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

Добавление информации в файл

Техническое задание

ГОСТ 19.201-78

Руководитель проекта С.С. Богатыренко

Исполнитель И.С. Тертышный

А.О. Захарова

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»». Полное наименование подпрограммы: «Добавление информации в файл «Модели файловой системы ОС реального времени»», краткое наименование «ДИФ ФС».

1. **Основание для разработки**

Разработка ведется на основании пояснительных записок к учебному проекту «Монитор обработки команд программы «Модель файловой системы ОС реального времени»» и «Добавление информации в файл «Модели файловой системы ОС реального времени»» и технического задания 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. «ДИФ ФС» - процедура добавления информации в файл.

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:

* Запись о файле является локальной структурой, содержит необходимые данные о файле (имя, размер и т.д.).
* Процедура «Добавление информации в файл» является частью методов 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, по итогу вызова метода создания файла, создается объект класса файловой системы, для которого в соответствии с командами, вводимыми пользователем, МК вызывает необходимые методы.

Описание полей структуры записи о файле:

|  |  |
| --- | --- |
| FRec 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 |

Задача процедуры добавление информации в файл:

* Реализация требуемой процедуры сводится к редактированию блоков в файле.

Процедура ищет в каталоге запись о файле с заданными именем и типом. При необходимости удаляет файл. Затем выполняется процедура «создание файла» для найденного файла с изменением на нужную величину размером, а тип старой записи меняется на «запись о свободной области». Если файл в файловой системе не найден, возвращается код ошибки о том, что файл не обнаружен в файловой системе.

Для выполнения этой задачи создана процедура «Добавление информации в файл», которая получает три входящих переменных: «Имя файла» (char name[6]), «Тип файла» (char type[4]) и размер файла (unsigned short len).

Принцип работы функции Append

1. Если файловой системы не существует:
   * + - 1. Выводится код ошибки: «1» (Файл «filesystem.txt» не найдет);

2. Перед записью совершается прохождение по каталогу и проверяется на совпадение имен:

* + - 1. Файла с таким именем не существует в каталоге
         1. Происходит запись файла;
      2. Файл с таким именем существует в каталоге
         1. Процедура возвращает код ошибки: «4» (Такой файл уже существует);

3. Процедура проходит по каталогу файлов и ищет свободную область, в которую помещается файл:

* + - 1. Запись поверх удаленного файла:
         1. Область, которая осталась после удаления другого файла, заполняется новым файлом, если она больше чем создаваемый файл, то создается запись в каталоге о свободной области соответствующего размера и происходит сдвиг последующих записей на одну позицию;
      2. Запись в неразмеченную область:
         1. «Запись о нераспределенной памяти» и «Признак конца записей» перезаписываются на следующую позицию;
         2. Если «Признак конца записей» вышел за границу сегмента, то совершается переход на следующий сегмент, после в заголовках всех сегментов увеличивается счетчик задействованных сегментов;
         3. Если все имеющиеся сегменты полностью заполнены записями, процедура возвращает код ошибки: «3» (Недостаточно места в каталоге);
         4. Если в системе недостаточно памяти для записи файла, возвращается код ошибки: «2» (Недостаточно места в системе);

4. После записи файла совершается переход в ту область памяти, где записан файл, которая заполняется единицами;

5. Выводится информация о завершении процедуры:

* + - * 1. Выводится код: «0» (Успешно завершено);
  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 неперехваченных 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
3. Техническое задание
4. Текст программы
5. Программа и методика испытаний
6. Инструкция системного программиста
7. Инструкция оператора
8. Отладка программы
9. Сдача программы заказчику
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.