1. IO部件：

在我们的操作系统中I/O部件仅仅是对真实I/O 的模拟，它的目的是为了实现对进程资源竞争的模拟，它的本质是一个简单的sleep函数，做到对进程占用I/O资源的模拟

具体的调度模拟如下图1.1：

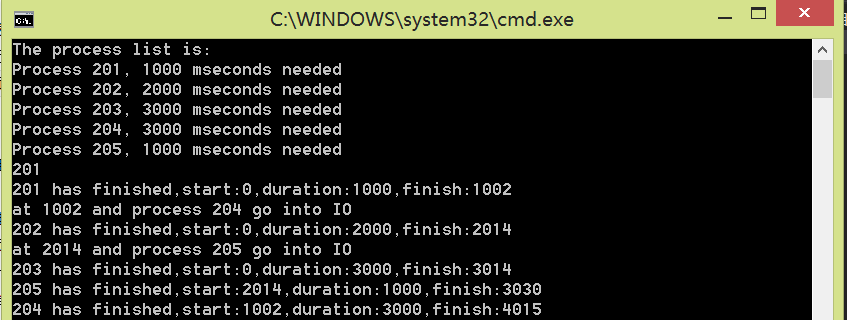


图1.1

后来，I/O模块为了更高效，采用了线程的模式，使用Wait函数以信号来确定I/O组件的使用，同时值得注意的是，因为I/O模块相对其他进程处理时间较长如图1.2：

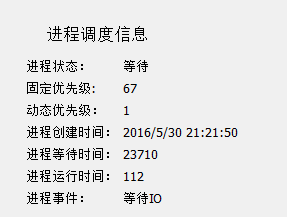


图1.2

所以在查看任务管理器的时候可能会出现显示一直waiting的情况

1. 打开文件列表：

void initOpenFile(OpenFile \* \_\_of);

对打开文件块进行初始化，设置属性filepath（char\*）为NULL，计数器counter为0；

void destroyOpenFile(OpenFile \* \_\_of);

销毁一个openfile块，直接free它的filepath

void addOpenFile(const char \* \_\_path);

添加一个打开文件，对于一个文件路径，首先在OpenfileList中查询该文件是否已被其他进程打开，若否，则新建一个openfile块，若是，则把对应的openfile块的计数器counter加一。

bool closeOpenFile(const char \* \_\_path);

删除一个打开文件，对于一个文件路径，首先查询他是否被打开，若否则报错 ，若是则把这个节点的计数器减一，若计数器减到零，则直接销毁这个节点。

bool checkOpenFile(const char \* \_\_path);

检查文件是否已经被打开，顺着第一个openfile节点找，直到找到相同的路径返回false，否则返回true

1. 进程指派：

PCB \* dispatchProcess();

这个函数的功能是选择一个动态优先级最高的进程执行，首先遍历整个链表，找到第一个状态为READY的PCB块，将他视为中间结果，然后与之后的各个PCB比较，找到优先级更高的则替换，直到遍历结束，返回最终结果。

void dispatch();

整体管理进程调度，在有进程需要调度时，调用相应的时间片函数，计算时间片长度，更新PC，把锦程之星，释放执行进程的信号量并进行计时；在没有进程需要调度的情况下，等待10秒后释放分配信号的锁。

1. 页表管理：

void pageUnvailable(ProcessID \_\_pid, int \_\_pgn);

通过pid找到对应PCB把他对应页表中的页号设置为unavailable（false）。

1. API：

void newPCB(PCB \* \_\_pcb, ProcessID \_\_pid, const char \* \_\_name, const char \* \_\_path, ProcessPriority \_\_pri, int \_\_pgnum);

初始化一个PCB，但是这个PCB并不会被挂到进程队列当中，状态为new，先记录PID，然后记录外部标识符，然后记录固定优先级，新建页表，再记录代码段长度，初始化数据段，完成一个PCB的新建

ProcessID newPID();

利用随机数函数产生初始PID，再除以最大PID值取余数再加一，在遍历这个歌进程队列查重，若有重复，则重新生成，直到无重复为止。

bool newProcess(const char \* \_\_path);

首先读取目标路径，检查文件类型，再提取文件信息，检查是否是可执行文件，若是数据区为空，则无法执行，再读取配置语句，计算优先级，再获取PID，分配虚存，将执行代码调入虚存，新建PCB，添加到打开文件到列表中。接着就可以把进程挂入队列中，在转移到就绪队列。

bool deleteProcess(ProcessID \_\_pid);

1. 动态优先级：

ProcessPriority getPri(int \_\_num);

进程初始Pri的获取是它的配置文件中的对应数值加一再乘3.375

ProcessPriority degeneration(ProcessPriority \_\_pri);

进程每执行一次，它的动态优先级变为原优先级的开根号再减一，同时把总的退化次数totalDegenerationCount加一

void priorityAdjust(PCB \* \_\_pcb);

这个函数的作用是动态优先级的调整，使得新调入等待队列的进程在退化次数上和队列的平均值，计算全局变量totalDegenerationCount的次数的除以总进程数，对新的进程退化相应次数

void resetQueue();

这个函数的作用是重置动态优先级和重置总退化次数，把动态优先级都设置为初始记录，把总退化次数归零。

1. 时间片分配：

int getTimeSlice(ProcessPriority \_\_pri);

每一个进程所分配的时间是它的优先级乘2再加20毫秒

void finishTimeSlice();

如果有正在执行的进程，保存他的PC的值，对动态优先级进行一次退化，再设置阻塞时间，把这个执行队列中的进程调入就绪队列中

1. 任务管理器：

任务管理是了解我们操作系统当前进程各类信息的图形化界面：



图8.1

从图8.1可知，透过基本的任务管理器界面我们可以了解到，进程的名称，ID，状态等各类信息，以及3条进程队列中的进程数目，所有显示的参数都是实时刷新的，以便我们对操作系统当前的状态有更好的掌控。

同时借助该界面我们还可以实现进程创建与删除：

（a）新建进程

进程的创建如图8.2所示，输入正确的打开进程文件的路径即可创建新的进程。

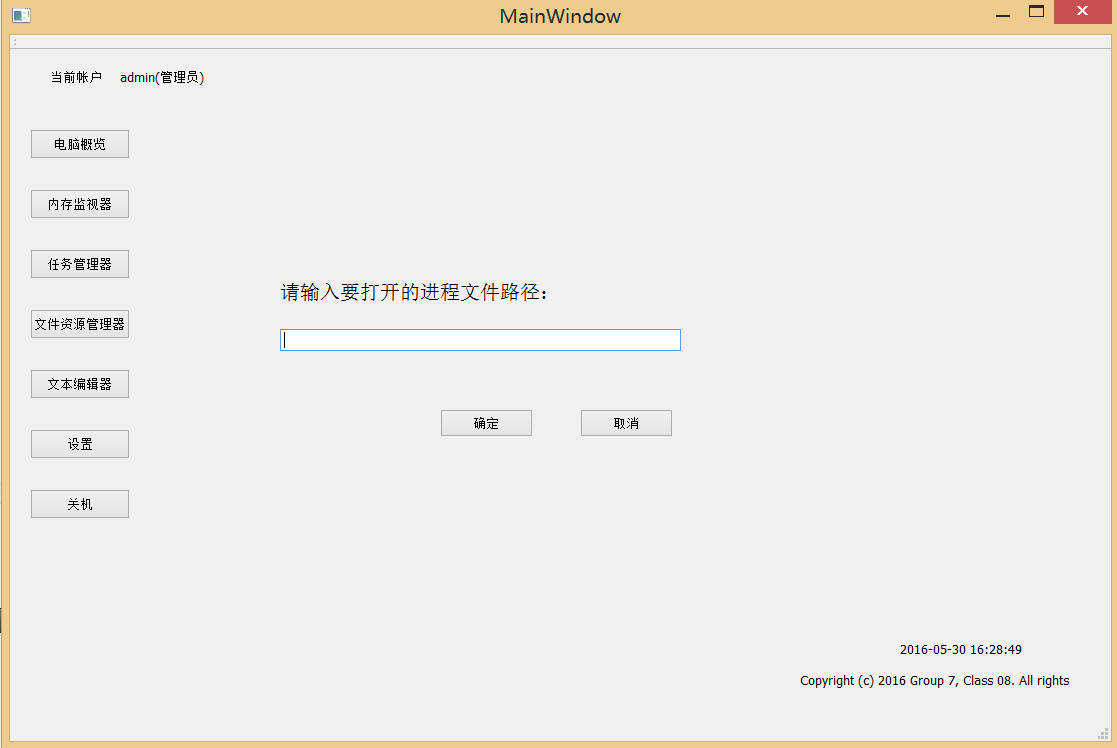


图8.2

但是，若是文件路径下不存在或非进程文件，则会报错：



图8.3

（b）删除进程：

首先单击选中要删除的进程条目，再点击右下角的删除进程即可。如图8.4和图8.5所示.

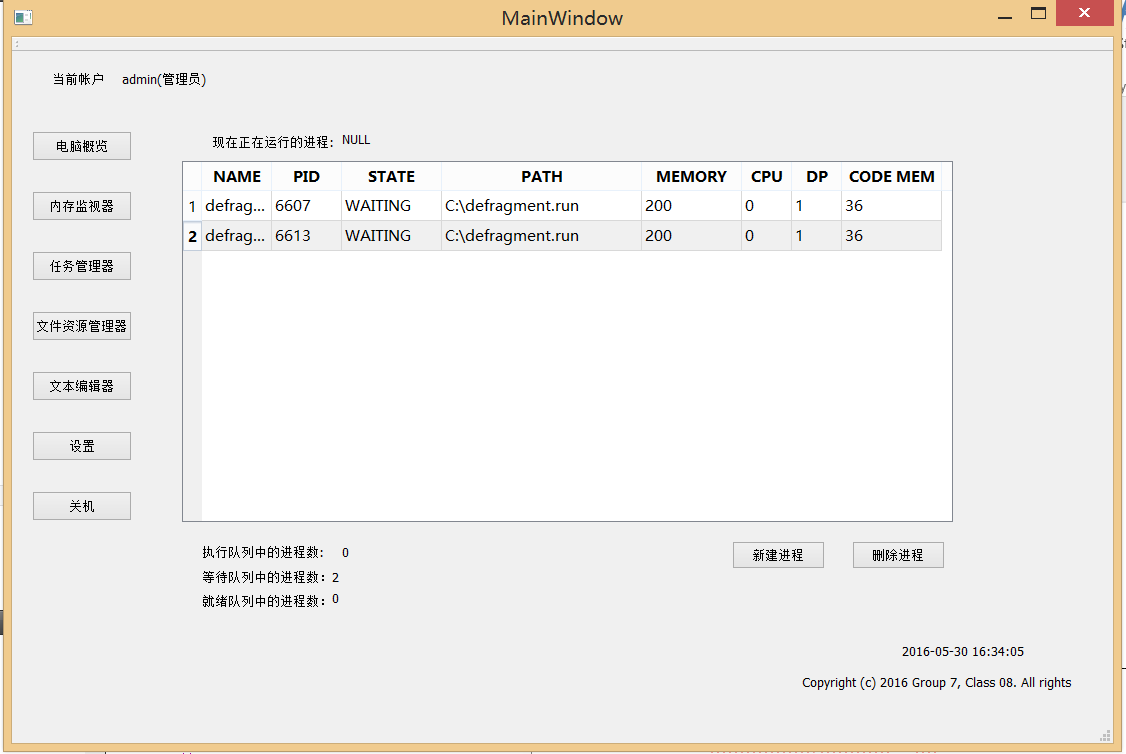


图8.4

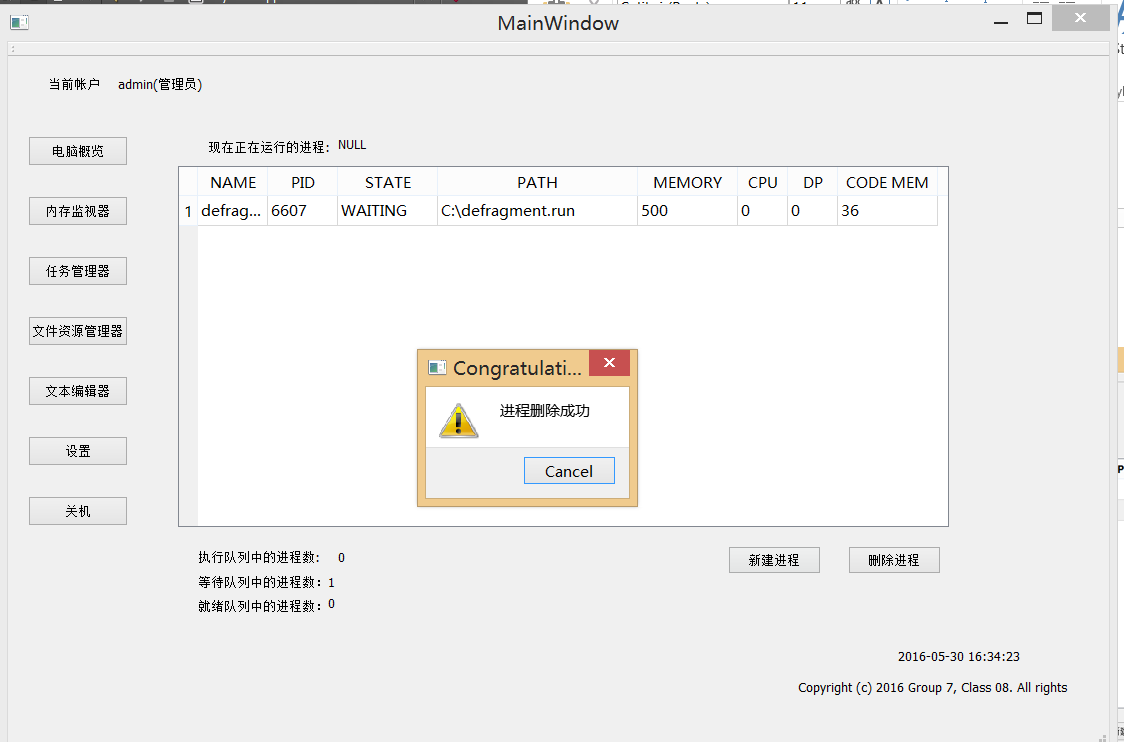


图8.5

你也可以选择单击右键进行进程的删除

（c）进程详细显示：

首先选中想要查看的特定进程，选中‘查看该进程详细’即可进入进程详细信息界面，如图8.6所示。

进程内部标识符：进程的ID等属性

进程外部标识符：进程的名称，文件路径等属性

进程状态信息：进程的PC，代码段内存等信息

进程调度信息：进程状态，优先级，时间等信息

进程页表：进程页号，有效位等信息。

段信息：进程的段信息

段表示：黑色代表代码段，剩余颜色代表数据段，白色代表为占用，用来展示碎片处理算法。

