实验七：存储管理模拟

返回目录

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：

```c
#include <stdio.h>

#define M 3
struct Page {
    int id; // 页号
    int addr;   // 地址
} p[M];
```

```
 8 int r[20];
 9 int cnt = 0;
10
11 void read();
12 void FIFO();
13 void LRU();
14 void OPT();
14 void pageInit();
16 int inPage(int v);
17
18 int main() {
19     read();
20     FIFO();
21     LRU();
22     OPT();
23     return 0;
24 }
25
26 void OPT() {
27     int ti = 0;
28     pageInit();
29     int c = 0;
30     printf("OPT Page1 Page2 Page3 state\n");
31     for (int i = 0; i < cnt; ++i) {
32         int s = inPage(r[i]);
33         if (c < 3) {
34             ++c;
35             for (int j = c - 1; j > 0; --j) p[j].addr
   = p[j - 1].addr;
36             p[0].addr = r[i];
37             ++ti;
38         } else if (s < 0) {
39             ++ti;
40             int t = 0;
41             int tt = 0;
42             for (int j = M - 1; j >= 0; --j) {
43                 int tmp = 100;
44                 for (int k = 1; k + i < cnt; ++k)
```

```
45                      if (r[i + k] == p[j].addr) {
46                          tmp = k;
47                          break;
48                      }
49                  if (tmp >= tt) {
50                      tt = tmp;
51                      t = j;
52                  }
53              }
54              for (int j = t; j > 0; --j) p[j].addr =
    p[j - 1].addr;
55              p[0].addr = r[i];
56          }
57          printf("%3d", r[i]);
58          for (int i = 0; i < M; ++i)
59              printf(" %5d", p[i].addr);
60          printf(" %4c\n", s >= 0 ? 'Y' : 'N');
61      }
62      printf("OPT: %d\n", ti);
63  }
64  // 最近最久未使用算法
65  void LRU() {
66      int ti = 0;
67      pageInit();
68      printf("LRU Page1 Page2 Page3 state\n");
69      for (int i = 0; i < cnt; ++i) {
70          printf("%3d", r[i]);
71          int s = inPage(r[i]);
72          for (int j = s >= 0 ? s : M - 1; j > 0; --j)
73              p[j].addr = p[j - 1].addr;
74          p[0].addr = r[i];
75          for (int i = 0; i < M; ++i)
76              printf(" %5d", p[i].addr);
77          printf(" %4c\n", s >= 0 ? 'Y' : 'N');
78          if (s < 0) ++ti;
79      }
80      printf("LRU: %d\n\n", ti);
81  }
```

```
82  // 先进先出算法
83  void FIFO() {
84      pageInit();
85      int ti = 0;
86      printf("FIFO Page1 Page2 Page3 state\n");
87      for (int i = 0; i < cnt; ++i) {
88          printf("%4d", r[i]);
89          if (inPage(r[i]) >= 0) {
90              for (int i = 0; i < M; ++i)
91                  printf(" %5d", p[i].addr);
92              printf("    Y\n");
93              continue;
94          } else {
95              for (int j = M - 1; j > 0; --j)
96                  p[j].addr = p[j - 1].addr;
97              p[0].addr = r[i];
98              for (int i = 0; i < M; ++i)
99                  printf(" %5d", p[i].addr);
100             printf("    N\n");
101             ++ti;
102         }
103     }
104     printf("FIFO: %d\n\n", ti);
105 }
106
107 int inPage(int v) {
108     for (int i = 0; i < M; ++i)
109         if (p[i].addr == v) return i;
110     return -1;
111 }
112
113 void pageInit() {
114     for (int i = 0; i < M; ++i) {
115         p[i].id = i + 1;
116         p[i].addr = -1;
117     }
118 }
119 // 输入页面请求的顺序
```

```
120 void read() {
121     int t;
122     while (scanf("%d", &t) != EOF)
123         r[cnt++] = t;
124 }
```

运行结果如图所示：