**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 |  | Вердин К.К. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Освоение работы с регулярными выражениями на языке Си на примере использующей их программы.

## Задание.

Вариант 2. На вход программе подаётся 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

Была подключена стандартная библиотека для ввода и вывода *<stdio.h>*, библиотека для работы со строками *<string.h>* и библиотека для работы с регулярными выражениями *<regex.h>*. Была проинициализирована строка, отвечающая за регулярное выражение *char \*regex\_string = "(\\w+)@[0-9a-zA-Z\_-]+: ?~ ?# (.+\n)"* Была объявлена структура для хранения информации о скомпилированном регулярном выражении *regex\_t regex\_сompiled*. Был объявляен массив *regmatch\_t group\_array[]* размером 3, который будет использоваться для хранения информации о совпадениях групп в регулярном выражении. Была вызвана функция для компиляции регулярного выражения *regcomp(&regex\_compiled, regex\_string, REG\_EXTENDED))*. Был инициализирован массив символов char s[100]. Был запущен цикл, который считывал входные данные, пока строка на вход не равнялась “Fin.” *while (strcmp(s, "Fin."))*. В *if* была вызвана функция *if (regexec(&regex\_compiled, s, 3, group\_array, 0) == 0)*, которая проверяет, соответствует ли строка s регулярному выражению. Если соответствие найдено, функция возвращает 0, иначе - ненулевое значение. Далее была реализована функция *void print\_groups\_of\_string(regmatch\_t \*group, char \*s)* которая выводит символы из первой и второй группы регулярного выражения. После выводавысвобождалась память под структуру для скомпилированного регулярного выражения. После чего программа завершалась.

Разработанный программный код см. в приложении А.

## Тестирование.

Результаты тестирования представлены в табл. 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 | - |
|  | Switch user: su :  roofsafast@8dasd4628200cd19 : ~ # d dadad ad a  box@84628200cd19: ~ $ ^C  Exit from box: box@5718c87efaa7:  ~ $ exit  exit from container:  root@5718c87efaa7: ~ # exit  kot@kot-ThinkPad:~$ ^C  Fin.  root@da\_\_da:~# su bsax  Fin. | roofsafast - d dadad ad a  root - exit  root - su bsax | - |

## Выводы

Была освоена работа с регулярными выражениями на языке Си на примере использующей их программы.

# Приложение А Исходный код программы

Название файла: Verdin\_Kirill\_lb1.c

#include <stdio.h>

#include <regex.h>

#include <string.h>

void print\_groups\_of\_string(regmatch\_t \*group, char \*s)

{

    for (size\_t j = group[1].rm\_so; j < group[1].rm\_eo; j++)

    {

        printf("%c", s[j]);

    }

    printf(" - ");

    for (size\_t j = group[2].rm\_so; j < group[2].rm\_eo; j++)

    {

        printf("%c", s[j]);

    }

}

int main()

{

    char \*regex\_string = "(\\w+)@[0-9a-zA-Z\_-]+: ?~ ?# (.+\n)";

    regmatch\_t group\_array[3];

    regex\_t regex\_compiled;

    regcomp(&regex\_compiled, regex\_string, REG\_EXTENDED);

    char s[100] = "";

    while (strcmp(s, "Fin."))

    {

        fgets(s, 100, stdin);

        if (regexec(&regex\_compiled, s, 3, group\_array, 0) == 0)

        {

            print\_groups\_of\_string(group\_array, s);

        }

    }

    regfree(&regex\_compiled);

    return 0;

}