**操作系统项目文档**

邱峰 梁圃核

**项目简介**

本次项目以《自己动手写操作系统》中chapter7/g中的代码为基础，对其进行了修改与扩充，实现了更多功能。

**算法改进**

**为进程增设状态，用于显示查看进程的工作情况**

typedef enum

{

kRUNNING, //运行

kREADY, //准备（和终止一样）

kRUNNABLE //就绪

}STATE;

**增设改变进程状态**

命令：

Start processID //启动 id为processID的进程

Kill processID //结束 id为 processID的进程

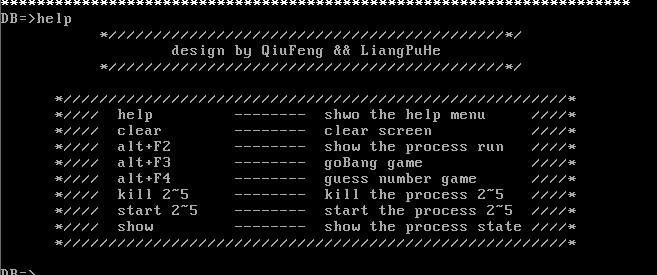
实现思想：

在调度算法中，指定调度算法仅选取state为runnable的进程，因此，kill processID时，将对应的进程state修改为kReady，start processID时，相应的重新修改其进程状态为runnable。

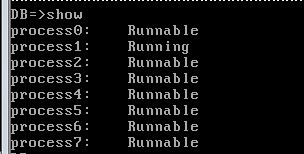
**增加命令菜单**

命令：

Help



Show



输入show命令即可查看所有进程状态

**修改调度算法，采用多级队列反馈调度**

实现思想：

由于进程数量并不是很多，因此添加2个队列firstQueue和secondQueue，分别存放不同优先级的进程。

初始时：对优先级大于等于10的进程加入到firstQueue中，其余的进程加入到secondQueue中。给定第一个队列中的所有进程2个时间片运行，给定第二个队列中的进程的时间片与其优先级相同。

调度：当一个进程的时间片运行完毕时，系统进入到调度算法，调度算法会从先扫描firstQueue中的进程，一旦firstQueue队列不为空，取出firstQueue中的第一个进程，调度结束（因此，在firstQueue调度中，采用FCFS的调度方法）。否则，在所有的secondQueue队列中，找出一个优先级最高的，并且state为kRUNNABLE的进程，让该进程运行（因此，在secondQueue中，采用的是优先级调度）。

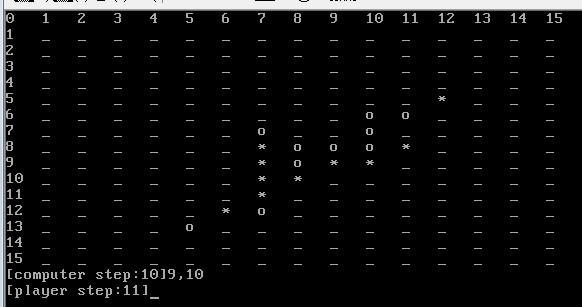
队列调整：当一个进程的时间片运行完时，如果该进程原来在firstQueue中，将其调整到secondQueue当中，并给与其优先级相同的时间片，以便在调度算法找secondQueue中进程时，能够调度到该进程。

重新初始化：如果所有的进程都已经调度结束（也就是所有的进程都执行完了给定的时间片）,则重新根据他们的优先级，按照初始化的方法，重新加入到firstQueue和secondQueue当中。

Ps：进程调度时，只会选择状态为kRUNNABLE的进程。

**在控制台3处增加goBangGame游戏**

* 实现与AI对战
* \*为电脑 o为玩家



五子棋的设计思想是利用屏幕打出五子棋棋盘，用户通过输入对应的坐标代表在该处下子，同时刷新屏幕。

AI的设计循环棋盘上的每一个可以放棋子的地方，返回一个该格子的“放子优先级”，选取一个优先级最大的地方为AI的放置处。

优先级的设计理念如下：

if (attack>=5) return 20; //5攻击

if (defence>=5) return 19; //5防御

if (attack==4 && (attackFree1>=1 && attackFree2>=1)) return 18; //4攻击 2边

if (attack==4 && (attackFree1>=1 || attackFree2>=1)) return 17; //4攻击 1边

if (defence==4 && (defenceFree1>=1 || defenceFree2>=1)) return 16; //4防御

if (attack==3 && (attackFree1>=2 && attackFree2>=2)) return 15; //3攻击 2边

if (defence==3 && (defenceFree1>=2 && defenceFree2>=2)) return 14; //3防御 2边

if (defence==3 && (defenceFree1>=2 || defenceFree2>=2)) return 13; //3防御 1边

if (attack==3 && (attackFree1>=2 || attackFree2>=2)) return 12; //3攻击 1边

if (attack==2 && (attackFree1>=3 && attackFree2>=3)) return 11; //2攻击 2边

if (defence==2 && defenceFree1+defenceFree2>=3) return 10; //2防御 2边

if (defence==2 && defenceFree1+defenceFree2>=3) return 9; //2防御 1边

if (attack==1 && attackFree1+attackFree2>=4) return 8;

if (defence==1 && defenceFree1+defenceFree2>=4) return 7;

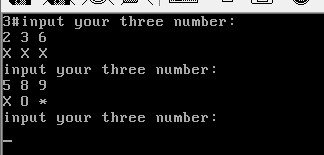
return 6;

其中attack为该地方如果是电脑放下后，最大的连接棋子数。

Defences为如果这个地方是玩家放下后，最大的连接棋子数。

AttackFree为连接棋子结束后，两边能有几个可以放棋子的地方被扩展

**在控制台4处增设游戏猜数字**



电脑随机产生3个数字，玩家进行猜数字游戏，如果所猜数字在结果中不存在，提示X，如果所猜数字在结果中存在，但位置不对，提示o，如果正确，提示\*。若玩家在四次机会内猜中，胜。

关于随机数的产生，这个方法我们试了好多种，最后是利用sys\_get\_tick函数获取系统运行时间的一个随机数实现的。经过函数变换，最终得到三个不同的0~10范围内的随机数。

三个不同的随机数获取方法如下：

a[0]=sys\_get\_ticks()%10;

a[1]=a[0];

a[2]=a[0];

while(a[1]==a[0])

a[1]=(a[1]\*sys\_get\_ticks()+7)%10;

printf("%d %d %d\n",a[0],a[1],a[2]);

t=0;

while(a[2]==a[0]||a[2]==a[1])

{

a[2]=(a[0]+a[1]+t)%10;

t++;

}将玩家的输入与机器产生随机数进行比较，显示结果。