|  |
| --- |
| 北 京 邮 电 大 学  实 验 报 告  课程名称： 操作系统原理  院系：计算机学院（国家示范性软件工程学院）  班级： 2021211318  姓名： 李伟泽  学号： 2021211462  教师： 赵方  成绩：  2023年 春季学期 |
| 实验目的 1. 了解虚拟存储技术的特点；  2. 掌握请求页式管理的页面置换算法。 实验环境 VMware Workstation Pro  Ubuntu22.10  Vi+GCC+GDB   1. **实验任务及内容**   1. 通过随机数产生一个指令序列，共 320 条指令。其地址按下述原则生成：  （1）50%的指令是顺序执行的；  （2）50%的指令是均匀分布在前地址部分；  （3）50%的指令是均匀分布在后地址部分；  具体的实施方法是：  A. 在[0，319]的指令地址之间随机选取一起点 M；  B. 顺序执行一条指令，即执行地址为 M+1 的指令；  C. 在前地址[0，M+1]中随机选取一条指令并执行，该指令的地址为 M’；  D. 顺序执行一条指令，其地址为 M’+1；  E. 在后地址[M’+2，319]中随机选取一条指令并执行；  F. 重复 A—E，直到执行 320 次指令。  2. 指令序列变换成页地址流  设：  （1）页面大小为 1K；  （2）用户内存容量为 4 页到 32 页；  （3）用户虚存容量为 32K。  在用户虚存中，按每 1K 存放 10 条指令排列虚存地址，即 320 条指令在虚存中的存放方式为：  第 0 条—第 9 条指令为第 0 页（对应虚存地址为[0，9]）；  第 10 条—第 19 条指令为第 1 页（对应虚存地址为[10，19]）；  ……………………  第 310 条—第 319 条指令为第 31 页（对应虚存地址为[310，319]）；  按以上方式，用户指令可组成 32 页。  3. 计算并输出下述各种算法在不同内存容量下的命中率。  A. 先进先出（FIFO）页面置换算法  B. 最近最久未使用（LRU）页面置换算法--最近最少使用算法  C. 最少使用（LFR）页面置换算法  D. 最佳（Optimal）页面置换算法  **设计思路：**  **所有算法都先检查页面是否在内存中，，如果页面不在内存中，则请求调页。检查内存是否存满，如果未满，则直接跳过，即将页面调入内存。如果满了，则使用相应算法替换掉一个页面。**  **FIFO：遍历所有的页结构体，找到存在内存中(即bool变量in为1)的页面，且存入内存time最小的页面，找到对应的帧，将其替换出去。**    **LRU：最近最久未使用算法。遍历存入内存的页面，指令线向前查找，设置变量t记录页面自上次被访问的时间，查找t最小的页面进行替换，即最近最久未使用的页面予以淘汰。**    **LFR：最近最少使用算法。使用page结构体的中的visited\_num记录被访问次数，查找所有页面中的被访问次数最少的页面进行替换。**    **OPT：向后查找，淘汰以后不再访问的页，或距最长时间后要访问的页面。**    **代码：**  *#include<iostream>*  *#include<stdlib.h>*  *#include<iomanip>*  *#include<queue>*  *#include<string>*  *#include<time.h>*  *using namespace std;*  *#define max 0x3f3f3f3f*    *int \_size;*  *double ratio;*    *struct Order*  *{*  *int id; //地址*  *int numpage; //属于页面号*  *};*  *Order order[320];*    *struct Page*  *{*  *int num;*  *bool in; //是否在内存中*  *int time; //时间*  *int visited\_num=0;//被访问次数*  *};*  *Page page[32];*    *void init0()*  *{*  *int first;*  *srand((unsigned)time(NULL)); //随机种子*  *for(int i=0;i<320;i+=6)*  *{*  *first=(int)(319.0\*rand()/RAND\_MAX)+1;*  *order[i].id=first; //任选一指令节点*  *order[i+1].id=order[i].id+1; //顺序访问下一条指令*  *order[i+2].id=(int)(1.0\*order[i].id\*rand()/RAND\_MAX); //前地址指令*  *order[i+3].id=order[i+2].id+1; //顺序访问下一条指令*  *order[i+4].id=(int)(1.0\*rand()\*(318-order[i+2].id)/RAND\_MAX)+order[i+2].id+2; //后地址指令*  *order[i+5].id=order[i+4].id+1; //顺序访问下一条指令*  *}*  *}*    *void init()*  *{*  *for(int i=0;i<320;i++)*  *{*  *order[i].numpage=order[i].id/10;*  *// cout<<i<<" "<<order[i].id<<" page-> "<<order[i].numpage<<endl;*  *}*  *for(int i=0;i<32;i++)*  *{*  *page[i].in=0;*  *page[i].num=i;*  *page[i].time=max-1;*  *}*  *}*    *void Fifo()*  *{*    *int clock=1; //记录时间*  *double totalnum=0; //指令不在内存的次数*  *for(int i=0;i<320;i++)*  *{*  *int numin=0;*  *for(int k=0;k<32;k++)*  *{*  *if(page[k].in) numin++;*  *}*  *if(page[order[i].numpage].in==1) continue; //指令所在页在内存中*  *if(page[order[i].numpage].in==0) //指令所在页不在内存中*  *{*  *if(numin<\_size)*  *{*  *totalnum++; //缺页发生,加入内存*  *page[order[i].numpage].in=1;*  *page[order[i].numpage].time=clock;*  *clock++;*  *// cout<<"进入 "<<order[i].numpage<<endl;*  *}*  *else*  *{*  *int Max=max;*  *int kk;*  *for(int k=0;k<32;k++)*  *{*  *if(page[k].in&&page[k].time<Max)//遍历所有页，找到页存在内存中，且存入内存time最小的页面，找到对应的帧，将其替换出去*  *{*  *Max=page[k].time;*  *kk=k;*  *}*  *}*  *page[kk].in=0;*  *// cout<<"退出 "<<kk<<endl;*    *totalnum++; //缺页发生,加入内存*  *page[order[i].numpage].in=1;*  *page[order[i].numpage].time=clock;*  *clock++;*  *// cout<<"进入 "<<order[i].numpage<<endl;*  *}*  *}*  *}*  *ratio=1-(totalnum/320.0);*  *cout<<"失效次数： "<<totalnum<<endl;*  *cout<<"命中率： "<<(int)(ratio\*100)<<"%"<<endl;*    *}*    *void Lru()*  *{*  *int clock=1; //记录时间*  *double totalnum=0; //指令不在内存的次数*  *for(int i=0;i<320;i++)*  *{*  *int numin=0;*  *for(int k=0;k<32;k++)*  *{*  *if(page[k].in) numin++;//统计存入内存的页面数量*  *}*  *if(page[order[i].numpage].in==1) //指令所在页在内存中*  *{*  *page[order[i].numpage].time=clock; //更新进入时间*  *clock++;*  *}*  *if(page[order[i].numpage].in==0) //指令所在页不在内存中*  *{*  *if(numin<\_size)//帧数大于存入内存的页数*  *{*  *totalnum++; //缺页发生,加入内存*  *page[order[i].numpage].in=1;*  *page[order[i].numpage].time=clock;*  *clock++;*  *// cout<<"进入 "<<order[i].numpage<<endl;*  *}*  *else*  *{*  *//遍历存入内存的页面，在之前，向前查找*  *int t=0;//记录页面自上次被访问的时间，查找t最小的页面进行替换，即最近最久未使用页面的页面予以淘汰*      *int MaxNum=999;*  *int Max=max;*  *int kk;*  *for(int k=0;k<32;k++)*  *{*  *if(page[k].in&&page[k].time<Max)*  *{*  *for(int j=i-1;j<0;j--){*  *if(order[j].id==order[i].id){//从后向前查找,找到id相同的指令,一层循环找最近的，外层循环找最小的j*  *if(j<kk)*  *{kk=j;}*  *break;*  *}*  *}*  *}*  *}*  *page[kk].in=0;*  *// cout<<"退出 "<<kk<<endl;*    *totalnum++; //缺页发生,加入内存*  *page[order[i].numpage].in=1;*  *page[order[i].numpage].time=clock;*  *clock++;*  *// cout<<"进入 "<<order[i].numpage<<endl;*  *}*  *}*  *}*  *ratio=1-(totalnum/320.0);*  *cout<<"失效次数： "<<totalnum<<endl;*  *cout<<"命中率： "<<(int)(ratio\*100)<<"%"<<endl;*    *}*  *void Lfr()*  *{*  *int clock=1; //记录时间*  *double totalnum=0; //指令不在内存的次数*  *for(int i=0;i<320;i++)*  *{*  *int numin=0;*  *for(int k=0;k<32;k++)*  *{*  *if(page[k].in) numin++;*  *}*  *if(page[order[i].numpage].in==1) continue; //指令所在页在内存中*  *if(page[order[i].numpage].in==0) //指令所在页不在内存中*  *{*  *if(numin<\_size)*  *{*  *totalnum++; //缺页发生,加入内存*  *page[order[i].numpage].in=1;*  *page[order[i].numpage].time=clock;*  *clock++;*  *// cout<<"进入 "<<order[i].numpage<<endl;*  *}*  *else*  *{*  *//查找所有页面中的被访问次数最少的页面进行替换*  *int kk;*  *int Min=321;*  *for(int k=0;k<32;k++)*  *{*  *if(page[k].in)//遍历所有在内存中的页，找到对应的帧，将其替换出去*  *{*  *if(page[k].visited\_num<Min){*  *kk=k;*  *}*  *}*  *}*  *page[kk].in=0;*  *// cout<<"退出 "<<kk<<endl;*    *totalnum++; //缺页发生,加入内存*  *page[order[i].numpage].in=1;*  *page[order[i].numpage].time=clock;*  *clock++;*  *// cout<<"进入 "<<order[i].numpage<<endl;*  *}*  *}*  *page[order[i].numpage].visited\_num++;*  *//cout<<page[order[i].numpage].visited\_num<<endl;*  *}*  *ratio=1-(totalnum/320.0);*  *cout<<"失效次数： "<<totalnum<<endl;*  *cout<<"命中率： "<<(int)(ratio\*100)<<"%"<<endl;*    *}*    *void Opt()*  *{*  *int clock=1; //记录时间*  *double totalnum=0; //指令不在内存的次数*  *for(int i=0;i<320;i++)*  *{*  *int numin=0;*  *for(int k=0;k<32;k++)*  *{*  *if(page[k].in) numin++;*  *}*  *if(page[order[i].numpage].in==1) //指令所在页在内存中*  *{*  *page[order[i].numpage].time=clock; //更新进入时间*  *}*  *if(page[order[i].numpage].in==0) //指令所在页不在内存中*  *{*  *if(numin<\_size)*  *{*  *totalnum++; //缺页发生,加入内存*  *page[order[i].numpage].in=1;*  *// cout<<"进入 "<<order[i].numpage<<endl;*  *}*  *else*  *{*  *bool n=false;*  *int j,kk;*  *for(int k=0;k<32;k++) //以后不使用*  *{*  *if(page[k].in==1)*  *{*  *for(j=i+1;j<320;j++)*  *{*  *if(order[j].numpage==page[k].num)*  *{*  *break;*  *}*  *}*  *if(j==320)*  *{*  *kk=k;*  *n=true;*  *break;*  *}*  *}*  *}*  *if(n)*  *{*  *page[kk].in=0;*  *// cout<<"退出 "<<kk<<endl;*  *totalnum++; //缺页发生,加入内存*  *page[order[i].numpage].in=1;*  *// cout<<"进入 "<<order[i].numpage<<endl;*  *continue;*  *}*  *for(int k=0;k<32;k++) //最长时间以后才会使用*  *{*  *if(page[k].in==1)*  *{*  *for(j=i+1;j<320;j++)*  *{*  *if(order[j].numpage==page[k].num)*  *{*  *if(page[k].time<=j)*  *{*  *page[k].time=j;*  *}*  *}*  *}*    *}*  *}*  *int min=0;*  *for(int k=0;k<32;k++)*  *{*  *if(page[k].in==1&&page[k].time>min)*  *{*  *kk=k;*  *min=page[k].time;*  *}*  *}*  *page[kk].in=0;*  *// cout<<"退出 "<<kk<<endl;*  *totalnum++; //缺页发生,加入内存*  *page[order[i].numpage].in=1;*  *// cout<<"进入 "<<order[i].numpage<<endl;*  *}*  *}*  *}*  *ratio=1-(totalnum/320.0);*  *cout<<"失效次数： "<<totalnum<<endl;*  *cout<<"命中率： "<<(int)(ratio\*100)<<"%"<<endl;*    *}*    *int main(int argc,char\* argv[])*  *{*  *init0();*  *while(1)*  *{*  *cout<<" 请输入内存容量（4-32）：";*  *cin>>\_size;*  *if(\_size>=4&&\_size<=32) break;*  *else cout<<"请正确输入内存容量！"<<endl;*  *}*    *cout<<endl;*  *init();*  *cout<<"先进先出（FIFO）算法： "<<endl;*  *Fifo();*    *cout<<endl;*  *init();*  *cout<<"最近最久未使用（LRU）算法： "<<endl;*  *Lru();*    *cout<<endl;*  *init();*  *cout<<"最不经常使用（LFR)算法： "<<endl;*  *Lfr();*  *cout<<endl;*  *init();*  *cout<<"最佳（Optimal）页面置换算法： "<<endl;*  *Opt();*  *return 0;*  *}*  **运行结果：**   实验心得及体会  * 1. 理解了虚拟存储技术的特点；   2. 掌握了请求页式管理的页面置换算法；   3. 加深了对FIFO、LRU、LFR以及OPT算法的理解。 |