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(ка) гр. 3343 _____ Гельман П.Е.

Преподаватель _____ Государкин Я.С.

Санкт-Петербург
2024

Цель работы

Цель лабораторной работы заключается в изучении регулярных выражений и их применении в программах 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

Для работы с регулярными выражениями используется библиотека regex.h

Сначала зададим четыре переменных: size_t groups, максимальное количество групп в регулярном выражении, regex_t regex, переменная для хранения скомпилированного регулярного выражения, regmatch_t matches[groups], массив, в котором хранятся индексы начала и конца групп, char* pattern, регулярное выражение.

Далее компиляция регулярного выражения осуществляется благодаря функции int regcomp(regex_t *preg, const char *pattern, int cflags). Если его нельзя скомпилировать, программа завершается ошибкой.

Затем происходит считывание текста из буфера построчно до строки «Fin.».

Проверка строки на соответствие заданному регулярному выражению происходит благодаря функции int regexec(const regex_t *preg, const char *String, size_t nmatch, regmatch_t *pmatch, int eflags). Если проверка прошла успешно, на экран выводится первая и вторая группы — имя пользователя и команда. Далее очищается память, выделенная под работу функции regcomp().

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.	Welcome to virtual machine management system: me@my-laptop:~# vm start ubuntu-vm Starting virtual machine ubuntu-vm... Virtual machine ubuntu-vm is now running. You can connect to it using SSH: me@my-laptop:~$ ssh ubuntu@192.168.1.100 Password: Welcome to Ubuntu 20.04 LTS! ubuntu@ubuntu-vm:~# ^C Exiting SSH session. Shutting down virtual machine: ubuntu@ubuntu-vm:~$ exit Virtual machine ubuntu-vm is shutting down... me@my-laptop:~$ ^C Fin.	me - vm start ubuntu-vm ubuntu - ^C
2.	kot@kot-ThinkPad:~$ docker run -d --name my_container busybox kot@kot-ThinkPad:~$ docker exec -it my_container /bin/sh root@container_id:~ # ls root@container_id:~ # touch new_file.txt root@container_id:~ # exit kot@kot-ThinkPad:~$ docker stop my_container kot@kot-ThinkPad:~$ docker rm my_container Fin.	root — ls root - touch new_file.txt root - exit

Выводы

Были изучены регулярные выражения и работа с ними на языке Си. Реализована программа, которая находит все строки в тексте, где команды вызваны от имени суперпользователя, и выводит их на экран. Для этого использовалась библиотека regex.h и ее функции.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

int main() {
    char s[101];
    size_t groups = 3;
    regex_t regex;
    regmatch_t matches[groups];

      char* pattern = "([a-zA-Z0-9_]+)@[a-zA-Z0-9_-]+: *~ *#
(.*)";
    if (regcomp(&regex, pattern, REG_EXTENDED)) {
        return 1;
    }

    while (fgets(s, 100, stdin)) {
        if (strstr(s, "Fin.\n") != NULL) {
            break;
        }

        if (regexec(&regex, s, groups, matches, 0) == 0) {

             if (matches[1].rm_so != -1 && matches[2].rm_so !=
-1) {
                 s[matches[1].rm_eo] = '\0';
                 s[matches[2].rm_eo] = '\0';
                      printf("%s - %s", &s[matches[1].rm_so],
&s[matches[2].rm_so]);
                 }
        }
    }

    regfree(&regex);

    return 0;
}
```