CS 3423.02

Assignment 1: Unix Utilities and BASH scripts

Due: Midnight next Monday (12:05am of 09/15/2014)

1 Overall Description:

Find out the occurences of each word (A Word Count Example) inside a directory or a file. Those files will be normal text files.
This project includes several parts and they will be graded separately. For example, if you made a mistake in the Part 1, but you still have a chance to get all points for the other three parts.

Description of these different parts can be seen in the following:

- Part 1: Find out all files related to INPUT (20%). The input parameter can be a directory or a file. If it is a directory, we should find out all files inside the given directoy by leveraging on existing utilities. If it is a file, we are done. Then we will redirect all of these found files into a OUTPUT file so that we can analyze these files one by one. Thus, we don't need to care about the difference of input any more.
- Part 2: Calculate the occurances of words inside a file (40%). We can ouput all occurrences of each word to a file.
- Part 3: Get the specified number of most/least popular words (20%). We will sort those records according to the number of occurrences. Then, we can output those most/least popular words.
- Part 4: Integrate these different parts together (20%).

1.1 Modular Programming

These different parts can be developed and evaluated separately. Also, you should do this. That is the idea of modular programming. Actually, that is the idea why you should have functions, files, and modules in every language.

1.2 General Rule

- All the file names should strictly follow those names that described in the following. Otherwise, you can only get 80% percent even if you do it all right for an assignment. Since we are using the autograding system to grade all program assignments in this semester, not following these rules can greatly increase the amount of workload for me and TA.
- All programs should be able to handle those errors in the first place. Failing to do this will loose some points. More information can be seen at
http://en.wikibooks.org/wiki/C_Programming/Error_handling.
- Readablity of a program. We will give some additional points to those programs with good comments and styles, while those ones are not readable can be deducted from some points.

2 Part 1

2.1 Submission Rule:

The script name for this part is **./getfiles.sh**, which is a bash script. The first line of this script will be:

```
#!/bin/bash
```

This script can be run like this:

```
  ./getfiles.sh INPUT OUTPUT
```

INPUT is the input file or directory and OUTPUT is the output filename that includes all regular files in the specified directory or file.
IMPORTANT: when there is an error in the input parameters, this script will return the value “1”. To do that, we can use the following statement:

```
exit 1;
```

2.2 Suggested Tests:

Make sure that you can verify your scripts using the following test cases:

1. The number of arguments is not correct. We only accept two arguments.
2. INPUT is the same as OUTPUT.
3. This given file or directoy can be not existing. Thus, OUTPUT will be empty.
4. INPUT is a file.
5. File name, either INPUT or files in the specified directory, can include space in the middle.
6. Directory may have subdirectories.

3 Part 2

3.1 Submission Rule

The script name for this part is **./wordcount.sh**, which is a bash script. The first line of this script will be:

```
#!/bin/bash
```

This script can be run like this:

```
  ./wordcount.sh INPUT OUTPUT
```

INPUT is the file that lists all regular files, one file each line. This can be the OUTPUT of the Part 1. OUTPUT is the output file that includes the results of word count. This output file includes a lot of lines, where each line describes the occurrences of a word.
The format of each line will be like this:

```
COUNT WORD
```

COUNT is the occurrence number of a word and WORD is the actual word. These two fields are separated by a Tab space.
One example will be like this:

```
50 my
100 word
20  count
```

3.2 More description

When we implement this part, we should make sure the following things:

- There is no different between uppercase and lowercase. For example, “We”, ”we” and ”WE” are considered the same and should be counted together. Thus, in end of output, we only output those number of occurrences of lowercase. If we have (“we”, “we”, “WE”), we will output:

  ```
  3 we
  ```

- We focus on those words made out of the alphabet, or connected with hyphens. For example, “we're” will only count “we” here while “'re” will be discarded. However, “accident-prone” will be counted as a word.
- We can simply discard the following cases.
  - Anything that are not consisted of letters. For example, words starting with an apostrophe (') or a hypen (-).
  - Words made with numbers. For example, 1234 and 1234a will be discarded.
  - HTML tags

3.3 Implementation Tips

We suggest to use SHELL, not awk. We can create two arrays, one is wordArray and another wordCountArray. wordArray is used to keep track of words, while wordCountArray is used to keep how many occurances of each word.

Total procedure: We handle each file according to the INPUT parameter. For each file, we will handle each line one by one. For every line, we translates a line into an array with different words. For each word, we first analyze whether this word is a valid word. If it is a valid word, then we check whether this word is inside the existing array. If yes, then we increment this corresponding item in wordCountArray. If not, then we add this item to wordArray.
After we handle all these files, we output different words and its count to the output file.

3.4 Suggested Tests:

Make sure that you can verify your scripts using the following test cases:

1. File name can include space in the middle.
2. Those words with special characters.
3. We can try our scripts on a large file, which you may try http://mapreduce.stanford.edu/datafiles/word_count.tar.gz.

4 Part 3

4.1 Submission Rule

The script name for this part is **./sort.sh**, which is a bash script. The first line of this script will be:

```
#!/bin/bash
```

This script can be run like this:

```
  ./sort.sh INPUT OUTPUT
```

The INPUT of this part will be the output of the second part, which has the format like the following:

```
50 my
100 word
20  count
```

After the sorting, we expect the following result.

```
100 word
50 my
20  count
```

The word with the more occurrences will appear first. Thus, we can use "head" and "tail" to see those words with the most/least number of occurrences.

4.2 Suggestions

We may use the "sort" utilities here.

5 Part 4

5.1 Submission Rule

The script name for this part is **./total.sh**. It will be run like this:

```
    ./total.sh INPUT NUMBER
```

We can integrate different parts together to list NUMBER of words, with their number of occurrences. The list will start with the biggest number of occurrences.