SWE2024-41: System Programming Assignment (Spring 2024)

Programming Assignment #1

Due: April 7th, 11:59 PM

1. Introduction

The purpose of this assignment is for you to gain some insight into how to use Unix File I/O and data structures.

2. Specification

The goal of this assignment is to write a program that can read a text file with the file name entered as a command line input and perform basic search functions. Once the program starts running, the program should wait for the user's keyboard input and perform one of the search functions found below based on the input.

① **Searching the occurrences of a single word**

If the program receives user input containing only a single word, search for all occurrences of the word in a given file.

- Find the word in the file and print it out on stdout in the form of

"[line number]:[start index of the word]"

② **Searching the occurrences of multiple words**

If the program receives user input containing multiple words separated with a single space and no quotation marks, search for lines containing all the words (word order in the line does not matter).

- Word order does not matter for the line (i.e. [word1] [word2] will match with "[word1] [word2]" and "[word2] [word1]".)
- Find the line(s) containing all input words in the file and print it on stdout in the form of

"[line number]"

③ **Searching the occurrences of consecutive words**

If the program receives user input containing multiple words that are wrapped in double quotation marks (“”), search the lines containing the whole phrase. It is possible to receive more than two words.

- There are no line breaks (‘\n’) inside the phrase.
- There are no spaces between “ and [word1], or [word2] and ” (i.e., “[word1] … [word2]”)
- Continuous spaces or tabs may exist inside the phrase.
- There are only 2 instances of “ in one input (i.e. There are no extra quotation marks for case 3)
- Find the line(s) containing the phrase in the file and print it on stdout in the form of

"[line number]:[start index of the phrase]"

- If a line contains multiple instances of the phrase, print all of them as shown below:

Example of Input File Line: HA HA HA HA /* first line of input file */

$ "HA HA"

1:0 1:3 1:6

④ **Searching simple regular expressing keyword locations**

If the program receives user input where an asterisk sign (*) is between two words (i.e., [word1] * [word2]), search for the lines containing at least one occurrence of at least one word being in between word1 and word2. In this case, input values are always two words.

- There are no spaces between [word1] and *, or * and [word2] (i.e., [word1]*[word2])
- Do not consider the case of [word2]*[word1]
- Find the location of keyword explained above and print it on stdout in the form of

"[line number]"

- If a line contains multiple instances of the phrase, print them only once:

Example of Input File Line: HA HA HA HA /* first line of input file */

$ HA*HA

1

Supplementary explanations

- Definitions of **line number** and **start index of the word (or phrase)**

  *** line number** – index of the line which contains the keyword. **Empty lines are counted. Starts from 1**

  *** start index of the word (or phrase)** –start index of the word (or keyword or phrase) in a line. **Whitespace and writing symbols are counted. Starts from 0**

- If the entered word occurs multiple times in the file, you must print each occurrence.
  - For ①, ③**, print all occurrences** even if they are in the same line.
  - For ②, ④, **only** need to print a line **once** even if there are duplicates.
- **If more than one output for a single search, you must add a single space after a single keyword location. You must print a new line when a search is done. For example,**

  $ "he is"

  1955:33 2315:42 5885:22

  (New Line)

- Further explanation about inputs including * or " :
  - Cases where * and " are included altogether will not be tested
  - For inputs that include ", refer to ③
  - For inputs that include *, refer to ④
  - For inputs that include neither * nor ", and include two or more words, refer to ②
  - For inputs that include neither * nor ", and include one word, refer to ①
- Search results should be output sequentially from the top of the file.
- Input text file name is given as command line argument (argv[1])
- It is assumed that the input text file is larger than the system's memory, and the length of the longest string in the input text file is assumed to be smaller than the system's memory. That is, it is not possible to place the whole input file into memory at once.
- Characters in input text files and keywords include only characters within the ASCII code range.
- There can be consecutive spaces in input text files.
- Input text files do not begin with a space.
- Keywords do not begin or end with spaces.
- **You may follow and refer to the details about keyword input at "6. Example" section.**
- The code executes infinitely waiting for input and terminates execution when the input keyword is "PA1EXIT".
- The maximum length of the input is 4,096 characters.

● Punctuation is counted as part of the word. (i.e., “word” does not match with “word!” and “word?”, but “word!” matches with “word!”)

3. Score Policy

- **It is based on a 100-point scale and must follow the format described above, otherwise no points will be awarded.**

- **Submissions after the deadline will be deducted 10 points per day. (After 10 days, 0 points for submission)**

- A certain amount of discussion is allowed, but the source code must be written by oneself.

- Scoring is carried out according to whether each test case passed or not, and the points assigned when the test case passed are included in the total score, and the points assigned when the test case was not passed are not included in the total score. There is no partial credit for each test case.

- Several input files are used for scoring, including the input file provided as an example, and the capacity of each input file can be up to several tens of GB.

- If your code takes more than 5 minutes to complete each test case, the score for that test case is not provided. All test cases are within 10 seconds of completion.

- The memory of the scoring server is assumed to be 1 GB.

4. Restriction

- This assignment is based on implementation in a **Linux environment.**

- Available header files are limited to the following files. If other headers are used, you get 0 points.

(1) At least one **hand-implemented** header file (required)

(2) unistd.h fcntl.h stdlib.h sys/types.h sys/stat.h errno.h

- The name of the binary executable file created through Makefile is set to **pa1** (lower case).

- Words: Strings separated by spaces; case insensitive.

- Examples of words that can be used in word search are as follows.

e.g.) tom, and, mckenzie, brother's, priests', wide-awake, sons', score:1

- Phrase: It is a string separated by newlines and case insensitive.

- Blank space: Tab, space, or new line.

- Words containing writing symbols are considered different words. For example, a search for **he** should include only **he** in the search results, not **her** or **he's**.

5. Submit instructions

pa1 (directory)

- ***.c (C code files)**
- ***.h (Header files)**
- **Makefile**

Compress the pa1 directory as a tarball with the format "STUDENTID_NAME_PA1.tgz" and submit it to iCampus before the deadline.

6. Example (Words highlighted in red are given as inputs)

$ ./pa1 500-Days-of-Summer_s.txt

500 days

6 6419

he is

50 1039 1822 1955 2256 2315 3494 3503 4353 4360 4445 4831 5101 5885 6325

"he is"

1955:33 2315:42 5885:22

he*is

2315 4445 5101

she he

1370 1512 1513 3423 3473 3478 3550 4255 4510 4515 5413 5672 6154 6188

loved

106:30 1122:9 1150:24 3961:25 4739:17 5921:28

…

$ ./pa1 500-Days-of-Summer_s.txt > result.out

he

hey

she

tom

summer is

"summer is"

landscape

together

we*her

no much

immediately

no*much

quarterback

we

PA1EXIT

$ diff –bsq result.out answer.out

Files result.out and answer.out are identical

$ diff result.out answer.out

$