山东大学 软件 学院

**操作系统** 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202000300125 | 姓名：贾星宇 | | 班级：2020级5班 |
| 实验编号：实验7 | | | |
| 实验题目：内存页面置换算法实验 | | | |
| 实验学时：4 | | 实验日期：2022年5月2日 | |
| 实验目的：  加深对于存储管理的了解，掌握虚拟存储器的实现原理；观察和了解重要的页面 置换算法和置换过程。练习模拟算法的编程技巧，锻炼分析试验数据的能力。 | | | |
| 硬件环境：  宿主机：lntel(R) Core(TM) i5-10210U CPU @ 1.60G Hz 2 .1 1 G Hz  虚拟机：ubuntu-20.04.1-desktop-amd64 | | | |
| 软件环境：  宿主机：Windows 10 家庭中文版  虚拟机：Code::Blocks | | | |
| 实验步骤与内容：  【编写运行实例实验程序】  编写示例程序：  Vmrp.h    Vmrp.cc            运行1：  g++ -g -c vmrp.cc vmrp.h  g++ vmrp.o -o vmrp  输入示例1  $ ./vmpr  Please input reference page numbers :12  Please input reference page string :1 2 3 4 1 2 5 1 2 3 4 5  Please input page frames :3  FIFO  1  1 2  1 2 3  4 2 3 ->1  4 1 3 ->2  4 1 2 ->3  5 1 2 ->4  5 1 2  5 1 2  5 3 2 ->1  5 3 4 ->2  5 3 4  Eliminate page:1 2 3 4 1 2  Number of page faults = 9  Rate of page faults = 75%  LRU  1  2 1  3 2 1  4 3 2 ->1  1 4 3 ->2  2 1 4 ->3  5 2 1 ->4  1 5 2  2 1 5  3 2 1 ->5  4 3 2 ->1  5 4 3 ->2  Eliminate page:1 2 3 4 5 1 2  Number of page faults = 10  Rate of page faults = 83.3%    输入示例2：  $ ./vmpr  Please input reference page numbers :12  Please input reference page string :1 2 3 4 1 2 5 1 2 3 4 5  Please input page frames :4  FIFO  1  1 2  1 2 3  1 2 3 4  1 2 3 4  1 2 3 4  5 2 3 4 ->1  5 1 3 4 ->2  5 1 2 4 ->3  5 1 2 3 ->4  4 1 2 3 ->5  4 5 2 3 ->1  Eliminate page:1 2 3 4 5 1  Number of page faults = 10  Rate of page faults = 83.3%  LRU  1  2 1  3 2 1  4 3 2 1  1 4 3 2  2 1 4 3  5 2 1 4 ->3  1 5 2 4  2 1 5 4  3 2 1 5 ->4  4 3 2 1 ->5  5 4 3 2 ->1  Eliminate page:3 4 5 1  Number of page faults = 8  Rate of page faults = 66.7%    从以上输出中可以看出FIFO置换算法的Belady异常现象，即当在相同的引用串下内存页帧数从3帧增加到4帧，页出错率反而从75％增加到了83.3%。而在相同的情况下LUR置换算法无此异常现象。  【分析独立实验要求】  请在以上示例实验程序中补充“增强二次机会”等置换算法的模拟程序。输入不同的内存页面引用串和实存帧数，观察并分析其页面置换效果和性能，并将其与 LRU 和 FIFO 算法进行比较。  改进以上示例实验程序，使之能够随机的产生内存页面引用串，以便能动态的观测各种置换算法的性能。  【编写独立实验代码】  1.Clock（）时钟二次机会置换算法      测试：    2.增强二次机会置换算法        结果：    3.LFU：最不经常使用置换算法    代码：    结果：    MFU 最常使用置换法： | | | |
| 结论分析与体会：  【分析实例实验程序思路】  vmrp.h文件中构造了一个类，用于实现内存页置换算法  在vmrp.c文件中实现如下方法：  1.构造函数：  用户输入要引用的页号大小、页号序列、内存实际页数来构造相应的数组  2.初始化函数  打印出方法名称、让缺页数为0、将引用页和淘汰页设置为-1  3.报告函数：  打印出淘汰页的顺序、缺页数目和缺页率。  Setw：当后面紧跟着的输出字段长度小于 n 的时候，在该字段前面用空格补齐，当输出字段长度大于 n 时，全部整体输出。  Setprecision：   |  |  |  | | --- | --- | --- | | **数 字** | **操作符** | **显示的值** | | 28.92786 | setprecision(3) | 28.9 | | 21.40 | setprecision(5) | 21.4 | | 109.50 | setprecision(4) | 109.5 | | 34.78596 | setprecision(2) | 35 |   4.LRU算法：  首先初始化  随后开始引用页个数次循环：  让next为第k次循环数值，随后进行实际页数次循环。如果遇到了实际页中与next相同的数字（表示要引用的页就在帧中），则让next赋为此值，同时将此值利用循环移到数组最前面并跳出循环，表示最近用了这个页  随后判断数组的顶部是否为next，如果是表明此页在帧中已经存在而且放到了数组顶部，因此打印出实际页中的页号并继续下一次引用页寻找；否则的话表示需要引用的页没有在实际帧的数组中，则让缺页数加1记下后将实际帧数组中的最后一项移入淘汰页数组中，随后实际帧数组整体后移并且将新引用的页next赋到实际帧数组的最顶端  随后同样打印当前帧中页的顺序同时报告淘汰了哪一个页。  最后用Report（）函数报告整体情况  5.FIFO算法  此算法一开始和LRU算法一致，在实际帧中寻找引用页，如果找到（i<FrameNumber）则打印并继续下一次循环找引用页（不用将找到的页加到数组最顶端）  随后如果没有找到，则将缺页数加一后将数组的第（1，2，3，……，实际帧大小，如此循环往复）个数据（表示为最早加入的页号）加入到淘汰页后替换成引用页，随后循环打印现在实际帧中的页号以及淘汰掉的页号  最后用Report（）函数报告整体情况  【分析独立实验思路】  6.时钟二次机会置换算法  核心思想：利用数组quote表示是否最近被引用过，利用quotecount作为指针，然后根据算法内容找到最近没有被引用的实际帧并替换之  7.增强二次机会算法  相似于时钟，不同点在于需要新增一个数组changed表示是否被改变，如果被改变则提高优先级，优先找到没有改变且最近没有被使用的>改变了但最近使用了的>没有改变但最近使用了的>改变了且最近使用了的  8.最不经常使用页置换算法  利用数组记录每个页被引用次数，每次引用++，找替换页的时候找引用次数最少的  9. 最常使用页置换算法  相似于上面，但要把最少的改成最大的 | | | |