**操作系统2021课后应用题作业2**

**姓名：\_\_\_陶泽华\_\_\_\_ 学号：\_\_\_\_191250133\_\_\_\_\_\_**

**提醒：直接在本文档填写解题答案，  
提交作业的文件名命名规范为【学号\_姓名\_作业2.doc】**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **题序** | **1**  **满分8分** | **2**  **满分6分** | **3**  **满分8分** | **4**  **满分8分** | **5**  **满分6分** | **6**  **满分8分** | **7**  **满分16分** | **8**  **满分16** | **9**  **满分16** | **10**  **满分8** | **总分** |
| **分值** |  |  |  |  |  |  |  |  |  |  |  |

1、假定磁盘有200个柱面，编号0～199，当前移动臂位于143号柱面上，并刚刚完成125号柱面的服务请求。如果请求队列的先后顺序是：86，147，91，177，94，150，102，175，130；试问：为了完成上述请求，下列算法移动臂所移动的总量分别是多少？并给出移动臂移动的顺序。①先来先服务算法；②最短查找时间优先算法；③双向扫描算法；④电梯调度算法。

**答**：**（12分，每小题3分）**

1. 先来先服务算法：143-86-147-91-177-94-150-102-175-130，总量为：57+61+56+86+83+56+48+73+45=565
2. 最短查找时间优先算法：143-147-150-130-102-94-91-86-175-177，总量为：4+3+20+28+8+3+5+89+2=162
3. 双向扫描算法：143-147-150-175-177-199-130-102-94-91-86，总量为：4+3+25+2+22+69+28+8+3+5=169
4. 电梯调度算法：143-147-150-175-177-130-102-94-91-86，总量为：4+3+25+2+47+28+8+3+5=125

2、有一个磁盘组共有10个盘面，每个盘面有100个磁道，每个磁道有16个扇区。若以扇区为分配单位，现问：(1)用位示图管理磁盘空间，则位示图占用多少空间?(2)若空白文件目录的每个目录项占5个字节，则什么时候空白文件目录大于位示图?

**答：（6分，每小题3分）**

1. 共有10\*100\*16=16000个扇区，则位示图占用16000/8=2000个字节
2. 空白文件目录每个目录项占5字节，要大于位示图，需要有2000/5=400个空白文件。则当空白文件个数>400时，空白文件目录大于位示图。

3、假设在Unix文件系统中，inode节点中分别含有10个直接地址的索引和一、二、三级间接索引。若设每个盘块有512B大小，每个盘块中可存放128个盘块地址，则(1)一个1MB的文件占用多少间接盘块？(2)一个25MB的文件占用多少间接盘块？**（8分，每小题4分）**

**答**：

直接块容量=10x512B/1024=5KB

一次间接容量=128x512B/1024=64KB

二次间接容量=128x128x512B/1024=8192KB

三次间接容量=128x128x128x512B/1024=1048576KB

1. 1MB=1024KB，一次间接盘块64KB/512B=128个，二次间接盘块（1024-69）KB/512B=1910个，所以一个1MB的文件分别占128个一次间接盘块和1910个二次间接盘块，共占2038个间接盘快。
2. 25MB=25x1024KB。一次间接盘块64KB/512B=128个，二次间接盘块8192 KB/512B=16384个，三次间接盘块（25x1024-8192-64-5）KB/512B=34678个。所以一个25MB的文件分别占128个一次间接盘块、16384个二次间接盘块和34678个三次间接盘块，共占51190个间接盘块。

4、【基本概念】**（8分，每小题4分）**

设有n个进程共享一个互斥段，如果：①每次只允许一个进程进入互斥段；②每次最多允许m个进程（m≤n）同时进入互斥段。

试问：以上两种情况下所采用的信号量初值是否相同？试给出信号量值的变化范围。

**答：**初值不相同

1. 信号量的初值为1，变化范围：[-n+1,1]
2. 信号量的初值为m，变化范围：[-n + m , m]

5、【基本概念】**（6分）**

有两个优先级相同的进程P1和P2，其各自程序如下，信号量S1和S2的初值均0。试问P1、P2并发执行后，x、y、z的值各为多少？

|  |  |
| --- | --- |
| P1() {  y=1;  y=y+3;  V(S1);  z=y+1;  P(S2);  y=z+y;  } | P2() {  x=1;  x=x+5;  P(S1);  x=x+y;  V(S2);  z=z+x;  } |

**答**：因为，S1和S2的初值均为0，所以P2在P(S1)操作时被阻塞，执行P1中的V(S1)操作，此时P2申请到S1信号量的资源可以继续执行。z=y+1=5，x=x+y=10。随后在P(S2)操作中P1被阻塞。随后P2进行V(S2)释放S2信号量的资源。

此时若先执行z=z+x=15，后执行y=z+y=19。并发执行后x=10，y=19，z=15

若先执行y=z+y=9，后执行z=z+x=15。并发执行后x=10，y=9，z=15

若z=y+1在P2全部执行完之后在执行，结果为x=10，y=9，z=5

**6、**【PV】四个进程Pi（i=0…3）和四个信箱Mj（j=0…3），进程间借助相邻信箱传递消息，即Pi每次从Mi中取一条消息，经加工后送入M(i+1)mod4，其中M0、M1、M2、M3分别可存放3、3、2、2个消息。初始状态下，M0装了三条消息，其余为空。试以P、V操作为工具，写出Pi（i=0…3）的同步工作算法。



**答：（8分）**

Semaphore put0=0 put1=3 put2=2 put3=2 // 对应信箱中可放的信息数

Semaphore get0=3 get1=0 get2=0 get3=0 // 对应信箱中已有的信息数

Semaphore mutex0=mutex1=mutex2=mutex3=1

process Pi(i=0…3){

while(1){

P(geti);

P(mutexi);

Message m=Mi.getMessage();

V(mutexi);

V(puti);

P(put(i+1)%4);

P(mutex(i+1)%4);

// 加工消息

M(i+1)%4.add(m);

V(mutex(i+1)%4);

V(get(i+1)%4);

}

}

7、【PV、管程】有一个阅览室，读者进入时必须先在一张登记表上登记，此表为每个座位列出一个表目，包括座位号、姓名，读者离开时要注销登记信息；假如阅览室共有100个座位。试用：①信号量和PV操作；②管程，实现用户进程的同步算法。

**（满分16分，每小题8分，即PV题8分，管程8分）**

答：

1. Semaphore seats=100; // 座位资源信号量

Semaphore table=1; // 同时只能有一个人在使用登记表

process readeri(){

P(seats);

P(table);

// 登记座位号、姓名

V(table);

// 阅读

P(table);

// 注销登记信息

V(table);

V(seats);

}

1. 管程：

typedef read=monitor{

semaphore mutex;

int mutex\_count;

int person; // 读者的人数

mutex=0;

mutex\_count = 0;

person = 0;

InterfaceModule IM;

define in(),out();

use wait(),enter(),signal(),leave();

void in(){

enter(IM);

if(person >=100) wait(mutex,mutex\_count,IM);

person ++;

// 登记信息

leave(IM);

}

void out(){

enter(IM);

person--;

// 注销登记信息

signal(mutex,mutex\_count,IM);

leave(IM);

}

};

begin:

process readeri(){

read.in();

// 阅读

read.out();

}

end;

8、【PV、管程】在一个盒子里，混装了数量相等的黑白围棋子。现在用自动分拣系统把黑子、白子分开，设分拣系统有二个进程P1和P2，其中P1拣白子；P2拣黑子。规定每个进程每次拣一子；当一个进程在拣时，不允许另一个进程去拣；当一个进程拣了一子时，必须让另一个进程去拣。试分别**使用PV操作和管程方法**写出两进程P1和P2能并发正确执行的程序。**（满分16分，每小题8分）**

**答**：

1. PV操作:

Semaphore w1=1；w2=0;// 一个进程拣了一子，必须让另一个进程去拣

P1(){

while(1){

P(w2);

// 拣白子

V(w1);

}

}

P2(){

while(1){

P(w1);

// 拣黑子

V(w2);

}

}

1. 管程：

typedef pick=monitor{

semaphore w,b;

int w\_count,b\_count;

bool flag;

flag=true; // 确保交替拣棋子

w=0;

b=0;

w\_count = 0;

b\_count= 0;

InterfaceModule IM;

define w(),b();

use wait(),enter(),signal(),leave();

void b(){

enter(IM);

if(flag) wait(b,b\_count,IM);

flag=true;

// 拣黑子

signal(w,w\_count,IM);

leave(IM);

}

void w(){

enter(IM);

if(!flag) wait(w,w\_count,IM);

flag=false;

// 拣白子

signal(b,b\_count,IM);

leave(IM);

}

};

begin:

process black{

while(1){

pick.b();

}

}

process white{

while(1){

pick.w();

}

}

end

9、【PV、管程】一组生产者进程和一组消费者进程共享9个缓冲区，每个缓冲区可以存放一个整数。生产者进程每次一次性地向3个缓冲区中写入整数，消费者进程每次从缓冲区取出一个整数。请用：①信号量和PV操作；②管程，写出能够正确执行的程序。**（满分16分，每小题8分）**

**答**：

1. PV操作

Semaphore cache=9; // 可用缓冲区个数

cache\_list[9];

int num=0; // 使用了几个缓冲区

Semaphore produce=1;

Semaphore consume=1;

Semaphore mutex=1;

Producer(){

while(1){

P(produce);

if(num <=6){ // 已经使用的缓冲区的个小于等于6个，可以继续生产，否则阻塞

P(mutex);

int count=0;

// 生产3个产品

for(int i=0;i<9;i++){

if(count==3) break;

if(cache\_list[i]==0){

cache\_list[i]=1;

// 生产3个产品，即写入3个整数

count++;

num++;

}

}

V(mutex);

}

V(produce);

}

}

Consumer(){

while(1){

P(consume);

if(num==0){

V(consume);

return;

}

for(int i = 0;i<9;i++){

if(catch\_list[i] == 1){

//读出一个整数

catch\_list[i]=0;

num--;

break;

}

}

V(consume);

}

}

1. 管程：

typedef ps=monitor{

int num; // 已经使用了的缓冲区的个数

cache\_list[9];

semaphore produce,consume;

int p\_count,c\_count;

produce=0;

consume=0;

p\_count = 0;

c\_count= 0; num=0;

InterfaceModule IM;

define produce(),consume();

use wait(),enter(),signal(),leave();

void produce(){

enter(IM);

if(num>6) wait(produce,p\_count,IM); // 空闲缓冲区小于3个，不能继续生产，阻塞生产者进程

int count=0;

// 生产3个产品

for(int i=0;i<9;i++){ // 写入3个整数

if(count==3){

count=0;

break;

}

if(cache\_list[i]==0){

// 写入一个整数

cache\_list[i]=1;

count++;

num++;

}

}

if(num>0)

signal(consume,c\_count,IM); // 唤醒消费者进程

leave(IM);

}

void consume(){

enter(IM);

if(num==0) wait(consume,c\_count,IM); // 缓冲区中没有产品，需要阻塞消费者进程

for(int i=0;i<9;i++){

if(cache\_list[i]==1){

// 取出一个整数

cache\_list[i]=0;

num--;

break;

}

}

if(num<=6) signal(produce,p\_count,IM); // 可以继续生产

else if(num>6) signal(consume,c\_count,IM); // 缓冲区空闲个数小于3，不可以继续生产，需要消费

leave(IM);

}

};

begin:

process Producer{

while(1){

ps.produce();

}

}

process Consumer{

while(1){

ps.consume();

}

}

end10、【银行家算法】系统有A、B、C、D共4种资源，在某时刻进程P0、P1、P2、P3和P4对资源的占有和需求情况如表，试解答下列问题：**（满分8分，4+4）**

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Allocation | Claim | Available |
| A B C D | A B C D | A B C D |
| P0 | 0 0 3 2 | 0 0 4 4 | 1 6 2 2 |
| P1 | 1 0 0 0 | 2 7 5 0 |  |
| P2 | 1 3 5 4 | 3 6 10 10 |  |
| P3 | 0 3 3 2 | 0 9 8 4 |  |
| P4 | 0 0 1 4 | 0 6 6 10 |  |

(1)系统此时处于安全状态吗？试给出一个可能的安全序列。**（4分）**

(2)若此时进程P2发出request1(1, 2, 2, 2)，系统能分配资源给它吗？为什么？**（4分）**

**答：**

1. 安全，序列：P0-P3-P1-P4-P2

首先满足P0，执行完成归还后为1 6 5 4，然后满足P3，执行完成归还后是1 9 8 6，然后满足P1，执行完成归还后是2 9 8 6，然后满足P4，执行完成归还后是2 9 9 10，然后满足P2，执行完成归还后是3 12 14 14

1. P2 申请1 2 2 2后情况如表所示

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Allocation | Claim | Available |
| A B C D | A B C D | A B C D |
| P0 | 0 0 3 2 | 0 0 4 4 | 0 4 0 0 |
| P1 | 1 0 0 0 | 2 7 5 0 |  |
| P2 | 2 5 7 6 | 2 4 8 8 |  |
| P3 | 0 3 3 2 | 0 9 8 4 |  |
| P4 | 0 0 1 4 | 0 6 6 10 |  |

此时可使用的资源为0 4 0 0，不能满足任意进程的执行，即不存在安全序列，系统不处于安全状态，所以不能分配资源给它。