**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине

«Операционные системы»

«Реализация интерпретатора команд ОС»

Студент

Группа АИ-20-1 \_\_\_\_\_\_\_\_\_\_ Глубоков Г.В.

Подпись, дата

Руководитель

к.т.н.,доц. Батищев Р. В.

Подпись, дата

Липецк 2022г.

Задание кафедры:

Реализовать упрощенный интерпретатор команд Windows, выполняющий внешние команды, обрабатывающий одну встроенную команду и реализующий перенаправление ввода/вывода. В качестве встроенной команды следует использовать программу, реализованную в лабораторной работе № 1 (ее нужно оформить как функцию).

# Ход работы:

## Код программы на C++:

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <termios.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#define LSH\_RL\_BUFSIZE 1024

char \*lsh\_read\_line(void)

{

char \*line = NULL;

ssize\_t bufsize = 0; // getline сама выделит память

getline(&line, &bufsize, stdin);

return line;

}

#define LSH\_TOK\_BUFSIZE 64

#define LSH\_TOK\_DELIM " \t\r\n\a"

char \*\*lsh\_split\_line(char \*line)

{

int bufsize = LSH\_TOK\_BUFSIZE, position = 0;

char \*\*tokens = malloc(bufsize \* sizeof(char\*));

char \*token;

if (!tokens) {

fprintf(stderr, "lsh: ошибка выделения памяти\n");

exit(EXIT\_FAILURE);

}

token = strtok(line, LSH\_TOK\_DELIM);

while (token != NULL) {

tokens[position] = token;

position++;

if (position >= bufsize) {

bufsize += LSH\_TOK\_BUFSIZE;

tokens = realloc(tokens, bufsize \* sizeof(char\*));

if (!tokens) {

fprintf(stderr, "lsh: ошибка выделения памяти\n");

exit(EXIT\_FAILURE);

}

}

token = strtok(NULL, LSH\_TOK\_DELIM);

}

tokens[position] = NULL;

return tokens;

}

int lsh\_launch(char \*\*args)

{

pid\_t pid, wpid;

int status;

pid = fork();

if (pid == 0) {

// Дочерний процесс

if (execvp(args[0], args) == -1) {

perror("lsh");

}

exit(EXIT\_FAILURE);

} else if (pid < 0) {

// Ошибка при форкинге

perror("lsh");

} else {

// Родительский процесс

do {

wpid = waitpid(pid, &status, WUNTRACED);

} while (!WIFEXITED(status) && !WIFSIGNALED(status));

}

return 1;

}

/\*

Объявление функций для встроенных команд оболочки:

\*/

int lsh\_cd(char \*\*args);

int lsh\_help(char \*\*args);

int lsh\_exit(char \*\*args);

int lsh\_laba(char \*\*args);

/\*

Список встроенных команд, за которыми следуют соответствующие функции

\*/

char \*builtin\_str[] = {

"cd",

"help",

"exit",

"laba"

};

int (\*builtin\_func[]) (char \*\*) = {

&lsh\_cd,

&lsh\_help,

&lsh\_exit,

&lsh\_laba

};

int lsh\_laba(char \*\*args)

{

if(args!=0) {

execl("/Users/german/Desktop/OC/Laba-1/out","laba.cpp","../Laba-1/first.txt", "../Laba-1/last.txt", "4", "7", "10", NULL);

}

return 1;

}

int lsh\_num\_builtins() {

return sizeof(builtin\_str) / sizeof(char \*);

}

/\*

Реализации встроенных функций

\*/

int lsh\_cd(char \*\*args)

{

if (args[1] == NULL) {

fprintf(stderr, "lsh: ожидается аргумент для \"cd\"\n");

} else {

if (chdir(args[1]) != 0) {

perror("lsh");

}

}

return 1;

}

int lsh\_help(char \*\*args)

{

int i;

printf("Наберите название программы и её аргументы и нажмите enter.\n");

printf("Вот список встроенных команд:\n");

for (i = 0; i < lsh\_num\_builtins(); i++) {

printf(" %s\n", builtin\_str[i]);

}

printf("Используйте команду man для получения информации по другим программам.\n");

return 1;

}

int lsh\_exit(char \*\*args)

{

return 0;

}

int lsh\_execute(char \*\*args)

{

int i;

if (args[0] == NULL) {

// Была введена пустая команда.

return 1;

}

for (i = 0; i < lsh\_num\_builtins(); i++) {

if (strcmp(args[0], builtin\_str[i]) == 0) {

return (\*builtin\_func[i])(args);

}

}

return lsh\_launch(args);

}

void lsh\_loop(void)

{

char \*line;

char \*\*args;

int status;

do {

printf("> ");

line = lsh\_read\_line();

args = lsh\_split\_line(line);

status = lsh\_execute(args);

free(line);

free(args);

} while (status);

}

int main(int argc, char \*\*argv)

{

// Загрузка файлов конфигурации при их наличии.

// Запуск цикла команд.

lsh\_loop();

// Выключение / очистка памяти.

return EXIT\_SUCCESS;

}

## Пример выполнения:

На рисунке 1 показаны выполнение команд.

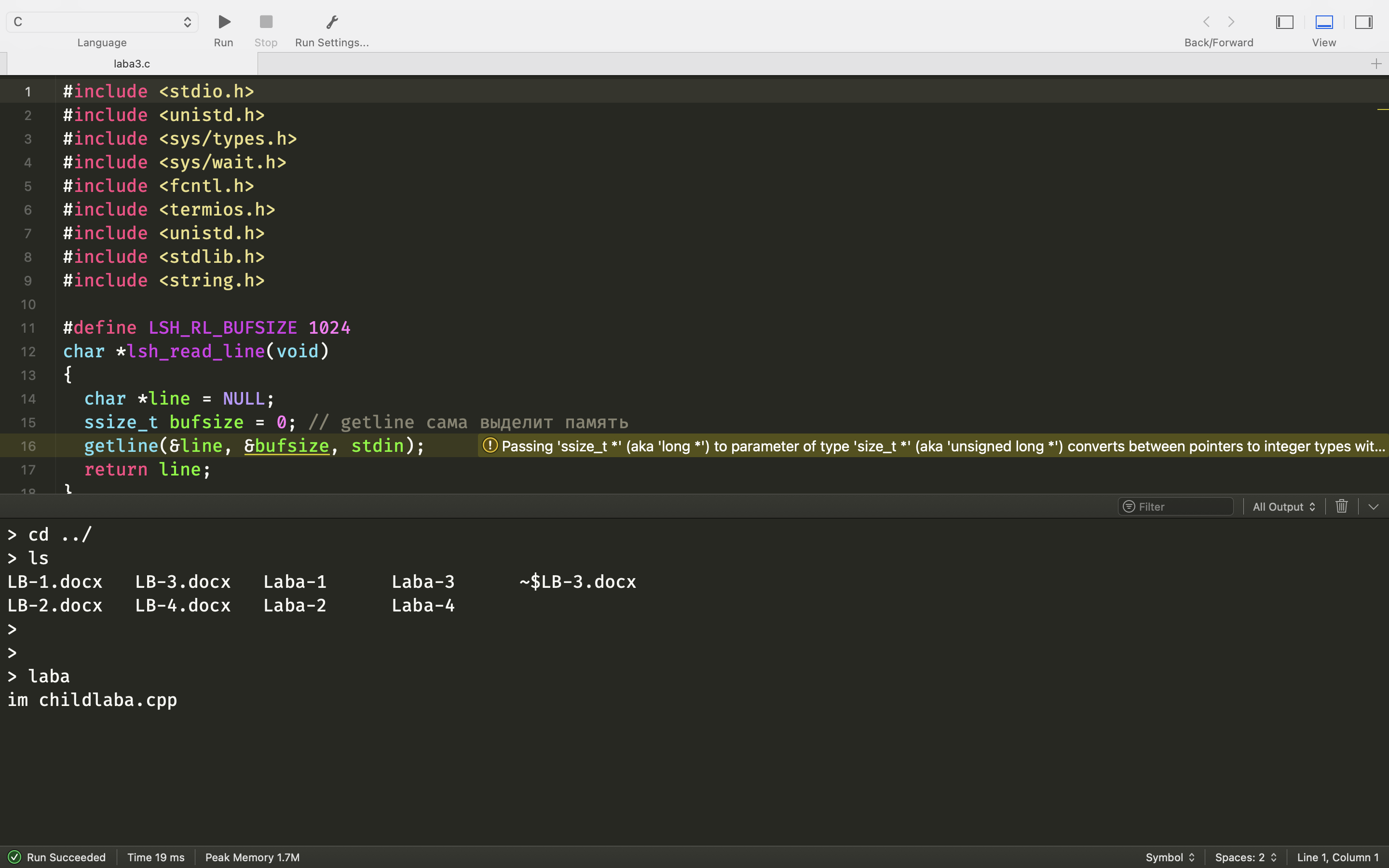


Рисунок 1 – Встроенная и внешняя команда