第一章

（1）操作系统是计算机系统中的一个系统软件，它是一组程序模块的集合，这组程序模块控制和管理计算机系统中的硬件和软件资源，合理地组织计算机工作流程，并为用户使用计算机提供方便。从资源管理的角度看，操作系统应具有处理机管理、存储器管理、设备管理及文件管理功能。

（2）操作系统的三种基本类型是批处理系统、分时系统和实时系统。批处理系统的主要特点是：用户脱机使用计算机、多道程序运行和成批处理。分时系统的主要特点是：同时性、交互性、独立性和及时性。实时系统的主要特点是及时性和高可靠性。

（3）多道程序设计是指在内存中同时存放道程序，这些程序在管理程序的控制下交替运行，共享处理机及系统中的其他资源。多道程序设计技术的主要特点是：多道、宏观上并行、微观上串行。

第二章

（1）进程是一个具有一定功能的程序关于某个数据集合的一次运行活动。一个进程最少有就绪、执行和阻塞三种状态。就绪状态是指进程已获得了除处理机以外的所有资源，一旦获得处理机就可以立即执行。执行状态是指进程获得必要的资源并正在处理机上执行。阻塞状态是指进程由于发生某事件而暂时无法执行下去，此时即使把处理机分配给该进程，它也无法运行。

（3）从这四条语句的变量引用情况看，语句S1和S2可以并发执行，语句S3应在S1及S2执行完成后进行，语句S4应在S3执行完成后进行。其前趋图如图1所示。

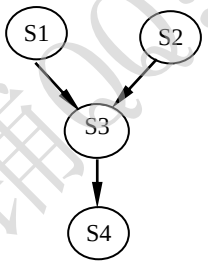

图 1　四条语句的前趋图

（5）进程与线程的主要区别如下：

• 调度方面：传统操作系统中，进程是拥有资源和独立调度的基本单位。在引入线程的操作系统中，线程是独立调度的基本单位，进程是资源拥有的基本单位。在同一进程中，线程的切换不会引起进程切换。在不同的进程中进行线程切换，将会引起进程切换。

• 拥有资源：不论是传统操作系统还是设有线程的操作系统，进程都是拥有资源的基本单位，而线程不拥有系统资源（也有一点必不可少的资源），但线程可以访问其隶属进程的系统资源。

• 并发性：在引入线程的操作系统中，不仅进程之间可以并发执行，而且同一进程内的多个线程之间也可并发执行。

• 系统开销：由于创建进程或撤消进程时，系统都要为之分配或回收资源，操作系统所付出的开销远大于创建或撤消线程时的开销。在进行进程切换时，涉及到整个当前进程CPU环境的保存及新调度到进程的CPU环境的设置；而线程切换时，只需保存和设置少量寄存器

内容，因此开销很小。另外，由于同一进程内的多个线程共享进程的地址空间，因此，多线程之间的同步与通信非常容易实现，甚至无须操作系统的干预。

第三章

（1）一次只能给一个进程访问的资源称为临界资源。进程中访问临界资源的那段代码称为临界区。访问临界资源时应遵循空闲让进、忙则等待、有限等待及让权等待。

（3）参见课件 ppt

（6）在本题中应设置五个信号量b2、b3、b4、b5、b6分别表示进程P2、P3、P4、P5、P6是否可以开始执行，其初值均为0。这六个进程的同步机制描述如下：

```
semaphore   b2=0;
semaphore   b3=0;
semaphore   b4=0;
semaphore   b5=0;
semaphore   b6=0;
main()
{
     cobegin
        P1();
        P2();
        P3();
        P4();
        P5();
        P6();
     coend
}
P1()
{
    ┆          /*“┆”表示进程中的语句，下同*/
    v(b2);
    v(b3);
}
P2()
{
   p(b2);
     ┆
   v(b4);
 }
P3()
{
   p(b3);
     ┆
   v(b5);
}
P4()
{
    p(b4);
      ┆
    v(b6);
  }
```

```
P5()
{
   p(b5);
     ┆
   v(b6);
  }

P6()
{
   p(b6);
   p(b6);
     ┆
}
```

（8）在本题中，应设置四个信号量mutex、empty、full1、full2。mutex用于实现对缓冲区的互斥访问，其初值为1；empty表示缓冲区中可用单元数目，其初值为N；full1表示已读入字符个数，其初值为0；full2表示已处理字符个数，其初值为0。为了描述方便起见，还应设置三个指针in、out1、out2，in指向下一个可用缓冲单元，out1指向下一个待处理字符，out2指向下一个待输出字符。它们并发执行的同步机制描述如下：

```
semaphore  empty=N;
semaphore  full1=0;
semaphore  full2=0;
semaphore  mutex=1;
char buffer[N];
int in=0,out1=0,out2=0;
main()
{
   cobegin
     R();
     M();
     P();
   coend
}
R()
{
   while(true)
     {
       char x;
       读入一个字符到x;
       p(empty);
       p(mutex);
       buffer[in]=x;
       in=(in+1)% N;
       v(mutex);
       v(full1);
     }
 }
M()
{
    char x;
    while(true)
      {
```

```
            p(full1);
            p(mutex);
            x=buffer[out1];
            if (x==" ")
               {
                   x=",";
                   buffer[out1]=x;
               }
            out1=(out1+1)% N;
            v(mutex);
            v(full2);
         }
    }
    P()
    {
    char x;
       while(true)
         {
            p(full2);
            p(mutex);
            x=buffer[out2];
            out2=(out2+1)% N;
            v(mutex);
            v(empty);
            输出字符x;
         }
    }
```

第四章

（1）产生死锁的必要条件是互斥条件、不剥夺条件、请求和保持条件、循环等待条件。互斥条件是指进程要求对所分配的资源进行排他性控制，即在一段时间内某资源仅为一个进程所占有。不剥夺条件是指进程所获得的资源在未使用完毕之前，不能被其他进程强行夺走，即只能由获得该资源的进程自己来释放。请求和保持条件是指进程每次申请它所需要的一部分资源，在等待分配新资源的同时，进程继续占有已分配到的资源。循环等待条件是指存在一种进程资源的循环等待链，链中的每一个进程已获得的资源同时被链中下一个进程所请求。

解决死锁问题常用的方法有忽略死锁、预防死锁、避免死锁、检测及解除死锁。预防死锁方法通过设置某些限制条件，去破坏产生死锁的四个必要条件中的一个或几个，来防止发生死锁。避免死锁方法在资源的动态分配过程中，用某种方法防止系统进入不安全状态，从而避免死锁。检测死锁方法通过系统的检测机构及时地检测出死锁的发生，并确定与死锁有关的进程，解除死锁方法用于将进程从死锁状态下解脱出来，检测死锁方法一般与解除死锁方法联合使用。

（5）

1）不会发生死锁。因为系统中只有3个进程，每个进程的最大资源需求量为2，且系统中资源总数为4，无论资源怎样分配，系统至少能够满足1个进程的最大资源需求量，该进程将顺利运行完毕，从而可以将分配给它的资源归还给系统，使其他进程也能顺利执行完成，故不

会出现死锁。

2）可能会发生死锁。因为系统中有 2 个进程，每个进程的最大资源需求量为 4，且系统中资源总数为 6，若系统将其中的 4 个资源分配给其中的一个进程，则该进程将顺利运行完毕，从而可将分配给它的资源归还给系统，使另一个进程也能顺利执行完成，以这种方式分配资源时不会发生死锁；若系统为每个进程分配 3 个资源，在此情况下，每个进程均获得了部分资源且系统中已没有空闲资源，当其中的一个进程再次申请资源时，因系统中无空闲资源而等待，另一个进程的情况类似，因此以这种方式分配资源会产生死锁。

（6）响应比高者优先调度算法就是在每次调度作业运行时，先计算后备作业队列中每个作业的响应比，然后挑选响应比最高者投入运行。

响应比=1+等待时间/运行时间

在8.0时，因为只有作业J1到达，系统将作业J1投入运行。作业J1运行2小时后（即10.0）完成。此时，其他三个作业均已到达，三个作业的响应比为：

作业J2的响应比=1+(10.0－8.6)/0.6=3.33
作业J3的响应比=1+(10.0－8.8)/0.2=7
作业J4的响应比=1+(10.0－9.0)/0.5=3

从计算结果看，作业J3的响应比最高，所以让作业J3先运行。作业J3运行0.2小时后（即10.2）完成，此时作业J2和作业J4的响应比为：

作业J2的响应比=1+(10.2−8.6)/0.6=3.67
作业J4的响应比=1+(10.2−9.0)/0.5=3.4

从上述计算结果看，作业J2的响应比较高，所以让作业J2先运行。

因此四个作业的执行次序为：作业J1、作业J3、作业J2、作业J4。各作业的周转时间如表2所示：

表2　响应比高者优先算法的性能

作业	到达时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
J1	8.0	2.0	8.0	10.0	2.0	1.0
J3	8.8	0.2	10.0	10.2	1.4	7.0
J2	8.6	0.6	10.2	10.8	2.2	3.67
J4	9.0	0.5	11.8	11.3	2.3	4.6

作业的平均周转时间为：(2.0+1.4+2.2+2.3)/4=1.975

作业的平均带权周转时间为：(1.0+7.0+3.67+4.6)/4=4.07

（9）

1)利用银行家算法对此时刻的资源分配情况进行分析，可得表3所示的安全性分析情况：

表3　安全性检测表

资源 / 进程	工作向量	还需要的资源	已分配资源	工作向量+已分配资源	完成情况
	r1 r2 r3	r1 r2 r3	r1 r2 r3	r1 r2 r3	

D	1	2	0	0	1	0	1	0	1	2	2	1	true
A	2	2	1	1	0	0	3	1	1	5	3	2	true
E	5	3	2	2	1	0	0	0	0	5	3	2	true
B	5	3	2	0	1	2	0	0	0	5	3	2	true
C	5	3	2	3	0	0	1	1	0	6	4	2	true

从上述情况分析中可以看出，此时存在一个安全序列{D，A，E，B，C}，故该状态是安全的。

（2）进程B提出请求$Request_B$(0，1，0)，按银行家算法进行检查：

- $Request_B$(0，1，0)≤$Need_B$(0，1，2)
- $Request_B$(0，1，0)≤Available(1，2，0)
- 试分配并修改相应数据结构，资源分配情况如表4所示.

表4　B申请资源后的资源分配表

资源 / 进程	已分配资源			还需要的资源			剩余资源		
	r1	r2	r3	r1	r2	r3	r1	r2	r3
A	3	1	1	1	0	0	1	1	0
B	0	1	0	0	0	2			
C	1	1	0	3	0	0			
D	1	0	1	0	1	0			
E	0	0	0	2	1	0			

- 再利用安全性算法检查系统是否安全，可得表5所示的安全性分析情况。

表5　安全性检测表

资源情况 / 进程	工作向量			还需要的资源			已分配资源			工作向量+已分配资源			完成情况
	r1	r2	r3	r1	r2	r3	r1	r2	r3	r1	r2	r3	
D	1	1	0	0	1	0	1	0	1	2	1	1	true
A	2	1	1	1	0	0	3	1	1	5	2	2	true
E	5	2	2	2	1	0	0	0	0	5	2	2	true
B	5	2	2	0	0	2	0	1	0	5	3	2	true
C	5	3	2	3	0	0	1	1	0	6	4	2	true

从上述情况分析中可以看出，此时存在一个安全序列{D，A，E，B，C}，故该状态是安全的，系统可以将资源分配给进程B。

（3）进程E提出请求$Request_E$(0，1，0)，按银行家算法进行检查：

- $Request_E$(0，1，0)≤$Need_E$(2，1，0)
- $Request_E$(0，1，0)≤Available(1，1，0)
- 试分配并修改相应数据结构，资源分配情况如表6所示。

表6　E申请资源后的资源分配表

资源 / 进程	已分配资源			还需要的资源			剩余资源		
	r1	r2	r3	r1	r2	r3	r1	r2	r3

A	3	1	1	1	0	0	1	0	0
B	0	1	0	0	0	2			
C	1	1	0	3	0	0			
D	1	0	1	0	1	0			
E	0	1	0	2	0	0			

- 再利用安全性算法检查系统是否安全，可得表7所示的安全性分析情况。

表7　安全性检测表

资源 / 进程	工作向量			还需要的资源			已分配资源			工作向量+已分配资源			完成情况
	r1	r2	r3	r1	r2	r3	r1	r2	r3	r1	r2	r3	
A	1	0	0	1	0	0	3	1	1	4	1	1	true
D	4	1	1	0	1	0	1	0	1	5	1	2	true
E	5	1	2	2	0	0	0	1	0	5	2	2	true
B	5	2	2	0	0	2	0	1	0	5	3	2	true
C	5	3	2	3	0	0	1	1	0	6	4	2	true

从上述情况分析中可以看出，此时存在一个安全序列{ A，D，E，B，C}，故该状态是安全的，系统可以将资源分配给进程E。

第五章

（2）在存储管理中，内零头是指分配给作业的存储空间中未被利用的部分，外零头是指系统中无法利用的小存储块。

在固定式分区分配中，为将一个用户作业装入内存，内存分配程序从系统分区表中找出一个能满足作业要求的空闲分区分配给作业，由于一个作业的大小并不一定与分区大小相等，因此，分区中有一部分存储空间被浪费掉了。由此可知，固定式分区分配中存在内零头。

在可变式分区分配中，为把一个作业装入内存，应按照一定的分配算法从系统中找出一个能满足作业需求的空闲分区分配给作业，如果这个空闲分区的容量比作业申请的空间容量要大，则将该分区一分为二，一部分分配给作业，剩下的一部分仍然留作系统的空闲分区。由此可知，可变式分区分配中存在外零头。

在页式虚拟存储系统中，用户作业的地址空间被划分成若干大小相等的页面，存储空间也分成与页大小相等的物理块，但一般情况下，作业的大小不可能都是物理块大小的整数倍，因此作业的最后一页中仍有部分空间被浪费掉了。由此可知，页式虚拟存储系统中存在内零头。

在段式虚拟存储系统中，作业的地址空间由若干个逻辑分段组成，每段分配一个连续的内存区，但各段之间不要求连续，其内存的分配方式类似于动态分区分配。由此可知，段式虚拟存储系统中存在外零头。

（4）所谓地址重定位，就是当一个程序装入到与其地址空间不一致的存储空间时而进行的地址调整过程，即将地址空间给出的逻辑地址映射到内存的物理地址上。地址重定位有静态重定位和动态重定位两种方式，前者的重定位是在程序装入内存时完成的，后者的重定位是在程序执行过程中每次访问信息时完成的。

采用内存分区管理时，可在硬件上设置一个“重定位寄存器”来实现程序运行时的动态重定位。此时，地址重定位是在程序执行期间由地址变换机构动态实现的，即物理地址等于逻辑地址加上重定位寄存器的内容。

（5）

第六章

（1）每页大小为 1024 字节，8 页的逻辑空间大小为 8K 字节，故逻辑地址的有效位为 13；在页式系统中，块与页大小相等，32 块的物理存储区大小为 32K 字节，故物理地址至少为 15 位。

（2）采用LRU页面置换算法的页面置换情况如表8所示。

表8　采用LRU页面置换算法的页面置换情况

页面走向	3	4	2	1	4	3	1	4	3	1	4	5
块1	3	3	3	1		1						1
块2		4	4	4		4						4
块3			2	2		3						5
缺页	缺	缺	缺	缺		缺						缺

缺页中断次数为6，缺页率为6/12=50%。

（3）

1）每块的长度为64KB/16=4KB，因页面0、1、2、3分别装入主存的2、4、1、6块中，所以该作业每一页在主存中的起始地址如下：

第0页在主存中的起始地址：4KB×2=8KB=8192B

第1页在主存中的起始地址：4KB×4=16KB=16384B

第2页在主存中的起始地址：4KB×1=4KB=4096B

第3页在主存中的起始地址：4KB×6=24KB=24576B

2）逻辑地址[0，100]的内存地址为：4KB×2+100=8292B

逻辑地址[1，50]的内存地址为：4KB×4+50=16434B

逻辑地址[2，0]的内存地址为：4KB×1+0=4096B

逻辑地址[3，60]的内存地址为：4KB×6+60=24636B

（4）对于逻辑地址[0，65]，因65＜200，其逻辑地址合法，对应的主存地址为600+65=665。

对于逻辑地址[1，55]，因55＞50，其逻辑地址非法，产生地址越界中断。

对于逻辑地址[2，90]，因90＜100，其逻辑地址合法，对应的主存地址为1000+90=1090。

对于逻辑地址[3，20]，因20＜150，其逻辑地址合法，但该段状态位为1，所以产生缺段中断，待缺段调入主存后再进行地址变换。

第七章

（2）设备驱动程序是直接控制设备进行运转的程序。设备驱动程序的功能是接收来自上层

的与设备无关软件的抽象请求，将这些请求转换成设备控制器可以接受的具体命令，再将这些命令发送给设备控制器，并监督这些命令是否正确执行。

（4）DMA 控制方式与中断控制方式的主要区别是：中断控制方式在每个数据传送完成后中断 CPU，而 DMA 控制方式则是在所要求传送的一批数据全部传送结束时中断 CPU；中断控制方式的数据传送是在中断处理时由 CPU 控制完成，而 DMA 控制方式则是在 DMA 控制器的控制下完成。

第八章

（2）把磁头从当前位置移动到指定磁头位置的操作过程叫寻道。访问磁盘时间由寻道时间、旋转延迟时间和传输时间组成。寻道时间是磁盘调度的主要目标，因为磁头是机械移动的，所以寻道时间比其他两个时间长得多，是影响磁盘调度的主要因素。

（5）打开文件操作的功能是将待访问文件的目录信息读入内存活动文件表中，建立起用户和文件的联系。关闭文件操作的功能是撤消主存中有关该文件的目录信息，切断用户与该文件的联系；若在文件打开期间，该文件作过某种修改，则应将其写回辅存。

（8）从 100 磁道开始，采用先来先服务磁盘调度算法进行调度的情况如表 9 所示。

表 9 采用先来先服务算法的调度情况

下一磁道	移动磁道数
23	77
376	353
205	171
132	73
19	113
61	42
190	129
398	208
29	369
4	25
18	14
40	22

移动磁道数总数为1596，平均寻道长度为133。

•从 100 磁道开始，采用最短寻道时间优先磁盘调度算法进行调度的情况如表 10 所示。

表 10 采用最短寻道时间优先算法的调度情况

下一磁道	移动磁道数
132	32
190	58
205	15
61	144
40	21
29	11
23	6
19	4

18	1
4	14
376	372
398	22

移动磁道数总数为700，平均寻道长度为58.3。

• 从 100 磁道开始且磁头向磁道号增加方向移动，采用扫描算法进行调度的情况如表 7.7 所示。

表 11 采用扫描算法的调度情况

下一磁道	移动磁道数
132	32
190	58
205	15
376	171
398	22
61	337
40	21
29	11
23	6
19	4
18	1
4	14

移动磁道数总数为692，平均寻道长度为57.7。