## 实验七：存储管理模拟

返回[目录](#_目录)

### 实验目的：

1. 掌握请求分页存储管理系统的基本原理
2. 实现一个模拟的页式虚拟存储管理系统

### 实验内容

编写一个程序，模拟一个页式虚拟存储管理系统。（不考虑地址转换）

其中，由系统随机产生进程；

进程大小、进程到达次序、时间、进程执行轨迹（页面访问顺序）也随机生成，但进程之间必须有并发存在，进程执行时间需有限，进程调度采用时间片轮转算法（以页面模拟）；

物理块分配策略采取固定分配局部置换；

分配算法采用按比例分配算法；

调页采用请求调页方式；

置换采用LRU算法；

驻留集大小可调，观察驻留集大小对缺页率的影响。

### 实验报告

完成实验内容并写出实验报告，报告应具有以下内容：

* 程序（含注释）
* 运行结果展示
* 对设计思路的详细说明

答：程序代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  14  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124 | #include <stdio.h>  #define M 3  struct Page {  int id; // 页号  int addr; // 地址  } p[M];  int r[20];  int cnt = 0;  void read();  void FIFO();  void LRU();  void OPT();  void pageInit();  int inPage(int v);  int main() {  read();  FIFO();  LRU();  OPT();  return 0;  }  void OPT() {  int ti = 0;  pageInit();  int c = 0;  printf("OPT Page1 Page2 Page3 state\n");  for (int i = 0; i < cnt; ++i) {  int s = inPage(r[i]);  if (c < 3) {  ++c;  for (int j = c - 1; j > 0; --j) p[j].addr = p[j - 1].addr;  p[0].addr = r[i];  ++ti;  } else if (s < 0) {  ++ti;  int t = 0;  int tt = 0;  for (int j = M - 1; j >= 0; --j) {  int tmp = 100;  for (int k = 1; k + i < cnt; ++k)  if (r[i + k] == p[j].addr) {  tmp = k;  break;  }  if (tmp >= tt) {  tt = tmp;  t = j;  }  }  for (int j = t; j > 0; --j) p[j].addr = p[j - 1].addr;  p[0].addr = r[i];  }  printf("%3d", r[i]);  for (int i = 0; i < M; ++i)  printf(" %5d", p[i].addr);  printf(" %4c\n", s >= 0 ? 'Y' : 'N');  }  printf("OPT: %d\n", ti);  }  // 最近最久未使用算法  void LRU() {  int ti = 0;  pageInit();  printf("LRU Page1 Page2 Page3 state\n");  for (int i = 0; i < cnt; ++i) {  printf("%3d", r[i]);  int s = inPage(r[i]);  for (int j = s >= 0 ? s : M - 1; j > 0; --j)  p[j].addr = p[j - 1].addr;  p[0].addr = r[i];  for (int i = 0; i < M; ++i)  printf(" %5d", p[i].addr);  printf(" %4c\n", s >= 0 ? 'Y' : 'N');  if (s < 0) ++ti;  }  printf("LRU: %d\n\n", ti);  }  // 先进先出算法  void FIFO() {  pageInit();  int ti = 0;  printf("FIFO Page1 Page2 Page3 state\n");  for (int i = 0; i < cnt; ++i) {  printf("%4d", r[i]);  if (inPage(r[i]) >= 0) {  for (int i = 0; i < M; ++i)  printf(" %5d", p[i].addr);  printf(" Y\n");  continue;  } else {  for (int j = M - 1; j > 0; --j)  p[j].addr = p[j - 1].addr;  p[0].addr = r[i];  for (int i = 0; i < M; ++i)  printf(" %5d", p[i].addr);  printf(" N\n");  ++ti;  }  }  printf("FIFO: %d\n\n", ti);  }  int inPage(int v) {  for (int i = 0; i < M; ++i)  if (p[i].addr == v) return i;  return -1;  }  void pageInit() {  for (int i = 0; i < M; ++i) {  p[i].id = i + 1;  p[i].addr = -1;  }  }  // 输入页面请求的顺序  void read() {  int t;  while (scanf("%d", &t) != EOF)  r[cnt++] = t;  } |

运行结果如图所示：

