请求调页存储管理方式模拟程序项目文档

2152054 沈加豪

1. **项目要求**

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。

在模拟过程中，如果所访问指令在内存中，则显示其物理地址，并转到下一条指令；如果没有在内存中，则发生缺页，此时需要记录缺页次数，并将其调入内存。如果4个内存块中已装入作业，则需进行页面置换。

所有320条指令执行完成后，计算并显示作业执行过程中发生的缺页率。

1. **开发/运行环境**

C# Winform

1. **界面设计**

由于C#窗体应用的便捷性，对于简单的模拟程序是一个良好的设计载体，所以本项目采用了C# Winform进行设计与实现。

初始页面：



窗口左侧是基本信息，包含作业指令总数为320，每页存放的指令数为10，作业占用的内存页数为4，以及页面置换算法的选择，可选为FIFO算法和LRU算法，指令执行顺序选择，可选为顺序执行、随机执行和混合执行，左下部是当前执行指令号的强调。

窗口中部是模拟页面，总共四个页，每页可存放10条指令，编号0~9，页面上方标记了页数和驻留周期数；中下部是四个按键，分别是“单步执行”、“连续执行”、“重置”和“暂停”。其中，点击“单步执行”，则会取一条指令运行；点击“连续执行”，则会连续取指令运行，点击“暂停”可以暂停正在连续执行的取指令操作，点击“暂停”后，“连续执行”键会变成“继续”键，再点击会继续操作，点击“重置”键会重置页面取指令的记录和操作，回到初始状态。

窗口右侧是已执行的指令，以列表形式呈现，包含执行指令的地址、缺页、换出页、换入页；右下部是缺页数和缺页率。

执行过程：



执行的指令在页面中用红色标注，并且上方的页数会发生变化，驻留时长也会变成“正在调用”，已执行指令的列表最下方的数据即为当前执行指令的信息，左下也有强调。如图表示正在执行3号指令。

缺页表示：





（执行40号指令时发生缺页并有调入调出页）

当发生缺页，右下方的缺页数会加一，缺页率则随着每条指令执行不断发生变化，且列表中调用的指令信息在“缺页”一栏会显示“是”，若有调出页和调入页，则也会显示。

暂停界面：



连续执行的过程若发生暂停，则会如图所示呈现暂停状态，且连续执行键变成继续键。

重置界面：



重置界面后则会回到开始时的页面状态。

更换算法或执行顺序：



无论在单步执行或连续执行过程中，都可中途改变算法和执行顺序，那么程序就会根据新的算法或顺序进行演示，过程连续。

执行完毕：



320条指令执行完毕后，左下角会显示完毕信息，右下角显示的缺页数和缺页率即为执行完毕后的最终结果，如果想要继续观察演示，则点击重置即可。

1. **模拟结果**

顺序执行，采用FIFO算法，缺页率为10%



随机执行，采用FIFO算法，缺页率为90%



混合执行，采用FIFO算法，缺页率为44.06%



顺序执行，采用LRU算法，缺页率为10%



随机执行，采用LRU算法，缺页率为87.81%



混合执行，采用LRU算法，缺页率为42.81%



1. **算法设计**

本程序支持FIFO算法和LRU算法进行调页操作，三种执行顺序也都支持。

FIFO算法的实现思想：设置一个指针block\_pointer，先指向第一个内存页框，最先进入的页面一定在此页框，后续的页面陆续进入后，当四个页框都填满时，由于指针仍指在首位，则替换在首位的页面，换入新页面，此时指针后移一位，也就是第二位，则第二位成为最先进入内存的页面，此时继续取指令，若发生缺页，则将指针指向的页面替换即可。由于指针指向的始终都是最先进入内存的页面，而调出则始终调出指针指向的页面，即实现了FIFO算法调页。

LRU算法的实现思想：设置一个计时数组time\_block[]，计算每个页面在内存中的驻留时长，当发生缺页需要调页时，就将驻留时间最长的页面调出即可。要注意的点在于，若页面在内存中被再次使用，则需刷新驻留时长，以保证其不为最近未使用页面。

每条指令被执行后，下一条指令的的取法取决于指令执行方式是顺序的还是随机的还是混合执行。顺序和随机很好实现，混合执行方式采用的是概率方式，有50%的概率顺序，25%的概率向前随机取指令，25%概率向后随机取指令。要注意的是visited数组存放是否被执行过，以便防止遍历重复。

1. **可行性分析**

本程序的演示功能较为方便，对于两种算法、多种执行顺序都有模拟，并且可以单步执行，以便观察，同时也可连续执行，速览执行过程。缺页数和缺页率实时显示，可以观察数据变化，执行过程也详细可查，可行性较强。