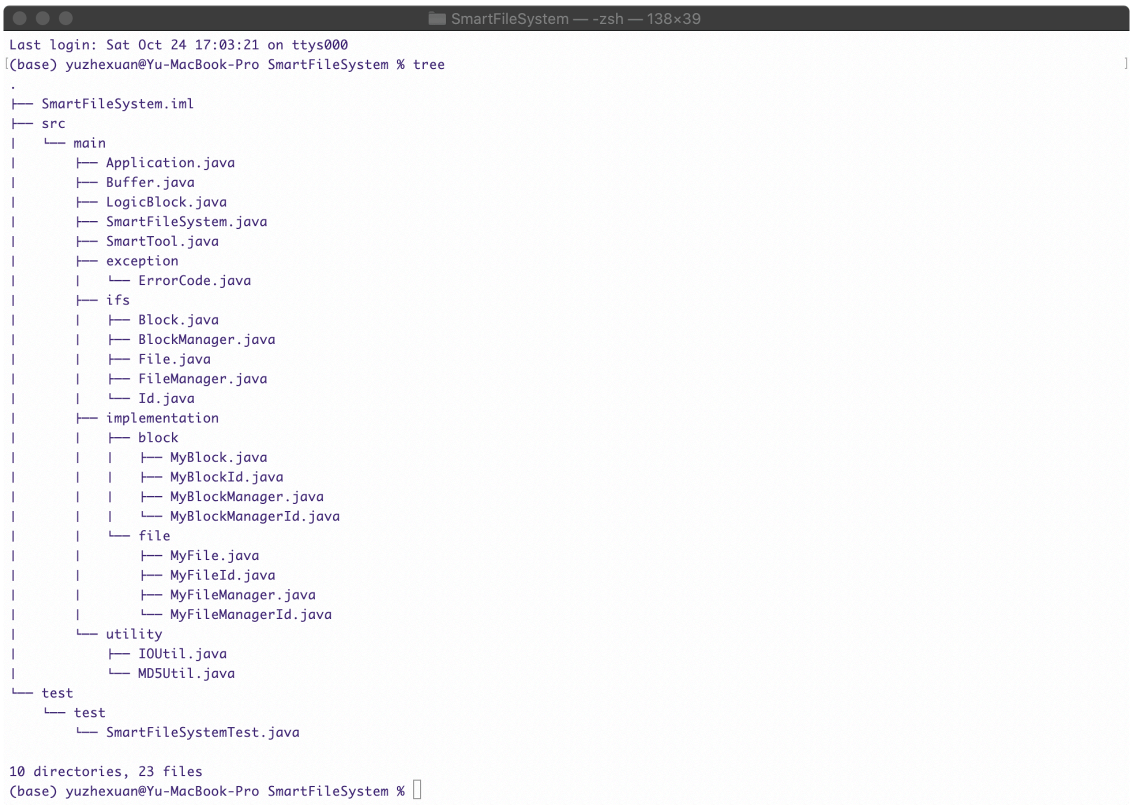
Computer System Engineering Lab1

> 18302010018 俞哲轩

项目结构

> `./src`为代码目录，`./test`为测试目录



```
Last login: Sat Oct 24 17:03:21 on ttys000
(base) yuzhexuan@Yu-MacBook-Pro SmartFileSystem % tree
.
├── SmartFileSystem.iml
├── src
│   └── main
│       ├── Application.java
│       ├── Buffer.java
│       ├── LogicBlock.java
│       ├── SmartFileSystem.java
│       ├── SmartTool.java
│       ├── exception
│       │   └── ErrorCode.java
│       ├── ifs
│       │   ├── Block.java
│       │   ├── BlockManager.java
│       │   ├── File.java
│       │   ├── FileManager.java
│       │   └── Id.java
│       ├── implementation
│       │   ├── block
│       │   │   ├── MyBlock.java
│       │   │   ├── MyBlockId.java
│       │   │   ├── MyBlockManager.java
│       │   │   └── MyBlockManagerId.java
│       │   └── file
│       │       ├── MyFile.java
│       │       ├── MyFileId.java
│       │       ├── MyFileManager.java
│       │       └── MyFileManagerId.java
│       └── utility
│           ├── IOUtil.java
│           └── MD5Util.java
└── test
    └── test
        └── SmartFileSystemTest.java

10 directories, 23 files
(base) yuzhexuan@Yu-MacBook-Pro SmartFileSystem % 
```

基础设计

> 包含类的主要字段和方法（自底向上）

`MyBlock`类：实现文件的物理储存

```
package main.implementation.block;

public class MyBlock implements Block, Serializable {
  private static final int CAPACITY = 512;
  private final MyBlockId id;
  private final MyBlockManager blockManager;
  private final String root;
  private final int size;

  public MyBlock(int id, MyBlockManager blockManager, String root)
  public byte[] read()
}
```

`MyBlockManager`类：管理自己名下的MyBlock

```
package main.implementation.block;

public class MyBlockManager implements BlockManager, Serializable {
  private final MyBlockManagerId id;
  private final String root;
  private final List<MyBlock> blocks;

  public MyBlockManager(int id, String root)
  public Block getBlock(Id indexId)
  public Block newBlock(byte[] b)
  // init()用于每次启动file system加载已有block
  private void init()
}
```

`MyFile`类：实现文件的读写相关的操作方法

```
package main.implementation.file;

public class MyFile implements File, Serializable {
  private final MyFileId id;
  private final MyFileManager fileManager;
  private final String root;
  private List<LogicBlock> blocks;
  private MyBlockManager[] blockManagers;
  private Buffer buffer;
  private long size;
  private long cursor;
  private long minModifyIndex;

  // 重载构造方法用于新建文件和恢复已有文件
  public MyFile(String id, MyFileManager fileManager, MyBlockManager[]
blockManagers, String root)
```

```
    public MyFile(MyFileManager fileManager, String root)
    public byte[] read(int length)
    public void write(byte[] b)
    public long move(long offset, int where)
    public void close()
    public void setSize(long newSize)

    // save()用于保存file meta
    private void save()
    // init()用于每次启动file system为已有文件分配buffer
    private void init(Buffer buffer, List<LogicBlock> blocks)
    // flush()用于file close的时候一次性写回所有内容
    private void flush()
}
```

`MyFileManager`类：管理自己名下的MyFile

```
package main.implementation.file;

public class MyFileManager implements FileManager, Serializable {
    private final MyFileManagerId id;
    private final MyBlockManager[] blockManagers;
    private final String root;
    private final List<MyFile> files;

    public MyFileManager(int id, MyBlockManager[] blockManagers, String root)
    public File getFile(Id fileId)
    public File newFile(Id fileId)
    // init()用于每次启动file system加载已有file
    private void init()
}
```

`SmartFileSystem`类：集成形成文件管理系统

```
package main;

public class SmartFileSystem {
    private static final int FILE_MANAGER_NUMBER = 3;
    private static final int BLOCK_MANAGER_NUMBER = 3;
    public final SmartTool tool;
    private final String root;
    private final MyFileManager[] fileManagers;
    private final MyBlockManager[] blockManagers;

    public SmartFileSystem()
    // init()用于每次启动file system构造file manager和block manager
    private void init()
    // getFile()用于用户获取文件
    public main.ifs.File getFile(int fileManagerNumber, String fileId)
    // newFile()用于用户新建文件
    public main.ifs.File newFile(int fileManagerNumber, String fileId)
}
```

程序逻辑

> 用户操作（自顶向下）

用户通过`Application`类启动程序，Application中集成了`SmartFileSystem`的实例，通过调用`getFile()`和newFile()方法，获取文件并对文件进行操作，再关闭文件之后，系统对每一个`block`自动选择`block manager`并由`block manager`负责将`block`进行物理储存

异常处理

异常代码

> 错误代码形如”1xx“的异常，是由程序运行错误引起的，且无法被解决，将会被抛出且中断程序
>
> 错误代码形如”2xx“的异常，是由文件错误引起的，将会调用处理程序，提示相关错误信息，且不中断程序
>
> 错误代码形如”3xx“的异常，是由用户输入参数不正确引起的，将会调用处理程序，提示相关错误信息，且不中断程序
>
> 错误代码形如”10xx“的异常，是由系统错误引起的，且无法被解决，将会被抛出且中断程序

```
// 1xx: exception and cannot be handled by software
public static final int IO_EXCEPTION = 101;
public static final int CLASS_NOT_FOUND_EXCEPTION = 102;
public static final int FILE_NOT_FOUND_EXCEPTION = 103;
public static final int MD5_INIT_FAILED = 104;

// 2xx: file error and can be handled by software
public static final int CHECKSUM_CHECK_FAILED = 201;

// 3xx: user unexpected input and can be handled by software
public static final int FILE_NOT_FOUND = 301;
public static final int FILE_ALREADY_EXIST = 302;
public static final int INVALID_ARGUMENT = 303;
public static final int END_OF_FILE = 304;
public static final int INVALID_CURSOR_MOVE = 305;
public static final int BLOCK_NOT_FOUND = 306;
public static final int MANAGER_NOT_EXIST = 307;
public static final int NOT_OPERATING_FILE_NOW = 308;

// 10xx: system error
public static final int SYSTEM_ERROR = 1000;
public static final int UNKNOWN = 1001;
```

异常解决和系统决策

1. 对于`IOException`、`ClassNotFoundException`、`FileNotFoundException`、`NoSuchAlgorithmException`、`SystemError`以及Unknown六大类的异常和错误，程序不会试图去解决，会抛出异常并将中断程序
2. 对于MD5函数校验失败，即同一block和其所有备份都损坏的情况，系统会打印错误信息，停止读取块内容并返回，不会中断程序
3. 对于获取不存在的file，系统会打印错误信息，返回读取失败（null值），不会中断程序
4. 对于新建已经存在file，系统会打印错误信息，返回新建失败（null值），不会中断程序
5. 对于读取文件内容时，读取长度设为负值，系统会打印错误信息，停止读取文件内容并返回，不会中断程序
6. 对于读取文件内容时，读取长度最终会导致光标超过文件长度，系统会打印错误信息，停止读取文件内容并返回，不会中断程序
7. 对于光标移动错误，即超过文件长度或是光标成为负值，系统会打印错误信息，光标不移动并返回光标错误值（-1值），不会中断程序
8. 对于读取一个不存在的块，系统会打印错误信息，停止读取块内容并返回，不会中断程序
9. 对于使用一个不存在的管理器，不论是file manager或是block manager，系统会打印错误，停止调用方法并返回，不会中断程序
10. 对于console使用时，并未打开文件便使用文件读写等操作方法时，系统会打印错误，停止调用方法并返回，不会中断程序
11. 光标变动：一般读写时，光标正常移动；文件大小变大时，光标保持原来位置不动；文件大小变小时，光标置零，即回到初始位置
12. `smart_cat()`、`smart_hex()`、`smart_write()`、`smart_copy()`四个方法允许用户不打开文件，直接对文件进行操作：`smart_cat()`打印文件的所有数据；`smart_hex()`以16进制打印块的数据；`smart_write()`在文件某位上写入数据；`smart_copy()`复制一个文件内容给另一个文件

More Design

buffer实现

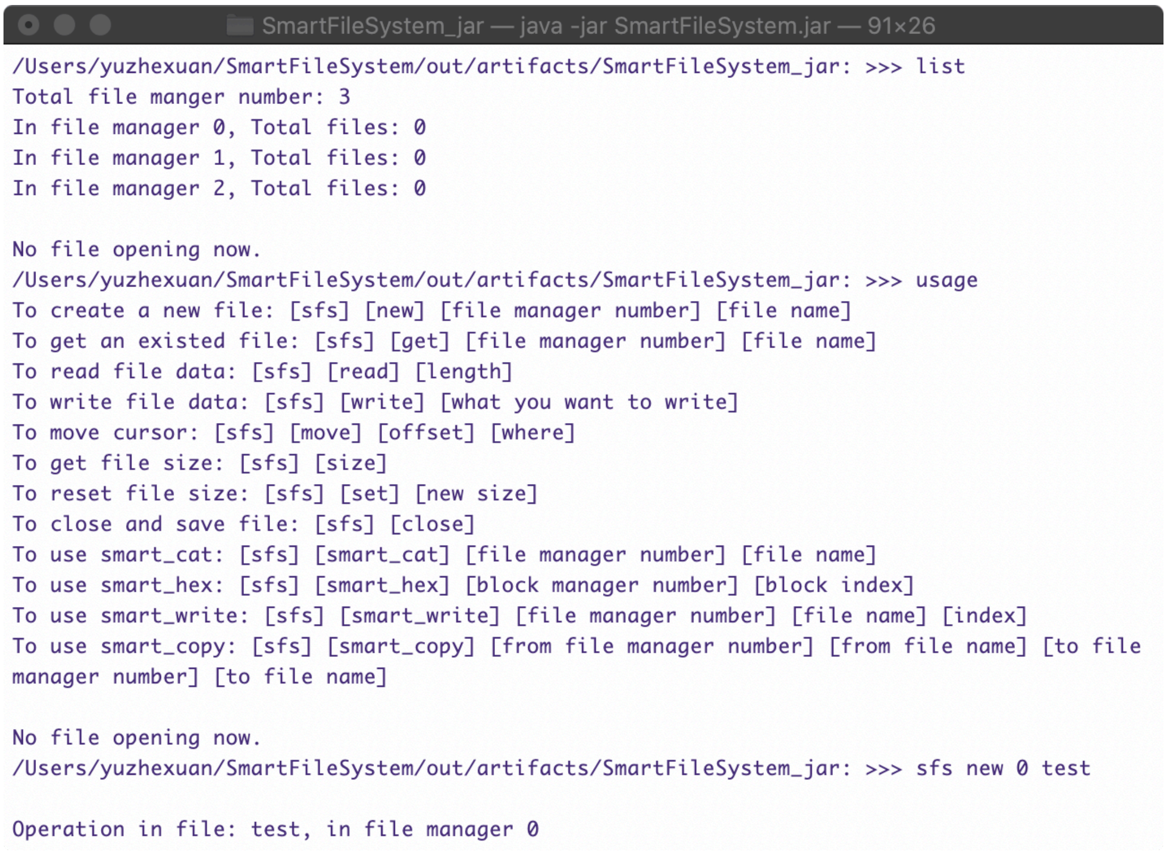
> 设计思路：
>
> ①buffer实现读写的加速：将读内容先检查是否在buffer中，如果在buffer中，直接读取buffer内容，否则读取block内容至buffer中，再从buffer中读取；将写内容先写到buffer，等buffer满了再一次性写回block中
>
> ②buffer大小的确定：如果小于一个block的大小，则与不使用buffer读写性能几乎没有差别，故buffer应该大于等于一个block的大小
>
> ③buffer的分配：真实系统中由顶层分配buffer，如一个文件系统统一维护一个buffer；在lab中，我们进行了合理的简化——每一个文件都会被系统分配到一个buffer
>
> ④buffer的申请：真实系统中如遇到buffer不够用，可以向顶层申请额外的buffer；在lab中，我们实现了类似的操作——当buffer不够用时，可以申请额外扩大buffer

```
package main;

```

```java
public class Buffer {
  private static final int TIMES = 10;
  private static final int CAPACITY = TIMES * MyBlock.getCAPACITY();
  private final byte[] buffer;
  private int size;

  public Buffer()
  public Buffer(int size)
  public Buffer(byte[] buffer, int size)
  public static Buffer allocate(Buffer oldBuffer, int enlargeSize)
  public static Buffer reshape(Buffer oldBuffer, int newSize)
  public void copy(byte[] src, int destPos, int length)
  public void copy(byte[] src, int srcPos, int destPos, int length)
  public boolean write(byte b)
  public void clear()
}
```

支持完全console操作

> Application支持完全console操作



```
/Users/yuzhexuan/SmartFileSystem/out/artifacts/SmartFileSystem_jar: >>> list
Total file manger number: 3
In file manager 0, Total files: 0
In file manager 1, Total files: 0
In file manager 2, Total files: 0

No file opening now.
/Users/yuzhexuan/SmartFileSystem/out/artifacts/SmartFileSystem_jar: >>> usage
To create a new file: [sfs] [new] [file manager number] [file name]
To get an existed file: [sfs] [get] [file manager number] [file name]
To read file data: [sfs] [read] [length]
To write file data: [sfs] [write] [what you want to write]
To move cursor: [sfs] [move] [offset] [where]
To get file size: [sfs] [size]
To reset file size: [sfs] [set] [new size]
To close and save file: [sfs] [close]
To use smart_cat: [sfs] [smart_cat] [file manager number] [file name]
To use smart_hex: [sfs] [smart_hex] [block manager number] [block index]
To use smart_write: [sfs] [smart_write] [file manager number] [file name] [index]
To use smart_copy: [sfs] [smart_copy] [from file manager number] [from file name] [to file
manager number] [to file name]

No file opening now.
/Users/yuzhexuan/SmartFileSystem/out/artifacts/SmartFileSystem_jar: >>> sfs new 0 test

Operation in file: test, in file manager 0
```