|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| федеральное государственное автономное образовательное учреждение  высшего профессионального образования |
| **«Национальный исследовательский ядерный университет «МИФИ»** |
| ФАКУЛЬТЕТ КИБЕРНЕТИКИ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ |
| КАФЕДРА УПРАВЛЯЮЩИХ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ (№ 29) |

|  |
| --- |
| Заказчик |
| М.Н. Петухов |
| 2014 года |

Модель файловой системы ОС реального времени

Модуль создания файловой системы с заданными параметрами

Руководство системного программиста

ГОСТ 19.404-79

Исполнитель Крючков А.

Малышев А.

Шевченко С.

Щипанов С.

Принял И.В. Алексеенко

Москва 2014

Оглавление

[1. Введение 3](#_Toc385282852)

[2. Общие сведения 3](#_Toc385282853)

[2.1. Функциональное назначение 3](#_Toc385282854)

[2.2. Вызов и загрузка 3](#_Toc385282855)

[3. Алгоритм программы 4](#_Toc385282856)

[3.1. Используемые классы и методы 4](#_Toc385282857)

[4. Модульное тестирование 6](#_Toc385282858)

[5. Исходный код модуля 7](#_Toc385282859)

# Введение

В данном документе представлен текст программного модуля «модуль создания файловой системы с заданными параметрами (форматирования) для модели файловой системы ОС реального времени, разрабатываемой в рамках курса Технологии Программирования» входящего в состав разрабатываемой модели «Демонстрационная модель файловой системы ОС реального времени». Помимо исходного кода программы документ содержит описание класса файловой системы и его методов. Программа написана в среде Microsoft Visual Studio 2010, на языке С++.

# Общие сведения

## 

## Функциональное назначение

Программа «Модель файловой системы ОС реального времени» предназначена для изучения устройства и функционирования файловых систем.

Модуль форматирования – модуль программы «Модель файловой системы ОС реального времени». Он предоставляет пользователю возможность форматирования файла, содержащего файловую систему и проверяет правильность вводимых параметров.

# Вызов и загрузка

Модуль форматирования вызывается монитором команд если пользователю необходимо отформатировать файловую систему или создать новый файл с файловой системой. Перед вызовом функции форматирования происходит вызов функции, открывающей существующий физический файл или создающей новый.

# Алгоритм программы

Модуль форматирования вызывается монитором команд при вызове команды форматирования и производит форматирование существующего файла, содержащего файловую систему.

Модуль форматирования при вызове производит проверку корректности вводимых данных (версия системы, имя тома, имя пользователя и имя системы), в случае неправильных входных параметров монитору команд выдается код ошибки. Затем происходит очистка файла и запись следующих блоков:

* Загрузочный блок (в модели не используется)
* Блок системной информации, хранящий в себе версию системы, имя тома, имя пользователя и имя системы
* 4 пустых блока, так же не использующихся в системе

Далее происходит запись каталога файлов. Каталог файлов состоит из 31 сегмента. Каждый сегмент состоит из 2 блоков. Первый блок каждого сегмента содержит описатель самого сегмента и 3 описателя файлов, второй блок каждого сегмента содержит 4 описателя файлов.

## Используемые классы и методы

Модуль форматирования использует следущие классы и структуры данных:

* Класс блока данных. Содержит 512 символов, функцию очистки блока, извлечения и записи строк и чисел.

|  |
| --- |
| class Block{  public:  char byteMassive[512];  void Clean();  void InsertString(int,string);  int getInt(int,int);  string getString(int,int);} |

* Класс сегмента каталога. Содержит массив из двух блоков, временные переменные описателя сегмента и функции очистки блоков и записи временной информации в блоки.

|  |
| --- |
| class Segment  {  public:  Block blockMassive[2];  int segmentCount;  int nextSegmentNumber;  int busySegmentCount;  int startBlockNumber;  void clean();  void write();  }; |

* Класс файловой системы. Содержит открытый поток файла и методы, реализующие функции, вызываемые монитором команд.

|  |
| --- |
| class FileSystem  {  public:  fstream memory;  void createFile(string); //открытие существующего физического файла или создание  int format(string,string,string,string); //форматирование файловой системы  Block readBlock(int); //считывание блока, заданного номером  void writeBlock(Block,int); //запись блока в заданную позицию файла  FileDescriptor getRecord(string); //извлечение описателя файла по имени  int writeRecord(FileDescriptor); //сохранение записи о файле в конец  int deleteRecord(string); //удаление записи о файле  void closeFileSystem();  int showTableOfContent();  int showContentInAlphavit();  int createNewFile();  int showInformation();  int deleteFile();  int reNameFile();  int addInformationToFile();  int changeFileType();  int compressionOfFileSystem();  int writeVolumeLabelAndOwner();  }; |

# Модульное тестирование

Модульное тестирование функции форматирования производилось с использованием фреймворка для модульного тестирования Google C++ Test Framework.

Для запуска тестов необходимо запустить fileSystem.exe с передаваемым параметром runtest.

Написанные тесты для модуля форматирования:

|  |
| --- |
| #include "FileSystem.h"  #include <gtest\gtest.h>  class TestFileSystem : public ::testing::Test {  public:  void SetUp() // инициализация тестируемого класса  {  f=new FileSystem;  f->createFile("3");  f->openFile("3");  }  FileSystem \*f;  };  TEST\_F(TestFileSystem, correctFormat) //проверка успешного форматирования  {  ASSERT\_EQ(f->format("23","tom","otherString","last"),0);  }  TEST\_F(TestFileSystem, incorrectFormatVersion) // проверка вывода кода ошибки при некорректной версии системы  {  ASSERT\_EQ(f->format("235","tom","otherString","last"),0);  }  TEST\_F(TestFileSystem, incorrectFormatTomName) //проверка кода ошибки при неправильном имени тома  {  ASSERT\_EQ(f->format("23","123456789123456789","otherString","last"),0);  }  TEST\_F(TestFileSystem, incorrectFormatUserName) //проверка кода ошибки при неверном имени пользователя  {  ASSERT\_EQ(f->format("23","12345678","123456789123456789","last"),0);  }  TEST\_F(TestFileSystem, incorrectFormatSystemName) //проверка кода ошибки при некорректном имени системы  {  ASSERT\_EQ(f->format("23","1234567","otherString","12345678901234567"),0);  }  int main(int argc, char \*\*argv)  {  if(argc > 1)  {  for (int i=1;i<argc;i++)  {  string tmp(argv[i]);  if (tmp.compare("runtest")==0)  {  ::testing::InitGoogleTest(&argc, argv);  RUN\_ALL\_TESTS();  getch();  }  }  }  return 0;  } |

Результаты прогонки тестов следующие:

|  |
| --- |
| c:\git\filesystem\testfilesystemformat.cpp(24): error: Value of: 0  Expected: f->format("235","tom","otherString","last")  Which is: 1  c:\git\filesystem\testfilesystemformat.cpp(29): error: Value of: 0  Expected: f->format("23","123456789123456789","otherString","last")  Which is: 1  c:\git\filesystem\testfilesystemformat.cpp(34): error: Value of: 0  Expected: f->format("23","12345678","123456789123456789","last")  Which is: 1  c:\git\filesystem\testfilesystemformat.cpp(39): error: Value of: 0  Expected: f->format("23","1234567","otherString","12345678901234567")  Which is: 1 |

# Исходный код модуля

|  |
| --- |
| #include "FileSystem.h"  int FileSystem::format(string version, string tomName, string userName, string systemName)  {  if(version.length()<3||tomName.length()<13||userName.length()<13||systemName.length()<13) return 1;  memory.clear();  Block firstBlock; //загрузочный блок  firstBlock.Clean();  memory.write((char\*)firstBlock.byteMassive, sizeof(firstBlock.byteMassive));  Block systemBlock;  systemBlock.Clean(); //системный блок  systemBlock.InsertString(469,"6");  systemBlock.InsertString(470,version);  systemBlock.InsertString(472,tomName);  systemBlock.InsertString(484,userName);  systemBlock.InsertString(496,systemName);  memory.write((char\*)systemBlock.byteMassive, sizeof(firstBlock.byteMassive));  for (int i=0;i<4;i++)  {  Block emptyBlock; //4 пустых блока  emptyBlock.Clean();  memory.write((char\*)emptyBlock.byteMassive, sizeof(emptyBlock.byteMassive));  }  Segment catalog[31];  for (int i=0;i<30;i++)  {  catalog[i].clean();  catalog[i].segmentCount=31;  catalog[i].nextSegmentNumber=(i+1);  catalog[i].busySegmentCount=0;  catalog[i].write();  memory.write((char\*)catalog[i].blockMassive[0].byteMassive, sizeof(catalog[i].blockMassive[0].byteMassive));  memory.write((char\*)catalog[i].blockMassive[1].byteMassive, sizeof(catalog[i].blockMassive[1].byteMassive));  }  catalog[30].clean();  catalog[30].segmentCount=31;  catalog[30].nextSegmentNumber=(0);  catalog[30].busySegmentCount=0;  catalog[30].write();  memory.write((char\*)catalog[30].blockMassive[0].byteMassive, sizeof(catalog[30].blockMassive[0].byteMassive));  memory.write((char\*)catalog[30].blockMassive[1].byteMassive, sizeof(catalog[30].blockMassive[1].byteMassive));  return 0;  } |