操作系统内存管理模拟系统操作手册

设计思路

本系统旨在模拟操作系统中的两种重要内存管理机制：动态分区存储管理和动态分页存储管理。通过图形化界面，用户可以直观地观察内存分配、回收以及页面置换的过程，从而深入理解操作系统内存管理的核心原理。系统采用Python语言结合Tkinter图形库实现，遵循面向对象的设计思想，将不同功能模块封装为独立的类，确保代码的可维护性和可扩展性。

在动态分区管理模块中，系统实现了三种经典的内存分配算法：最先适应（First Fit）、最佳适应（Best Fit）和最坏适应（Worst Fit）。用户可以通过界面创建不同大小的进程，系统会根据选定的算法在可用分区表中寻找合适的空闲区进行分配。当进程释放内存时，系统会自动检测并合并相邻的空闲区，保持内存的连续性和高效利用。整个分配和回收过程都通过文本和图形两种方式实时展示，使用户能够清晰地观察内存状态的变化。

在动态分页管理模块中，系统模拟了请求式分页管理的工作流程。当访问某个页面时，系统会检查该页是否在内存中。如果在内存中，则直接计算物理地址；如果不在内存中，则触发缺页中断，需要将所需页面从外存调入内存。当内存空间不足时，系统采用先进先出（FIFO）算法选择一个页面进行置换。页表结构包含页号、存在标志、内存块号、修改标志和磁盘位置等关键信息，完整记录了每个页面的状态。系统通过可视化方式展示内存块的使用情况和页面置换过程，帮助用户理解虚拟内存的工作原理。

系统功能与操作流程

本系统提供了两个主要功能模块：动态分区管理和动态分页管理，用户可以通过顶部的标签页进行切换。

在动态分区管理界面中，用户首先需要在控制面板输入进程ID和所需内存大小，然后选择分配算法（最先适应、最佳适应或最坏适应）。点击"分配内存"按钮后，系统会尝试为该进程分配内存，并在右侧的内存状态区域显示分配结果，同时在下方的可视化区域用不同颜色的矩形表示内存块的使用情况。如果需要释放某个进程占用的内存，只需输入该进程的ID，然后点击"释放内存"按钮即可。系统会自动将释放的内存与相邻的空闲区合并，并更新显示。所有操作都会记录在操作日志区域，方便用户追踪内存变化的历史。

例如，用户可以依次创建三个进程：进程1（200单位）、进程2（300单位）和进程3（150单位）。假设使用最先适应算法，系统会按照进程创建的顺序依次分配内存。如果此时释放进程1，然后尝试创建进程4（100单位），系统会将进程4放置在之前进程1释放的位置。如果切换到最佳适应算法，系统会选择最适合的空闲区进行分配，避免内存碎片。

在动态分页管理界面中，用户首先需要创建一个作业，指定作业ID、大小和分配的内存块数。例如，创建一个大小为16KB的作业，分配4个内存块。创建成功后，用户可以在指令操作区执行各种内存访问指令，包括加法、减法、乘法、除法、存储和加载操作。每条指令需要指定操作类型、页号和页内偏移。点击"执行指令"按钮后，系统会计算物理地址，并在访问日志区域显示结果。如果访问的页面不在内存中，系统会触发缺页中断，将所需页面调入内存，必要时会置换出其他页面。

按照作业要求中的示例，用户可以创建一个7页的作业，分配4个内存块，初始页表状态如题目所示。然后依次执行12条指令：(1) + 0 072、(2) / 1 050、(3) × 2 015、(4) 存 3 026、(5) 取 0 056、(6) - 6 040、(7) + 4 056、(8) - 5 023、(9) 存 1 037、(10) + 2 078、(11) - 4 001、(12) 存 6 086。系统会显示每条指令的物理地址和缺页情况，与题目中的预期结果一致。例如，第一条指令访问页0的072偏移，物理地址为5192，不发生缺页；而第六条指令访问页6的040偏移，会发生缺页中断，需要淘汰第0页。