
Разработка приложения для эмуляции работы файловой системы

Лабораторная работа

Ревизия: 0.1

История изменений

09.04.2010 – Версия 0.1. Первичный документ. Влад Ковтун

Содержание

История изменений	2
Содержание	3
Лабораторная работа 6. Разработка приложения для эмуляции работы файловой системы	4
Вопросы	4
Постановка задачи	4
Цель	4
Задачи	4
Задание	4
Простой пример	7
Требования	7
Методические указания для самостоятельной работы	8
Тестирование	8
Теоретические сведения	8
Литература	8

Лабораторная работа 6. Разработка приложения для эмуляции работы файловой системы

Вопросы

1. Постановка задачи.
2. Методические рекомендации.

Постановка задачи

Цель

Разработать приложение, которое эмулирует работу файловой системы посредством командной строки.

Задачи

1. Ознакомиться с понятиями файловой системы в ОС.
2. Ознакомиться с особенностями выполнения команд из командной строки MS-DOS.
3. Разработать консольное приложение которое эмулирует работу файловой системы по управлению:
 - Файлами.
 - Каталогами.
 - Связями.
4. Оформить функциональную спецификацию на приложение. В функциональной спецификации обязательно указать, каким образом производится тестирование на корректность.
5. Оформить отчет к лабораторной работе.

Задание

Опишем основные команды, которые следует реализовать.

MD – создание директории

Форма команды: `MD [drive:]path`

Замечание: команда MD не создает промежуточные директории в пути.

Примеры

- a) `MD C:\Test` – создает директорию Test\ в корневом каталоге.
- b) `MD Test` – создает поддиректорию Test\ в текущей директории.
- c) `MD C:\Dir1\Dir2\NewDir` – создает поддиректорию «NewDir», если директория «C:\Dir1\Dir2» существует.

CD – изменяет текущую директорию

Формат команды: `CD [drive:][path]`

Замечание: Использование команды CD без параметров не допускается.

Примеры

- a) `CD C:` – установить корневой каталог как текущий.
- b) `CD C:\Dir1` – установить «C:\Dir1» как текущую директорию.
- c) `CD Dir1` – установить поддиректорию Dir1 текущей директории, как новую текущую.

RD – удаляет директорию

Удаляется директория, если та не пустая (не содержит никаких файлов или поддиректорий).

Command format: **RD** [drive:]path

Замечания: Не допускается удаление текущей директории таким образом.

Примеры

RD Dir2 – удаляет поддиректорию «Dir2».

DELTREE – удаляет директорию и все ее поддиректории

Формат команды: **DELTREE** [drive:]path

Замечание: невозможно удалить директорию, которая содержит текущую директорию, как одну из поддиректорий.

Смотрите также дополнительные замечания 3 и 4.

MF – создает файл

Формат команды: **MF** [drive:]path

Замечание: Если такой файл уже существует по указанному пути, приложению следует перейти к следующей команде без каких-либо уведомлений и формирования ошибок.

Примеры

MF Dir2\Dir3\file.txt – создает файл с именем file.txt в текущей директории «Dir2\Dir3».

MHL – создает жесткую связь к файлу/директории и размещает ее по заданному пути

Формат команды: **MHL** [drive:]source [drive:]destination

Замечание: Путь Destination обязан содержать только путь, без указания какого-либо имени файла. Соответствующая связь будет создана, как указано в замечании 10. Если такая связь уже существует, тогда приложению следует перейти к следующей команде без каких-либо уведомлений и формирования ошибок.

Примеры

a) **MHL** C:\Dir2\Dir3\file.txt C:\Dir4 – создает жесткую связь к файлу «file.txt» и помещает ее в директорию «C:\Dir4»

b) **MHL** C:\Dir1\Dir4 C:\Dir2 – создает жесткую связь к директории «C:\Dir1\Dir4» и помещает ее в директорию «C:\Dir2»

MDL – создает динамическую связь к файлу/директории и помещает ее по заданному пути

Формат команды: **MDL** [drive:]source [drive:]destination

Замечание: Путь Destination обязан содержать только путь, без указания какого-либо имени файла. Соответствующая связь будет создана, как указано в замечании 10. Если такая связь уже существует, тогда приложению следует перейти к следующей команде без каких-либо уведомлений и формирования ошибок.

DEL – удаляет файл или связь

Формат команды: **DEL** [drive:]path

Замечания: Смотрите замечания 3 и 4.

Примеры

DEL C:\Dir2\Dir3\file.txt – удаляет файл «file.txt» из директории «C:\Dir2\Dir3».

COPY – копирует существующую директорию/файл/связь по заданному пути.

Формат команды: **COPY** [drive:]source [drive:]destination

Замечание: Программе копирует директорию со всем ее содержимым. Путь Destination не должен содержать никаких имен файлов, в противном случае приложению следует формировать сообщение об ошибке.

Примеры

a) `COPY Dir2\Dir3\ C:\Dir1` - копирует из текущей директории поддиректорию Dir2\Dir3\ в C:\Dir1.

b) `COPY Dir2\Dir3\file.txt C:\Dir1` - копирует файл «file.txt» из текущей директории из поддиректории Dir2\Dir3\ в C:\Dir1.

c) `COPY Dir2\Dir3\file.txt C:\Dir1\newfile.txt` - выполнение команды вызовет ошибку

MOVE – переносит существующую директорию/файл/связь по заданному пути

Формат команды: `MOVE [drive:]source [drive:]destination`

Замечания: Приложение переносит директорию со всем ее содержимым. В случае, когда директорию, которую следует перенести, содержит жесткие связи, приложению следует прекратить выполнение команды MOVE и всего пакетного файла.

В случае, когда найдена динамическая связь/связи и жестких связей не существует, тогда следует модифицировать динамическую связь/связи и она/они будет содержать модифицированное содержимое относительно старой.

Примеры

a) Предположим имеется следующая структура директорий

```
C:
|_ DIR1
|_|_ DIR2
|_|_|_ DIR3
|_|_|_ readme.txt
|_|
|_|_ EDIR4
|_|_|_ hlink[C:\DIR1\DIR2\DIR3\readme.txt]
|_|
|_|_ GDIR5
|_|_|_ dlink[C:\DIR1\DIR2\DIR3\readme.txt]
```

И приложение находит команду `MOVE C:\DIR1\DIR2\DIR3 C:\DIR1\EDIR4` в пакетном файле.

В соответствии требованиям, приложение следует прекратить выполнение, т.к. существует жесткая связь `hlink[C:\DIR1\DIR2\DIR3\readme.txt]`.

b) Предположим имеется следующая структура директорий

```
C:
|_ DIR1
|_|_ DIR2
|_|_|_ DIR3
|_|_|_ readme.txt
|_|
|_|_ EDIR4
|_|
|_|_ GDIR5
|_|_|_ dlink[C:\DIR1\DIR2\DIR3\readme.txt]
```

После команды `MOVE C:\DIR1\DIR2\DIR3 C:\DIR1\EDIR4` структура директорий и `dlink` будет изменена на следующую

```
C:
|_ DIR1
|_|_ DIR2
|_|
|_|_ EDIR4
|_|_|_ DIR3
|_|_|_ readme.txt
|_|
|_|_ GDIR5
|_|_|_ dlink[C:\DIR1\EDIR4\DIR3\readme.txt]
```

Дополнительные замечания

1. Начальная файловая система содержит только корневую директорию обозначенную как «C:».
2. Имена команд, файлов и директорий являются регистронезависимыми. Команды Cd, CD, Cd, cD обозначают одно и тоже.
3. Файл не может быть удален, если он обладает жесткой связью и может быть удален в случае динамической связи.
4. Если файл удаляется, что все его динамические связи удаляются также. Если файл обладает как жесткой, так и динамической связью, приложению следует их оставлять без изменений.
5. Имена файлов и директорий обязаны удовлетворять следующему соглашению:
 - a. Имена файлов или директорий не должны превышать 8 символов в длину.
 - b. Расширения файлов не должны превышать 3 символов в длину.
 - c. Допускаются только буквенные [a...z] и цифровые [0...9] символы в именах файлов или директорий и расширений.
6. Никакие команды кроме `cd` не могут изменять текущую директорию.
7. Если путь не содержит префикса «C:\» тогда приложению следует выполнять действия в текущей директории.
8. В случае, если происходит любая ошибка, приложению следует прекратить свою работу и вывести понятное сообщение об ошибке.
9. Вывод следует организовывать в алфавитном порядке (по возрастанию) и приведенном ниже.
10. Формат вывода динамических и жестких связей: `hlink[full path]` и `dlink[full path]`, соответственно.
11. Не допускается создавать связи для связей.

```
C:
|_DIR1
|_|_DIR2
|_|_|_DIR3
|_|_|_readme.txt
|_|
|_|_EDIR4
|_|_|_temp.dat
|_|
|_|_GDIR5
|_|_|_hlink[C:\Dir1\EDir\4temp.dat]
```

Простой пример

Входные данные для примера:

```
MD Dir1
MD Dir1\Dir2
CD Dir1
MD EDir4
MD GDir5
MD Dir2\Dir3
MF Dir2\Dir3\readme.txt
CD EDir4
MF temp.dat
```

Результат:

```
C:
|_DIR1
|_|_DIR2
|_|_|_DIR3
|_|_|_readme.txt
|_|
|_|_EDIR4
|_|_|_temp.dat
|_|
|_|_GDIR5
```

Требования

- Архитектура приложения строится по модульному принципу.
- За основу принимается стандартная библиотека C++ (в случае разработки на C++).
- Рекомендуется использовать защищенные ресурсы и указатели.

- Обязательным является обработка исключений.
- Исходный код обязан быть комментирован. Для C# следует использовать нотацию Microsoft [3], для C++ следует использовать нотацию doxygen [2].

Методические указания для самостоятельной работы

При подготовке к лабораторной работе необходимо:

- Ознакомиться с понятиями ввода-вывода в ОС.
- Ознакомиться с контейнерами в C++/C#. Например:
 - Map, list, vector.
 - Dictionary, SortedDictionary, List, SortedList.

Тестирование

Тестирование приложения осуществляется в несколько этапов:

- Разработка Unit Test в рамках проекта. Данный подход позволит проверить корректность реализации алгоритма.
- Воспользоваться тестовыми пакетными файлами для проверки корректности функционирования.
- Разработать простейшее приложение, которое формирует большие тестовые пакетные файлы, которые затем использовать для тестирования основного приложения.

Теоретические сведения

Необходимая информация по Windows платформе доступна в [1].

Литература

1. Microsoft Developer Network. URL: <http://www.msdn.com>
2. Формирование документации к исходному коду с помощью средства doxygen. URL: www.nrjetix.com/r-and-d/lectures