HW2: Data Plot

Due: Thursday, October 21, 2021, at 11:59pm

Assignment Goal

The main goal of this assignment is to learn more about shell scripting and using regular expressions and string processing programs, particularly `grep` and `sed`. You will also learn about accessing files from the web and incorporating an R script in your work.

Hints and Tips

Taken in its entirety this assignment may seem quite long. You are encouraged to read through the described approach which is designed to break the assignment into manageable pieces. By completing and testing each piece independently you will be successful in the overall project.

Documentation

In addition to the lecture notes, you may find "The Linux Pocket Guide" a useful reference for completing this assignment.

Online manuals:

- bash
- sed
- grep
- Comprehensive R Network

In general, whenever you need to use a new tool, you should get into the habit of looking for documentation online. There are usually good tutorials and examples that you can learn from. As you work on this assignment, if you find that you would like more information about a tool (sed, grep, or R), try searching for the name of the tool or the name of the tool followed by keywords such as "tutorial" or "documentation". Also be sure to use the regular Unix documentation (man pages and info command), and experiment with commands to try out various options and see what they do.

Implementation Details

Download the file: hw2.tar. Extract all the files for this assignment using the following command:

```
    $ tar -xvf hw2.tar
```

You should now see a directory called `hw2`.

If you see it, you are ready to start the assignment. If this did not work for you, please post a message on the HW2 discussion describing the problem to see if someone has any ideas, or contact a TA or the instructor.

Background

You are interested in evaluating how 'user-friendly' different CSE courses are. You have a theory that courses with more information on their web pages will be easier to navigate, and you want to test your theory. To achieve this you have decided to plot the sizes of the front page for each course and look for trends in the data. In order to get the most current information you want the front page for 21au (Autumn quarter, 2021).

To get started you use the CSE webpage on which all the current courses are listed. You could look at each course webpage manually, but you decide to write a script to automatically extract webpages, measure their size, and generate the required plot.

Part 1: Extract the course URLS from the main webpage

We have provided the web page listing all the courses in html format. (This is the `courses-index.html` file provided if the assignment set up.) To run an experiment automatically on each URL in this list, we need to extract the URLs and write them into a text file. There are several ways in which this can be done, and different utilities (`sed`, `grep`) can help.

You **must** use `grep` and/or `sed` even if you know other programs or languages (`awk`, `perl`, `python`, ...) that could do similar things in different ways. But it's fine to use `egrep` and extended regular expressions in `sed` and `grep` if you wish.

In a file called `getcourses`, write a script that extracts the course URLs and writes them into a text file. The script should take two arguments: the name of the output file for results and the name of the input html file. It should extract valid course URLs and not other URLs.

For example, executing:

```
    $ ./getcourses courselist courses-index.html
```

Should write content similar to the following into `courselist`:

```
http://courses.cs.washington.edu/courses/cse120/21au/
http://courses.cs.washington.edu/courses/cse131/21au/
http://courses.cs.washington.edu/courses/cse142/21au/
...
```

If the user provides fewer than 2 arguments, the script should print an error message and exit with a return code of 1.

If the text file provided as argument (for the output) exists, the script should overwrite it with a warning.

If the html file provided as argument (for the input) does not exist, the script should print an appropriate error message and exit with a return code of 1.

If the script does not report any errors, it should exit with a return code of 0.

Hints: step-by-step instructions

1. courses-index.html is taken from https://cs.washington.edu/education/courses/. You can view this page in a web browser to see what is there, and if you right-click and 'inspect' you can see which html code is associated with which part of the page.
2. Use `grep` to find all the lines that contain the string `http`. Test if it works before proceeding to step 2.
3. Use `sed` to replace everything that precedes the URL with the empty string. Test if it works before proceeding to step 3. For this assignment, your sed command(s) must match the `http://...` URL strings themselves, not surrounding text in the table. (i.e., your sed command must use a pattern that matches the URLs although, of course, it probably will contain more than that if needed to isolate the URL strings. But it can't just be .* surrounded by patterns that match whatever appears before and after the URLs in this particular data file.)
4. Use `sed` to repl`. At this point you can also add the `21au` that points to the webpage for this quarter. Test if everything works - does your final list of URLs look correct, or do you need different replacement text?
5. Notice that there are some decisions you can make about which of the courses get included in your analysis. You can make any decision that seems reasonable to you about which courses to include and which ones to leave out. You can comment on your decision in your code.

Part 2: Download a page and compute its size

In a file called `perform-measurement`, write a bash script that takes a URL as an argument and outputs the size of the corresponding page in bytes.

For example, executing your script with the URL of homework 1 on the class website as argument:

```
   $ ./perform-measurement
http://courses.cs.washington.edu/courses/cse374/21sp/assignments
/index.html
```

should output *only* 5828 to standard output:

```
5828
```

(This number was correct at the time this assignment was prepared, but might be somewhat different if the page is modified some time in the future.)

If the user does not provide any arguments, the script should print an appropriate error message and exit with a return code of 1.

If the user provides an erroneous argument or if downloading the requested page fails for any other reason, the script should simply print the number "0" (zero). In this case, or if the page is downloaded successfully, the script should exit with a return code of 0 after printing the number to standard output.

Hints:

- The `wget` program downloads files from the web. Use `man wget` to see its options.
- Your script may create *temporary* files if you want. The `mktemp` program produces unique file names for temporary files. If you create a temporary file, you should remove it before your script exits. Generally it is best to create temporary files like this in `/tmp`.
- Experiment with the following commands: `wc a-test-file` and `wc < a-test-file`.
- To suppress the output of a command, try to redirect its output to `/dev/null`. For example try `ls > /dev/null`

Part 3: Run the experiment by measuring each webpage

To perform the experiment, your need to execute the script `perform-measurement` on each URL inside the file `courselist`. Once again, you would like to do this automatically with a script.

In a file called `run-analysis`, write a shell script that:

- Writes its output to a file. The name of that file should be given by the user as the first argument.
- Takes a file with a list of URLs as the second argument and executes `perform-measurement` on each URL in the file.
- For **each valid** URL, `run-analysis` should produce the following output, separated by spaces:
  course-number page-size
  For invalid URLs (i.e. those where `perform-measurement` outputs 0), it should produce no output.
  The `course-number` is the three-digit course number. You can extract this from the URL you give to `perform-analysis` using a similar method to the one you used to parse the original course listings. You will want only the three digit numerals - not any letter section extensions. The `page-size` is the result of `perform-measurement`.
- Because it can take a long time for the experiment to finish, your script should provide feedback to the user. The feedback should indicate the progress of the experiment.
  - Before executing `perform-measurement` on a URL, your script should print the following message: "`Performing byte-size measurement on <URL>`".
  - Once `perform-measurement` produces a value, if the value is greater than zero, the script should output the following message: "`...successful`". If the value is zero, this means some error has occurred, and the script should output the following message: "`...failure`".
- When `run-anaylsis` finishes, it should exit with a return code of 0.

To debug your script, instead of trying it directly on `courselist`, we provide you with a smaller file: `popular-small.txt`. You should execute your script on `popular-small.txt` until it works, and then work on `courselist`.

Executing your script as follows:

```
   $ ./run-analysis dataout popular-small.txt
```

Should produce output similar to the following:

```
Performing byte-size measurement on http://courses.cs.washington.edu/courses/cse374/20sp/
...successful
Performing byte-size measurement on http://courses.cs.washington.edu/i.will.return.an.error
...failure
Performing byte-size measurement on http://courses.cs.washington.edu/courses/cse374/20sp/assignments/hw3.html
...successful
```

And the content of `results-small.txt` should be similar to the ones below. *Note that the exact values are slightly different from the following numbers.*

```
374 3803
374 19098
```

Part 4: Put all the parts together and generate a plot

It is hard to understand the results just by looking at a list of numbers, so you would like to produce a graph. More specifically, you would like to produce a scatterplot, where the x-axis will show the course number and the y-axis will show the size of the index page.

Luckily, you've used R for some of your statistics courses, so you find a script called `scatterplot.R` Note that this script expects your experimental results to be stored in a file called `dataout`.

You will procude a plot by calling the R script at the command line. You can make sure your system has R installed by typing `R --version` at your command prompt. If it is not installed, you may need to install it.

The script should produce a file called `scatterplot_out.jpg`. You can view this file with any image viewer.

Part 5: Write-up

You will want to submit one more file, which should be called `hw2summary`. This file should contain the following:

1. Your name, CSE 374 Homework 2, date
2. This list of commands you run to complete this analysis, once the scripts are done. This is no more than one line for each of the scripts described above.
3. You may include an explanation about decisions you made in implementation. This may lend clarity to your solution.
4. A brief statement about any trends you see in the data based on the scatter plot. Do these trends allow you to draw any conclusions?

5. Notice that the R script also prints a number to the terminal when it runs. Include a brief description of what this number represents. (You will want to look at the R script itself for help with this.)

Assessment

- This homework will be worth a total of 50 points. There are four major components which will be approximately equal in value, plus a smaller weight on the write-up.
- Identifying information including your name, CSE 374 Homework 2, the problem number, and the date should appear as comments in each of your files (with the exception of the .jpg file.
- Your grade depends on turning in correct scripts, etc., that run with `bash` on either of our reference systems (`klaatu` or the current CSE Linux virtual machine).
- In good style, including indentation and line breaks. Refer to the CSE 374 guide for hints on style.
- Scripts should handle arguments with spaces in them, and print error messages appropriately
- You should pay attention to the details of the specification, as scripts will be evaluated against those standards. However, there may be things that are left unspecified (such as whether you include courses that have a letter appended to their number), and you may choose a reasonable course of action in those cases.
- You will want to use the tools suggested in this write-up whenever possible. There are many ways to solve this problem, but, we are guiding you towards a solution designed to exercise the tools we have taught in class.

Turning In

Please submit your files via Gradescope. (You should have received an email inviting you to Gradescope. You will find the assignment HW2 listed under course CSE 374. You may upload multiple files. You will need to submit `getcourses`, `perform-measurement`, `runanalysis`, `scatterplot-out.jpg`, and `hw2summary`.