
设计文档

1. 基本术语说明

使用C++编写一个运行在命令行模式下的，虚拟磁盘的软件。虚拟磁盘软件能够在内存中模拟一个磁盘，通过接受命令，可以在内存中完成一些文件操作的功能。

1.1 要求

- 使用VS作为编译工具（Vs2008、2010、2013、2015均可），代码使用SVN管理
- 可以使用任意版本的STL、可以使用最新的C++标准。
- 每个工作日必须上传当日代码带SVN，且上传的代码必须能够通过编译，不能有任何错误和警告。
- 使用面向对象的c++。 使用3种以上设计模式进行编写。
- 运行期间或者程序结束时，不得出现异常退出。不得有内存泄露。
- 要有足够的健壮性和容错性。
- 所有的路径要能够支持中文、支持正反斜杠

2. 基本词汇表

2.1 路径

- `相对路径`：相对路径指的是相对当前路径的路径。例如：当前目录是C:\a\b\c，那么../abc 表示的是C:\a\b\abc目录。
- `绝对路径`：绝对路径指的是以盘符开始的路径，例如C:\a\b\c\1.txt

2.2 符号链接

- `软连接`：符号链接（软链接）是一类特殊的文件， 其包含指向其它文件或者目录的引用。其他信息详见维基百科的【符号链接】词条，以及windows的mklink命令。

2.3 通配符

- *：星号⭐表示路径中的0个或多个字符
- ?：问号(?)表示路径中的1个字符。

3. 概述

在内存中模拟 Windows CMD 命令行的部分操作

使用 map<string, FileBase> 暂存数据，整体是一个嵌套的，树形的结构

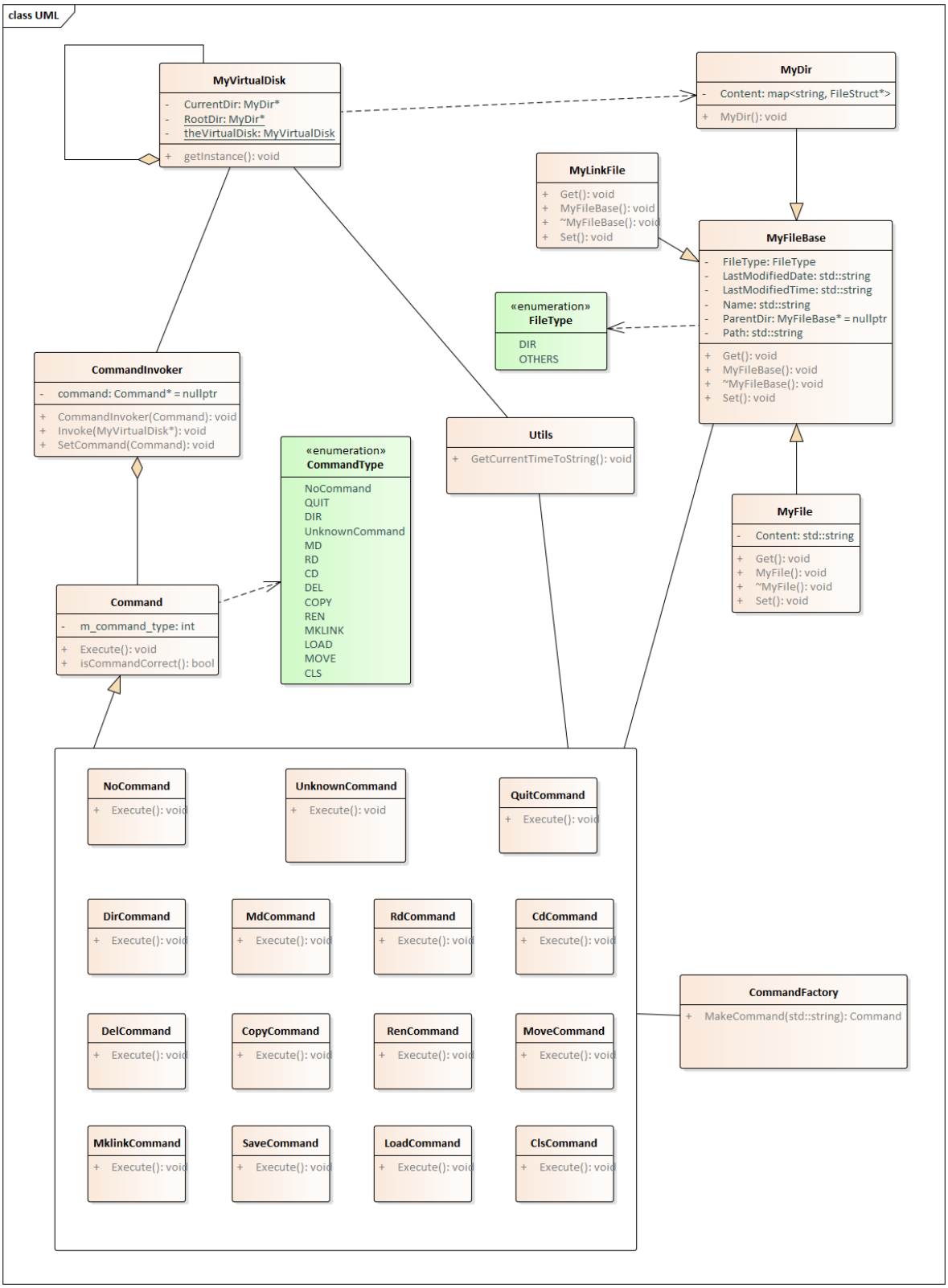
部分命令需要递归的遍历这棵树，使用DFS

项目使用了三种设计模式，分别是 单例模式，工厂模式，命令模式

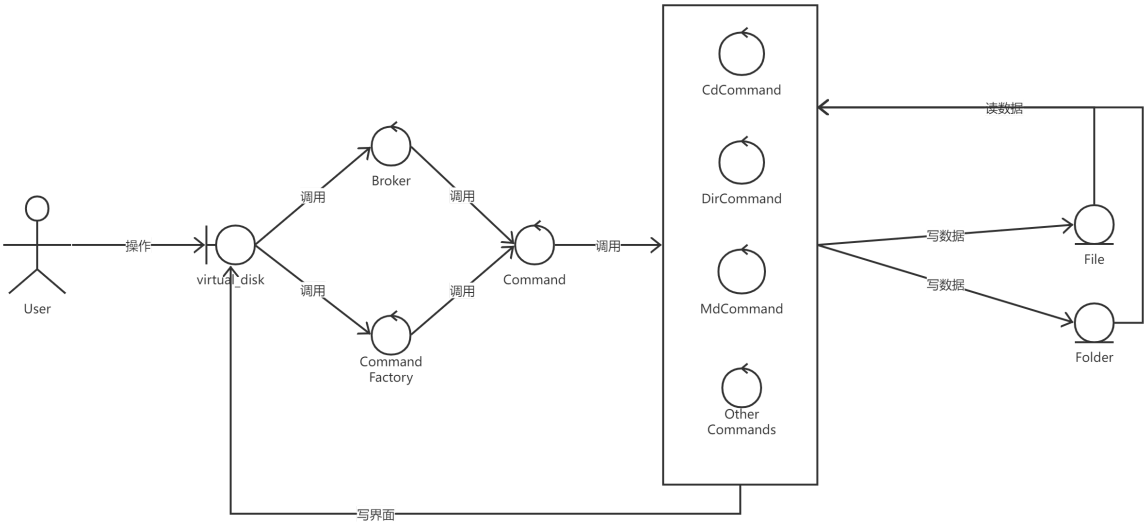
- 单例模式：使用在虚拟硬盘对象的实例化中，整个项目只有一个virtualDisk实例
- 工厂模式：使用工厂模式根据用户输入的不同的参数，生产不同的命令
- 命令模式：将工厂可以生产的命令进行封装，由invoker调用

4. 详细设计说明

4.1 UML 类图


class UML
MyVirtualDisk
- CurrentDir: MyDir*
- RootDir: MyDir*
- theVirtualDisk: MyVirtualDisk
+ getInstance(): void
MyDir
- Content: map<string, FileStruct*>
+ MyDir(): void
MyLinkFile
+ Get(): void
+ MyFileBase(): void
+ ~MyFileBase(): void
+ Set(): void
MyFileBase
- FileType: FileType
- LastModifiedDate: std::string
- LastModifiedTime: std::string
- Name: std::string
- ParentDir: MyFileBase* = nullptr
- Path: std::string
+ Get(): void
+ MyFileBase(): void
+ ~MyFileBase(): void
+ Set(): void
«enumeration»
FileType
DIR
OTHERS
CommandInvoker
- command: Command* = nullptr
+ CommandInvoker(Command): void
+ Invoke(MyVirtualDisk*): void
+ SetCommand(Command): void
Utils
+ GetCurrentTimeToString(): void
«enumeration»
CommandType
NoCommand
QUIT
DIR
UnknownCommand
MD
RD
CD
DEL
COPY
REN
MKLINK
LOAD
MOVE
CLS
MyFile
- Content: std::string
+ Get(): void
+ MyFile(): void
+ ~MyFile(): void
+ Set(): void
Command
- m_command_type: int
+ Execute(): void
+ isCommandCorrect(): bool
NoCommand
+ Execute(): void
UnknownCommand
+ Execute(): void
QuitCommand
+ Execute(): void
DirCommand
+ Execute(): void
MdCommand
+ Execute(): void
RdCommand
+ Execute(): void
CdCommand
+ Execute(): void
CommandFactory
+ MakeCommand(std::string): Command
DelCommand
+ Execute(): void
CopyCommand
+ Execute(): void
RenCommand
+ Execute(): void
MoveCommand
+ Execute(): void
MklinkCommand
+ Execute(): void
SaveCommand
+ Execute(): void
LoadCommand
+ Execute(): void
ClsCommand
+ Execute(): void

4.2 鲁棒图



5. 接口与数据结构设计

5.1 内存数据结构

- 文件基类

```
class MyFileBase
{
public:
    MyFileBase();
    MyFileBase(std::string name, std::string path, FileType file_type);
    virtual ~MyFileBase();

    // get set
    std::string GetName() const;
    void SetName(std::string name);
    std::string GetPath() const;
    void SetPath(std::string path);
    FileType GetType() const;
    void SetType(FileType file_type);
    std::size_t GetSize() const;
    void SetSize(std::size_t size);
    std::string GetLastModifiedTime() const;
    void SetLastModifiedTime(std::string last_modified_time);

    MyFileBase & GetParentDir() const;
    void SetParentDir(MyFileBase *parent_dir);

private:
    std::size_t m_size = 0; //文件大小
    std::string m_name = ""; //名字
    std::string m_path = ""; // 绝对路径
    FileType m_type = FileType::OTHER; // 文件类型

    std::string m_last_modified_time = Utils::GetNowTimeToString(); // 最新修改时间

    std::unique_ptr<MyFileBase> m_parent_dir = nullptr; // 父目录
};
```

- 文件夹子类

```
class MyDir
    : public MyFileBase
{
public:
    MyDir();
    MyDir(std::string name, std::string path, FileType file_type, MyDir
*parent_dir);
    virtual ~MyDir();

private:
    std::map<std::string, MyFileBase *> m_children; // 子目录
};
```

- 文件子类

```
class MyFile : MyFileBase
{
public:
    MyFile();
    virtual ~MyFile();
private:
// 还未实现
};
```

- 文件类型枚举类

```
enum class FileType
{
    DIR,
    OTHER
};
```

6. 配置说明

无

7. 数据库设计

无

8. 风险评估

无

9. 拓展讨论

无

10. 工作量评估

框架实现已经完成，耗时3天

一共15个命令，预计每天完成3个，一共5天

任务	预计完成时间（天）	实际耗时（天）
需分设计	2	3
框架实现	3	3
命令部分	5	/
自测	2	/