**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,* которыйсохраняет начальную позицию соответствующей текстовой строки в целевой строке и *regoff\_t rm\_eo,* которыйсохраняет 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 |
|  | 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 | root@84628200cd19: ~ # su box и root@5718c87efaa7: ~ # exit - команды в оболочке суперпользователя, нужно вывести пары <имя пользователя> - <имя команды>, т.е в первой строке root - su box, во второй root – exit. |
|  | t&@8619: ~ # 7!  rt@7\_19:~ #3yt6!  \_ot@81-9: ~# 7!  root@819: ~$ 3  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;

}