**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 |  | Волохов М. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

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

## 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 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

Программа задаёт регулярное выражение в массив pattern, после компиляции выражения оно будет записано в regexCompiled. Далее идёт проверка на верную компиляцию. После компиляции выражения задаётся массив txtString, в котором будут храниться строки входных данных. Далее идёт цикл while, который будет выполняться, до того, как будет встречена строка “Fin.”. В цикле происходит поочерёдный ввод строк. Строки проверяются на соответствие регулярному выражению. Если строка соответствует, то запускаются циклы, которые выводят первую и четвёртую группы выражения (имя пользователя и команда). После вывода данных массив обнуляется. После выполнения цикла освобождается память массива и регулярного выражения.

Программный код см. в приложении А

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| **№п/п** | **Входные данные** | **Выходные данные** | **Комментарий** |
| **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 | - |

## 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: Main.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <regex.h>

int main() {

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

regex\_t regexCompiled;

regmatch\_t groups[4];

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

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

return 0;

}

char \*txtString = (char \*)malloc(sizeof(char) \* 1000);

// Reading until "Fin.\n"

while (fgets(txtString, 1000, stdin) != NULL) {

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

// Check for regex match

if (regexec(&regexCompiled, txtString, 5, groups, 0) == 0) {

// Print

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

printf("%c", txtString[i]);

}

printf(" - ");

// Print

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

printf("%c", txtString[j]);

}

}

}

// Free allocated memory

free(txtString);

regfree(&regexCompiled);

return 0;

}