Федеральное агентство связи (Россвязь).

Федеральное государственное бюджетное образовательное учреждение

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

*Кафедра прикладной математики и кибернетики*

КУРСОВОЙ ПРОЕКТ

По дисциплине «Операционные системы»

Выполнил студент группы ИП-811

Адов А.С.

Проверил: преподаватель

Малков Е.А.

Новосибирск 2020

[Постановка задачи 3](#_Toc60010942)

[Текст программы 4](#_Toc60010943)

[Результаты выполнения программы 12](#_Toc60010944)

[Заключение 26](#_Toc60010945)

# Постановка задачи

Реализовать программу для частичного управления службами и процессами в системе windows/linux (диспетчер задач) с помощью директив препроцессора и системных вызовов.

# Текст программы

#include <stdio.h>

#include <stdlib.h>

//#define SYSTEM\_WINDOWS

#define SYSTEM\_LINUX

#define INCORRECT\_END\_PROGRAMM 0b00000001

#define ERROR\_ALLOCATION\_MEMORY 0b00000010

#define ERROR\_REALLOCATION\_MEMORY 0b00000100

#define SPACE\_BETWEEN\_COMPOSITE\_AND\_END\_ZERO 2

#define MAXIMUM\_BUFFER\_SIZE 50

#define FIRST\_CODE\_SYMBOL\_DIGIT 0x30

#define LAST\_CODE\_SYMBOL\_DIGIT 0x39

#ifdef SYSTEM\_WINDOWS

#define SIZE\_COMMAND\_KILL\_PROCESS\_PID 13

#define SIZE\_COMMAND\_KILL\_PROCESS\_NAME 12

#define SIZE\_COMMAND\_STOP\_SERVICE 7

#define SIZE\_COMMAND\_START\_SERVICE 8

#define COMMAND\_SHOW\_PROCESS "tasklist"

#define COMMAND\_SHOW\_SERVICE "sc query"

#define COMMAND\_KILL\_PROCESS\_PID "taskkill /PID"

#define COMMAND\_KILL\_PROCESS\_NAME "taskkill /IM"

#define COMMAND\_STOP\_SERVICE "sc stop"

#define COMMAND\_START\_SERVICE "sc start"

#define COMMAND\_CLEAR\_SCREEN "CLS"

#endif

#ifdef SYSTEM\_LINUX

#define SIZE\_COMMAND\_KILL\_PROCESS\_PID 10

#define SIZE\_COMMAND\_KILL\_PROCESS\_NAME 11

#define SIZE\_COMMAND\_STOP\_SERVICE 12

#define SIZE\_COMMAND\_START\_SERVICE 13

#define COMMAND\_SHOW\_PROCESS "ps -a"

#define COMMAND\_SHOW\_SERVICE "service --status-all"

#define COMMAND\_KILL\_PROCESS\_PID "kill -TERM"//

#define COMMAND\_KILL\_PROCESS\_NAME "pkill -TERM"//

#define COMMAND\_STOP\_SERVICE "systemctl stop"

#define COMMAND\_START\_SERVICE "systemctl start"

#define COMMAND\_CLEAR\_SCREEN "clear"

#endif

void Draw\_Menu()

{

printf("1) Show Process\n");

printf("2) Show Services\n");

printf("3) Kill Process by pid\n");

printf("4) Kill Process by name\n");

printf("5) Stop Services\n");

printf("6) Start Services\n");

printf("9) Clear Screen\n");

printf("0) Exit\n");

}

int Get\_User\_String(char\* user\_string[], int\* size\_user\_string)

{

char\* buffer = NULL;

int quantity\_symbol = -1;

buffer = (char\*)calloc(MAXIMUM\_BUFFER\_SIZE, sizeof(char));

if(buffer == NULL)

{

return ERROR\_ALLOCATION\_MEMORY;

}

printf("write pid or name\n");

getchar();

do

{

quantity\_symbol++;

fread(&(buffer[quantity\_symbol]),sizeof(char),1,stdin);

if(quantity\_symbol > MAXIMUM\_BUFFER\_SIZE)

{

realloc(buffer,(quantity\_symbol + MAXIMUM\_BUFFER\_SIZE));

if(buffer == NULL)

{

return ERROR\_REALLOCATION\_MEMORY;

}

}

}while(buffer[quantity\_symbol] != '\n');

\*size\_user\_string = quantity\_symbol;

\*user\_string = (char\*)calloc(quantity\_symbol, sizeof(char));

if(\*user\_string == NULL)

{

return ERROR\_ALLOCATION\_MEMORY;

}

for(short int i = 0; i < quantity\_symbol; i++)

{

(\*user\_string)[i] = buffer[i];

}

free(buffer);

buffer = NULL;

return 0;

}

int Create\_Composite\_System\_Command(char\* system\_command[], char user\_string[], char composite[], int size\_user\_string)

{

\*system\_command = (char\*)calloc(size\_user\_string,sizeof(char));

if(\*system\_command == NULL)

{

return ERROR\_ALLOCATION\_MEMORY;

}

sprintf(\*system\_command,"%s %s", composite, user\_string);

return 0;

}

int main()

{

char\* system\_command = NULL;

char\* user\_string = NULL;

int size\_user\_string = 0;

int error\_flag = 0b0;

char key\_choice = '0';

do

{

Draw\_Menu();

scanf(" %c",&key\_choice);

switch(key\_choice)

{

case('1'):

{

system(COMMAND\_SHOW\_PROCESS);

break;

}

case('2'):

{

system(COMMAND\_SHOW\_SERVICE);

break;

}

case('3'):

{

error\_flag = (error\_flag | Get\_User\_String(&user\_string, &size\_user\_string));

size\_user\_string = (size\_user\_string + SIZE\_COMMAND\_KILL\_PROCESS\_PID + SPACE\_BETWEEN\_COMPOSITE\_AND\_END\_ZERO);

error\_flag = (error\_flag | Create\_Composite\_System\_Command(&system\_command, user\_string, COMMAND\_KILL\_PROCESS\_PID, size\_user\_string));

system(system\_command);

size\_user\_string = 0;

free(user\_string);

user\_string = NULL;

free(system\_command);

system\_command = NULL;

break;

}

case('4'):

{

error\_flag = (error\_flag | Get\_User\_String(&user\_string, &size\_user\_string));

size\_user\_string = (size\_user\_string + SIZE\_COMMAND\_KILL\_PROCESS\_NAME + SPACE\_BETWEEN\_COMPOSITE\_AND\_END\_ZERO);

error\_flag = (error\_flag | Create\_Composite\_System\_Command(&system\_command, user\_string, COMMAND\_KILL\_PROCESS\_NAME, size\_user\_string));

system(system\_command);

size\_user\_string = 0;

free(user\_string);

user\_string = NULL;

free(system\_command);

system\_command = NULL;

break;

}

case('5'):

{

error\_flag = (error\_flag | Get\_User\_String(&user\_string, &size\_user\_string));

size\_user\_string = (size\_user\_string + SIZE\_COMMAND\_STOP\_SERVICE + SPACE\_BETWEEN\_COMPOSITE\_AND\_END\_ZERO);

error\_flag = (error\_flag | Create\_Composite\_System\_Command(&system\_command, user\_string, COMMAND\_STOP\_SERVICE, size\_user\_string));

system(system\_command);

size\_user\_string = 0;

free(user\_string);

user\_string = NULL;

free(system\_command);

system\_command = NULL;

break;

}

case('6'):

{

error\_flag = (error\_flag | Get\_User\_String(&user\_string, &size\_user\_string));

size\_user\_string = (size\_user\_string + SIZE\_COMMAND\_START\_SERVICE + SPACE\_BETWEEN\_COMPOSITE\_AND\_END\_ZERO);

error\_flag = (error\_flag | Create\_Composite\_System\_Command(&system\_command, user\_string, COMMAND\_START\_SERVICE, size\_user\_string));

system(system\_command);

size\_user\_string = 0;

free(user\_string);

user\_string = NULL;

free(system\_command);

system\_command = NULL;

break;

}

case('9'):

{

system(COMMAND\_CLEAR\_SCREEN);

break;

}

case('0'):

{

return error\_flag;

}

default:

{

printf("error: incorrect choice\n");

break;

}

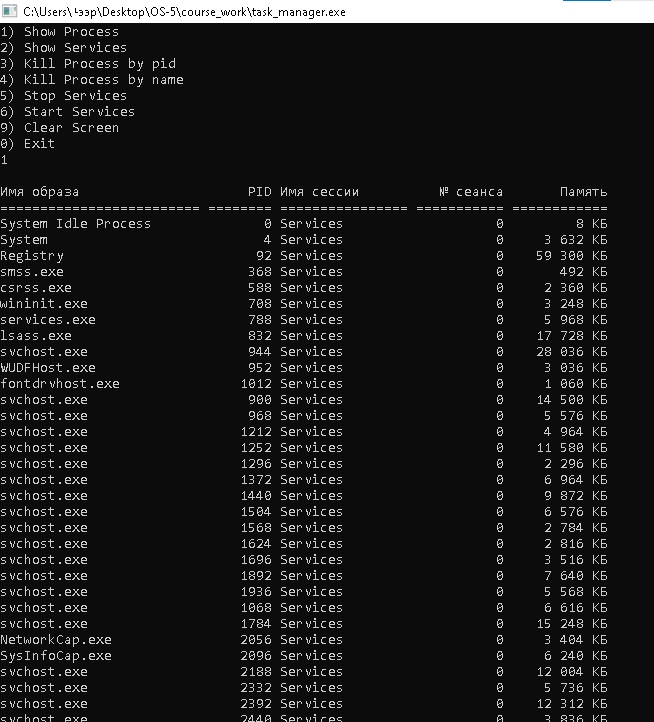
}

} while(key\_choice != '0');

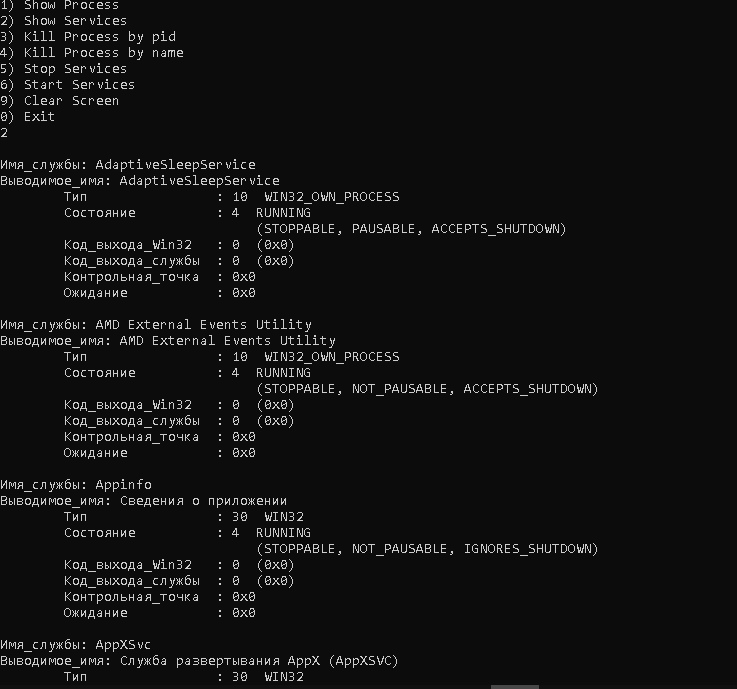
return (error\_flag | INCORRECT\_END\_PROGRAMM);

}

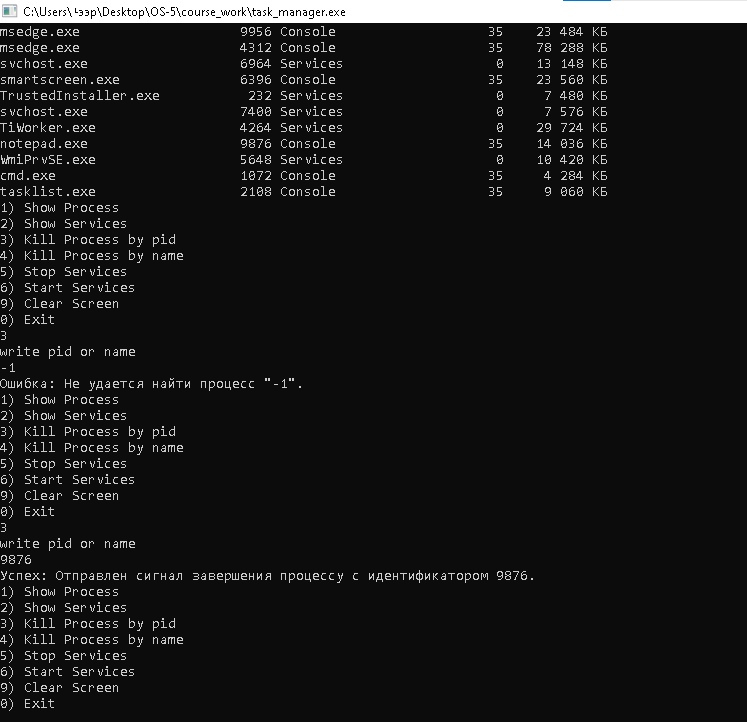
# Результаты выполнения программы



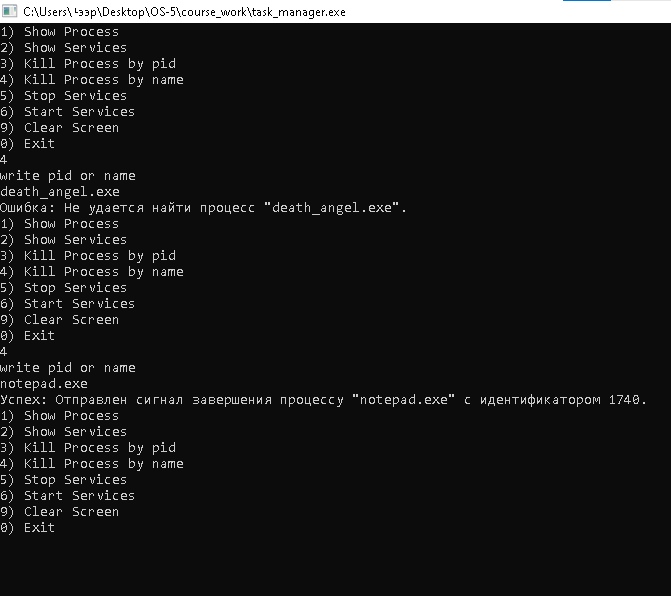
***Рис.1 «Результат выполнения программы при выборе 1-ого пункта меню(показать процессы) в системе windows»***



***Рис.2 «Результат выполнения программы при выборе 2-ого пункта меню(показать службы) в системе windows»***



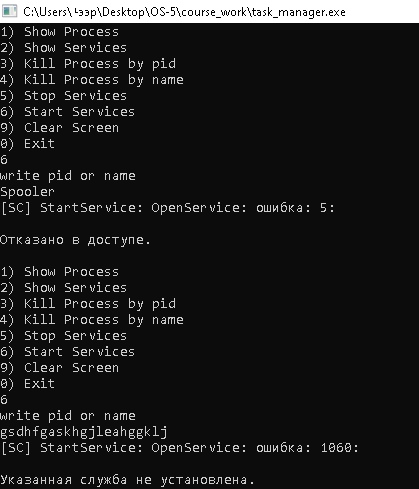
***Рис.3 «Результат выполнения программы при выборе 3-ого пункта меню(убить процесс по его pID) в системе windows»***



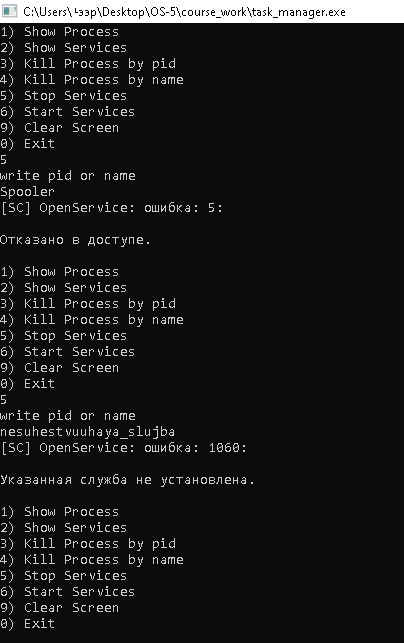
***Рис.4 «Результат выполнения программы при выборе 4-ого пункта меню(убить процесс по его имени) в системе windows»***



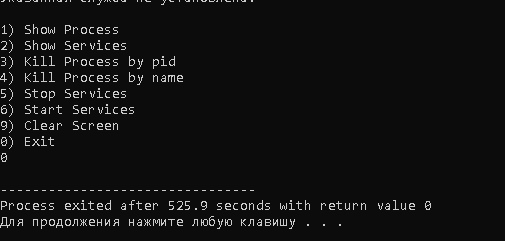
***Рис.5 «Результат выполнения программы при выборе 9-ого пункта меню(очистить экран) в системе windows»***

**

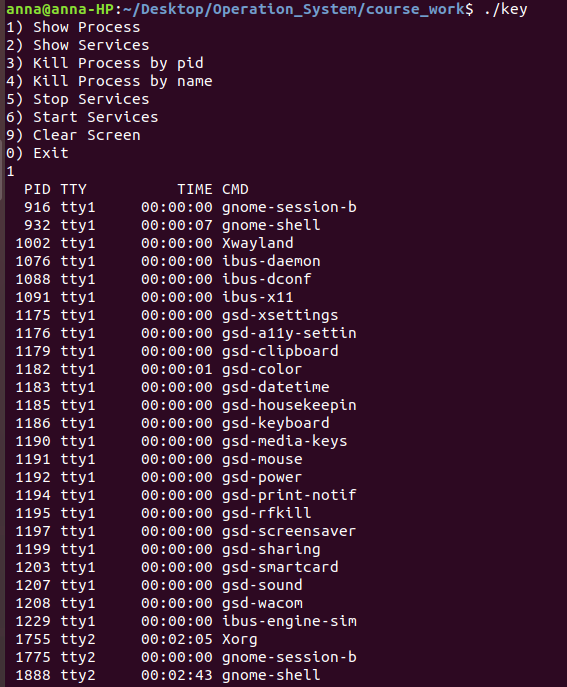
***Рис.6 «Результат выполнения программы при выборе 6-ого пункта меню(запустить службу) в системе windows»***



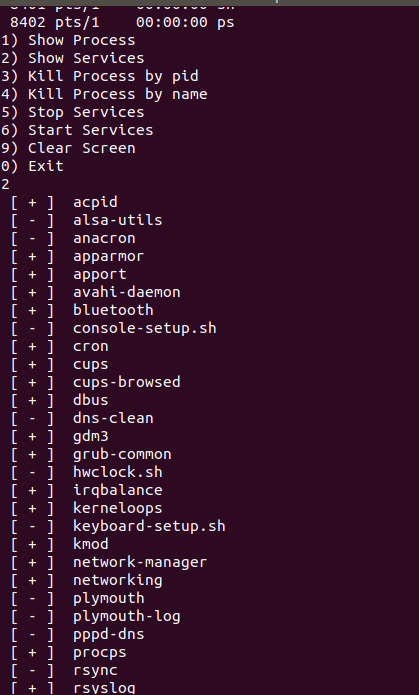
***Рис.7 «Результат выполнения программы при выборе 5-ого пункта меню(остановить службу) в системе windows»***



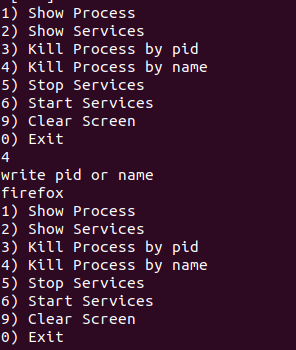
***Рис.8 «Результат выполнения программы при выборе 0-ого пункта меню(выйти из программы) в системе windows»***

**

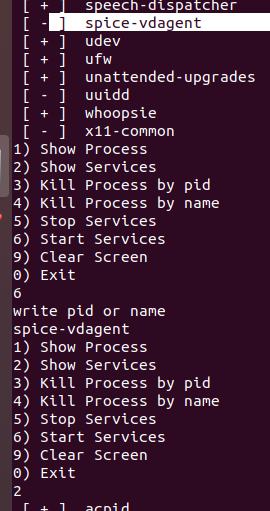
***Рис.9 «Результат выполнения программы при выборе 1-ого пункта меню(показать процессы) в системе linux»***

**

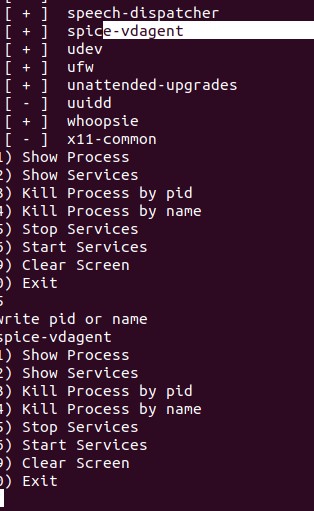
***Рис.10 «Результат выполнения программы при выборе 2-ого пункта меню(показать службы) в системе linux»***



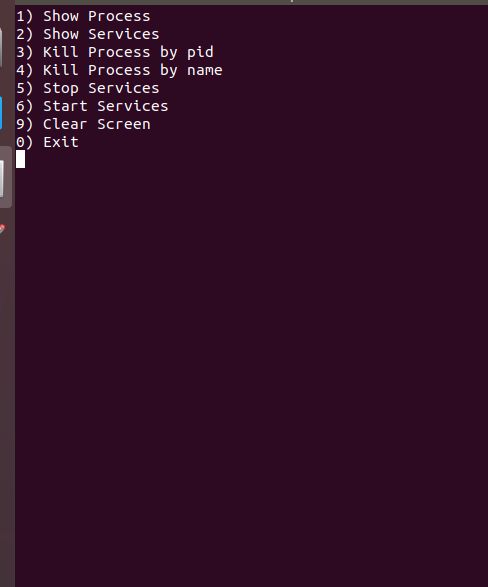
***Рис.11 «Результат выполнения программы при выборе 4-ого пункта меню(убить процесс по его имени) в системе linux»***

**

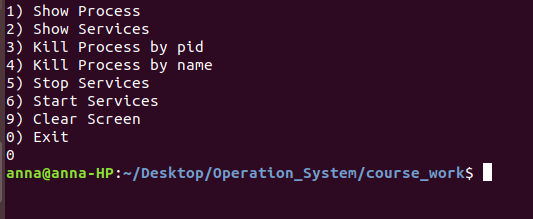
***Рис.12 «Результат выполнения программы при выборе 6-ого пункта меню(запустить службу) в системе linux»***

**

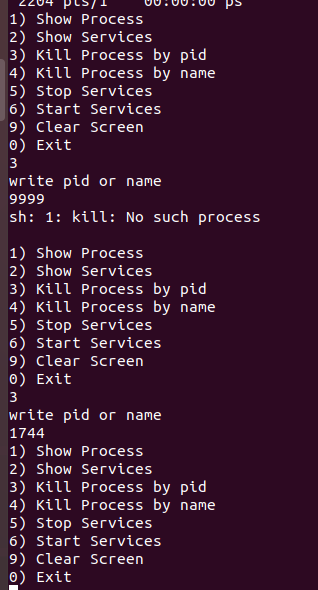
***Рис.13 «Результат выполнения программы при выборе 5-ого пункта меню(остановить службу) в системе linux»***

**

***Рис.14 «Результат выполнения программы при выборе 9-ого пункта меню(очистить экран) в системе linux»***



***Рис.15 «Результат выполнения программы при выборе 0-ого пункта меню(выйти из программы) в системе linux»***



***Рис.16 «Результат выполнения программы при выборе 3-ого пункта меню(убить процесс по его pID) в системе linux»***

# Заключение

Реализовал программу для частичного управления службами и процессами в системе windows/linux (диспетчер задач) с помощью директив препроцессора и системных вызовов.