

操作系统课程设计实验报告



[计科7班 第4小组]

[裴子祥 学号2015211921]

[陈博韬 学号2015211328]

[汤浩然 学号2015211323]

[张瑞康 学号2015211315]

[吴昌顺 学号2015211318]

[王文斌 学号2017526018]

[周成浩 学号2015211320]

[指导老师：孟祥武]

2018-5-18

目录

[1. 整体框架和流程 2](#_Toc514575132)

[2. 三个模块的交互 5](#_Toc514575133)

[2.1 文件系统与进程模块 5](#_Toc514575134)

[2.2 进程模块与内存模块 6](#_Toc514575135)

[2.3 文件系统（磁盘）与内存模块 6](#_Toc514575136)

[3. 演示图以及相应说明 7](#_Toc514575137)

[3.1. 文件系统 7](#_Toc514575138)

[3.2. 内存模块 12](#_Toc514575139)

[3.3. 进程模块 16](#_Toc514575140)

[4. 重要函数 24](#_Toc514575141)

[4.1 文件系统 24](#_Toc514575142)

[4.2 内存模块 26](#_Toc514575143)

[4.3 进程模块 26](#_Toc514575144)

[5. 成员心得体会 28](#_Toc514575145)

# 整体框架和流程

1. **框架与模块**

进自然语言描述：相对独立的三部分：文件系统，进程模块和内存管理。

文件系统可以做到独立（进程和内存独立没有意义），内存通过与进程的联系间接和文件相关，进程和文件直接相关。独立的文件可以时文本文件等，有相关的文件可以是可执行文件等。

核心内容是进程，内存造成了进程的实现，而文件作为外壳提供数据。

**文件系统：**

图形化界面，类windows文件组织形式，树型目录

文件夹与文件两种节点，文件属性有只读和可读可写两种

可查看当前选中路径，文件信息，已经打开文件信息。

文件夹、文件通用操作：

鼠标右键功能菜单

新建文件、文件夹

删除文件、文件夹（可非空）

重命名文件、文件夹

拷贝、粘贴、剪切文件、文件夹

文件操作：

打开文件会有两种形式，第一种纯文本，即弹出文本对话框；第二种，进程文本，其内容表征一个进程的具体操作及其所需时间（类甘特图内容）。

更改文件读写属性，如只读->可读可写

打开文件，对于只读文件打开后不可编辑

可以直接在textdialog中编辑文件，保存，退出

发生错误冲突处理：

同路径下文件夹、文件重名。

重命名、新建时重名。

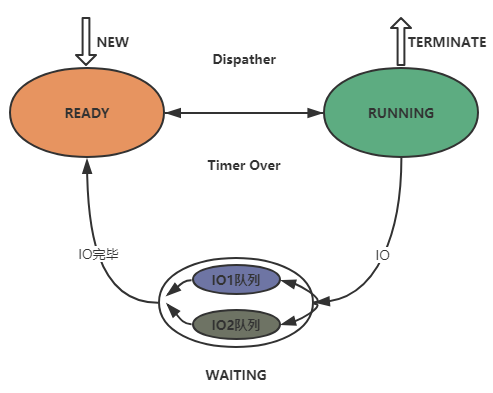
粘贴时，重名冲突（这里直接不可粘贴）。粘贴时不能为子路径（否则，循环粘贴）

删除，修改属性时，不能选择已经打开的文件。

**进程管理：**

PCB结构设计。

进程有占用CPU，占用IO1,占用IO2的操作，进程状态转换图：



三个状态（RUNNING、READY、WAITING(IO1与IO2)），其中WAITING态中有四个子状态，OCCUPY\_IO1、WAIT\_IO1；OCCUPY\_IO2、WAIT\_IO2。

创建进程（默认随机创建、填写部分信息创建、根据文件内容创建）

手动杀死进程

三个状态

实现的调度算法如下：

FCFS先来先服务

RR时间片轮转（时间片为5s）

SJF最短作业优先

SJF抢占式

PRIO优先级

PRIO抢占式

HRRN高响应比优先

**内存管理：**

管理方法：段式、页式、段页式

页的置换算法：

SC（二次机会算法）

LFU（最不经常使用置换算法）

段式内存管理，内存初始化，进程分配内存，内存释放，碎片整理。

段页式在内存中的实现

包括页面的置换算法,段表\页表的数据结构

内存的展示有内存使用效率,段表\页表的数据展示(在文件中)

内存\虚存帧情况,每次使用页面置换后都会更新文件以观察页帧情况

*页式：*

数据结构图示：

**进程信息结构数组：**

|  |  |  |
| --- | --- | --- |
| **进程id** | **内存大小** | **页号链表头结点** |

**页表结构数组：**

|  |  |  |
| --- | --- | --- |
| **页号** | **内存块号** | **有效位** |

**内存块数组：**

|  |  |
| --- | --- |
| **块号** | **有效位** |

1. **流程**

自然语言描述：

一个模块联合流程可以解释为：运行可执行文件->创建相应进程->为进程分配内存（或调入虚存）->进程运行（视情况进行页面置换）->进程结束请求->释放内存->进程结束。

文件系统：在界面中，选择文件或文件夹，鼠标右键或点击功能按钮去完成想要的10余项操作。

进程模块：三种进程创建方式（默认随机，部分信息创建，文件完整信息创建）

内存管理：在初始化文件中修改配置选择当前的管理方式与内存大小和换页算法，其它效果将在进程创建后发生，例如系统利用率，内存不足提示，换页操作等。

页式流程描述：

（1）提前建立数据结构：记录进程id、内存大小和内存分配页号的结构数组，页表结构数组（页号和内存块地址），内存块数组（内存块地址和该块被占用情况记录），并初始化数组。

（2）进程创建时，传入进程id和进程所需内存大小，根据内存块被占用情况给该进程分配所需内存。如果内存不够，则分配一部分内存给该进程，之后会进行页面置换来支持各进程的运行。

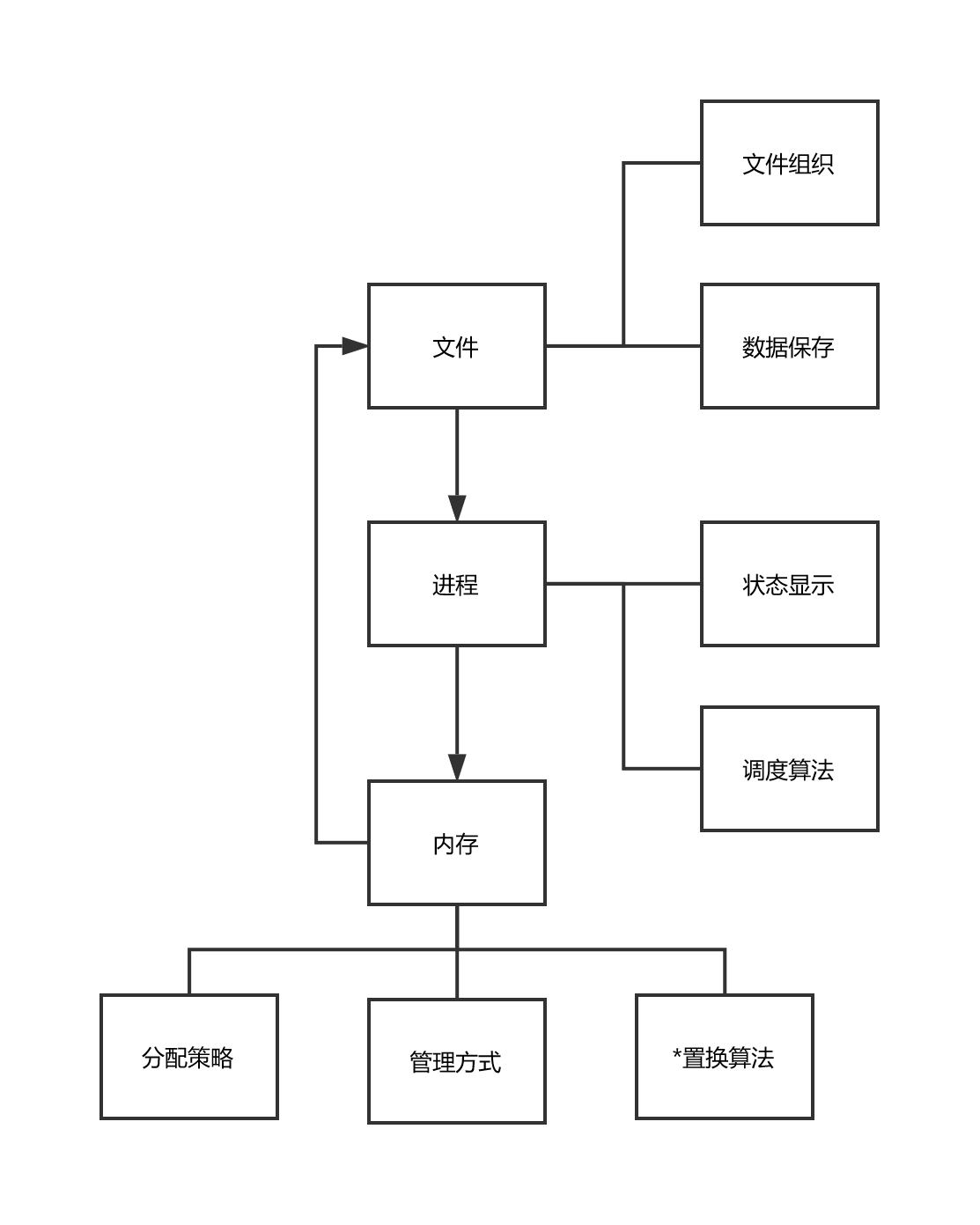
（3）在提前建立好的链表结构中，添加进程id，把分配好的内存块的块号（即内存块地址）添加到页表中，把该项的页号记录到进程id数组中（即记录页地址）

（4）进程启动时，调用页式内存管理，进程数组中写入该进程信息，检测内存块数组中空闲块数量，把可分配给该进程的空闲块有效位设置为0（表示被占用），在页表数组中分配一项（即一页），在其中写上内存块号，设置其有效为为0（表示被占用），然后把页表中该页页号写入进程数组中该进程的页号链表中。

（5）如果步骤（4）中分配内存时，空闲块数目不够，则调用LFU或者SC算法进行页面置换操作，生成若干个空闲块，并分配给该进程。

**Figure 1.2.1 页式流程图**

1. **框图**



**Figure 1.3.1‑2 模块主要结构图**

# **三个模块的交互**

1. 文件系统与进程模块

i. 打开文件来创建进程：

1. 需求：为了更好的模拟一个真正的操作系统。例如在windows环境下，模拟其可执行文件，一旦执行，则创建相应进程。

2. 实现手段：文件中保存了该进程的所有信息（进程号，优先级，执行时间，进程大小，占用CPU时间，占用IO接口，占用各IO接口时间等）。一旦运行则在进程中添加拥有这些属性的进程。

3. 保护手段：为了“可执行文件”不被随意修改，默认设置为只读状态。

1. 进程模块与内存模块

i. 为进程分配内存：

1. 需求：模拟进程执行需要使用一定大小的内存。并且在内存大小不够的情况下不予以使用内存的许可或者放入虚存。若虚存大小不够，只能不予分配内存。

2. 实现手段：使用操作系统模拟一块内存或虚存（必要的情况下），在创建进程时会申请使用进程大小的内存，操作系统再按照内存分配策略来处理该请求。

3. 虚存和灵活实现：采用段页式分配时会使用一块一定大小的磁盘用作虚存，并在进程运行时（必要时）进行页面置换。而且内存大小，虚存大小，策略，页面置换算法，页大小都可以进行修改。

ii. 释放内存：

1. 需求：模拟进程结束时（强制或非强制），准确的释放内存。

2. 实现手段：段页式举例，指定进程所许可的内存（或虚存）都含有唯一对应该进程的信息-进程id，释放指定id下所有内存（或虚存）。

1. 文件系统（磁盘）与内存模块

i. 内存和虚存硬件上情况：

1. 需求：为了演示，需要观察系统运行时内存和虚存的物理情况。

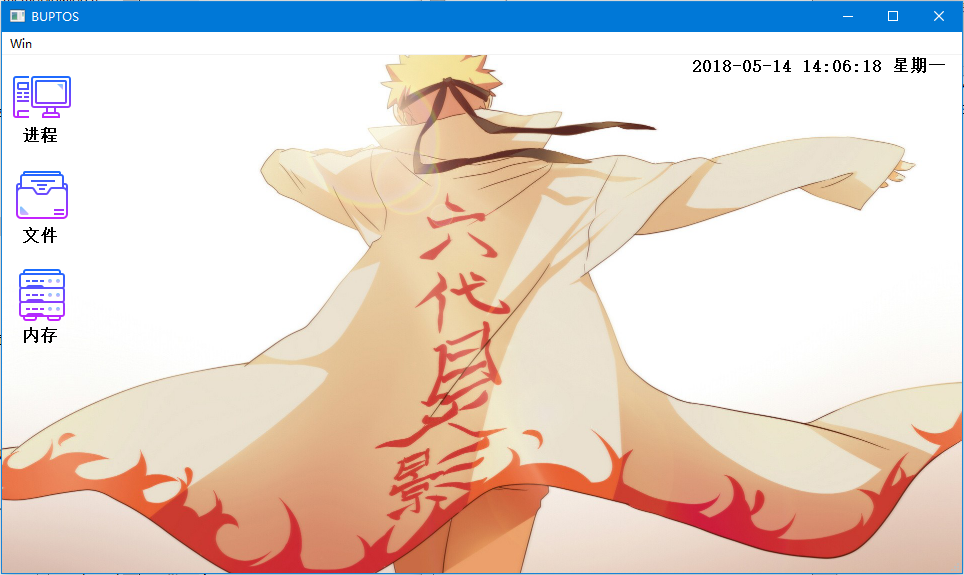
2. 实现手段：模拟一块内存（或虚存）大小的数组，记录每页（帧）情况，并输出到文件。

ii. \*页面置换：

1. 需求：为了演示和验证算法，需要观察页面置换结果。

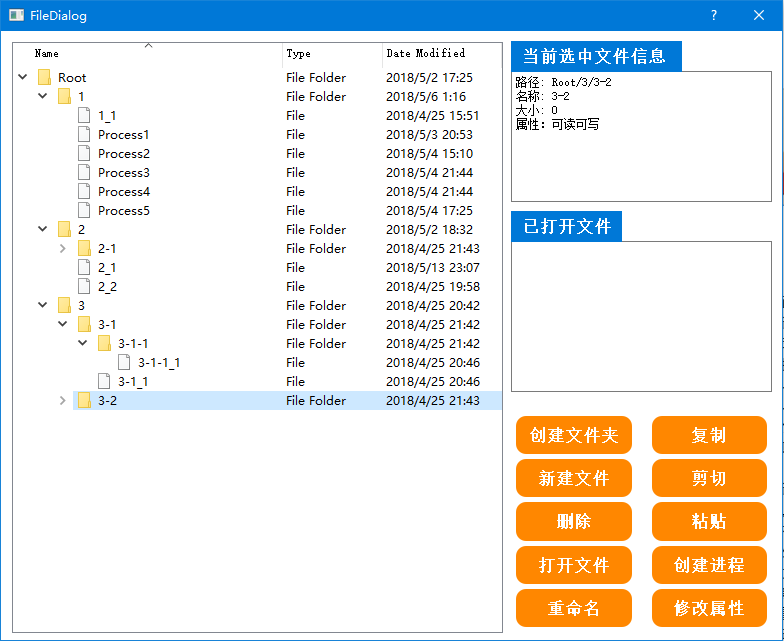
2. 实现手段：改变“硬件”上页（帧）情况，并输出到文件。

# 演示图以及相应说明

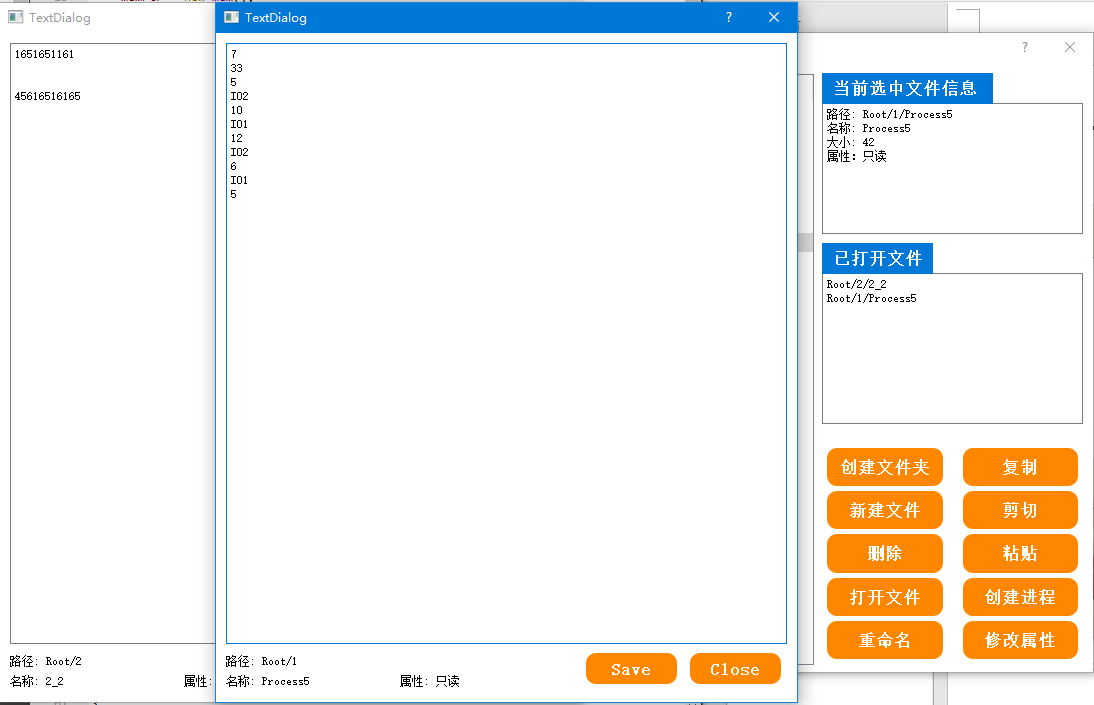


**Figure 3‑1** 主界面：进程、文件、内存、时间

1. 文件系统

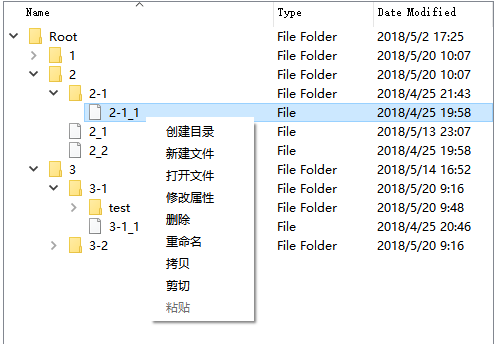


**Figure 3.1.3‑2 文件系统主界面**



**Figure 3.1.3‑3 打开文件，即打开文本编辑框**

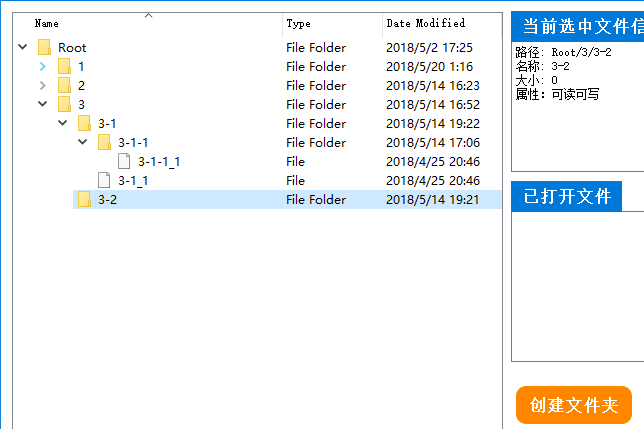
*鼠标右键菜单栏*



当在文件系统中右键时，弹出右键菜单，和外面的按钮功能相同，点击后会在当前选中路径下执行响应操作。

*创建文件夹*

选中文件夹3-2，点击“创建文件夹”，输入test

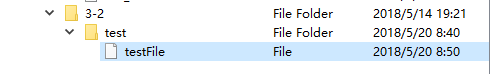


可以看到test文件夹被创建，如下：



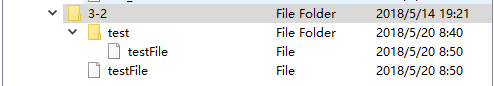
*新建文件*

选中test,新建文件夹testFile



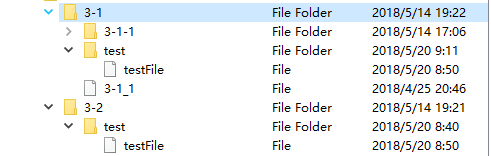
*复制文件*

选中testFile,点击复制，再选中文件夹3-2，点击粘贴



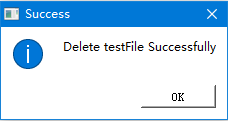
复制文件夹

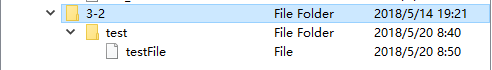
选中文件夹test，点击复制，再选中文件夹3-1，点击粘贴，成功粘贴



*删除文件*

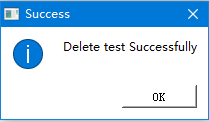
选中文件夹3-2下下的testFile，点击删除。删除成功





*删除文件夹*

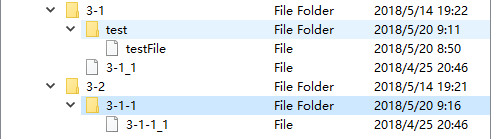
选中文件夹3-2下的test文件夹，点击删除。删除成功。





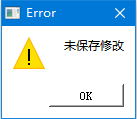
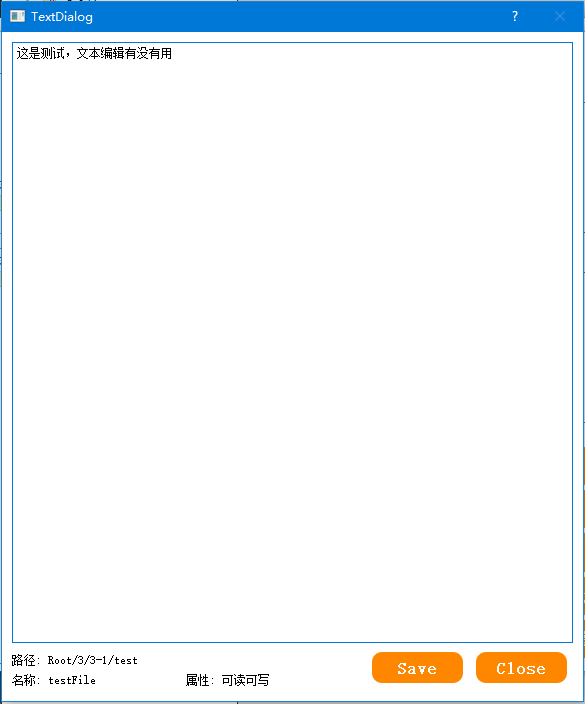
*剪切文件夹*

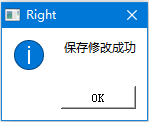
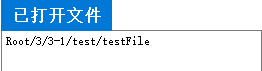
选中文件夹3-1-1，点击剪切，选中文件夹3-2，点击粘贴。可以看到文件夹3-1-1被剪切至文件夹3-2中。



*打开文件*

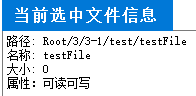
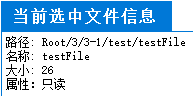
选中文件testFile，它的属性为可读可写。双击或点击“打开文件”。已打开文件中会显示当前打开的文件，弹出文本编辑对话框，输入内容，直接点击“close”，提示保存文件修改。点击“save”,再点击“close”。





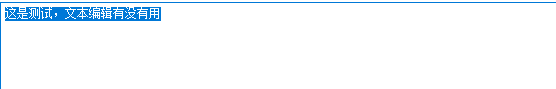
*文件属性修改（只读与可读可写性的切换）*

选择testFile,在右侧信息栏可以看到，属性为可读可写。点击“修改属性”。

🡪

打开文件后，对话框中，也显示只读，并且文本框不可编辑





当然，对于只读文件，点击修改属性后，变为可读可写。

*重命名*

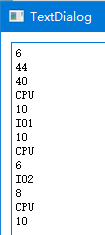
选中testFile文件，点击重命名。输入NewName。重命名成功



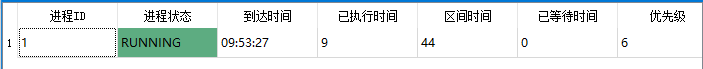
*通过文件创建进程*

选中文件夹1中Process1,点击“创建进程”，

其进程参数内容如下：

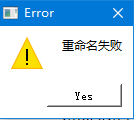


在进程对话框中显示，新创建的进程1

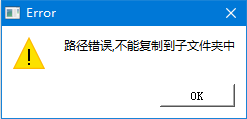


*错误处理*

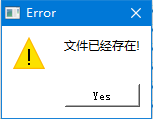
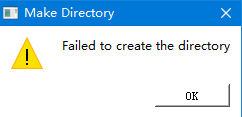
重命名时，文件名重名，重命名文件2\_2为2\_1



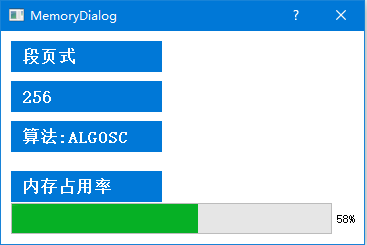
复制、剪切、粘贴时，不能复制到子文件中，选中文件夹3-2，点击“复制”，再点击文件夹3-2中子文件夹3-1-1，点击“粘贴”。报错。



新建文件或文件夹时，与当前路径下文件重名，在文件夹2-1下新建文件2\_1,在文件夹2下新建文件夹2-1，都会报错

1. 内存模块



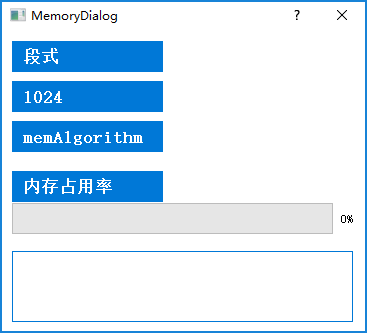
**Figure 3.2.3‑4 内存显示主界面**

*内存实质是其他部分在不可见范围（一般情况）的实现，演示也以观察方式为主。*

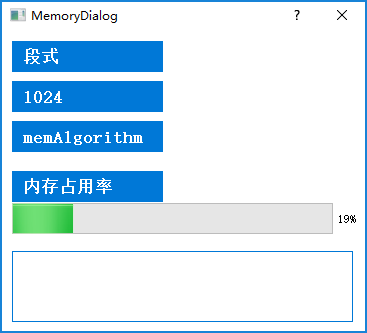
*段式：*

选择管理方式为段式：

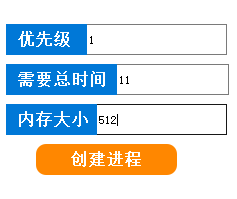
在进程创建等的条件下，段式仅观察内存利用率：

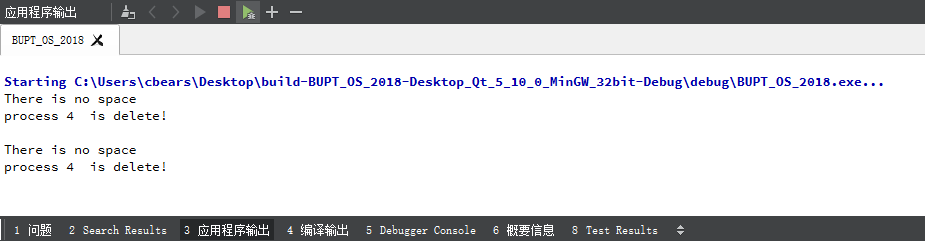


创建进程成功，其内存会相应变化并且表现为内存占用率变大：

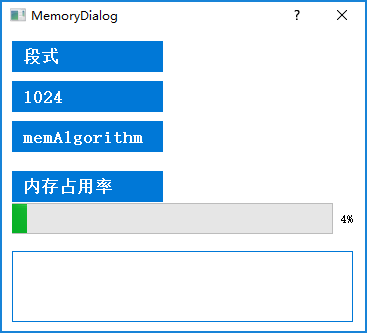


若创建大于剩余内存的进程，会报错，并且不会为该进程分配内存，进程创建失败：



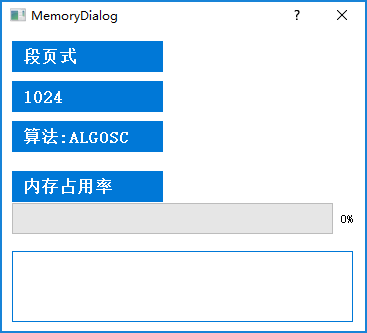
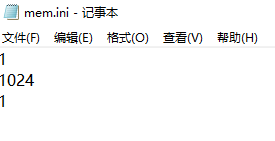


删除进程后，内存会相应变化，表现为内存占用率变小：



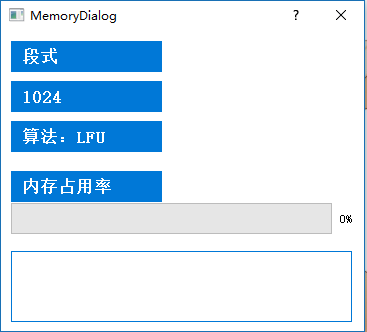
*段页式：*

选择内存管理方式为段页式并且指定算法（SC为例）：



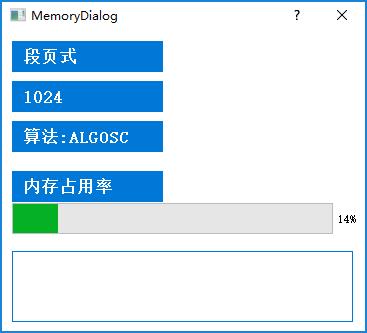
*页式：*

选择内存管理方式为页式并且指定算法（LFU为例）：

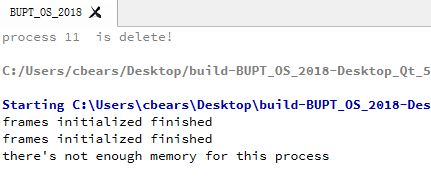


两种算法：

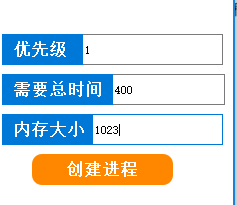
1. SC：
   1. 创建进程，表现为内存占用率变大：



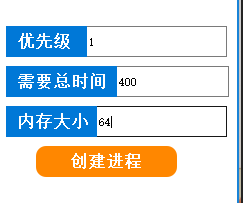
* 1. 创建进程大于剩余实存+剩余虚存，不予分配内存：



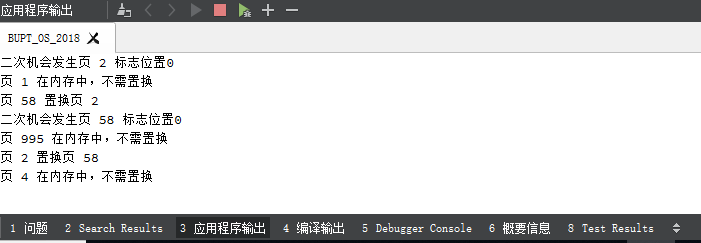
* 1. 页面置换：
     1. 创建一个进程，使得实存仅剩1MB：



* + 1. 创建一个进程大于1MB的程序，使得满足页面置换条件：



* + 1. 页面置换发生：



页面置换信息（输出）：

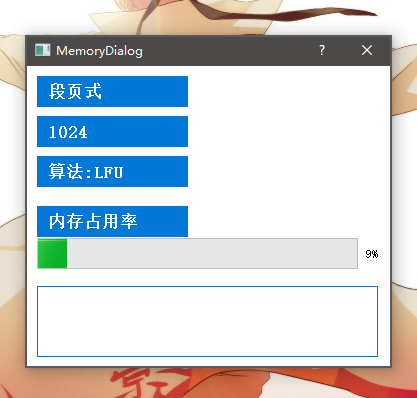
二次机会发生时，被选中置换页SC标志位为1=>SC标志位置0；

被选中置换页SC标志位0=>置换该页；

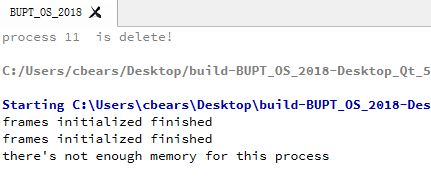
被选中页在内存中=>不需置换。

* + 1. 进程结束后，相应行为结束。

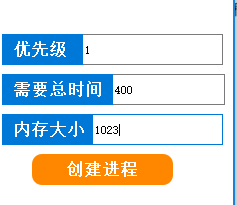
1. LFU：
   1. 创建进程，表现为内存占用率变大：



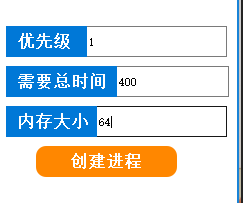
* 1. 创建进程大于剩余实存+剩余虚存，不予分配内存：



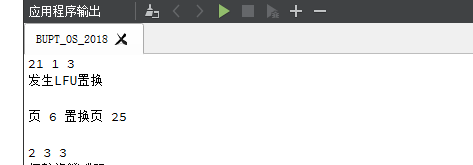
* 1. 页面置换：
     1. 创建一个进程，使得实存仅剩1MB：



* + 1. 创建一个进程大于1MB的程序，使得满足页面置换条件：



* + 1. 页面置换发生：



* + 1. 进程结束后，相应行为结束。

1. 进程模块



**Figure 3.3.3‑5 进程模块主界面**

*三种进程创建方式：*

（1）随机分配参数——只占用CPU

进程界面创建进程按钮上方有三个参数，不输入这三个参数直接创建进程即为随机分配参数方式。直接点击多次“创建进程”后：



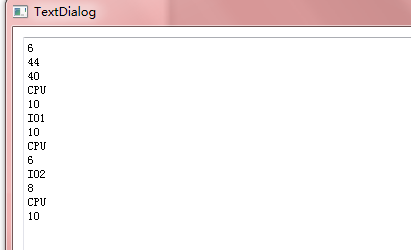
（2）手动输入参数——不带IO竞争——只占用CPU

手动输入三个参数，点击创建进程按钮。

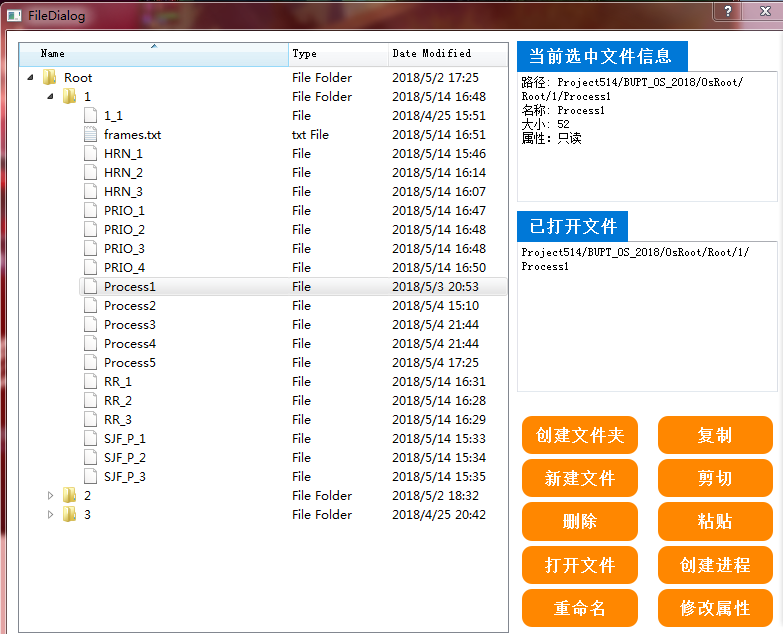
 

（3）文件输入参数

新建文件，格式如下：前三行分别为进程优先级、进程运行总时间、进程占用内存大小，后面各行为占用COU\IO1\IO2的时间。例如下图所示：



创建进程，上一步中新建的文件，点击文件系统界面的创建进程，即创建出基于文件内容运行的进程。如下图所示：





*进程删除方式：*

进程删除，鼠标选中要删除的进程，点击进程界面的删除进程，进程被删除。

点击前:



选中的是ID号为12的进程

点击后:



ID为12的进程已被删除

*FCFS算法*

运行创建进程的随机分配参数方法创建若干进程，采用默认的FCFS算法调度，可以看到总是先到达的进程开始运行。

创建若干进程：



可以看到界面的进程是根据就绪队列的顺序显示的，由此可见这是根据FCFS算法调度的，即先到达的先运行。

*RR时间片轮转调度*





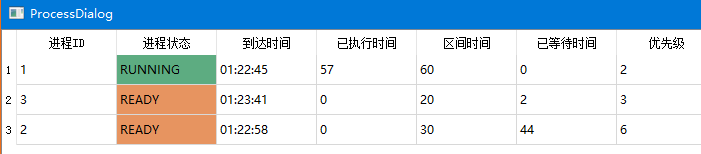


时间片大小为5，每次正在running的进程只能执行5秒就会回到等待队列，等待下一时间片的到来，或者进入WAIT态进行IO操作。

*PRIO优先级调度，非抢占式*



开始2个进程，优先级分别为2和6。新建进程3，优先级设为3。

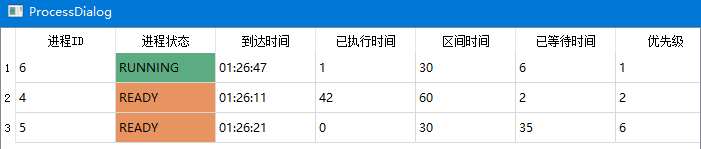


进程3排在了进程2的前面，因为优先级更靠前。

*PRIO优先级调度，抢占式*



进程4进程5在正常运行，新建优先级为1的进程6



进程6优先级更高，抢占了正在运行的进程4的CPU。

*SJF算法,非抢占式*



此时，就绪队列中进程4在就绪队列队首.新建进程5，其区间时间为20小于进程4.



进程5所需时间小于进程4，被调度到队首。

*SJF算法,抢占式*

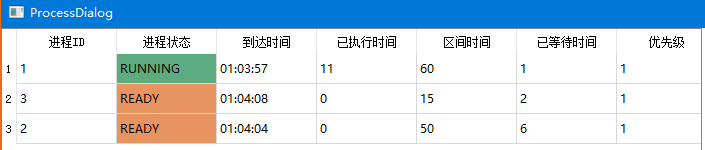


进程8正在运行,区间时间35。创建进程9，所需时间25。

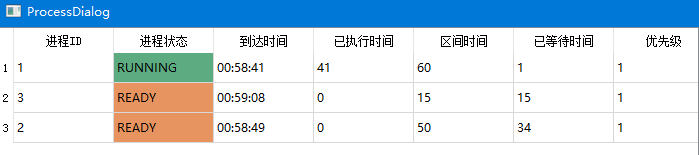


进程9抢占进程8，使用cpu，进程8进入就绪队列。

*HRRN高响应比优先算法*



只有2个进程时，进程1所需60，进程2所需时间50，进程3所需时间15。



在经过一段时间后，进程3的响应比高于进程2，所以排在了进程3前面。完成动态优先级调度算法。

# **重要函数**

1. 文件系统

**FileDialog + TextDialog**

**FileDialog:**

private:

QFileSystemModel \*fileModel; //文件系统模式

QString lastPath; //用于剪切复制

int copyOrCut; //0无,1为复制文件,2为剪切文件,3为复制文件夹

QVector<QString> openedFileList; //已打开文件列表

//在当前选中路径下，创建文件夹

void FileDialog::mkdir()

//通过路径删除非空文件夹，文件夹内可有子文件夹与子文件

void removeDir(const QString &folderFullPath)

//删除选中文件夹（可非空）或文件，调用removeDir

void FileDialog::deleteItem()

//创建文件，若重名则创建失败，否则成功创建一个空文件

void FileDialog::createFile()

//用文本框打开文件，相当于打开notepad显示文件内容

void FileDialog::openFile()

//关闭文件，根据文件路径，从已打开文件队列中移除

void FileDialog::closeFile(QString file\_path)

//重命名，若重命名与同路径文件重名则失败报错

void FileDialog::rename()

//复制、剪切、粘贴当前选中条目（文件或文件夹）

void FileDialog::copyFileOrDir()

void FileDialog::cutFile()

bool copyDir(const QString &fromDir, const QString &toDir)

void FileDialog::paste()

//读取文件内容，并以此创建进程

void FileDialog::on\_createProcessButton\_clicked()

//切换文件属性，只读->可读可写；可读可写->只读

void FileDialog::on\_rwChangeButton\_clicked()

//右击文件或文件夹弹出功能菜单

void FileDialog::on\_fileTreeView\_customContextMenuRequested()

**TextDialog:**

class TextDialog : public QDialog //文本对话框，notepad

{

Q\_OBJECT

public:

explicit TextDialog(FileDialog \*parent = 0, QString fullPath="", QString fileName="");

~*TextDialog*();

QString filePath; //文件全路径

QString fileName; //文件名

bool isEdited = false; //是否被修改

private slots:

void on\_saveTextButton\_clicked(); //保存文件按钮

void on\_closeButton\_clicked(); //关闭文件

void isEditedChange(); //编辑状态改变

signals:

void closeSignal(QString file\_path); //给FileDialog信号，此文件关闭

private:

void initial(); //打开后，初始化界面，并输出文件内容，一个文件对应一个对话框

};

1. 内存模块

二次机会算法函数：void algorithm\_sc();

选择管理方式为段页式，算法为SC时，进程不为空就会执行。

LFU算法函数：int LFU();

选择管理方式为段页式，算法为LFU时，进程不为空就会执行。

内存分配函数：bool arrange\_p(int id,int size);

传入参数为进程号和进程大小，然后由操作系统为其分配内存。

内存释放函数：int page\_delete (int ID);

传入参数为进程号，发起结束进程请求后，释放该进程所有内存。

分段函数：int pre\_segs4seg(int id, int size); （int pre\_segs4page(int id, int size);）

传入参数为进程号和进程大小，为进程分段。

1. 进程模块

*创建进程函数*

int createProcess(int priority, int needTime, int memS=20)

此函数用于创建不带IO竞争的进程，三个参数分别是进程优先级、进程运行总时间、进程占用内存大小。

int createProcess(QString filePath, QString fileName)

此函数用于通过文件创建带有IO竞争的进程，两个参数分别是文件路径，文件名。

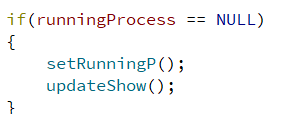
*删除进程函数*

void deleteProcess(int id)

此函数用于通过文件ID删除相应函数，从队列中将代表该进程的PCB块删除。

*先来先服务算法*

void FCFS()



判断runningProcess是否为空，若为空则表示当前无正在运行的进程，使用setRunningP函数使就绪队列第一个进程进入运行状态。



若不为空则说明有进程正在运行，获取当前正在运行进程正在使用设备的标号PC，若标号等于其设备队列的大小表示该进程已使用完其需要的所有设备，结束该进程。若当前正在使用的设备为cpu则判断该进程占用cpu的时间是否达到所需时间，若小于则时间增加，否则设备标号加一，使用下一设备。若当前使用设备为IO1或IO2则利用runningToWaiting函数将进程从运行状态转入等待队列，再调度新的进程。

接着处理等待队列，若队列大小大于0则判断进程是否完成该设备的使用，是则返回就绪队列，否则继续使用时间增加。

*用于sort（）的比较函数*

static bool compByReachTime(const Pcb \*pcb1, const Pcb\* pcb2)

此函数根据进程到达时间比较先后，用于sort()算法中的比较函数。

*进程状态转换函数*

void setRunningP();

void runningToReady();

void runningToWaiting(int IOkind);

void waitingToReady(int IOkind);

void terminateP();

这些函数都是用于进程状态的的切换。

*选择调度函数*

void dispatchSelect();

此函数用于选择调度算法，根据选择调用相应算法函数进行调度。

*选择是否抢占函数*

void isPreemptiveSelect();

此函数用于选择调度算法是否是抢占式的。

*更新进程界面函数*

void updateShow();

此函数用于更新进程界面的显示内容，实时显示各进程信息。

*最短作业优先算法*

void SJF();

函数主体与FCFS函数相同既读取就绪队列的第一个进程，最短优先的排序在进程插入就绪队列的时候实现。

*高响应比算法*

void HRRN();



利用sort函数对就绪队列中进程根据相应比大小进行排序。

*时间片轮转调度算法*

const int RR\_CYCLE = 5; //时间片长度

void ProcessDialog::RR()

*优先级调度算法*

void ProcessDialog::PRIO()

函数主体与FCFS函数相同既读取就绪队列的第一个进程，优先级的排序在进程插入就绪队列的时候实现。

static bool compByPriority(const Pcb \*pcb1, const Pcb\* pcb2); //按优先级排序

# **成员心得体会**

**裴子祥2015211921（组长）：**

这次操作系统课程设计实践时间周期较长，在每周的讨论会议上，组内交流现阶段成果与问题，功能实现不断改进，这是一个徐徐渐进的过程。这次实验中主要负责了文件系统及其目录结构的编写，按照windows文件组织方式与其功能进行模拟实现，对文件与文件夹处理，并实现notepad功能。磁盘使用本机windows磁盘，从逻辑到实际文件组织，分层设计，完整地实现10余项功能并能够自主报错。另外负责三个模块间的互相连接与结合使用，使得做出的效果更像是一个操作系统。以文件内容去定义一个自定义进程，也能很好地展示进程调度问题，也能输出内存换页情况等内容。对于操作系统的理解也不是纸上谈兵，通过此次的实践，对文件组织，进程管理，内存管理都有了更好的掌握，也加强了自身代码能力水平，更加注意代码规范与技巧并精简代码，更易懂易读。虽然我们已经实现大部分操作系统功能，但这是远远不足的，操作系统是一个大而完善的软件，需要考虑的问题还有磁盘管理，死锁问题等，这些在我们的系统中并没有很好的体现，还有很多改进空间。很荣幸担任小组组长，组员们也很给力，大家合力完成此次课程实践，边学习边动手实现。

**陈博韬2015211328：**

本人在这次作业中和其他组员同时负责了内存部分的内容，通过实践结合老师课上所述知识，从而对整个操作系统无论是各个模块的内容还是整体框架和模块间沟通都有了更深刻的认识。特别是算法实现的时候，其原理十分简单，但是代码实现又显得比较复杂。如果要在有一定体积代码的前提下保证不会出错，就要求本人实现内容的时候对其有着真正正确的理解。

不光是自己负责的部分，在与其他组员沟通时（负责同一块内容或者不同内容的），也对其他人负责的部分有了更加深刻的理解。

整体下来，锻炼了自己写代码的能力，增加了与其他人一起编写代码的经验，对操作系统有了更好的了解。

**汤浩然2015211323：**

本次操作系统课程设计我负责的是内存管理模块的工作。这次课程设计使我对上个学期所学的操作系统知识的理解更加深刻，同时由于自己参与相关的设计工作，对于内存方面的了解从一个概念上比较抽象的认识提高到了更加具体实际的使用上。这次实验中我具体设计了段式内存及段页式内存的释放，碎片的整理，及一些换页算法的实现上。对于内存的结构及组织方式都有了更加直观的认识。在设计实现换页算法的过程中更是深刻了解换页的具体步骤，与实现方法。许多原本对于某些理解有误的知识点页在这次实验后给纠正了过来，如分段实际上是将PROCESS进行分段而不是内存分段，虚存与内存什么时候发生调换等。在进行内存与进程对接工作时，也让我对内存与进程如何协调工作有了更直观的理解。

除此之外，这次课程设计也更让我懂得如何与其他人共同协作开发一个具体的程序。

**张瑞康2015211315：**

在本次操作系统课程设计结课之际，回顾自己的这次自制模拟操作系统之旅，感慨良多，收获满满。总结起来有以下几点：

1、 团队合作的重要性。一个具备较完善功能的工程，靠一个人的力量是万万实现不了的，唯有团队合作才有实现的可能。团队合作的重点在于沟通，在于统一。沟通是过程，方法，统一是目标，宗旨。在历时两个月的课程设计过程中，团队经历十数次的会议沟通，私下成员间数不清多少次的交流，从最开始分工到中期定好各自模块的接口，到最终整合在一块。我们实现了接口的统一，思想的统一，设计的统一。最后才有了这一版本的模拟操作系统。当然，一个好的团队一定离不开一个好的领导者，我们的组长裴子祥在这个过程中发挥了极其重要的作用，他是团队的关键，是合作的灵魂。

2、 纸上得来终觉浅。这次课程设计这是让我感触良多的一点，在上一个人学期学习操作系统的时候，觉得自己各个方面的知识都掌握的不错，最后考试也考得很满意，但是到了这一次要动手的时候，才发现实际的实现比自己在书上学来的东西要难的多。比如：进程队列要用哪种数据结构实现，进程的同步怎么实现等等这样的问题。通过这一次的课程设计，我解决了类似于这样的的问题，也加深了对于课程内容理解，坚定了自己以后的学习生涯中事事恭行的决心。

3、 发挥自己想象的空间。在这一次课程设计中，我们添加了一个比较有意思的功能，就是可以动态切换调度算法。当然这个功能在实际的操作系统中是不会出现的，但是我认为这样的功能不仅存在乐趣，而且具有实现的意义。在已有知识的基础上发挥自己的想象，创造一些有意思的小功能，这个过程让自己收获很多。

**吴昌顺2015211318：**

通过操作系统课程设计，加深了对页式内存分配方式的理解。尤其是查找资料的过程中，对内存管理机制有了更宏观地认识，对于设置快表、进行页面置换等这些方法在充分利用计算机硬件资源、提高存储效率上所发挥出的优势，体会更加深刻。

**王文斌2017526018：**

在本次课设中主要担任进程模块中进程调度算法（时间片轮转与优先及调度）的编写。在进程类（数据结构）由组长编写后，对于算法本生的理解显得更为重要。通过本次课设，对于进程调度的原理更加清晰。

**周成浩2015211320：**

通过本次实验，我先是对操作系统有了总体的了解，以前模糊不清的地方都明白了，比如操作系统的结构和运行机制。在本次实验中，我负责的是进程中两种调度算法的实现。在准备阶段，我对进程的含义以及工作原理等进行了深入的了解，在对自己所要做的工作有了清晰的认识之后，我对所要实现的算法进行预估，在与其他同学沟通并确定数据结构及相关函数之后着手编写代码。回顾实验过程，我养成了严谨的编程习惯，也熟悉了团队项目的过程，在操作系统项目和团队合作两方面都有了更深刻的理解和认识。