实 验 报 告

课程名称：操作系统试验

实 验 三：内存页面置换算法

班　　级：

学生姓名：***Steven***

学　　号：

专　　业：

指导教师：

学　　期：2021－2022 学年秋季学期

成　　绩：

云南大学信息学院

一、实验目的

1、掌握内存的分区、分页和分段管理的基本概念和原理，掌握内存的虚拟空间和物理空间的对应关系；

2、掌握内存分配中的连续和非连续分配、固定分配和动态分配等概念，掌握几种内存分配方法的分配过程和回收过程；

3、掌握内存的页面置换算法，包括先进先出（FIFO），最近最久未使用（LRU），最不经常适用（LFU），最近未使用（NRU），最佳置换（OPT）等方法，以及理解算法间的优劣差别，了解缺页率、belady 现象等内容。

二、知识要点

1、内存的虚拟地址和物理地址映射；

2、内存的分区管理、分页管理、分段管理和段页式管理；

3、页面置换算法，包括先进先出（FIFO），最近最久未使用（LRU），最不经常适用（LFU），最近未使用（NRU），最佳置换（OPT）等方法。

三、实验预习（要求做实验前完成）

1、了解 linux 系统中常用命令的使用方法；

2、掌握内存的虚拟地址和物理地址的描述；

3、掌握内存的分页管理的基本原理和过程；

4、掌握基本的内存页面置换算法，包括先进先出（FIFO），最近最久未使用（LRU）等。

四、实验内容和试验结果

结合课程所讲授内容以及课件中的试验讲解，完成以下试验。请分别描述程序的流程，附上源代码，并将试验结果截图附后。

1、模拟内存的页式管理，实现内存的分配和调用，完成虚拟内存地址序列和物理内存的对应。在内存调用出现缺页时，调入程序的内存页。在出现无空闲页面时，使用先进先出（FIFO）算法实现页面置换。

```
void FIFO(int total_pf) {
    initialize(total_pf);
    diseffect = 0;
    pfc *p;
    busypf_head = busypf_tail = NULL;
    for (int i = 0; i < TOTAL_INSTRUCTION; i++) {
```

```
        // 找到需要的页号
        if (pl[page[i]].pfn == INVALID) {// 页面不在内存
            diseffect++;
            if (freepf_head == NULL) {// 没有空闲页框，从 busypf
表中释放一个页框
                p = busypf_head->next;
                pl[busypf_head->pn].pfn = INVALID;
                freepf_head = busypf_head;
                freepf_head->next = NULL;
                busypf_head = p;
            }
            // 从 freepf 中取出第一个页框，页面换入该页框
            p = freepf_head->next;
            freepf_head->next = NULL;
            freepf_head->pn = page[i];
            pl[page[i]].pfn = freepf_head->pfn;
            // 将该页框加入 busypf 中
            if (busypf_tail == NULL) {
                busypf_head = busypf_tail = freepf_head;
            } else {
                busypf_tail->next = freepf_head;
                busypf_tail = freepf_head;
            }
            freepf_head = p;
        }
    }
    printf("FIFO:%.2f%%\n", (1 - (float) diseffect /
TOTAL_INSTRUCTION) * 100);
}
```

```
root@localhost:/C_projects/Exp3# exp3_1
Query Order|请求顺序:
7  0  1  2  0  3  0  4  2  3  0  3  2  1  2  0  1
Three Pages|3页:FIFO:29.41%
Four Pages| 4页:FIFO:47.06%
```

2、参考第一题的页式内存管理，在出现无空闲页面时，改使用最近最久未使用（LRU）算法。

```
void LRU(int total_pf) {
    int min, minj;
    int present_time = 0;
    pfc *p = NULL, *prep = NULL;
    initialize(total_pf);

    for (int i = 0; i < TOTAL_INSTRUCTION; i++) {
        if (pl[page[i]].pfn == INVALID) {
```

```
            diseffect++;
            if (freepf_head == NULL) {// 找到time最小的页
                min = 32767;
                for (int j = 0; j < TOTAL_PAGE; j++) {
                    if (min > pl[j].time && pl[j].pfn != 
INVALID) {
                        min = pl[j].time;
                        minj = j;
                    }
                }
                // 其页框进入freepf
                prep = NULL;
                p = busypf_head;
                while (p != NULL) {
                    if (p->pn == minj) {
                        break;
                    }
                    prep = p;
                    p = p->next;
                }
                // 从busy链表中取出
                if (prep == NULL) {
                    busypf_head = p->next;
                } else {
                    prep->next = p->next;
                }
                // 放入free链表
                freepf_head = p;
                freepf_head->next = NULL;
                pl[minj].pfn = INVALID;
                pl[minj].time = -1;
            }
            // 新页换入freepf中的第一个页框
            p = freepf_head;
            freepf_head = freepf_head->next;
            p->pn = page[i];
            p->next = NULL;
            if (busypf_tail == NULL) {
                busypf_head = p;
            } else {
                busypf_tail->next = p;
            }
            busypf_tail = p;
            pl[page[i]].pfn = p->pfn;
```

```
            pl[page[i]].time = present_time;
        } else
            pl[page[i]].time = present_time;
        present_time++;
    }
    printf("LRU:%.2f%%\n", (1 - (float) diseffect /
TOTAL_INSTRUCTION) * 100);
}
```

```
root@localhost:/C_projects/Exp3# exp3_2
Query Order|请求顺序:
7  0  1  2  0  3  0  4  2  3  0  3  2  1  2  0  1
Three Pages|3页:LRU:35.29%
```

3、对比前两题实现的页面置换算法，以相同的内存调用序列数据做实验，输出缺页率，尝试讨论它们的差别。

```
root@localhost:/C_projects/Exp3# exp3_3
Query Order|请求顺序:
7  0  1  2  0  3  0  4  2  3  0  3  2  1  2  0  1
Three Pages|3页:FIFO:29.41%
Three Pages|3页:LRU:35.29%
Four Pages| 4页:FIFO:47.06%
Four Pages| 4页:LRU:58.82%
```

在同样的内存调用序列和同样数量的页框数的情况下，LRU 算法的命中率比 FIFO 算法的命中率高，缺页率低。