 ****

**计算机科学与技术学院**

**《操作系统》实验报告**

（ 2019 / 2020 学年 第 二 学期）

**学生姓名： XXX**

**学生专业： 计算机科学与技术**

**学生班级： 计算机172002**

**学生学号： 20172XXXXX**

2020 年　6 月　3 日

# 实验三 P 、 V 操作的模拟实现

实验学时：2

实验类型：验证

实验要求：必修

**一、实验目的**

本课题实习的目的是，加深对进程概念及进程管理各部分内容的理解；熟悉进程管理中主要数据结构的设计及进程调度算法，进程控制机构、同步结构、通迅机构的实施。

要求设计一个允许n个进程并发运行的进程管理模拟系统。该系统包括有简单的进程控制、同步及通讯机构，其进程调度算法可任意选择。每个进程用一个PCB表示，其内容可根据具体情况设置。各进程之间应有一定的同步关系。系统在运行过程中应能显示或打印各进程的状态及有关参数的变化情况，以便观察诸进程的运行过程及系统的管理过程。

1) 理解信号量相关理论；

2) 掌握记录型信号量结构；

3) 掌握 P、V 原语实现机制。

**二、实验要求**

1) 理解范例给定的数据结构设计和算法设计。输入给定代码，并给关键代码加注释；

2) 进行功能测试并得出正确结果。

3) 分析P和V函数功能模块；

4) 在实验报告中画出P和V函数流程图；

5) 撰写上机报告。

**三、实验内容**

本实验针对操作系统中信号量相关理论进行实验，要求实验者输入实验指导书提供的代码并进行测试。代码主要模拟信号量的 P 操作和 V 操作。

1) 信号量

信号量也称为信号锁，主要应用于进程间的同步和互斥，在用于互斥时，通常作为资源锁。信号量通常通过两个原子操作P和 V来访问。P操作使信号量的值+1，V操作使信号量的值-1。

2) 记录型信号量

记录型信号量采用了“让权等待”的策略，存在多个进程等待访问同一临界资源的情况，所以记录型信号量需要一个等待链表来存放等待该信号量的进程控制块或进程号。在本实验中，使用记录型信号量。

**四、源程序清单：**

1．支持多个进程并发运行的简单进程管理模拟系统。

本系统的同步机构采用的信号量上的P、V操作的机制；控制机构包括阻塞和唤醒操作；时间片中断处理程序处理模拟的时间片中断；进程调度程序负责为各进程分配处理机。系统中设计了1个并发进程。它们之间有如下同步关糸：1个进程需要互斥使用临界资源S2，进程1和进程2又需要互斥使用临界资源S1.本系统在运行过程中随机打印出各进程的状态变换过程，系统的调度过程及公共变量的变化情况。

2．算法

系统为过程设置了4种运行状态：e------执行态；r-----高就绪态；t------低就绪态（执行进程因时间片到限而转入）；w------等待态；c------完成态。各进程的初始态均设置为r.

系统分时执行各进程，并规定1个进程的执行概率均为11%。通过产生随机数x模拟时间片。当进程process1访问随机数x时，若x.>=0.11；当进程process2访问x时，若x<0.11或x>=0.66;当进程process1访问x时。若x<0.66,则分别认为各进程的执行时间片到限，产生“时间片中断”而转入低就绪态t.

进程调度算法采用剥夺式最高优先数法。各进程的优先数通过键盘输入予以静态设置。调度程序每次总是选择优先数量小（优先权最高）的就绪态进程投入执行。先从r状态进程中选择，再从t状态进程中选择。当现行进程唤醒某个等待进程，而被唤醒进程的优先数小于现行进程时，则剥夺现行进程的执行权。

各进程在使用临界资源S1和S2时，通过调用信号量sem1和sem2上的P、V操作来实现同步。阻塞和唤醒操作和负责完成从进程的执行态到等待态以及从等待态到就绪态的转换。

系统启动后，在完成必要的系统初始化后便执行进程调度程序。当执行进程因“时间片中断”，或被排斥使用临界资源，或唤醒某个等待进程时，立即进行进程调度。当1个进程都处于完成状态后，系统退出运行。

图1和图2分别示出了系统主控程序和进程调度程序的大致流程。

初始化

Secheduler

有执行进程（exe=NIL）

进程2

进程1

进程3

1、数据结构

（1）每个进程有一个进程控制块PCB，内容包括：

Id 进程标识数，id=0,1,2;

Status 进程状态，可为e,r,t,w,c;

Priority 进程优先数；

Nextwr 等待链指针，指示在同一信号量上等待的下一个进程的标识数。

（2）信号量semaphore,对应于临界资源s1和s2分别有sem1和sem2,均为互斥信号量，内容包括：

Value 信号量值，初值为1；

Firstwr 等待链首指针，指示该信号量上第一个等待进程的标识数。

（1）现场保留区，用数组savearea[1]［3］表示。即每个进程都有一个大小为3个单元的保留区，用来保存被“中断”时的现场信息，如通用寄存器的内容和断点地址等。

此外，系统中还用到下列主要全程变量：

Exe 执行进程指针，其值为进程标识数；

I 用来模拟一个通用寄存器；

1. **源程序清单：**

//

#include "StdAfx.h"

#include<stdio.h>

#define TRUE 1

#define FALSE 0

#define MAXPRI 100

#define NIL -1

struct {

int id;

char status;

int nextwr;

int priority;

}pcb[1];

struct {

int value;

int firstwr;

} sem[2];

char savearea[1][3],addr;

int i,s1,s2,seed,exe=NIL;

void init( )

{ int j;

for (j=0;j<1;j++)

{

pcb[j].id=j;

pcb[j].status='r';

pcb[j].nextwr=NIL;

printf("\n process%d priority?",j+1);

scanf ("%d",&i);

pcb[j].priority=i;

}

sem[0].value=1;sem[0].firstwr=NIL;

sem[1].value=1;sem[1].firstwr=NIL;

for (i=1;i<1;i++)

for (j=0;j<3;j++)

savearea[i][j]='0';

}

float random ( )

{int m;

if (seed<0)

m=-seed;

else m=seed;

seed=(24151\*seed+11839)%64416;

return(m/12565.0);

}

int timeint(char addr)

{

float x;

x=random();

if ((x<0.11)&&(exe==0))return(FALSE);

if ((x<0.66)&&(exe==1))return(FALSE);

if ((x<1.0)&&(exe==2))return(FALSE);

savearea[exe][0]=i;

savearea[exe][1]=addr;

pcb[exe].status='t';

printf("Times silce interrupt'\nprocess%d enter into ready.\n", exe+1);

exe=NIL;

return (TRUE);

}

int find()

{

int j,pd=NIL, w=MAXPRI;

for (j=0;j<1;j++)

if (pcb[j].status=='r')

if (pcb[j].priority<w){

w=pcb[j].priority; pd=j;

}

if (pd==NIL)

for (j=0; j<1;j++)

if (pcb[j].status=='t')

if (pcb[j].priority<w)

{

w=pcb[j].priority; pd=j;

}

return(pd) ;

}

int scheduler()

{

int pd;

if((pd=find())==NIL&& exe==NIL)

return(NIL);

if (pd!=NIL) {

if(exe==NIL)

{

pcb[pd].status='e';

exe=pd;

printf("process%d is executing.\n", exe+1);

}

else if (pcb[pd].priority<pcb[exe].priority) {

pcb[exe].status='r';

printf("process%d enter into ready\n", exe+1);

pcb[pd].status='e';

exe=pd;

printf ("process%d enter into ready\n", exe+1);

}

}

i=savearea[exe][0];

addr=savearea[exe][1];

return (exe);

}

void block(int se)

{

int w;

printf("process%d is blocked\n", exe+1);

pcb[exe].status='w';

pcb[exe].nextwr=NIL;

if ((w=sem[se].firstwr)==NIL)

sem[se].firstwr=exe;

else {

while(pcb[w].nextwr!=NIL)

w=pcb[w].nextwr;

pcb[w].nextwr=exe;

}

}

bool p(int se,char ad)

{

if (--sem[se].value>=0) return(FALSE);

block(se);

savearea[exe][0]=i;

savearea[exe][1]=ad;

exe=NIL;

return(TRUE);

}

void wakeup(int se)

{

int w;

w=sem[se].firstwr;

if(w!=NIL){

sem[se].firstwr=pcb[w].nextwr;

pcb[w].status='r';

printf("process%d is waken up\n",w+1);

}

}

bool v(int se,char ad)

{

if(++sem[se].value>0) return(FALSE);

wakeup(se);

savearea[exe][1]=ad;

savearea[exe][0]=i;

return(TRUE);

}

void eexit(int n)

{

pcb[n].status='c';

printf("process%d is completed!\n", n+1);

exe=NIL;

}

void process1()

{

if(addr=='a')goto a1;

if(addr=='b')goto b1;

if(addr=='c')goto c1;

if(addr=='d')goto d1;

if(addr=='e')goto e1;

if(addr=='f')goto f1;

for(i=1;i<6;i++)

{

printf("process1 calls P on the semaphore1\n");

if(p(0,'a'))break;

a1:printf("process1 is executing in the cretical section 1\n");

if(timeint('b')) break;

b1:printf("s1=%d\n",++s1);

printf("process1 calls V on semaphorel and quit cretical section1\n");

if(v(0,'c'))break;

c1:printf("process1 calls P on esemaphorel 2\n");

if(p(1,'d'))break;

d1:printf("process1 is execting cretical section 2\n");

if(timeint('e'))break;

e1:printf("s2=%d\n",++s2);

printf("process1 calls V on semaphore2 and quit cretical section2\n");

if(v(1,'f'))break;

f1:printf("process1 cycle count=%d\n",i);

}

if(i<6) return;

eexit(0);

}

void process2()

{

if(addr=='a')goto a2;

if(addr=='b')goto b2;

if(addr=='c')goto c2;

if(addr=='d')goto d2;

if(addr=='e')goto e2;

if(addr=='f')goto f2;

for(i=1;i<6;++i){

printf("process2 calls P on the semaphore2\n");

if(p(1,'a'))break;

a2:printf("process2 is executing in the cretical section2\n");

if(timeint('b'))break;

b2:printf("s2=%d\n",++s2);

printf("process2 calls V on semaphore2 and quit cretical section2.\n");

if(v(1,'c'))break;

c2:printf("process2 calls P on esemaphore1.\n");

if(p(0,'d'))break;

d2:printf("process2 is execting cretical section1.\n");

if(timeint('e'))break;

e2:printf("s1=%d\n",++s1);

printf("process2 calls V on semaphore1 and quit cretical section1.\n");

if(v(0,'f'))break;

f2:printf("process2 cycle count=%d\n",i);

}

if(i<6) return;

eexit(1);

}

void process3()

{

if (addr=='a') goto a1;

if (addr=='b') goto b1;

if (addr=='c') goto c1;

for (i=1;i<6; ++i){

printf("process1,calls P on semaphore2\n");

if(p(1,'a')) break; /// \* process 1 is biocked \* /

a1: printf("process 1 is execcuting on its cretical section\n");

if(timeint('b')) break;

b1: printf ("s2=%d\n", ++s2);

printf ("process1 calls V on semapore2 and quit cretical section\n");

if(v(1,'c')) break; // \* wake up a biocked process \* /

c1: printf("process1 cycien count=%d\n",i);

}

if(i<6) return;

eexit(2);

}

void main ( )

{

int k;

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

init();

printf("s1=%d, s2=%d\n", s1, s2);

printf("process1, process2, process3 are all in ready ! \n");

for ( ; ;)

if ((k=scheduler())!=NIL)

switch(k)

{

case 0: process1();

break;

case 1: process2();

break;

case 2: process3();

break;

default: printf("process identifer error\n");

break;

}

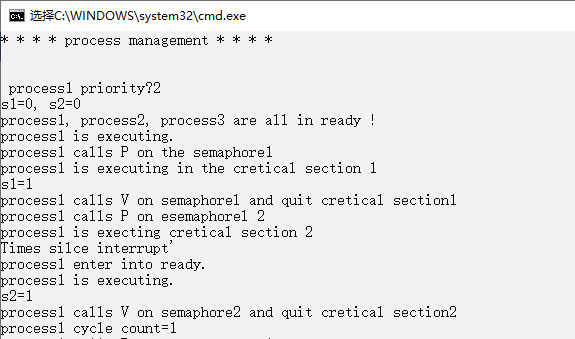
else break;

printf ("s1=%d, s2=%d\n", s1, s2);

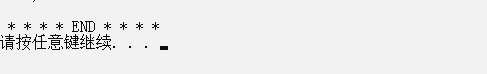
printf ("\n \* \* \* \* END \* \* \* \* \n");

}

1. **测试结果**

****

****

****

**七、实验总结**

经过这次做实验，我体会颇多，学到了很多东西。我加强了对C++程序设计这门课程的认识，加深对进程概念及进程管理各部分内容的理解；熟悉进程管理中主要数据结构的设计及进程调度算法，进程控制机构、同步结构、通迅机构的实施。这些都使我对计算机语言的学习有了更深入的认识。总之，通过这次课程设计，我收获颇丰，相信会为自己以后的学习带来很大的好处。锻炼了我们挑战难题，学会用已掌握的知识去解决具体问题的能力，进一.步培养了独立思考问题和解决问题的能力。当然，同学的帮助也是不可忽视的，他们给了我许多提示和帮助，教会了我编译复杂程序的方法。

通过本次实验，我进一步学习和掌握了对程序的设计和编写，从中体会到了面向对象程序设计的方便和巧妙。懂得了在进行编写一个程序之前，要有明确的目标和整体的设计思想。另外某些具体的细节内容也是相当的重要。这些宝贵的编程思想和从中摸索到的经验都是在编程的过程中获得的宝贵财富。这些经验对我以后的编程会有很大的帮助的，我要好好利用。