**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：2014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** | 20143040407 | **姓名** | 郭逸坤 | **分工** | 文件 | | **工作量比例** | 22% | **成绩** |  |
| **学号** | 201430340404 | **姓名** | 陈涛 | **分工** | 设备 | | **工作量比例** | 20% | **成绩** |  |
| **学号** | 201430340418 | **姓名** | 谢宇琪 | **分工** | 内存 | | **工作量比例** | 20% | **成绩** |  |
| **学号** | 201430340409 | **姓名** | 黄明星 | **分工** | 进程管理 | | **工作量比例** | 24% | **成绩** |  |
| **学号** | 201430340408 | **姓名** | 黄茂鹏 | **分工** | 界面 | | **工作量比例** | 16% | **成绩** |  |
| 实验题目 | 题目六 模拟操作系统实现 | | | | | | | | | |
| 自我评价 |  | | | | | | | | | |
| 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** |  | | | |

1. **需求分析**

明确陈述说明程序设计的任务，强调的是程序要做什么，主要包括：

任务要求：模拟一个采用多道程序设计方法的单用户操作系统，该操作系统包括进程管理、存储管理、设备管理、文件管理和用户接口四部分。

1. 输入的形式和输入值的范围；

①从用户界面输入字符串形式的8个命令，如下：

1. 创建文件： create
2. 删除文件： delete
3. 显示文件： type
4. 拷贝文件： copy
5. 建立目录： mkdir
6. 删除空目录： rmdir
7. 进入退出某个目录:：cd
8. 新建可执行文件：newexe

➁create新建文件时，type打开文件，从用户界面输入文本内容，并点击保存

。目前仅支持第一次保存，暂不支持修改。

1. 输出的形式；

①进程管理部分

系统时间，相对时钟（时间片）、正运行进程ID、正在执行指令、进程中间执行结果、进程最终结果以及就绪队列、阻塞队列（均以字符串形式输出并显示于用户界面）

②存储管理部分

内存分区情况（列表形式输出显示）

③设备管理部分

1. B、C三种设备的使用情况和相应等待使用的进程ID（列表形式输出显示）

④文件管理部分

文件树状目录、文本文件的内容和磁盘使用情况（文件内容字符串形式输出显示，单个磁盘用红蓝小方块表示）

⑤用户接口部分

显示软件系统的界面

1. 程序所能达到的功能；

总体功能：实现一个采用多道程序设计方法的单用户操作系统，包括进程管理、存储管理、设备管理、文件管理和用户接口四部分。

①进程管理部分功能

1. 实现简单的CPU，包括指令的读取、解析、执行等
2. 实现系统时钟，控制CPU和系统的节拍
3. 实现主要的寄存器
4. 进程管理主要包括进程调度、进程的创建和撤销、进程的阻塞和唤醒， 中断作用的实现

②存储管理部分功能

1. 划分系统区和用户区
2. 实现内存空间的分配和回收、存储保护（采用动态分区存储管理和首次适配）

③设备管理部分功能

1. 实现设备的分配和回收，可实现进程的阻塞和唤醒，并管理等待设备的进程
2. 设备使用倒计时来表示占用某个设备

④文件管理部分功能

1. 实现了进入退出目录、新建文件、删除文件、复制文件、删除空目录、新建exe文件、读取文件。
2. 显示文件目录树
3. 显示磁盘使用情况，红色表示未被占用，蓝色表示已经被占用。

⑤用户接口部分

1. 将进程执行中的各种状态、数据以及内存、磁盘使用情况显示出来
2. 可输入各种文件命令，响应并显示各种文件命令操作

(4)测试数据：包括正确的输入及其输出结果和含有错误的输入及其输出结果。

①若可执行文件是：

X=12

X++

!A7

X--

!B5

end

模拟CPU在执行时，应能显示每一条指令、执行中间结果、最终结果以及对应进程ID；内存、进程队列、设备以及磁盘方面均应有相应的实时显示

②

创建文件

正确输入create gyk.txt，

正确输出 目录树更新，在该路径下生成gyk.txt。

错误输入 create gykk.txt

错误输出 文本域提示文件命名不正确

删除文件

正确输入 delete gyk.txt

正确输出 目录树更新，该路径下的gyk.txt消失，对应磁盘块变为红色

错误输入 delete gykk.txt

错误输出 文本域提示文件名不正确

错误输入 delete g.txt

错误输出 文本域提示文件不存在

显示文件：

正确输入type gyk.txt

正确输出 文本域显示文件内容，如果是第一次打开，输入内容后点击保存即可保存。磁盘块更新。

错误输入 type gykk.txt

错误输出 文本域提示文件名不正确

拷贝文件：

正确输入 copy gyk.txt aaa/bbb

正确输出 目录树更新，磁盘块更新，aaa/bbb下出现gyk.txt

错误输入 copy gyk.txt aaa/ccc

错误输出 文本域提示路径不存在

建立目录：

正确输入mkdir aaa

正确输出 目录树更新

错误输入 makdir aaaa

错误输出 文本域提示目录名错误

删除空目录：

正确输入 rmdir aaa

输入 如果该目录下没有文件，则可以删除，更新目录树，更新磁盘块；如果该目录下有文件或者目录，则提示该目录不为空。

进入退出某个目录:

正确输入cd aaa/cd ..

输入 命令行更新root>aaa:，root:,如果在root：情况下输入cd .. 则会提示错误

新建可执行文件：

正确输入 newexe eee.exe ,在文本框中写入指令，并点击保存

正确输出 目录树更新，磁盘块更新。

1. **概要设计**

说明本程序中用到的所有抽象数据类型的定义、主程序的流程以及各程序模块之间的层次 (调用)关系。

1. **详细设计**

实现概要设计中定义的所有数据类型，对每个操作只需要写出伪码算法；对主程序和其他模 块也都需要写出伪码算法(伪码算法达到的详细程度应能够按照伪码算法在计算机键盘上直接输 入高级程序设计语言程序)；画出函数的调用关系图。

***存储管理主程序伪代码：***

public class MemoryManage {

//内存分区表

//申请pcb总数count

//pcb表

//进程表

private static total\_Size total = new total\_Size();

private static int pcbCount = 10;

private static List<PCBManager> pcbs = new ArrayList<PCBManager>();

private static List<process> jobs = new ArrayList<process>();

}

public static boolean 申请空白PCB(PCBManager pcb){

if(申请count>10){

申请失败;

}

else{

申请成功;

}

}

public static int 请求分配内存(int 进程标识符,int 大小,String 文件路径) throws IOException{

//创建进程对象

//获取内存分区表

//按内存地址排序

for(<内存分区表总长度){

if(未分配){

if(分区大小>进程所需内存大小){

//内存申请成功

total.alloc(i, job);

//调用分配算法分配内存

}

}

}

if(>分区表总长度){

//内存申请失败

}

//返回内存起始地址 }

public static void 释放内存(int 起始地址){

//获取分区表

List<Size> lsize = total.getList();

int i ;

int index = -1;//标记符

for(i<内存分区表总长度){

if(找到匹配的内存的起始地址){

for(int j=0;j<进程表长度;j++){

if(找到匹配的进程标识符){

//回收分区

total.search(jobs.get(j));

jobs.remove(j);

//释放内存

Return;

}

}

}

}

//否则释放内存失败

}

public static boolean释放pcb(int index){

int j;

for(j<pcb总数量){

if(找到匹配的进程标识符){

//删除pcb

pcbs.remove(j);

System.out.println("进程控制块为"+j+"已删除");

break;

}

}

if(j>pcb总数量){

System.out.println("没有该进程控制块");

return false;

}

else

return true;

}

public static void sort(List<lsize> lsize){

//按内存地址排序算法

}

public static void memoryList(){

//打印分区信息

}

***其他模块伪代码:***

***单独分区模块伪代码:***

public class Size {

//start address

private int start;//起始地址

private int space;//空间大小

private int states = -1;//分配状态

private ArrayList<String> ins = new ArrayList<String>();

public Size(int start ,int space )}

public Size(int start, int space, int states){}

public int getStart(){}

public void setStart(int start){}

public int getSpace(){}

public void setSpace{}

public int getState(){}

Public void setState(){}

public void setIns(String filename) throws IOException{}

public String getIns(int i){}

***分区表代码***

public class total\_Size {

//分区表

private List<Size> block\_list = new LinkedList<Size>();

//min block

private int minSize = 0;

public total\_Size(){

Size size = new Size(0,512); // 分配内存

block\_list.add(size);//add

}

public List<Size> getList(){}

public void setList(List<Size> block\_list){}

public boolean JudgeSzie(int i {}

public void alloc (int index , process job) throws IOException {

//分配算法 首次适配

Size size = block\_list.get(index);

if(size.getSpace() - job.getSize() <= this.minSize){

size.setState(job.getIndex());

}

if(size.getSpace()-job.getSize()>this.minSize){

int newStart = size.getStart()+ job.getSize();

int newSpace = size.getSpace() - job.getSize();

Size newSize = new Size(newStart,newSpace);

block\_list.add(newSize);

size.setSpace(job.getSize());

size.setState(job.getIndex());

size.setIns(job.getFilename());

}

//return hole.getStartAddress();

}

//进程结束，回收分区

public void search(process job){

Size size =null;

int flag = -1 ;//记录当前位置

for(int i=0; i<block\_list.size();i++){

//找到被回收的分区

if(block\_list.get(i).getState() ==job.getIndex()){

// 设置自己为空闲

}

}

if(flag == -1){

return;

}

//当前分区的后面的分区地址 当前adrs+space

int nextAdrs = size.getStart() + size.getSpace();

//合并空闲分区

for(int i=0; i<block\_list.size() ; i++){

if(JudgeSzie(i)){

Size nowSize = block\_list.get(i);

if(nowSize.getStart()+nowSize.getSpace() == size.getStart()){

merge(i,flag);

}

}

}

}

public void merge(int i, int j){

//合并分区算法

i\_size.setSpace(i\_size.getSpace() + j\_siz.getSpace());

block\_list.remove(j);

}

***进程模块伪代码***

public class process {

private int size;

private int index;

private String filename;

public process (int index,int size, String filename){ //初始化 }

public int getIndex(){}

public void setIndex(int index){}

public int getSize(){}

public void setSize(int size){}

public String getFilename(){}

public void setFilename(String filename){}

设备

package com.os;

import java.util.LinkedList;

// 设备管理类

public class DeviceManagement {

// 单例

private static DeviceManagement instance;

// 设备的数量

private static final int TOTAL\_A = 2;

private static final int TOTAL\_B = 3;

private static final int TOTAL\_C = 3;

private LinkedList<DeviceStatus> usingList=new LinkedList<DeviceStatus>();

private LinkedList<DeviceStatus> waitingList=new LinkedList<DeviceStatus>();

private DeviceManagement (){}

// 获取类的实例

public static synchronized DeviceManagement getInstance() {

if (instance == null) {

instance = new DeviceManagement();

}

return instance;

}

// 一个进程运行的时候

public void OnProcessRun( Process pro, Device dev)

{

int devCount = 0;

DeviceStatus s = new DeviceStatus();

s.pro = pro;

s.dev = dev;

// 遍历使用队列，如果有可用的设备，则将信息加入到usingQueue中，如果没有可用的设备，则把信息添加到waitingQueue中

for (DeviceStatus x : usingList) {

if( x.dev.getName().compareTo(dev.getName())==0 )

{

devCount++;

if( dev.getName().compareTo("A")==0 ) {

// 如果大于了最大设备个数，则加入到等待队列

if( devCount>=TOTAL\_A ) {

waitingList.offer(s);

return;

}

}

else {

if( devCount>=TOTAL\_B || devCount>=TOTAL\_C ) {

waitingList.offer(s);

return;

}

}

}

}

// 如果没有超出最大数量，那么则使用此设备

usingList.offer(s);

}

// 一个进程结束的时候

public void OnProcessStop( Process pro )

{

Device reDevice = new Device();

// 设备使用完毕，清除相关条目信息，然后把等待队列中的进程加入到列表中

for (DeviceStatus x : usingList) {

if( x.pro.getName().compareTo(pro.getName())==0 ) {

reDevice = x.dev;

usingList.remove(x);

break;

}

}

// 从等待队列中把使用此设备的进程取出

for (DeviceStatus x : waitingList) {

if( x.dev.getName().compareTo(reDevice.getName())==0 ) {

usingList.offer(x);

waitingList.remove(x);

break;

}

}

}

// 显示目前设备的占用情况

public void ShowStatus() {

System.out.println("【设备使用情况】");

for (DeviceStatus x : usingList) {

System.out.println("进程：" + x.pro.getName() + " - 设备：" + x.dev.getName());

}

System.out.println("【等待队列】");

for (DeviceStatus x : waitingList) {

System.out.println("进程：" + x.pro.getName() + " - 设备：" + x.dev.getName());

}

}

}

文件模块



1. **调试分析**

内容包括：

1. 调试过程中遇到的问题是如何解决的以及对设计与实现的讨论和分析；
2. 算法的时间复杂性(包括基本操作和其他算法的时间复杂性的分析)和改进设想；
3. 设计过程的经验和体会；

设备：通过对这次操作系统课程设计的亲自参与和操作，使我深刻体会到了：只要你想做只要你想学没有弄不懂得事情，工程里面也不能不在乎细节，等等。感觉很受益匪浅。懂得了操作系统包括的四部分内容：文件管理和用户接口、存储管理、设备管理、进程管理之间的内在联系。加深了我对这门课程的理解。锻炼了自己在考虑全局也不是细节的能力。

文件：文件个人觉得是整个系统中最为复杂的部分，从看懂文档到慢慢了解，思考构造，花了很长的时间。考虑到对java界面不够熟悉，所以采用了用命令行来执行操作的方法。存储文件后要改变原目录中的起始盘块和文件分配表是比较难实现的步骤。

内存：本次的课程设计我负责的是存储管理部分，说实话，这其实对我来说算是比较大难度的课程设计，主要是因为本身的java程序设计并没有学的很扎实，其次对操作系统的许多课程知识也开始遗忘，一开始的确是无从下手，后来，通过询问班级里java程序设计厉害的同学，才开始有了思路。通过学习的知识，我知道要开始一个进程要先向内存申请分配内存空间，而进程结束后，要释放已分配的内存空间，而如何分配内存空间给所需要的进程又是依靠动态分区存储管理方式，采用首次适配、下次适配或者最佳适配算法来动态管理存储方式，而每个内存分区表都是由一个一个的分区组成，所以我就是沿着这个思路，由小及大开始解决问题。先从每个分区起手，定义一个分区的类（Size.class），类中包含每个分区起始地址、分区大小、分配状态的get/set方法，然后用链表结构List<Size>构成分区表，采用首次适配的分配方式来构造内存分配方法，并构造内存回收释放方法，接下是进程类（Process.class），该类包含进程控制块标识符、大小、文件所在路径的get/set方法，最后是主类实现从申请空白PCB块，请求分配内存，回收释放的全过程。

通过这次的课程设计，学会了更清晰、由小及大的思考方式，学会将大的问题逐步细分，分解成小的问题来思考，然后再从小的问题一步一步解决，最终解决大的问题，其次同学们也帮助了很多，给了我许多提醒和值得参考的方法，这次的课程设计实验收货的确不少，在巩固了所学的知识的同时，也学会了更进一步的实践，提高了实践能力，也能更深一层的了解操作系统的操作原理。

界面： 这次操作系统课程设计让我深刻体会到了UI设计的双面性.刚开始团队要求的java编写(我对java几乎了解甚少),后来通过百度查询运用了windowBuilder插件进行拖拉操作,后期修改部分代码完成了系统的界面操作。

1. 实现过程中出现的主要问题及解决方法。

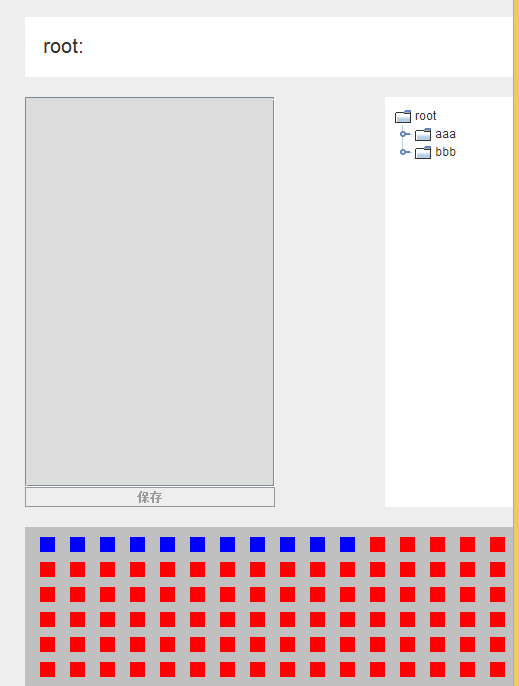
设备：在模拟设备管理的功能的过程中，很多次不可避免的碰到了死锁的问题，遇到过很多难题，经历过难题得不到解决的过程，但是通过查阅资料，收获不菲，很好地理解了银行家算法，达到了课程设计的目的，增强了动手能力。

文件：问题集中于如何模拟磁盘、如何存入文件，如何修改文件分配表，如何记录目录项，如何显示目录等等。模拟磁盘利用了byte 数组，保存到disk文件中，方便下次读取。目录的显示用了swing的jtree，通过遍历磁盘中的目录项添加到节点中，形成树。用byte数组模拟磁盘。再界面上输入内容后，提取内容转化为字节保存到byte数组中，解决了文件存储问题。

界面：JLabel无法显示背景颜色,后来用了Panel来代替使用.

1. **用户使用说明**

打开系统后，文件操作模块支持以下指令，用户可在root:后输入指令，每一步的操作仅限于当前目录下的内容。



1. cd，cd xxx进入某个目录，如果有错误，会在下面的灰色文本域内显示；cd .. 是返回上一层。
2. makdir,makdir xxx新建目录，仅支持最多3个字符，根目录只支持8个目录。如果有错误，会在下面灰色文本域内显示。
3. create,create xxx.txt建立文本文件，仅支持最多3个字符，根目录不能建立文件，如果有错误，会在下面灰色文本域内显示。
4. type，type xxx.txt打开文本文件，如果有错误，会在下面灰色文本域内显示。如果第一次打开文件，文件内没有内容可添加，并按保存按钮保存内容。暂不支持修改内容。
5. delete,delete xxx.txt删除文件，如果有错误，会在下面灰色文本域内显示。
6. rmdir,rmdir xxx删除空目录，如果有错误，会在下面灰色文本域内显示。暂不支持删除非空目录。
7. copy,copy xxx.txt xxx/xxx复制文件到某个目录（绝对路径），如果有错误，会在下面灰色文本域内显示。
8. Newexe,newexe xxx.exe新建可执行文件，仅支持最多3个字符，根目录不能建立文件，如果有错误，会在下面灰色文本域内显示。新建后下面灰色文本域会变为白色，可在里面输入指令，每次输入一条指令敲击一次回车。例如（x=1回车 x=2 回车end）。

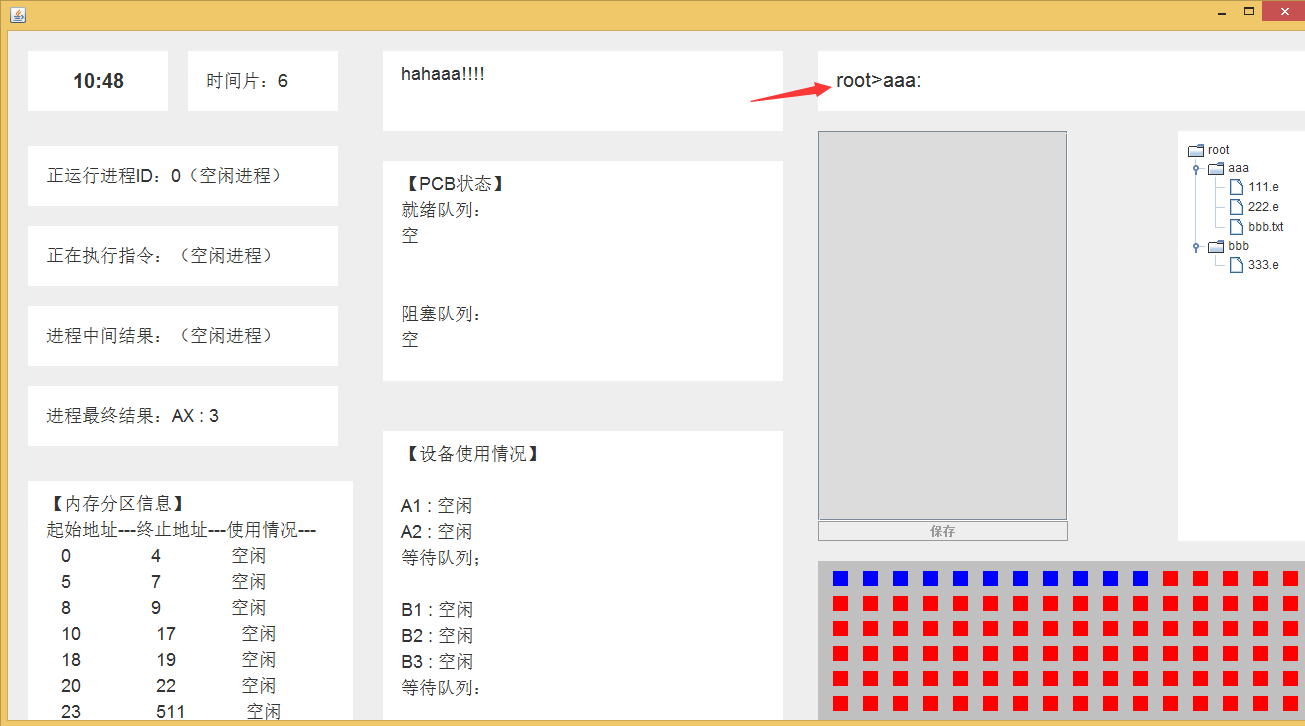
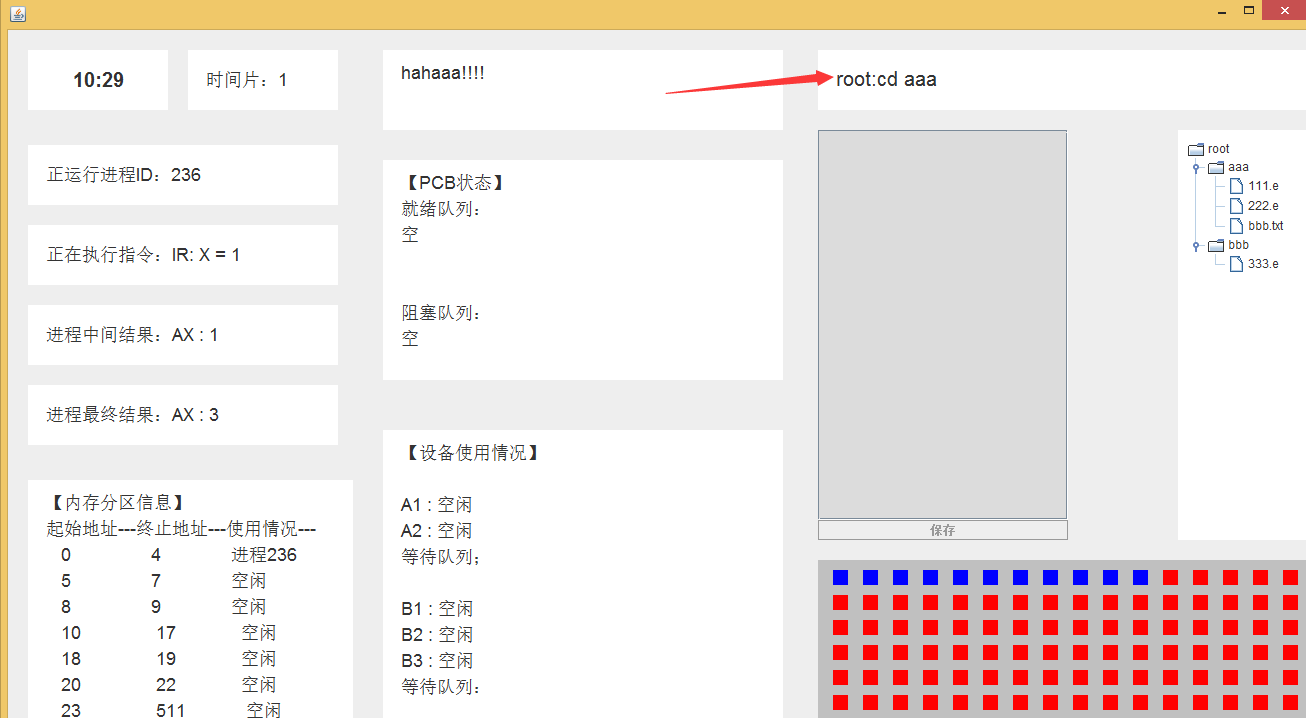
每次执行操作后，可以看到下方磁盘块发生变化，蓝色代表被占用，右侧是目录树可以查看当前目录状况。

1. **测试与运行结果**

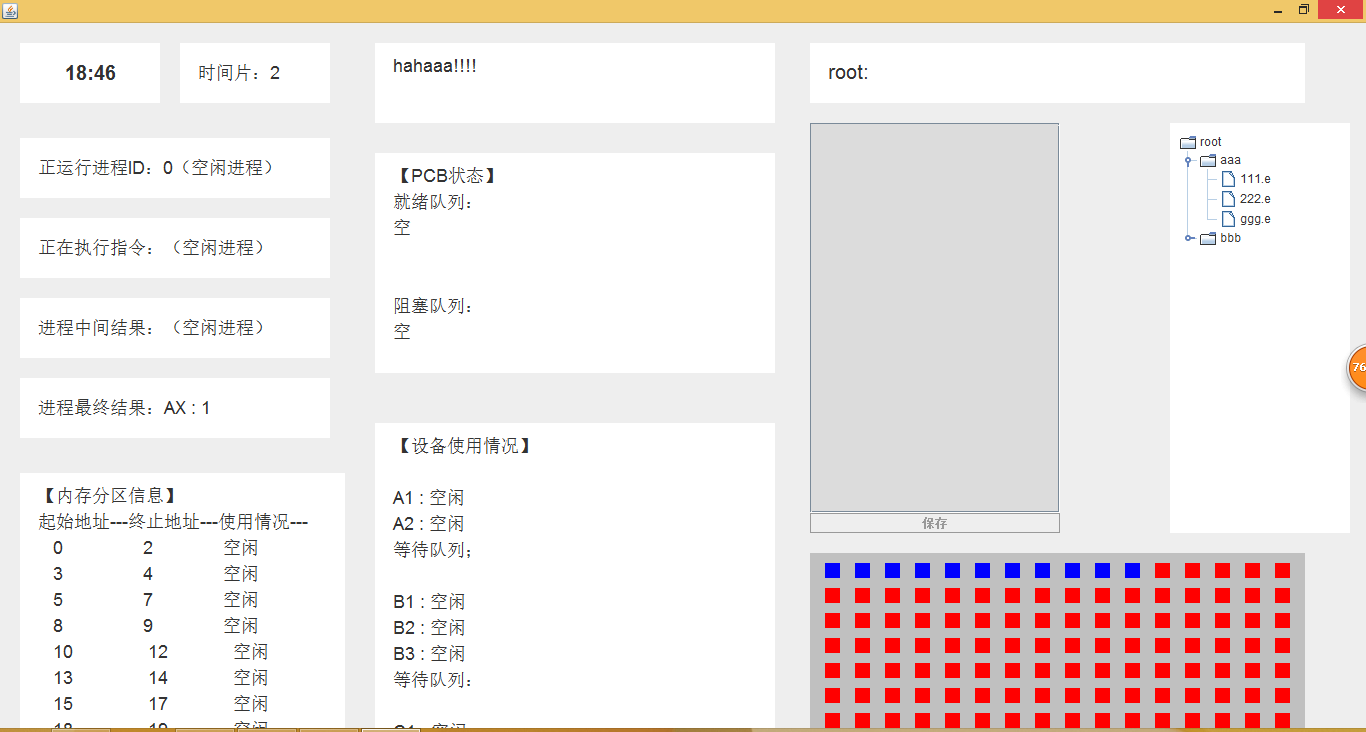
列出你的测试结果和运行情况（即运行时的关键画面），包括输入和输出。这里的测试数 据应该完整和严格，最好多于需求分析中所列。

值得注意的是，报告的各种文档资料，要在程序开发的过程中逐渐充实形成，而不是最后补 写。必要时可在报告中附部分关键源代码，但不需要附全部源代码。

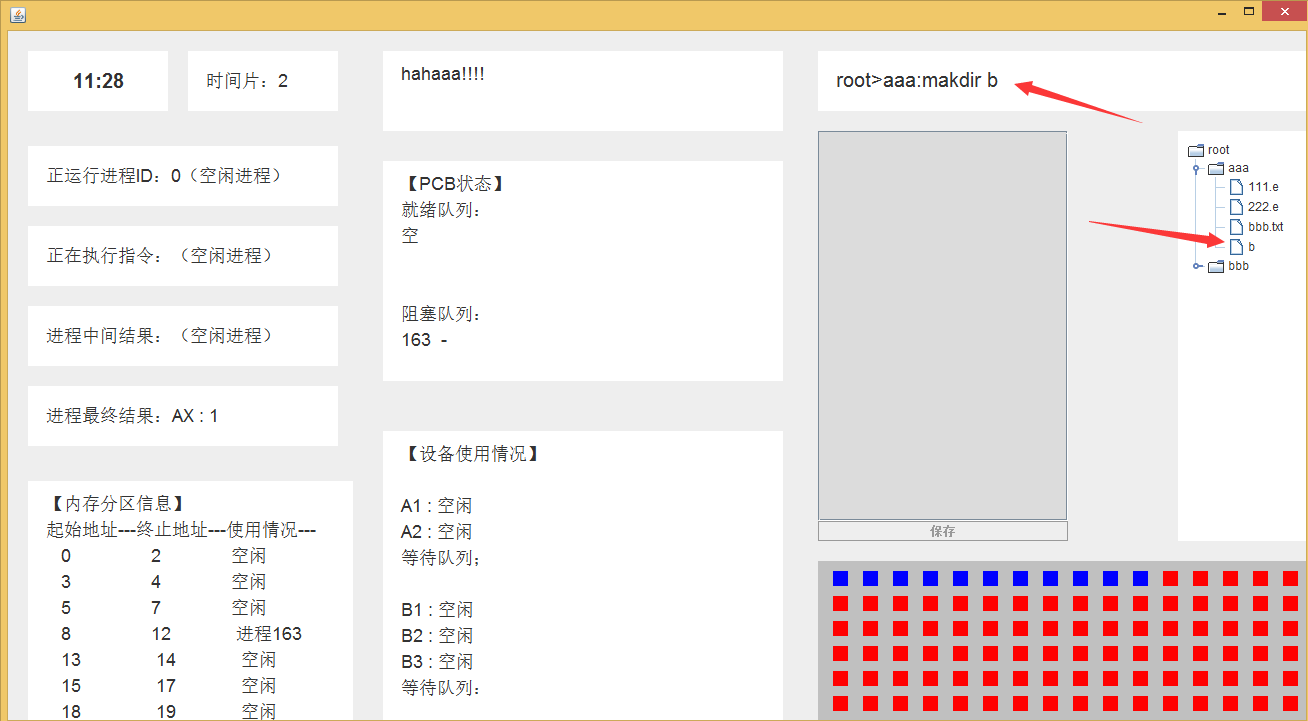
1.cd aaa进入aaa目录



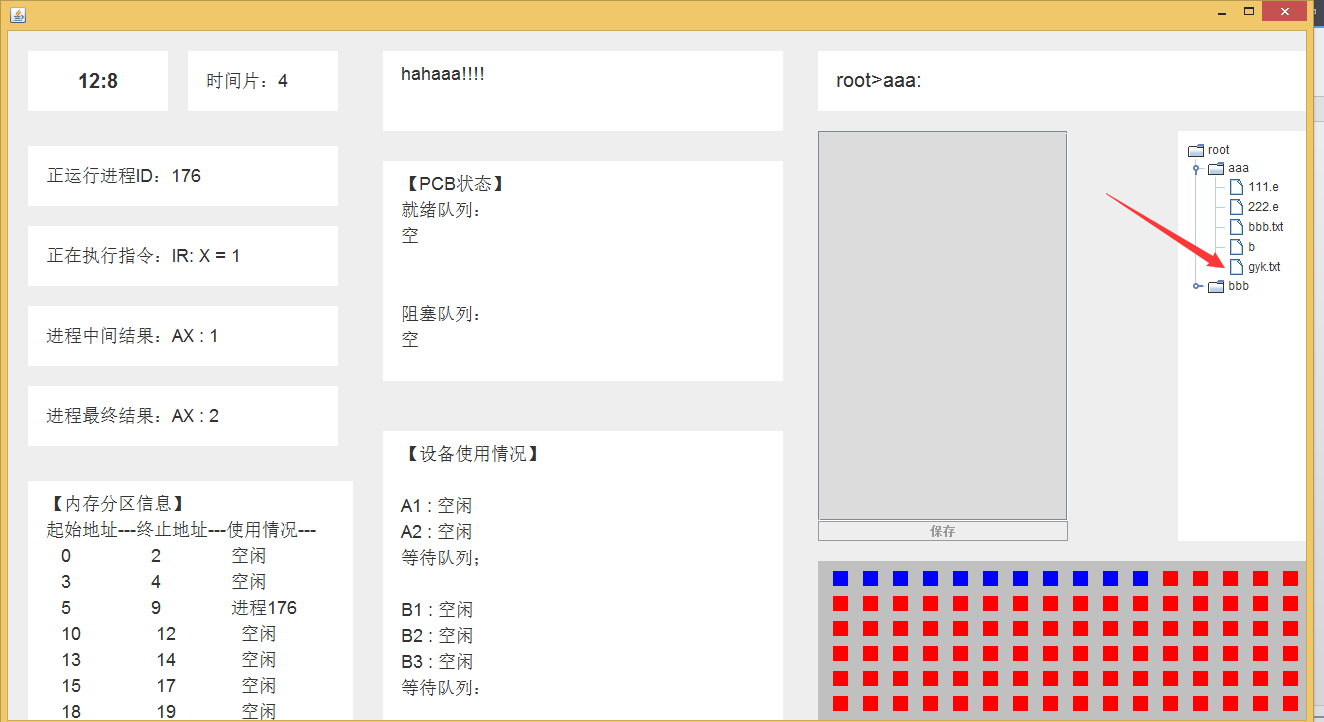
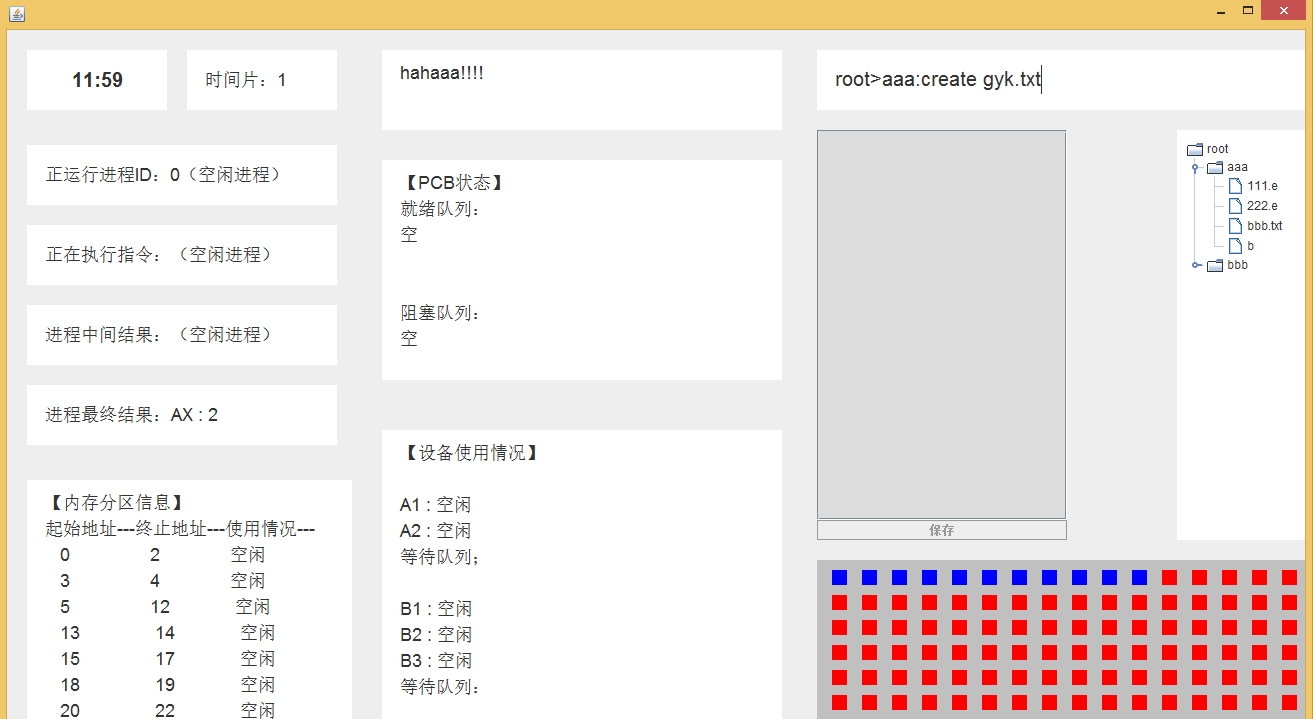
cd .. 退回上级目录



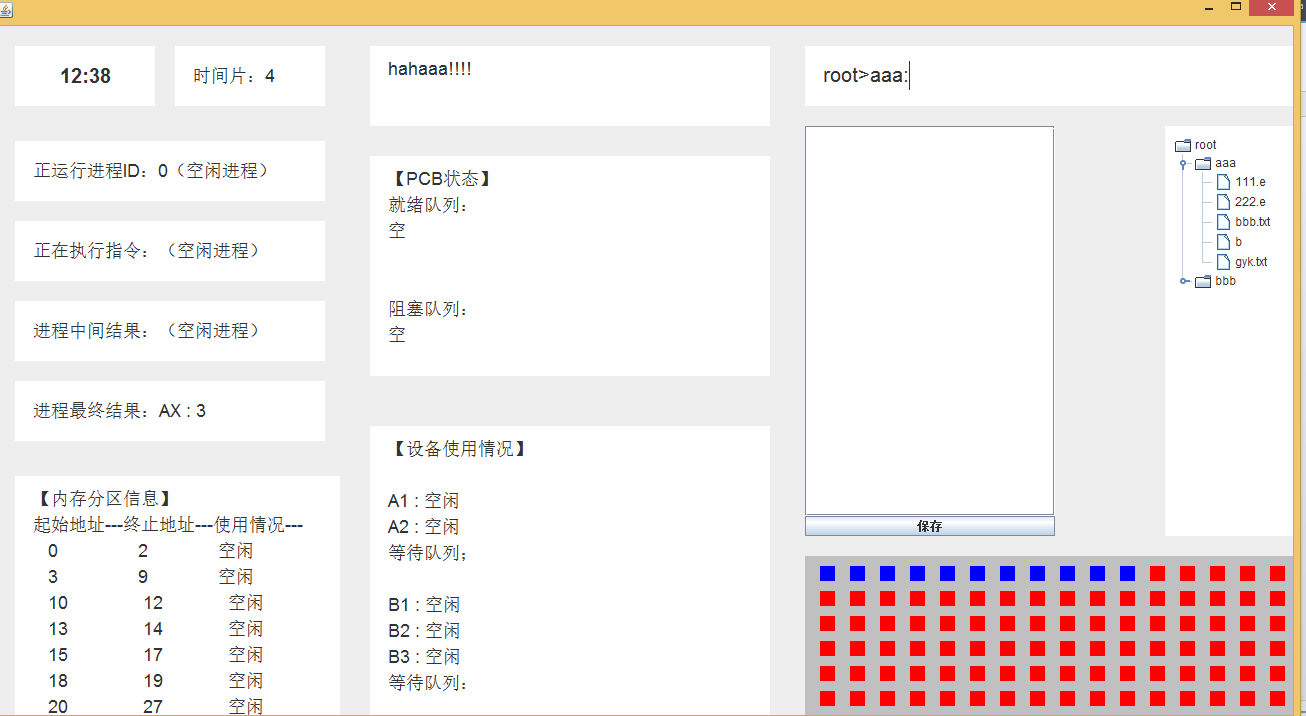
1. makdir b在aaa目录下建立b目录



1. create gyk.txt在aaa目录下建立gyk.txt文件

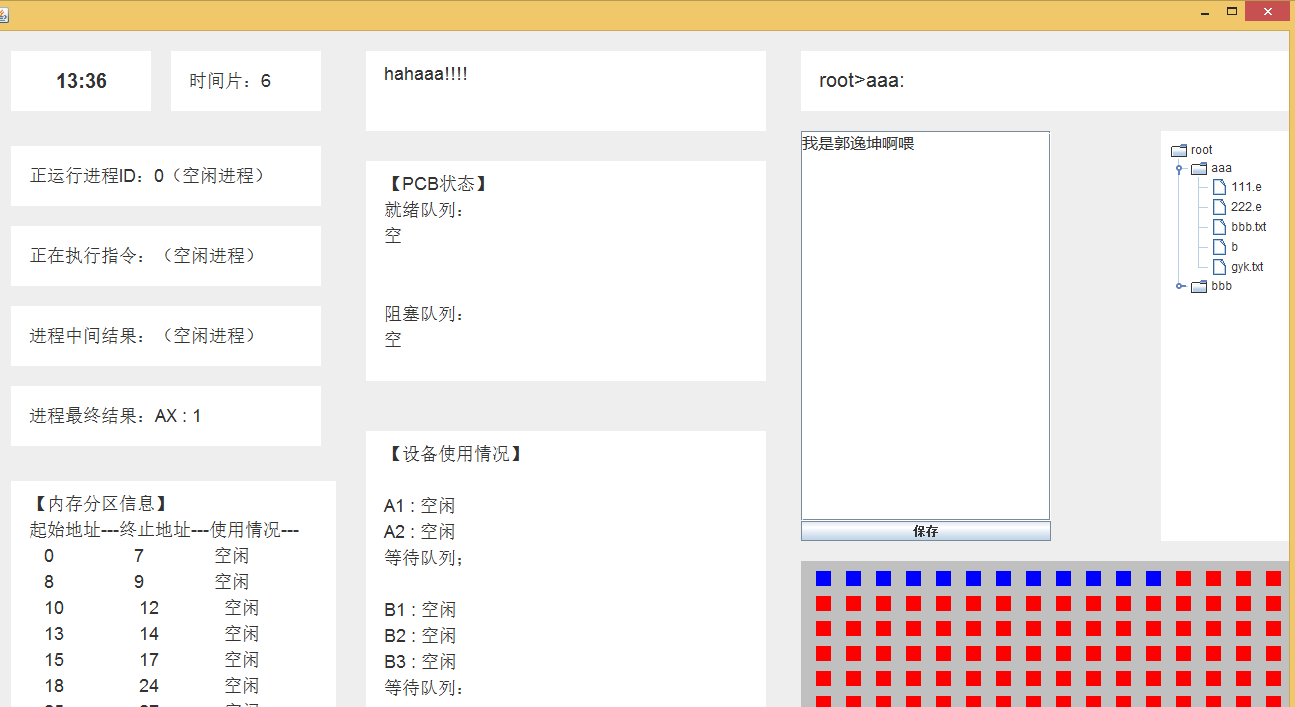


1. type gyk.txt 打开gyk.txt文件

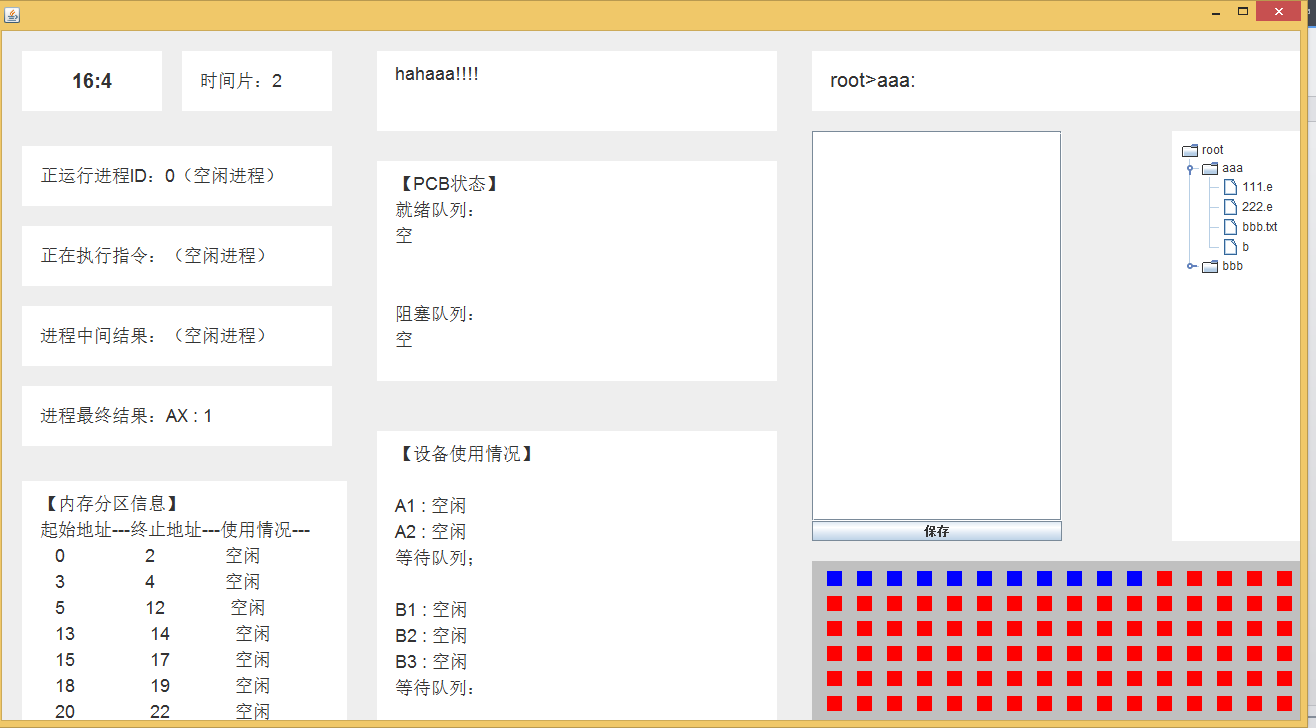


当前文件内容为空，所以文本域内是空白的

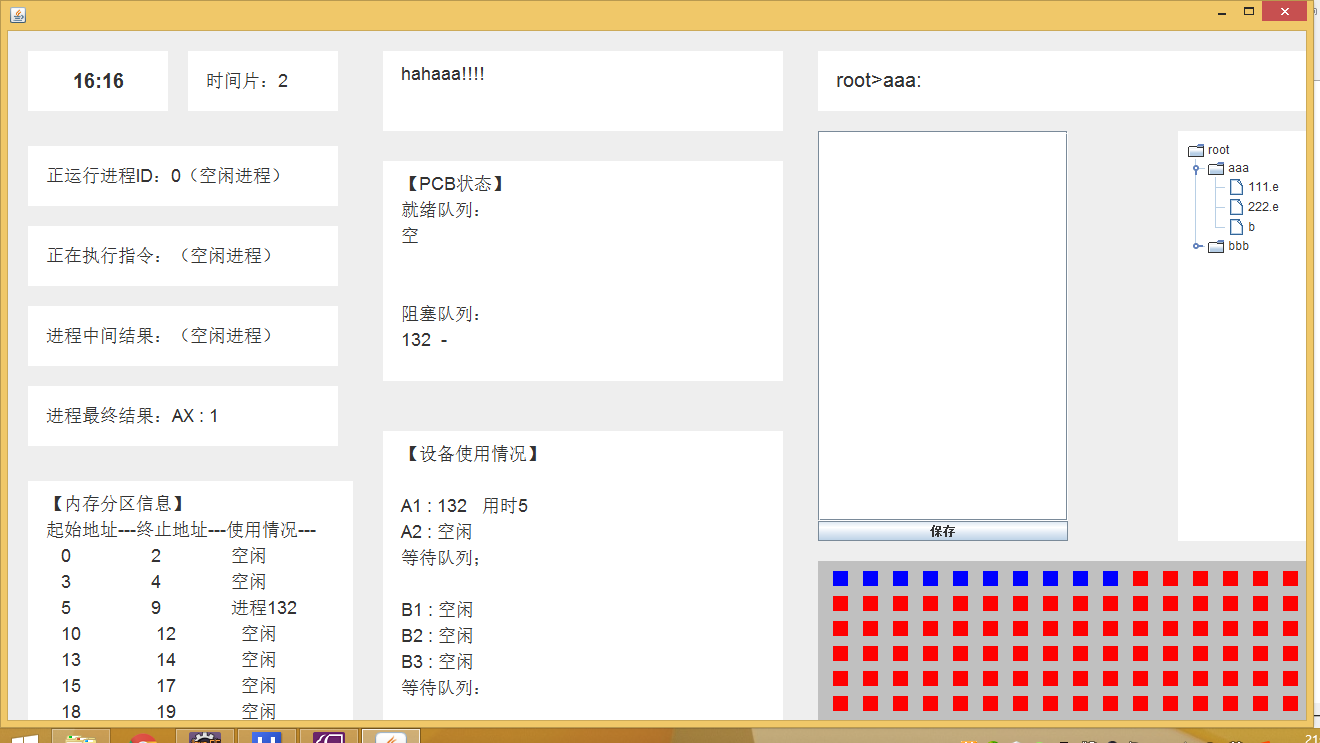
，输入文字，点击保存，并再次打开，文字被保存



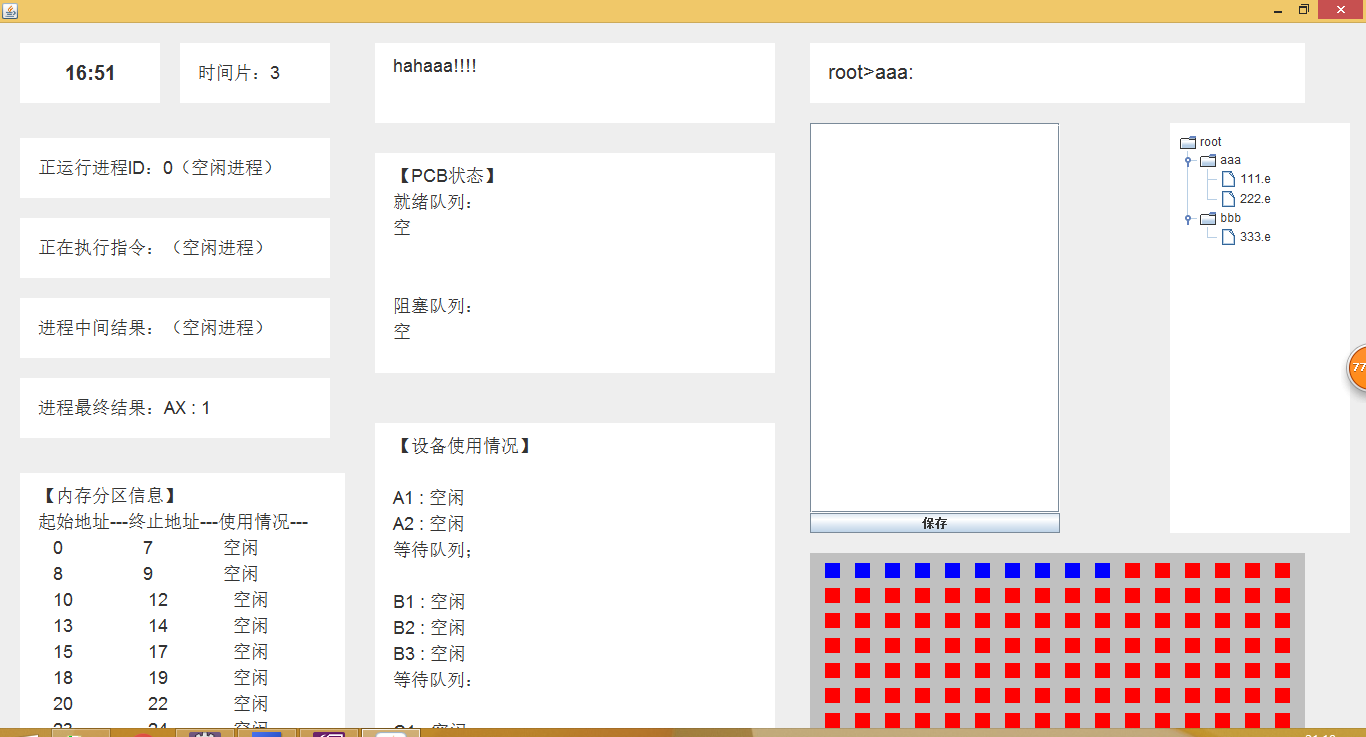
1. delete gyk.txt删除gyk.txt文件



Delete bbb.txt



1. rmdir b删除b目录



1. newexe ggg.exe新建exe文件

