CSE410 - Advanced Programming In UNIX

Assignment 2

Assignment goal

You will get experience writing bash scripts. You will experiment with the grep, sed, and awk utilities. You will learn about a program called gnuplot for plotting files.

Note: The instructions for this assignment may seem a bit long. This is because we try to give you plenty of sample outputs and hints for each question. We hope this will help you complete the assignment faster.

Submission deadline: October 12, 2015 Midnight.

Online manuals:

- bash
- awk
- sed
- grep
- gnuplot

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 (awk, sed, grep, or gnuplot), try searching for the name of the tool or the name of the tool followed by keywords such as "tutorial" or "documentation".

Documentation

In addition to the lecture notes, you may find many useful reference for completing this assignment.

http://www.makeuseof.com/tag/5-downloadable-books-to-teach-yourself-linux/

There is also a bash manual (http://www.gnu.org/software/bash/manual/bashref.html) available on the web

Command line text editor (vim or nano): http://www.openvim.com/

Getting ready

Download the file: hw2.tar.gz.

Extract all the files for this assignment using the following command:

> `tar zxvf hw2.tar.gz`

You should now see a directory called `hw2.`

If you see it, you are ready to start the assignment.

1. Download a page and compute its size

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

For example, executing your script with the URL of the class syllabus page as argument:

```
> ./perform-measurement.sh
http://www.cs.washington.edu/education/courses/303/10wi
/syllabus.html
```

should output *only* the number 8011:

```
> 8011
```

If the user does not provide any arguments, the script should print an informative error message and exit.

If the user provides an erroneous argument or downloading the requested page fails for any other reason, the script should simply print the number "0" (zero).

Hints:

- Remember to change permissions on `perform-measurement.sh` to make it executable.
- The `wget` program downloads files from the web. Use `man wget` to see its options.

- Your script may create *temporary* files. The `mktemp` program produces unique file names for temporary files. If you create a temporary file, you should remove it before your script exits.
- Experiment with the following commands: `wc my-test-file` and `wc < my-test-file`.
- To suppress the output of a command, try to redirect its output to `/dev/null`. For example try`ls > /dev/null`

2. Parsing the html list of websites

The list of popular websites is in html format. 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 (awk, sed, grep) can help.

We would like you to use `grep` and `sed`.

In a file called `parse.sh`, write a script that extracts the URLs and writes them into a text file. The script should take two arguments: the name of the input html file and the name of the output file for the results.

For example, executing:

```
> parse.sh popular.html popular.txt
```

Should write the following content into popular.txt:

```
http://www.100bestwebsites.org/oledata.mo
http://www.yahoo.com/
http://www.google.com/
...
```

If the user provides fewer than 2 arguments, the script should print an error message and exit.

If the html file provided as argument does not exist, the script should print an error message and exit.

If the txt file provided as argument (for the output) exists, the script should simply overwrite it without any warning.

Q: How come popular.txt contains only 88 urls? Is it a bug in your script or a bug in the data? You don't need to answer this question in writing, just think about it for yourself.

Hints: step-by-step instructions

- First, use `grep` to find all the lines that contain the string `http`. Test if it works before proceeding to step 2.
- Second, use `sed` to replace everything that precedes the URL with the empty string. Test if it works before proceeding to step 3.
- Third, use `sed` to replace everything after the URL with the empty string as well. Test if everything works.

Note: the first line that will appear in the result file (`http://www.100bestwebsites.org/oledata.mo`) should technically not be there, but don't worry about it. Just leave it there or remove it manually.

3. Running the experiment

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

In a file called `run-experiment.sh`, write a shell script that:

- Takes a file with a list of URLs as argument and executes `perform-measurement.sh` on each URL in the file.
- For **each** URL, `run-experiment.sh` should produce the following output, separated by spaces:

  ```
  rank URL page-size
  ```

  The `rank` of a page is the line number of the corresponding URL in `popular.txt`. For example, the URL on the first line has rank 1, the URL on the second line has rank 2, and so on until the last URL/line in the file. The `URL` is the same string as the argument you gave

to `perform-measurement.sh`. The `page-size` is the result of `perform-measurement.sh`.

- `run-experiment.sh` should write its output to a file. The name of that file should be given by the user as second argument.
- If `perform-measurement.sh` returns zero for a URL, `run-experiment.sh` should not write any output to the file for that URL.
- 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.sh` on a URL, your script should print the following message: "`Performing measurement on <URL>...`".
    - Once `perform-measurement.sh` produces a value, if the value is greater than zero, the script should output the following message: "`...success`". If the value is zero, this means some error has occurred, and the script should output the following message: "`...failed`".

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

Executing your script as follows:

```
> run-experiment.sh popular-small.txt results-small.txt
```

Should produce the following output:

```
Performing measurement on
http://www.cs.washington.edu/education/courses/303/10wi
/
...success
Performing measurement on
http://www.cs.washington.edu/education/courses/303/10wi
/syllabus.html
...success
Performing measurement on http://i.will.return.an.error
...failed
Performing measurement on
http://www.cs.washington.edu/education/courses/303/10wi
```

```
/schedule.html
...success
Performing measurement on
http://www.cs.washington.edu/education/courses/303/10wi
/homework.html
...success
Performing measurement on
http://www.cs.washington.edu/education/courses/303/10wi
/resources.html
...success
```

And the content of `results-small.txt` should be similar to the ones below. Note that the exact values will change as we edit the class website! In particular, schedule.html will grow with every lecture!

```
1
http://www.cs.washington.edu/education/courses/303/10wi
/ 554
2
http://www.cs.washington.edu/education/courses/303/10wi
/syllabus.html 8011
4
http://www.cs.washington.edu/education/courses/303/10wi
/schedule.html 15612
5
http://www.cs.washington.edu/education/courses/303/10wi
/homework.html 859
6
http://www.cs.washington.edu/education/courses/303/10wi
/resources.html 1291
```

As another example, after executing your script as follows:

```
> run-experiment.sh popular.txt results.txt
```

The file `result.txt`, should contain results similar to the ones shown below (when you run your experiment, the exact values may differ)

```
2 http://www.yahoo.com/ 123874
3 http://www.google.com/ 5639
4 http://www.amazon.com/ 90405
...
```

4. Plotting the results

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 rank of a website and the y-axis will show the size of the index page.

Luckily, you talk about your problem to your friend Alice. She suggests that you use a program called`gnuplot` to produce the graph. Because she used it many times before, Alice helps you write the necessary gnuplot script called `produce-scatterplot.gnuplot.` Note that the gnuplot file expects your experimental results to be stored in a file called `results.txt.`

Produce the graph with the following command:

```
> gnuplot produce-scatterplot.gnuplot
```

The script should produce a file called `scatterplot.eps`. You can view it with `gv`: (Can use any image viewer)

```
> gv scatterplot.eps
```

Or you can turn it into a pdf file (You can skip if not working)

```
> epstopdf scatterplot.eps
```

You find that it is still difficult to draw conclusions about the sizes of the index pages by looking only at the scatterplot. You decide to produce a CDF instead.

The CDF, or cumulative distribution function, shows for each file-size x, the fraction of `index.html`files that have a size no greater than x. Writing the script to produce the CDF is an extra-credit question (see problem 5 below), but we would like everyone to see the result. We thus provide you read and execute permissions on the binary executable version of our solution script. Plot the CDF by executing the following two commands. You can thus produce the CDF by executing the following two commands:

```
> /cse/courses/cse303/10wi/bin/transform.sh.x
results.txt transformed-results.txt
```

```
> gnuplot produce-cdf.gnuplot
```

The script should produce a file called `cdf.eps`.

Note: because you need both execute and **read** permissions to execute a bash script, we replaced `transform.sh` with a binary executable version `transform.sh.x`. We created this version with a publicly available shell script compiler: http://www.datsi.fi.upm.es/~frosal/frosal.html

Write your answers to the following questions in a file called `problem4.txt`:

Q1: Examine the gnuplot file `produce-scatterplot.gnuplot`. Ignoring the first line, explain what the rest of the script does.

Q2: Looking at the scatterplot, what can you conclude about the relationship between the popularity of a site and the size of its `index.html` file? Are these result what you expected?

Q3: Looking at the CDF, what is the median size of the `index.html` files?

Q4: Examine the file `transformed-results.txt`. Which website has the smallest `index.html`? Which website has the largest `index.html`? Which website has the median-size `index.html`? Note that in file `transformed-results.txt`, the first column shows the file size in `Kilobytes (KB)`. The second column shows the cumulative fraction (expressed as a percentage value between 0 and 100) of files with an `index.html` size no higher than the value in the first column. The third column shows the URL of the website whose `index.html` has exactly the size shown in the first column (the URL appears without the leading "http://" and without the trailing "/").

Q5: Looking at the CDF, what can you conclude about the distribution of the file sizes? Is it what you expected?

5. Extra credit: transform.sh

Write the script `transform.sh`.

To write `transform.sh`, one approach is to proceed in three steps:

- Step 1: use `sort` to order the results by increasing file-size.
- Step 2: use `sed` to remove the "`http://`" and the trailing "`/`" from the URLs.

- Step 3: iterate over the sorted results to compute the CDF.

Note: Please attempt this question only if you found the rest of the assignment easy.

Submission

You should turn in the following files:

- Problem 1: perform-measurement.sh
- Problem 2: parse.sh
- Problem 3: run-experiment.sh, results.txt
- Problem 4: problem4.txt, transformed-results.txt
- Optional problem 5: transform.sh

In Piazza Submission process:

Type: note

To: instructors

Folder: assignment N (N is your assignment number)

Summary: Your name and ID, Assignment N

Details: small description and attachment in zip