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[12] – метка тома | |
| char ownername[12] – имя владельца | |
| char sysname[12] – 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[6] – имя файла |
| char type[4] – 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), потоковые переменные (которые будут использоваться в качестве буферов вывода сообщений), флаги использования буферов, переменную типа команда. Команда реализуется как отдельная структура, содержащая название команды и перечень параметров. 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…}.

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

|  |  |  |
| --- | --- | --- |
| Monitor CM – объект класса монитор команд | FS SYSTEM - объект класса файловой системы | |
| stringstream screen – буфер для вывода сообщений | |
| stringstream err – буфер для вывода сообщений об ошибках | |
| bool screenflag – флаг использования буфера сообщений | |
| bool errflag – флаг использования буфера сообщений об ошибках | |
| command com – объект типа команда | 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 выполняет очистку содержимого переменной com, выводит на экран приглашение к вводу команды, записывает введенные значения в переменную com, выполняет проверку и преобразование типов данных параметров в соответствии с именем команды, проверяет корректность введенных данных. Если на любом из этапов обнаруживается несоответствие введенных данных установленным требованиям, то на экран выводится сообщение об этом и выполняется оператор continue, который начинает новую итерацию цикла.

* Выполнение функции execute()

После проверки введенных данных происходит вызов процедуры, соответствующей введенной команде с передачей ей требуемых параметров. Процедуры могут выводить данные в буфер вывода сообщений и, при необходимости, в буфер сообщений об ошибках, при этом они должны устанавливать флаги использования этих буферов.

* Выполнение функции errcheck()

После завершения работы процедуры выполняется проверка кода её завершения и на экран выводятся необходимые сообщения. При этом, если процедура должна была вывести на экран информацию, проверяется состояние флага использования буфера вывода сообщений, если он установлен, то содержимое буфера выводится на экран, а флаг снимается. Если процедура вернула код ошибки, то аналогичные действия производятся с буфером вывода сообщений об ошибках.

* Вывод сообщения о выходе.

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

Текст сообщений, выводимых пользователю, смотри в приложении 3. 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[12] |
| Владелец | Char[12] |
| 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[6] | 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[4] |
| Размер | Unsigned short |
| delete | Имя файла | Char[6] | 0-успешное завершение  1-непредвиденная ошибка  2-файл с таким именем не найден | Удалить файл |
| Тип файла | Char[4] |
| rename | Старое имя файла | Char[6] | 0-успешное завершение  1-непредвиденная ошибка  2-файл с таким именем не найден | Переименовать файл |
| Тип файла | Char[4] |
| Новое имя файла | Char[6] |
| append | Имя файла | Char[6] | 0-успешное завершение  1-непредвиденная ошибка  2-файл с таким именем не найден  3-недостаточно места в системе | Добавить информацию в файл |
| Тип файла | Char[4] |
| Добавляемый размер | Unsigned short |
| changetype | Имя файла | Char[6] | 0-успешное завершение  1-непредвиденная ошибка  2-файл с таким именем не найден | Изменить тип файла |
| Старый тип файла | Char[4] |
| Новый тип | Char[4] |
| defrag | - | - | 0-успешное завершение  1-непредвиденная ошибка | Сжать файловую систему |
| userandlabel | Владелец | Char[12] | 0-успешное завершение  1-непредвиденная ошибка | Записать владельца и метку тома |
| Метка тома | Char[12] |
| 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 «FS\_K5-291\_2012».

Данный комплекс программ предполагается сдавать заказчику в следующем виде:

* + - * Каталог FS\_K5-291\_2012

Каталог FS

Тексты модулей комплекса программ

Каталог Debug

Файл FS.exe – отладочная версия

Каталог Release

Файл FS.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 – 25.11.12
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[12];//метка тома

char ownername[12];//имя владельца

char sysname[12];//имя системы

};

struct FRec{ //запись о файле

unsigned short rectype; //тип записи

char name[6];//имя файла

char type[4]; //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

SB sb;//системный блок

segment \*catalog;//каталог-массив сегментов

};

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

class Monitor{//класс монитора команд

struct command{//структура команды

char name[14];//имя команды

char par1[13];//параметры команды

char par2[13];

char par3[13];

};

FS SYSTEM;// объект класса файловой системы

stringstream screen;//буфер для вывода сообщений

stringstream err;//буфер для вывода сообщений об ошибках

bool screenflag, errflag;//флаги использования буферов

command com;//обрабатываемая команда

Monitor()//конструктор

void readcom();//считывание команды

void execute();//выполнение команды

void errcheck();//проверка ошибок

};

**Приложение 3. Сообщения, выводимые пользователю в процессе работы программы.**

|  |  |
| --- | --- |
| Наименование | 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”. |
| Неверная команда | Введено неверное имя команды. |
| Неверное количество параметров команды | Количество параметров команды не соответствует ожидаемому. |
| Неверные параметры команды | Ведены некорректные значения параметров команды. |
| Успешное выполнение команды | Команда выполнена успешно. |
| Непредвиденная ошибка при выполнении команды | Во время выполнения команды возникла непредвиденная ошибка, команда не выполнена. |
| Ошибка в МК | Во время работы программы возникла непредвиденная ошибка. Программа завершает своё выполнение. |
| Выход | Завершение работы. |