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. 3383 _____ Боривец С. Ю.

Преподаватель _____ Куликов И. А.

Санкт-Петербург

2023

Цель работы

Изучить принцип работы регулярных строк, освоить библиотеку regex.h, связанную с этой темой, а также использовать полученные знания, чтобы написать программу на языке программирования C, которая будет решать поставленную задачу.

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

Библиотеки:

stdio.h — требуется для ввода и вывода(getchar, printf).

stdlib.h — требуется для работы с динамической памятью(malloc, realloc, free).

string.h — требуется для команды сравнения строк(strcmp).

regex.h — требуется для работы с регулярными выражениями(regcomp, regexec, regfree).

Макросы:

ADD_SIZE — изначальное количество ячеек памяти, а также добавляемое количество ячеек при недостатке места для символов/предложений.

Переменные:

size_t com_mas_memory — количество ячеек динамической памяти для хранения предложений текста.

char **com_mas — указатель на массив указателей на предложения, которые вводит пользователь.

size_t count_com — счётчик количества предложений.

char ch — в этой переменной хранится символ, который вводит пользователь.

size_t command_memory — количество ячеек динамической памяти для хранения символов предложения.

char *command — указатель на предложение, вводимое пользователем, где может быть команда.

size_t count_ch — счётчик количества символов в предложении.

regex_t buf — переменная, где хранится скомпилированное регулярное выражение.

char *pattern — переменная, где хранится указатель на регулярное выражение в виде строки.

int rc — переменная, где хранится результат команд библиотеки regex.h(0, если успешно выполнена компиляция regcomp или строка подошла к шаблону регулярного выражения regexec).

size_t nmatch — количество групп захвата.
regmatch_t match — массив переменных типа regmatch_t, где хранится информация о начале и конце групп захвата.

Функция main():

Создается переменная com_mas_memory, в которой будет храниться размер массива с предложениями. Выделяется динамическая память с помощью команды malloc, сохраняется в переменной com_mas, создается переменная счётчик count_com. Объявляется переменная ch.

Далее идет цикл do-while, который отвечает за ввод и сохранение предложений. Для каждого предложение выделяется память, объявляется переменная для наблюдения за памятью, счетчик символов в предложении. Если количество символов становится больше ранее выделенной памяти, то происходит перевыделение памяти с помощью функции realloc, память становится больше на величину ADD_SIZE. После записи строки происходит сравнение со строкой «Fin.», которая считается концом текста по условию. Если появляется предложение «Fin.», то цикл завершает работу.

Затем идут манипуляции с регулярными выражениями. Объявляются переменные buf, pattern(регулярное выражение в виде строки), rc, nmatch, match, описанные ранее. Сначала проверяется, компилируется ли строка. Если строка не компилируется, то выводится ошибка. В ином случае происходит поиск предложений, подходящих под регулярное выражение с помощью команды regexec. Если строка подходит, то благодаря переменной match находится начало и конец нужных групп захвата(всего их 3), выводится имя пользователя и команда.

Вся выделенная динамическая память освобождается с помощью команд free и regfree.

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
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	OK

Выводы

Был изучен принцип работы регулярных строк, освоена библиотеку regex.h, связанная с этой темой, а также были использованы полученные знания, чтобы написать программу на языке программирования C, которая решает поставленную задачу.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define ADD_SIZE 20

int main(){
    size_t com_mas_memory = ADD_SIZE;
         char  **com_mas  =  (char  **)malloc(sizeof(char  *)  *
com_mas_memory);
    size_t count_com = 0;
    char ch;
    do {
        size_t command_memory = ADD_SIZE;
               char  *command  =  (char  *)malloc(sizeof(char)  *
command_memory);
        size_t count_ch = 0;
        while((ch = getchar()) != '\n'){
            if (count_ch + 3 > command_memory) {
                command_memory += ADD_SIZE;
                command = (char *)realloc(command, sizeof(char) *
command_memory);
            }
            command[count_ch] = ch;
            count_ch++;
            command[count_ch] = '\0';
            if (strcmp(command, "Fin.\0") == 0) break;
        }
        if (count_com + 2 > com_mas_memory) {
            com_mas_memory += ADD_SIZE;
             com_mas = (char **)realloc(com_mas, sizeof(char *) *
com_mas_memory);
        }
        com_mas[count_com] = command;
        count_com++;
    } while (strcmp(com_mas[count_com - 1], "Fin.\0") != 0);
    regex_t buf;
    char *pattern = "(\\w+)@[a-zA-Z0-9_-]+: ?~ ?# (.*)";
    int rc;
    size_t nmatch = 3;
    regmatch_t match[3];
    if ((rc = regcomp(&buf, pattern, REG_EXTENDED)) == 0) {
        for(int i = 0; i < count_com; i++) {
              if ((rc = regexec(&buf, com_mas[i], nmatch, match,
0)) == 0) {
                 for (int j = match[1].rm_so; j < match[1].rm_eo;
j++) {
                    printf("%c", com_mas[i][j]);
                }
                printf(" - ");
```

```
                        for (int j = match[2].rm_so; j < match[2].rm_eo;
j++) {
                            printf("%c", com_mas[i][j]);
                        }
                        printf("\n");
                    }
                }
            } else {
                printf("Ошибка: Регулярное выражение некорректно");
            }
            regfree(&buf);
            for(int i = 0; i < count_com; i++) {
                free(com_mas[i]);
            }
            free(com_mas);
            return 0;
        }
```