**北京邮电大学软件学院**

**2017-2018学年第1学期实验报告**

**课程名称： 操作系统**

**实验名称： 虚拟存储器管理& Linux 文件系统调用**

**实验完成人：**

**姓名：**\_\_\_\_田宇\_\_\_\_

**学号：**\_\_\_2016212011\_\_\_\_\_

**成绩：**\_\_\_\_\_\_\_\_

**指导教师：**\_\_\_\_\_\_\_\_\_\_\_\_陈晋鹏\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**日 期： 2018 年 6 月 13 日**

1. **实验目的**

（一）虚拟储存管理

1. 了解虚拟存储技术的特点；

2. 掌握请求页式管理的页面置换算法。

（二）Linux 文件系统调用

1.掌握 linux 提供的文件系统调用的使用方法；

2.熟悉文件系统的系统调用用户接口；

3.了解操作系统文件系统的工作原理和工作方式。

1. **实验内容**
2. **虚拟内存管理**

1. 通过随机数产生一个指令序列，共 320 条指令。其地址按下述原则生成：

（1）50%的指令是顺序执行的；

（2）50%的指令是均匀分布在前地址部分；

（3）50%的指令是均匀分布在后地址部分；

具体的实施方法是：

A. 在[0，319]的指令地址之间随机选取一起点 M；

B. 顺序执行一条指令，即执行地址为 M+1 的指令；

C. 在前地址[0，M+1]中随机选取一条指令并执行，该指令的地址为 M’；

D. 顺序执行一条指令，其地址为 M’+1；

E. 在后地址[M’+2，319]中随机选取一条指令并执行；

F. 重复 A—E，直到执行 320 次指令。

2. 指令序列变换成页地址流

设：

（1）页面大小为 1K；

（2）用户内存容量为 4 页到 32 页；

（3）用户虚存容量为 32K。

在用户虚存中，按每 K 存放 10 条指令排列虚存地址，即 320 条指令在虚存中的存放方式为：

第 0 条—第 9 条指令为第 0 页（对应虚存地址为[0，9]）；

第 10 条—第 19 条指令为第 1 页（对应虚存地址为[10，19]）；

……………………

第 310 条—第 319 条指令为第 31 页（对应虚存地址为[310，319]）；

按以上方式，用户指令可组成 32 页。

3. 计算并输出下述各种算法在不同内存容量下的命中率。

A. 先进先出（FIFO）页面置换算法

B. 最近最久未使用（LRU）页面置换算法--最近最少使用算法

C. 最少使用（LFR）页面置换算法

D. 最佳（Optimal）页面置换算法

1. **Linux 文件系统管理**

编写一个文件工具 filetools，使其具有以下功能：

0.退出

1.创建新文件

2.写文件

3.读文件

4.修改文件权限

5.查看当前文件权限并退出

提示用户输入功能号，并根据用户输入的功能选择相应的功能

1. **实验环境**

Linux

1. 源代码

#include<cstdio>

#include<cstring>

#define SizeOfPage 100

#define SizeOfBlock 128

#define M 4//主存中放4个页面

struct info//页表信息结构体

{

bool flag;//页标志，1表示该页已在主存，0表示该页不在主存

long block;//块号

long disk;//在磁盘上的位置

bool dirty;//更新标志

}pagelist[SizeOfPage];

long po;//队列标记

long P[M];//假设内存中最多允许M个页面

void init\_ex1()

{

memset(pagelist,0,sizeof(pagelist));//内存空间初始化

/\*分页式虚拟存储系统初始化\*/

pagelist[0].flag=1;

pagelist[0].block=5;

pagelist[0].disk=011;

pagelist[1].flag=1;

pagelist[1].block=8;

pagelist[1].disk=012;

pagelist[2].flag=1;

pagelist[2].block=9;

pagelist[2].disk=013;

pagelist[3].flag=1;

pagelist[3].block=1;

pagelist[3].disk=021;

}

void work\_ex1()//模拟分页式存储管理中硬件的地址转换和产生缺页中断过程

{

bool stop=0;

long p,q;//页号，单元号

char s[128];//初始定义块长

do

{

printf("请输入指令的页号和单元号:\n");

if(scanf("%ld%ld",&p,&q)!=2)

{

scanf("%s",s);

if(strcmp(s,"exit")==0)//如果输入的为“exit”那么就退出，进入重选页面

{

stop=1;

}

}

else

{

if(pagelist[p].flag)//如果该页标志flag为1，说明该页已在主存中

{

printf("绝对地址=%ld\n",pagelist[p].block\*SizeOfBlock+q);

//计算出绝对地址，绝对地址=块号x块长（默认128）+单元号

}

else

{

printf("\*%ld\n",p);

//如果该页标志flag为0，说明该页不在主存，则产生了一次缺页中断

}

}

}while(!stop);

}

void init\_ex2()

{

/\*

\*以下部分为先进先出（FIFO）页面调度算法处理缺页中断的初始化，其中也包含了对于当前的存储器内容的初始化

\*

\*/

po=0;

P[0]=0;P[1]=1;P[2]=2;P[3]=3;

//对内存中的4个页面进行初始化，并且使目前排在第一位的为0

memset(pagelist,0,sizeof(pagelist));//内存空间初始化

pagelist[0].flag=1;

pagelist[0].block=5;

pagelist[0].disk=011;

pagelist[1].flag=1;

pagelist[1].block=8;

pagelist[1].disk=012;

pagelist[2].flag=1;

pagelist[2].block=9;

pagelist[2].disk=013;

pagelist[3].flag=1;

pagelist[3].block=1;

pagelist[3].disk=021;

}

void work\_ex2()//模拟FIFO算法的工作过程

{

long p,q,i;

char s[100];

bool stop=0;

do

{

printf("请输入指令的页号、单元号，以及是否为内存指令:\n");

if(scanf("%ld%ld",&p,&q)!=2)

{

scanf("%s",s);

if(strcmp(s,"exit")==0)//如果输入的为“exit”就退出，进入重选界面

{

stop=1;

}

}

else

{

scanf("%s",s);

if(pagelist[p].flag)//如果该页标志flag为1，说明该页已在主存中

{

printf("绝对地址=%ld\n",pagelist[p].block\*SizeOfBlock+q);//计算绝对地址

if(s[0]=='Y'||s[0]=='y')//内存指令

{

pagelist[p].dirty=1;//修改标志为1

}

}

else

{

if(pagelist[P[po]].dirty)

//当前的页面被更新过，需把更新后的内容写回外存

{

pagelist[P[po]].dirty=0;

}

pagelist[P[po]].flag=0;

//将flag置0，表明当前页面已被置换出去

printf("out%ld\n",P[po]);

//显示根据FIFO算法被置换出去的页面

printf("in%ld\n",p);//显示根据FIFO算法被调入的页面

pagelist[p].block=pagelist[P[po]].block;//块号相同

pagelist[p].flag=1;//将当前页面flag置1,表明已在主存中

P[po]=p;//保存当前页面所在的位置

po=(po+1)%M;

}

}

}while(!stop);

printf("数组P的值为:\n");

for(i=0;i<M;i++)//循环输入当前数组的数值，即当前在内存中的页面

{

printf("P[%ld]=%ld\n",i,P[i]);

}

}

void select()//选择哪种方法进行

{

long se;

char s[128];

do

{

printf("请选择题号(1/2):");

if(scanf("%ld",&se)!=1)

{

scanf("%s",&s);

if(strcmp(s,"exit")==0)//如果输入为exit则退出整个程序

{

return;

}

}

else

{

if(se==1)

//如果se=1说明选择的模拟分页式存储管理中硬件的地址转换和产生缺页中断

{

init\_ex1();//初始化

work\_ex1();//进行模拟

}

if(se==2)//如果se=2说明选择的是FIFO算法来实现页面的置换

{

init\_ex2();//初始化

work\_ex2();//进行模拟

}

}

}while(1);

}

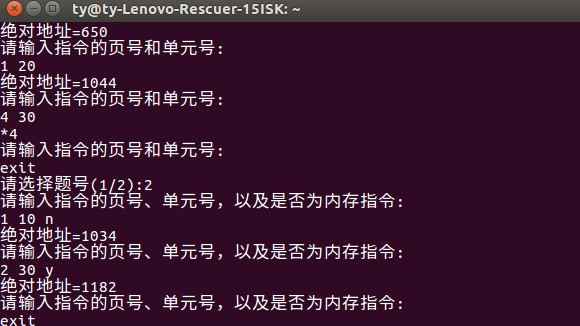
int main()

{

select();//选择题号

return 0;

}



#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/stat.h>

#include<syslog.h>

#include<string.h>

#include<stdlib.h>

#define MAX 128

int chmd();

int chmd ()

{

int c;

mode\_t mode=S\_IWUSR;

printf(" 0. 所有用户读写执行\n 1. 所有用户只可读\n 2. 所有用户只可写 \n 3. 所有用户只可执行\n 4. 用户组读写执行\n 5. 用户组只可读\n 6. 用户组只可写\n 7. 用户组只可执行\n ");

printf("请选择0-7:");

scanf("%d",&c);

switch(c)

{

case 0: chmod("newfile",S\_IRWXU);break;

case 1: chmod("newfile ",S\_IRUSR);break;

case 2: chmod("newfile ",S\_IWUSR);break;

case 3: chmod("newfile ",S\_IXUSR);break;

case 4: chmod("newfile ", S\_IRWXG);break;

case5: chmod("newfile ",S\_IRGRP);break;

case6: chmod("newfile ",S\_IWGRP);break;

case7: chmod("newfile ", S\_IXGRP);break;

default:printf("error choice!\n");

}

return(0);

}

Void main()

{

int fd;

int num;

int choice;

char buffer[MAX];

struct stat buf;

char\* path="/bin/ls";

char\* argv[4]={"ls","-l","newfile",NULL};

while(1)

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("0. 退出\n");

printf("1. 创建新文件\n");

printf("2. 写文件\n");

printf("3. 读文件\n");

printf("4. 修改文件权限\n");

printf("5. 查看当前文件的权限修改文件权限\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("请选择(0-5):");

scanf("%d",&choice);

switch(choice)

{

case 0:close(fd);

exit(0);

case 1:

fd=open("newfile",O\_RDWR|O\_TRUNC|O\_CREAT,0750);

if(fd==-1)

printf("File Create Failed!\n");

else

printf("fd = %d\n",fd);

break;

case 2:

num=read(0,buffer,MAX);

write(fd,buffer,num);

break;

case 3:

read(fd,buffer,MAX);

write(1,buffer,num);

break;

case 4:

chmd ();

printf("成功改变格式!\n");

break;

case 5:

execv(path,argv);

break;

default:

printf("error choice!\n");

}

}

}

