands
- **ws.sh**
- **wsc.sh**.

For problem 2, please turn in **mydata.sh**. As a general comment, make sure you set permissions on your scripts so that I can read and run them; for example, use

```
$ chmod a+rx mydata.sh
```

After you complete the Questionnaire, I will create accounts for you on csuglab and on CMS. We will use CMS submission for this assignment. If by any chance the system is down, please send the files to petride@cs.cornell.edu with the subject **cs214 hw 1**.

The deadline is strict, unless you talk to me in advance and explain me the reason for your late submission.

How to get help

If you encounter problems, this is how you can get help:

- check the **man** pages for the commands I suggested using, or for others that are useful; there is a link to the bash man page from the course web page
- check the newsgroup to see if other people have similar problems, or post your question (of course, do not post the whole problem on the newsgroup)
- come to ffice hours, and if you cannot make it, send me email for an appointment.

Good luck!