|  |
| --- |
| 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 добавления информации в существующий файл

Описание программы

ГОСТ 19.404-79

Исполнитель Копытин М.В.

Надершин М.Н.

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 добавления информации в существующий файл 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 добавления информации – модуль программы «Модель файловой системы ОС реального времени». Он предоставляет пользователю возможность добавлять в уже существующие файлы новую информацию.

# Вызов и загрузка

Модуль форматирования вызывается монитором команд если пользователю необходимо добавить в файл информацию.

# Алгоритм программы

Модуль добавления информации вызывается монитором команд при вызове команды добавления информации.

Модуль при вызове производит проверку корректности вводимых данных (имя файла, количество добавляемой информации), в случае неправильных входных параметров монитору команд выдается код ошибки. Затем происходит добавление информации в файл:

## Используемые классы и методы

Модуль форматирования использует следующие классы и структуры данных:

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

|  |
| --- |
| struct FileDescriptor  {  public:  string descriptorType;  string fileName;  string fileType;  int blockCount;  string creationDate;  string firstBlockNumber;  }; |

* 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

|  |
| --- |
| FileSystem::addInformationToFile()  {  string FileName;  int InformationCount;  int CountLength=8;  int resultCode = 1;  int lenCritical = 48;  int CntCritical = 2 ^ 9;  FileDescriptor AddFile;  FileDescriptor NextFile;  int ResultCount;  int freeSpace;  int RecordNumber,NextRecordNumber;  int i;  cout << "Введите имя файла" << endl;  cin >> FileName;  if (FileName.length() > lenCritical)  return(1);  cout << "Введите кол-во добавляемой информации" << endl;  cin >> InformationCount;  if (InformationCount > CntCritical)  return(2);  AddFile = getRecord(FileName);  if ((AddFile.descriptorType) == "0010000000000000")  return(1);  RecordNumber=getRecordNumber(FileName);  NextRecordNumber=RecordNumber+1;  NextFile=getRecord(NextRecordNumber);      if (((toInt(AddFile.firstBlockNumber) + AddFile.blockCount + InformationCount) /  512) > (toInt(NextFile.firstBlockNumber)))  {  AddFile.blockCount+=InformationCount;  for(i=217;((getRecord(i).fileType)=="");i--)    AddFile.firstBlockNumber=toString((toInt(getRecord(i).firstBlockNumber)+  getRecord(i).blockCount+1),8);  deleteRecord(FileName);    writeRecord(AddFile);  resultCode=0;  }  else  {  AddFile.blockCount=(AddFile.blockCount+InformationCount);  deleteRecord(FileName);  writeRecord(AddFile,RecordNumber);  resultCode=0;  }  return(resultCode);  } |