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
1.	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
2.	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;

}
```