Jordan Hacking | s3723766

S3723766@student.rmit.edu.au

Assessment 1 Report

Operating Systems Principles

Contents

[Task One 2](#_Toc111484386)

# Task 1

This task was a baseline for the rest of the assessment, creating the definition of what a word is allowing for the creation of a bash function and C++ function to filter an inputted text document with a single word on each line.

## Definition of a word

The definition of what the filter classifies as a word is as follows:

* A word must be between 3 and 15 (inclusive) characters long
* A word must only be lowercase characters
* A word must not have any numbers
* A word cannot contain any special characters
* A word cannot have a hyphen separating segments
* A word cannot contain more than 2 consecutive characters

Using this definition of a word we can denote the validity of the following words:

* do (length 2: invalid)
* dog (length 3: valid)
* happy (length 5: valid)
* uncharacterised (length 15: valid)
* uncharacteristically (length 20: invalid)
* happy (all lowercase: valid)
* SAD (all uppercase: invalid)
* mistake! (contains special character: invalid)
* football (all lowercase: valid)
* to-do (contains hyphen separating words: invalid)
* match5 (contains number: invalid)
* peTer (not all lowercase: invalid)
* good (2 consecutive characters: valid)
* keeper (3 consecutive characters: invalid)

## Shell function

With this definition of a word, we can devise a shell bash script (**Task1.sh including in submission**) using grep, pipe, and sort to filter an input file and output only words which are by the definition above, valid.

#!/bin/bash

**if** **[** **!** **$#** **-**eq 2 **]**

**then**

**echo** "Missing file, usage: 'filter.sh INPUT\_FILENAME OUTPUT\_FILENAME'"

**exit** 0

**fi**

**if** **[** **-e** **$1** **]**

**then**

**grep** **-**E "^[a-z]{3,15}$" **$1** **|** **grep** **-**v **-**E "([a-z])\1{2}" **|** **sort** -u **>** **$2**

**else**

**echo** "Input file does not exist, usage: 'filter.sh INPUT\_FILENAME OUTPUT\_FILENAME'"

**fi**

The first if statement in script is just checking the presence of both the input parameters, it wasn’t stated as a requirement so I will skip over it. The second if statement is checking for the presence of the input file, then proceeds to perform a few unix commands, all brought together through the pipe (‘|’) command then streamed into the output file using “>”.

1. **grep -E “^[a-z]{3,15}$”**
   1. This command is using grep to extract the lines of valid length, using -E allows grep to act like its counterpart egrep and accept extended regular expression (regex).

The expression “^[a-z]{3,15}$” is a simple regex for any word with length between 3 and 15 inclusive that is only the range of characters between **a** and **z** (only lowercase characters in the alphabet).

1. **grep -v -E “([a-z])\1{2}”**
   1. as mentioned above grep -E has been explained, however, grep -v allows us to invert the regex, meaning match everything that is **NOT** in the regex. The regex ([a-z])\1{2} means any occurrence of the same letter consecutive more than twice. The string baaa would match in this expression, but because we’re using the invert, it will **NOT** be included in the grep output.
2. **sort -u**
   1. sort -u simply sorts the file based on the first character but ignoring duplicate entries.

## C++ Function

Both C++ files are included in the submission

### Task1.cpp (entry point)

#include <iostream>

int TaskFilter**(**const std**::**string**&** input**,** const std**::**string**&** output**);**

int main**(**int argc**,** char **\*** argv**[])** **{**

**if(**argv**[**1**]** **==** **nullptr** **||** argv**[**2**]** **==** **nullptr)** **{**

std**::**cout **<<** "Invalid usage: './Task1 INPUTFILE OUTPUTFILE'" **<<** std**::**endl**;**

**return** 0**;**

**}**

std**::**cout **<<** "Using input file: " **<<** argv**[**1**]** **<<** std**::**endl**;**

//If TaskFilter returns false, there was an error filtering

**if(!**TaskFilter**(**argv**[**1**],** argv**[**2**]))** **{** **return** 0**;** **}**

std**::**cout **<<** "Filtering complete, file '" **<<** argv**[**2**]** **<<** "' created!" **<<** std**::**endl**;**

**}**

### TaskFilter.cpp (filtering function)

#include <sys/stat.h>

#include <unistd.h>

#include <fstream>

#include <iostream>

#include <algorithm>

#include <string>

#include <set>

/\*

checks a file exists and is accessible

\*/

bool check\_filename **(**const std**::**string**&** name**)** **{**

struct stat buffer**;**

**return** **(**stat **(**name**.**c\_str**(),** **&**buffer**)** **==** 0**);**

**}**

/\*

This function take a string and checks if the length if valid,

if a string is less than 3 or greater than 15 false is returned as it is not valid

if not true is returned

\*/

bool valid\_length**(**const std**::**string string**)** **{**

**if(**string**.**length**()** **<** 3 **||** string**.**length**()** **>** 15**)**

**return** **false;**

**return** **true;**

**}**

/\*

This function takes in a string and checks it only contains lowercase

characters

\*/

bool only\_lowercase\_chars**(**const std**::**string string**)** **{**

**return** string**.**find\_first\_not\_of**(**"abcdefghijklmnopqrstuvwxyz"**)** **==** std**::**string**::**npos**;**

**}**

/\*

This function takes a string and loops through each character

and checks if the next 2 letters are the same character

if true then the string is invalid as it contains more than 2 conseq

chars

\*/

bool conseq\_chars**(**std**::**string string**)** **{**

**for(**int i **=** 0**;** i **<** **(**int**)**string**.**length**()** **-** 2**;** i**++)**

**if(**string**[**i**]** **==** string**[**i**+**1**]** **&&** string**[**i**]** **==** string**[**i**+**2**])**

**return** **true;**

**return** **false;**

**}**

/\*

TaskFilter takes in an input file and output file

exported function for the file (entry point)

\*/

int TaskFilter**(**const std**::**string**&** input**,** const std**::**string**&** output**)** **{**

//Check if the input file is present and accessible, if no print usage

**if(!**check\_filename**(**input**))** **{**

std**::**cerr **<<** "File '" **<<** input **<<** "' not found, usage: './Task1 INPUTFILE OUTPUTFILE'" **<<** std**::**endl**;**

**return** 0**;**

**};**

//Declare the input and clean files

std**::**ifstream InputFile**(**input**);**

std**::**ofstream CleanFile**(**output**);**

//By using a set to contain the readLines we are ensuring they're no duplicate entries

std**::**set**<**std**::**string**>** readLines**;**

**for(**std**::**string curLine**;** std**::**getline**(**InputFile**,** curLine**);)** **{**

//If the string is not between 3 and 15 (inclusive), skip

**if(!**valid\_length**(**curLine**))** **{** **continue;** **}**

//If the string is not only lowercase chars ([A-Z][0-9] and special chars, skip)

**if(!**only\_lowercase\_chars**(**curLine**))** **{** **continue;** **}**

//If the string contains more than 2 conseq character, skip

**if(**conseq\_chars**(**curLine**))** **{** **continue;** **}**

//Pushing the string to the set handles the uniqueness of each line

readLines**.**insert**(**curLine**);**

**}**

//Push all the lines that got inserted into the set to the clean file.

**for(**std**::**string string**:** readLines**)** **{**

CleanFile **<<** string **<<** "\n"**;**

**}**

//Close the input and outfile file streams

InputFile**.**close**();**

CleanFile**.**close**();**

**return** 1**;**

**}**

## Filtered Word List

For this assessment I am using word list 3 from <https://www.keithv.com/software/wlist/> the output of both the shell and the C++ filtering function have been submitted, Task1C++Output.txt and Task1ShellOutput.txt

## Performance Metrics

### Shell

### C++