Выполнила: Белоусова Е., ИП-911

Задача

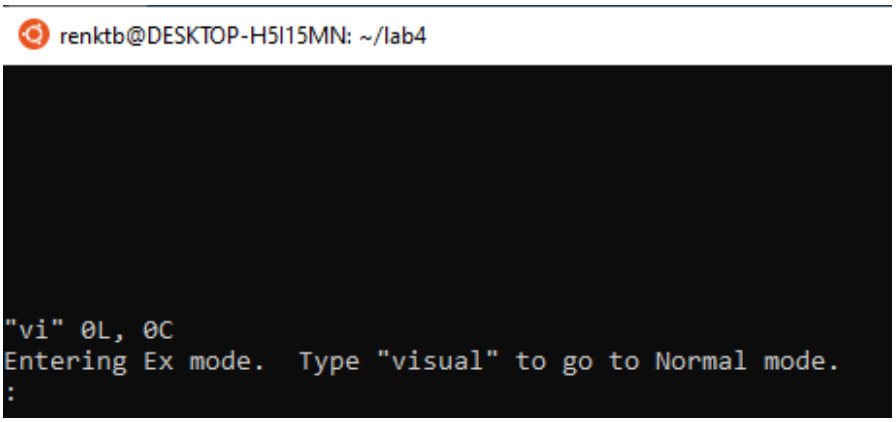
Цель: знакомство с созданием процессов с помощью POSIX API в Linux и Windows.
Упражнение: Протестируйте различные варианты программ, рассмотренных на Лекции 4.
Задание: Разработайте оболочку для выполнения (некоторых) команд командной строки Windows, используя в качестве синонимов названия команд GNU/bash.

Описание работы программы

Тесты вариантов программ из Лекции 4.

```
renktb@DESKTOP-H5I15MN:~/lab4$ gcc -o ex1 ex1.c
renktb@DESKTOP-H5I15MN:~/lab4$ ./ex1 vi
```

Программа запустила редактор VIM.




```
Before child process creating: PARENT ID = 13540
renktb@DESKTOP-H5I15MN:~/lab4$
```

Раскомментируем строку execvp("ls", argv). Создается процесс, вызывающий ls для текущей директории.

```
renktb@DESKTOP-H5I15MN:~/lab4$ ./ex1
Before child process creating: PARENT ID = 13557
ex1  ex1.c  vi
renktb@DESKTOP-H5I15MN:~/lab4$
```

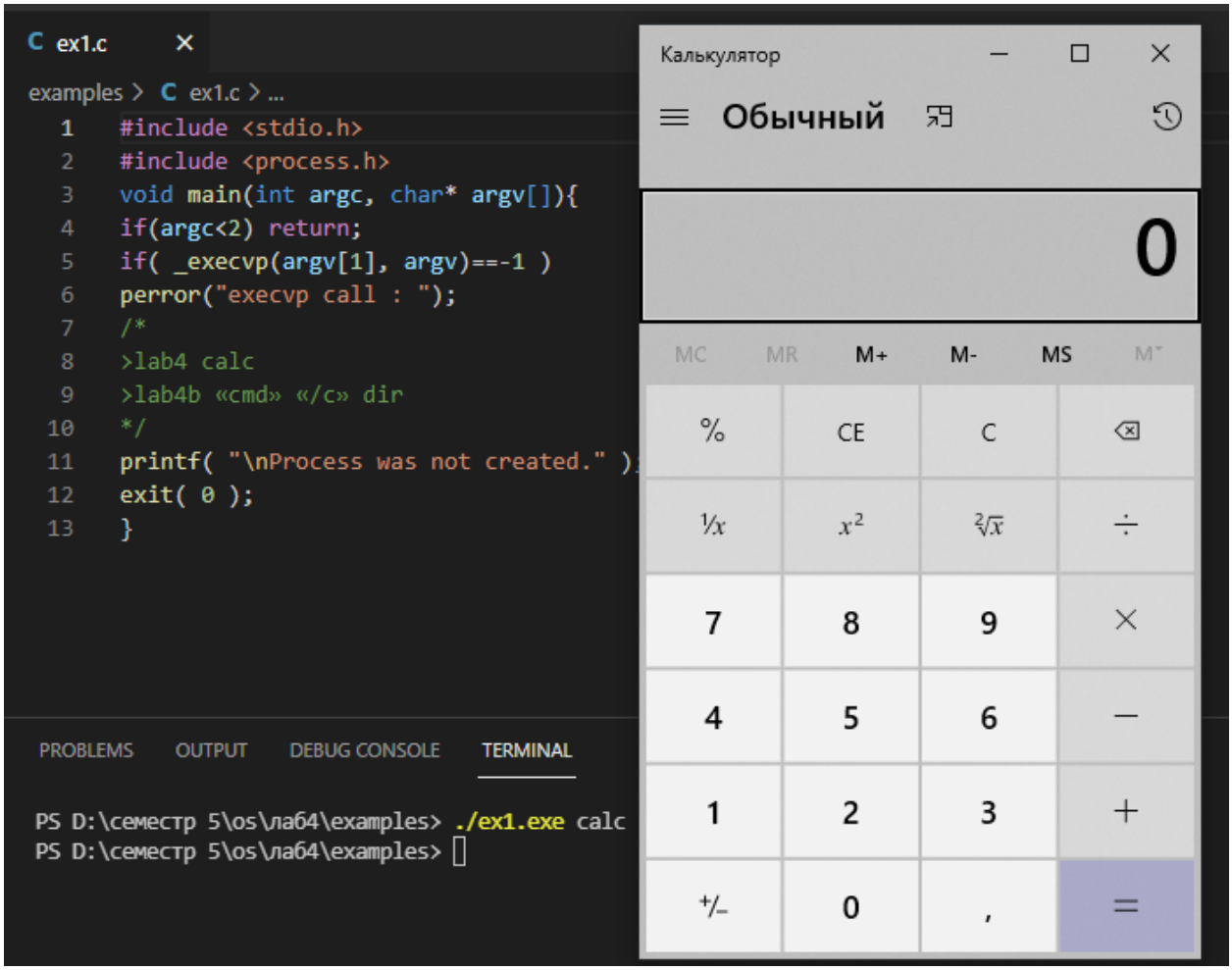
Совместное использование fork и execvp:

```
renktb@DESKTOP-H5I15MN:~/lab4$ ./ex2 -l *.c
The main program is yet running!
renktb@DESKTOP-H5I15MN:~/lab4$ -rw-r--r-- 1 renktb renktb 327 Oct  3 22:53 ex1.c
-rw-r--r-- 1 renktb renktb 241 Oct  3 23:02 ex2.c
```

В третьем примере мы создаем дочерний процесс, затем убиваем его.

```
renktb@DESKTOP-H5I15MN:~/lab4$ touch ex3.c
renktb@DESKTOP-H5I15MN:~/lab4$ vi ex3.c
renktb@DESKTOP-H5I15MN:~/lab4$ gcc -o ex3 ex3.c
renktb@DESKTOP-H5I15MN:~/lab4$ ./ex3
Before RECREATION 13580
I'm not yet dead! My ID is 13580
I'm not yet dead! My ID is 13580
I'm not yet dead! My ID is 13580
Who I am? My ID is 13581
I'm not yet dead! My ID is 13580
I'm not yet dead! My ID is 13580
renktb@DESKTOP-H5I15MN:~/lab4$
```

Следующая программа создает процесс, вызывая программу, название которой мы передаем аргументом. Попробуем вызывать калькулятор:



Попробуем вызывать командную строку и передать ей команду для выполнения:

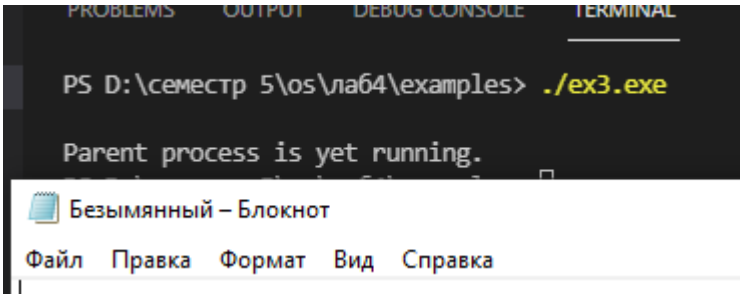
```
PS D:\семестр 5\os\лаб4\examples> ./ex1.exe <cmd> </c> dir
execvp call : : No such file or directory

Process was not created.
PS D:\семестр 5\os\лаб4\examples> ./ex1.exe "cmd" "/c" dir
PS D:\семестр 5\os\лаб4\examples>  Том в устройстве D имеет метку Новый том
 Серийный номер тома: BAD3-38C4

 Содержимое папки D:\семестр 5\os\лаб4\examples

03.10.2021  23:11    <DIR>          .
03.10.2021  23:11    <DIR>          ..
03.10.2021  23:11               260 ex1.c
03.10.2021  23:11           127 488 ex1.exe
03.10.2021  23:11             2 580 ex1.obj
               3 файлов        130 328 байт
               2 папок  432 596 721 664 байт свободно
```

Попробуем выполнить то же самое, но зададим аргументы статическим массивом:

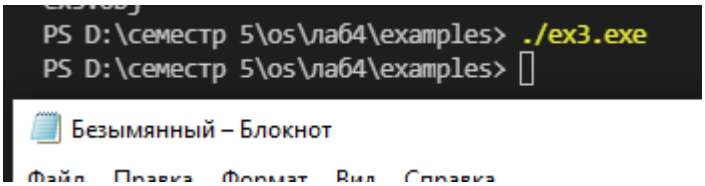
```
PS D:\семестр 5\os\лаб4\examples> ./ex2.exe
PS D:\семестр 5\os\лаб4\examples>  Том в устройстве D имеет метку Новый том
 Серийный номер тома: BAD3-38C4

 Содержимое папки D:\семестр 5\os\лаб4\examples

03.10.2021  23:17    <DIR>          .
03.10.2021  23:17    <DIR>          ..
03.10.2021  23:11               260 ex1.c
03.10.2021  23:11           127 488 ex1.exe
03.10.2021  23:11             2 580 ex1.obj
03.10.2021  23:16               178 ex2.c
03.10.2021  23:17           126 464 ex2.exe
03.10.2021  23:17             2 786 ex2.obj
               6 файлов        259 756 байт
               2 папок  432 596 586 496 байт свободно
```

```c
#include <stdio.h>
#include <process.h>
void main()
{
    char* argv[] = { "cmd", "/C", "dir", NULL };
    _execvp(argv[0], argv);
    printf( "\nProcess was not created.");
    exit( 0 );
}
```

Воспользуемся _spawnwp для создания процесса (вызов блокнота). Аргумент mode определяет действие, предпринимаемое вызывающим процессом перед вызовом функции _spawn и во время ее выполнения. В примере мы используем _P_NOWAIT (продолжает выполнять вызывающий процесс параллельно с новым процессом) и _P_OVERLAY (перекрывает вызывающий процесс новым процессом, уничтожая вызывающий процесс (тот же эффект, что и при вызовах функций _exec)).

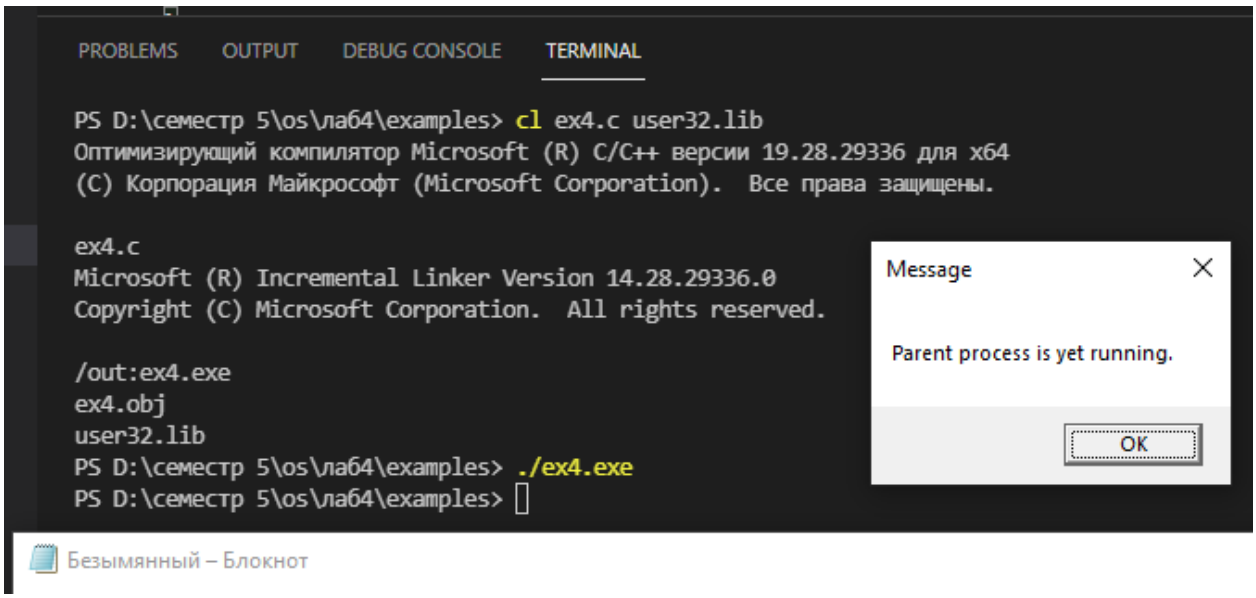


_P_NOWAIT



_P_OVERLAY

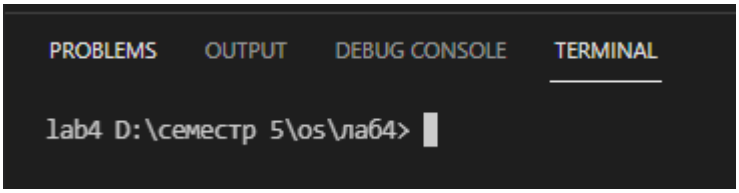
В следующей программе создадим окно с сообщением о том, что процесс всё еще выполняется и вызовем блокнот. Так же будем использовать _P_NOWAIT и _P_OVERLAY. При _P_NOWAIT сообщение о выполнении процесса появляется.



А при _P_OVERLAY окошко не появляется, т.к. новый процесс перекрывает выполнение старого.

Задание

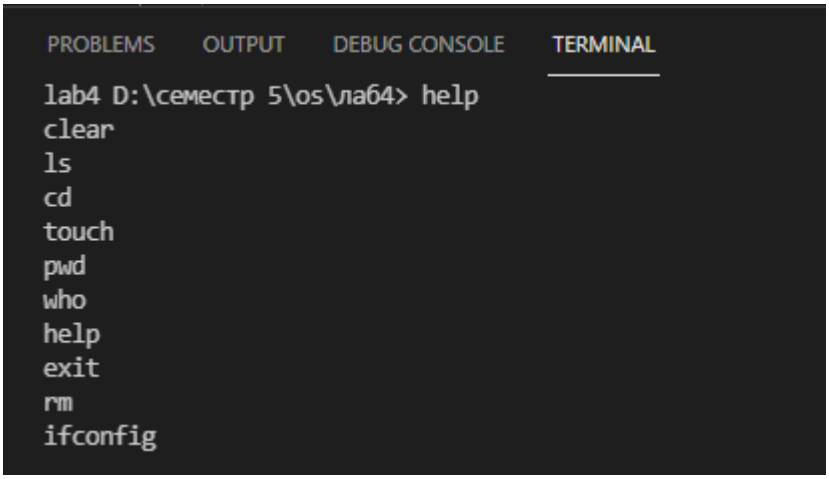
Основная программа osLab4.exe. При ее запуске определяется директория, из которой запускается программа. Далее запускается бесконечный цикл, в котором будет выводиться текущая директория и ожидаться ввод команды.

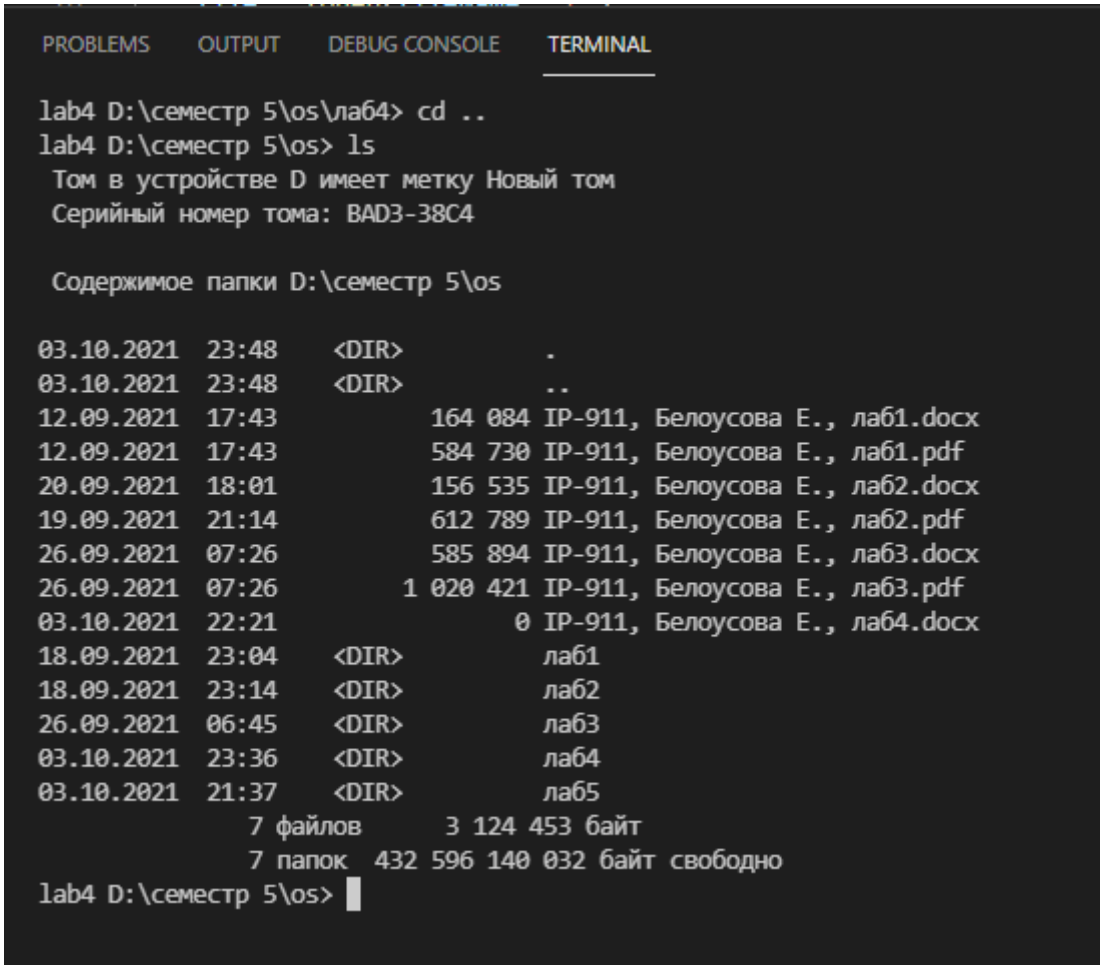


После ввода команды вызывается функция commandHandler. В ней команда отделяется от остальных аргументов и определяется ее индекс в массиве доступных команд. Далее с помощью

switch выбирается необходимый блок действий. Для исполнения каждой команды создается новый процесс с помощью _spawnl и аргумента _P_WAIT (приостанавливает вызывающий поток до тех пор, пока не будет завершено выполнение нового процесса).

Список доступных команд:





```
lab4 D:\семестр 5\os\лаб4> who
grant
lab4 D:\семестр 5\os\лаб4> ifconfig

Настройка протокола IP для Windows


Адаптер Ethernet Radmin VPN:

   DNS-суффикс подключения . . . . . :
   IPv6-адрес. . . . . . . . . . . . : fdfd::1a43:a320
   Локальный IPv6-адрес канала . . . : fe80::1d21:8295:b7a5:17d4%5
   IPv4-адрес. . . . . . . . . . . . : 26.67.163.32
   Маска подсети . . . . . . . . . . : 255.0.0.0
   Основной шлюз. . . . . . . . . . : 26.0.0.1
```

```
lab4 D:\семестр 5\os\лаб4> cd C:\
lab4 C:\> pwd
C:\
lab4 C:\> ls
 Том в устройстве C не имеет метки.
 Серийный номер тома: 8E5A-7636

 Содержимое папки C:\

14.08.2021  16:09    <DIR>          $WINDOWS.~BT
14.08.2021  18:20    <DIR>          ESD
31.01.2021  17:39    <DIR>          Intel
07.12.2019  16:14    <DIR>          PerfLogs
26.09.2021  09:33    <DIR>          Program Files
26.09.2021  08:12    <DIR>          Program Files (x86)
07.09.2021  02:20    <DIR>          Qt
31.01.2021  17:32    <DIR>          Users
28.09.2021  20:20    <DIR>          Windows
               0 файлов              0 байт
               9 папок  103 372 623 872 байт свободно
lab4 C:\>
```

Листинг

```
//osLab4.c
#include <process.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include <locale.h>

#define BUFSIZE MAX_PATH
//cmd.exe /k "C:\Program Files (x86)\Microsoft Visual
Studio\2019\Enterprise\VC\Auxiliary\Build\vcvars64.bat" `& powershell
//cd  D:\"семестр 5"\os\"лаб4"
TCHAR commandPath[MAX_PATH];
TCHAR currentPath[MAX_PATH];
const int commandsCount = 10;
```

```
const char* commands[] = {
	"clear",
	"ls",
	"cd",
	"touch",
	"pwd",
	"who",
	"help",
	"exit",
	"rm",
	"ifconfig"
};

void printCurrentPath()
{
	GetCurrentDirectory(BUFSIZE, currentPath);
	printf("lab4 %s> ", currentPath);
}

void commandHandler(char* command)
{
	char basicCommand[255];
	strcpy(basicCommand, command);
	char* arg = strtok(command, " ");
	int commandIndex = -1;
	for (int i = 0; i < commandsCount; i++)
	{
		if (strcmp(arg, commands[i]) == 0)
		{
			commandIndex = i;
			break;
		}
	}

	TCHAR commandArg[MAX_PATH];
	sprintf(commandArg, "%s\\%s", commandPath, arg);
	TCHAR tmp[255];
	TCHAR currentAbsPath[255];
	TCHAR commandAbsPath[255];
	sprintf(currentAbsPath, "\"%s\"", currentPath);
	sprintf(commandAbsPath, "\"%s\"", commandPath);
	switch (commandIndex)
	{
		case(0): // clear
			_spawnl(_P_WAIT, commandArg, "clear", NULL);
			break;
		case(1): // ls
			command = strtok(NULL,"");
			if (command == NULL)
			{
				_spawnl(_P_WAIT, commandArg, currentAbsPath, NULL);
			}
			else
			{
```

```
                _spawnl(_P_WAIT, commandArg, command, NULL);
        }
        break;
case(2): // cd
        TCHAR* tmp;
        tmp = strtok(NULL, "");
        // printf("%u\n", GetConsoleCP());
        if (SetCurrentDirectory(tmp) == 0)
        {
                printf("Wrong path.\n");
        }

        break;
case(3): // touch
        command = strtok(NULL,"");
        if (command == NULL)
        {
                printf("There are too few arguments for this command.");
        }
        else
        {
                GetCurrentDirectory(BUFSIZE, currentPath);
                char fileName[255];
                sprintf(fileName, "\"%s\\%s\"", currentPath, command);

                _spawnl(_P_WAIT, commandArg, fileName,NULL);
        }
        break;
case(4): // pwd
        _spawnl(_P_WAIT, commandArg, currentAbsPath, NULL);
        break;
case(5): // who
        _spawnl(_P_WAIT, commandArg, "who", NULL);
        break;
case(6): // help
        //printf("%s\n", commandPath);
        _spawnl(_P_WAIT, commandArg, commandAbsPath, NULL);
        break;
case(7):
        SetConsoleCP(866);
SetConsoleOutputCP(866);
        exit(0);
case(8): // rm
        command = strtok(NULL,"");
        if (command == NULL)
        {
                printf("There are too few arguments for this command.");
        }
        else
        {
                GetCurrentDirectory(BUFSIZE, currentPath);
                char fileName[255];
                sprintf(fileName, "\"%s\\%s\"", currentPath, command);
```

```
                                _spawnl(_P_WAIT,commandArg,fileName,NULL);
                        }
                        break;
                case(10): // ifconfig
                        _spawnl(_P_WAIT, commandArg, "ifconfig", NULL);
                        break;
                default:
                        system(basicCommand);
                        break;
        }

}

int main()
{
        GetCurrentDirectory(BUFSIZE,commandPath);
        setlocale(LC_ALL,"Russian");
        system("cls");
        SetConsoleCP(1251);
   SetConsoleOutputCP(1251);
        while (1)
        {
                char command[255];
                printCurrentPath();
                gets(&command);
                commandHandler(command);
        }

        return 0;
}
//clear.c
#include <stdio.h>
#include <stdlib.h>
#include <process.h>

int main()
{
        system("cls");
}
//help.c
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <locale.h>

#define N 255

int main(int argc, char* argv[])
{
        setlocale(LC_ALL,"Russian");
        FILE *file;
   char c;
        char fileName[255];
        sprintf(fileName, "%s\\help.txt", argv[0]);
```

```
        //printf("%s\n", fileName);
   file = fopen(fileName, "r");
        if (file == NULL)
        {
                return -1;
        }
        c = fgetc(file);
        while (c != EOF)
        {
                printf("%c",c);
                c = fgetc(file);
        }

   printf("\n");
   fclose(file);
        return 0;
}
//ifconfig.c
#include <stdio.h>
#include <stdlib.h>
#include <process.h>

int main()
{
        system("ipconfig");
}
#include <stdio.h>
//ls.c
#include <stdlib.h>
#include <process.h>
#include <locale.h>

int main(int argc, char* argv[])
{
        setlocale(LC_ALL,"Russian");
        char buffer[255];
        sprintf(buffer, "dir \"%s\"", argv[0]);
        system(buffer);
        return 0;
}
//pwd.c
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <locale.h>

int main(int argc, char* argv[])
{
        setlocale(LC_ALL,"RUSSIAN");
        // for(int i = 0; i < argc; i++)
        // {
        //      printf("%s",argv[i]);
        // }
        printf("%s\n", argv[0]);
```

```
	return 0;
}
//rm.c
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <locale.h>

int main(int argc, char* argv[])
{
   if (remove(argv[0]) != 0)
   {
      printf("Error\n");
   }
   return 0;
}
//touch.c
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <locale.h>

int main(int argc, char* argv[])
{
	// for(int i=0; i<argc; i++)
	// {
	//	printf("%s\n", argv[i]);
	// }
	// char* path = malloc(argc + 1);
	// for (int i = 0; i < argc; i++)
	// {
	//	strcat(path, argv[i]);
	// }
   // printf("%s\n", path);
	FILE* file = fopen(argv[0], "a+");
	fclose(file);
	return 0;
}

//who
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <locale.h>

int main()
{
	system("echo %username%");
}
```