**大连理工大学**

**本科实验报告**

课程名称： 操作系统试验

学院（系）： 电信学部

专 业： 计算机专业

班 级： 电计1303班

学 号： 201385066

学生姓名： 乔羽

2016年 6 月 25 日

**大连理工大学实验报告**

学院（系）： 电信学部 专业： 计算机专业 班级： 电计1303

姓 名： 乔羽 学号： 201385066 组： \_\_\_

实验时间： 2016.5 2016.6 实验室： 综合楼411,412 实验台：

指导教师签字： 成绩：

**实验名称： 进程控制**

1. 实验目的和要求

加深对于进程并发执行概念的理解。实践并发进程的创建和控制方法。观察和体验进程的动态特性。进一步理解进程生命期期间创建、变换、撤销状态变换的过程。掌握进程控制的方法，了解父子进程间的控制和协作关系。练习Linux系统中进程创建与控制有关的系统调用的编程和调试技术。

1. 实验环境

Ubuntu14.04 64bit操作系统

三、实验内容

（1）

子2

子1

父

getpid()---获取进程的pid

每个进程都执行自己独立的程序，打印自己的pid；

父进程打印子进程的pid;

子1

（2）

父

子2

getpid()---获取进程的pid

每个进程都执行自己独立的程序，打印自己的pid；

父进程打印两个子进程的pid;

（3）写一个命令处理程序，能处理max(m,n), min(m,n)，average(m,n,l)这几个命令（使用exec函数族）。

Max函数

Min函数

Average函数

Exec函数族调用

1. 程序代码
2. 作业1：

#include <stdio.h>

#include <unistd.h>

int main()

{

if(fork()==0){

printf("This is the son process.\n");

printf("The process ID is %d.\n",getpid());

if(fork()==0){

printf("This is grandson process.\n");

printf("The process ID is %d.\n",getpid());

}

}

else{

printf("This is father Process.\n");

printf("The process ID is %d.\n",getpid());

}

return 0;

}

1. 作业2：

#include <stdio.h>

#include <unistd.h>

int main()

{

if(fork()==0){

printf("This is son1 process.\n");

printf("The process ID is %d.\n",getpid());

}

else{

if(fork()==0){

printf("This is son2 process.\n");

printf("The process ID is %d.\n",getpid());

}

else{

printf("This is father process.\n");

printf("The process ID is %d.\n",getpid());

}

}

return 0;

}

1. 作业3：

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#define N 10

int main()

{

char num1[N];

char num2[N];

char num3[N];

char command[N];

int flag=0;

printf("please input your operation:(max,min,average)\n");

scanf("%s",command);

if(strcmp(command,"max")==0){

printf("please input 2 arguments:\n");

scanf("%s",num1);

scanf("%s",num2);

flag=1;

}

if(strcmp(command,"min")==0){

printf("please input 2 arguments:\n");

scanf("%s",num1);

scanf("%s",num2);

flag=2;

}

if(strcmp(command,"average")==0){

printf("please input 3 arguments:\n");

scanf("%s",num1);

scanf("%s",num2);

scanf("%s",num3);

flag=3;

}

if(fork()==0){

switch(flag){

//文件路径，命令名，参数，结束标志，参数列表

case 1:execl("/home/robot/操作系统试验/max","./max",num1,num2,NULL);

break;

case 2:execl("/home/robot/操作系统试验/min","./min",num1,num2,NULL);

break;

case 3:execl("/home/robot/操作系统试验/average","./average",num1,num2,num3,NULL);

break;

default:printf("请输入正确的操作.\n");

break;

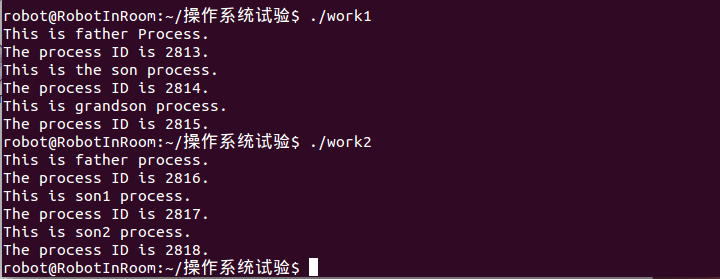
}

}

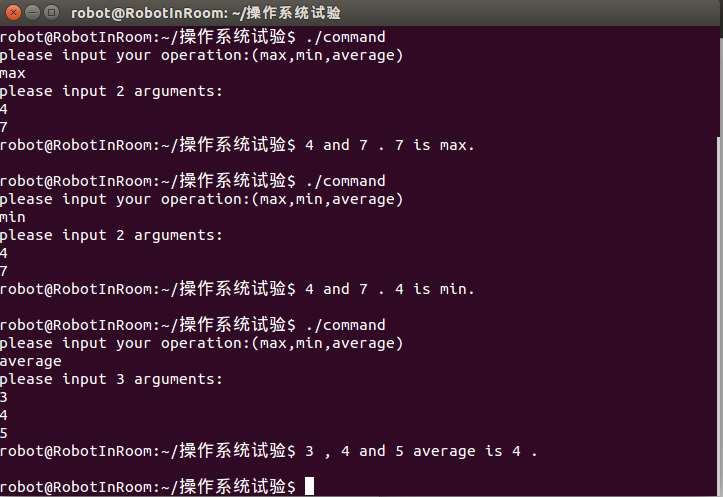
return 0;

}

1. 运行结果
2. 作业1：
3. 作业2：（均如下图所示）



1. 作业3：



1. 实验结果与分析

实验结果与之前的要求一样，正常得到显示。

七、体会

通过这次上机，我了解了fork函数的运行方法，同时更深刻的了解了进程的并行执行的本质，印证了在课堂上学习的理论知识。同时通过编写实验内容（3）的命令处理程序，学会了exec函数族工作原理和使用方法。通过这次上机实验让我加深了对课堂上学习的理论知识的理解，收获很多。

**大连理工大学实验报告**

学院（系）： 电信 专业： 计算机专业 班级： 电计1303

姓 名： 乔羽 学号： 201385066 组： \_\_\_

实验时间： 2016.5 2016.6 实验室： 综合楼411,412 实验台：

指导教师签字： 成绩：

**实验名称：存储管理上机作业**

1. 实验目的和要求

加深对于存储管理的了解，掌握虚拟存储器的实现原理；观察和了解重要的页面置换算法和置换过程。练习模拟算法的编程技巧，锻炼分析试验数据的能力。

1. 实验环境

Ubuntu14.04 64bit操作系统

三．实验内容

1. 示例实验程序中模拟两种置换算法：LRU算法和FIFO算法。

2. 能对两种算法给定任意序列不同的页面引用串和任意页面实内存数目的组合测试，显示页置换的过程。

3. 能统计和报告不同置换算法情况下依次淘汰的页号、缺页次数（页错误数）和缺页率。

四、程序代码

（1）FIFO置换算法：

#include <stdio.h>

#define N 15

#define M 5

int main(int argc,char \*argv[])

{

int i,j,k=0,temp=0,count=0,flag=0,num=0,flag2=0;

int a[N];//页栈堆列

int b[M]={0,0,0,0,0};//内存数组

int c[M]={0,0,0,0,0};//调入时间数组

printf("请输入页面调用序列：\n");

for(;i<N;i++){

scanf("%d",&a[i]);

}

//内存中已存在，直接退出循环

for(i=0;i<N;i++){

for(j=0;j<M;j++){

if(b[j]==a[i]){

flag=1;

break;

}

}

if(flag==1){

flag=0;

for(j=0;j<M;j++)

printf("%d\t",b[j]);

printf("\n");

continue;

}

count++;

//内存中不存在，则插入，并赋给c数组相应值

for(j=0;j<M;j++){

if(b[j]==0){

flag2=1;

c[j]=j;

b[j]=a[i];

break;

}

}

if(flag2==1){

flag2=0;

for(j=0;j<M;j++)

printf("%d\t",b[j]);

printf("\n");

continue;

}

//内存已满，淘汰c值最小的

for(j=0;j<M;j++){

if(c[j]==0){

temp=b[j];

break;

}

}

for(k=0;k<M;k++){

if(k!=j){

c[k]-=1;

}

}

c[j]=4;

b[j]=a[i];

printf("淘汰页号%d\n",temp);

for(j=0;j<M;j++)

printf("%d\t",b[j]);

printf("\n");

}

printf("缺页次数为%d\n",count);

printf("缺页率为%f\n",count/(float)N);

return 0;

}

1. LRU置换算法：

#include <stdio.h>

#define N 15

#define M 5

int main(int argc,char \*argv[])

{

int i,j,k=0,count=0,flag=0,num=0,flag2=0;

int a[N];//页栈堆列

int b[M]={0,0,0,0,0};//内存页堆列

printf("请输入页面调用序列：\n");

for(;i<N;i++){

scanf("%d",&a[i]);

}

//

for(i=0;i<N;i++){

for(j=0;j<M;j++){

if(b[j]==a[i]){

flag=1;

for(k=j;k>0;k--){

b[k]=b[k-1];

}

b[0]=a[i];

break;

}

}

if(flag==1){

flag=0;

for(j=0;j<M;j++)

printf("%d\t",b[j]);

printf("\n");

continue;

}

//

count++;

for(j=0;j<M;j++){

if(b[j]==0){

flag2=1;

for(k=j;k>0;k--){

b[k]=b[k-1];

}

b[0]=a[i];

break;

}

}

if(flag2==1){

flag2=0;

for(j=0;j<M;j++)

printf("%d\t",b[j]);

printf("\n");

continue;

}

//

printf("淘汰页号%d\n",b[M-1]);

for(k=M;k>0;k--){

b[k]=b[k-1];

}

b[0]=a[i];

for(j=0;j<M;j++)

printf("%d\t",b[j]);

printf("\n");

}

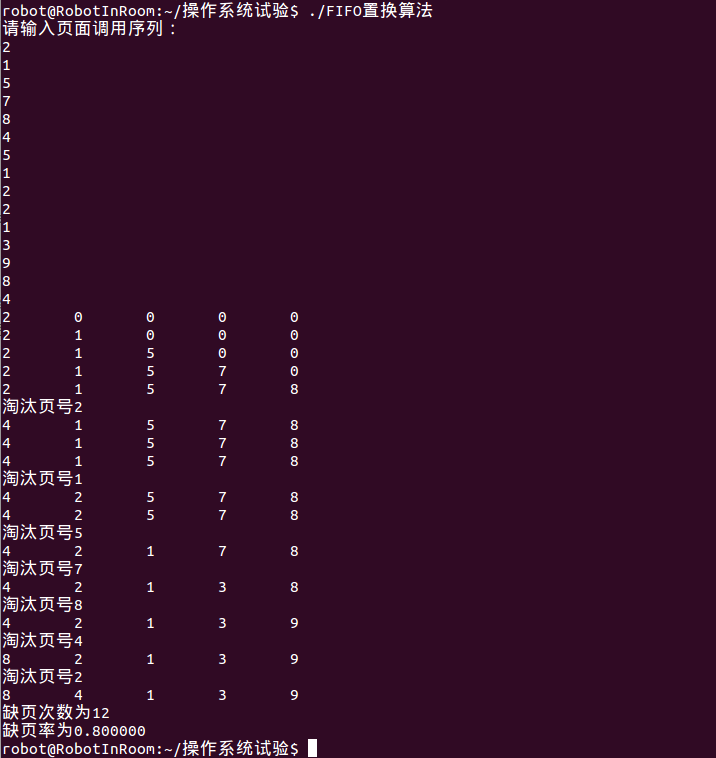
printf("缺页次数为%d\n",count);

printf("缺页率为%f\n",count/(float)N);

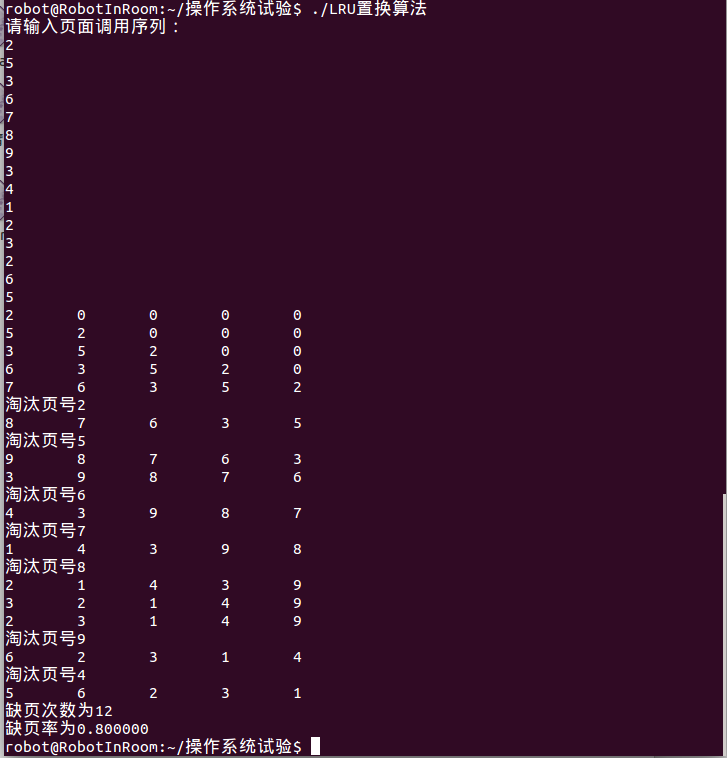
return 0;

}

1. 运行结果
2. FIFO置换算法：



1. LRU置换算法：



1. 实验结果与分析

实验结果与算法的原理所推断的结果完全一致。

1. 分析

通过本次试验，我对页面置换的相关算法有了更深层次的了解，对于我学习好操作系统这门课程也有很大的帮助。

**大连理工大学实验报告**

学院（系）： 电信 专业： 计算机专业 班级： 电计1303

姓 名： 乔羽 学号： 201385066 组： \_\_\_

实验时间： 2016.5 2016.6 实验室： 综合楼411,412 实验台：

指导教师签字： 成绩：

**实验名称：磁盘移臂调度算法实验**

一、 实验目的和要求

加深对于操作系统设备管理技术的了解，体验磁盘移臂调度算法的重要性；掌握几种重要的磁盘移臂调度算法，练习模拟算法的编程技巧，锻炼研究分析试验数据的能力。

二、 实验环境

Ubuntu14.04 64bit 操作系统

三．实验内容

1. 示例实验程序中模拟两种磁盘移臂调度算法：SSTF算法和SCAN算法

2. 能对两种算法给定任意序列不同的磁盘请求序列，显示响应磁盘请求的过程。

3. 能统计和报告不同算法情况下响应请求的顺序、移臂的总量。

四、程序代码

(1)SSTF移臂调度算法:

#include<stdio.h>

#include<math.h>

#define MAXMUM 10000

#define N 15

void main()

{

int i,n=0,min,mark=-1,sum=0,now;

int a[N];

int flag[N];

printf("请输入磁头位置：\n");

scanf("%d",&now);

printf("请输入磁盘请求序列：\n");

for(i=0;i<N;i++)

scanf("%d",&a[i]);

for(i=0;i<N;i++)

flag[i]=0;

min=MAXMUM;

printf("输出响应磁盘请求的顺序：\n");

printf("%d\t",now);

for(;n<N;n++)

{

for(i=0;i<N;i++)

if(flag[i]==0)

if(min>abs(a[i]-now))

{

min=abs(a[i]-now);

mark=i;

}

flag[mark]=1;

sum+=min;

printf("%d\t",a[mark]);

min=MAXMUM;

now=a[mark];

}

printf("\n输出移臂的总量：\n");

printf("%d\n",sum);

}

1. SCAN移臂调度算法：

#include<stdio.h>

#include<math.h>

#define N 15

void sort(int a[N],int len)//排序

{

int i,j,t;

for(j=0;j<len;j++)

{

for(i=0;i<len-j-1;i++)

if(a[i]>a[i+1])

{

t=a[i];

a[i]=a[i+1];

a[i+1]=t;

}

}

}

void main()

{

int i,n=0,min,mark=-1,sum=0,now,direct;

int a[N];

printf("请输入磁头位置：\n");

scanf("%d",&now);

printf("请输入方向:(0表示像小，1表示向大）\n");

scanf("%d",&direct);

printf("请输入磁盘请求序列：\n");

for(i=0;i<N-1;i++)

scanf("%d",&a[i]);

a[N-1]=now;

printf("输出响应磁盘请求的顺序：\n");

printf("%d\t",now);

sort(a,N);

for(n=0;n<N;n++){

if(a[n]==now){

break;

}

}

if(direct==0){

for(i=n-1;i>=0;i--){

printf("%d\t",a[i]);

sum+=now-a[i];

now=a[i];

}

for(i=n+1;i<N;i++){

printf("%d\t",a[i]);

sum+=a[i]-now;

now=a[i];

}

}

else{

for(i=n+1;i<N;i++){

printf("%d\t",a[i]);

sum+=a[i]-now;

now=a[i];

}

for(i=n-1;i>=0;i--){

printf("%d\t",a[i]);

sum+=now-a[i];

now=a[i];

}

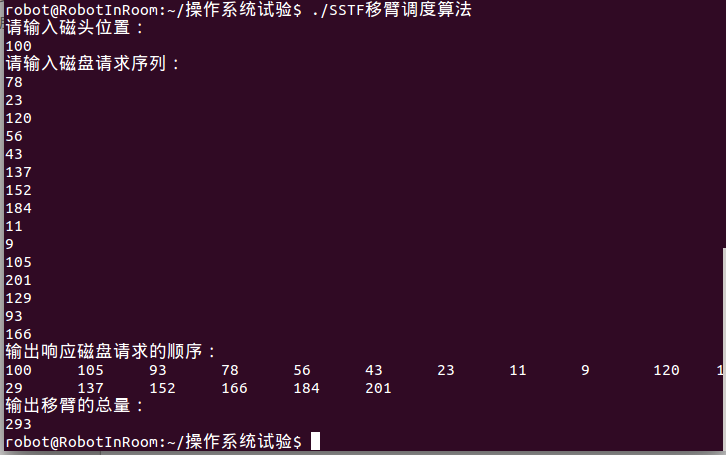
}

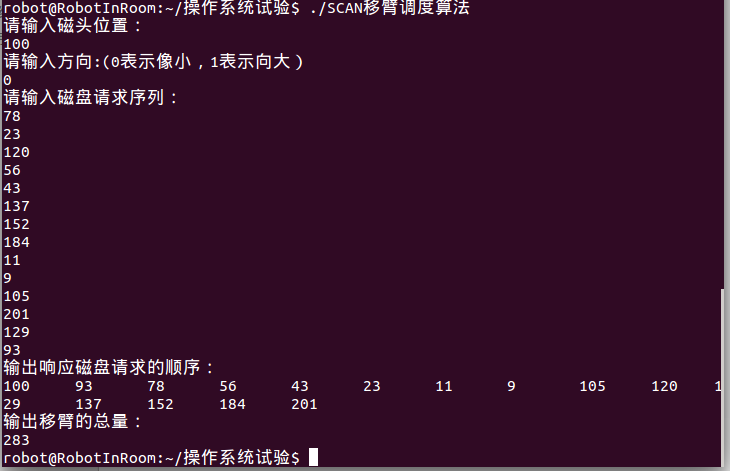
printf("\n输出移臂的总量：\n");

printf("%d\n",sum);

}

1. 运行结果

（1）SSTF移臂调度算法:  
 

(2)SCAN移臂调度算法:  


六、实验结果与分析

实验结果与算法原理一致

1. 体会

本次试验让我对移臂调度算法有了更好的了解。