|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 算法名称 | 基本思想 | 实现过程 | 优点 | 缺点 |
| OPT（MIN） 最佳置换 | 选择**最长时间内不被访问**的页面淘汰，这样可以将淘汰它的代价降到最低（淘汰其他页面代价必定比它高） | 根据已知的**页面访问顺序**（实际上并不可知），淘汰时计算已在内存中的页面**下次访问时间**，选择时间最长的页面淘汰 | 缺页次数最少，没有Belady异常 | 不可实现，因为无法得知将来的页面访问顺序 |
| FIFO先进先出页面置换 | 优先淘汰**最早进入内存**的页面，即在内存中驻留时间最久的页面 | 只需把调入内存的页面根据**先后次序链接成队列**，设置一个指针总指向最早的页面（队首），淘汰时选择队首页面淘汰 | 实现简单 | 与进程实际运行时的规律不适应，且会可能会发生**Belady异常** |
| LRU 最近最久未使用置换 | 选择**最近最长时间未访问过**的页面予以淘汰，即以**过去预测将来**，实现OPT 的可实现的近似效果 | * 时间戳法：每个在内存中的页面记录它**最近一次被访问的时间戳**，淘汰时选择**时间戳最小**的淘汰 * 页码栈法：栈中存放所有在内存中的页面，当某页面被访问时，从**原栈中移动到栈顶**（可用双向链表实现），淘汰时选择**栈底页面**淘汰 | 缺页率低，没有Belady异常 | 每次访问页面都要更新时间戳或栈，内存管理**开销很大** |
| CLOCK  算法 | 试图用比较小的开销接近LRU算法的性能，有多种变体，也称为最近未用（Not Recently Used, NRU）算法 | 简单版：  页连成**循环链表**，每页设置一位**访问位（u或者A）**，当首次装入内存或被访问时，置为1。淘汰时从当前指针遍历循环链表，**遇1置0，遇0淘汰该页**，将指针指向下一个页面 | 开销小，且性能接近LRU算法，改进版减少**换出已修改页面**的I/O次数（修改要写回磁盘） | 缺页很少时退化为FIFO |
| 改进型：  简单版上再增加一个**修改位**，每页状态如下（淘汰顺序）：   1. 最近未被访问，也未被修改（u=0，m=0） 2. 最近未被访问，但被修改（u=0，m=1） 3. 最近被访问，但未被修改（u=1，m=0） 4. 最近被访问，被修改（u=1，m=1）   淘汰过程：   1. 寻找1类页面，有则淘汰，无进入第二步 2. 寻找2类页面淘汰，对于**3，4类页面的u置0**（变为1，2类），若循环一圈后仍无2类则**返回第1步**（此时被修改后页面只可能属于1，2类，之后可能会被访问或修改变成3，4类） |