北京交通大学 2012 年硕士研究生入学考试试卷

科目代码：923 科目名称：操作系统原理

注意事项：答案一律写在答题纸上，写在试卷上的答案不予装订和评分

一、 简题（本部分共 5 个小题，每个小题 10 分，共 50 分）

1. 方便性和有效性是设计操作系统的两个主要目标，以两种操作系统的技术为例，分别说明它们是如何实现这两个目标的（一个实现方便性的例子，一个实现有效性的例子）。

答题思路：

方便性：操作系统提供了良好的用户接口，用户按需要输入命令，操作系统按命令去控制程序的执行；用户也可以在程序中调用操作系统的功能模块完成相应服务，而不必了解硬件的物理特性。

操作系统的有效性包含两个方面的含义，即提高系统资源利用率，提高系统吞吐量。

[1] 有效地管理和分配硬件、软件资源，提高系统工作效率。

[2] 操作系统扩充硬件的功能，使硬件的功能发挥得更好.

[3] 操作系统使用户合理共享资源，防止各用户间的相互干扰.

[4] 操作系统合理地组织计算机的工作流程，使用户程序能顺利完成。

2. 主流微型计算机中分页存储系统中页面的大小通常设定为 1KB、2KB/4KB 等。如果页面大小设置为更大或者更小，会带来哪些好处和问题？

解析：我们都知道，页面大，页表小，节省页表空间，而且查找快，缺页中断发生的次数相对而言少一些。而且一次换页的时间长，页内碎片（属于内部碎片了）导致的浪费比较大。页表小的则刚好相反。

页表小了可以节省存储空间，减少内部碎片带来的浪费。进程调页的速度也比较快。等等。当然，王道有同学也给了一种说法。如下：

对于较大的页面，有以下的好处和问题：

[1] 可以节省地址映射的存储空间；

[2] 使虚拟存储器的实现变得更加简单

[3] 主存与辅存之间传送较大页面比传送较小页面更有效。

[4] 联想存储器的项数优先，对于同样的快表，较大页面意味着可搞笑实现更多存储空间的地址转换，减少快表失效次数。

较小页面的优点：是我们解法的一部分说法。

3. 如果用于进程同步的信号量的 P、V 操作不用原语实现，会产生什么后果？举例说明。

参考答案：看看我们 2010 年的真题里面，有这么一个：

两段程序分别如下（其中，R 是寄存器，counter 为共享变量而且初始值为 1）：

程序 A	程序 B
R=counter;	R=counter;
R=R+1;	R=R-1;
counter=R	counter=R

这两个程序就是一个例子，要是不用原语实现，会有 0,1,2 三个结果。即没有试下互斥，结果具有不可再现性。

4. 磁盘长期使用后，读写磁盘中数据的速度就会变慢，而执行磁盘碎片整理程序后，速度就提高了，为什么？

参考答案：

[1] 文件磁盘存储不连续。 计算机系统经过一段时间的使用后， 由于文件修改、删除、新建等操作混合在一起产生的积累效应，许多文件在磁盘上的存储变得不连续。导致读写磁盘中的数据的速度变慢。

[2] 有效提高文件系统性能。这时，使用“磁盘碎片整理程序”整理磁盘碎片，可以有效提高文件系统的性能。

[3] 磁盘是一个靠盘片的高速旋转进行读写的设备，当系统需要读写的数据连续存储在同一个磁道时，效率较高；当数据的存储不连续时，由于寻道开销，效率会有较大下降。用户在读写文件时，经常是顺序进行的（虽然系统支持随机存取）。另外，即使再随机存取的情况下，文件的连续存储也有利于磁盘缓存系统的工作。

5. 虚拟内存的容量可以比物理内存大得多，但是访问速度和物理内存相近，为什么？

分析：本题我们探索了很久，没有找到太好的答案。我们认为，解释成基于程序的局部性原理的预测，调入当前访问位置的附近区域内容。

因而，我们给出了一种解答方法。要是你的解答方法不同，请与我们一起讨论。呵呵。。。群体力量大，相信我们会有合理的大家认同的答案的。

虚拟存储器，是指具有请求调入功能和置换功能，能从逻辑上对内存容量加以扩充的一种存储器系统。并基于一定的调页算法，加上局部性原理等有效措施，使得即将要访问的内容以比较高的可能性提前被调入内存中，而无需等到要访问该内容了才去外存调取，从而减少时间开销。其逻辑容量由内存容量和外存容量之和所决定，其运行速度接近于内存速度，而每位的成本却又接近于外存。

二、 计算论证题（本部分共 5 个小题，每个小题 12 分，共 60 分）

1. 假设共有 5 个作业 J1、J2、J3、J4、J5，它们到达时刻分别是 0、2、2、5、6，服务时间分别是 3、10、5、6、2。采用高响应比优先调度算法，计算这些作业的平均周转时间和平均带权周转时间。

参考答案：

[1] 最开始达到的是 J1,此时系统的其他 5 个作业还没有到达，所以 J1 开始投入执行。在 3 时执行完毕。此时，J2 等待了 1 秒，J3 等待了 1 秒，J4、J5 还没有到达。响应比 =（等待时间+要求服务时间）/ 要求服务时间=1+等待时间/要求服务时间。J2 的响应比为 1+1/10=1.1。J3 的响应比 1+1/5=1.2。所以调度 J3 运行。

[2] J3 执行完毕，时间到了 8。J4、J5 此时都已经到达了。因而，剩下的 J2、J4、J5 的响应比如下：J2=1+6/10=1.6；J4=1+3/6=1.5；J5=1+2/2=2。所以，J5 被调度执行。

[3] J5 执行完毕，是在 10. 此时系统中仅剩两个作业 J2、J4。J2、J4 的响应比分别是 J2=1+8/10=1.8；J4=1+5/6=1.81。 所以，调度 J4 执行。16 时执行完 J4，就轮到 J2 执行了。J2 在 26 时执行完毕。

综上，执行序列是 J1、J3、J5、J4、J2。

周转时间是作业从提交到完成的时间。

作业	J1	J2	J3	J4	J5
周转时间	3	24	6	11	4

平均周转时间=（3+24+6+11+4）/5=48/5=9.6

带权周转时间=周转时间/服务时间

所以平均带权周转时间=（3/3 + 24/10 +6/5 +11/6 +4/2）/5=1.72

2. 某文件系统采用索引物理结构存放文件，磁盘空间为 1000GB。一个目录项可以存储 10 个盘块的地址，前 9 个为直接地址，最后一个为一级间址。若盘块的大小为 512B，则该文件系统最大能支持的文件大小是多少？

参考答案：注意，磁盘是外存，不是内存，别混淆了。

一个目录项可以存储 10 个盘块，9 个直接地址支持的文件长度是$9 \times 512B = 4.5KB$。而一个一级间址。题目指出，一个目录项可以存放 10 个盘块的地址，所以，一级间址可以存放 10 个目录项的地址。故而支持的最大文件长度为 $512B\times 9 + 10 \times 512B=19\times 512B = 9.5KB$

3. 在请求分段存储系统中，每个段空间最大为 16KB。假定某个时段该用户的段表如下图所示。

段号	起始地址	段长(B)
0	5K	3K
1	9K	600
2	12K	10K
3	—	—

试问：

(1) 逻辑地址 0xA92B 对应的物理地址是多少？

解析这类题目，先看看段号（或请求分页系统的题目的页号）是否在内存中，若不在内存中，那么就是段故障了。若是在内存中，在看看段内地址（或者请求分页系统的题目对应的页内地址）有没有越界，要是越界了，发出越界中断。

解析：转成二进制 1010 1001 0010 1011 ，前面两位表示段号，后面 14 位表示段内地址。那么，可以知道段号是 10B，也就是段号 2。段号 2 的起始地址为 12K，大小为 10KB。

看看段内地址：1010 1001 0010 1011 。而第 2 段的最大段内地址是 10K，利用快速写法写出 10K。就是 2 进制的 0010 10 00 0000 0000。可以看看，逻辑地址 0xA92B 的段内地址是 1010 1001 0010 1011。可以看到该地址的段内地址部分（下划线部分）显然大于 10K。所以，越界了。

(2) 逻辑地址 0x071F 对应的物理地址是多少？

解析：转换成 2 进制的 0000 0111 0001 1111。对应的段号是 0，起始地址是 5K。接着看看块内地址部分有没有越界。0000 0111 0001 1111 。可以发现，段内地址 1K~2K 之间。没有越界。转换成物理地址，

计算一下如下：

0000 0111 0001 1111+ 0001 0100 0000 0000=0001 1011 0001 1111，换回 16 进制的 0x1B1F

(3) 逻辑地址 0xE068 对应的物理地址是多少？

解析：转换成 2 进制的 1110 0000 0110 1000。对应的段号是 3。但是段号是 3 的段并没有在内存中，所以发生了缺段中断了。

4. 有一个请求页式虚拟存储系统。如果分配给某进程 3 个内存物理块，开始时内存中预装入第 1、2、3 个页面，该进程的页面访问序列如下：

1、2、4、2、6、2、1、5、6、1

(1) 若采用最佳页面置换算法，缺页率为多少？

解析：回顾一下最佳置换算法的概念，最佳置换算法是由 Belady 于 1966 年提出的一种理论上的算法。其所选择的被淘汰页面，将是以后永不使用的，或许是在最长(未来)时间内不再被访问的页面。

页面访问序列	1	2	4	2	6	2	1	5	6	1
内存页面 1	1	1	1	1	1	1	1	1	1	1
内存页面 2		2	2	2	2	2	2	5	5	5
内存页面 3			4	4	6	6	6	6	6	6
缺页	√	√	√		√			√		

可以看到，发生了 5 次缺页。缺页率为 50%。

(2) 若采用最近最少使用的页面置换算法 LRU，缺页率为多少？

解析：最近最少使用也叫做最近最久未使用算法。注意 LRU 和 FIFO 是常考的页面置换算法，一定得会，要不然可能就悲剧了！

页面访问序列	1	2	4	2	6	2	1	5	6	1
内存页面 1	1	1	1	1	6	6	6	5	5	5
内存页面 2		2	2	2	2	2	2	2	6	6
内存页面 3			4	4	4	4	1	1	1	1

缺页	√	√	√		√		√	√	√	

可以看到，发生了 7 次缺页，缺页率为 70%。

5. 有 3 个进程 P1、P2、P3 总共需要资源<R1,R2>的最大数目依次为<7,5>、<3,3>、<4,4>(均是互斥共享资源)。在 t 时刻 P1、P2、P3 正占用的资源<R1,R2>数目依次是<1,2>、<1,1>、<1,0>；系统中资源 R1 和 R2 的数量还各自剩余 3 个和 2 个；而且，进程 P1 正在申请 4 个资源 R1，2 个资源 R2；P2 正在申请 1 个资源 R1，1 个资源 R2；P3 正在申请 1 个资源 R1，2 个资源 R2。

   问：

   (1) 此刻系统是否处于安全状态？为什么？

   参考答案：

进程与资源情况	最大需求		已获得的资源数		need	
	A	B	A	B	A	B
P1	7	5	1	2	6	3
P2	3	3	1	1	2	2
P3	4	4	1	0	3	4

   系统可用的 R1、R2 类资源=（3，2）。P2 可以调度执行。执行完毕，系统可用的 R1、R2 类资源=（4，3）。Need(P1)=(6,3),Need(P3)=(3,4)。系统已经无法继续执行了。所以系统并不安全。

   (2) 此刻系统是否已经发生死锁？为什么？

   解析：t 时刻虽然不安全，但是 P2 还是可以执行的。说明死锁还没有发生。这也说明了一个问题，不安全的状态不一定死锁。但是，死锁了，却一定是不安全状态。

三、 算法题（本部分共三个大题，第 1、2 大题分别 14 分，第三大题 10 分，共计 40 分）

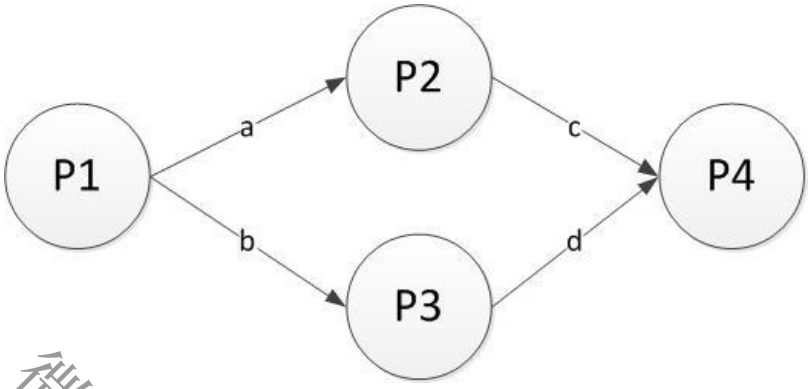
1. 四个进程 P1 必须在 P2、P3 开始之前完成，P2、P3 必须在 P4 开始之前完成。而且 P2、P3 不能并发执行。

   试写出这四个进程的同步互斥算法。（15 分）

   参考答案前趋图是一个常考的知识点，要会啊！交大好几年的真题里面都有。连我本科上课的作业

也有，其实我本科没有布置过几次作业。

我们需要建立 4 个信号量 a,b,c,d，然后一个供 P2、P3 互斥的信号量 mutex。所以画了下面的前趋图，两个进程之间的小箭头上的变量表示信号量。




```
Var a,b,c,d: semaphore:=0,0,0,0;
/*
因为这几个信号量为 1 将唤醒其所在的箭头后面的进程执行，而唤醒操作是由其所在的箭头前面的进程执行完了之后进行的。所以，初值设为 0。
*/
mutex:semaphore:=1;//别搞成 0 了，要不然，两个进程都进行不下去了。互斥信号量一般初值为 1
begin
    parbegin
        P1:signal(a);signal(b);end;
        /*
        P1 执行完了，通知 P2、P3,告诉它们，它们可以开始了。
        */
        begin wait(a);wait(mutex);P2;signal(c);signal(mutex);end;
        /*
        P2 接到 P1 可以开始的通知，再捕获互斥信号量 mutex，就具备开始运行的条件了。执行完 P2，释放互斥信号量 mutex,并通知 P4，我这边完成了。
        */
        begin wait(b);wait(mutex);P3;signal(d);signal(mutex);end;
        /*
```

P3 接到 P1 可以开始的通知，再捕获互斥信号量 mutex，就具备开始运行的条件了。执行完 P3，释放互斥信号量 mutex,并通知 P4，我这边完成了。

*/

begin wait(c); wait(d); P5; end;

/*

P4 等候 P2、P3 的“准许”，接到 P2、P3 先后发出的“我执行完了，你可以开始了”的通知，具备开始运行的条件了。执行完 P4。进程序列就执行完毕了。

*/

parend

end

【注意】我注释成这样，是为了方便你们理解，你若是考研了也注释成这样，老师就傻眼了。你出了考场，告诉我你也注释成这样，那我就罪大了。所以，不要注释成这样，费时，老师也没时间看！

[思考题] 若是上面的程序中，

wait(a);wait(mutex);P3;signal(c);signal(mutex);end;

与

wait(b);wait(mutex);P3;signal(d);signal(mutex);end;

中的 wait(a)和 wait(b)都放到 wait(mutex)的后面，这样行吗？为什么？

2. Alice 有一个私人邮箱 mbox，最多可以存放 N 封邮件。Alice 可以从该邮箱中读取邮件，且每读一封邮件该邮件就自动删除；其他用户都可以向该邮箱发送邮件。请设计 Alice 读邮件和其他用户发邮件操作的同步互斥算法。（15 分）

参考答案：嗨，大家好，又轮到海哥的 PV 算法了。呵呵…

关于这类题型，我们就不吐槽了。就是读者写者问题的改编。所以，大家要特别注意读者写者问题了。道理明白越深，对我们越有好处。考研要是来一个，10 多分就有了。最多可以存放 N 封邮件，很像我们读者写者那样阅览室最多可以装 N 个人。

关于这个题目，开始读的时候，总觉得有点不清不楚的。也没有说互斥和同步在哪儿体现了。所以，我们补上了一句：Alice 读邮件的时候，其他写者不让写。其他写者给 Alice 发邮件的时候，Alice 不能读。就是互斥访问邮箱。当然，我们还定义了一下，比如说，一个人在给 Alice 发邮件的时候，其他人不能给

Alice 发邮件！

于是，从《期末考试题目和解析》第二章去找 Dtab 复制粘贴的干活：

解析：Var mutex,empty,mailcount:semaphore:=1,N,0;

/*注意了，不要定义成下面这个！！！！

Writecount:integer:=0;

想象一下，若是我们这里也 P（Writecount）或者 V（Writecount）,写法对吗？

呵呵…当然是不对的。PV 操作都是针对信号量的。一个整型的变量也 PV，这习惯不好啊！

*/

```
Begin
  Parbegin
    Reader:Begin     //发邮件
      repeat
        wait(empty);
       /*
       给 Alice 发邮件之前，先看看人家的邮箱是不是满了。要是满了就得等，不然就白忙活了。
       */
        wait(mutex) ;
         /*
        等待信号量，每个时刻，最多只能有一个人操作邮箱（我们假设是这样）
        */
        写邮件，写完了，给 Alice 发过去！
         Signal(mutex);//释放互斥信号
       Signal(mailcount);//来信件了，Alice 的邮箱里多一封信
      until false;
    End

  Reader:
   Begin//Alice 读邮件了
    repeat
```

wait(mailcount);//有邮件，那么就申请互斥信号量，去读邮件

wait(mutex);

/*

申请互斥信号量。注意，要是这一句和上面的 wait(mailcount)换一个位置，假如系统开始没有邮件，但是 Alice 申请到了 mutex 信号量。那么想给 Alice 发邮件就费劲了。弄到 empty 信号量很容易，但是死都弄不到互斥信号量啊！

*/

读邮件，（读完系统自动删除）

signal(empty)；

/*

系统删除邮件了，是不是要释放 empty 信号量啊？呵呵…想想要是系统删除邮件，没有 signal(empty)，那么 Alice 最多只能收到 N 封邮件，之后给 Alice 发邮件的人就再也申请不到 empty 信号量了。所以，这里要 signal(empty)，千万别忘了。

*/

signal(mutex) //写完了，释放写信号，若有人想写信给 Alice ,可以竞争 mutex 信号量，等待机会写信了。

//其他写者可写

until false;

End

Parend

End

【思考题】呵呵…试着想想，要是可以同时有多个发给 Alice 信件的人，又该怎么写？【提示，这就很像期末考试 2004-2005 年的 Dtab 那个题了。呵呵…很经典，要把握啊！】

3. 两个进程 P1、P2 并发执行，并用信号量 M1、M2 分别实现对两个互斥共享资源 R1 和 R2 的互斥访问。这两个进程以什么次序执行会导致死锁？在不影响程序功能的情况下，请改写算法防止死锁，同时尽可能保持较高的资源利用率。（10 分）

程序如下：

Var M1,M2:Semaphore=1,1;

```
begin
  parbegin
    P1 begin:                          P2 begin:
      wait(M1);                          wait(M2);
      access R1;                         access R2;
      wait(M2);                          wait(M1);
      access R1 and R2;                  access R1;
      signal(M1);                        signal(M1);
      signal(M2);                        signal(M2);
      end                                end
  parend
```

参考答案：

[1] 很明显，若进程 P1 执行 wait(M1)还没有 wait(M2)成功的时候，P2 若 wait(M2)成功，就死锁了。P1 在等待 M2，P2 在等待 M1。

[2] 先让两个进程去争夺 M1，抢到 M1 的进程才给它抢 M2，这种方法解除了死锁，也保证了 P1、P2 两个进程谁先抢到 M1，都可以顺利得到 M2 而投入运行。目的就达到了。呵呵…仰天大笑出门去，我辈岂是聪明人！笨的人习惯了，引以为豪啊！

```
Var M1,M2:Semaphore=1,1;
begin
  parbegin
    P1 begin:                          P2 begin:
      wait(M1);                          wait(M1);
      access R1;                         wait(M2);
      wait(M2);                          access R2;
      access R1 and R2;                  access R1;
      signal(M1);                        signal(M1);
      signal(M2);                        signal(M2);
      end                                end
```