Palindrome Finder Script

Daniel Jackson - 100125289

November 8, 2017

1 Pre-Processing

The script starts with the creation of directories and temporary files that are used to process the input text later on during run time. Once all temporary files and directories have been created, filtering begins. cat along with tr and grep is used to remove punctuation and digits, convert lower case characters to upper case, and make sure that only words that are 3 characters or longer are selected from the input string. The results from this statement are then outputted to the first temporary file. sed is then used to filter down results further by using a regular expression to select any remaining words that consist of all the same character (e.g eeeee). The first part of the expression "b" represents a word boundary, and is used to make sure that only whole words are selected as opposed to possibly selecting substrings. ([a-z]) the one and only capture group of the expression gets lower case characters only, as any uppercase characters will have already been converted to lower. '1' gives a back reference to the capture group, and the + matches if there is one or more of the preceding character in the string. Preprocessing is useful as it can make sure code is modular and easy to read, and it negates the need for verbose loops for validation purposes in the main palindrome check function.

2 Main Function

Once all pre-processing has finished, a temporary file containing only whole words that are 3 or more characters long in series has been created. The next step is for a function to be used to check individual strings to see if they are in fact a palindrome. This is done by first passing the argument for the function into a local variable so that data such as the length of the string can be obtained. Variables were created for the length and midpoint of the string for readability and maintainability. This is required so that a loop can be created to loop through each character of the string for comparative purposes. The way in which my loop works is: the loop starts at 0, incrementing towards the middle value, to slightly save processing time as opposed to passing through for every character in the input string. During each pass of this loop the first and last character relative to the outside of the string inwards in the string is compared;

if inequality is found between the two characters, an exit code of 1 is returned to signal to the shell that the function was not successful.

3 Output and Post-Processing

Now that the palindrome check function is defined and input has been filtered from processing, a loop to read through the file is used. Originally, I used for loop nested within a while loop in order to delimit on each line, and then separate each line into individual strings. However, I found that this was unnecessary as the same result could be found by just using the inner for loop on the whole file. Each pass of this loop only contains an if statement; which checks whether the palindrome check function has successfully ran (exit code 0) and if so will output that word into the second temporary file. Once the loop has completed a final cat statement is used along with sort and uniq to group duplicate results, count them and make sure that they are in descending order of count. Finally the temp directory is removed to clear up the files used during run time.