Computing Lab #2: Unix scripting, LaTeX

Due: Tuesday, September 19, 2017

Summary: The power of scripting. Introduction to LaTeX. (Please consult the References posted to myCourses for more information on any of the topics discussed in this lab)

Points: 10

I. TASK #1: BASIC UNIX SKILLS: MORE COMMANDS AND WRITING A SCRIPT (5 POINTS)

Task: Graph your web browser's CPU usage vs time. You are encouraged to use the internet and Unix manuals (for example "man ps" to find out what ps does) to solve this problem. You will write a Unix script to solve this problem.

- **Part a** Build a Unix command which returns the CPU usage of your web browser.

  1. Make sure all the web browsers are closed. Then open up a new one. Make sure only one web browser and tab are open.
  2. Your operating system assigns every program running on your computer a process ID (PID). Use the Unix program ps to find the web browser's PID. The best way to do this is display all PIDs and then find the one corresponding to the browser.

     HINT: open up the manual by doing "man ps", then read the "EXAMPLES" section
  3. Use the Unix program ps to display information for only the web browser process. The output should look something like

     ```
       PID TTY          TIME CMD
      2221 ?        00:39:56 chrome
     ```
  4. Use the Unix program ps to display the "cpu utilization of the process" (the percentage of the CPU being consumed by the web browser) for the web browser. The output should look something like:

     ```
     %CPU
      1.1
     ```

     HINT: you will need to supply the program ps an option "-o %cpu"
  5. Use the Unix programs ps, tail, and the pipe operator "|" so that the only information displayed is the CPU usage:

     ```
     1.1
     ```

     HINT: "tail -n 1" will display only the last line of a file

- **Part b** Write a Unix script which displays the web browser's CPU usage every 1 second.

  1. Let us first write a simple Unix script to print hello. To do this, use vim or emacs to create a file called "hello.sh" and edit it as follows:

     ```
     #!/bin/bash

     echo "Hello!"
     ```
  2. Use the Unix program chmod to give yourself permission to execute "hello.sh". chmod changes the file permissions (who is allowed to read, write and execute the file). Give yourself execute permission of the file hello.sh

     ```
     >>> chmod u+x hello.sh
     ```

     NOTE: u = user and "u+x" means add execution permission to the user. You are the user.

3. Now execute your first Unix script

```
>>> ./hello.sh
```

4. Sometimes its useful to have variables within a script. In a Unix script, variables work as follows. Edit your hello.sh script to be

```
#!/bin/bash

VAR=17 # Unix variable, assigned 17

echo "Hello! Var = " $VAR  # Notice you need a dollar sign, which evaluates VAR
```

and now try running it.

5. You are now ready to write a Unix script which prints the web browser's cpu usage every 1 second. To do this, use vim or emacs to create a file called "cpu_usage.sh". Use my starter script given below, and fill in the missing parts where you are asked:

```
#!/bin/bash

MYPID= # FILL THIS IN WITH THE PID NUMBER OF THE BROWSER PROCESS
echo "PID IS" $MYPID
ps -p $MYPID -o comm # this will display the process name

while [ "1" == "1" ]  # trick to keep the loop running
do
  x=‘date +%s‘  # this will grab the current time from your computer
  y=‘FILL THIS IN WITH THE UNIX COMMAND YOU BUILT TO GET THE CPU USAGE‘
  sleep 1  # pause for 1 second
  echo $x $y  # print the current time and CPU usage
done
```

6. Use the Unix program chmod to give yourself permission to execute "cpu_usage.sh".
7. Run the script. Every one second your script should display "Unix coordinate time" in seconds and the cpu usage. To kill your script, hit ctrl+c.
8. Use the Unix redirect operator (see References in myCourses to read more about pipes and redirections) to redirect your script's output to another file called "CPUData.dat". Let your script run for a minute or two.

```
>>> ./cpu_usage.sh >> CPUData.dat
```

Meanwhile, browse the web, look at videos and other compute intensive work with your browser.

9. Kill your script and inspect the newly created CPUData.dat file.
10. Using any means (Matlab, Python or something else) make a simple graph showing time vs CPU usage. See if you can spot interesting events like when you started a video.

Your answers to Task 1 are the script cpu_usage.sh and the figure showing CPU usage vs time.

II. TASK #2: INTRODUCTION TO LaTeX(5 POINTS)

Task: Setup your laptop for LaTeX. Compile a LaTeXtemplate. Please see the myCourses website for some additional references and help.

LaTeX(https://www.latex-project.org/): "LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. LaTeX is the de facto standard for the communication and publication of scientific documents."

In order to use LaTeX, you must first install it on your system; for Macs, the MacTEX package is a relatively painless way to do so: https://tug.org/mactex. This installs command-line tools for you to use. However, some people

may prefer a GUI interface. There are many operating-system- specific tools for this. E.g., TEXShop and TEXworks packages on Mac. On linux systems (such as Ubuntu) you can use the apt-get package manager, or its gui interface synaptic, to download LaTeX. Another option is to use web-based LaTeXsoftware like https://www.sharelatex.com/.

For this problem you will compile a LaTeXfile into a PDF. This will check you have LaTeXproperly installed.

Move into the class git project folder (located on your computer, since you've already cloned it), and pull in some recent changes I've made:

```
>>> git pull
```

You should see a new folder with a latex file called simpleSolution.tex. If you'd like, you can use this latex file as the basis for any future homeworks.

Compile this latex file into a pdf file with

```
>>> pdflatex simpleSolution.tex
>>> pdflatex simpleSolution.tex
```

There may be other ways to compile it too – you should use whatever method you're most comfortable with.

III. UPLOAD YOUR WORK TO GIT

Congratulations! You've completed the lab. Now lets upload your work to git.

1. Open up a terminal and change into your personal git project folder (not the class folder).

   ```
   >>> cd /path/to/your/git/folder #  Each person's computer is different -- so you'll need to find it.
   ```

2. Create a new directory to place your work

   ```
   >>> mkdir lab2
   ```

3. Move your answers to task 1 (a script and a figure) and task 2 (the pdf you compiled) into the lab2 folder.

4. Make git aware of this new file by doing

   ```
   >>> ls     # COMMENT: make sure you see "lab2" in the folder
   >>> git add lab2
   ```

5. Commit the files to the git project along with a commit message:

   ```
   >>> git commit -m "Lab 2 answers"
   ```

6. Upload the files to Bitbucket

   ```
   >>> git push
   ```

7. Finally, open the web browser and go to your bitbucket page. Check the files have been uploaded.