|  |
| --- |
| 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：唐子潇  学号：2021211460  教师： 赵方 金昕  成绩：  2023年春季学期 |
| 一、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。 二、实验环境 Windows操作系统下安装的Clion、MinGw。  **三、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。  #include <iostream> #include <cstdlib> #include <ctime> #include <queue>  #define ins\_num 320 #define mem\_size 32 //内存大小（4~32页） using namespace std;  class ins{ public:  ins() : num(-1), pg\_num(-1) {};  int num; int pg\_num; }ins\_list[ins\_num];  class page{ public:  page() : num(-1), wt(0) {};  explicit page(int p\_num, int p\_wt) : num(p\_num), wt(p\_wt){};  int num; int wt;//权值 };  void init\_ins\_list() {  time\_t t;  srand((unsigned)time(&t));  int cur\_ins=rand()%ins\_num;//当前指令  ins\_list[0].num=cur\_ins; ins\_list[0].pg\_num=cur\_ins/10;  cur\_ins=(cur\_ins+1)%ins\_num;//顺序执行下一指令  ins\_list[1].num=cur\_ins; ins\_list[1].pg\_num=cur\_ins/10;  for(int i=2; i<ins\_num; i+=2){  int front, rear;//取指区间  if((i/2)%2){  front=0; rear=cur\_ins;  }  else{  front=cur\_ins+1; rear=319;  }  cur\_ins=front+rand()%(rear-front);  ins\_list[i].num=cur\_ins; ins\_list[i].pg\_num=cur\_ins/10;  cur\_ins=(cur\_ins+1)%ins\_num;  ins\_list[i+1].num=cur\_ins; ins\_list[i+1].pg\_num=cur\_ins/10;  } }  void show\_ins\_list() {  cout<<"指令序列所在页如下："<<endl;  for(int i=0; i<ins\_num; i++){  printf("%2d ", ins\_list[i].pg\_num);  if((i+1)%20==0) cout<<endl;  }  cout<<"==========================================================="<<endl; }  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  其中，我主要负责先进先出（FIFO）页面置换算法：  int FIFO() {  int pg\_ft=0; queue <int> pg\_tb;//存储当前位于内存的页号  for(int i=0; i<ins\_num; i++){  bool pg\_in\_mem=false;//标识位是否在内存中  int j;  for(j=(int)pg\_tb.size(); j>0; j--){  //printf("%2d ", pg\_tb.front());  if(pg\_tb.front()==ins\_list[i].pg\_num)  pg\_in\_mem= true;  pg\_tb.push(pg\_tb.front());  pg\_tb.pop();  }  //cout<<endl;  if(!pg\_in\_mem){  pg\_ft++;  if(pg\_tb.size()==mem\_size) pg\_tb.pop();  pg\_tb.push(ins\_list[i].pg\_num);  }  }  return pg\_ft; }  int main() {  system("chcp 65001");  init\_ins\_list();  printf("内存大小为：%2d页\n", mem\_size);  cout<<"Page Fault times:"<<endl;  cout<<"FIFO: "<<FIFO()<<endl;  return 0; } 四、实验心得 通过本次实验，我了解了虚拟存储技术的特点，掌握并练习了请求页式管理的页面置换算法，尤其对先进先出（FIFO）页面置换算法有了更加深刻的认识。 |