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

Техническое задание

ГОСТ 19.201-78

Исполнитель С.С. Богатыренко

Принял И.В. Алексеенко

Москва 2012

**Содержание:**

1. Введение.
2. Основание для разработки.
3. Назначение и область применения.
4. Требования к программе.
   1. Требования к функциональным характеристикам.
   2. Требования к надежности.
   3. Требования к составу и параметрам технических средств.
   4. Требования к информационной и программной совместимости.
   5. Требования к маркировке и упаковке
5. Требования к программной документации
6. Стадии и этапы разработки
7. Порядок контроля и приемки
8. **Введение**

Полное наименование программы: «Монитор обработки команд программы «Модель файловой системы ОС реального времени»», краткое наименование «МК ». Программный комплекс применяется для моделирования работы файловой системы.

1. **Основание для разработки**

Разработка ведется на основании пояснительной записки к учебному проекту «Монитор обработки команд программы «Модель файловой системы ОС реального времени»».

1. **Назначение и область применения**

Программа «Модель файловой системы ОС реального времени» предназначена для изучения устройства и функционирования файловых систем. МК - основной модуль программы. В нем реализован консольный пользовательский интерфейс, обеспечивающий доступ к подпрограммам, имитирующим работу файловой системы. Предоставляя пользователю набор команд для вызова соответствующих подпрограмм, монитор обеспечивает правильность их вызова и контролирует результаты их выполнения.

1. **Требования к программе**
   1. **Требования к функциональным характеристикам**

Задача модели заключается в том, чтобы отразить все существенные аспекты функционирования файловой системы и её структуру (описанные в разделе «Постановка задачи» пояснительной записки к проекту) при построении программы используется следующий подход:

* Создается специальный класс файловой системы, объект класса содержит информационные поля, в которые помещается информация блока системной информации файловой системы и каталог, представляющий собой массив из необходимого количества сегментов.
* Сегмент реализуется как локальная структура класса файловой системы, он содержит информационные поля с информацией заголовка сегмента и массив записей о файлах.
* Запись о файле также является локальной структурой, содержит необходимые данные о файле (имя, размер и т.д.).
* Все процедуры, реализующие операции с файловой системой являются методами класса файловой системы.
* Так как в процессе работы модели не осуществляется работы с реальной информацией, блоки, представляющие собой файлы и свободное пространство, в классе файловой системы отсутствуют.
* В процессе работы модели создается объект класса файловой системы, для которого в соответствии с командами, вводимыми пользователем, МК вызывает необходимые методы.
* В МК имеется отдельная команда, позволяющая создать на жестком диске текстовый файл, состав которого соответствует как структуре имеющегося в модели объекта класса файловой системы, так и требованиям, указанным в разделе «Постановка задачи» пояснительной записки к проекту.

Описание полей класса файловой системы (см. приложение 1):

|  |  |  |  |
| --- | --- | --- | --- |
| FS SYSTEM – объект класса файловой системы | SB sb – объект класса блок системной информации | unsigned short vers – версия системы | |
| char tomename[13] – метка тома | |
| char ownername[13] – имя владельца | |
| string sysname – имя системы | |
| segment \*catalog – динамический массив из N объектов типа сегмент | unsigned short catsize – число сегментов в каталоге | |
| 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] – массив из 63 объектов типа запись о файле | unsigned short rectype – тип записи |
| char name[7] – имя файла |
| char type[5] – тип файла |
| unsigned short len – число блоков, занятых файлом |
| unsigned short date – дата создания файла |

Задачи монитора команд заключаются в следующем:

* Предоставить пользователю консольный интерфейс для ввода команд, управляющих файловой системой;
* Организовать проверку правильности ввода команд и их параметров пользователем на соответствие шаблону;
* При корректных запросах пользователя вызывать требуемые процедуры;
* Обрабатывать различные варианты завершения процедур, выводить на экран соответствующие сообщения;
* Предоставить возможность вывода сообщений пользователю вызываемыми процедурами;
* Предоставить пользователю информацию о работе всей программы и об использовании отдельных команд;
* Создавать на жестком диске файл со структурой, соответствующей созданной модели файловой системы.

Для выполнения этих задач предполагается создать специальный класс монитора команд. Объект этого класса содержит: объект класса файловой системы (с которым и будет осуществляться работа) и переменную типа команда. Команда реализуется как отдельная структура, содержащая название команды, перечень параметров а также вспомогательные переменные, необходимые в процессе работы с командой(преобразования полученного от пользователя запроса к виду, удобному для вызова необходимой функции, вызова этой функции, а затем обработки возвращенного функцией кода). Согласно пояснительной записке к проекту, название команды не превышает 13 символов, количество параметров – не более трех, длина параметра (символьная) – не более 12 символов. Класс предоставляет методы для считывания команды, её выполнения и обработки кода её завершения. Так как каждая команда имеет свои особенные параметры и коды завершения, в этих методах предполагается использование конструкции switch(){case…}.

Для того, чтобы предоставить возможность методам класса файловой системы выводить сообщения пользователю, а также для обработки ошибок предполагается использовать глобальные потоковые переменные и флаги работы с ними (устанавливаются при использовании функцией потоковой переменной). Специальный глобальный флаг предполагается использовать для реализации выхода из приложений. Описание глобальных переменных смотри в приложении 3.

Описание полей класса монитора команд (см. приложение 2):

|  |  |  |
| --- | --- | --- |
| Monitor CM – объект класса монитор команд | FS SYSTEM - объект класса файловой системы | |
| string commands[28] – массив, содержащий список команд | |
| command com – объект типа команда | int numc – номер команды в списке команд |
| int nump – число параметров команд |
| long size – переменная для записи передаваемых в вызываемые методы числовых параметров (размеров файлов) |
| int retcode – код, возвращаемый вызванным методом |
| char name[14] – имя команды |
| char par1[13] – 1-й параметр команды |
| char par1[13] ] – 2-й параметр команды |
| char par1[13] ] – 3-й параметр команды |

Описание работы МК:

* Инициализация

Создание объекта класса монитора команд, присвоение начальных значений переменным.

* Вывод приветствия

На экран выводится приветствие и информация о программе.

* Бесконечный цикл While(1)
* Выполнение функции readcom()

Функция также работает в бесконечном цикле while(1).Она выполняет очистку содержимого объекта com, выводит на экран приглашение к вводу команды, записывает введенные значения в поля объекта com, выполняет предварительную проверку введенных параметров на соответствие шаблону. Если на любом из этапов обнаруживается несоответствие введенных данных установленным требованиям (слишком много параметров, слишком большая длина параметров и т.д.), то на экран выводится сообщение об этом и выполняется оператор continue, который начинает итерацию цикла ввода команды сначала. При корректном вводе имени команды и всех параметров выполняется оператор break, который завершает ввод команды.

* Выполнение функции execute()

Данная функция осуществляет поиск введенного имени команды в списке команд, при нахождении его в списке номер команды записывается в объект com. Далее проводится окончательная проверка введенных параметров на соответствие шаблону для данной команды (верное количество параметров, верный тип параметров и т.д.). Если данные корректны, то вызывается соответствующий команде метод класса файловой системы, код, который он возвращает, сохраняется в объекте com. Если на любом этапе обнаруживается несоответствие данных шаблону (неверное имя команды, отсутствие параметров и т.д.), то в переменной com устанавливается специальный код ошибки (10 - ошибка, перехваченная монитором команд), в буфер ошибок записывается соответствующее ошибке сообщение и устанавливается флаг буфера ошибок.

* Выполнение функции errcheck()

Эта функция проверяет номер команды в объекте com и для выполненной команды осуществляет проверку кода, возвращенного методом, вызванным по этой команде функцией execute (либо установленного в самой функции execute) . Функция выводит сообщение, соответствующее возвращенному коду, проверяет состояние буферов сообщений и, при необходимости, выводит их содержимое. Если по окончании работы функции не установлен флаг выхода из программы, то начинается новая итерация цикла работы монитора, в противном случае осуществляется выход из приложения.

* Вывод сообщения о выходе.

Команды, вводимые пользователем, должны иметь формат: имя\_команды параметр1 параметр2……параметрN

Текст сообщений, выводимых пользователю, смотри в приложении 4. Сообщения, выводимые отдельными процедурами (через буферы вывода сообщений), в приложении не приведены.

Описание отдельных процедур:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя команды | Передаваемые параметры | Типы параметров | Возвращаемые коды | Описание |
| format | Размер системы(в блоках) | Unsigned short | 0-успешное завершение  1-непредвиденная ошибка | Форматировать носитель | |
| Метка тома | Char[13] |
| Владелец | Char[13] |
| content | - | - | 0-успешное завершение  1-непредвиденная ошибка  2-файлы отсутствуют | Вывести содержимое как есть |
| sortcontent | - | - | 0-успешное завершение  1-непредвиденная ошибка  2-файлы отсутствуют | Вывести содержимое по алфавиту |
| freespace | - | - | 0-успешное завершение  1-непредвиденная ошибка  2-свободные области отсутствуют | Вывести информацию о свободном месте |
| create | Имя файла | Char[7] | 0-успешное завершение  1-непредвиденная ошибка  2-недостаточно места в системе  3-недостаточно места в каталоге  4-такой файл уже существует | Создать файл |
| Тип файла | Char[5] |
| Размер | Unsigned short |
| del | Имя файла | Char[7] | 0-успешное завершение  1-непредвиденная ошибка  2-файл с таким именем не найден | Удалить файл |
| Тип файла | Char[5] |
| rename | Старое имя файла | Char[7] | 0-успешное завершение  1-непредвиденная ошибка  2-файл с таким именем не найден  3-файл с такими именем и типом уже существует | Переименовать файл |
| Тип файла | Char[5] |
| Новое имя файла | Char[7] |
| append | Имя файла | Char[7] | 0-успешное завершение  1-непредвиденная ошибка  2-файл с таким именем не найден  3-недостаточно места в системе  4-недостаточно места в каталоге | Добавить информацию в файл |
| Тип файла | Char[5] |
| Добавляемый размер | Unsigned short |
| changetype | Имя файла | Char[7] | 0-успешное завершение  1-непредвиденная ошибка  2-файл с таким именем не найден  3-файл с такими именем и типом уже существует | Изменить тип файла |
| Старый тип файла | Char[5] |
| Новый тип | Char[5] |
| defrag | - | - | 0-успешное завершение  1-непредвиденная ошибка | Сжать файловую систему |
| userandlabel | Владелец | Char[13] | 0-успешное завершение  1-непредвиденная ошибка | Записать владельца и метку тома |
| Метка тома | Char[13] |
| createFS | - | - | 0-успешное завершение  1-непредвиденная ошибка  2-не удалось записать файл на жесткий диск | Создать файл, соответствующий имеющейся конфигурации файловой системы |
| help | - | - | 0-успешное завершение  1-непредвиденная ошибка | Вывести список команд |
| имя\_команды? | - | - | 0-успешное завершение  1-непредвиденная ошибка | Вывести информацию о команде |
| exit | - | - | 0-успешное завершение  1-непредвиденная ошибка | Выйти из программы |

* 1. **Требования к надежности**

При работе программы должен обеспечиваться высокий уровень надежности, что подразумевает отсутствие зацикливания программы (бесконечного выполнения без реакции на действия пользователя), возникновения не перехваченных исключительных ситуаций, приводящих к непредвиденному завершению работы программы, а также случаев некорректного выполнения программой своих функций.

Для предотвращения аварийного завершения программы используются стандартные средства языка программирования (операторы try{} catch{}). Прочие проблемы надежности решаются тщательным всесторонним тестированием программы.

* 1. **Требования к составу и параметрам технических средств.**

Для успешного выполнения программы требуется IBM – совместимый компьютер под управлением ОС Windows XP и выше, имеющий клавиатуру для ввода информации оператором и монитор для вывода информации и жесткий диск или съемный носитель для записи информации. Компьютер должен иметь процессор с тактовой частотой не менее 233 МГц и объем оперативной памяти не менее 64 Мб.

* 1. **Требования к информационной и программной совместимости**

При разработке программного комплекса должна быть обеспечена полная совместимость между МК и подключаемыми функциями. Эта задача решается в МК, где контролируется количество и тип передаваемых в функции параметров.

Для разработки используется среда MS Visual Studio 2010 и выше, благодаря чему достигается совместимость компиляторов на разных машинах в течение разработки.

Скомпилированное в Release – конфигурации приложение будет содержать все библиотеки, необходимые для запуска под управлением ОС Windows.

* 1. **Требования к маркировке и упаковке**

Носитель с программой помечается «ФС\_K5-291\_2012».

Данный комплекс программ предполагается сдавать заказчику в следующем виде:

* + - * Каталог ФС\_K5-291\_2012

Каталог ФС

Тексты модулей комплекса программ

Каталог Debug

Файл ФС.exe – отладочная версия

Каталог Release

Файл ФС.exe – рабочая версия

1. **Требования к программной документации**

Предварительный состав программной документации: пояснительная записка, техническое задание, текст и описание программы, программа и методика испытаний, инструкция системного программиста, инструкция оператора.

1. **Стадии и этапы разработки**
2. Пояснительная записка – 25.10.12
3. Техническое задание – 8.11.12
4. Текст программы - 15.11.12
5. Программа и методика испытаний – 22-29.11.12
6. Инструкция системного программиста -22-29.11.12
7. Инструкция оператора - 22-29.11.12
8. Отладка программы - 6.12.12
9. Сдача программы заказчику – 13.12.12
10. **Порядок контроля и приемки**

Приемка программы осуществляется преподавателем. Корректность работы программы устанавливается путем проведения специальных тестов, проверяющих работу программы на различных наборах входных данных в различных условиях на основании ПЗ, ПМИ и РО.

**Приложение 1. Описание полей класса файловой системы.**

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);//конструктор класса, будет форматировать систему

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();

friend class Monitor;

~FS();

};

**Приложение 2. Описание класса монитора команд.**

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

};

**Приложение 3. Глобальные переменные, используемые программой.**

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

**Приложение 4. Сообщения, выводимые пользователю в процессе работы программы.**

|  |  |
| --- | --- |
| Причина появления | Текст сообщения |
| Приветствие | Вас приветствует модель файловой системы ОС реального времени. Данная программа моделирует работу файловой системы и имитирует её основные функции.  Чтобы получить информацию о предоставляемых командах введите “help”. |
| **Сообщения об ошибках** | |
| Превышено максимально допустимое количество символов в команде | Превышена допустимая длина команды! |
| В команде обнаружены русские символы | Русские символы не разрешены! |
| Введено неверное число параметров команды | Неверное число параметров! |
| Введено неверное имя команды | Неверное имя команды! |
| Введен неверный первый параметр команды | Неверный первый параметр! |
| Введен неверный второй параметр команды | Неверный второй параметр! |
| Введен неверный третий параметр команды | Неверный третий параметр! |
| Попытка выполнить операцию с неотформатированной системой | Файловая система не отформатирована! |
| Непредвиденная ошибка во время выполнения команды | Во время выполнения команды произошла непредвиденная ошибка! |
| Попытка получить содержимое пустой файловой системы | Файлы в системе отсутствуют! |
| Попытка получить информацию о свободных областях полностью заполненной системы | Свободные области в системе отсутствуют! |
| Недостаток места в системе при записи/перезаписи файла | Недостаточно места в системе! |
| Недостаток места в каталоге при записи/перезаписи файла | Недостаточно места в каталоге! |
| Попытка создания уже существующего файла | Такой файл уже существует! |
| Попытка обращения к несуществующему файлу | Такого файла нет в системе! |
| Ошибка записи файла на жесткий диск | Не удалось записать файл на жесткий диск! |
| Ошибка в МК | Во время работы программы возникла непредвиденная ошибка. Программа завершает своё выполнение. |
| **Сообщения об успешном выполнении** | |
| format | Файловая система отформатирована |
| content, sortcontent | Содержимое файловой системы:  «содержимое» |
| freespace | Информация о свободных областях:  «информация» |
| create | Файл «имя.тип» с размером «размер» успешно создан |
| del | Файл «имя.тип» удален |
| rename | Файл «имя.тип» переименован в «имя2.тип» |
| append | Размер файл «имя.тип» увеличен на «размер» |
| changetype | Тип файла «имя.тип» изменен на «.тип2» |
| defrag | Файловая система сжата |
| userandlabel | Имя владельца и метка тома изменены на «метка тома» и «имя владельца» |
| createFS | Файл filesystem.txt успешно создан |
| exit | Завершение работы. |
| **Справочные сообщения** | |
| help | Команды в системе имеют следующий формат:  имя\_команды параметр\_1 параметр\_2 параметр\_3  Количество параметров у разных команд может быть от 0 до 3  При задании параметров не допускается использование символов русского алфавита  В системе имеются следующие команды:  format размер\_системы метка\_тома имя\_владельца  content  sortcontent  freespace  create имя\_файла тип\_файла размер\_файла  del имя\_файла тип\_файла  rename старое\_имя\_файла тип\_файла новое\_имя\_файла  append имя\_файла тип\_файла добавляемый\_размер  changetype имя\_файла старый\_тип\_файла новый\_тип\_файла  defrag  userandlabel имя\_владельца метка\_тома  createFS  help  exit  Для получения информации о команде введите её имя с вопросительным знаком в конце, например format? |
| format? | format размер\_системы метка\_тома имя\_владельца  Данная команда создает файловую систему заданного размера, устанавливает её метку тома и имя владельца  Размер задается в блоках, он должен быть больше 9-ти и не должен превышать 65536-ти  Имя владельца и метка тома не должны быть длиннее 12-ти символов |
| content? | content  Данная команда выводит перечень файлов в системе в том порядке, в котором они там находятся |
| sortcontent? | sortcontent  Данная команда выводит перечень файлов в системе в алфавитном порядке |
| freespace? | freespace  Данная команда выводит перечень пустых областей в системе |
| create? | create имя\_файла тип\_файла размер\_файла  Данная команда создает в системе файл с заданными именем и типом, имеющий заданный размер  Имя создаваемого файла не должно быть длиннее 6-ти символов, тип - 4-х  Размер должен быть ненулевым и не должен превышать 65468-ми |
| del? | del имя\_файла тип\_файла  Данная команда удаляет из системы файл с заданными именем и типом  Имя файла не должно быть длиннее 6-ти символов, тип - 4-х |
| rename? | rename старое\_имя\_файла тип\_файла новое\_имя\_файла  Данная команда меняет имя файла с заданными именем и типом на новое  Имя файла не должно быть длиннее 6-ти символов, тип - 4-х |
| append? | append имя\_файла тип\_файла добавляемый\_размер  Данная команда увеличивает размер файла с заданными именем и типом на указанную величину  Имя файла не должно быть длиннее 6-ти символов, тип - 4-х  Размер должен быть ненулевым и не должен превышать 65468-м |
| changetype? | changetype имя\_файла старый\_тип\_файла новый\_тип\_файла  Данная команда меняет тип файла с заданными именем и типом на новый  Имя файла не должно быть длиннее 6-ти символов, тип - 4-х |
| defrag? | defrag  Данная команда дефрагментирует файловую систему |
| userandlabel? | userandlabel имя\_владельца метка\_тома  Данная команда меняет имя владельца и метку тома в системе на новые  Имя владельца и имя тома не должны быть длиннее 12-ти символов |
| createFS? | createFS  Данная команда создает на жестком диске файл filesystem.txt со структурой, соответсвующей созданной файловой системе |
| help? | help  Данная команда выводит список имеющихся команд и их параметров |
| exit? | exit  Данная команда осуществляет выход из приложения |