**操作系统2024春课后应用题作业2**

**姓名：马林耿 学号：221250123**

**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** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **总分** |
| 满分8分 | **满分4分** | **满分6分** | **满分6分** | **满分6分** | 满分6分 | 满分8分 | 满分16分 | 满分16 | 满分16 | 满分8 |
| **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**：**（8分，每小题2分）**

1. 先来先服务算法：

移动臂移动总量为：(143 - 86) + (147 - 86) + (147 - 91) + (177 - 91) + (177 - 94) + (150 - 94) + (150 - 102) + (175 - 102) + (175 - 102) + (175 - 130) = 57 + 61 + 56 + 86 + 83 + 56 + 48 + 73 + 45 = 565

移动臂顺序：143 --> 86 --> 147 --> 91 --> 177 --> 94 --> 150 --> 102 --> 175 --> 130

1. 最短查找时间优先算法：

移动臂移动的总量为:(150 - 143) + (150 - 86) + (177 - 86) = 162

移动臂移动的顺序为:143 --> 147 --> 150 --> 130 --> 102 --> 94 --> 91 --> 86 --> 175 --> 177

1. 双向扫描算法：

移动臂移动的总量为:(199 - 143) + (199 - 86) = 169

移动臂移动的顺序为:143 --> 147 --> 150 --> 175 --> 177 --> 199 --> 130 --> 102 --> 94 --> 91 --> 86

1. 电梯调度算法:

移动臂移动的总量为:(177 - 143) + (177 - 86) = 125

移动臂移动的顺序为:143 --> 147 --> 150 --> 175 --> 177 --> 130 --> 102 --> 94 --> 91 --> 86

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：（4分，每小题2分）**

1. 该磁盘共有扇区数为 10 \* 100 \* 16 = 16000 个

故位示图占用空间为 16000 / 8 = 2000 字节

1. 当空白文件目录数大于 2000 / 5 = 400 时,空白文件目录大于位示图

3、通常来说，操作系统显示的磁盘可用空间是数据区的大小，不包括数据块之前的预留区域。对于一个容量未4TB、块大小为4KB的硬盘，仅分配信息的区域就占用\_\_①\_\_\_MB。假设平均每16KB数据对应一个inode，每个inode大小为256B，这inode表大小为\_\_②\_\_，占整个硬盘的\_\_③\_\_%。

**答：（6分，每小题2分）**

1. (4TB / 4KB) \* 1bit = 230 / 8 Byte = 128 MB
2. (4TB - 128MB - x) / 16KB \* 256 B = x B

得出 x = 63.015GB ≈ 63GB

1. 63GB / 4TB = 1.54%
2. 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1024B大小，每个盘块中可存放256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？

**答**：**（6分，每小题3分）**

1. 直接块容量：1024B \* 10 / 1024 = 10KB

一级间接容量：256 \* 1024B / 1024 = 256KB

二级间接容量：256 \* 256KB = 65536KB = 64MB

三级间接容量：256 \* 65536KB = 16777216KB

直接盘块占用10个，一级间接盘块占用256个，二级间接块占用 (1024KB - 10KB - 256KB) \* 1024B/KB / 1024B = 758 个，因此占用 758 + 256 = 1014 个间接块。

1. 直接块占用10个，一级间接块占用256个，二级间接块占用（25 \* 1024KB - 10KB - 256KB)\*1024B/KB / 1024B = 25334个，因此占用25590个间接块。

5、【基本概念】**（6分，每小题3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：**

1. 所采用的信号量的初值不同。

对于情况1，信号量初始值为1,范围为[1-n, 1]。没有进程进入互斥段时，信号量为1。n - 1 个进程等待进入时，信号量为1 - n。

对于情况2，信号量初始值为m,范围为[m-n, m]。没有进程进入互斥段时，信号量为m。n - m 个进程等待进入时，信号量为m - n。

6、【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**：对语句进行编号

|  |  |
| --- | --- |
| P1() {   1. y=1; 2. y=y+3;   V(S1);   1. z=y+1;   P(S2);   1. y=z+y;   } | P2() {   1. x=1; 2. x=x+5;   P(S1);   1. x=x+y;   V(S2);   1. z=z+x;   } |

其中1, 2, 5, 6并行运行互不干扰，运行后 y = 4, x = 6。

下面考虑剩余语句的执行顺序，有多种情况：

1. 执行顺序3，7，4，8（或7，3，4，8）结果x = 10, y = 9, z = 15
2. 执行顺序3，7，8，4（或7，3，8，4）结果x = 10, y = 19, z = 15
3. 执行顺序7，8，3，4 结果 x = 10, y = 9, z = 5

**7、**【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) mod 4]，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）**

|  |
| --- |
| 1. semaphore mutex1, mutex2, mutex3, mutex0; // 限制同一信箱同时只能有一个进程访问 2. mutex1 = mutex2 = mutex3 = mutex0 = 1; 3. semaphore put1, put2, put3, put0; // 能否放入信箱 4. put1 = put2 = put3 = put0 = 0; 5. semaphore get1, get2, get3, get0; // 能否取出信箱 6. get1 = get2 = get3 = 0; get0 = 3; 7. int in1, in2, in3, in0, out1, out2, out3, out0; // 存取指针 8. in1 = in2 = in3 = in0 = out1 = out2 = out3 = out0 = 0; |

|  |  |
| --- | --- |
| 1. Process P0() { 2. **while** (**true**) { 3. P(get0); 4. P(mutex0); 5. {读取信箱消息 } 6. message = M0[out0]; 7. out0 = (out0 + 1) % 3; 8. V(mutex0); 9. V(put0); 11. {加工消息} 13. P(put1); 14. P(mutex1); 15. {写入信箱消息} 16. M1[in1] = message 17. in1 = (in1 + 1) % 3; 18. V(mutex1); 19. V(get1); 20. } 21. } | 1. Process P1() { 2. **while** (**true**) { 3. P(get1); 4. P(mutex1); 5. {读取信箱消息 } 6. message = M1[out1]; 7. out1 = (out1 + 1) % 3; 8. V(mutex1); 9. V(put1); 11. {加工消息} 13. P(put2); 14. P(mutex2); 15. {写入信箱消息} 16. M2[in2] = message 17. in2 = (in2 + 1) % 3; 18. V(mutex2); 19. V(get2); 20. } 21. } |
| 1. Process P2() { 2. **while** (**true**) { 3. P(get2); 4. P(mutex2); 5. {读取信箱消息 } 6. message = M2[out2]; 7. out2 = (out2 + 1) % 3; 8. V(mutex2); 9. V(put2); 11. {加工消息} 13. P(put3); 14. P(mutex3); 15. {写入信箱消息} 16. M3[in3] = message 17. in3 = (in3 + 1) % 3; 18. V(mutex3); 19. V(get3); 20. } 21. } | 1. Process P3() { 2. **while** (**true**) { 3. P(get3); 4. P(mutex3); 5. {读取信箱消息 } 6. message = M3[out3]; 7. out3 = (out3 + 1) % 3; 8. V(mutex3); 9. V(put3); 11. {加工消息} 13. P(put0); 14. P(mutex0); 15. {写入信箱消息} 16. M0[in0] = message 17. in0 = (in0 + 1) % 3; 18. V(mutex0); 19. V(get0); 20. } 21. } |

8、【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. 使用信号量和PV操作

|  |
| --- |
| 1. **struct** { 2. **char** readerName[20]; 3. **int** number; 4. } A[100] 5. **for** (**int** i = 0; i < 100; i++) { 6. A[i].number = i; 7. A[i].readerName = null; 8. } 9. semaphore mutex, seat; 10. mutex = 1; seat = 100; |
| 1. Process Reader(**char** name[20]) { 2. **while** (**true**) { 3. {进入阅览室} 4. P(seat); 5. P(mutex); 6. **for** (**int** i = 0; i < 100; i++) { 7. **if** (A[i].readerName == null) { 8. {读者得到座位i} 9. A[i].readerName = name; 10. **break**; 11. } 12. } 13. V(mutex); 14. {读者选择座位i,坐下读书} 15. P(mutex); 16. {读者离开座位i} 17. A[i].readerName = null; 18. V(mutex); 19. V(seat); 20. {离开阅览室} 21. } 22. } |

1. 使用管程

|  |  |
| --- | --- |
| 1. type readbook = MONITOR { 2. semaphore s; 3. **int** s\_count, i, seatCount; 4. **char** name[100]; 5. seatCount = 0; //已被占用的座位数 6. InterfaceModule IM; 7. DEFINE enterroom(), leaveroom(); 8. USE enter(), leave(), wait(), signal(); | |
| 1. **void** enterroom(**char**[] readername) { 2. enter(IM); 3. **if** (seatCount >= 100) 4. wait(s, s\_count, IM); 5. seatCount++; 6. **for** (**int** i = 0; i < 100; i++) { 7. **if** (name[i] == null) { 8. name[i] = readername; 9. **break**; 10. } 11. } 12. {获得座位}; 13. leave(IM); 14. } | 1. **void** leaveroom(**char**[] readername) { 2. enter(IM); 3. seatCount--; 4. **for** (**int** i = 0; i < 100; i++) { 5. **if** (name[i] == readername) { 6. name[i] = null; 7. **break**; 8. } 9. } 10. signal(s, s\_count, IM); 11. leave(IM); 12. } |
| 1. process readeri() { 2. {进入阅览室} 3. readbook.enterroom(readername); 4. {阅读}; 5. readbook.leaveroom(readername); 6. {离开阅览室}; 7. } | |

9、【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**：

使用PV操作：

|  |  |
| --- | --- |
| 1. semaphore s1, s2; 2. s1 = 1; s2 = 0; | |
| 1. Process P1() { 2. **while** (1) { 3. P(S1); 4. {拣白子} 5. V(S2); 6. } 7. } | 1. Process P2() { 2. **while** (1) { 3. P(S2); 4. {拣黑子} 5. V(S1); 6. } 7. } |

使用管程：

|  |  |
| --- | --- |
| 1. type pickup\_chess = MONITOR { 2. **bool** white\_turn = **true**; 3. semaphore s\_white, s\_black; 4. **int** s\_white\_count, s\_black\_count; 5. InterfaceModule IM; 6. DEFINE white(), black(); 7. USE enter(), leave(), wait(), signal(); | |
| 1. **void** white() { 2. enter(IM); 3. **if** (!white\_turn) 4. wait(s\_white, s\_white\_count, IM); 5. white\_turn = **false**; 6. {拣白子}; 7. signal(s\_black, s\_black\_count, IM); 8. leave(IM); 9. } | 1. **void** black() { 2. enter(IM); 3. **if** (white\_turn) 4. wait(s\_black, s\_black\_count, IM); 5. white\_turn = **true**; 6. {拣黑子}; 7. signal(s\_white, s\_white\_count, IM); 8. leave(IM); 9. } |
| 1. Process P1() { 2. pickup\_chess.white(); 3. others; 4. } | 1. Process P2() { 2. pickup\_chess.black(); 3. others; 4. } |

10、【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. 使用PV操作：

|  |  |
| --- | --- |
| 1. **int** buf[9]; 2. **int** count = 0, getptr = 0, putptr = 0; 3. semaphore empty = 9, full = 0, mutex = 1; | |
| 1. Process Producer() { 2. **while** (**true**) { 3. {生产3个整数}; 4. P(empty); 5. P(empty); 6. P(empty); 7. P(mutex); 8. buf[putptr] = num[0]; 9. putptr = (putptr + 1) % 9; 10. buf[putptr] = num[1]; 11. putptr = (putptr + 1) % 9; 12. buf[putptr] = num[2]; 13. putptr = (putptr + 1) % 9; 14. V(mutex); 15. V(full); 16. V(full); 17. V(full); 18. } 19. } | 1. Process Consumer() { 2. **while** (**true**) { 3. P(full); 4. P(mutex); 5. num = buf[getptr]; 6. getptr = (getptr + 1) % 9; 7. V(mutex); 8. V(empty); 9. } 10. } |

1. 使用管程：

|  |  |
| --- | --- |
| 1. type producer\_consumer = MONITOR { 2. **int** buf[9]; 3. **int** count, getptr, putptr; 4. count = 0; getptr = 0; putptr = 0; 5. semaphore put, get; 6. **int** put\_count, get\_count; 7. InterfaceModule IM; 8. DEFINE put(), get(); 9. USE enter(), leave(), wait(), signal(); | |
| 1. process put(**int** x1, **int** x2, **int** x3) { 2. enter(IM); 3. **if** (count > 6) 4. wait(put, put\_count, IM); 5. count += 3; 6. buf[putptr] = x1; 7. putptr = (putptr + 1) % 9; 8. buf[putptr] = x2; 9. putptr = (putptr + 1) % 9; 10. buf[putptr] = x3; 11. putptr = (putptr + 1) % 9; 12. signal(get, get\_count, IM); 13. signal(get, get\_count, IM); 14. signal(get, get\_count, IM); 15. leave(IM); 16. } | 1. process get() { 2. enter(IM); 3. **if** (count == 0) 4. wait(get, get\_count, IM); 5. y = buf[getptr]; 6. getptr = (getptr + 1) % 9; 7. count--; 8. **if** (count < 7) { 9. signal(put, put\_count, IM); 10. } **else** **if** (count > 0) { 11. signal(get, get\_count, IM); 12. } 13. leave(IM); 14. } |
| 1. process producer\_i() { 2. **while**(1) { 3. {生产3个整数}; 4. producer\_consumer.put(a1, a2, a3); 5. } 6. } | 1. process consumer\_j() { 2. **while**(1) { 3. y = producer\_consumer.get(); 4. {消费整数y}; 5. } 6. } |

11、【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. 此时系统处于安全状态，一安全序列为 P0 - P3 - P4 - P1 - P2。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 进程 | Available | Cik-Aik | Allocation | Available + Allocation | Possible |
| A B C D | A B C D | A B C D | A B C D |
| P0 | 1 6 2 2 | 0 0 1 2 | 0 0 3 2 | 1 6 5 4 | True |
| P3 | 1 6 5 4 | 0 6 5 2 | 0 3 3 2 | 1 9 8 6 | True |
| P4 | 1 9 8 6 | 0 6 5 6 | 0 0 1 4 | 1 9 9 10 | True |
| P1 | 1 9 9 10 | 1 7 5 0 | 1 0 0 0 | 2 9 9 10 | True |
| P2 | 2 9 9 10 | 2 3 5 6 | 1 3 5 4 | 3 12 14 14 | True |

1. 不能分配。假设分配给 P2,此时Available = (0, 4, 0, 0)，不能满足任何一个进程，此时系统处于不安全状态，因此不可分配。