**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Программирование»**

Тема: Регулярные выражения

|  |  |  |
| --- | --- | --- |
| Студент гр. 3344 |  | Пачев Д.К. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

**Цель работы**

Написать программу на языке Си с использованием регулярных выражений для фильтрации текста из командной строки. Развить навыки составления регулярных выражений.

**Задание**

Вариант 2.На вход программе подается текст, представляющий собой набор предложений с новой строки. Текст заканчивается предложением "**Fin.**" В тексте могут встречаться примеры запуска программ в командной строке Linux. Требуется, используя регулярные выражения, найти только примеры команд в оболочке суперпользователя и вывести на экран пары <имя пользователя> - <имя\_команды>. Если предложение содержит какой-то пример команды, то гарантируется, что после нее будет символ переноса строки.

Примеры имеют следующий вид:

* Сначала идет имя пользователя, состоящее из букв, цифр и символа **\_**
* Символ @
* Имя компьютера, состоящее из букв, цифр, символов **\_** и **-**
* Символ **:** и **~**
* Символ **$**, если команда запущена в оболочке пользователя и **#**, если в оболочке суперпользователя. При этом между двоеточием, тильдой и **$** или **#** могут быть пробелы.
* Пробел
* Сама команда и символ переноса строки.

## **Выполнение работы**

В начале в переменную text считывается текст до конечного предложения «Fin.» с помощью цикла while. Далее этот текст разделяется на отдельные части по разделителю \n, эти части хранятся с помощью массива указателей sentences\_array.

Затем компилируется регулярное выражение функцией regcomp(), с помощью цикла for программа проходится по каждому элементу массива, потом проверяется соответствие строки регулярному выражению при помощи функции regexec(), если соответствует, то выводится на экран ответ в формате  <имя пользователя> - <имя\_команды>

**Тестирование**

Результаты тестирования представлены в Таблице 1

Таблица 1 - Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | Run docker container:  kot@kot-ThinkPad:~$ docker run -d --name  stepik stepik/challenge-avr:latest  You can get into running /bin/bash  command in interactive mode:  kot@kot-ThinkPad:~$ docker  exec -it stepik "/bin/bash"  Switch user: su :  root@84628200cd19: ~ # su box  box@84628200cd19: ~ $ ^C  Exit from box: box@5718c87efaa7:  ~ $ exit  exit from container:  root@5718c87efaa7: ~ # exit  kot@kot-ThinkPad:~$ ^C  Fin. | root - su box  root - exit | Верно |
|  | root@5718c87efaa7: ~ # python3 main.py  Fin. | root - python3 main.py | Верно |

**Выводы**

В ходе лабораторной работы была написана программа на языке C с использованием регулярных выражение, которая фильтрует текст командной строки и ищет в нем команды суперпользователя.

# ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <regex.h>

int main(void) {

char \*text = malloc(sizeof(char) \* 1);

char ch;

int len = 0;

while (1) {

ch = getchar();

text = realloc(text, sizeof(char) \* len + 2);

text[len++] = ch;

text[len] = '\0';

if (strstr(text, "Fin.")) {

break;

}

}

char \*\*sentences\_array = malloc(sizeof(char \*));

int len\_sentences\_array = 0;

char \*token = strtok(text, "\n");

while (token) {

sentences\_array[len\_sentences\_array++] = strdup(token);

sentences\_array = realloc(sentences\_array, sizeof(char \*) \* (len\_sentences\_array + 1));

token = strtok(NULL, "\n");

}

for (int i = 0; i < len\_sentences\_array; i++) {

regex\_t regex;

int max\_groups = 7;

regmatch\_t group\_array[max\_groups];

int value;

if (regcomp(&regex, "([A-Za-z0-9\_]+)@([A-Za-z0-9\_-]+)(: ?)(~ ?)(\\#) ?(.+)", REG\_EXTENDED)) {

printf("can't compile regular expression\n");

return 0;

};

value = regexec(&regex, sentences\_array[i], max\_groups, group\_array, 0);

if (value == 0) {

int is\_sep = 0;

for (int k = 1; k < max\_groups; k++) {

if (group\_array[k].rm\_so == -1) {

break;

}

for (int j = group\_array[k].rm\_so; j < group\_array[k].rm\_eo; j++) {

if (k == 1) {

printf("%c", sentences\_array[i][j]);

} else if (k == 6) {

if (!is\_sep) {

printf(" - ");

is\_sep = 1;

}

printf("%c", sentences\_array[i][j]);

}

}

}

printf("\n");

}

regfree(&regex);

}

return 0;

}