|  |  |  |  |
| --- | --- | --- | --- |
|  | 2017/11/29 |  | |
| |  |  | | --- | --- | |  |  | |  |  | |  |  | |  |  | | |  | |
| 操作系统实验报告  *实验五 虚拟内存页面置换算法* | | | |
|  |  | | |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |
|  |  | | 软件工程一班\_秦源\_1525161007 |

# 一 需求分析

**问题描述：**

设计程序模拟先进先出FIFO、最佳置换OPI和最近最久未使用LRU页面置换算法的工作过程。假设内存中分配给每个进程的最小物理块数为m，在进程运行过程中要访问的页面个数为n，页面访问序列为P1, … ,Pn，分别利用不同的页面置换算法调度进程的页面访问序列，给出页面访问序列的置换过程，计算每种算法缺页次数和缺页率。

**程序要求：**

1）利用先进先出FIFO、最佳置换OPI和最近最久未使用LRU三种页面置换算法模拟页面访问过程。

2）模拟三种算法的页面置换过程，给出每个页面访问时的内存分配情况。

3）输入：最小物理块数m，页面个数n，页面访问序列P1, … ,Pn，算法选择1-FIFO，2-OPI，3-LRU。

4）输出：每种算法的缺页次数和缺页率。

# 二 概要设计

最佳(Optimal)置换算法：

最佳置换算法，其所选择的被淘汰页面，将是以后永不使用的，或是在最长(未来)时间内不再被访问的页面。采用最佳置换算法，通常可保证获得最低的缺页率。

先进先出(FIFO)页面置换算法：

该算法总是淘汰最先进入内存的页面，即选择在内存中驻留时间最久的页面予以淘汰。

最近最久未使用(LRU)置换算法：

最近最久未使用(LRU)的页面置换算法，是根据页面调入内存后的使用情况进行决策的。由于无法预测各页面将来的使用情况，只能利用“最近的过去”作为“最近的将来”的近似，因此，LRU置换算法是选择最近最久未使用的页面予以淘汰。

# 三 详细设计

main.sh文件声明了静态变量以及默认数据。

Func.sh文件定义函数以及主要的算法。(细节见原代码注释)

# 四 调试分析

最佳置换算法，由于人们目前还无法预知一个进程在内存的若干个页面中，哪一个页面是未来最长时间内不再被访问的，因而该算法是无法实现的。

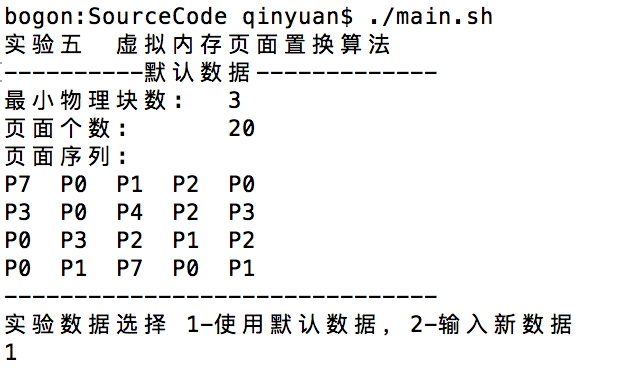
先进先出(FIFO)页面置换算法与进程实际运行的规律不相适应，因为在进程中，有些页面经常被访问，比如，含有全局变量、常用函数、例程等的页面，FIFO算法并不能保证这些页面不被淘汰。

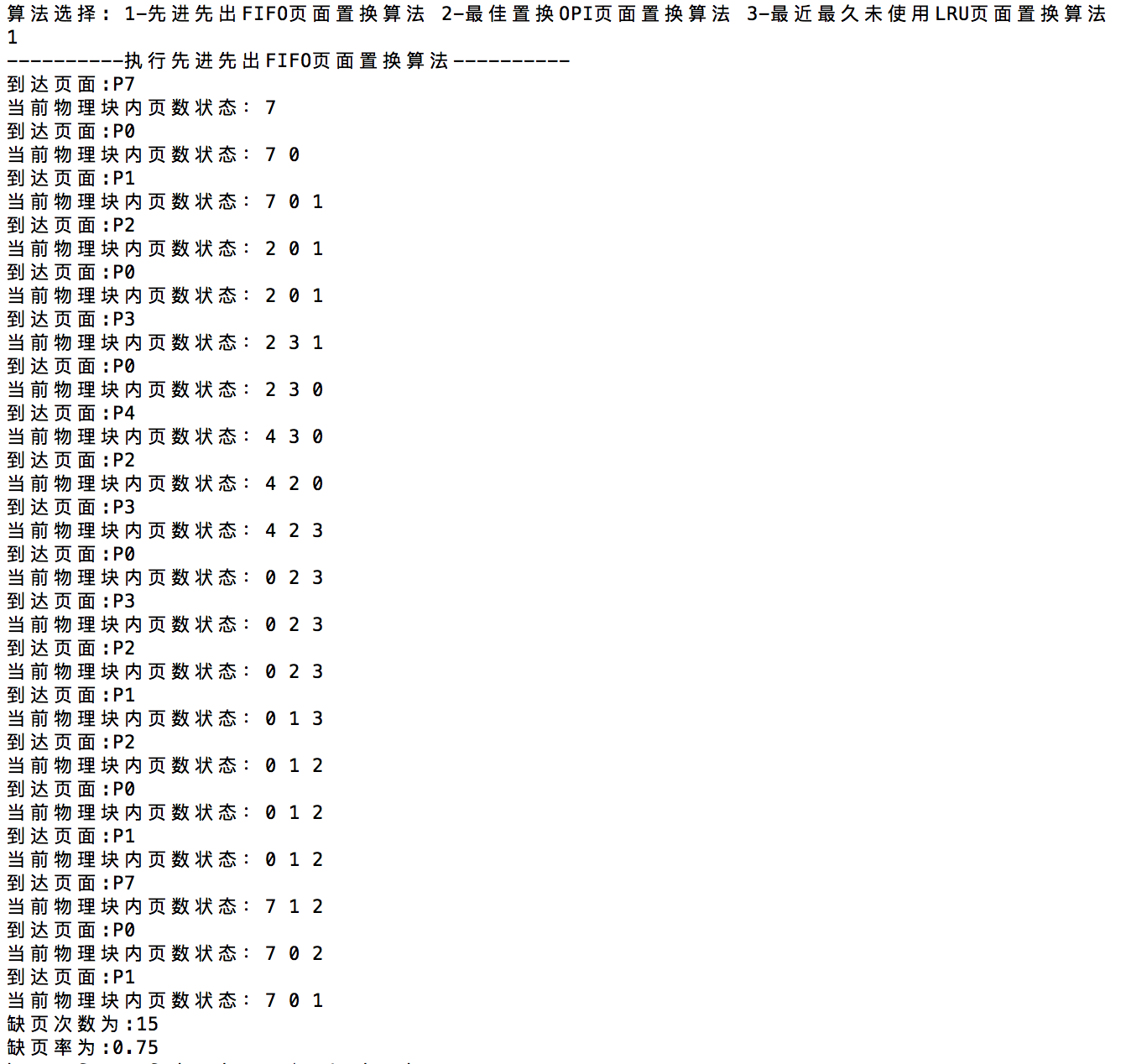
FIFO置换算法性能之所以较差，是因为它所依据的条件是各个页面调入内存的时间，而页面调入的先后并不能反映页面的使用情况。

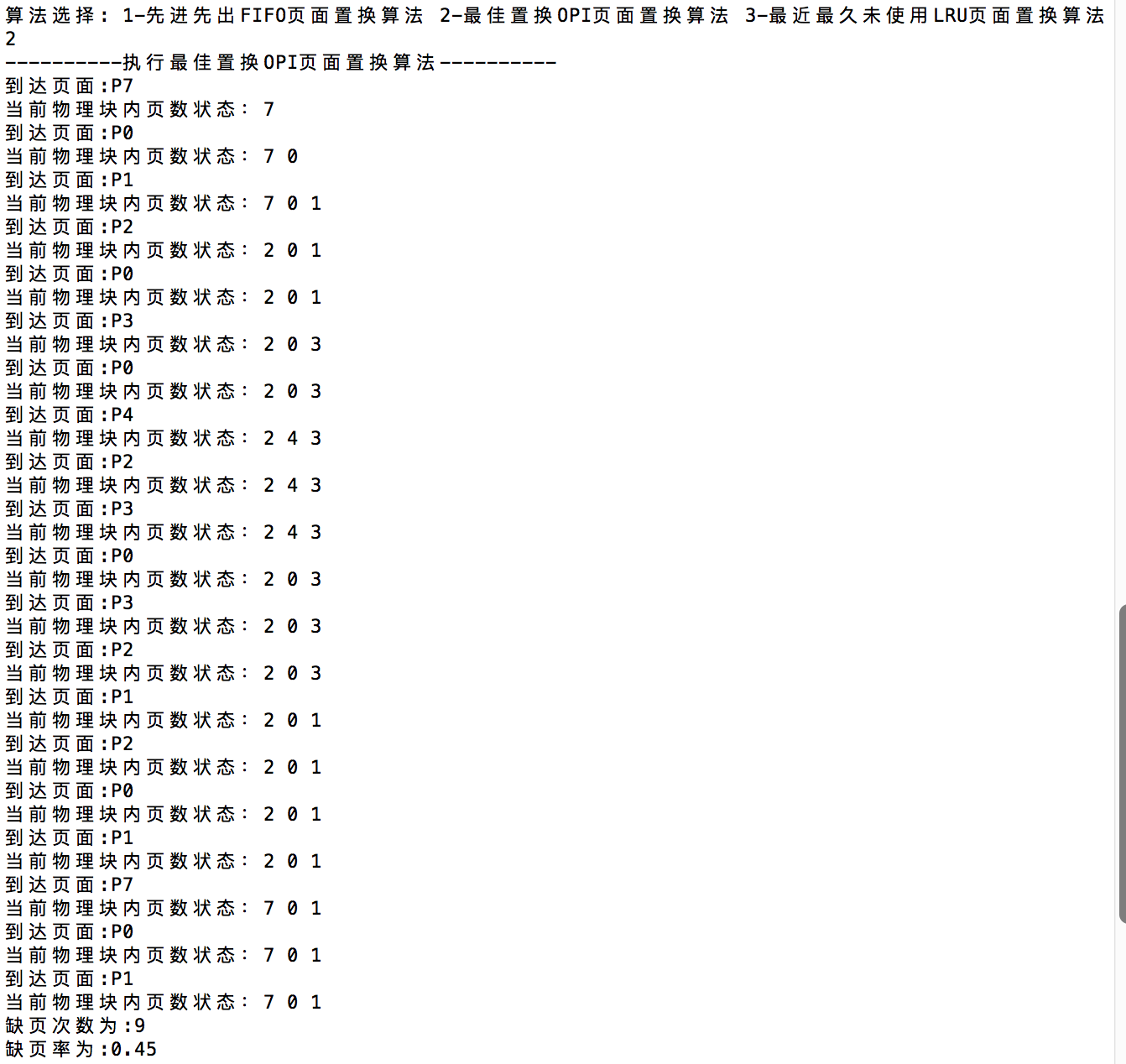
# 五 用户使用说明

使用终端，将运行目录切换到源代码所在路径，运行main.sh。根据提示即可使用。

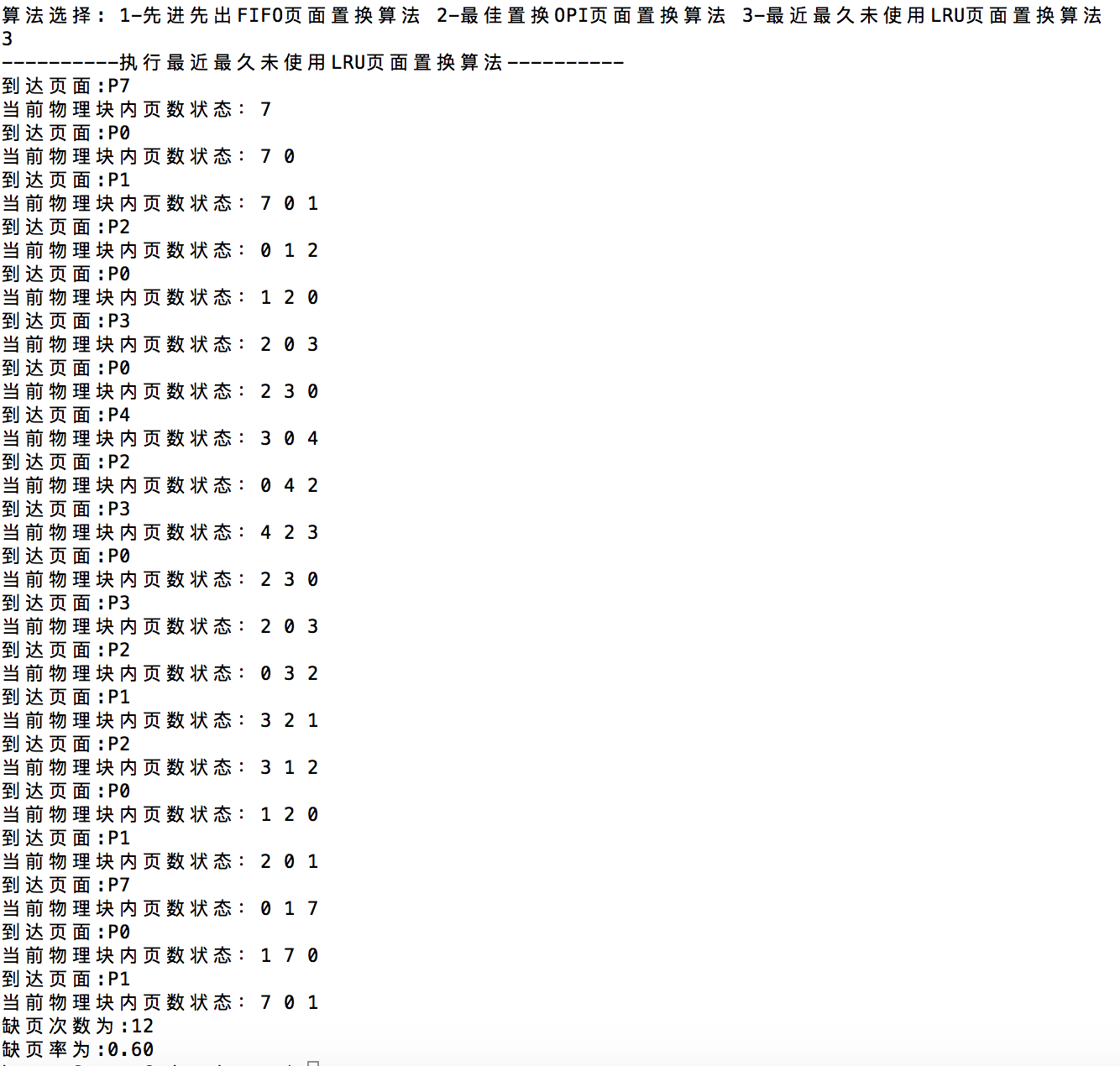
# 六 测试结果



先进先出(FIFO)页面置换算法:

最佳置换算法:

最近最久未使用LRU页面置换算法:



# 七 附录

---------------------------main.sh----------------------------------------------------------------------------------------------

#! /bin/bash

. func.sh

#保留小数位数

sc=2

#最小物理块数m

m=3

#页面个数n

n=20

#页面序列

PageOrder=(7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1)

echo "实验五 虚拟内存页面置换算法"

echo "----------默认数据-------------"

echo -e "最小物理块数:\t${m}"

echo -e "页面个数:\t${n}"

echo -e "页面序列:"

for i in `seq 1 $n`; do

echo -en "P${PageOrder[$[$i-1]]} "

if [[ $[$i%5] -eq 0 ]]; then

echo

fi

done

echo "-------------------------------"

echo "实验数据选择 1-使用默认数据，2-输入新数据"

read keypressData

case "$keypressData" in

1 )

;;

2 )

echo "请输入最小物理块数m:"

read new\_m

m="${new\_m}""

echo "请输入页面个数n:"

read new\_n

n="${new\_n}""

echo "请输入页面序列:(长度$n)"

read -a new\_P

PageOrder=("${new\_P[@]}")

;;

\* )

echo "输入无效,请输入 '1' 或 '2' 选择!"

exit

;;

esac

echo "算法选择: 1-先进先出FIFO页面置换算法 2-最佳置换OPI页面置换算法 3-最近最久未使用LRU页面置换算法"

read keypressKind

case "$keypressKind" in

1 )

echo "----------执行先进先出FIFO页面置换算法----------"

OPI\_FIFO 2

;;

2 )

echo "----------执行最佳置换OPI页面置换算法----------"

OPI\_FIFO 1

;;

3 )

echo "----------执行最近最久未使用LRU页面置换算法----------"

LRU

;;

\* )

echo "输入无效,请输入 '1' 或 '2' 或 '3' 选择!"

exit

;;

esac

---------------------------func.sh----------------------------------------------------------------------------------------------

#! /bin/bash

INIT(){

#模拟物理块状态

declare -a Simulate

#计算持续时间

declare -a PageCount

#当前页号

declare -i PageNum

#缺页次数

declare -i LackNum

#缺页率

declare LackPageRate

#FIFO替换点的Index默认0

FIFO\_Change\_Index=0

}

#OPI置换

OPI\_CHANGE(){

#判断是否发生缺页中断,默认发生中断

local cut=1

for i in `seq 0 $[$m-1]`; do

if [[ "$1" -eq "${Simulate[$i]}" ]]; then

cut=0

fi

done

#发生中断

if [[ "${cut}" -eq 1 ]]; then

LackNum=$[${LackNum}+1]

#寻找被替换的页号

local delPageIndex=-1

local delPageLength=0

#循环块数

for i in `seq 0 $[$m-1]`; do

local tempLength=0

for j in `seq $2 $[$n-1]`; do

if [[ "${PageOrder[j]}" -eq "${Simulate[i]}" ]]; then

break

fi

tempLength=$[${tempLength}+1]

done

if [[ "${tempLength}" -gt "${delPageLength}" ]]; then

delPageLength="${tempLength}"

delPageIndex="$i"

fi

done

#开始替换

Simulate[${delPageIndex}]="$1"

fi

}

#FIFO置换

FIFO\_CHANGE(){

#判断是否发生缺页中断,默认发生中断

local cut=1

for i in `seq 0 $[$m-1]`; do

if [[ "$1" -eq "${Simulate[$i]}" ]]; then

cut=0

fi

done

#发生中断

if [[ "${cut}" -eq 1 ]]; then

LackNum=$[${LackNum}+1]

#开始替换

Simulate[${FIFO\_Change\_Index}]="$1"

FIFO\_Change\_Index=$[(${FIFO\_Change\_Index}+1)%${m}]

fi

}

#OPI最佳置换算法

OPI\_FIFO(){

#初始化

INIT

#PageNum从第一页到最后一页

PageNum=0

#起始缺页次数为0

LackNum=0

#页数少于物理块数,存在情况,页面种类小于物理块数

if [[ "$n" -le "$m" ]]; then

for i in `seq 0 $[$n-1]`; do

echo "到达页面:P${PageOrder[$i]}"

if [[ "$i" -eq 0 ]]; then

Simulate[0]="${PageOrder[0]}"

else

for j in `seq 0 $[${#Simulate[@]}-1]`; do

if [[ "${PageOrder[$i]}" -eq "${Simulate[$j]}" ]]; then

echo "当前物理块内页数状态：${Simulate[@]}"

continue 2

fi

done

Simulate[${#Simulate[@]}]="${PageOrder[$i]}"

fi

echo "当前物理块内页数状态：${Simulate[@]}"

done

LackNum=$[$n-${#Simulate[@]}]

LackPageRate=$(echo "scale=${sc};${LackNum}/${n}"|bc)

echo "缺页次数为:${LackNum}"

if [[ "${LackNum}" -eq "$n" ]]; then

echo "缺页率为:1"

else

LackPageRate=$(echo "scale=${sc};${LackNum}/${n}"|bc)

echo "缺页率为:0${LackPageRate}"

fi

exit

fi

#定义Simulate满标记,0为满

FullSimulate=1

#页数大于物理块数,存在情况,页面种类小于物理块数

for i in `seq 0 $[$n-1]`; do

echo "到达页面:P${PageOrder[$i]}"

PageNum="$[${PageNum}+1]"

if [[ "$i" -eq 0 ]]; then

LackNum=$[${LackNum}+1]

Simulate[0]="${PageOrder[0]}"

if [[ "${#Simulate[@]}" -eq "$m" ]]; then

FullSimulate=0

echo "当前物理块内页数状态：${Simulate[@]}"

break

fi

else

for j in `seq 0 $[${#Simulate[@]}-1]`; do

if [[ "${PageOrder[$i]}" -eq "${Simulate[j]}" ]]; then

echo "当前物理块内页数状态：${Simulate[@]}"

continue 2

fi

done

LackNum=$[${LackNum}+1]

Simulate[${#Simulate[@]}]="${PageOrder[$i]}"

if [[ "${#Simulate[@]}" -eq "$m" ]]; then

FullSimulate=0

echo "当前物理块内页数状态：${Simulate[@]}"

break

fi

fi

echo "当前物理块内页数状态：${Simulate[@]}"

done

#Simulate装满之后

if [[ "${FullSimulate}" -eq 0 ]]; then

local startPagenum=${PageNum}

for (( i\_page = ${startPagenum}; i\_page < $n; i\_page++ )); do

PageNum="$[${PageNum}+1]"

local -i tempPage="${PageOrder[$i\_page]}"

echo "到达页面:P${tempPage}"

#置换页面,接收两个参数，1是tempPage,2是PageNum

if [[ "$1" -eq 1 ]]; then

OPI\_CHANGE ${tempPage} ${PageNum}

elif [[ "$1" -eq 2 ]]; then

FIFO\_CHANGE ${tempPage} ${PageNum}

fi

echo "当前物理块内页数状态：${Simulate[@]}"

done

fi

echo "缺页次数为:${LackNum}"

if [[ "${LackNum}" -eq "$n" ]]; then

echo "缺页率为:1"

else

LackPageRate=$(echo "scale=${sc};${LackNum}/${n}"|bc)

echo "缺页率为:0${LackPageRate}"

fi

}

#LRU置换

LRU\_CHANGE(){

#判断是否发生缺页中断,默认发生中断

local cut=1

for i in `seq 0 $[$m-1]`; do

if [[ "$1" -eq "${Simulate[$i]}" ]]; then

cut=0

#不发生中断时将该也放到栈低

tempValue=${Simulate[$i]}

unset Simulate[$i]

Simulate=("${Simulate[@]}")

Simulate[${#Simulate[@]}]="${tempValue}"

fi

done

#发生中断

if [[ "${cut}" -eq 1 ]]; then

LackNum=$[${LackNum}+1]

#出栈第一个，新进入放最后

unset Simulate[0]

Simulate=("${Simulate[@]}")

Simulate[${#Simulate[@]}]="$1"

fi

}

#最近最久未使用LRU置换算法 采用堆栈实现

LRU(){

#初始化

INIT

#PageNum从第一页到最后一页

PageNum=0

#起始缺页次数为0

LackNum=0

#页数少于物理块数,存在情况,页面种类小于物理块数

if [[ "$n" -le "$m" ]]; then

for i in `seq 0 $[$n-1]`; do

echo "到达页面:P${PageOrder[$i]}"

if [[ "$i" -eq 0 ]]; then

Simulate[0]="${PageOrder[0]}"

else

for j in `seq 0 $[${#Simulate[@]}-1]`; do

if [[ "${PageOrder[$i]}" -eq "${Simulate[$j]}" ]]; then

echo "当前物理块内页数状态：${Simulate[@]}"

continue 2

fi

done

Simulate[${#Simulate[@]}]="${PageOrder[$i]}"

fi

echo "当前物理块内页数状态：${Simulate[@]}"

done

LackNum=$[$n-${#Simulate[@]}]

LackPageRate=$(echo "scale=${sc};${LackNum}/${n}"|bc)

echo "缺页次数为:${LackNum}"

if [[ "${LackNum}" -eq "$n" ]]; then

echo "缺页率为:1"

else

LackPageRate=$(echo "scale=${sc};${LackNum}/${n}"|bc)

echo "缺页率为:0${LackPageRate}"

fi

exit

fi

#定义Simulate满标记,0为满

FullSimulate=1

#页数大于物理块数,存在情况,页面种类小于物理块数

for i in `seq 0 $[$n-1]`; do

echo "到达页面:P${PageOrder[$i]}"

PageNum="$[${PageNum}+1]"

if [[ "$i" -eq 0 ]]; then

LackNum=$[${LackNum}+1]

Simulate[0]="${PageOrder[0]}"

if [[ "${#Simulate[@]}" -eq "$m" ]]; then

FullSimulate=0

echo "当前物理块内页数状态：${Simulate[@]}"

break

fi

else

for j in `seq 0 $[${#Simulate[@]}-1]`; do

if [[ "${PageOrder[$i]}" -eq "${Simulate[$j]}" ]]; then

local tempValue=${Simulate[$j]}

unset Simulate[$j]

Simulate=("${Simulate[@]}")

Simulate[${#Simulate[@]}]="${tempValue}"

echo "当前物理块内页数状态：${Simulate[@]}"

continue 2

fi

done

LackNum=$[${LackNum}+1]

Simulate[${#Simulate[@]}]="${PageOrder[$i]}"

if [[ "${#Simulate[@]}" -eq "$m" ]]; then

FullSimulate=0

echo "当前物理块内页数状态：${Simulate[@]}"

break

fi

fi

echo "当前物理块内页数状态：${Simulate[@]}"

done

#Simulate装满之后

if [[ "${FullSimulate}" -eq 0 ]]; then

local startPagenum=${PageNum}

for (( i\_page = ${startPagenum}; i\_page < $n; i\_page++ )); do

PageNum="$[${PageNum}+1]"

local -i tempPage="${PageOrder[$i\_page]}"

echo "到达页面:P${tempPage}"

#置换页面,接收两个参数，1是tempPage,2是PageNum

LRU\_CHANGE ${tempPage} ${PageNum}

echo "当前物理块内页数状态：${Simulate[@]}"

done

fi

echo "缺页次数为:${LackNum}"

if [[ "${LackNum}" -eq "$n" ]]; then

echo "缺页率为:1"

else

LackPageRate=$(echo "scale=${sc};${LackNum}/${n}"|bc)

echo "缺页率为:0${LackPageRate}"

fi

}