

FileSystem

- 使用说明
 - 基础系列
 - smart系列
- 文件结构
- 重要的类
 - Block
 - 功能简介
 - 主要代码
 - 持久化
 - BlockManager
 - 功能简介
 - 主要的代码
 - LogicBlock
 - 功能简介
 - 主要的代码
 - File
 - 功能简介
 - 主要代码
 - 持久化
 - FileManager
 - 功能简介
 - 主要代码
- 错误信息
 - 不能处理
 - 能处理
 - 未知错误

FileSystem

使用说明

一个文件管理系统，有简单的命令行工具。支持伸缩且有一定稳定性。

支持下列操作(^ω^)。

基础系列

- **help**: 在控制台打出操作教学
- **create fileId managerId**: 在指定的FileManager下创建名字是fileId的一个新文件（如果之前该名字的文件不存在）
- **get fileId managerId**: 找给定FileManager下叫fileId的文件（如果存在）并用于后续操作
- **size**: 必须先找到一个文件并处于操作状态，在操作台内打出文件大小
- **resize**: 必须先找到一个文件并处于操作状态，强制修改文件大小，多出来的位会变成‘0’
- **move off where**: 移动文件光标，**off**是移动的位数，**where**是开始的位置，0是现有位置，1是从头开始。
- **close**: 关闭现有文件，保存所做改动

smart系列

除了 **smart ls** 之外均需要选中一个文件能使用

- **smart cat**: 在控制台展示文件的所有内容
- **smart hex**: 在控制台展示文件的16进制内容
- **smart write**: 写入内容，两种用法
 1. **smart write content**: 直接在光标处写入内容
 2. **smart write content cursorPos**: 在给定的位置写入内容
- **smart copy fileId managerId**: 在指定的FileManager下创建名字是fileId的当前文件副本。
- **smart ls**: 树状展示各个文件的内容（具体到block）

文件结构

如下。

meta-data用于存储文件和**block**。

> out

▼ src

▼ Block

Ⓢ Block

Ⓢ BlockManager

Ⓢ LogicBlock

▼ Exception

⚡ ErrCode

▼ File

Ⓢ File

Ⓢ FileBuffer

Ⓢ FileManager

> Interface

▼ meta-data

▼ Utils

Ⓢ HashUtil

Ⓢ LoadUtil

Ⓢ PersistenceUtil

Ⓢ RandomUtil

Ⓢ FileSystem

Ⓢ Main

Ⓢ Tools

重要的类

许多类，比如FileSystem，Tools 和 Utils 文件夹下的各个工具类，起到的主要是辅助或包装的作用。虽然有的也有较重要的功能，但是依附于以下几个主要的类存在，因此不多做介绍。

- Block
- BlockManager
- LogicBlock
- File
- FileManager

接下来细说一下这几个类

Block

功能简介

单独的数据块，存了8个byte数据。block对象内不存储data，只提供信息用于找到对应.data文件。

新建块的时候会直接进行文件的写入。

主要代码

主要的method和属性如下

```
public class Block implements Interface.Block, Serializable {
    public static final int CAPACITY = 8;
    private final int id;
    private final BlockManager blockManager;
    private final int size;

    //两个constructor，分别用来(关闭重启后)加载之前的块和建立新的块
    public Block(int id, BlockManager blockManager) {}
    public Block(int id, BlockManager blockManager, byte[] content) {}

    /**
     * 根据block信息找到对应文件，检查checksum并读取block内容
     * @return 正常的话会返回block内容，如果block损坏或意外被修改或文件根本找不到
     则返回null
     */
    @Override
    public byte[] read() {}
}
```

持久化

会分为 id.meta 和 id.data 两个文件存储。

- data就是block内容本身没啥值得一说的。
- meta是两行分开的存储，第一行是block的size，第二行是哈希算法之后的checksum值。

BlockManager

功能简介

管理Block的类，主要功能有二

1. 在文件请求时分发新的的Block
2. 查找Block，与此同时维护了一个自己产生过的Block的list。

主要的代码

主要的method和属性如下

```
public class BlockManager implements Interface.BlockManager,
Serializable {
    private final int id;
    private final List<Block> blocks;

    //新建并初始化BlockManager (加载已经建立过的Block)
    public BlockManager(int id){}

    @Override
    //查找给定id的block
    public Block getBlock(int indexId) {}

    //给出一个由给定内容生成的块
    @Override
    public Block newBlock(byte[] content) {}
}
```

LogicBlock

功能简介

为了方便解释，接下来我们将之前的block称为物理块。

因为在本系统中统一文件下的每个块不一定在同一个Blockmanager下，有多个duplication且可能会丢失，所以在物理块和文件中做LogicBlock的一层封装。

每个逻辑块保存了所有的对应物理块，当文件寻找某个块的信息时会在物理块中找到完好的（通过比对data和meta的信息）并返回。

在新建时会把产生的duplicate分在不同的BlockManager下。

加入了修复功能，当有的块data坏了但是没全坏的时候将会复制好的来修改。但是暂时没法修复meta，因为来不及了(^ω^)

主要的代码

主要的method和属性如下

```
public class LogicBlock implements Serializable {
    public static final int BLOCK_CNT = 3;
    private final Block[] physicalBlocks;

    //新建logicblock, 会随机分到不同的manager下
    public LogicBlock( BlockManager[] managers, byte[] content ){}

    //读取块的内容
    public byte[] read(){}

    //修data
    private void fix(Block damagedBlock, byte[] rightContent){
    }
```

File

功能简介

文件类，维护了以下内容

- 对应逻辑块的List: blocks
- 存有文件内容的FileBuffer: buffer
- 文件大小: size

- 光标所在位置: cursorPos
- 存入时的修改开始位置: modifyBeginIndex

只有当文件被关闭的时候会将内容持久化存入。

主要代码

主要功能如下

```
public File(int id, FileManager fileManager, BlockManager[]
blockManagers) {
    //新建File
    public File(int id, FileManager fileManager, BlockManager[]
blockManagers) {}

    //读取已有的文件
    public File(FileManager fileManager, int fileId) {}

    //读取length的内容并移动光标
    @Override
    public byte[] read(int length) {}

    //写入content
    @Override
    public void write(byte[] content) {}

    //移动光标
    @Override
    public long move(long offset, int where) {}

    //关闭并保存文件
    @Override
    public void close() {}

    //强制修改文件大小, 如变大则在新的空间放0
    @Override
    public void setSize(long newSize) {}

    //将meta中的信息写入
    public void copy(HashMap<String, Object> meta){}

}
```

持久化

只存meta，data去一个一个块的找。

因为要存的东西较多且复杂，因此直接用ObjectWriter工具做持久化。将id，对应的BlockManagers，size，对应的LogicBlocks分别写入哈希表并序列化后写入以文件id命名的文件。

FileManager

功能简介

类似于BlockManager，主要功能有2

1. 寻找给定id的file
2. 当收到请求的时候给出file

主要代码

```
public class FileManager implements Interface.FileManager, Serializable
{
    private final int id;
    private final BlockManager[] blockManagers;
    private final List<File> files;

    @Override
    public File getFile(int fileId){}

    @Override
    public File newFile(int fileId){}
}
```

错误信息

不能处理

```
IO_EXCEPTION = 1
```

能处理

```
BLOCK_NOT_FOUND = 2 //找不到对应block  
FILE_NOT_FOUND = 3 //找不到对应文件  
BLOCK_DAMAGED = 4 //块受损 (meta, data信息对不上)  
INVALID_ARG = 5 //无效参数 (主要是移光标时传入错误类型参数)  
EOF = 6 //试图将光标移出文件  
FILE_ALREADY_EXIST = 7 //文件重名
```

未知错误

```
UNKNOWN = 1000
```