**华 南 农 业 大 学 数 学 与 信 息（软 件） 学 院**

《操作系统分析与设计实习》成绩单

开设时间：2016学年第一学期

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 小组成员、组内分工、工作量比例、各成员个人成绩 | | | | | | | | | | | |
| **学号** | 201430330113 | **姓名** | 蓝灿荣 | **分工** | 全部 | | | **工作量比例** | 100% | **成绩** |  |
| **学号** |  | **姓名** |  | **分工** |  | | | **工作量比例** |  | **成绩** |  |
| **学号** |  | **姓名** |  | **分工** |  | | | **工作量比例** |  | **成绩** |  |
| **学号** |  | **姓名** |  | **分工** |  | | | **工作量比例** |  | **成绩** |  |
| 实 验 题 目 | 模拟磁盘文件系统实现 | | | | | | | | | | |
| 自 我 评 价 | 这次操作系统课程设计由本人自己单独完成的。使用的是javascript electron写跨平台桌面应用，不用安装其他依赖(如java环境)，可以在linux,mac,windows(除xp)运行。但是由于学习新技术(electron,es6,nodejs,vuejs)的成本高，可能造成没有完成功能的情况，遇到的bug和无助也挺多的，但是学到了挺多东西。 | | | | | | | | | | |
| 教 师 评 语 | 评价指标：   * 题目内容和要求完成情况 优 □ 良 □ 中 □ 差 □ * 对算法原理的理解程度 优 □ 良 □ 中 □ 差 □ * 程序设计水平 优 □ 良 □ 中 □ 差 □ * 程序运行效果及正确性 优 □ 良 □ 中 □ 差 □ * 课程设计报告结构清晰 优 □ 良 □ 中 □ 差 □ * 报告中总结和分析详尽 优 □ 良 □ 中 □ 差 □ | | | | | | | | | | |
|  |  | | | | | **教师签名** | 张丽霞 | | | | |

1. 需求分析

设计一个简单的文件系统,用文件模拟磁盘,用数组模拟缓冲区,要求:

(1) 支持多级目录结构,支持文件的绝对读路径;

(2) 文件的逻辑结构采用流式结构,物理结构采用链接结构中的显式链接方式;

(3) 采用文件分配表 FAT;

(4) 实现的命令包括建立目录、列目录、删除空目录、建立文件、删除文件、显示

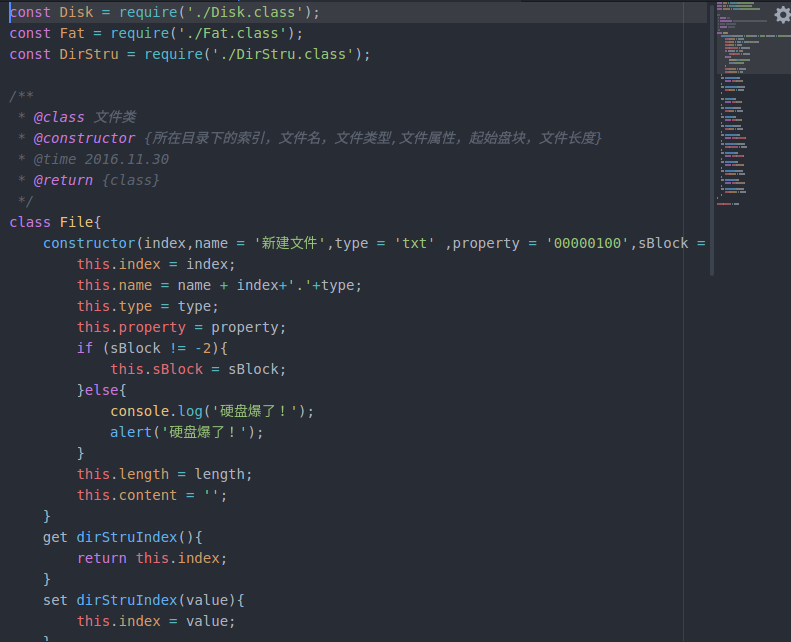
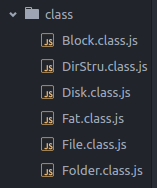
文件内容、打开文件、读文件、写文件、关闭文件、改变文件属性。可以采用

命令行界面执行这些命令,也可以采用“右击快捷菜单选择”方式执行命令。

(5) 最后编写主函数对所作工作进行测试。

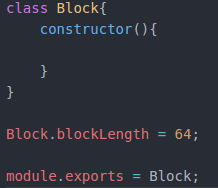
1. 概要设计

本程序主要使用了ECMAScript 6的Class类构建抽象数据类型。

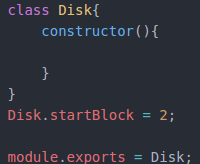


本程序设计了Block,Disk,DirStru,Fat,File,Folder类，分别是磁盘块静态类，磁盘静态类，目录结构类,Fat表静态类，文件类，文件夹类。

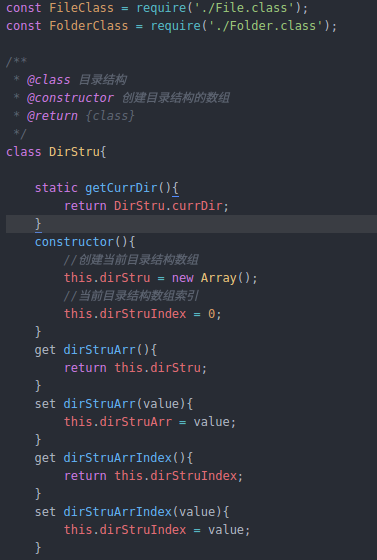
1. Block类



1. Disk类



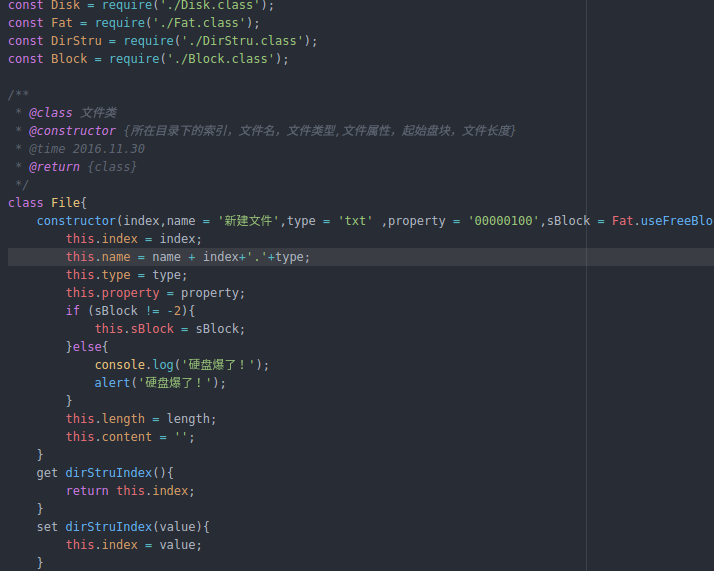
1. DirStru类

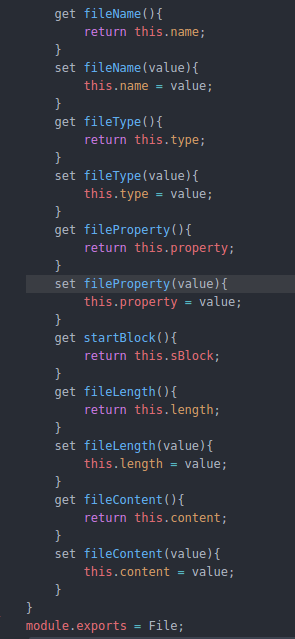


1. Fat类



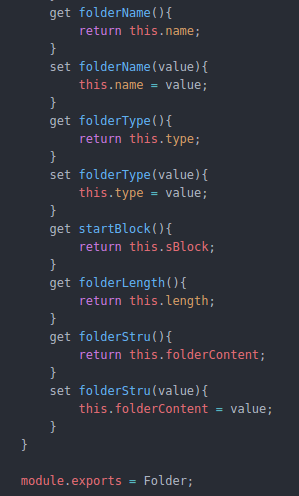
1. File类



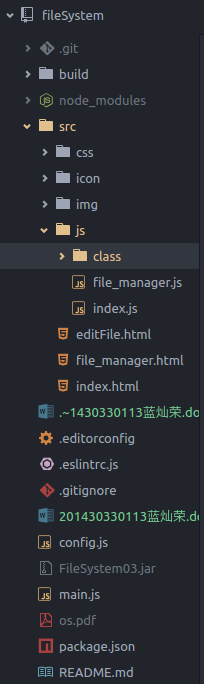


1. Folder类





目录结构：



主程序流程：

由于是用electron来做，所以在命令行运行了npm start之后，程序首先经过main.js，main.js启动界面(./src/index.html)进入./src/js/index.js，然后在界面双击“文件管理”就进入了文件管理系统界面(./src/file\_manager.html)，程序也随之进入了./src/js/file\_manager.js。于是对界面操作进可以对本程序定义的类进行操作。

各模块之间的层次调用：

以main.js为主进程，对于调用原生系统UI元素的子进程(如file\_manager.js)都要通过IPC进程通信跟main.js进行交互。然后子进程可以调用任何定义的抽象数据类型。

1. 详细设计
2. 界面

用electron和html来构建基于开源浏览器chromium的桌面应用，因为界面也可以使用html,css,javascript来写。主要使用了metro UI CSS这套UI框架来构建Win8风格的类操作系统的界面程序。

1. ECMAScript 6类
   1. Block

class Block{

constructor(){

}

}

Block.blockLength = 64;

module.exports = Block;

* 1. DirStru:

const FileClass = require('./File.class');

const FolderClass = require('./Folder.class');

/\*\*

\* @class 目录结构

\* @constructor 创建目录结构的数组

\* @return {class}

\*/

class DirStru{

static getCurrDir(){

return DirStru.currDir;

}

constructor(){

//创建当前目录结构数组

this.dirStru = new Array();

//当前目录结构数组索引

this.dirStruIndex = 0;

}

get dirStruArr(){

return this.dirStru;

}

set dirStruArr(value){

this.dirStruArr = value;

}

get dirStruArrIndex(){

return this.dirStruIndex;

}

set dirStruArrIndex(value){

this.dirStruIndex = value;

}

addFile(){

let file = new FileClass(this.dirStruArrIndex);

this.dirStruArrIndex += 1;

this.dirStruArr.push(file);

return file;

}

delFile(dirStruArrIndex){

this.dirStruArr[dirStruArrIndex] = null;

this.dirStruArr.splice(dirStruArrIndex,1);

}

addFolder(){

let folder = new FolderClass(DirStru,this.dirStruArrIndex);

this.dirStruArrIndex += 1;

this.dirStru.push(folder);

return folder;

}

delFolder(dirStruArrIndex){

this.dirStruArr[dirStruArrIndex] = null;

this.dirStruArr.splice(dirStruArrIndex,1);

}

}

DirStru.currDir = '/';

module.exports = DirStru;

* 1. Disk

class Disk{

constructor(){

}

}

Disk.startBlock = 2;

module.exports = Disk;

* 1. Fat

/\*\*

\* Fat表静态类，占用硬盘0,1块

\*/

class Fat{

/\*\*

\* 遍历fat表，返回空闲块的序号

\* @return int

\*/

static useFreeBlock(){

for (let i = 2; i < Fat.fatArr.length; i++){

if (Fat.fatArr[i] === 0){

Fat.fatArr[i] = 255;

return i;

}

}

return -2;

}

constructor(){

}

}

Fat.fatArr = new Array(128);

Fat.fatArr[0] = 255;

Fat.fatArr[1] = 255;

for (let i = 2; i < Fat.fatArr.length; i++){

Fat.fatArr[i] = 0;

}

module.exports = Fat;

* 1. File

const Disk = require('./Disk.class');

const Fat = require('./Fat.class');

const DirStru = require('./DirStru.class');

const Block = require('./Block.class');

/\*\*

\* @class 文件类

\* @constructor {所在目录下的索引，文件名，文件类型,文件属性，起始盘块，文件长度}

\* @time 2016.11.30

\* @return {class}

\*/

class File{

constructor(index,name = '新建文件',type = 'txt' ,property = '00000100',sBlock = Fat.useFreeBlock(),length = Block.blockLength){

this.index = index;

this.name = name + index+'.'+type;

this.type = type;

this.property = property;

if (sBlock != -2){

this.sBlock = sBlock;

}else{

console.log('硬盘爆了！');

alert('硬盘爆了！');

}

this.length = length;

this.content = '';

}

get dirStruIndex(){

return this.index;

}

set dirStruIndex(value){

this.index = value;

}

get fileName(){

return this.name;

}

set fileName(value){

this.name = value;

}

get fileType(){

return this.type;

}

set fileType(value){

this.type = value;

}

get fileProperty(){

return this.property;

}

set fileProperty(value){

this.property = value;

}

get startBlock(){

return this.sBlock;

}

get fileLength(){

return this.length;

}

set fileLength(value){

this.length = value;

}

get fileContent(){

return this.content;

}

set fileContent(value){

this.content = value;

}

}

module.exports = File;

* 1. Folder

const Disk = require('./Disk.class');

const Fat = require('./Fat.class');

const Block = require('./Block.class');

/\*\*

\* @class 文件夹类

\* @constructor {目录结构类(依赖),所在目录下的索引,文件夹名,文件夹属性,起始盘块,文件夹长度}

\* @time 2016.11.30

\* @return {class}

\*/

class Folder{

constructor(dependency,index,name = '新建文件夹',type = '00001000',sBlock = Fat.useFreeBlock(),length = Block.blockLength){

this.index = index;

this.name = name + index;

this.type = type;

if (sBlock != -2){

this.sBlock = sBlock;

}else{

console.log('硬盘爆了！');

alert('硬盘爆了！');

}

this.length = length;

this.folderContent = new dependency(); //由于循环依赖，强行使用依赖注入来使用DirStruClass

}

get dirStruIndex(){

return this.index;

}

set dirStruIndex(value){

this.index = value;

}

get folderName(){

return this.name;

}

set folderName(value){

this.name = value;

}

get folderType(){

return this.type;

}

set folderType(value){

this.type = value;

}

get startBlock(){

return this.sBlock;

}

get folderLength(){

return this.length;

}

get folderStru(){

return this.folderContent;

}

set folderStru(value){

this.folderContent = value;

}

}

module.exports = Folder;

1. 调试分析
2. 遇到的问题
   1. 不会调试electron桌面应用：搜索了好久，又问了师兄，再看electron的文档，解决了。
   2. Nodejs循环依赖问题：使用了依赖注入解决问题。
   3. 数据更新了，界面却没更新：Vuejs对于数组数据有特殊处理的方法，Google了一下和看了挺久的Vuejs文档，解决了。
   4. 在Linux下打包本程序为windows应用的时候出错：Linux下的wine没有装好。
3. 设计过程的经验和体会
   1. 由于使用新技术electron,vuejs,es6经验不是很充足，踩了好多坑，不过也学会了多看官方文档还有英文解释。
   2. 写的代码不够多，架构没有搭好，后面基本上是在补坑。
4. 实现过程中出现的主要问题及解决方法
   1. 主要是不熟悉文档，造成的bug：多看文档
   2. 架构不合理：重写了几次
5. 用户使用说明
6. 可执行程序

打开fileSystem文件夹，双击fileSystem.exe即可运行，进入主界面之后双击”文件管理“图标即可以进入文件管理系统。

1. 源程序

## **环境依赖（必须要安装这些才能调试）：**git: ****2.9.3，****nodejs: ****7.2.0，****npm: ****3.10.9****

## **使用说明**

## cd fileSystem

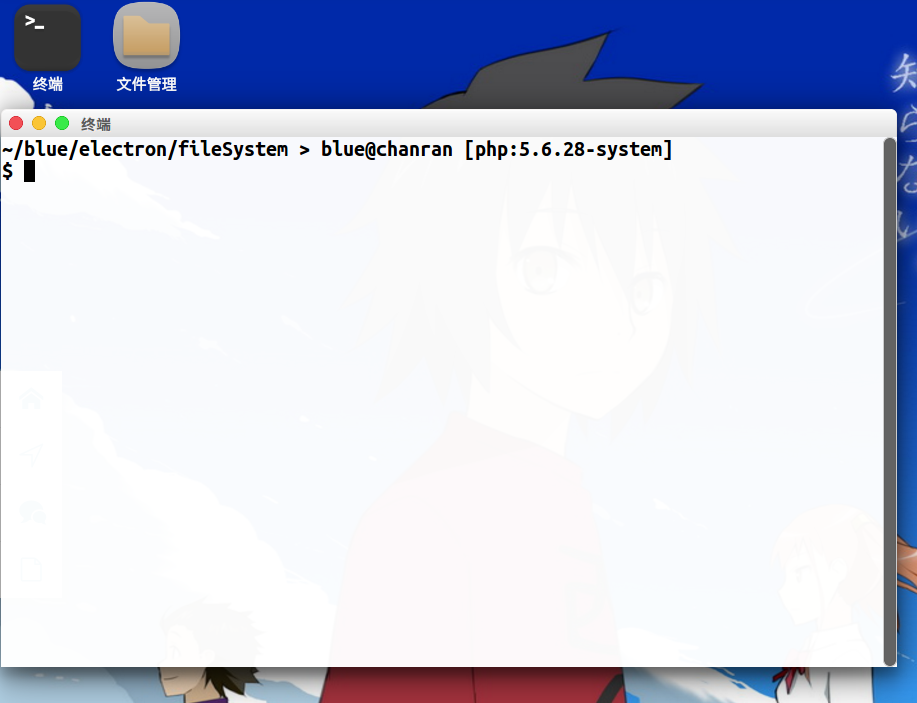
npm i d

npm start

1. 测试与运行结果
2. 界面
   1. 主界面



* 1. 双击终端可以调用Linux gome terminal



* 1. 点击关于我



* 1. 点击文件管理--文件管理系统



* 1. 在中间空白处右键新建文件或者文件夹



* 1. 双击文本文件

