****

操作系统原理

实验报告

|  |  |
| --- | --- |
| 学生姓名 | 焦双城 |
| 学 号 | 8208190527 |
| 专业班级 | 计算机科学与技术1906 |
| 指导教师 | 郑美光 |
| 学 院 | 计算机学院 |
| 完成时间 | 2021/5/28 |

目录

[一 实验概述 1](#_Toc73397269)

[(一) 实验目的 1](#_Toc73397270)

[(二) 实验内容及要求 1](#_Toc73397271)

[1.实验内容 1](#_Toc73397272)

[2.实验具体要求 1](#_Toc73397273)

[二 需求分析 2](#_Toc73397274)

[三 总体设计 2](#_Toc73397275)

[（一）总体设计思路 2](#_Toc73397276)

[（二）总体工作流程图 2](#_Toc73397277)

[四 详细设计与实现 6](#_Toc73397278)

[（一）数据结构设计 6](#_Toc73397279)

[（二）进程圆动画函数的设计与实现 6](#_Toc73397280)

[（三）事件绑定函数的设计与实现 6](#_Toc73397281)

[（四）内存块动画函数的设计与实现 6](#_Toc73397282)

[（五）首次适应算法函数的设计与实现 6](#_Toc73397283)

[（六）增加新进程模块的设计与实现 7](#_Toc73397284)

[五 实验运行结果 7](#_Toc73397285)

一 实验概述

1. 实验目的

多道系统中，进程与进程之间存在同步与互斥关系。当就绪进程数大于处理机数时，需按照某种策略决定哪些进程先占用处理机。在可变分区管理方式下，采用首次适应算法实现主存空间的分配和回收。

本实验模拟实现处理机调度及内存分配及回收机制，以对处理机调度的工作原理以及内存管理的工作过程进行更深入的了解。

(二) 实验内容及要求

1. 实验内容

（1）选择一个调度算法，实现处理机调度；

（2）结合（1）实现主存储器空间的分配和回收。

2. 实验具体要求

（1）设计一个抢占式优先权调度算法实现多处理机调度的程序，并且实现在可变分区管理方式下，采用首次适应算法实现主存空间的分配和回收。

（2）PCB内容包括：进程名/PID；要求运行时间（单位时间）；优先权；状态；进程属性：独立进程、同步进程（前趋、后继）。

（3）可以随机输入若干进程，可随时添加进程，并按优先权排序；

（4）从就绪队首选进程运行：优先权-1；要求运行时间-1；要求运行时间为0时，撤销该进程；一个时间片结束后重新排序，进行下轮调度；

（5）考虑两个处理机，考虑同步进程的处理机分配问题，每次调度后，显示各进程状态，运行进程要显示在哪个处理机上执行。

（6）规定道数，设置后备队列和挂起状态。若内存中进程少于规定道数，可自动从后备队列调度一作业进入。被挂起进程入挂起队列，设置解挂功能用于将制定挂起进程解挂入就绪队列。

（7）结合实验一pcb增加所需主存大小，主存起始位置；采用首次适应算法分配主存空间。

（8）自行假设主存空间大小，预设操作系统所占大小并构造未分分区表。表目内容：起址、长度、状态（未分/空表目）。

（9）进程完成后，回收主存，并与相邻空闲分区合并。

（10）最好采用图形界面；

二 需求分析

基本原理：优先级调度算法和首次适应算法；软件（网页）的基本功能：实现点击按钮开始新一个时间片，进程在后备队列、就绪队列、挂起队列、处理机之间移动，内存块的增删动画，并且可以新增进程。

三 总体设计

（一）总体设计思路

全局设定：对于整个流程的运行设定是点击按钮来控制单个时间片的完成，即点击之后会显示当前时间片的运行结果。

初始化设定：首先初始设定的所有的进程会直接进入后备队列，然后按照优先级来进行排序，依次进入就绪队列，受到内存和道数的控制，后续新来的进程也是先放入后备队列。

运行设定：选取就绪队列中优先级较高且没有前驱/前驱已经运行完毕的进程进入处理机，由于点击按钮控制时间片这个设定，两个处理机基本上是同步的（除非就绪队列只有一个满足进入处理机的条件），每次在处理机的进程只保留一个时间片，之后就会重新进入就绪队列再进行新一轮的选择。

挂起设定：处在就绪队列以及处理机的进程可以挂起，挂起之后内存回收，就绪队列可能因此进程数小于道数，后备队列通过首次适应算法判断是否能进入就绪队列。

解挂设定：解挂的进程通过首次适应算法判断是否能进入就绪队列，为其分配内存，解挂进程的加入可能导致就绪队列进程数大于道数，故就绪队列的长度还是要比道数高的。

（二）总体工作流程图



图一 总体工作流程图1

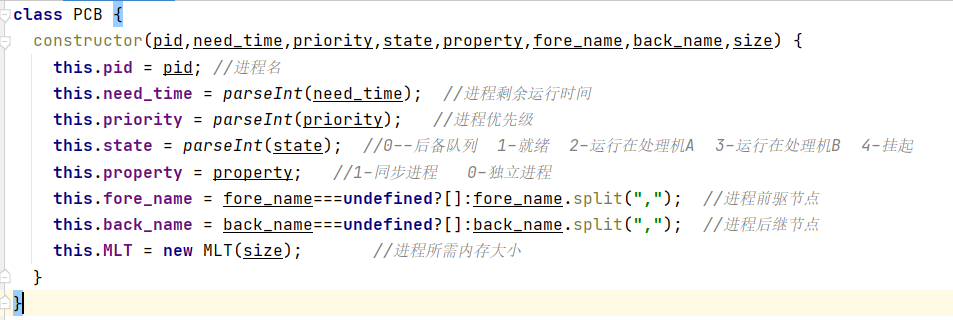


图二 总体工作流程图2

四 详细设计与实现

（一）数据结构设计

PCB属性如图二所示，其实还有两个属性，就是前驱和后继节点的PCB，由于js的对象全是地址，故可直接用fore和back两个数组来存放对应前驱和后继节点的PCB，便于后续查找前驱后继节点。



图二 PCB数据结构设计

（二）进程圆动画函数的设计与实现

设计：首先要判断后备队列里的进程圆的内部绑定数据是否已更改为状态处于就绪队列，如果有则将其的class更改为对应于非后备队列队列的class，去除后备队列的class，然后对所有带有非后备队列（not\_back）类的进程圆进行重定位，用动画来改变其位置。同样的对于带有后备队列（back）类的进程圆也需要重定位然后用动画来移动。然后直接调用内存动画函数。

实现：详情请见circle\_change函数（616-689行）

（三）事件绑定函数的设计与实现

设计：首先要取消所有进程圆的绑定事件，以免引起事件错乱，然后对于所有进程圆增加mouseover弹出详细信息框和mouseout移除详细信息框的事件，因为信息框里的内容会随时间改变，为非后备队列绑定点击事件，用于挂起和解挂，点击后改变该进程圆的绑定数据，改变就绪队列和挂起队列，然后再次执行进程圆动画函数。

实现：详情请见click\_event函数（691-839行）

（四）内存块动画函数的设计与实现

设计：内存块的动画主要包括增删动画，增删的依据是就绪队列里的数据（处理机的数据也保存在就绪队列里），根据就绪队列的增删来同步于内存块的增删。

实现：详情请见memory\_change函数（545-614行）

（五）首次适应算法函数的设计与实现

设计：首先将就绪队列里的所有起址和终址（起址加大小）取出来，然后排好序，由于没有交叉，所以就算是两边都各自排序仍然保有对应关系。然后就可以通过遍历每个gap来判断是否可以放入了，从大内存块起址开始，符合首次适应算法。如果内存分配失败，则提醒用户后备队列里的进程应内存不足而无法进入就绪队列。

实现：详情请见isInQueue函数（497-542行）

（六）增加新进程模块的设计与实现

设计：首先从文件中获取首行数据，然后指针向下偏移一行，将该数据封装成PCB之后压入后备队列中，重排后备队列里的进程圆，最后就是如果没有新数据了那么就弹出一个简易的确认框，提醒用户已无新数据。

实现：429-496行，绑定按钮点击事件

五 实验运行结果

实验运行结果见操作系统实验视频.mp4。

结束语

首先谈谈使用前端的原因，上个学期开始加入了可视化训练营，学习了前端可视化知识，第一次学习前端知识，直到现在。这是我第一次使用前端来完成课程实验，选择的原因主要是想要运用自己的专业来做一些课程实验，并在其中进一步提升自己的能力。

实验的一些处理的细节是跟室友们一起讨论的结果，他们使用的是QT，而我采用的是前端，但是代码都是我们自己想自己手打的，基本上都花了约三天才打完，主要是一些处理上的繁琐以及最近实验繁多引起的一些压抑，所以整体效率不算特别高。现在想起来也感慨万分，自己打了800多行代码，原本还想做一个全自动的进程调度，结果最后还是身心疲惫的放弃了，因为这个学期傻傻的选了很多的选修课，课程实验还蛮多的，加上课程上的学习实际还是有点差的，所以就只能暂且于此了。

在绝对的ddl的压迫下，就还是得学会做减法，学会取舍吧。暑假准备把这次使用放到github上，养成一个项目放github仓库的好习惯。

这次做的实验使用前端来说还是有一定难度的，虽然在决定之前，自己有设想过一些思路，不过在具体实现起来还是蛮复杂的，一些鼠标事件的绑定，一些鼠标事件的取绑，由于不能重画，那么只能通过改变circle的class，然后通过对类的选集来控制特定的进程圆，然后事件中又会带有动画。

不过比较好的一点就是我将很多代码块抽离成了函数，这样代码结构就清晰很多，代码也不至于过度冗余。

由于时间比较赶，内存调度是逻辑和动画同步开始做的，所以在一些细节方面可能就处理得不是很清楚了，比如上机检查老师提问的那个问题，因为我所有的图形都是通过特定的数据加以绑定并且操控运动的，在进行移动动画时，我会重新绑定数据，以此来控制内存块的增删，所以在挂起时，我仅仅做了移动就绪队列与挂起队列里的数据，而并没有对数据进行修改（当然这会导致详细信息框里的内容出问题，刚发现的一个bug），但是我每个进就绪队列里的进程，都会先执行首次适应算法来修改它的起址，所以实际对于动画来说是没有什么影响的。

然后可能我当时一些细节上没讲太清除，就是在绑定数据上，数据是通过pid进行区分的，重新绑定数据可以删除与上次数据相比缺少的数据对应的元素，也可以增加相对应新增数据的元素。这就实现了内存块的移进和移出的动画了。

最后再感激一下自己使用了前端吧，在处理一些细节上时深入地理解了一些原来懵懵懂懂的一些东西，然后想说的是顶住压力宿舍里一个人使用前端做动画也挺不容易的，他们其实也会做前端，只是怕时间可能花得太多吧，所以最后也就我一个人在做，还得抗住他们时不时的冷嘲热讽，因为毕竟我们都雪了可视化的时间，在前端里做一个动画还是比较简单的，而QT的话因为他们没深入学嘛，所以就没准备做动画了，以此由于这个差异，受点冷嘲热讽也挺正常的，不过最后我还是挺满意这次做出来的效果的。