# 《操作系统》实验报告

| | | | |
|---|---|---|---|
| 年级、专业、班级 | | 姓名 | |
| 实验题目 | 内存管理 | | |
| 实验时间 | | 实验地点 | |
| 实验成绩 | | 实验性质 | □验证性 □设计性 ■综合性 |

教师评价：

□算法/实验过程正确；□源程序/实验内容提交 □程序结构/实验步骤合理；

□实验结果正确； □语法、语义正确； □报告规范；

其他：

评价教师签名：

## 一、实验目的

理解操作系统关于内存管理的一些方法。

熟悉常用的页面置换策略的基本原理。

通过模拟实验分析不同置换策略的性能差异。

## 二、实验项目内容

在 linux 环境下用 C 语言编写程序，模拟进程在执行时内存中的页框置换过程。

读取文件中给定进程访问的逻辑页号序列，其中单号学号同学做 workload1~6，双号学号同学做 workload7~12。

设置内存页框大小为 N（N 分别取值为 100，500，1000，2000，5000）。

采用 3 种不同的页面置换算法：FIFO,CLOCK,LRU。

画图比较不同页面置换算法对应的缺页率并分析原因(固定页框大小为 1000)。

画图比较不同内存页框大小对应的缺页率并分析原因(固定置换算法为LRU)。

分析不同 workload 平均缺页率存在差异产生的原因。

三、实验过程或算法（源程序）

详细代码见代码附件，以下为程序基本思路

:

采用数组 int page[N]模拟虚拟页框，读取文件中的虚拟地址做整除 N 即可得出虚拟页框号，若命中则 hitTime++，否则根据先后顺序分别放入虚拟页框中，此时 missTime++；当虚拟页框满后需要进行替换操作，替换 int page[N]中最靠前的一项(即先进入的页框)，替换项用模拟的指针 ptr 代表

:

方式同，不同点在于虚拟页框满后将最先进入的或者最近未使用的一项换出，方法体现在若命中过，则将被命中的页框排到 int page[N]的末尾，其他项依次前移，下次的被替换项依旧是数组中的第一号元素

:

用一个结构体定义了带使用位的元素，当虚拟页框为空时依次填入元素并将使用位置 1，当命中后也将使用位置 1，用一个模拟指针 ptr 代表当前指针指向的位置，当需要替换页框时从指针指向位置向后查找使用位为 0 的第一个元素，在此过程中遇到使用位为 1 的元素将使用位置 0，替换后指针继续指向当前元素位置不变

## 四、实验结果及分析和（或）源程序调试过程

代码：

:

```
#include<>

um==temp)

                    {

                        page[i].useFlag=1;

                        ptr++;

                        hitTime++;

                        flag=1;

                        break;

                    }

                }

                if(flag==1)um=temp;

                    page[count%N].useFlag=1;

                    ptr++;

                }

                else

                {

                    while(page[ptr%N].useFlag!=0)

                    {

                        page[ptr%N].useFlag=0;

                        ptr++;

                    }

                    page[ptr%N].num=temp;

                }
```

```
                }
                count++;
            }
            fclose(fp);
            float rate;
            rate=missTime/(float)count;
            printf("%s:\n",fileName[fileNum]);
            printf("Page fault time:%d\n",missTime);
            printf("Page hit time:%d\n",hitTime);
            printf("Total:%d\n",count);
            printf("Rate of fault = %f\n",rate);
        }
        return 0;
}
:
#include<>
//FIFO
#define N 1000

int page[N]={0};

int main()
{
        char
fileName[][15]={"workload7","workload8","workload9","workload10","workload1
1","workload12"};
        int fileNum;
        for(fileNum=0;fileNum<6;fileNum++)
```

```
{
 FILE* fp;
 int missTime=0;
 int hitTime=0;
 int count=0;
 int ptr=0;
 if((fp=fopen(fileName[fileNum],"r"))==NULL)
 {
     printf("The file can not be oppen\n");
     return -1;
 }
 while(!feof(fp))
 {
     int temp;
     int flag=0;
     fscanf(fp,"%d",&temp);
     temp/=N;
     int i;
     for(i=0;i<N;i++)
     {
         if(page[i]==temp)
         {
             hitTime++;
             flag=1;
             break;
         }
     }
```

```
            if(flag==1)//hit
            {
                count++;
                continue;
            }
            else//miss
            {
                missTime++;
                if(missTime<N)
                    page[count%N]=temp;
                else
                {
                    page[ptr%N]=temp;
                    ptr++;
                }
                count++;
            }
        }
        fclose(fp);
        float rate;
        rate=missTime/(float)count;
        printf("%s:\n",fileName[fileNum]);
        printf("Page fault time:%d\n",missTime);
        printf("Page hit time:%d\n",hitTime);
        printf("Total:%d\n",count);
        printf("Rate of fault = %f\n",rate);
    }
```

```
            return 0;
}
:
#include<>
//LRU
#define N 1000

int page[N]={0};

int main()
{
            char
fileName[][15]={"workload7","workload8","workload9","workload10","workload1
1","workload12"};
            int fileNum;
            for(fileNum=0;fileNum<6;fileNum++)
            {
             FILE* fp;
             int missTime=0;
             int hitTime=0;
             int count=0;
             int ptr=0;
             if((fp=fopen(fileName[fileNum],"r"))==NULL)
             {
                 printf("The file can not be oppen\n");
                 return -1;
             }
             while(!feof(fp))
```

```
        {
            int temp;
            int flag=0;
            fscanf(fp,"%d",&temp);
            temp/=N;
            int i;
            for(i=0;i<N;i++)
            {
                if(page[i]==temp)
                {
                    flag=1;
                    hitTime++;
                    int j;
                    int c=page[i];
                    for(j=i;j<N-1;j++)
                    {
                        page[j]=page[j+1];
                    }
                    page[N-1]=c;
                    break;
                }
            }
            if(flag==1)//hit
            {
                count++;
                continue;
            }
```

```
                else//miss
                {
                    missTime++;
                    if(missTime<N)
                        page[count%N]=temp;
                    else
                    {
                        int j;
                        for(j=0;j<N-1;j++)
                        {
                            page[j]=page[j+1];
                        }
                        page[N-1]=temp;
                    }
                        }
                        count++;
                    }
                }
                fclose(fp);
                float rate;
                rate=missTime/(float)count;
                printf("%s:\n",fileName[fileNum]);
                printf("Page fault time:%d\n",missTime);
                printf("Page hit time:%d\n",hitTime);
                printf("Total:%d\n",count);
                printf("Rate of fault = %f\n",rate);
            }
            return 0;
```

}

结果截图：

三个程序运行结果：

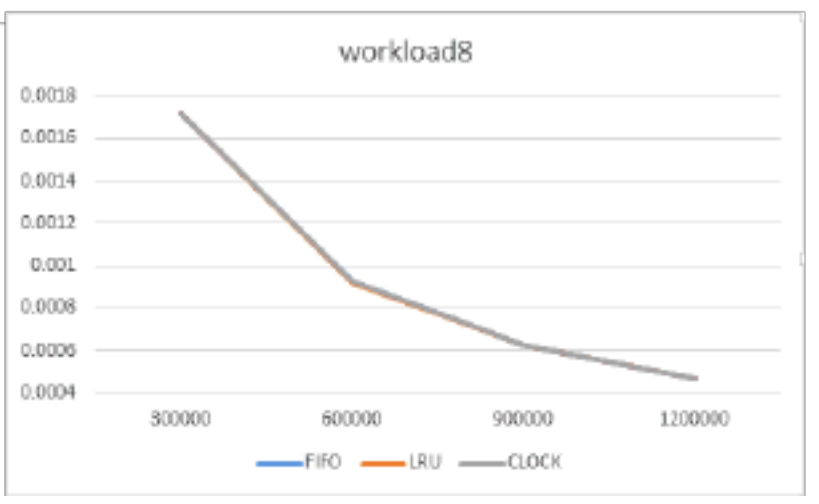
```
user1@ubuntu:~$ ./FIFO
workload7:
Page fault time:191
Page hit time:153671
Total:153862
Rate of fault = 0.001241
workload8:
Page fault time:515
Page hit time:1336790
Total:1337305
Rate of fault = 0.000385
workload9:
Page fault time:1516
Page hit time:320737
Total:322253
Rate of fault = 0.004704
workload10:
Page fault time:548
Page hit time:346727
Total:347275
Rate of fault = 0.001578
workload11:
Page fault time:1289
Page hit time:201827
Total:203116
Rate of fault = 0.006346
workload12:
Page fault time:119
Page hit time:202946
Total:203065
Rate of fault = 0.000586
user1@ubuntu:~$
```

```
user1@ubuntu:~$ ./LRU
workload7:
Page fault time:195
Page hit time:153667
Total:153862
Rate of fault = 0.001267
workload8:
Page fault time:516
Page hit time:1336789
Total:1337305
Rate of fault = 0.000386
workload9:
Page fault time:1327
Page hit time:320926
Total:322253
Rate of fault = 0.004118
workload10:
Page fault time:547
Page hit time:346728
Total:347275
Rate of fault = 0.001575
workload11:
Page fault time:1268
Page hit time:201848
Total:203116
Rate of fault = 0.006243
workload12:
Page fault time:118
Page hit time:202947
Total:203065
Rate of fault = 0.000581
user1@ubuntu:~$
```

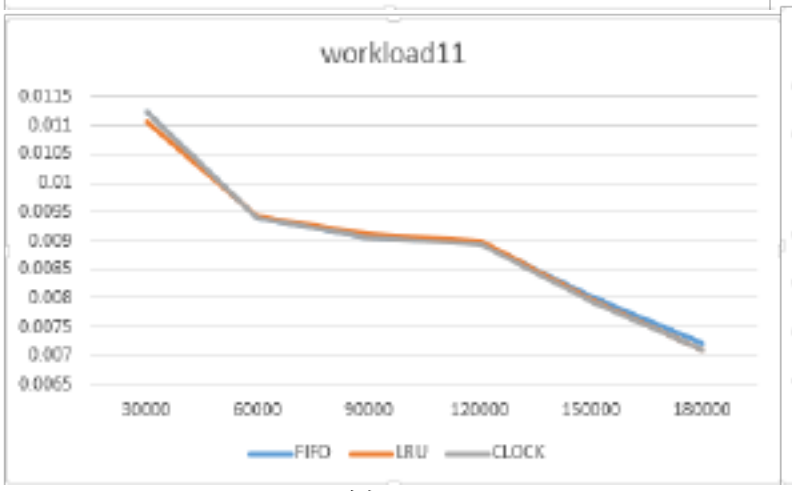
```
user1@ubuntu:~$ ./CLOCK
workload7:
Page fault time:191
Page hit time:153671
Total:153862
Rate of fault = 0.001241
workload8:
Page fault time:515
Page hit time:1336790
Total:1337305
Rate of fault = 0.000385
workload9:
Page fault time:1366
Page hit time:320887
Total:322253
Rate of fault = 0.004239
workload10:
Page fault time:530
Page hit time:346745
Total:347275
Rate of fault = 0.001526
workload11:
Page fault time:1273
Page hit time:201843
Total:203116
Rate of fault = 0.006267
workload12:
Page fault time:118
Page hit time:202947
Total:203065
Rate of fault = 0.000581
user1@ubuntu:~$
```

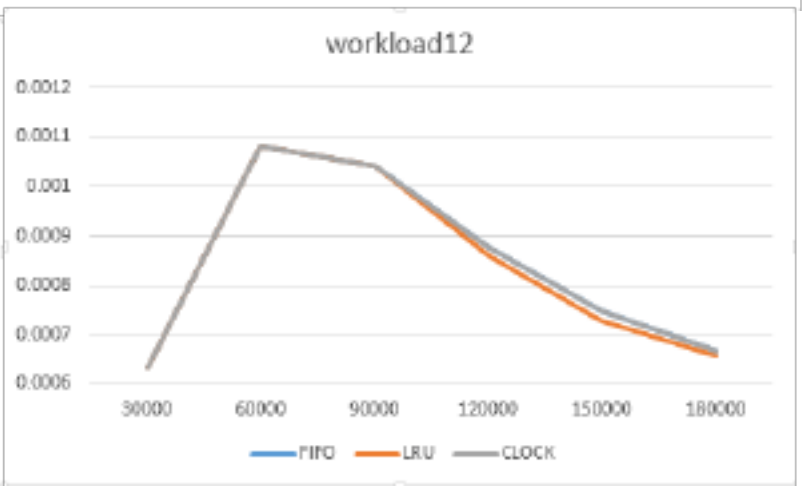
图表：

三种算法：



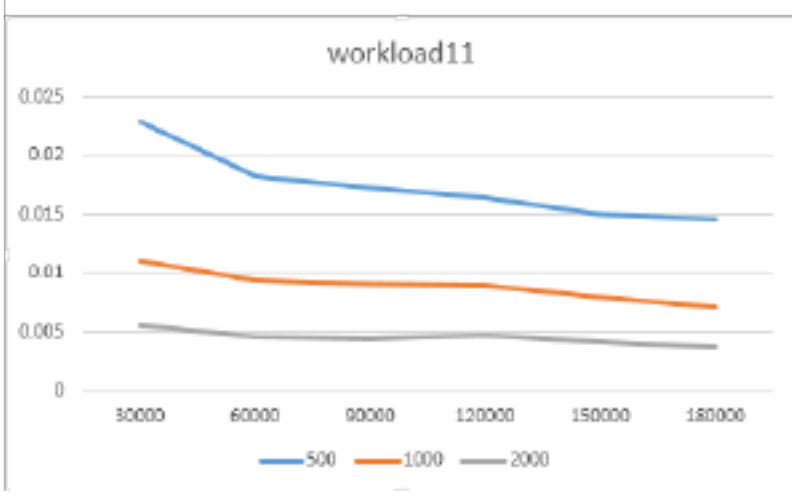



不同叶框大小 LRU 算法：








不同 workload 产生不同缺页率的原因主要在于每个 workload 中调用同一页的次数不同，缺页次数与替换次数各有不同，导致的缺页率也会不同