

# 《操作系统原理》实验报告(三)

姓名	学号	专业班级	时间
----	----	------	----

## 一、实验目的

- 1) 理解页面淘汰算法原理，编写程序演示页面淘汰算法。
- 2) 验证Linux虚拟地址转化为物理地址的机制。
- 3) 理解和验证缺页处理的流程。

## 二、实验内容

- 1) Windows/Linux模拟实现FIFO或LRU页面淘汰算法。
- 2) Linux下利用/proc/pid/pagemap计算某个变量或函数虚拟地址对应的物理地址等信息。建议优麒麟或麒麟系统。
- 3) 研读并修改Linux内核的缺页处理函数do\_no\_page 或页框分配函数 get\_free\_page，并用printk打印调试信息。注意：需要编译内核。建议优麒麟或麒麟系统。

## 三、实验环境和核心代码

### 3.1 模拟实现FIFO算法

开发环境：windows 11，编辑工具：vscode，编译工具：gcc

使用C语言模拟实现FIFO页面淘汰算法。

```
1  #include<stdio.h>
2  #include<time.h>
3  #include<stdlib.h>
4  #include<stdbool.h>
5
6  #define num_of_command 320 // 指令个数
7  #define max_of_RAM 32
8
9  char RAM[max_of_RAM]; // 内存，设：最大为32个页框
10
11 int FIFO(char pages[], int pagesNum, int can_use)
12 {
13     // 先清空 can_use 个页框，不妨设前 can_use 个页框为系统分配给该程序的
14     for (int i = 0; i < can_use; i++)
15     {
16         RAM[i] = NULL;
17     }
18     // 记录缺页数
19     int count = 0;
```

```

20 // 记录本次页面应填充的位置
21 int location = 0;
22 // 开始执行指令（遍历页面流）
23 for (int i = 0; i < pageNum; i++)
24 {
25     bool flag = true; // 记录是否缺页（true:缺页, false:不缺页）
26     for (int j = 0; j < can_use; j++)
27     {
28         if (RAM[j] == pages[i])
29         {
30             flag = false;
31             break;
32         }
33     }
34     // 缺页淘汰
35     if (flag == true)
36     {
37         count++; // 缺页数 + 1
38         // 找到了需要被淘汰的页面
39         RAM[location] = pages[i];
40         location = (location + 1) % can_use;
41     }
42 }
43 printf("FIFO: 缺页次数为%3d, 缺页率为%.2f%%\n", count, (float)(100 *
((float)count / (float)pageNum));
44 return count;
45 }
46
47 int command[num_of_command];
48 char pages[num_of_command];
49 void initialize()
50 {
51     for (int i = 0; i < num_of_command; i++)
52     {
53         command[i] = rand() % 260;
54         pages[i] = command[i] / 10 + 'A';
55     }
56 }
57
58 int main()
59 {
60     int FIFO_count = 0;
61     int num_of_box = 4;
62     srand((unsigned)time(NULL));
63     while(num_of_box++ <= max_of_RAM)
64     {
65         initialize();
66         printf("Num of box: %d\t", num_of_box);
67         FIFO(pages, num_of_command, num_of_box);
68     }
69     return 0;
70 }

```

## 3.2 利用pagemap计算地址

开发环境: Ubuntu 20.04, 内核版本: 5.15.67, 编辑工具: vim & gedit

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <fcntl.h>
4  #include <stdint.h>
5  #include <string.h>
6
7  // get visual Address 获取虚拟地址
8  void get_VA(unsigned long vaddr)
9  {
10     int pid = getpid();
11     printf(">>> 当前pid: %d\n", pid);
12     printf(">>> 虚拟地址: %lx\n", vaddr);
13     printf(">>> 虚拟页号: %lx\n", vaddr / getpagesize());
14     printf(">>> 页内偏移地址: %lx\n", vaddr % getpagesize());
15
16     unsigned long v_offset = vaddr / getpagesize() * sizeof(uint64_t);
17
18     uint64_t item = 0; // 存储对应项的值
19
20     int fd = open("/proc/self/pagemap", O_RDONLY);
21     if (fd == -1) // 判断是否打开失败
22     {
23         printf("open /proc/self/pagemap error\n");
24         return;
25     }
26     // 将游标移动到相应位置, 即对应项的起始地址且判断是否移动失败
27     if (lseek(fd, v_offset, SEEK_SET) == -1)
28     {
29         printf("seek error\n");
30         return;
31     }
32     // 读取对应项的值, 并存入item中, 且判断读取数据位数是否正确
33     if (read(fd, &item, sizeof(uint64_t)) != sizeof(uint64_t))
34     {
35         printf("read item error!\n");
36         return;
37     }
38     // 判断当前物理页是否在内存中,
39     if (((uint64_t)1 << 63) & item) == 0)
40     {
41         printf("page present is 0\n");
42         return;
43     }
44
45     // 获得物理页号, 即取item的bit (0~54)
46     uint64_t phy_pageIndex = (((uint64_t)1 << 55) - 1) & item;
47     printf(">>> 物理页框号: %lx\n", phy_pageIndex);
```

```

48
49     // 获取物理地址
50     unsigned long paddr = (phy_pageIndex * getpagesize()) + vaddr %
getpagesize();
51     printf(">>> 物理地址: %lx\n", paddr);
52 }
53
54 const int a = 100; // 全局变量
55 int main()
56 {
57     int b = 100;      // 局部变量
58     static int c = 100; // 局部静态变量
59     const int d = 100; // 局部常量
60
61     printf(">>> 页面大小: %x\n", getpagesize());
62     printf("全局常量:\n");
63     get_VA((unsigned long)&a);
64
65     printf("\n局部变量:\n");
66     get_VA((unsigned long)&b);
67
68     printf("\n全局静态变量:\n");
69     get_VA((unsigned long)&c);
70
71     printf("\n局部常量:\n");
72     get_VA((unsigned long)&d);
73     // while (1);
74     return 0;
75 }

```

## 四、实验结果

### 4.1 模拟实现FIFO算法

```

C:\Users\dekr\Desktop\未命名2.exe
Num of box: 5    FIFO: 缺页次数为267, 缺页率为83.44%
Num of box: 6    FIFO: 缺页次数为244, 缺页率为76.25%
Num of box: 7    FIFO: 缺页次数为234, 缺页率为73.13%
Num of box: 8    FIFO: 缺页次数为223, 缺页率为69.69%
Num of box: 9    FIFO: 缺页次数为212, 缺页率为66.25%
Num of box: 10   FIFO: 缺页次数为219, 缺页率为68.44%
Num of box: 11   FIFO: 缺页次数为172, 缺页率为53.75%
Num of box: 12   FIFO: 缺页次数为194, 缺页率为60.63%
Num of box: 13   FIFO: 缺页次数为179, 缺页率为55.94%
Num of box: 14   FIFO: 缺页次数为157, 缺页率为49.06%
Num of box: 15   FIFO: 缺页次数为141, 缺页率为44.06%
Num of box: 16   FIFO: 缺页次数为115, 缺页率为35.94%
Num of box: 17   FIFO: 缺页次数为123, 缺页率为38.44%
Num of box: 18   FIFO: 缺页次数为109, 缺页率为34.06%
Num of box: 19   FIFO: 缺页次数为 97, 缺页率为30.31%
Num of box: 20   FIFO: 缺页次数为 79, 缺页率为24.69%
Num of box: 21   FIFO: 缺页次数为 73, 缺页率为22.81%
Num of box: 22   FIFO: 缺页次数为 60, 缺页率为18.75%
Num of box: 23   FIFO: 缺页次数为 56, 缺页率为17.50%
Num of box: 24   FIFO: 缺页次数为 42, 缺页率为13.12%
Num of box: 25   FIFO: 缺页次数为 36, 缺页率为11.25%
Num of box: 26   FIFO: 缺页次数为 26, 缺页率为8.13%
Num of box: 27   FIFO: 缺页次数为 26, 缺页率为8.13%
Num of box: 28   FIFO: 缺页次数为 26, 缺页率为8.13%
Num of box: 29   FIFO: 缺页次数为 26, 缺页率为8.13%
Num of box: 30   FIFO: 缺页次数为 26, 缺页率为8.13%
Num of box: 31   FIFO: 缺页次数为 26, 缺页率为8.13%
Num of box: 32   FIFO: 缺页次数为 26, 缺页率为8.13%
Num of box: 33   FIFO: 缺页次数为 26, 缺页率为8.13%

```

## 4.2 利用pagemap计算地址

```
dekrtd@dekrtd-Lenovo-XiaoXinPro-16ACH-2021: ~/桌面$ sudo ./os_exp_3.2
[sudo] dekrtd 的密码:
>>> 页面大小: 1000
全局常量:
>>> 当前pid: 4760
>>> 虚拟地址: 557925a7b0f8
>>> 虚拟页号: 557925a7b
>>> 页内偏移地址: f8
>>> 物理页框号: 147a7f
>>> 物理地址: 147a7f0f8

局部变量:
>>> 当前pid: 4760
>>> 虚拟地址: 7ffd37943410
>>> 虚拟页号: 7ffd37943
>>> 页内偏移地址: 410
>>> 物理页框号: 16cb59
>>> 物理地址: 16cb59410

全局静态变量:
>>> 当前pid: 4760
>>> 虚拟地址: 557925a7d010
>>> 虚拟页号: 557925a7d
>>> 页内偏移地址: 10
>>> 物理页框号: 13e1e6
>>> 物理地址: 13e1e6010

局部常量:
>>> 当前pid: 4760
>>> 虚拟地址: 7ffd37943414
>>> 虚拟页号: 7ffd37943
>>> 页内偏移地址: 414
>>> 物理页框号: 16cb59
>>> 物理地址: 16cb59414
dekrtd@dekrtd-Lenovo-XiaoXinPro-16ACH-2021: ~/桌面$
```

## 五、实验错误排查和解决方法

### 5.1 模拟实现FIFO算法

#### 5.1.1 程序报错: identifier "bool" is undefined:

问题出现的原因是 C 语言中没有 bool 类型 因此, 需要在程序中加入头文件 `#include<stdbool.h>`。

## 5.2 利用pagemap计算地址

### 5.2.1 运行程序时发现物理页框号均为0

问题出现的原因是程序的权限不够, 应该在可执行文件前加上 `sudo`。

## 六、实验参考资料和网址

- 教学课件
- 操作系统-内存管理实验指导书: <https://wenku.baidu.com/view/ffd39261011ca300a6c390ee.html?e=view& wkts =1682560478932>
- 用C++实现LRU算法: <https://blog.csdn.net/z702143700/article/details/48374201>
- 用C++实现FIFO算法: [https://blog.csdn.net/weixin\\_44384477/article/details/109538424](https://blog.csdn.net/weixin_44384477/article/details/109538424)