# Finding Palindromes using Bash

Nahimul Islam

November 19 2017

# 1 Introduction

Assignment requires a Bash script that will identify palindromes found in a txt file provided by stdin or file specified from command line argument.

# 2 Implementation

Bash script `palindrome`:

```
#!/usr/bin/env bash
# path to interpreter as to support compatibility
# Finding palindromes from a text file stdin

#create dictionary file
grep -E '.{3}' /usr/share/dict/british-english > db.txt
```

Use of `grep` to search for Linux dictionary file and return 'British English' if found and output into a text file db.txt to use later. `-E ' .{3}'` is an extended regular expression that finds the pattern of the directory to retrieve desired file. This should allow a file to be created so can be used to compare the palindromes found later and remove non-dictionary words. Another alternative could be to include a text file containing dictionary words but retrieving from Linux allows an unnecessary file to be included, although can be removed later. Problem can be that it may not be able to find the desired directory.

```
#create remove word to test file
touch removeW.txt
```

Create empty text file `removeW.txt` using `touch` as had trouble with file not existing in the directory – possibly issue with `rm` command, so create the file at the start of the script. This may not be needed as the grep command `grep -vwF -f db.txt finish.txt > removeW.txt` should create the `removeW.txt` file but had errors with file not being created/existing.

```
#palindrome function
#
 palin()
{
sed 's/[0-9]*//g;s/[/\._-]//g'\
|tr -d '[[:punct:][:digit:]@]' \
| sed -E -e '/^(.)\1+$/d' \
| tr -s '[[:space:]]' \
| tr '[[:space:]]' '\n'
```

```
}
```

This first command in the pipeline works by using `sed` remove the numerical `'[0-9]'` and special characters `'[/\._-]'` from the file input stream. The `'g'` at the end of the string checks globally. Next is using `tr -d` to delete punctuation and digits if the above `sed` command has not done so. `"@"` character is an array like construct of parameters to scan through the text finding words at different positions. `sed -E -e '/^(.)\d'` scans values starting at `^` through `(.)` then deleting using the selector. So, this function should delete the words that are not needed.

```
paste <(palin <"$1") <(palin <"$1" | rev) \
| awk '$1 == $2 && (length($1) >= 3) { print $1 }' \
| sort | uniq -ic > done.txt
```

This uses the `palin` function and deletes words less than three letters. `Awk` used to scan letters with `$1 == $2, as` long as they `>=3` and if the first letter and last are the same print. The `sort | uniq -ic` counts the number of occurrences of the words and ignores case. Creating a list.

```
#removes repeated letters
awk '{l=$0; if (length($2)!=gsub(substr($2,1,1), "", $2)) print l}'
done.txt > finish.txt
```

Use of awk to remove repeated letters. It scans the length.

```
#compares text files to remove words
grep -vwF -f db.txt finish.txt > removeW.txt|
```

This command compares two files and finds matching strings. `-F` finds fixed strings, **-**v invert-match to select non-matching lines and ***-w*** is regular expression that selects lines that form whole words then removes matched. This is so when the dictionary words in file ***db.txt*** match that of the text file containing palindromes `finish.txt` it can detect words that are not real in the English dictionary and output them into remove.txt to pipeline use for the next command below.

```
grep -vwF -f removeW.txt finish.txt|
```

Same as the command above just with comparing words that are not in the English dictionary `removeW.txt` to the palindromes `finish.txt` and remove them from that text file. Using these two commands because it performs the function twice. Could have done it in one command but separating into two allowed easier to work with.

```
awk '!/ere|eke/'
```

`Awk` command is good for manipulating files. Here it is used to remove the words ‘ere’ and ‘eke’. When executing the above grep command to get rid of non-dictionary words, it did not remove these two. Unsure as to why, could be because the palindrome script is to keep words to a minimum of three characters long.

```
#rm db.txt finish.txt removeW.txt done.txt
```

`rm` used to remove these text files created during the operation so to not keep unnecessary clutter of files. Alternatively, could have created temp files - `$mktemp` or temp directory `$ mktemp -d.`

# 3 Testing

The program does run correctly most of the time as output from `stdin ./palindrome Ulysses.txt`

```
  5 bib
 29 bob
  5 civic
  2 dad
 10 deed
335 did
  4 eve
116 eye
  1 gig
  2 hah
 12 level
  2 madam
  1 minim
 13 non
  8 noon
 11 nun
  1 pap
  5 peep
  1 pip
  2 poop
  9 pop
  2 pup
 10 sees
  1 solos
  1 tot
  2 wow
```

Sometimes the `grep -vwF -f removeW.txt finish.txt` command does not work and keeps the non-dictionary words in `remove.txt` below. It takes multiple tries to get the right output.

```
  1 ama
  2 bub
  1 denned
  1 huuh
  1 kraark
  1 lil
  6 maam
  1 mem
  1 rever
  1 sas
  1 ses
```

```
1 tattarrattat
1 tut
1 txt
```

Could not get file from command line argument to work:
`cat Ulysses.txt | ./palindrome > output.txt`
As missing arguments in script. Tried implementing it with an if statement, if argument is present, but did not explore any further due to time.

Had trouble with creating files. This command below is executed at the start so does not cause any issues when is gets to using db.txt
`grep -E '.{3}' /usr/share/dict/british-english > db.txt.` As with `touch removeW.txt`. Has errors stating about it not existing so again create it at the start. Assume that it links with the issue of `grep -vwF -f removeW.txt finish.txt` not executing properly, so created at the start to make sure it does. Maybe also an issue when removing files at the end with `#rm db.txt finish.txt removeW.txt done.txt` that is why the line is commented as interferes sometimes. However, it does remove the files at the end of the script.

Awk used to remove words 'ere' and 'eke' as were not removed during `grep -vwF -f db.txt finish.txt > removeW.txt.` Do not appear when searching the dictionary file, so unsure as to why the command did not delete. Inputting other text file into command is fine - also tested using a txt file of Charles Dickens – Americans

To extract words not present in the English dictionary, decided to use the `/usr/share/dict/british-english` to output into a text instead of creating a separate text file. Both methods work the same. Decided to use this method, as I assume this path is available all in UNIX/Linux.

To run script palindrome either use source palindrome Ulysses.txt or ./palindrome Ulysses.txt.

## 4 Conclusion

In conclusion the program mostly works although there are problems with some of the functions as described above through testing. If having more time add if status to check if code run successfully and out the correct exit status. Also possibly test with more files and different commands.

The command line argument input not working which if I had more time I would have tries different ways to invoke args into the pipeline and also better formatting.