#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#define TRUE 1

#define FALSE 0

#define INVALID -1

#define NULL 0

#define total\_instruction 320 /\*指令流长\*/

#define total\_vp 32 /\*虚页长\*/

#define clear\_period 50 /\*清零周期\*/

typedef struct{ /\*页面结构\*/

int pn, pfn, counter, time;

}pl\_type;

pl\_type pl[total\_vp]; /\*页面结构数组\*/

struct pfc\_struct{ /\*页面控制结构\*/

int pn, pfn;

struct pfc\_struct \*next;

};

struct pfc\_struct pfc[total\_vp],\*freepf\_head, \*busypf\_head, \*busypf\_tail;

int diseffect, a[total\_instruction];

int page[total\_instruction], offset[total\_instruction];

void initialize();

void FIFO( );

void LRU( );

void OPT( );

void LFU();

void CLOCK( );

int main()

{

int S,i;

srand(10\*getpid());

/\*由于每次运行时进程号不同，故可用来作为初始化随机数队列的“种子”\*/

S=(int)(319.0\*rand()/RAND\_MAX)+1;

for(i=0;i<total\_instruction;i+=4) /\*产生指令队列\*/

{

a[i]=S; /\*任选一指令访问点\*/

a[i+1]=a[i]+1; /\*顺序执行下一条指令\*/

a[i+2]=(int)(1.0\*a[i]\*rand()/RAND\_MAX); /\*执行前地址指令m'\*/

a[i+3]=a[i+2]+1; /\*执行后地址指令\*/

S=(int)(1.0\*rand()\*(318-a[i+2])/RAND\_MAX)+a[i+2]+2;

}

for(i=0;i<total\_instruction;i++) /\*将指令序列变换成页地址流\*/

{

page[i]=a[i]/10;

offset[i]=a[i]%10;

}

for(i=4;i<=32;i++) /\*用户内存工作区从4个页面到32个页面\*/

{

printf("%2d page frames\t",i);

FIFO(i);

LRU(i);

OPT(i);

LFU (i);

CLOCK(i);

}

return 0;

}

void initialize(int total\_pf) /\*初始化相关数据结构\*/

{

int i;

diseffect=0;

for(i=0;i<total\_vp;i++)

{

pl[i].pn=i;

pl[i].pfn=INVALID; /\*置页面控制结构中的页号，页面为空\*/

pl[i].counter=0;

pl[i].time=-1; /\*置页面控制结构中的访问次数，时间为-1\*/

}

for(i=1;i<total\_pf;i++)

{

pfc[i-1].next=&pfc[i];

pfc[i-1].pfn=i-1;

} /\*建立pfc[i-1]和pfc[i]之间的链接\*/

pfc[total\_pf-1].next=NULL;

pfc[total\_pf-1].pfn=total\_pf-1;

freepf\_head=&pfc[0];

/\*空页面队列的头指针为pfc[0]\*/

}

void FIFO(int total\_pf)

{

int i;

struct pfc\_struct \*p;

initialize(total\_pf); /\*初始化相关页面控制用数据结构\*/

busypf\_head=busypf\_tail=NULL; /\*忙页面队列，队列尾链接\*/

for(i=0;i<total\_instruction;i++)

{

if(pl[page[i]].pfn==INVALID) /\*页面失效\*/

{

diseffect+=1; /\*失效次数\*/

if(freepf\_head==NULL) /\*无空闲页面\*/

{

p=busypf\_head->next;

pl[busypf\_head->pn].pfn=INVALID;

freepf\_head=busypf\_head; /\*释放忙页面队列中的第一个页面\*/

freepf\_head->next=NULL;

busypf\_head=p;

}

p=freepf\_head->next; /\*按FIFO方式调新页面入内存页面\*/

freepf\_head->next=NULL;

freepf\_head->pn=page[i];

pl[page[i]].pfn=freepf\_head->pfn;

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:%6.4f ",1-(float)diseffect/320);

}

void LRU(int total\_pf)

{ }

void OPT(int total\_pf)

{ }

void LFU(int total\_pf)

{ }

void CLOCK(int total\_pf)

{ }