内存管理项目程序说明

一.项目简介

项目名称：内存管理模拟程序

开发环境：Windows平台的c#(.NET Framework)

项目概述:假设每个页面可存放10条指令，分配给一个作业的内存块为4。模拟一个作业的执行过程，该作业有320条指令，即它的地址空间为32页，目前所有页还没有调入内存。

二.界面简介

最终的界面如下图所示：

界面的左边展示每个物理页存储的指令信息。右边有页表信息以及开始按钮，清空数据按钮，速度调节拉杆，选择调度算法按钮。

三.调度算法设计

采用了LRU和FIFO两种调度算法。LRU即最近最少使用算法，考虑的是对于当前指令的前一段时间使用过的指令，若发生中断，则按照算法思想找出对应的内存块替换。FIFO算法即先进先出调度算法，考虑的是指令进入内存块的先后顺序，若发生中断，则先进的指令被新指令替换掉。

四.程序结构介绍

1.全局静态类Use

- Container[2,4]:一个二行四列的二维数组，列：每一列代表一个物理内存页的信息。行： 第一行存的是具体指令所属逻辑地址的页号（如第252条指令属于逻辑地址的25页);第二行存的是此时此刻内存块中四条指令在逻辑页的最高位置。
- buf[320]:一个长度为320的数组，存的是随机生成的指令编号。
- InterruptCount:记录缺页的次数。
- StopTime:逻辑页调入内存块停滞时间，初始值设为1000ms，并可根据游标的滑动变换速度。
- RandomNum(): 根据要求50%的指令是顺序执行的，25%是均匀分布在前地址部分，25%是均匀分布在后地址部分。所以设定周期为4的随机赋值操作。先用随机数函数生成$[0,320)$中的一个数记作$temp$,然后在$[0,temp)$和$[temp,320)$中分别随机生成一个数$temp1,temp2$。于是在一个周期内$buf[i]=temp1,buf[i+1]=temp1+1,buf[i+2]=temp2,buf[i+3]=temp2+1$，之后79个周期依次操作即可完成赋值。
- InitBuf()用来对数组Container进行初始化，其初始值全部设为-1。

2.算法类Algorithm

- 静态成员方法LRUalgorithm():根据Container存的数据判断指令是否在内存中。如果所访问指令在内存中，就显示它的物理地址，并转到下一条指令；如果不在内存中,则发生缺页,InterruptCount++。如果4个内存块中已装入作业，则需进行页面置换。
- 静态成员方法FIFOalgorithm():与前面算法不同的是，上面有一个比较置换的过程，而先进先出的调度算法一直都是从头开始往后调度置换，然后再循环回到起点继续调度置换。

3.操作类Operator

操作类用了关键词partial,partial Form1代表Form1的部分类，放在另一个文件中写。

- 静态成员方法ChangeNum():用来修改界面一些Lable和Button的Text值，以动画的方式展现出来，加强了可视化的效果。

4.界面类Form1

定义了一系列的Button,Label来实现内存管理项目的模拟。

- 私有方法ButClick():绑定了开始模拟的行为，点击之后，程序会自动跳转到LRUalgorithm()中执行相应的操作，并在运行完算法之后，在界面下方输出320次后结果的缺页率。

- 私有方法ClearDataClick():绑定了清除数据的行为，点击之后，程序会自动清空一些数组或标签的值，或者重新初始化赋值，或者置为NULL。

五.不足之处

- 项目使用的随机指令生成算法不完善：虽然算法实现了1：1：4=均匀分布前地址：均匀分布后地址：顺序执行。但注意到前地址主要集中在下标为0,4,8(即下标i%4==0)的地方;后地址主要集中在下标为2，6，10(即下标i%4==2)的地方,这样一来由于“不完全随机”，可能会对缺页率有小的影响。**ps:**经过后期测试，影响小于1%。