**进程调度的设计与实现**

1. 中文摘要

无论是在批处理系统还是分时系统中，用户进程数一般都多于处理机数、这将导致它们互相争夺处理机。另外，系统进程也同样需要使用处理机。这就要求进程调度程序按一定的策略，动态地把处理机分配给处于就绪队列中的某一个进程，以使之执行。进程调度常用的算法有：先来先服务调度，优先级调度调度，时间片轮转调度和多级反馈队列调度。由于处理机是最重要的计算机资源，提高处理机的利用率及改善系统必（吞吐量、响应时间），在很大程度上取决于处理机调度性能的好坏，因而，处理机调度便成为操作系统设计的中心问题之一。

1. 关键词

进程调度 算法 多级反馈队列

1. 前言

1、目的与意义

（1）、综合应用下列知识点设计并实现操作系统的进程调度：邻接表，布尔数 组，非阻塞输入，图形用户界面 GUI，进程控制块，进程状态转换，多级反馈队列进程调度算法。

（2）、加深理解操作系统进程调度的过程。

（3）、加深理解多级反馈队列进程调度算法。

2、内容

（1）、采用一种熟悉的语言，如 C、PASCAL 或 C++等，编制程序，最好关键代码采用 C/C++，界面设计可采用其它自己喜欢的语言。

（2）、采用多级反馈队列调度算法进行进程调度。

（3）、每个进程对应一个 PCB。在PCB 中包括进程标识符pid、进程的状态标识 status、进程优先级priority、进程的队列指针next 和表示进程生命周期的数据项life（在实际系统中不包括该项）。

（4）、创建进程时即创建一个 PCB，各个进程的 pid 都是唯一的，pid 是在 1到 100 范围内的一个整数。可以创建一个下标为 1 到 100 的布尔数组，“真”表示下标对应的进程标识号是空闲的，“假”表示下标对应的进程标识号已分配给某个进程。

（5）、进程状态 status 的取值为“就绪ready”或“运行run”，刚创建时，状态为“ready”。被进程调度程序选中后变为“run”。

（6）、进程优先级 priority 是0 到49 范围内的一个随机整数。

（7）、进程生命周期 life 是 1 到5 范围内的一个随机整数。

（8）、初始化时，创建一个邻接表，包含 50 个就绪队列，各就绪队列的进程优先级 priority 分别是0 到49。

（9）、为了模拟用户动态提交任务的过程，要求动态创建进程。进入进程调度循环后，每次按 ctrl+f 即动态创建一个进程，然后将该PCB 插入就绪队 列中。按 ctrl+q 退出进程调度循环。

（10）、在进程调度循环中，每次选择优先级最大的就绪进程来执行。将其 状态从就绪变为运行，通过延时一段时间来模拟该进程执行一个时间片 的过程，然后优先级减半，生命周期减一。设计图形用户界面 GUI，在窗口中显示该进程和其他所有进程的PCB 内容。如果将该运行进程的生命周期不为 0，则重新把它变为就绪状态，插入就绪队列中；否则该进程执行完成，撤消其 PCB。以上为一次进程调度循环。

（11）、在上机实现该程序之后，要求写出实验报告，其中包括实验名称、 实验目的、实验内容、程序的主要流程图、实验心得和主要源程序清单等。

四、算法设计

1、采用关键技术

MFC对话框图形界面，多级反馈队列调度算法。

多级反馈队列调度算法：

（1）、进程在进入待调度的队列等待时，首先进入优先级最高的Q1等待。

（2）、首先调度优先级高的队列中的进程。若高优先级中队列中已没有调度的进程，则调度次优先级队列中的进程。

（3）、对于同一个队列中的各个进程，按照时间片轮转法调度。比如Q1队列的时间片为N，那么Q1中的作业在经历了N个时间片后若还没有完成，则进入Q2队列等待，若Q2的时间片用完后作业还不能完成，一直进入下一级队列，直至完成。

（4）、在低优先级的队列中的进程在运行时，又有新到达的作业，那么在运行完这个时间片后，CPU马上分配给新到达的作业（抢占式）

2、主要设计思想

（1）对于多级反馈队列调度算法的后台实现，采用随机数的方式生成一个进程PCB，利用链表的方式存储这些进程PCB，将新生成的挂于所在优先级的那一个链表的最后面，从而生成一个具有优先级的进程PCB队列，并利用一个对话框将生成的队列显示出来。对话框中有一个创建进程的按钮，每按一次生成一个进程PCB。

（2）在执行的过程中严格按照多级反馈队列调度算法的方式进行执行，分别有全部执行与逐条执行的选择（全部执行时无法创建新进程，逐条进行则可以），执行时首先从优先级最高的地方开始扫描，如果该链表不为空，先到先服务，从链头开始执行，采用时间片轮转的方式开始执行，并将life参数减1，如果life不为0，则将优先级减半并插入到相应的优先级链尾。

（3）每次执行都从优先级最高的开始扫描，确保每次都能把优先级最高的进行执行，包括逐条执行时候新产生的进程。

3、设计流程

（1）创建一个MFC对话框

（2）编写进程节点类PCB

（3）编写OnInitDialog()函数，初始化对话框

（4）编写更新函数，用于更新对话框的显示

（5）编写创建进程按钮，将新建的进程PCB插入到链表中，并调用更新函数进行显示

（6）编写执行进程按钮，从优先级最高的链开始，将所有进程执行完，并及时更新对话框显示

（7）编写逐条执行按钮，从优先级最高的链开始，执行找到的第一个进程PCB，并及时更新对话框显示

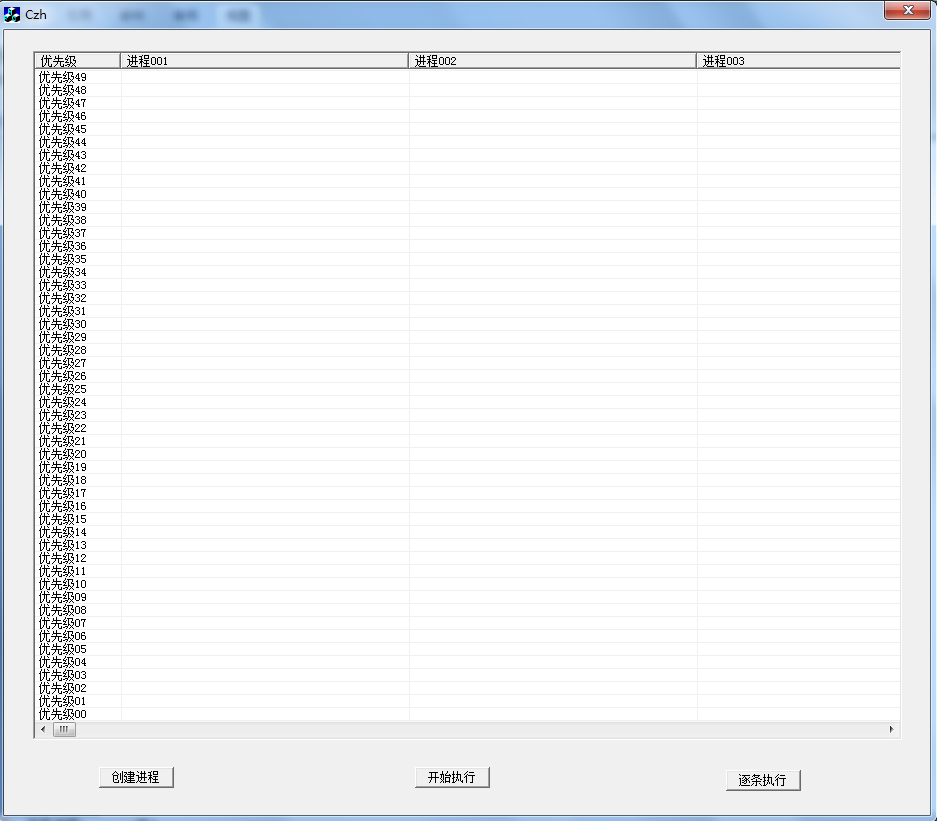
五、所采用软硬件平台

系统：win7

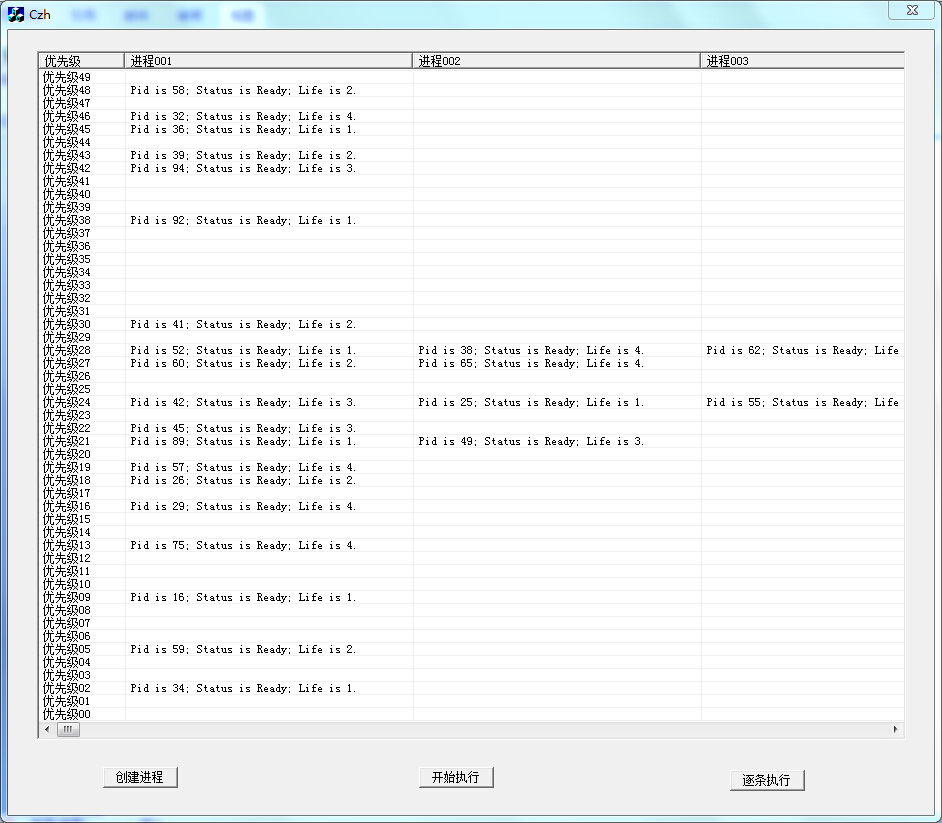
所用软件：vs2012

六、结果与结果分析

1、初始化

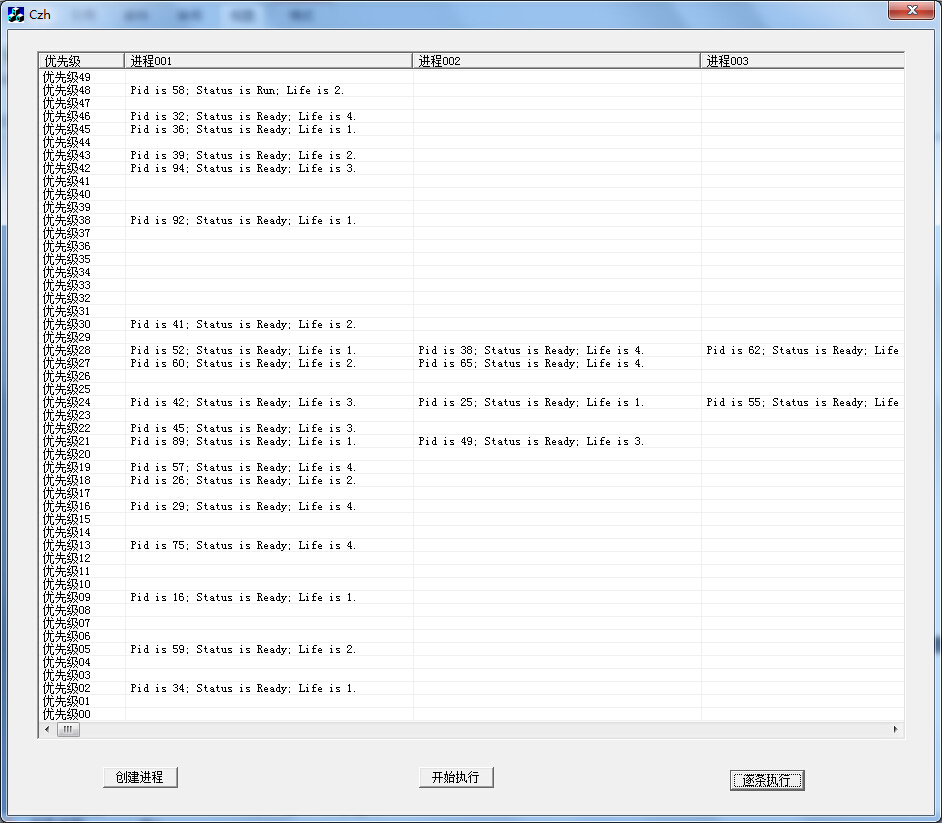


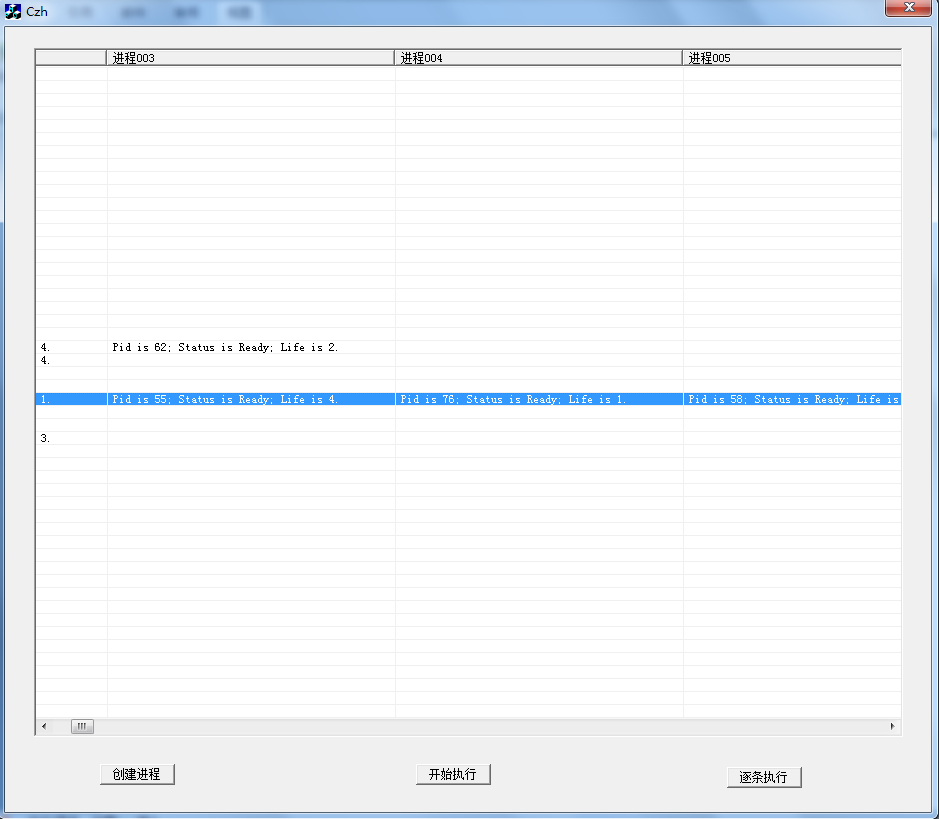
2、创建进程



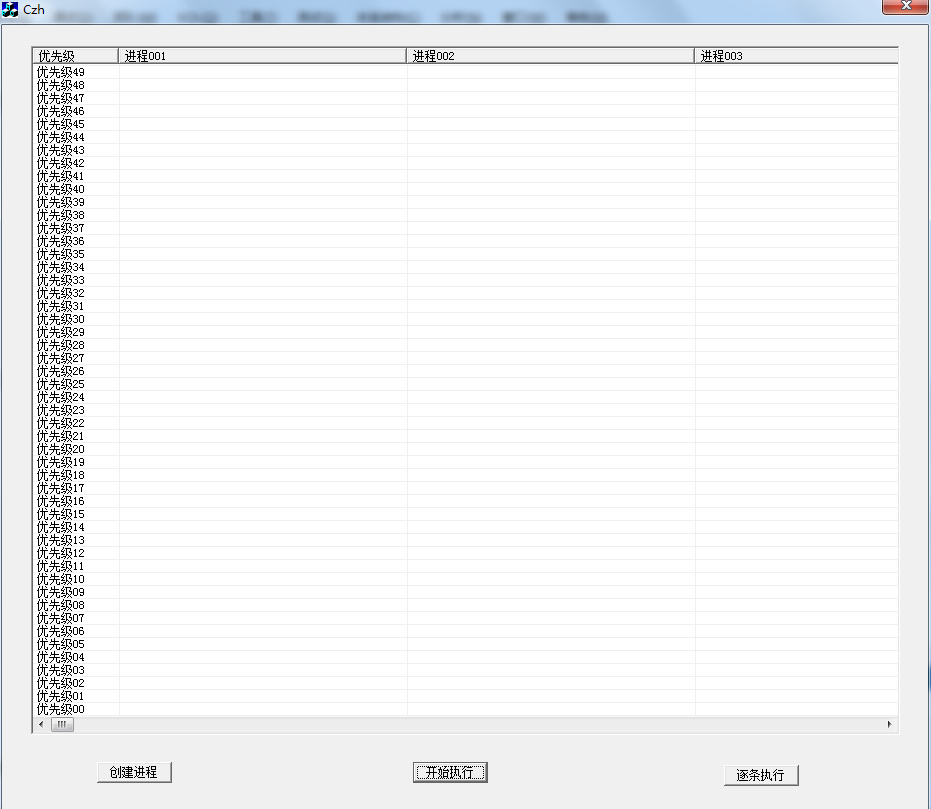
3、逐条执行进程

（1）显示正在执行



（2）执行结束，由于life不为0，转到优先级24

4、全部执行至结束



5、结果分析

（1）该结果基本符合多级反馈队列调度算法的要求，能够满足优先级、先来先服务、时间片轮转和抢占式的要求，利用MFC有比较好的显示效果。

（2）不足：由于没有采用多线程的技术，所以在运行“执行所有”的功能时，只能安静的看其做完，稍有动作就会死机，显示不下去，并且要是进程较多，会有比较久的等待时间。

七、参考文献

《操作系统教程》机械工业出版社，作者谢旭升、朱明华、张练兴、李宏伟

《百度百科》

附录：

主要功能模块

（1）编写进程节点类PCB

#if !defined(BB)

#define BB

bool static pi[101];

#endif

class node

{

public:

int pid;//进程编号

int status;//是否正在运行标志

int priority;//优先级

int life;//生命周期

node \*next;//指针指向先一个节点

//bool static pi[101];

node()//初始化

{

srand( (unsigned)time(NULL));//srand()函数产生一个以当前时间开始的随机种子.应该放在for等循环语句前面 不然要很长时间等待

while(1)

{

int p=rand()%101;//随机产生进程编号

if(p!=0)

if(!pi[p])

{

pid=p;

pi[p]=true;

break;

}

}

status=0;//标志位不在运行

priority=rand()%50;//随机产生优先级

life=0;

while(life==0)//随机产生生命周期

{

life=rand()%6;

}

next=0;

}

};

（2）编写OnInitDialog()函数，初始化对话框，在OnInitDialog()函数中加入以下代码

BOOL CCzhDlg::OnInitDialog()

{

CDialog::OnInitDialog();

// Add "About..." menu item to system menu.

// IDM\_ABOUTBOX must be in the system command range.

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != NULL)

{

CString strAboutMenu;

strAboutMenu.LoadString(IDS\_ABOUTBOX);

if (!strAboutMenu.IsEmpty())

{

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

// Set the icon for this dialog. The framework does this automatically

// when the application's main window is not a dialog

SetIcon(m\_hIcon, TRUE); // Set big icon

SetIcon(m\_hIcon, FALSE); // Set small icon

// TODO: Add extra initialization here

for(int i=0;i<50;i++)

root[i]=0;

char a[]={"进程xxx"};

int j,i;

mylist.ModifyStyle( 0, LVS\_REPORT); // 报表模式

mylist.SetExtendedStyle(mylist.GetExtendedStyle()|LVS\_EX\_GRIDLINES|LVS\_EX\_FULLROWSELECT); // 间隔线+行选中

//生成表头

mylist.InsertColumn(0,"优先级");

for( i=1;i<=101;i++)

{

if(i<10)//小于10，表示为00X

{

a[4]=48;

a[5]=48;

a[6]=48+i;

}

else

if(i<100)//大于10小于100，表示为0XX

{

a[4]=48;

a[5]=48+i/10;

a[6]=48+i%10;

}

else//大于100，表示为1XX

{

a[4]=49;

a[5]=48;

a[6]=48+i%100;

}

mylist.InsertColumn(i,a);

}

//设置列的宽度。

CRect rect;

mylist.GetClientRect(rect);

mylist.SetColumnWidth(0,rect.Width()/10); //第一列较小

mylist.SetColumnWidth(1,rect.Width()/3); //其他都比较大，由于进程中记录的信息较多

for( i=2;i<=100;i++)

{

mylist.SetColumnWidth(i,rect.Width()/3);

}

//初始化第一列

mylist.DeleteAllItems();

char b[]="优先级xx";

for(j=49,i=0;j>=0;j--,i++)

{

if(j>=10)

{

b[6]=48+j/10;

b[7]=48+j%10;

}

else

{

b[6]=48;

b[7]=48+j;

}

mylist.InsertItem(i,b);

}

return TRUE; // return TRUE unless you set the focus to a control

}

（3）编写更新函数，用于更新对话框的显示，编写update函数如下：

void CCzhDlg:: update(int priority)

{

//关闭窗口的更新

LockWindowUpdate();

//获得当前更新的优先级的所在行数，越大则越靠前

int curent=49-priority;

CString str,str1,str2,str3;

node \*n=root[priority];

int i=1;

//更新一整行

while(n!=0)

{

//处理显示内容

str="";

str1.Format("%d", n->pid);

if(n->status==0)

str2="Ready";

else

str2="Run";

str3.Format("%d", n->life);

str="Pid is "+str1+"; Status is "+str2 +"; Life is "+str3+".";

mylist.SetItemText(curent,i,str);

i++;

n=n->next;

}

//处理最后一个显示为空，由于一行中经常有更新完最后一个已删除的情况

str="";

mylist.SetItemText(curent,i,str);

UnlockWindowUpdate();//打开窗口的更新

::UpdateWindow(mylist.m\_hWnd);

}

（4）编写创建进程按钮，将新建的进程PCB插入到链表中，并调用更新函数进行显示

void CCzhDlg::OnBnClickedButton1()

{

// TODO: 在此添加控件通知处理程序代码

node \*n=new node();

node \*current;

//产生进程，并将其挂在所在优先级的最后面

current=root[n->priority];

if(current==0)//为空则直接挂上

{

root[n->priority]=n;

}

else//不空则挂在最后面

{

while(current->next!=0)

{

current=current->next;

}

current->next=n;

}

update(n->priority);//调用更新函数，更新所在行

}

（6）编写执行进程按钮，从优先级最高的链开始，将所有进程执行完，并及时更新对话框显示

void CCzhDlg::OnClickedButton2()

{

// TODO: 在此添加控件通知处理程序代码

int i=49,nn1,nn2;

node \*n,\*n1;

//从优先级最高的地方开始检测，如果存在进程则将其运行

for(;i>=0;i--)

{

while(root[i]!=0)//链表不为空，持续执行

{

//将所在结点设置为正在运行，并更新

n=root[i];

n->status=1;

nn1=n->priority;

update(nn1);

//睡眠3秒，，置为运行时间片到，并检测其life是否为0

::Sleep(3000);

n->life--;

n->status=0;

if(n->life)

{

//不为零，优先级减半，去掉原本所在位置，并将其挂在新优先级最后面

n->priority=n->priority/2;

nn2=n->priority;

n1=root[n->priority];

root[i]=root[i]->next;

if(n1==0)

{

root[n->priority]=n;

n->next=0;

}

else

{

while(n1->next!=0)

{

n1=n1->next;

}

n1->next=n;

n->next=0;

}

//更新做过修改的那两行

update(nn1);

update(nn2);

}

else//为零，则去掉原本所在位置，并更新所在行

{

root[i]=root[i]->next;

pi[n->pid]=true;

update(nn1);

delete n;

}

//break;

}

}

}

（7）编写逐条执行按钮，从优先级最高的链开始，执行找到的第一个进程PCB，并及时更新对话框显示

void CCzhDlg::OnClickedButton3()

{

// TODO: 在此添加控件通知处理程序代码

int i=49,nn1,nn2;

node \*n,\*n1;

//从优先级最高的地方开始检测，如果存在进程则将第一个进程运行，并结束

for(;i>=0;i--)

{

if(root[i]!=0)

{

//将所在结点设置为正在运行，并更新

n=root[i];

n->status=1;

nn1=n->priority;

update(nn1);

//睡眠3秒，，置为运行时间片到，并检测其life是否为0

::Sleep(3000);

n->life--;

n->status=0;

if(n->life)

{

//不为零，优先级减半，去掉原本所在位置，并将其挂在新优先级最后面

n->priority=n->priority/2;

nn2=n->priority;

n1=root[n->priority];

root[i]=root[i]->next;

if(n1==0)

{

root[n->priority]=n;

n->next=0;

}

else

{

while(n1->next!=0)

{

n1=n1->next;

}

n1->next=n;

n->next=0;

}

//更新做过修改的那两行

update(nn1);

update(nn2);

}

else//为零，则去掉原本所在位置，并更新所在行

{

root[i]=root[i]->next;

pi[n->pid]=true;

update(nn1);

delete n;

}

break;//结束寻找

}

}

}

2、