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. 3342 _____ Гончаров С.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Изучить принцип работы регулярных выражений и использовать их в программе на языке C.

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

Код использует библиотеку regex.h для работы с регулярными выражениями. Создается переменная regex, которая компилируется в регулярное выражение regcomp(&regex, regular_expression, REG_EXTENDED). Создается переменная sentence[100], в переменную записывается предложение из stdin с помощью функции fgets. Цикл while проверяет является ли текущее предложение завершающим (**Fin.**). Внутри цикла реализован вывод всех подходящих нам строк. Обращаясь к groups[1] - отвечает за первую группу регулярного выражения и отображает имя пользователя. Посимвольно, с помощью цикла, выводим имя пользователя от начала groups[1].rm_so до конца подстроки groups[1].rm_eo. Groups[3] хранит в себе команды который ввел пользователь. Поле groups[2] хранит имя компьютера пользователя, но для решения задачи оно не требуется. В конце освобождаем память, в которой хранилось регулярное выражение regfree(&regex). Программа выводит подходящие строки в формате <имя пользователя> - <имя_команды>.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
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

Был разработан код на языке программирования C, написано регулярное выражение и изучены способы работы с ними.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <regex.h>
#include <string.h>
#include <stdlib.h>
#define BREAK_SENTENCE "Fin."

int main() {
    regex_t regex;
    regmatch_t groups[4];
    char*    regular_expression    =    "([A-Za-z0-9_]+)@([A-Za-z0-
9_-]+):\\s?~\\s?# (.+)";
    regcomp(&regex, regular_expression, REG_EXTENDED);

    char sentence[100];

    while (strcmp(sentence, BREAK_SENTENCE)) {

        fgets(sentence, 100, stdin);

        if (regexec(&regex, sentence, 4, groups, 0) == 0) {

            for (int i = groups[1].rm_so; i < groups[1].rm_eo; i++)
{
                printf("%c", sentence[i]);
            }
            printf(" - ");

            for (int i = groups[3].rm_so; i < groups[3].rm_eo; i++)
{
                printf("%c", sentence[i]);
            }

        }
    }
    regfree(&regex);
    return 0;
}
```