**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. 3341 |  | Романов А.К. |
| 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:

* *char \*text –* хранит введенный пользователем текст.
* *char \*\*sentences —* массив, в котором хранятся строки текста
* *int number\_of\_sentences —* содержит число строк текста

Функции:

* *InputText* осуществляет посимвольное считывание текста из консоли. Считывание прерывается, в случае если введено «Fin.»
* *Split* разделяет введенный текст на отдельные строки.
* *Result* компилирует регулярное выражение, необходимое для поиска соответствующи строк. После чего в случае успешной комплияции выражения, на соответствие провреряются поочередно все строки введенного текста, хранящиеся в *sentences*. При этом, если строка соответствует регулярному выражению, то в группы захвата попадают искомые имя пользователя и команда, которые выводятся программой в консоль через тире.
* *main* инициализирует *number\_of\_sentences,* а затем вызывает функции InputText и Split для создания text и sentences соответственно. После вызывается функция Result, а затем очищается занятая динамическая память.

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 | e.moevm confirmed |
|  | name%@5479: ~ # act!  user@1\_99:~ #something!  ro\_ot@13-3: ~# act!  [user@690](mailto:user@69420): ~ # command  root@819: ~$ perform  Fin. | ro\_ot - act!  user - command | 1. Не подходит, т.к. в имени пользователя содержатся некорректные символы (%).  2. Не подходит, поскольку после # не стоит пробел.  3. Подходит, все корректно. (Проверили валидность имени, содержащего \_)  4. Подходит. Между двоеточием, тильдой и решеткой могло быть любое количество пробелов.  5. Не подходит, команда запущена не в оболочке супер-пользователя. |

## 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 <string.h>

#include <stdlib.h>

#include <regex.h>

char\* InputText(int\* number\_of\_sentences);

char\*\* Split(char\* text, int\* number\_of\_sentences);

void Result(char\*\* sentences, int\* number\_of\_sentences);

void ResultSite(char\*\* sentences, int\* number\_of\_sentences);

int main(){

int number\_of\_sentences = 0;

char\* text = InputText(&number\_of\_sentences);

char\*\* sentences = Split(text, &number\_of\_sentences);

Result(sentences, &number\_of\_sentences);

free(text);

for (int i = 0; i < number\_of\_sentences; i++){

free(sentences[i]);

}

free(sentences);

}

char\* InputText(int\* number\_of\_sentences){

char\* text = (char\*)malloc(sizeof(char));

char end\_of\_text[5] = "Fill";

char symbol = getchar();

int position = 0;

int capacity = 1;

while (1==1){

text[position] = symbol;

if(position+1 >= capacity){

capacity \*= 2;

text = (char\*)realloc(text, capacity \* sizeof(char));

}

if(symbol == '\n'){

(\*number\_of\_sentences)++;

}

position++;

for(int i = 0; i < 3; i++){

end\_of\_text[i] = end\_of\_text[i+1];

}

end\_of\_text[3] = symbol;

if(strncmp(end\_of\_text, "Fin.", 4) == 0){

text[position] = '\0';

break;

}

symbol = getchar();

}

return text;

}

char\*\* Split(char\* text, int\* number\_of\_sentences){

char\*\* sentences = (char\*\*)malloc((\*number\_of\_sentences)\*sizeof(char\*));

int position = 0;

int start = 0;

int end = 0;

for(int i = 0; i < strlen(text); i++){

if(text[i]=='\n'){

end = i;

int pos = 0;

sentences[position] = (char\*)malloc(sizeof(char)\*(end-start+1));

for(int j = start; j <= end; j++){

sentences[position][pos] = text[j];

pos++;

}

sentences[position][pos] = '\0';

position++;

if (end + 1 < strlen(text)){

start = end + 1;

}

else{

break;

}

}

}

return sentences;

}

void Result(char\*\* sentences, int\* number\_of\_sentences){

int flag = 0;

char mask[] = "([A-Za-z0-9\_]+)@[A-Za-z0-9\_-]+: \*?~ \*?# (.\*\n)";

size\_t groups = 3;

regex\_t mask\_compiled;

regmatch\_t arr[groups];

if (regcomp(&mask\_compiled, mask, REG\_EXTENDED)){

printf("Error: could not compile regular expression\n");

}

for(int i = 0; i < (\*number\_of\_sentences); i++){

if(regexec(&mask\_compiled, sentences[i], groups, arr, 0) == 0){

if(flag != 0){

printf("\n");

}

else{

flag = 1;

}

for(int x = arr[1].rm\_so; x < arr[1].rm\_eo; x++){

printf("%c", sentences[i][x]);

}

printf(" - ");

for(int x = arr[2].rm\_so; x < arr[2].rm\_eo-1; x++){

printf("%c", sentences[i][x]);

}

}

}

}