**动态分区存储管理和动态分页存储管理的模拟实现**

* 1. 设计目的

内存是计算机系统的核心之一，合理的内存管理策略不仅直接影响内存的利用率，而且对提升系统性能有着重要的作用。通过对内存的动态分区管理过程和动态分页管理过程的模拟实现，掌握动态分区管理中内存分配、回收策略及动态分页管理中请求式分页管理方法，并进一步理解分区管理、分页管理的特点和虚拟内存实现方式。

* 1. 设计内容

1. 模拟实现动态分区管理中内存的分配和回收过程。对于资源请求表中的要求内存长度，从可用分区表中寻找出合适的空闲区进行分配；分配空闲区之后，更新可用分区表；进程或作业释放内存资源时，应和相邻的空闲区进行链接合并，同时更新可用分区表。要求分别模拟最先适应、最佳适应和最坏适应分配算法。资源请求表、可用分区表等数据结构的定义可参考教材第五章中的结构定义也可自行设计，要求模拟出不同分配算法的分配过程和回收过程并显示（资源请求情况，能否分配，分配前后可用分区表的变化，资源回收请求，回收前后可用分区表的变化等）。
2. 模拟实现动态分页管理中请求式分页管理的过程。如果访问的页面在内存，计算出相应的物理地址，如果访问的页面不在内存，产生缺页中断，将所缺页从外存调入，如果内存没有空间需要将内存的一页淘汰，再将所缺页调入，然后计算出相应的物理地址。发生缺页中断时，查找装入新页面的内存块的处理方式采用局部置换算法，即每个作业分得一定的内存块，只能在分得的内存块内查找空闲块（例如一个4页的作业分得2个内存块，只能在这2个内存块中查找空闲块），若无空闲内存块，则从该作业中选择一个页面淘汰出内存，页面淘汰算法采用先进先出算法。页表结构除页号和该页对应的内存块号外，至少还要包括存在标志（该页是否在内存）、磁盘位置（该页的副本在磁盘上的位置）和修改标志（该页是否修改过）。假定内存64KB，每个内存块1024字节，即物理地址中块号6位、块内地址10位；作业最大64KB，即逻辑地址中页号6位、页内地址10位。要求模拟出请求式分页管理过程并显示（作业执行指令，物理地址，是否产生缺页中断，产生缺页中断时页面置换情况，页表的状态变化等）。例如，某作业的页表如下（系统分配给该作业4个内存块）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **页号** | **标志** | **内存块号** | **修改标志** | **在磁盘上的位置** |
| 0 | 1 | 5 | 0 | 010 |
| 1 | 1 | 8 | 0 | 012 |
| 2 | 1 | 9 | 0 | 013 |
| 3 | 1 | 1 | 0 | 021 |
| 4 | 0 |  | 0 | 022 |
| 5 | 0 |  | 0 | 023 |
| 6 | 0 |  | 0 | 125 |

作业执行的指令序列如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **操作** | **页号** | **页内地址** | **序号** | **操作** | **页号** | **页内地址** |
| （1） | + | 0 | 072 | （7） | + | 4 | 056 |
| （2） | / | 1 | 050 | （8） | — | 5 | 023 |
| （3） | ╳ | 2 | 015 | （9） | 存(save) | 1 | 037 |
| （4） | 存(save) | 3 | 026 | （10） | + | 2 | 078 |
| （5） | 取(load) | 0 | 056 | （11） | — | 4 | 001 |
| （6） | — | 6 | 040 | （12） | 存(save) | 6 | 086 |

则作业执行时每条指令的物理地址和是否发生缺页中断的情况如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **物理地址** | **缺页情况** | **序号** | **物理地址** | **缺页情况** |
| （1） | 5192 | 不缺页 | （7） | 8248 | 缺页，淘汰第1页 |
| （2） | 8242 | 不缺页 | （8） | 9239 | 缺页，淘汰第2页 |
| （3） | 9231 | 不缺页 | （9） | 1061 | 缺页，淘汰第3页 |
| （4） | 1050 | 不缺页 | （10） | 5198 | 缺页，淘汰第6页 |
| （5） | 5176 | 不缺页 | （11） | 8193 | 不缺页 |
| （6） | 5160 | 缺页，淘汰第0页 | （12） | 8278 | 缺页，淘汰第4页 |

1. 模拟实现过程要求使用可视化图形界面进行操作和显示（鼓励采用动画等实时动态方式显示执行过程）。

**项目优点**

**1. 功能丰富**

* **多种分配算法**：支持最先适应、最佳适应、最坏适应三种常见的内存分配算法，用户可以根据需要选择不同的算法进行内存分配。
* **可视化界面**：通过 Tkinter 库创建了直观的图形用户界面，用户可以清晰地观察到内存的使用情况，包括已分配和空闲的内存块。
* **操作日志记录**：记录所有的内存分配和释放操作，方便用户查看和分析系统的运行历史。
* **撤销和重做功能**：支持撤销和重做操作，用户可以方便地纠正误操作。
* **内存大小设置**：用户可以动态设置总内存大小，使程序能够适应不同的运行环境。

**2. 代码结构清晰**

* **模块化设计**：采用模块化的设计思想，将不同的功能封装在不同的类和方法中，如内存管理模块、界面显示模块和事件处理模块，提高了代码的可维护性和可扩展性。
* **注释详细**：代码中包含了详细的注释，便于理解和调试。

**3. 用户体验好**

* **快捷键支持**：支持使用快捷键进行操作，提高了用户的操作效率。
* **提示信息**：在操作过程中，会给出相应的提示信息，如分配失败、释放失败等，方便用户了解操作结果。

**项目缺点**

**1. 性能问题**

* **排序开销**：在释放内存时，需要对空闲分区列表进行排序，排序操作的时间复杂度为 *O*(*n****l****o****g****n*)，当空闲分区数量较多时，会影响系统的性能。
* **查找开销**：在进行内存分配时，需要遍历空闲分区列表来查找合适的分区，时间复杂度为 *O*(*n*)，可能会导致分配操作的效率较低。

**2. 可扩展性有限**

虽然采用了模块化设计，但在某些方面的可扩展性仍然有限，如添加新的内存分配算法时，需要对代码进行一定的修改，这里可以考虑使用策略模式，降低耦合性，方便扩展。