**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**

Студентка гр. 3341 _____ Яковлева А.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

**Цель работы**

Целью работы является освоение работы с регулярными выражениями на языке C.

Для достижения поставленной цели требуется решить следующие задачи:

- ознакомиться с регулярными выражениями;
- научиться их использовать;
- написать программу, решающую задачу в соответствии с индивидуальным условием с использованием регулярных выражений.

**Задание**

2 вариант.

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**

Используемые переменные:

- макрос *MAX_STRING_LENGTH* - максимальная длина строки
- макрос *LAST_STRING* - последнее предложение текста
- макрос *END_STRING* - символ конца строки '\0'
- *current_string* указатель на текущую строку
- *regex_string* строка с регулярным выражением
- *regex_compiled* хранит скомпилированное регулярное выражение
- *group_array* массив групп захвата размера *regex_compiled.re_nsub + 1* (1 добавляется, т.к. вся подходящая строка является группой) типа *regmatch_t* - структуры, состоящей из *regoff_t rm_so,* который сохраняет начальную позицию соответствующей текстовой строки в целевой строке и *regoff_t rm_eo,* который сохраняет конечную позицию соответствующей текстовой строки в целевой строке.

Регулярное выражение "(\\w+)@(\\w|-)+: *~ *# ([^\n]+)\n$" начинается с имени пользователя, состоящего из букв, цифр и символа _ , содержит не менее 1 символа, т.е. \w+, объединим имя пользователя в группу, т.к его нужно вывести, получим (\w+). Далее идёт символ @. Потом имя компьютера, состоящее из букв, цифр, символов _ и -, т.е (\w|-), содержит не менее 1 символа, т.е. (\w|-)+. Нужно найти команды суперпользователя, поэтому далее идут символы :~#, между которыми могут быть пробелы, т.е. : *~ *#. Далее идёт пробел. Затем сама команда и символ перевода строки, т.е все символы за исключение символа '\n', встречаются не менее 1 раза, получим [^\n]+, объединим команду в группу, т.к. её нужно вывести, т.е. ([^\n]+), далее \n$, т.к. '\n' - последний символ строки.

Функции:

- *read_string* принимает указатель на текущую строку, посимвольно считывает строку до символа '\n' и добавляет к строке *END_STRING* ('\0').
- *output_matched_string* принимает указатель на текущую строку и на массив групп захвата. Имя пользователя соответствует подстроке с индекса *group_array[1].rm_so* по *group_array[1].rm_eo* – 1. Символ текущей строки,

расположенный по индексу *group_array[1].rm_eo* заменяется *END_STRING* ('\0'), тогда вывод строки *current_string* + *group_array[1].rm_so* соответсвует имени пользователя. Команда, введёная пользователем, соответствует подстроке с индекса *group_array[3].rm_so* по *group_array[3].rm_eo* – 1. Символ текущей строки, расположенный по индексу *group_array[3].rm_eo* заменяется *END_STRING* ('\0'), тогда вывод строки *current_string* + *group_array[3].rm_so* соответсвует команде, введённой пользователем.

- *main* создаёт *current_string* размера *MAX_STRING_LENGTH*sizeof(char)*, *regex_string, regex_compiled,* с помощью функции *regcomp* компилирует указанный шаблон регулярного выражения. Если выражение скомпилировалось успешно, то создаёт *group_array;* с помощью функции *strncmp* проверяет совпадает ли текущая строка с *LAST_STRING* и, пока не совпадает, вызывает функцию *read_string,* функцией *regexec* проверяет соотвествие строки регулярному выражению, при соответствии вызывает функцию *output_matched_string,* затем освобождает память, выделенную под текущее предложение.

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:<br>kot@kot-ThinkPad:~$ docker run -d --name stepik stepik/challenge-avr:latest<br>You can get into running /bin/bash<br>command in interactive mode:<br>kot@kot-ThinkPad:~$ docker exec -it stepik "/bin/bash"<br>Switch user: su :<br>root@84628200cd19: ~ # su box<br>box@84628200cd19: ~ $ ^C<br>Exit from box:<br>box@5718c87efaa7:<br>~ $ exit<br>exit from container:<br>root@5718c87efaa7: ~ # exit<br>kot@kot-ThinkPad:~$ ^C<br>Fin. | root - su box<br>root - exit | root@84628200cd19: ~ # su box и root@5718c87efaa7: ~ # exit - команды в оболочке суперпользователя,<br>нужно вывести пары <имя пользователя> - <имя команды>, т.е в первой строке root - su box, во второй root – exit. |
| 2. | t&@8619: ~ # 7!<br>rt@7_19:~ #3yt6!<br>_ot@81-9: ~# 7!<br>root@819: ~$ 3<br>Fin. | _ot - 7! | первое не подходит, т.к. имя пользователя не может содержать &, второе не подходит, т.к. отсутствует пробел после #, третье подходит, т.к. |

| | | | |
|---|---|---|---|
| | | | имя пользователя может содержать _ и между :, ~, # могут и не быть пробелы, четвёртое не подходит, т.к. $ означает, что команда запущена не в оболочке суперпользователя. |

**Выводы**

Была освоена работа с регулярными выражениями на языке C.

Для достижения поставленной цели были решены следующие задачи:

- ознакомление с регулярными выражениями;
- их использование;
- написана программа, которая, используя регулярные выражения, находит только примеры команд в оболочке суперпользователя и выводит на экран пары <имя пользователя> - <имя команды>.

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define MAX_STRING_LENGTH 100
#define LAST_STRING "Fin."
#define END_STRING '\0'

void read_string(char* current_string)
{
    int i = -1;
    do
    {
        scanf ("%c", &current_string[++i]);
    }while (current_string[i] != '\n');
    current_string[++i] = END_STRING;
}

void output_matched_string(char* current_string, regmatch_t* group_array)
{
    current_string[group_array[1].rm_eo] = END_STRING;
    current_string[group_array[3].rm_eo] = END_STRING;
    printf("%s   -   %s\n",   current_string   +   group_array[1].rm_so,
current_string + group_array[3].rm_so);
}

int main ()
{
    char* current_string = malloc(MAX_STRING_LENGTH*sizeof(char));
    char* regex_string = "(\\w+)@(\\w|-)+: *~ *# ([^\n]+)\n$";
    regex_t regex_compiled;


    if (regcomp(&regex_compiled, regex_string, REG_EXTENDED) == 0)
    {
        regmatch_t group_array[regex_compiled.re_nsub + 1];
        while (strncmp(current_string, LAST_STRING, strlen(LAST_STRING)))
        {
            read_string(current_string);
          if          (regexec(&regex_compiled,          current_string,
regex_compiled.re_nsub + 1, group_array, 0) == 0)
              output_matched_string(current_string, group_array);
        }
    }
    free(current_string);

    return 0;
}
```