ФГБОУ «Национальный исследовательский ядерный университет «МИФИ»

Кафедра «Управляющие интеллектуальные системы»

Группа К5-291

|  |
| --- |
| Заказчик |
| М.Н. Петухов |
| 2012 года |

Модель файловой системы ОС реального времени

Монитор обработки команд файловой системы

Текст программы

ГОСТ 19.401-78

Исполнитель С.С. Богатыренко

Принял И.В. Алексеенко

Москва 2012

# Аннотация

В данном документе представлен текст программы «Монитор обработки команд программы «Модель файловой системы ОС реального времени»». Помимо исходного кода программы документ содержит описание классов файловой системы и монитора команд и их методов. Программа написана в среде Microsoft Visual Studio 2010, на языке С++.

Перед прочтением данного документа рекомендуется ознакомиться с пояснительной запиской к учебному проекту «Монитор обработки команд программы «Модель файловой системы ОС реального времени»» и техническим заданием к учебному проекту «Монитор обработки команд программы «Модель файловой системы ОС реального времени»».

# Содержание

[1. Аннотация 1](#_Toc341918371)

[2. Содержание 2](#_Toc341918372)

[3. Основная часть 3](#_Toc341918373)

[3.1. Содержимое файла Code.cpp - исходный код программы. 3](#_Toc341918374)

[3.2. Содержимое файла Monitor.h – описание класса монитора команд и его методов. 4](#_Toc341918375)

[3.3. Содержимое файла FS.h – описание класса файловой системы и его методов. 28](#_Toc341918376)

# Основная часть

# **3.1. Содержимое файла Code.cpp - исходный код программы.**

#include <iostream>

#include <sstream>

#include <fstream>

#include <cstring>

#include <string>

#include <cmath>

#include "FS.h"

#include "Monitor.h"

using namespace std;

void main()

{try

{

setlocale(0, "Rus");

Monitor MON;//Инициализируем монитор команд

cout<<"Вас приветствует модель файловой системы ОС реального времени. ";//Вводим приветствие

cout<<"Данная программа моделирует работу файловой системы и имитирует её основные функции.\n";

cout<<"Чтобы получить информацию о предоставляемых командах введите “help”.\n";

while(1)//В бескончном цикле считываем, выполняем команду и проверяем возвращенный код

{

MON.readcom();

MON.execute();

MON.errcheck();

if(exitflag==1)//Если установлен флаг выхода, то выходим

break;

}

}

catch(...)//Перехватываем ошибки

{

cout<<"Во время работы программы возникла непредвиденная ошибка. Программа завершает своё выполнение.\n";

}

}

# **3.2. Содержимое файла Monitor.h – описание класса монитора команд и его методов.**

using namespace std;

class Monitor{//класс монитора команд

public:

struct command{//структура команды

char name[14];//имя команды

char par1[13];//параметры команды

char par2[13];

char par3[13];

int numc;//номер команды

int nump;//число параметров команды

long size;//используется для передачи функциям числовых параметров

int retcode;//код, возвращенный командой

};

FS SYSTEM;// объект класса файловой системы

string commands[28];//массив, содержащий имена команд

command com;//обрабатываемая команда

Monitor();//конструктор

void readcom();//считывание команды

void execute();//выполнение команды

void errcheck();//проверка ошибок

};

Monitor::Monitor()//конструктор, создает пустую файловую систему,забивает поля в мониторе нулями, инициализирует массив команд

{

SYSTEM=FS();

for(int i=0;i<13;i++)

{

com.par1[i]=0;

com.par2[i]=0;

com.par3[i]=0;

com.name[i]=0;

}

com.name[13]=0;

com.numc=0;

com.nump=0;

com.size=0;

com.retcode=0;

screenflag=errflag=isformatted=0;

commands[0]="format";

commands[1]="content";

commands[2]="sortcontent";

commands[3]="freespace";

commands[4]="create";

commands[5]="del";

commands[6]="rename";

commands[7]="append";

commands[8]="changetype";

commands[9]="defrag";

commands[10]="userandlabel";

commands[11]="createFS";

commands[12]="help";

commands[13]="exit";

commands[14]="format?";

commands[15]="content?";

commands[16]="sortcontent?";

commands[17]="freespace?";

commands[18]="create?";

commands[19]="del?";

commands[20]="rename?";

commands[21]="append?";

commands[22]="changetype?";

commands[23]="defrag?";

commands[24]="userandlabel?";

commands[25]="createFS?";

commands[26]="help?";

commands[27]="exit?";

}

void Monitor::readcom()//Считывание команды

{

char line[54];//Максимальная длина команды - 53, создаем массив под команду

int i=0;//Счетчики

int j=0;

while(1)//Цикл для ввода команды

{

for(i=0;i<13;i++)//Удаляем введенные ранее команды

{

com.par1[i]=0;

com.par2[i]=0;

com.par3[i]=0;

com.name[i]=0;

}

com.name[13]=0;

com.nump=0;

for(i=0;i<54;i++)//Забьем пробелами массив команды

line[i]=32;

cout<<"->";//Печатаем приглашение

cin.get(line,54);//Получаем строку с командой

if(cin.fail())//Если строка - пустая, то очищаем поток и начинаем ввод заново

{

cin.clear();

cin.get();

continue;

}

if(cin.get()!='\n')//Если в потоке остались символы, то превышен размер команды, начинаем ввод заново

{

cout<<"Превышена допустимая длина команды!\n";

while(cin.get()!='\n')

continue;

continue;

}

for(i=0,j=0;i<54;i++)//Если в строке-одни пробелы, то начинаем ввод заново

{

if((line[i]!=32)&&(line[i]!=0))

j++;

}

if(j==0)

{

continue;

}

for(i=0,j=0;i<54;i++)//Если в строке есть русские символы-начинаем ввод заново

{

if(line[i]<0)

{

j++;

}

}

if(j!=0)

{

cout<<"Русские символы не разрешены!\n";

continue;

}

while(line[0]==32)//Если в начале строки- пробелы, то убираем их в конец строки

{

for(i=0;i<53;i++)

line[i]=line[i+1];

line[53]=32;

}

for(i=strlen(line)-1;i>0;i--)//Все лишние пробелы из команды переносим в конец строки

{

if((line[i]==32)&&((line[i+1]==32)||(line[i+1]==0)))

{

for(int j=i;j<53;j++)

line[j]=line[j+1];

line[53]=32;

}

}

com.nump=0;

for(i=strlen(line)-1;i>0;i--)//Считаем количество параметров в команде

{

if(line[i]==32)

{

com.nump++;

}

}

if(com.nump>3)//Если параметров слишком много, то начинаем ввод заново

{

cout<<"Неверное число параметров!\n";

continue;

}

for(i=0;(i<13)&&(line[i]!=32)&&(line[i]!=0);i++)//Считываем имя команды

com.name[i]=line[i];

if((i==13)&&(line[i]!=32)&&(line[i]!=0))//Проверка корректности длины

{

cout<<"Неверное имя команды!\n";

continue;

}

if(com.nump>0)

{

i++;

for(j=0;(j<12)&&(line[i+j]!=32)&&(line[i+j]!=0);j++)//Считываем первый параметр

com.par1[j]=line[i+j];

if((j==12)&&(line[i+j]!=32)&&(line[i+j]!=0))

{

cout<<"Неверный первый параметр!\n";

continue;

}

i+=j;

}

if(com.nump>1)

{

i++;

for(j=0;(j<12)&&(line[i+j]!=32)&&(line[i+j]!=0);j++)//Считываем второй параметр

com.par2[j]=line[i+j];

if((j==12)&&(line[i+j]!=32)&&(line[i+j]!=0))//Проверка корректности длины

{

cout<<"Неверный второй параметр!\n";

continue;

}

i+=j;

}

if(com.nump>2)

{

i++;

for(j=0;(j<12)&&(line[i+j]!=32)&&(line[i+j]!=0);j++)//Считываем третий параметр

com.par3[j]=line[i+j];

if((j==12)&&(line[i+j]!=32)&&(line[i+j]!=0))//Проверка корректности длины

{

cout<<"Неверный третий параметр!\n";

continue;

}

}

break;

}

}

void Monitor::execute()//Выполнение команды

{ int i=0;//Счетчики

int j=0;

int len;//Используется для записи длины параметра

for(i=0;i<28;i++)//Проверяем имя команды

{

if(com.name==commands[i])

break;

}

com.numc=i;

switch(com.numc)

{

case 0://format

if(com.nump!=3)//Если задано неверное число параметров, то вызываем сообщение об ошибке

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

len=strlen(com.par1);

for(i=0;i<len;i++)//Проверяем, нет ли неверных символов в числовом параметре

{

if(!isdigit(com.par1[i]))

{

i=-1;

break;

}

}

if(i==(-1))

{

errflag=1;

err<<"Неверный первый параметр!\n";

com.retcode=10;

break;

}

if(len>5)//Проверяем, не слишком ли длинный числовой параметр

{

errflag=1;

err<<"Неверный первый параметр!\n";

com.retcode=10;

break;

}

com.size=0;

for(i=(len-1);i>=0;i--)//Переводим числовой параметр из текста в число

com.size+=(com.par1[i]-48)\*pow(double(10),double(len-i-1));

if((com.size>65536)||(com.size<9))//Проверяем, корректен ли размер числа

{

errflag=1;

err<<"Неверный первый параметр!\n";

com.retcode=10;

break;

}

com.retcode = SYSTEM.format(com.size,com.par2,com.par3);//Вызываем нужный метод

break;

case 1://content

if(com.nump!=0)//Если задано неверное число параметров, то вызываем сообщение об ошибке

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

if(!isformatted)//Если файловая система не отформатирована, то сообщаем об ошибке

{

errflag=1;

err<<"Файловая система не отформатирована!\n";

com.retcode=10;

break;

}

com.retcode = SYSTEM.content();

break;

case 2://sortcontent

if(com.nump!=0)//Если задано неверное число параметров, то вызываем сообщение об ошибке

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

if(!isformatted)//Если файловая система не отформатирована, то сообщаем об ошибке

{

errflag=1;

err<<"Файловая система не отформатирована!\n";

com.retcode=10;

break;

}

com.retcode = SYSTEM.sortcontent();

break;

case 3://freespace

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров\n!";

com.retcode=10;

break;

}

if(!isformatted)//Если файловая система не отформатирована, то сообщаем об ошибке

{

errflag=1;

err<<"Файловая система не отформатирована\n!";

com.retcode=10;

break;

}

com.retcode = SYSTEM.freespace();

break;

case 4://create

if(com.nump!=3)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

if(!isformatted)//Если файловая система не отформатирована, то сообщаем об ошибке

{

errflag=1;

err<<"Файловая система не отформатирована!\n";

com.retcode=10;

break;

}

if(strlen(com.par1)>6)//Проверяем длину параметров

{

errflag=1;

err<<"Неверный первый параметр!\n";

com.retcode=10;

break;

}

if(strlen(com.par2)>4)

{

errflag=1;

err<<"Неверный второй параметр!\n";

com.retcode=10;

break;

}

len=strlen(com.par3);

for(i=0;i<len;i++)//Проверяем, нет ли неверных символов в числовом параметре

{

if(!isdigit(com.par3[i]))

{

i=-1;

break;

}

}

if(i==(-1))

{

errflag=1;

err<<"Неверный третий параметр!\n";

com.retcode=10;

break;

}

if(len>5)//Проверяем, не слишком ли длинный числовой параметр

{

errflag=1;

err<<"Неверный третий параметр!\n";

com.retcode=10;

break;

}

com.size=0;

for(i=(len-1);i>=0;i--)//Переводим числовой параметр из текста в число

com.size+=(com.par3[i]-48)\*pow(double(10),double(len-i-1));

if((com.size>65468)||(com.size==0))//Проверяем, корректен ли размер числа

{

errflag=1;

err<<"Неверный третий параметр!\n";

com.retcode=10;

break;

}

com.retcode = SYSTEM.create(com.par1,com.par2,com.size);

break;

case 5://del

if(com.nump!=2)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

if(!isformatted)//Если файловая система не отформатирована, то сообщаем об ошибке

{

errflag=1;

err<<"Файловая система не отформатирована!\n";

com.retcode=10;

break;

}

if(strlen(com.par1)>6)//Проверяем длину параметров

{

errflag=1;

err<<"Неверный первый параметр!\n";

com.retcode=10;

break;

}

if(strlen(com.par2)>4)

{

errflag=1;

err<<"Неверный второй параметр!\n";

com.retcode=10;

break;

}

com.retcode = SYSTEM.del(com.par1,com.par2);

break;

case 6://rename

if(com.nump!=3)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

if(!isformatted)//Если файловая система не отформатирована, то сообщаем об ошибке

{

errflag=1;

err<<"Файловая система не отформатирована!\n";

com.retcode=10;

break;

}

if(strlen(com.par1)>6)//Проверяем длину параметров

{

errflag=1;

err<<"Неверный первый параметр!\n";

com.retcode=10;

break;

}

if(strlen(com.par2)>4)

{

errflag=1;

err<<"Неверный второй параметр!\n";

com.retcode=10;

break;

}

if(strlen(com.par3)>6)

{

errflag=1;

err<<"Неверный третий параметр!\n";

com.retcode=10;

break;

}

com.retcode = SYSTEM.rename(com.par1,com.par2,com.par3);

break;

case 7://append

if(com.nump!=3)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

if(!isformatted)//Если файловая система не отформатирована, то сообщаем об ошибке

{

errflag=1;

err<<"Файловая система не отформатирована!\n";

com.retcode=10;

break;

}

if(strlen(com.par1)>6)//Проверяем длину параметров

{

errflag=1;

err<<"Неверный первый параметр!\n";

com.retcode=10;

break;

}

if(strlen(com.par2)>4)

{

errflag=1;

err<<"Неверный второй параметр!\n";

com.retcode=10;

break;

}

len=strlen(com.par3);

for(i=0;i<len;i++)//Проверяем, нет ли неверных символов в числовом параметре

{

if(!isdigit(com.par3[i]))

{

i=-1;

break;

}

}

if(i==(-1))

{

errflag=1;

err<<"Неверный третий параметр!\n";

com.retcode=10;

break;

}

if(len>5)//Проверяем, не слишком ли длинный числовой параметр

{

errflag=1;

err<<"Неверный третий параметр!\n";

com.retcode=10;

break;

}

com.size=0;

for(i=(len-1);i>=0;i--)//Переводим числовой параметр из текста в число

com.size+=(com.par3[i]-48)\*pow(double(10),double(len-i-1));

if((com.size>65468)||(com.size==0))//Проверяем, корректен ли размер числа

{

errflag=1;

err<<"Неверный третий параметр!\n";

com.retcode=10;

break;

}

com.retcode = SYSTEM.append(com.par1,com.par2,com.size);

break;

case 8://changetype

if(com.nump!=3)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

if(!isformatted)//Если файловая система не отформатирована, то сообщаем об ошибке

{

errflag=1;

err<<"Файловая система не отформатирована!\n";

com.retcode=10;

break;

}

if(strlen(com.par1)>6)//Проверяем длину параметров

{

errflag=1;

err<<"Неверный первый параметр!\n";

com.retcode=10;

break;

}

if(strlen(com.par2)>4)

{

errflag=1;

err<<"Неверный второй параметр!\n";

com.retcode=10;

break;

}

if(strlen(com.par3)>4)

{

errflag=1;

err<<"Неверный третий параметр!\n";

com.retcode=10;

break;

}

com.retcode = SYSTEM.changetype(com.par1,com.par2,com.par3);

break;

case 9://defrag

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

if(!isformatted)//Если файловая система не отформатирована, то сообщаем об ошибке

{

errflag=1;

err<<"Файловая система не отформатирована!\n";

com.retcode=10;

break;

}

com.retcode = SYSTEM.defrag();

break;

case 10://userandlabel

if(com.nump!=2)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

if(!isformatted)//Если файловая система не отформатирована, то сообщаем об ошибке

{

errflag=1;

err<<"Файловая система не отформатирована!\n";

com.retcode=10;

break;

}

com.retcode = SYSTEM.userandlabel(com.par1,com.par2);

break;

case 11://createFS

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

if(!isformatted)//Если файловая система не отформатирована, то сообщаем об ошибке

{

errflag=1;

err<<"Файловая система не отформатирована!\n";

com.retcode=10;

break;

}

com.retcode = SYSTEM.createFS();

break;

case 12://help

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"Команды в системе имеют следующий формат:\nимя\_команды параметр\_1 параметр\_2 параметр\_3\nКоличество параметров у разных команд может быть от 0 до 3\nПри задании параметров не допускается использование символов русского алфавита\n";

cout<<"В системе имеются следующие команды:\n";

cout<<"format размер\_системы метка\_тома имя\_владельца\n";

cout<<"content\n";

cout<<"sortcontent\n";

cout<<"freespace\n";

cout<<"create имя\_файла тип\_файла размер\_файла\n";

cout<<"del имя\_файла тип\_файла\n";

cout<<"rename старое\_имя\_файла тип\_файла новое\_имя\_файла\n";

cout<<"append имя\_файла тип\_файла добавляемый\_размер\n";

cout<<"changetype имя\_файла старый\_тип\_файла новый\_тип\_файла\n";

cout<<"defrag\n";

cout<<"userandlabel имя\_владельца метка\_тома\n";

cout<<"createFS\n";

cout<<"help\n";

cout<<"exit\n";

cout<<"Для получения информации о команде введите её имя с вопросительным знаком\n в конце, например format?\n";

com.retcode=0;

}

catch(...)

{

com.retcode=1;

}

break;

case 13://exit

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

exitflag=1;

com.retcode=0;

}

catch(...)

{

com.retcode=1;

}

break;

case 14://format?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"format размер\_системы метка\_тома имя\_владельца\n";

cout<<"Данная команда создает фаловую систему заданного размера, устанавливает её \nметку тома и имя владельца\n";

cout<<"Размер задается в блоках, он должен быть больше 9-ти и не должен превышать 65536-ти\n";

cout<<"Имя владельца и метка тома не должны быть длиннее 12-ти символов\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 15://content?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"content\n";

cout<<"Данная команда выводит перечень файлов в системе в том порядке, в котором \nони там находятся\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 16://sortcontent?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"sortcontent\n";

cout<<"Данная команда выводит перечень файлов в системе в алфавитном порядке\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 17://freespace?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"freespace\n";

cout<<"Данная команда выводит перечень пустых областей в системе\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 18://create?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"create имя\_файла тип\_файла размер\_файла\n";

cout<<"Данная команда создает в системе файл с заданными именем и типом, \nимеющий заданный размер\n";

cout<<"Имя создаваемого файла не должно быть длиннее 6-ти символов, тип - 4-х\n";

cout<<"Размер должен быть ненулевым и не должен превышать 65468-ми\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 19://del?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"del имя\_файла тип\_файла\n";

cout<<"Данная команда удаляет из системы файл с заданными именем и типом\n";

cout<<"Имя файла не должно быть длиннее 6-ти символов, тип - 4-х\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 20://rename?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"rename старое\_имя\_файла тип\_файла новое\_имя\_файла\n";

cout<<"Данная команда меняет имя файла с заданными именем и типом на новое\n";

cout<<"Имя файла не должно быть длиннее 6-ти символов, тип - 4-х\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 21://append?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"append имя\_файла тип\_файла добавляемый\_размер\n";

cout<<"Данная команда увеличивает размер файла с заданными именем и типом на \nуказанную величину\n";

cout<<"Имя файла не должно быть длиннее 6-ти символов, тип - 4-х\n";

cout<<"Размер должен быть ненулевым и не должен превышать 65468-ми\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 22://changetype?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"changetype имя\_файла старый\_тип\_файла новый\_тип\_файла\n";

cout<<"Данная команда меняет тип файла с заданными именем и типом на новый\n";

cout<<"Имя файла не должно быть длиннее 6-ти символов, тип - 4-х\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 23://defrag?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"defrag\n";

cout<<"Данная команда дефрагментирует файловую систему\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 24://userandlabel?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"userandlabel имя\_владельца метка\_тома\n";

cout<<"Данная команда меняет имя владельца и метку тома в системе на новые\n";

cout<<"Имя владельца и имя тома не должны быть длиннее 12-ти символов\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 25://createFS?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"createFS\n";

cout<<"Данная команда создает на жестком диске файл filesystem.txt со структурой,\n соответсвующей созданной файловой системе\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 26://help?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"help\n";

cout<<"Данная команда выводит список имеющихся команд и их параметров\n";

}

catch(...)

{

com.retcode=1;

}

break;

case 27://exit?

try{

if(com.nump!=0)

{

errflag=1;

err<<"Неверное число параметров!\n";

com.retcode=10;

break;

}

cout<<"exit\n";

cout<<"Данная команда осуществляет выход из приложения\n";

}

catch(...)

{

com.retcode=1;

}

break;

default://Команда не найдена

errflag=1;

err<<"Неверное имя команды!\n";

com.retcode=10;

break;

}

}

void Monitor::errcheck()

{ string buf("");

switch(com.numc)

{

case 0://format

switch(com.retcode)

{

case 0://Успешное выполнение

cout<<"Файловая система отформатирована\n";

isformatted=1;

break;

case 1://Непредвиденная ошибка

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 10://Ошибка пользователя, перехваченная монитором

errflag=0;

buf=err.str();

cout<<buf;

break;

}

break;

case 1://content

switch(com.retcode)

{

case 0://Успешное выполнение, вывод оглавления

cout<<"Содержимое файловой системы:\n";

if(screenflag)

{

screenflag=0;

buf=screen.str();

cout<<buf;

}

break;

case 1://Непредвиденная ошибка

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 2://Нет файлов в системе

cout<<"Файлы в системе отсутствуют!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 10://Ошибка пользователя, перехваченная монитором

errflag=0;

buf=err.str();

cout<<buf;

break;

}

break;

case 2://sortcontent

switch(com.retcode)

{

case 0://Успешное выполнение, вывод отсортированного оглавления

cout<<"Содержимое файловой системы:\n";

if(screenflag)

{

screenflag=0;

buf=screen.str();

cout<<buf;

}

break;

case 1://Непредвиденная ошибка

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 2://Нет файлов в системе

cout<<"Файлы в системе отсутствуют!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 10://Ошибка пользователя, перехваченная монитором

errflag=0;

buf=err.str();

cout<<buf;

break;

}

break;

case 3://freespace

switch(com.retcode)

{

case 0://Успешное выполнение, вывод мнформации о свободных областях

cout<<"Информация о свободных областях:\n";

if(screenflag)

{

screenflag=0;

buf=screen.str();

cout<<buf;

}

break;

case 1://Непредвиденная ошибка

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 2://Нет файлов в системе

cout<<"Свободные области в системе отсутствуют!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 10://Ошибка пользователя, перехваченная монитором

errflag=0;

buf=err.str();

cout<<buf;

break;

}

break;

case 4://create

switch(com.retcode)

{

case 0://Успешное выполнение

cout<<"Файл "<<com.par1<<'.'<<com.par2<<" с размером "<<com.size<<" успешно создан\n";

if(screenflag)

{

screenflag=0;

buf=screen.str();

cout<<buf;

}

break;

case 1://Непредвиденная ошибка

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 2://Нет файлов в системе

cout<<"Недостаточно места в системе!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 3://Нет файлов в системе

cout<<"Недостаточно места в каталоге!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 4://Нет файлов в системе

cout<<"Такой файл уже существует!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 10://Ошибка пользователя, перехваченная монитором

errflag=0;

buf=err.str();

cout<<buf;

break;

}

break;

case 5://del

switch(com.retcode)

{

case 0://Успешное выполнение, файл удален

cout<<"Файл "<<com.par1<<'.'<<com.par2<<" удален\n";

if(screenflag)

{

screenflag=0;

buf=screen.str();

cout<<buf;

}

break;

case 1://Непредвиденная ошибка

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 2://Нет такого файла

cout<<"Такого файла нет в системе!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 10://Ошибка пользователя, перехваченная монитором

errflag=0;

buf=err.str();

cout<<buf;

break;

}

break;

case 6://rename

switch(com.retcode)

{

case 0://Успешное выполнение, файл переименован

cout<<"Файл "<<com.par1<<'.'<<com.par2<<" преименован в "<<com.par3<<'.'<<com.par2<<'\n';

if(screenflag)

{

screenflag=0;

buf=screen.str();

cout<<buf;

}

break;

case 1://Непредвиденная ошибка

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 2://Нет такого файла

cout<<"Такого файла нет в системе!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 10://Ошибка пользователя, перехваченная монитором

errflag=0;

buf=err.str();

cout<<buf;

break;

}

break;

case 7://append

switch(com.retcode)

{

case 0://Успешное выполнение

cout<<"Размер файла "<<com.par1<<'.'<<com.par2<<" увеличен на "<<com.size<<'\n';

if(screenflag)

{

screenflag=0;

buf=screen.str();

cout<<buf;

}

break;

case 1://Непредвиденная ошибка

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 2://Нет файлов в системе

cout<<"Такого файла нет в системе!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 3://Нет файлов в системе

cout<<"Недостаточно места в системе!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 10://Ошибка пользователя, перехваченная монитором

errflag=0;

buf=err.str();

cout<<buf;

break;

}

break;

case 8://changetype

switch(com.retcode)

{

case 0://Успешное выполнение, файал переименован

cout<<"Тип файла "<<com.par1<<'.'<<com.par2<<" изменен на "<<'.'<<com.par3<<'\n';

if(screenflag)

{

screenflag=0;

buf=screen.str();

cout<<buf;

}

break;

case 1://Непредвиденная ошибка

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 2://Нет такого файла

cout<<"Такого файла нет в системе!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 10://Ошибка пользователя, перехваченная монитором

errflag=0;

buf=err.str();

cout<<buf;

break;

}

break;

case 9://defrag

switch(com.retcode)

{

case 0://Успешное выполнение

cout<<"Файловая система сжата\n";

if(screenflag)

{

screenflag=0;

buf=screen.str();

cout<<buf;

}

break;

case 1://Непредвиденная ошибка

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 10://Ошибка пользователя, перехваченная монитором

errflag=0;

buf=err.str();

cout<<buf;

break;

}

break;

case 10://userandlabel

switch(com.retcode)

{

case 0://Успешное выполнение

cout<<"Имя владельца и метка тома изменены на "<<com.par1<<" и "<<com.par2<<'\n';

if(screenflag)

{

screenflag=0;

buf=screen.str();

cout<<buf;

}

break;

case 1://Непредвиденная ошибка

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 10://Ошибка пользователя, перехваченная монитором

errflag=0;

buf=err.str();

cout<<buf;

break;

}

break;

case 11://createFS

switch(com.retcode)

{

case 0://Успешное выполнение

cout<<"Файл filesystem.txt успешно создан\n";

if(screenflag)

{

screenflag=0;

buf=screen.str();

cout<<buf;

}

break;

case 1://Непредвиденная ошибка

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 2://Ошибка записи

cout<<"Не удалось записать файл на жесткий диск!\n";

if(errflag)

{

errflag=0;

buf=err.str();

cout<<buf;

}

break;

case 10://Ошибка пользователя, перехваченная монитором

errflag=0;

buf=err.str();

cout<<buf;

break;

}

break;

case 12://help

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 13://exit

switch(com.retcode)

{

case 0:

cout<<"Завершение работы\n";

break;

case 1:

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

}

break;

case 14://format?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 15://content?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 16://sortcontent?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 17://freespace?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 18://create?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 19://del?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 20://rename?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 21://append?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 22://changetype?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 23://defrag?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 24://userandlabel?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 25://createFS?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 26://help?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

case 27://exit?

if(com.retcode==1)

cout<<"Во время выполнения команды произошла непредвиденная ошибка!\n";

break;

default://Команда не найдена

errflag=0;

buf=err.str();

cout<<buf;

break;

}

screen.str("");//Очищаем буферы

err.str("");

cout<<'\n';

for(int i=0;i<13;i++)//Удаляем команду

{

com.par1[i]=0;

com.par2[i]=0;

com.par3[i]=0;

com.name[i]=0;

}

com.name[13]=0;

com.numc=0;

com.nump=0;

com.size=0;

com.retcode=0;

}

# **3.3. Содержимое файла FS.h – описание класса файловой системы и его методов.**

using namespace std;

std::stringstream screen;//буфер для вывода сообщений

std::stringstream err;//буфер для вывода сообщений об ошибках

bool screenflag, errflag;//флаги использования буферов

bool isformatted;//флаг наличия отформатированной системы

bool exitflag=0;//флаг выхода

struct SB{ //системный блок

unsigned short vers; //версия системы

char tomename[13];//метка тома

char ownername[13];//имя владельца

string sysname;//имя системы

};

struct FRec{ //запись о файле

unsigned short rectype; //тип записи

char name[7];//имя файла

char type[5]; //тип файла

unsigned short len; //число блоков, занимаемых файлом

unsigned short date;//дата создания файла

};

struct segment{//сегмент

unsigned short catsize;//размер каталога

unsigned short nextseg;//номер следующего сегмента

unsigned short seginuse;//счетчик используемых сегментов

unsigned short startblock;//номер блока на носителе, с которого размещаются файлы, описанные в сегменте

FRec files[63];//таблица записей о файлах

};

class FS{//файловая система

public:

SB sb;//системный блок

segment \*catalog;//каталог-массив сегментов

FS();

int format(unsigned long size, char\* NT, char\* NO);

int content();

int sortcontent();

int freespace();

int create(char\* name, char\* type, unsigned short size);

int del(char\* name, char\* type);

int rename(char\* oldname, char\* type,char\* newname);

int append(char\* name, char\* type,unsigned short size);

int changetype(char\* name, char\* oldtype,char\* newtype);

int defrag();

int userandlabel(char\* owner,char\* label);

int createFS();

~FS();

};

FS::FS(){//Пустой конструктор

sb.vers=1;

sb.sysname="K5-291";

for(int i=0;i<12;i++)

{ sb.tomename[i]=0;

sb.ownername[i]=0;

}

catalog=NULL;

}

int FS::format(unsigned long size, char\* NT, char\* NO) {

try{

int i=0;

for (i=0; NT[i]; i++)//записывается информация в системный блок

sb.tomename[i]=NT[i];

for (; i<12; i++)

sb.tomename[i]=0;

for (i=0; NO[i]; i++)

if (NO[i]!='\0') sb.ownername[i]=NO[i];

for (; i<12; i++)

sb.ownername[i]=0;

unsigned long CS=size/62+1;//расчитывается...

unsigned char cs=CS;

if (cs>31) cs=31;

if (cs==0) cs=1;

catalog= new segment[cs];//...и выделяется память под каталог

for (unsigned char i=0; i<cs; i++){

catalog[i].catsize=cs;//заполняются заголовки сегментов

catalog[i].nextseg=i+1;

catalog[i].seginuse=1;

catalog[i].startblock=0;

catalog[i].files[0].rectype=5;//в сегментах создаются записи-признаки конца записей(аналог символа '0', указывает, что нужно прекратить чтение из сегмента)

for(int j=0;j<7;j++)

catalog[i].files[0].name[j]=0;//поля записи о файле инициализируются нулями

for(int j=0;j<5;j++)

catalog[i].files[0].type[j]=0;

catalog[i].files[0].len=0;

catalog[i].files[0].date=0;

}

catalog[0].startblock=6+2\*cs;// в первом сегменте редактируется заголовок,...

catalog[0].files[1]=catalog[0].files[0];//...запись-признак конца записей сдвигается вниз,...

catalog[0].files[0].rectype=2;//...на первое место добавляется запись о нераспределенной памяти...

catalog[0].files[0].len=size-(6+2\*cs);// ...размером, соответствующим размеру оставшейся памяти...

for(int j=0;j<7;j++)

catalog[0].files[0].name[j]=0;//...поля записи инициализируются нулями...

for(int j=0;j<5;j++)

catalog[0].files[0].type[j]=0;

catalog[0].files[0].date=0;

catalog[cs-1].nextseg=0;//в последнем сегменте каталога указывается, что после него нет сегметов

}

catch(...)

{

return 1;

}

return 0;

}

int FS::content()

{try

{

int cs;//размер каталога

int i,j;//счетчики

bool flag=0;//флаг

cs=catalog[0].catsize;

for(i=0;i<cs;i++)//проходим по всем сегментам каталога

{

for(j=0;catalog[i].files[j].rectype!=5;j++)//проходим по всем записям сегмента до конца записей

{

if(catalog[i].files[j].rectype==1)//для всех найденных записей о файлах выводим их имя и тип

{

if(!screenflag)

screenflag=1;//устанавливаем флаг задействования буфера вывода

screen<<catalog[i].files[j].name<<'.'<<catalog[i].files[j].type<<'\n';

if(!flag)//если нашли хоть один файл-устанавливаем флаг

flag=1;

}

}

}

if(flag)//если флаг установлен, то возвращаем код успешного завершения, если нет-то ошибки

return 0;

else

return 2;

}

catch(...)

{

return 1;//если перехватили непредвиденную ошибку-возвращаем код ошибки

}

}

int FS::sortcontent()

{ try

{ int retcode = content();//вызываем функцию вывода оглавления, она записывает оглавление в глобальный буфер screen

string buf("");//буфер для содержимого

string bufitem("");//буфер для записи

int oldpos=0;//переменные для перемещения по строке buf

int newpos=0;

struct list//структура для списка записей

{string item;//запись

list\* next;//указатель на следующую запись в списке

};

list\* head=0;//голова списка

list\* curr=0;//текущий элемент списка

list\* newitem=0;//новый элемент списка

switch(retcode)

{

case 0:

if(screenflag)//считываем содержимое глобального буфера screen в строку buf

{

screenflag=0;

buf=screen.str();

screen.str("");//обнуляем глобальный буфер

}

while(newpos+1<buf.size())//пока не достигнем конца строки buf

{

newpos=buf.find('\n',oldpos);//находим символ перехода на новую строку, т.е. конец текущей записи

bufitem=buf.substr(oldpos,newpos-oldpos+1);//считываем символы от конца предыдущей записи до конца текущей

oldpos=newpos+1;//назначаем номер символа конца текущей записи номером конца предыдущей записи

newitem = new list;//создаем новый элемент списка и записываем в него считанную запись

newitem->item=bufitem;

newitem->next=0;

if(head==0)//если список пуст, то назначем новый элемент головой списка

head=newitem;

else if(head->item>newitem->item)//если по алфавиту запись в новом элементе меньше, чем запись в голове списка, то назначаем новый элемент головой

{

newitem->next=head;

head=newitem;

}

else//ищем место по алфавиту для нового элемента в списке и ставим его в список

{ curr=head;

while(curr->next!=0)

{

if((curr->item<newitem->item)&&(curr->next->item>newitem->item))

{

newitem->next=curr->next;

curr->next=newitem;

break;

}

curr=curr->next;

}

if(curr->next==0)//если записи всех элементов в списке по алфавиту меньше записи в новом элементе, то ставим элемент в конец списка

curr->next=newitem;

}

}

curr=head;

while(curr->next!=0)//выводим в глобальный буфер список файлов в алфавитном порядке

{

screen<<curr->item;

curr=curr->next;

}

screenflag=1;

screen<<curr->item;

curr=head;

while(curr->next!=0)//очищаем память

{

newitem=curr;

curr=curr->next;

delete newitem;

}

delete curr;

return 0;

break;

case 1:

return 1;//если content вернула код ошибки, то тоже возвращаем его же

break;

case 2:

return 2;

break;

}

return 1;//если функция выполнилась до сюда и ничего до сих пор не вернула, значит в ней точно есть ошибка

}

catch(...)

{

return 1;//если перехватили ошибку, то возвращаем код ошибки

}

}

int FS::freespace()

{

try

{

int cs;//размер каталога

int i,j;//счетчики

bool flag=0;//флаг

cs=catalog[0].catsize;

for(i=0;i<cs;i++)//проходим по всем сегментам каталога

{

for(j=0;catalog[i].files[j].rectype!=5;j++)//проходим по всем записям сегмента до конца записей

{

if(catalog[i].files[j].rectype==2)//для всех найденных записей свободных областях выводим их размер

{

if(!screenflag)

screenflag=1;//устанавливаем флаг задействования буфера вывода

screen<<catalog[i].files[j].name<<'.'<<catalog[i].files[j].type<<'\n';

if(!flag)//если нашли хоть одну свободную область-устанавливаем флаг

flag=1;

}

}

}

if(flag)//если флаг установлен, то возвращаем код успешного завершения, если нет-то ошибки

return 0;

else

return 2;

}

catch(...)

{

return 1;//если перехватили непредвиденную ошибку-возвращаем код ошибки

}

}

int FS::create(char\* name, char\* type, unsigned short size)

{

try

{

int cs=catalog[0].catsize;//размер каталога

int i,j,k;//счетчики

bool flag=0;//флаг

int currseg, currfile;//номера сегмента и записи, где будет создана новая запись

int lastseg;//номер последнего задействованного сегмента

int lastfile;//номер последей записи в сегменте

time\_t rawtime;//переменные для работы с датой

tm \* timeinfo;

int year,day,month;

time ( &rawtime );//получаем дату

timeinfo = localtime ( &rawtime );

year = timeinfo->tm\_year-80;

month = timeinfo->tm\_mon;

day = timeinfo->tm\_mday;

for(i=0;i<cs;i++)//проходим по всем сегментам каталога

{

for(j=0;catalog[i].files[j].rectype!=5;j++)//просматриваем записи о файлах, пока не достигнем записи-признака конца записей

{

if((!strcmp(name,catalog[i].files[j].name))&&(!strcmp(type,catalog[i].files[j].type)))//если находим запись о файле с такими именем и типом, то возвращаем код ошибки

{

return 4;

}

}

}

for(i=0;i<cs;i++)//проходим по всем сегментам каталога

{

for(j=0;catalog[i].files[j].rectype!=5;j++)//просматриваем записи о файлах, пока не достигнем записи-признака конца записей

{

if((catalog[i].files[j].rectype==2)&&(catalog[i].files[j].len>=size))//ищем свободную область подходящего размера

{

flag=1;

break;

}

}

if(flag==1)

break;

}

if(flag==0)

return 2;//если не нашли подходящей области-возвращаем код ошибки

currseg=i;//записываем номера найденных сегмента и записи о свободной области

currfile=j;

if(catalog[currseg].files[currfile].len==size)//если размер найденной области в точности совпадает с размером файла, то записываем файл на её место

{

catalog[currseg].files[currfile].rectype=1;

strcpy(catalog[currseg].files[currfile].name,name);

strcpy(catalog[currseg].files[currfile].type,type);

catalog[currseg].files[currfile].date=( month + 32\*day + 1024\*year);//так записываем дату

return 0;

}

//сдвигаем записи в каталоге на 1 позицию от найденных:

lastseg=catalog[0].seginuse-1;

for(j=0;catalog[lastseg].files[j].rectype!=5;j++)//находим последнюю запись в последнем задействованном сегменте

continue;

lastfile=j;

if((lastfile==62)&&(lastseg==cs))//если каталог заполнен, то возвращаем код ошибки

return 3;

if(currseg!=lastseg)//если запись о файле заносится не в последний сегмент

{if(lastfile==62)//если этот сегмент заполнен, но ещё есть свободные, то задействуем новый сегмент и переносим в него последнюю запись из предыдущего

{

for(i=0;i<cs;i++)

catalog[i].seginuse++;//увеличиваем число задействованных сегментов

catalog[lastseg+1].files[1]=catalog[lastseg+1].files[0];//переносим в новый сегмент последнюю запись из предыдущего

catalog[lastseg+1].files[0]=catalog[lastseg].files[61];

catalog[lastseg].files[61]=catalog[lastseg].files[62];//сдвигаем признак конца записей назад на 1 позицию (необходимо для применения нижеследующего алгоритма)

lastfile--;

}

for(j=lastfile;j>=0;j--)

catalog[lastseg].files[j+1]=catalog[lastseg].files[j];//сдвигаем файлы в последнем сегменте

for(i=lastseg-1;i>currseg;i--)//в сегментах, промежуточных между тем, в который записывается файл и последним задействованным сегментом сдвигаем записи на 1 вперед

{

catalog[i+1].files[0]=catalog[i].files[61];

for(j=60;j>=0;j++)

catalog[i].files[j+1]=catalog[i].files[j];

}

}

for(j=0;catalog[currseg].files[j].rectype!=5;j++)//в том сегменте, куда записывается файл, ищем конец записей

continue;

lastfile=j;

if(lastfile==62)//если сегмент заполнен, то переносим последнню запись в следующий и сдвигаем остальные записи на 1 вниз

{

if(currseg==lastseg)//если редактируемый сегмент-последний

{

for(i=0;i<cs;i++)

catalog[i].seginuse++;//увеличиваем число задействованных сегментов

catalog[currseg+1].files[1]=catalog[currseg+1].files[0];//сдвигаем в новом последнем сегменте признак конца записей

}

catalog[currseg+1].files[0]=catalog[currseg].files[61];//переносим из редактируемого сегмента последнюю запись в следующий сегмент

for(j=60;j>currfile;j--)

catalog[currseg].files[j+1]=catalog[currseg].files[j];

}

else

{

for(j=lastfile;j>currfile;j--)

catalog[currseg].files[j+1]=catalog[currseg].files[j];

}

//делаем запись о файле в свободной области и запись об оставшемся свободном месте следом за ней

catalog[currseg].files[currfile+1].rectype=2;

catalog[currseg].files[currfile+1].len=catalog[currseg].files[currfile].len-size;

for(k=0;k<7;k++)

catalog[currseg].files[currfile+1].name[k]=0;

for(k=0;k<5;k++)

catalog[currseg].files[currfile+1].type[k]=0;

catalog[currseg].files[currfile+1].date=0;

catalog[currseg].files[currfile].rectype=1;

catalog[currseg].files[currfile].len=size;

strcpy(catalog[currseg].files[currfile].name,name);

strcpy(catalog[currseg].files[currfile].type,type);

catalog[currseg].files[currfile].date=( month + 32\*day + 1024\*year);

for(i=currseg;i<catalog[0].seginuse-1;i++)//корректируем номера блоков начала размещения файлов в сегментах

{k=catalog[i].startblock;

for(j=0;catalog[i].files[j].rectype!=5;j++)

k+=catalog[i].files[j].len;

catalog[i+1].startblock=k;

}

return 0;

}

catch(...)

{

return 1;//если перехватываем необработанное прерывание-возвращаем код ошибки

}

}

int FS::del(char\* name, char\* type)

{

try

{

int cs=catalog[0].catsize;//размер каталога

int i,j;//счетчики

for(i=0;i<cs;i++)//проходим по всем сегментам каталога

{

for(j=0;catalog[i].files[j].rectype!=5;j++)//просматриваем записи о файлах, пока не достигнем записи-признака конца записей

{

if((!strcmp(name,catalog[i].files[j].name))&&(!strcmp(type,catalog[i].files[j].type)))//если находим нужный файл, то:

{

catalog[i].files[j].rectype=2;//меняем тип записи на запись о свободной области

for(int k=0;k<7;k++)

catalog[i].files[j].name[k]=0;//поля записи о файле обнуляем

for(int k=0;k<5;k++)

catalog[i].files[j].type[k]=0;

catalog[i].files[j].date=0;

return 0;

}

}

}

return 2;//если не нашли запись с такими именем и типом, то возвращаем код ошибки

}

catch(...)

{

return 1;//если перехватываем необработанное прерывание-возвращаем код ошибки

}

}

int FS::rename(char\* oldname, char\* type,char\* newname)

{ try

{

int i,j;//счетчики

int cs=catalog[0].catsize;//размер каталога

bool flag=0;

int segn,recn;//номера сегмента и записи о файле для искомого файла

for(i=0;i<cs;i++)//проходим по всем сегментам каталога

{

for(j=0;catalog[i].files[j].rectype!=5;j++)//проходим по всем записям сегмента до конца записей

{

if((!strcmp(oldname,catalog[i].files[j].name))&&(!strcmp(type,catalog[i].files[j].type)))//если находим запись о файле, который переименовываем-уcтанавливам флаг и записываем номера сегмента и записи о файле

{

flag=1;

segn=i;

recn=j;

break;

}

}

if(flag)

break;

}

if(!flag)//если флаг не установлен-возвращаем код ошибки

return 2;

for(i=0;i<cs;i++)//проходим по всем сегментам каталога

{

for(j=0;catalog[i].files[j].rectype!=5;j++)//просматриваем записи о файлах, пока не достигнем записи-признака конца записей

{

if((!strcmp(newname,catalog[i].files[j].name))&&(!strcmp(type,catalog[i].files[j].type)))//если находим запись о файле с новыми именем и типом, то возвращаем код ошибки

{

return 3;

}

}

}

//изменяем имя файла

for(i=0;newname[i]!=0;i++)

catalog[segn].files[recn].name[i]=newname[i];

for(;i<7;i++)

catalog[segn].files[recn].name[i]=0;

return 0;

}

catch(...)

{

return 1;//если перехватили ошибку, то возвращаем код ошибки

}

}

int FS::append(char\* name, char\* type,unsigned short size)

{

try

{

int i,j;//счетчики

int cs=catalog[0].catsize;//размер каталога

bool flag=0;

int segn,recn;//номера сегмента и записи о файле для искомого файла

int retcode;//возвращаемый функцией код

int newfilesize;//новый размер файла

FRec bufrec;//буферная запись о файле

for(i=0;i<cs;i++)//проходим по всем сегментам каталога

{

for(j=0;catalog[i].files[j].rectype!=5;j++)//проходим по всем записям сегмента до конца записей

{

if((!strcmp(name,catalog[i].files[j].name))&&(!strcmp(type,catalog[i].files[j].type)))//если находим запись о файле, размер которого меняем-уcтанавливам флаг и записываем номера сегмента и записи о файле

{

flag=1;

segn=i;

recn=j;

break;

}

}

if(flag)

break;

}

if(!flag)//если флаг не установлен-возвращаем код ошибки

return 2;

//если рядом с файлом лежит свободная область подходящего размера, то просто берем из неё свободное место

if((recn!=62)&&(catalog[segn].files[recn+1].rectype==2)&&(catalog[segn].files[recn+1].len>size))

{ catalog[segn].files[recn+1].len-=size;

catalog[segn].files[recn].len+=size;

return 0;

}

if((recn!=0)&&(catalog[segn].files[recn-1].rectype==2)&&(catalog[segn].files[recn-1].len>size))

{ catalog[segn].files[recn-1].len-=size;

catalog[segn].files[recn].len+=size;

return 0;

}

newfilesize=catalog[segn].files[recn].len+size;//запоминаем новый размер файла

bufrec=catalog[segn].files[recn];//запоминаем старую запись о файле

retcode=del(name,type);//удаляем старую запись о файле

switch(retcode)

{ case 0:

retcode=create(name,type,newfilesize);//создаем новую запись о файле с новым размером

switch(retcode)//обрабатываем возвращенный код

{

case 0:

return 0;

break;

case 1:

catalog[segn].files[recn]=bufrec;//если вдруг не удалось создать новую запись, то восстанавливаем старую

return 1;

break;

case 2:

catalog[segn].files[recn]=bufrec;

return 3;

break;

case 3:

catalog[segn].files[recn]=bufrec;

return 4;

break;

case 4:

catalog[segn].files[recn]=bufrec;

return 1;

break;

}

break;

case 1:

return 1;

break;

case 2:

return 1;

break;

}

return 1;

}

catch(...)

{

return 1;

}

}

int FS::changetype(char\* name, char\* oldtype,char\* newtype)

{try

{

int i,j;//счетчики

int cs=catalog[0].catsize;//размер каталога

bool flag=0;

int segn,recn;//номера сегмента и записи о файле для искомого файла

for(i=0;i<cs;i++)//проходим по всем сегментам каталога

{

for(j=0;catalog[i].files[j].rectype!=5;j++)//проходим по всем записям сегмента до конца записей

{

if((!strcmp(name,catalog[i].files[j].name))&&(!strcmp(oldtype,catalog[i].files[j].type)))//если находим запись о файле, который переименовываем-уcтанавливам флаг и записываем номера сегмента и записи о файле

{

flag=1;

segn=i;

recn=j;

break;

}

}

if(flag)

break;

}

if(!flag)//если флаг не установлен-возвращаем код ошибки

return 2;

for(i=0;i<cs;i++)//проходим по всем сегментам каталога

{

for(j=0;catalog[i].files[j].rectype!=5;j++)//просматриваем записи о файлах, пока не достигнем записи-признака конца записей

{

if((!strcmp(name,catalog[i].files[j].name))&&(!strcmp(newtype,catalog[i].files[j].type)))//если находим запись о файле с новыми именем и типом, то возвращаем код ошибки

{

return 3;

}

}

}

//изменяем тип файла

for(i=0;newtype[i]!=0;i++)

catalog[segn].files[recn].type[i]=newtype[i];

for(;i<5;i++)

catalog[segn].files[recn].type[i]=0;

return 0;

}

catch(...)

{

return 1;//если перехватили ошибку, то возвращаем код ошибки

}

}

int FS::defrag()

{

try

{ int i,j,k;//счетчики

int cs=catalog[0].catsize;//размер каталога

int filecount=0;//счетчтик записей о файлах

int segcount=0;//нужное количество сегментов

int freecount=0;//счетчик свободного места

int written=0;//счетчик записанных в новый каталог файлов

segment\* newcatalog=0;//указатель на новый каталог

for(i=0;i<cs;i++)//проходим по всем сегментам каталога

{

for(j=0;catalog[i].files[j].rectype!=5;j++)//просматриваем записи о файлах, пока не достигнем записи-признака конца записей

{

if(catalog[i].files[j].rectype==1)//считаем количество записей о файлах

filecount++;

if(catalog[i].files[j].rectype==2)//считаем количество свободного пространства в системе

freecount+=catalog[i].files[j].len;

}

}

if((filecount+1)%62==0)//считаем число задействованных сегментов в новом каталоге

segcount=(filecount+1)/62;

else

segcount=(filecount+1)/62+1;

newcatalog= new segment[cs];//выделяем память под новый каталог такого же размера, как и старый

for(i=0;i<cs;i++)//проходим по всем сегментам каталога

{

for(j=0;catalog[i].files[j].rectype!=5;j++)//просматриваем записи о файлах, пока не достигнем записи-признака конца записей

{

if(catalog[i].files[j].rectype==1)//переписываем записи о файлах в новый каталог

{

newcatalog[written/62].files[written%62]=catalog[i].files[j];

written++;

}

}

}

//Добавляем в последний задействованный сегмент нового каталога запись о нераспределенной памяти и признак конца записей

newcatalog[written/62].files[written%62].rectype=2;

newcatalog[written/62].files[written%62].len=freecount;

newcatalog[written/62].files[written%62].date=0;

for( k=0;k<7;k++)

newcatalog[written/62].files[written%62].name[k]=0;

for( k=0;k<5;k++)

newcatalog[written/62].files[written%62].type[k]=0;

newcatalog[written/62].files[written%62+1].rectype=5;

newcatalog[written/62].files[written%62+1].len=0;

newcatalog[written/62].files[written%62+1].date=0;

for( k=0;k<7;k++)

newcatalog[written/62].files[written%62+1].name[k]=0;

for( k=0;k<5;k++)

newcatalog[written/62].files[written%62+1].type[k]=0;

for(i=0;i<segcount-1;i++)//добавляем в остальные задействованные сегменты признаки конца записей

{

newcatalog[i].files[62].rectype=5;

for(int j=0;j<7;j++)

newcatalog[i].files[62].name[j]=0;

for(int j=0;j<5;j++)

newcatalog[i].files[62].type[j]=0;

newcatalog[i].files[62].len=0;

newcatalog[i].files[62].date=0;

}

for(i=segcount;i<cs;i++)//добавляем в оставшиеся незадействованными сегменты признаки конца записей

{

newcatalog[i].files[0].rectype=5;

for(int j=0;j<7;j++)

newcatalog[i].files[0].name[j]=0;

for(int j=0;j<5;j++)

newcatalog[i].files[0].type[j]=0;

newcatalog[i].files[0].len=0;

newcatalog[i].files[0].date=0;

}

k=0;

newcatalog[0].startblock=k;

for(i=0;i<segcount-1;i++)//записываем номера блоков начала размещения файлов в сегментах

{

for(j=0;newcatalog[i].files[j].rectype!=5;j++)

k+=newcatalog[i].files[j].len;

newcatalog[i+1].startblock=k;

}

for(i=0;i<cs;i++)//заполняем заголовки сегментов

{

newcatalog[i].catsize=cs;

newcatalog[i].nextseg=i+1;

newcatalog[i].seginuse=segcount;

}

newcatalog[cs-1].nextseg=0;

delete [] catalog;//освобождаем память от старого каталога

catalog=newcatalog;//назначаем новый каталог используемым

return 0;

}

catch(...)

{

return 1;

}

}

int FS::userandlabel(char\* owner,char\* label)

{

try

{ //записываем имя владельца и метку тома

int i=0;

for(i=0;owner[i]!=0;i++)

sb.ownername[i]=owner[i];

for(;i<13;i++)

sb.ownername[i]=0;

for(i=0;label[i]!=0;i++)

sb.tomename[i]=label[i];

for(;i<13;i++)

sb.tomename[i]=0;

return 0;

}

catch(...)

{

return 1;//если перехватываем ошибку-возвращаем код ошибки

}

}

int FS::createFS()

{ try

{typedef char block[512];//Объявляем тип данных блок

typedef char seg[1024];//Объявляем тип данных сегмент

int i=0;//Счетчик

block buf,bl1, bl0;//Буферный блок, блок нулей и блок единиц

for(int i=0;i<512;i++)//Инициализируем блоки

{

buf[i]='0';

bl0[i]='0';

bl1[i]='1';

}

ofstream out;

out.open("filesystem.txt",ios::out|ios::binary);//Открываем файл на двоичный ввод

if(!out.is\_open())

return 2;//если не удалось открыть файл, то возвращаем код ошибки

out.write((char\*)&bl0,sizeof block);//Пишем в файл один пустой блок("загрузочный сектор")

//Cоздадим блок системной информации

buf[469]=6+48;

buf[471]=sb.vers+48;

for(i=0;sb.tomename[i];i++)

buf[472+i]=sb.tomename[i];

for(;i<13;i++)

buf[472+i]='0';

for(i=0;sb.ownername[i];i++)

buf[484+i]=sb.ownername[i];

for(;i<13;i++)

buf[484+i]='0';

for(i=0;i<sb.sysname.length();i++)

buf[496+i]=sb.sysname[i];

for(;i<13;i++)

buf[496+i]='0';

out.write((char\*)&buf,sizeof block);//Запишем блок системной информации и следующие 3 блока

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

out.write((char\*)&bl0,sizeof block);

//запишем сегменты каталога в файл

seg bufseg;//буферный сегмент

int j=0;//счетчики

int k=0;

int cs=catalog[0].catsize;

for(i=0;i<cs;i++)//проходим по всем сегментам

{

for(j=0;j<1024;j++)//обнуляем буферный сегмент

bufseg[j]='0';

bufseg[0]=catalog[i].catsize/256;//записываем содержимое заголовка сегмента в буфер, при этом переводим unsigned short в char[2]

bufseg[1]=catalog[i].catsize%256;

bufseg[2]=catalog[i].nextseg/256;

bufseg[3]=catalog[i].nextseg%256;

bufseg[4]=catalog[i].seginuse/256;

bufseg[5]=catalog[i].seginuse%256;

bufseg[6]='0';

bufseg[7]='0';

bufseg[8]=catalog[i].startblock/256;

bufseg[9]=catalog[i].startblock%256;

for(j=0;catalog[i].files[j].rectype!=5;j++)//записываем в буфер содержимое таблицы записей о файлах, пока не достигнем признака конца записей

{

bufseg[10+j\*16]='0';

bufseg[11+j\*16]=catalog[i].files[j].rectype+48;

for(k=0;(catalog[i].files[j].name[k])&&(k<7);k++)

bufseg[12+j\*16+k]=catalog[i].files[j].name[k];

for(;k<7;k++)

bufseg[12+j\*16+k]='0';

for(k=0;(catalog[i].files[j].type[k])&&(k<5);k++)

bufseg[18+j\*16+k]=catalog[i].files[j].type[k];

for(;k<5;k++)

bufseg[18+j\*16+k]='0';

bufseg[22+j\*16]=catalog[i].files[j].len/256;

bufseg[23+j\*16]=catalog[i].files[j].len%256;

bufseg[24+j\*16]=catalog[i].files[j].date/256;

bufseg[25+j\*16]=catalog[i].files[j].date%256;

}

bufseg[10+j\*16]='0';//запись-признак конца записей тоже записываем в буфер

bufseg[11+j\*16]=catalog[i].files[j].rectype+48;

for(k=0;(catalog[i].files[j].name[k])&&(k<7);k++)

bufseg[12+j\*16+k]=catalog[i].files[j].name[k];

for(;k<7;k++)

bufseg[12+j\*16+k]='0';

for(k=0;(catalog[i].files[j].type[k])&&(k<5);k++)

bufseg[18+j\*16+k]=catalog[i].files[j].type[k];

for(;k<5;k++)

bufseg[18+j\*16+k]='0';

bufseg[22+j\*16]=catalog[i].files[j].len/256;

bufseg[23+j\*16]=catalog[i].files[j].len%256;

bufseg[24+j\*16]=catalog[i].files[j].date/256;

bufseg[25+j\*16]=catalog[i].files[j].date%256;

for(k=26+j\*16;k<1024;k++)//неиспользуемую часть таблицы в буфере заполняем нулями

bufseg[k]='0';

out.write((char\*)&bufseg,sizeof seg);//записываем буферный сегмент в файл

}

for(i=0;i<cs;i++)//записываем в файл "блоки информации", блоки, занятые файлами-записываем как занятые единичками, пустые блоки-обозначаем нулями

{

for(j=0;catalog[i].files[j].rectype!=5;j++)

{

if(catalog[i].files[j].rectype==2)

{

for(k=0;k<catalog[i].files[j].len;k++)

out.write((char\*)&bl0,sizeof block);

}

if(catalog[i].files[j].rectype==3)

{

for(k=0;k<catalog[i].files[j].len;k++)

out.write((char\*)&bl1,sizeof block);

}

}

}

out.close();

}

catch(...)

{

return 1;//при возникновении непредвиденных ошибок возвращаем код ошибки

}

return 0;

}

FS::~FS(){

delete catalog;

};