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 _____ Корниенко А.Е.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Целью данной работы является ознакомление с регулярными выражениями и их реализацией 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

В начале компилируется регулярное выражение в соответствии с условием работы. В случае ошибки программа завершается и выводит сообщение “can’t complete”.

Далее происходит построчный ввод текста пользователем. Программа завершается, когда будет введена строка “Fin.” . Введённые строки обрабатываются при помощи цикла while.

При помощи переменных char** users_names и char** users_commands осуществляется динамическое хранение подходящих под регулярное выражение строк. При помощи условного оператора if осуществляется проверка на правильность выделения памяти. Используя функцию regexec проверяется соответствие строки регулярному выражению.

После того, как пользователь ввёл текст, при помощи цикла for выводится результат программы – подходящие предложения. Далее очищается динамическая память, используя free().

После завершения вывода информации память, выделенная для хранения регулярного выражения, освобождается функцией 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
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

Было проведена работа по ознакомлению с регулярными выражениями и с функциями библиотеки regex.h языка Си.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

int main(){
    char*      regexString      =      "([a-zA-Z0-9_]+)@([a-zA-Z0-9_-
]+)\\:\\s*\\~\\s*(\\$|\\#)\\s(.+)";
    size_t maxGroups = 5;


    regex_t regexCompiled;
    regmatch_t groupArray[maxGroups];


    if(regcomp(&regexCompiled, regexString, REG_EXTENDED)){
        printf("can't compile");
        return 0;

    }

    char s[101];
    int capacity = 25; //start count string
    char** users_names = malloc(capacity * sizeof(char*));
    char** users_commands = malloc(capacity * sizeof(char*));


    int count = 0;
    int n = 0;
    while(1){
        fgets(s,100,stdin);
        if(strstr(s, "Fin.") != NULL)
            break;

        if(regexec(&regexCompiled,  s,maxGroups,  groupArray,0)  ==
0){
            if(s[groupArray[3].rm_so] == '#'){
                count ++;
                if (count >= capacity){
                    capacity*= 1.5;
                    if(users_names  !=  NULL  &&  users_commands  !=
NULL){
                        users_names = realloc(users_names, capacity
* sizeof(char*));
                        users_commands   =   realloc(users_commands,
capacity * sizeof(char*));
                    }
                }
```

```
                    int   size_c_names   =   (int)groupArray[1].rm_eo   -
(int)groupArray[1].rm_so + 1;
                    if(users_names != NULL){
                        users_names[count - 1] = malloc(size_c_names *
sizeof(char));
                    }
                    if(users_names[count - 1] != NULL){
                        for(int                    j                    =
groupArray[1].rm_so;j<groupArray[1].rm_eo;j++){
                            users_names[count - 1][n++] = s[j];
                        }
                        users_names[count - 1][size_c_names] = '\0';
                    }



                    int  size_c_command  =  (int)groupArray[4].rm_eo  -
(int)groupArray[4].rm_so + 1;

                    if(users_commands != NULL){
                        users_commands[count        -        1]        =
malloc( size_c_command * sizeof(char));
                    }

                    if(users_commands[count - 1] != NULL){
                        n = 0;
                        for(int    k    =    groupArray[4].rm_so;k    <
groupArray[4].rm_eo - 1;k++){
                            users_commands[count - 1][n ++] = s[k];
                        }
                        n = 0;
                        users_commands[count   -   1][size_c_command]   =
'\0';
                    }
                }
            }



        }


        for(int i = 0;i < count; i ++){
            printf("%s - %s\n",users_names[i],users_commands[i]);
        }
        if(users_names != NULL && users_commands != NULL){
            for(int i = 0;i < count; i ++){
                free(users_names[i]);
                free(users_commands[i]);
            }

            free(users_names);
            free(users_commands);
        }

        regfree(&regexCompiled);

        return 0;
```

```
}
```