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 _____ Клюкин А.В.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Получить практические навыки в работе с регулярными выражениями и научиться их применять.

Задание.

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 слеующий 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

Изначально происходит считывание текста до конечной фразы и его запись. После объявляется регулярное выражение, как строка. Важно заметить, что в си так же необходимо экранировать “\” в регулярном выражении, поэтому иногда они идут подряд. После в тексте ищется последовательность, удовлетворяющее условию и если оно находится, то с помощью .rm_so берется индекс начала нужной подстроки и до ее конца выводится с нужным форматированием. После часть текста копируется на место выведенной части, тем самым убирая её и позволяя продолжить поиск совпадений. Далее высвобождается динамически выделенная память.

Тестирование.

Результаты тестирования представлены в табл. 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

Выводы

Были применены навыки работы с регулярными выражениями. Этот инструмент оказался довольно универсальным для поиска сложных выражений.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Klyukin_Aleksandr_lb1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <regex.h>

int main() {
    char *text = NULL;
    char buffer[100];
    size_t text_length = 0;
    while (1) {
        fgets(buffer, 100, stdin);
        if (strstr(buffer, "Fin.") != NULL) {
            break;
        }
        text = realloc(text, text_length + strlen(buffer) + 1);
        strcpy(text + text_length, buffer);
        text_length += strlen(buffer);
    }
    const char *pattern = "[A-Za-z_0-9]+@[A-Za-z0-9_-]+: ?\\~ ?\\# ?.";
    regex_t regex;
    int reti;
    regmatch_t match;
    reti = regcomp(&regex, pattern, REG_EXTENDED);
    reti = regexec(&regex, text, 1, &match, 0);
    while (reti == 0) {
        for (int i = match.rm_so; i < match.rm_eo; i++) {
            if(text[i] == '@'){
                printf("%c", ' ');
                while (text[i] != '#')
                {
                    i = i + 1;
                }
                printf("-");
                i= i + 1;
                while (text[i] != '\n')
                {
                    printf("%c", text[i]);
                    i = i + 1;
                }
            }
            printf("%c", text[i]);
            text[i] = ' ';
        }
          memmove(text + match.rm_so, text + match.rm_eo, strlen(text) -
match.rm_eo + 1);
        reti = regexec(&regex, text, 1, &match, 0);
    }
    regfree(&regex);
    free(text);
    return 0;
}
```