操作系统之

内存管理

张仕奇 1152671

项目名称：内存管理

开发工具：Java

项目要求：

1： 动态分区分配方式

内容

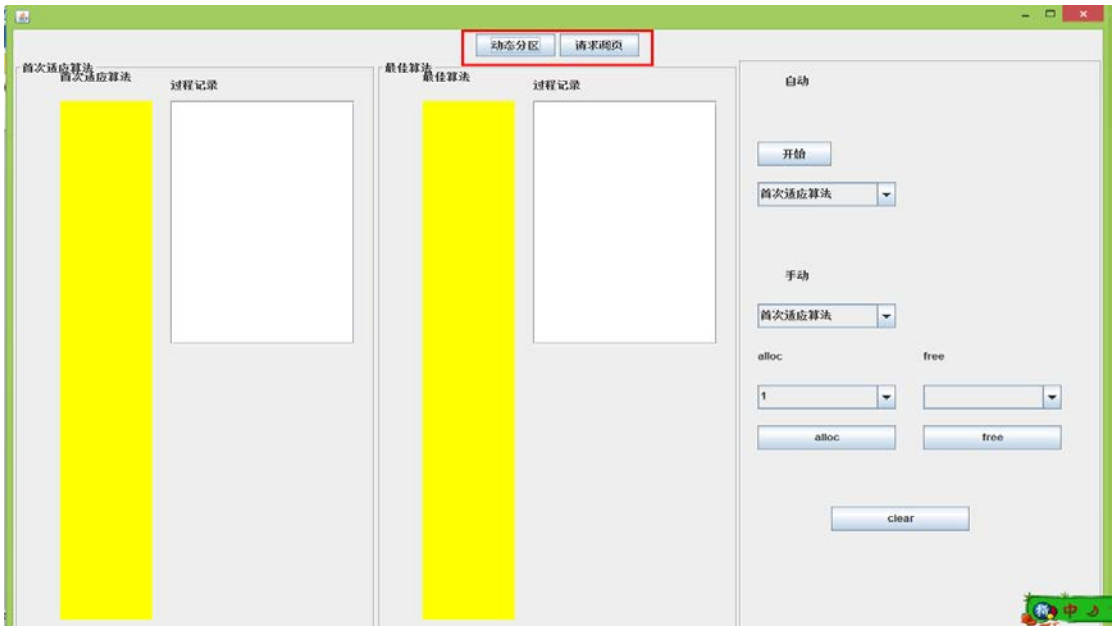
假设初始态下，可用内存空间为 640K，并有下列请求序列，请分别用首次适应算法和最佳适应算法进程内存块的分配和回收，并显示出每次分配和回收后的空闲分区链的情况来。

2：请求分区分配方式

内容

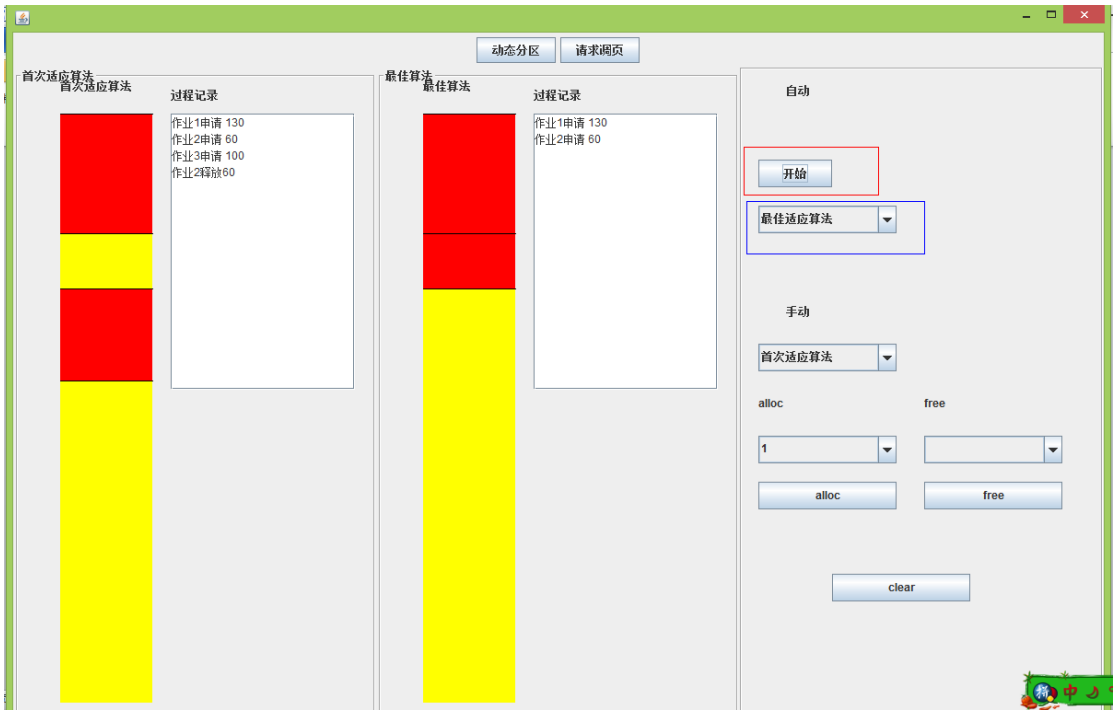
假设每个页面可存放 10 条指令，分配给一个作业的内存块为 4。模拟一个作业的执行过程，该作业有 320 条指令，即它的地址空间为 32 页，目前所有页还没有调入内存。

程序说明：

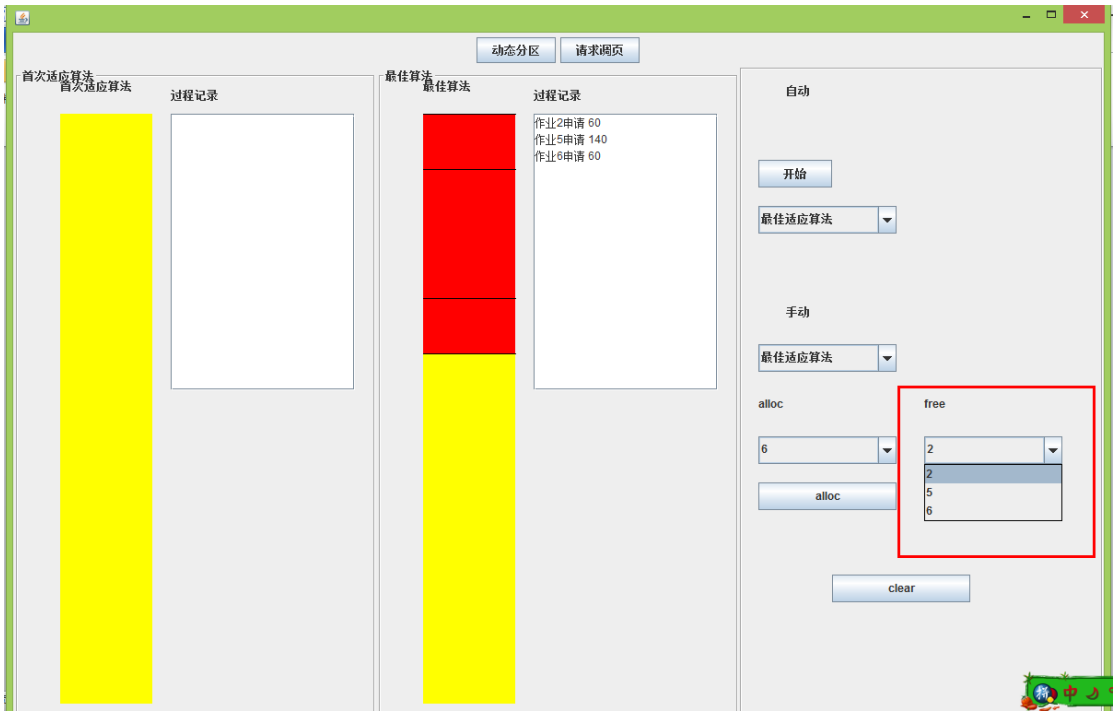


在界面的上方，有两个按钮，一个是动态分区，一个是页面置换。点击可以切换到各自的界面。以上界面的动态分区的，以下界面是页面置换的。

首先对动态分区进行说明。

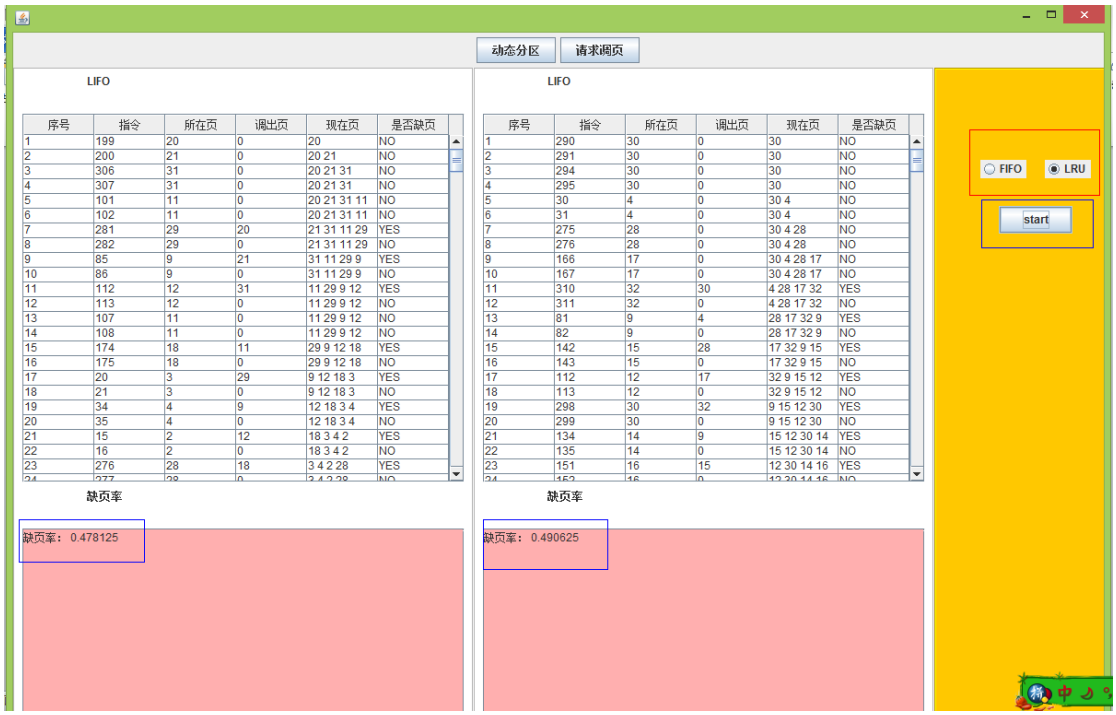


先是自动运行，下拉菜单选择算法，点击开始进行自动模拟。两个算法可以同时运行。还可以手动进行分配和释放。



通过下列菜单选择进程号和算法，为 1-6，选择后 free 的下拉菜单会出现相应的进程号。点击 free 可以将该进程释放。在切换算法时会清空菜单和模拟图。

页面置换说明：



选择算法，然后点击 start，在左侧的 table 可以看见运行结果，在下面的 textarea 显示缺页率。有几点说明，当调出页显示为 0 的时候，说明没有页调查，现在也显示当前 4 个块中页的情况。

程序说明：

一． 动态分区

Block 用来存放分配或空闲块的上界和大小，task 用来存放任务进程的 id，大小。

首次适应算法：

Alloc：通过 id 号找到该进程记为 task，通过循环让 task.size()和空闲列表里的块的大小相比较，如果相等，则删除该空空闲块，已用块加入刚才任务的块；如果大于空闲块的上界变为原来上界减去 task.size（），大小为原来 size-task.size()，删除该空闲块，再在刚才空闲块的位置 new 一块新的块存放刚才剩下的块；如果不是以上两种情况，进入小一个比较。

Free：通过 id，找到在 usedlist 中所对应的块，然后通过循环和 freelist 中的块进行比较，如果两个块相邻能合并成一块，则将两个合并，如果不能，则将刚释放的块 new 出一个放在 freelist 中。

最佳适应算法：

Alloc 和 free 与以上基本相同，只是在 alloc 中加入 sortList（）。用来把空闲列表中块的由块的大小从小到大排序，分配是找最小的进行分配。

页面置换

fstart：FIFO 算法产生随机命令，lstart：LRU 算法产生随机命令。

```
//FIFO的页面置换算法
   int f_checkBlock(int page) {
//如果有没有用的页则将该页分配给page页
       for (int i = 0; i < 4; i++) {
           if (page == block[i]) {
               useTimes[page]+=1;
               return 1;
           }
       }
//如果有没有用的页则将该页分配给page页
       for (int i = 0; i < 4; i++) {
           if (block[i] == 0) {
               block[i] = page;
               useTimes[page]+=1;
               return 1;
           }
       }
//将第一个页面置换
       writeOut = block[0];
       block[0] = block[1];
       block[1] = block[2];
       block[2] = block[3];
       block[3] = page;
```

```
        return 0;
    }
    //LRU的页面置换算法

    int l_checkBlock(int page){
如果4个块中存在该页，则将该页使用次数+1
        for (int i = 0; i < 4; i++) {
            if (page == block[i]) {
                useTimes[page]+=1;
                return 1;
            }
        }
//如果有没有用的页则将该页分配给page页
        for (int i = 0; i < 4; i++) {
            if (block[i] == 0) {
                block[i] = page;
                useTimes[page]+=1;
                return 1;
            }
        }
        int max=500,index = 0;
        for(int i=0;i<4;i++){//找到使用次数最少的进行页面置换
            int num=block[i];
            if(useTimes[num]<max){
                max=useTimes[num];
                index=num;
            }
        }
        writeOut = block[index];
        block[index] = page;
        useTimes[page]++;
        return 0;
    }
```

项目小结：

这次项目我把两个要求都做了。总体感觉都比较简单。

对于动态分区，我认为难点在于绘图，应为以前没有用过 paintComponent,所以这次用时琢磨了比较长的时间才弄懂，一个很重要的东西是当图发生变化时要记得 repaint().还有就是 uesdlist，freelist,block 这几个建立是十分重要的。

对于页面置换，产生随机数的时候我遇到了一点困难：原来在循环中处理的时候在运行到 100 多条指令的时候会陷入死循环或溢出，通过调试我发现在我原来写的算法中如果指令是 318，或 319 时如果顺序之后，则会发生溢出或死循环。然后我加入了条件，如果发生这样情况，则在前面的指令产生随机数。

项目改进想法：

动态分区中手动运行中我还没有写条件控制，比如说如果连续分配几个进程，大小超过了总大小的问题。还有是可以把该程序改进为可以自己分配总大小和每个进程的大小。

对于页面置换，我的改进想法是能够自己设置内存块的个数，指令的总个数，每个页能存放指令的个数。

参考的指令执行顺序我觉得有一点问题。执行完一条指令之后顺序执行下一条指令，这样会导致 FIFO，LRU 的执行结果差异不大。