**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 |  | Ханнанов А.Ф. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Получить навыки в составлении регулярных выражений. Научиться применять их в работе на языке Си.

## 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 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

Программа начинается с того, что задаёт регулярное выражение в массив regexString, после компиляции выражения оно будет находиться в regexCompiled. Далее идёт проверка на успешную компиляцию. После успешной компиляции выражения задаётся динамический массив text\_line, в котором по очереди будут храниться строки входных данных. Следующим шагом идёт цикл, который будет выполняться, пока не будет встречена строка “Fin.”. В цикле идёт поочерёдный ввод строк. Эти строки проверяются на соответствие регулярному выражению. Если строка ему соответствует, то запускаются циклы, которые выводят первую и четвёртую группы выражения (имя пользователя и команда соответственно). После вывода этих данных массив обнуляется. После выполнения цикла высвобождается память массива и регулярного выражения.

## 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 | - |

## Выводы

Были изучены правила составления регулярных выражений. Получены навыки написания регулярных выражений и их применения на практике, используя язык Си.

# Приложение А Исходный код программы

Название файла: Khannanov\_Artem\_lb1.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <regex.h>

#define MAX\_GROUPS 5

int main() {

char \*regexString = "(\\w+)@([A-Za-z0-9\_-]+)(: ?\\~ ?\\# )(.+\n)";

regex\_t regexCompiled;

regmatch\_t groupArray[MAX\_GROUPS];

if (regcomp(&regexCompiled, regexString, REG\_EXTENDED)) {

printf("[Can't compile expression]\n");

return 0;

}

char \*text\_line = (char \*)malloc(sizeof(char) \* 1000);

while (fgets(text\_line, 1000, stdin) != NULL) {

if (strncmp(text\_line, "Fin.\n", 5) == 0) break;

if (regexec(&regexCompiled, text\_line, 5, groupArray, 0) == 0) {

for (size\_t i = groupArray[1].rm\_so; i < groupArray[1].rm\_eo; i++) {

printf("%c", text\_line[i]);

}

printf(" - ");

for (size\_t j = groupArray[4].rm\_so; j < groupArray[4].rm\_eo; j++) {

printf("%c", text\_line[j]);

}

}

free(text\_line);

text\_line = (char \*)malloc(sizeof(char) \* 1000);

}

free(text\_line);

regfree(&regexCompiled);

return 0;

}