第一章

1、操作系统的两大主要作用是什么？

为应用程序提供一个资源集的清晰抽象（另一种说法：操作系统给用户提供了一个可扩展的机器。个人理解是通过对底层的抽象，对外提供各种接口支持扩展）；管理各种软硬件资源。

3. 分时系统和多道程序系统的区别是什么？

在分时系统中，多个用户可以使用他们自己的终端同时访问和执行计算系统上的计算。 多道程序设计系统允许用户同时运行多个程序。 所有分时系统都是多道程序设计系统，但并非所有多道程序设计系统都是分时系统，因为多道程序设计系统可以在只有一个用户的 PC 上运行（分时系统肯定是多道程序系统，多道程序系统不一定是分时系统）

10. 内核态和用户态有哪些区别？解释在设计操作系统时存在两种不同的模式有什么帮助。

大多数现代 CPU 提供两种执行模式：内核态和用户态。 CPU 可以执行其指令集中的每条指令，并在内核态下执行时使用硬件的各种功能。 但是用户态只能执行部分指令，执行时仅使用部分功能。 拥有两种模式允许设计人员以用户态运行用户程序，从而拒绝他们访问关键指令。

12、下面哪一条指令只能在内核态使用？

- a 禁止所有的中断
- b 读日期-时间时钟
- c 设置日期-时间时钟
- d 改变存储器映像

答案：a,c,d

13、考虑一个有两个 CPU 的系统，且每一个 CPU 有两个线程（超线程）。假设有三个程序 P0、P1\P2，分别以运行时间 5ms, 10ms,20ms 开始，运行这些程序需要多少时间？假设这三个程序都是 100%限于 CPU，在运行时无阻赛，并且一旦设定就不改变 CPU。

完成这些程序的执行可能需要 20,25 或 30 毫秒，具体取决于操作系统如何安排它们。如果 P0 和 P1 在同一个 CPU 上进行调度，而 P2 在另一个 CPU 上进行调度，则需要 20 毫秒。 如果 P0 和 P2 安排在同一个 CPU 上并且 P1 安排在另一个 CPU 上，则需要 25 毫秒。 如果 P1 和 P2 安排在同一个 CPU 上并且 P0 安排在另一个 CPU 上，则需要 30 毫秒。 如果所有三个都在同一个 CPU 上，则需要 35 毫秒。

14、一台计算机有一个四级流水线，每一级都花费相同的时间执行其工作，即1ns，这台机器每秒可执行多少条指令？

答：每一纳秒的指令都从管道中出现。 这意味着机器每秒执行 10 亿条指令。 根本没关系管道有多少个阶段。 每级 1 nsec 的 10 级流水线每秒也会执行 10 亿条指令。 重要的是完成的指令弹出管道末端的频率。

17.什么是陷阱指令？在操作系统中他的用途。

陷阱指令将一个处理器的执行模式从用户模式切换到内核模式。该指令允许用户程序调用操作系统内核中的函数。

21.下列资源能使用哪种多路复用（时间、空间或者两者皆可）：CPU、内存、磁盘、网卡、打印机、键盘以及显示器？

时间复用：CPU，网卡，打印机，键盘。
空间复用：内存，磁盘。
两者：显示。

第二章

5、一个计算机系统的内存有足够的空间容纳 5 个程序。这些程序又一半时间处于等待 I/O 的空闲状态。请问 CPU 时间浪费的比例是多少？

五个进程都空闲的概率是 1/32，所以 CPU 空闲时间是 1/32。

6、一个计算机的 RAM 有 4GB，其中操作系统占 512M。所有进程都占 256M 并且特征相同。要是 CPU 利用率达到 99%, 最大 I/O 等待是多少？
内存中有足够的空间容纳 14 个进程。如果一个进程的 I/O 是 P，那么它们都在等待 I/O 的概率是 Pˆ14。通过将其等于 0.01，我们得到方程 pˆ 14=0.01。解决这个问题，我们得到 p=0。72，因此我们可以容忍高达 72%I/O 等待的进程

7、多个作业能够并行运行，比它们顺序执行完成的要快。假设有两个作业同时开始执行，每个需要 20 分钟的 CPU 时间。如果顺序执行，那么最后一个作业需要多长时间可以完成？如果并行执行又需要多长时间？假设 I/O 等待占 50%。

CPU 利用率计算公式：CPU 利用率 = 1 - pˆn 。设运行作业所需要的时间为 T。

如果每个工作都有 50%的 I/O 等待，那么在没有竞争的情况下，完成该工作需要 40 分钟。如果按顺序运行，第二个将在第一个启动后 80 分钟完成。对于两个作业，CPU 利用率大约为 1-0.5^2。因此，每一个系统每分钟可获得 0.375 cpu 分钟的实时数据。要累积 20 分钟的 CPU 时间，作业必须运行 20/0.375 分钟，或大约 53.33 分钟。因此，按顺序运行的作业在 80 分钟后完成，但并行运行的作业在 53.33 分钟后完成。

8、考虑一个 6 级多道程序系统（内存可同时容纳 6 个程序）。假设每个进程 I/O 等待时间 40%，那么 CPU 利用率是多少？

所有进程等待 I/O 的概率为 0.4^6，即 0.004096。因此，CPU 利用率=1-0。004096=0:995904。

12、在图 2-8 中，给出了一个多线程 Web 服务器。如果读取文件的惟一途径是正常的阻塞 read 系统调用，那么 Web 服务器应该使用用户级线程还是内核级线程？为什么？

当工作者线程从磁盘读取 Web 页时，它就会被阻塞。如果使用用户级线程，该动作将阻塞整个进程，而破坏多线程的价值。这就是使用内核线程的原因：某些线程的阻塞不会影响到其他线程。

14、既然计算机中只有一套寄存器，为什么图 2-12 中寄存器集合按每个线程中的内容列出而不是按每个进程中的内容列出？

当一个线程停止时，它在寄存器中有值。它们必须被保存，就像进程停止时，必须保存寄存器。多线程和多进程没有什么不同，所以每个线程需要自己的寄存器保存区。（线程空间里，虽然地址共享，但是却拥有着完全独立的寄存器，这是为了线程能够独立的去完成一些工作，如果寄存器也共享，那么线程的存在就没有意义了）

15、线程可以被时钟中断抢占吗？如果可以，在什么情况下可以？如果不可以，为什么不可以？

用户级线程不能被时钟抢占，除非整个进程

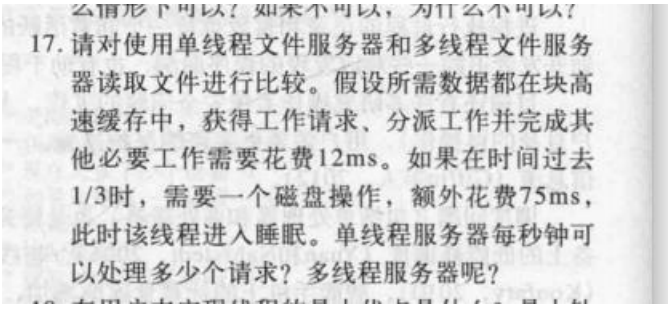
量子已经用完了（尽管透明的时钟中断可能会发生）。

内核级线程可以单独抢占。在后一种情况下，如果

线程运行太长，时钟将中断当前进程，因此

当前线程。内核可以自由地从同一进程中选择不同的线程，以便在需要时运行下一个进程

（这题有点问题）


17. 请对使用单线程文件服务器和多线程文件服务器读取文件进行比较。假设所需数据都在块高速缓存中，获得工作请求、分派工作并完成其他必要工作需要花费12ms。如果在时间过去1/3时，需要一个磁盘操作，额外花费75ms，此时该线程进入睡眠。单线程服务器每秒钟可以处理多少个请求？多线程服务器呢？

在单线程情况下，cache 命中需 15ms，cache 未命中需要 90ms。其加权平均为 2/315+1/390。因此，平均请求为 40ms，而服务器每秒可处理 25 个。对于多线程服务器，所有磁盘等待都是重叠的，因此每个请求都耗时 15ms，而服务器每秒可处理 66.6666 个请求。

18.在用户空间实现线程，其最大的优点是什么？最大的缺点是什么？

最大的优势就是效率。不需要陷入内核来切换线程。最大的缺点是，如果一个线程阻塞，整个进程都会阻塞。

26、在 2.3.4 节中，描述了一种有高优先级进程 H 和低优先级进程 L 的情况，导致了 H 陷入死循环。若采用轮转调度算法而不是优先级调度算法，还会发生这样问题吗？请给予讨论。

答：在轮转调度算法下。L 迟早会运行，最终它将会离开临界区。关键是，在优先级调度算法下，L 永远不会运行；循环，它定期得到一个正常的时间片，所以有机会离开其临界区。

29、将生产者-消费者问题扩展成一个多生产者-多消费者的问题，生成（消费）者都写（读）一个共享的缓冲区，每个生产者消费者都在自己的线程中执行。图 2-28 中使用信号量的解法在这个系统中还可行吗？

是的，它会按原样运作。 在给定的时刻，只有一个生产者（消费者）可以向缓冲区添加（移除）一个项目。

30. 考虑对于两个进程P0和P1的互斥问题的解决方案。假设变量初始值为0。P0的代码如下：

```
/* Other code*/
while(turn !=0){}/*Do nothing and wait/*/
Critical Section/*...*/
turn=0;
/* Other code*/
```

P1的代码是将上述代码中的0替换为1。该方法是否能处理互斥问题中所有可能的情形?

答：该解决方案满足互斥，因为两个过程都不可能在其关键部分。 也就是说，当 turn 为 0 时，P0 可以执行其临界区，但不能执行 P1。 同样，当 turn 为 1.但是，这假定 P0 必须先运行。 如果 P1 产生某些东西并将其放入缓冲区，那么当 P0 可以进入其临界区时，它会发现缓冲区为空并阻塞。 而且，该解决方案需要严格交替两个过程，这是不希望的。

32、请说明仅通过二元信号量和普通机器指令如何实现计数信号量（即可以保持一个任意值得信号量）

将每个计数信号量与 2 个二值信号量联合：M 用于互斥；S 用于阻塞。另外，每个计数信号量都组合一个用于保存 up 次数减去 down 次数的计数器，以及在该信号量上阻塞的进程列表。为了实现 down 操作，进程首先通过对 M 执行 down 操作，以获得对信号量、计数器以及列表的独占访问权。然后，将计数器减 1。如果大于等于 0, 只需对 M 执行 up 操作并退出即可。如果 M 为负数，就将该进程放入阻塞进程列表中。接着，对 M 执行 up 操作，对 S 执行 down 操作来阻塞该进程。为了实现 up 操作，首先对 M 执行 down 操作以获得互斥，然后将计数器加 1。如果计数器大于 0, 则没有进程阻塞，就只需对 M 执行 up 操作。不过，如果计数器小于等于 0, 则必须从列表中移出某些进程。最后，按次序对 B 和 M 执行 up 操作。

Wait Operation:

```
wait(M);
counter--;
if(counter < 0)
{
    signal(M);
    wait(B);
}
signal(M);
```

（这是对于计数信号量的 p 操作，counter--代表可用空位减一，如果有五个空位，有五个进程来使用这些空位，那么这五个进程都需要执行这个 p 操作，因为此时 counter 都是大于 0 的，所以这些进程不会阻塞，正常执行，但是万一出现了第六个进程，那么 counter 就会变成-1，此时该进程将阻塞，并等待一个唤醒操作。

Signal Operation:

```
wait(M);
counter++;
if(counter <= 0)
    signal(B);
else
    signal(M);
```

（这是对于计数信号量的 v 操作，counter++用于空位加一，这个操作用在那些占用了该空间进程中，写在已经结束操作并释放了这块资源之后。Counter++，如果大于 0，说明原本的 counter 是大于等于 0 的，代表没有进程被挂在阻塞队列上，那么就不需要去做唤醒操作，如果万一发现 counter 是小于等于 0 的，那么代表有进程被阻塞了，需要执行唤醒操作。

34、如果线程在内核中实现，可以使用内核信号量对同一个进程中的两个线程进行同步吗？如果线程在用户空间实现呢？假设在其他进程中没有线程必须访问该信号量。请讨论你的答案。

答：对于内核线程，线程可以在信号量上阻塞，而内核可以运行该进程中的其它线程。因而，使用信号量没有问题。而对于用户级线程，当某个线程在信号量上阻塞时，内核将认为整个进程都被阻塞，而且不再执行它。因此，进程失败。

36、一个快餐店有四类雇员：(1) 领班，接收顿客点的菜单；（2）厨师，准备饭菜；（3）打包工，将饭菜装在袋子里；（4) 收银员，将食品袋交给顾客并收钱。每个雇员可被看作一个进行通信的顺序进程。它们采用的进程间通信方式是什么？请将这个模型与 UNIX 中进程联系起来。

答：雇员之间通过消息传递进行通信：在该例中，消息为订单、食物和袋子。在 UNIX 中，该 4 个进程通过管道连接。

37、假设有一个使用信箱的消息传递系统，当向满信箱发消息或从空信箱收消息时，进程都不会阻塞，相反，会得到一个错误代码。进程响应错误代码的处理方式为一遍一遍地重试，直到成功为止。这种方式会导致竞争条件吗？

答：它不会导致竞争条件（不会丢失任何东西），不过它是完全的忙等待。

42. 请说明在 Round-robin 调度算法中时间片长度和上下文切换时间是怎么互相影响的。

时间量=时间片长度

如果上下文切换时间很大，那么时间量值必须是成比例的大。否则，上下文切换的开销可能是相当高。如果典型的 CPU 突发时间小于时间量，选择大的时间量值可能导致系统效率低下。如果上下文切换时间很小或可以忽略不计，则时间量子值可以更自由地选择。

44、有 5 个待运行作业，估计它们的运行时 N 分别是 9, 6, 3, 5 和 X。采用哪种次序运行这些作业将得到最短的平均响应时间？（答案将依赖于 X。）

答：最短作业优先可以使得平均响应时间最短。

0 < X ≤ 3: X, 3, 5, 6, 9.

3 < X ≤ 5: 3, X, 5, 6, 9.

5 < X ≤ 6: 3, 5, X, 6, 9.

6 < X ≤ 9: 3, 5, 6, X, 9.

X > 9: 3, 5,6, 9, X.

45、有 5 个批处理作业 A 到 E，它们几乎同时到达一个计算中心。估计它们的运行时间分别为 10，6，2，4 和 8 分钟。其优先级（由外部设定）分别为 3，5，2，1 和 4，其中 5 为最高优先级。对于下列每种调度算法，计算其平均进程周转时间，可忽略进程切换的开销。

a）轮转法。

b）优先级调度。

c）先来先服务（按照 10，6，2，4，8 次序运行）。

d）最短作业优先。

对 a)，假设系统具有多道程序处理能力，每个作业均公平共享 CPU 时间，对 b)到 d），假设任一时刻只有一个作业运行，直到结束。所有的作业都完全是 CPU 密集型作业。

答：对于时间片轮转，在头 10 分钟里，每个作业获得 1/5 的 CPU 时间。在第 10 分钟时，C 结束。在接下来的 8 分钟里，每个作业获得 1/4 的 CPU 时间，然后 D 完成，然后，在接下来的 6 分钟内，余下的 3 个作业各获得 1/3 的 CPU 时间，直到 B 结束，以此类推。因此，5 个作业的完成时间分别为是 10，18，24，28 和 30，平均为 22 分钟。对于优先级调度，B 最先运行，6 分钟完成。其它作业分别在第 14，24，26 和 30 分钟完成，平均为 20 分钟。如果作业按 A->E 的次序执行，则分别在第 10,16，18，22 和 30 分钟完成，因此，平均为 19.2 分钟。最后，最短作业优先调度的完成时间分别为第 2，6，12，20 和 30 分钟，平均为 14 分钟。

46、运行在 CTSS 上的一个进程需要 30 个时间片完成。该进程必须被调入多少次，包括第一次(在该进程运行之前）？

答：第一次得到 1 个时间片。随后获得 2,4,8 和 15 个时间片，因此必须经过 5 次交换。

49、用 a = 1/2 的老化算法用来预测运行时间。先前的四次运行，从最老的一个到最近的一个，其运行时间分别是 40ms，20ms，40ms 和 15ms。下一次的预测时间是多少？

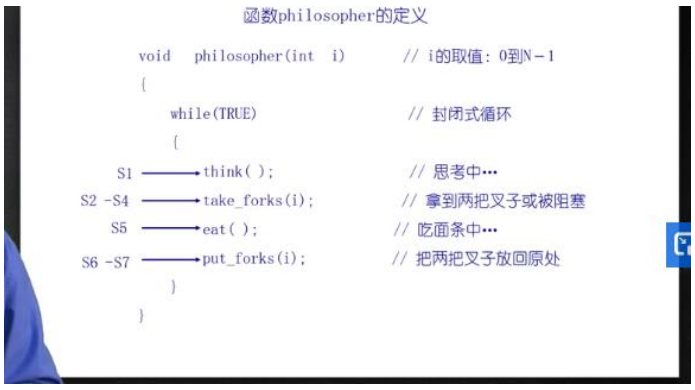
答：预测值的顺序为 40，30，35，所以下一次是 25。（把前两个数据加一起，除以 2，然后把结果和下一个数据加一起，除以 2。）

51、在哲学家就餐问题中使用如下规则：编号为偶数的哲学家先拿他左边的叉子，再拿他右边的叉子；编号为奇数的哲学家先拿他右边的叉子再拿他左边的叉子。这条规则能否避免死锁？

可以避免。 总会有至少一个空闲的叉子和至少一个可以同时获得两个叉子的哲学家。 因此，不会有僵局。 您可以尝试 N = 2，N = 3 和 N = 4，然后进行推广。

54、在哲学家就餐问题的解法 （图 2-46）中，为什么在过程 take_forks 中将状态变量置为 HUNGRY?

答：如果某个哲学家阻塞，其邻居稍后能够在 test 中检测其状态，发现他已经饥饿，当叉子可用时，就可以唤醒他了。




```
void  take_forks(int  i)              // i的取值：0到N－1
{
    P(mutex);                          // 进入临界区
    state[i] = HUNGRY;                 // 我饿了！
    test_take_left_right_forks(i);     // 试图拿两把叉子
    V(mutex);                          // 退出临界区
    P(s[i]);                           // 没有叉子便阻塞
}
```

```
void  test_take_left_right_forks(int  i)    //i：0到N－1
{
    if(state[i]       ==  HUNGRY &&         // i：我自己，or 其他人
       state[LEFT]    !=  EATING &&
       state[RIGHT]   !=  EATING)
    {
       state[i] = EATING;                   // 两把叉子到手
       V(s[i]);                             //通知第i人可以吃饭了
    }
}
```

```
void  put_forks(int  i)               // i的取值：0到N－1
{
    P(mutex);                          // 进入临界区
    state[i] = THINKING;               // 交出两把叉子
    test_take_left_right_forks(LEFT);  // 看左邻居能否进餐
    test_take_left_right_forks(RIGHT); // 看右邻居能否进餐
    V(mutex);                          // 退出临界区
}
```

第三章

17. 假设一个机器有38位的虚拟地址和32位的物理地址。
(a) 与一级页表比较，多级页表的主要优点是什么？
(b) 若采用二级页表，页面大小为16KB，每个页表项为4字节，应该对第一级页表域分配多少位？对第二级页表域分配多少位？请解释原因。

考虑一下，
（a）多级页表减少了由于其层次结构而需要在内存中的页表的实际页数。事实上，在一个具有大量指令和数据位置的程序中，我们只需要顶层页表（一页）、一个指令页和一个数据页。

（b）为三页字段中的每个字段分配 12 位。“偏移”字段需要
14 位到地址 16KB。它为页字段留下 24 位。因为每个
条目为 4 字节，一页可容纳 2 个 12 页的表条目，因此-
需要 12 位来索引一页。所以为每个页面分配 12 位
字段将寻址所有 2 个 38 字节。

19. 一个32位地址的计算机使用两级页表。虚拟地址被分成9位的第一级页表域、11位的二级页表域和一个偏移量，页面大小是多少？在地址空间中一共有多少个页面？

20 位用于虚拟页码，剩下 12 位用于偏移。这将生成一个 4-kb 的页面。虚拟页的 20 位意味着 2 个 20 页。

20. 一个计算机使用32位的虚拟地址，4KB大小的页面。程序和数据都位于最低的页面(0~4095)，栈位于最高的页面。如果使用传统（一级）分页，页表中需要多少个表项？如果使用两级分页，每部分有10位，需要多少个页表项？

1 对于一级页面表，需要 2 32/2 12 或 1 百万页。因此，页表必须有 1 百万个条目。对于两级分页，主页表有 1K 个条目，每个条目都指向第二页表。只使用其中两个。因此，总共只需要三个页表条目，一个在顶层表中，一个在底层表中。

.

24. 一台机器有48位虚拟地址和32位物理地址，页面大小是8KB，如果采用一级线性页表，页表中需要多少个表项？

有 8kb 的页面和 48 位的虚拟地址空间，虚拟页面的数量为 248/213，即 235（约 340 亿）。

27. 假设虚拟页码索引流中有一些重复的页索引序列，该序列之后有时会是一个随机的页码索引。例如，序列0，1，…，511，431，0，1，…，511，332，0，1，…中就包含了0，1，…，511的重复，以及跟随在它们之后的随机页码索引431和332。
(a) 在工作负载比该序列短的情况下，标准的页面置换算法（LRU，FIFO，Clock）在处理换页时为什么效果不好？
(b) 如果一个程序分配了500个页框，请描述一个效果优于LRU、FIFO或Clock算法的页面置换方法。

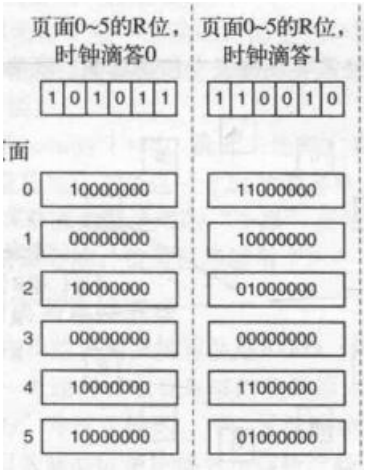
a）除非页面帧数为 512，即整个序列的长度，否则每个引用都会出现页面错误。（b）如果有 500 帧，将第 0-498 页映射到固定帧，只改变一帧。

29. 考虑图3-15 b中的页面序列。假设从页面B到页面A的R位分别是11011011。使用第二次机会算法，被移走的是哪个页面？

在本例中，将选择 0 位的第一页（D 被移走）

30. 一台小计算机有4个页框。在第一个时钟周期时R位是0111（页面0是0，其他页面是1），在随后的时钟周期中这个值是1011、1010、1101、0010、1010、1100、0001。如果使用带有8位计数器的老化算法，给出最后一个时钟周期后4个计数器的值。

Page 0: 0110110
Page 1: 01001001
Page 2: 00110111
Page 3: 10001011



31. 请给出一个页面访问序列，使得对于这个访问序列，使用Clock算法和LRU算法得到的第一个被置换的页面不同。假设一个进程分配了3个页框，访问串中的页号属于集合0，1，2，3。

顺序：0，1，2，1，2，0，3。在 LRU 中，第 1 页将替换为第 3 页。在时钟中，将替换第 1 页，因为所有页面都将被标记，并且光标位于第 0 页。

（时钟调度一开始令访问位都是 1，然后访问已有页面时，指针不动，被访问的那些页面如果本来就是 1，那就保持不变，如果是 0，就变成 1。访问没有的页面时，从当前指针开始转动，找到第一个是 0 的置换，并且把扫过的是 1 的页面全部变成 0，调入新页面后，将其置为 1，并且指针指向下一个位置。

36. 一个计算机有4个页框，载入时间、最近一次访问时间和每个页的R位和M位如下所示（时间以一个时钟周期为单位）：

页面	载入时间	最近一次访问时间	R	M
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0
3	110	285	1	1

(a) NRU算法将置换哪个页面？
(b) FIFO算法将置换哪个页面？
(c) LRU算法将置换哪个页面？
(d) 第二次机会算法将置换哪个页面？

NRU 删除第 2 页。FIFO 删除第 3 页。LRU 删除第 1 页。第二次机会删除第 2 页。

（NRU 算法就是根据访问时间往前找，找到第一个访问位是 0 的，直接置换掉。）

41. 一台计算机为每个进程提供65536字节的地址空间，这个地址空间被划分为多个4KB的页面。一个特定的程序有32768字节的正文、16386字节的数据和15870字节的堆栈。这个程序能装入这个机器的地址空间吗？如果页面大小是512字节，能放得下吗？记住，一个页面不能同时包含两个或者更多的不同种类段，例如，一页里不能同时有代码段和数据段。

文本为 8 页，数据为 5 页，堆栈为 4 页。程序不适合，因为它需要 17*4096 字节的页面。对于 512 字节的页面，情况不同。这里的文本是 64 页，数据是 33 页，堆栈是 31 页，共有 128*512 字节的页，这是合适的。对于较小的页面大小，可以，但对于较大的页面则不行。（较大的页面会导致页面空间浪费，因为不同类型的数据不能放一起，页面一大就会导致有些页面不会存满。而小的页面就不一样，可以减少浪费。）

47. 一个程序中有两个段，段0中为指令，段1中为读/写数据。段0有读/执行保护，段1有读/写保护。内存是按需分页式虚拟内存系统，它的虚拟地址为4位页号，10位偏移量。页表和保护如下所示（表中的数字均为十进制）：

段0		段1	
读/执行		读/写	
虚拟页号	页框号	虚拟页号	页框号
0	2	0	在磁盘
1	在磁盘	1	14
2	11	2	9
3	5	3	6
4	在磁盘	4	在磁盘
5	在磁盘	5	13
6	4	6	8
7	3	7	12

对于下面的每种情形，给出动态地址所对应的实（实际）内存地址，或者指出发生了哪种失效（缺页中断或保护错误）。

(a) 读取页：段1，页1，偏移3；
(b) 存储页：段0，页0，偏移16；
(c) 读取页：段1，页4，偏移28；
(d) 跳转到：段1，页3，偏移32。

47. Here are the results:

	Address	Fault?
(a)	(14, 3)	No (or 0xD3 or 1110 0011)
(b)	NA	Protection fault: Write to read/execute segment
(c)	NA	Page fault
(d)	NA	Protection fault: Jump to read/write segment

第四章

1. 给出文件/etc/passwd的五种不同的路径名。（提示：考虑目录项“.”和“..”。）

1、使用 . 或 …切换文件路径

```
/etc/passwd
/./etc/passwd
/./../etc/passwd
/./../../etc/passwd
/etc/../etc/passwd
/etc/../etc/../etc/passwd
```

/etc/../etc/../etc/../etc/passwd
/etc/../etc/../etc/../etc/../etc/passwd

2. 在Windows中，当用户双击资源管理器中列出的一个文件时，就会运行一个程序，并以这个文件作为参数。操作系统需要知道运行的是哪个程序，请给出两种不同的方法。

Windows 方法是使用文件扩展名。每个扩展名对应一个文件类型和一些处理该类型的程序。另一种方法是记住哪个程序创建了文件并运行该程序。麦金托什就是这样工作的。（一种是使用扩展名，另一种是不使用扩展名，但是直接由操作系统去记忆了哪个程序创建，并且要运行这个文件）

6. 某一些操作系统提供系统调用rename给文件重命名，同样也可以通过把文件复制到新文件并删除原文件而实现文件重命名。请问这两种方法有何不同？

对。重命名调用不会更改创建时间或上次修改时间，但创建新文件会使其获取当前时间作为创建时间和上次修改时间。另外，如果磁盘快满了，复制可能会失败。

8. 有一个简单操作系统只支持单一目录结构，但是允许该目录中有任意多个文件，且带有任意长度的名字。这样可以模拟层次文件系统吗？如何进行？

使用文件名，如/usr/ast/file。虽然它看起来像一个分层路径名，但实际上它只是一个包含嵌入斜杠的单一名称。

9. 在UNIX和Windows中，通过使用一个特殊的系统调用把文件的“当前位置”指针移到指定字节，从而实现了随机访问。请提出一个不使用该系统调用完成随机存取的替代方案。

一种方法是向 read 系统调用添加一个额外的参数，该参数指示要从哪个地址读取。实际上，每次读取都有可能在文件中执行查找。这种方案的缺点是（1）在每次读取调用中都会增加一个参数，（2）要求用户跟踪文件指针的位置。

10. 考虑图4-8中的目录树，如果当前工作目录是/usr/jim，则相对路径名为../ast/x的文件的绝对路径名是什么？

dotdot 组件将 search 移动到/usr，因此…/ast 将其放入/usr/ast。因此…/ast/x 与/usr/ast/x 相同。

12. 描述一个损坏的数据块对以下三种形式的文件的影响：(a) 连续的，(b) 链表的，(c) 索引的。

如果一个数据块在一个连续的分配系统中损坏，那么只有这个块会受到影响；文件的其余块可以被读取。在链接分配的情况下，无法读取损坏的块；而且，从该损坏块开始的所有块的位置数据都将丢失。在索引分配的情况下，只影响损坏的数据块。

13. 一种在磁盘上连续分配并且可以避免空洞的方案是，每次删除一个文件后就紧缩一下磁盘。由于所有的文件都是连续的，复制文件时需要寻道和旋转延迟以便读取文件，然后全速传送。在写回文件时要做同样的工作。假设寻道时间为5ms，旋转延迟为4 ms，传送速率为8MB/s，而文件平均长度是8 KB，把一个文件读入内存并写回到磁盘上的一个新位置需要多长时间？运用这些数字，计算紧缩16GB磁盘的一半需要多长时间？

启动传输需要 9 毫秒。要以 80MB/秒的传输速率读取 2^13 字节，需要 0.0977 毫秒，总共需要 9.0977 毫秒。再写回去需要 9.0977 毫秒。因此，复制文件需要 18.1954 毫秒。要压缩 16 GB 磁盘的一半，需要复制 8 GB 的存储空间，即 2^20 个文件。在每个文件 18.1954 毫秒时，这需要 19079.25 秒，即 5.3 小时。显然，每次删除文件后压缩磁盘不是一个好主意。

15. 某些数字消费设备需要存储数据，比如存放文件等。给出一个现代设备的名字，该设备需要文件存储，并且适合连续的空间分配。

数码照相机在非易失性存储介质（如闪存）上按顺序记录一些照片。当相机复位时，介质被清空。此后，图像按顺序一次记录一张，直到介质满为止，此时它们被上载到硬盘。对于此应用程序，相机内部（例如，在图片存储介质上）的连续文件系统是理想的。

16. 考虑图4-13中的i节点。如果它含有用4个字节表示的10个直接地址，而且所有的磁盘块大小是1024KB，那么文件最大可能有多大？

间接块可以容纳 128 个磁盘地址。与 10 个直接磁盘地址一起，最大文件有 138 个块。因为每个块都是 1 KB，所以最大的文件是 138 KB。

17. 一个班的学生信息存储在一个文件中，这些记录可以被随意访问和更新。假设每个学生的记录大小都相同，那么在连续的、链表的和表格/索引的这三种分配方式中，哪种方式最合适？

对于随机访问，表/索引和连续都是合适的，而链接分配并不是因为它通常需要对给定记录进行多个磁盘读取。（只有链表是要从头开始一个一个读取的）

18. 考虑一个大小始终在4KB和4MB之间变化的文件，连续的、链表的和表格/索引的这三种分配方式中，哪个方式最合适？

由于文件大小变化很大，连续分配将效率低下，需要重新分配磁盘空间，因为随着文件大小的增加，可用块的大小和压缩将随着文件大小的缩小而增加。链接和表/索引分配都将是有效的；在这两者之间，表/索引分配对于随机访问场景将更加有效。

24. 空闲磁盘空间可用空闲块表或位图来跟踪。假设磁盘地址需要D位，一个磁盘有B个块，其中有F个空闲。在什么条件下，空闲块表采用的空间少于位图？设D为16位，请计算空闲磁盘空间的百分比。

位图需要 B 位。自由列表需要 df 位。如果 d f<b，则空闲列表需要较少的位。或者，如果 f/b<1/d，则空闲列表较短，其中 f/b 是空闲块的分数。对于 16 位磁盘地址，如果可用磁盘的 6%或更少，则可用列表较短。

25. 一个空闲块位图开始时和磁盘分区首次初始化类似，比如：1000 0000 0000 0000 （首块被根目录使用），系统总是从最小编号的盘块开始寻找空闲块，所以在有6块的文件A写入之后，该位图为 1111 1110 0000 0000。请说明在完成如下每一个附加动作之后位图的状态：
(a) 写入有5块的文件B。
(b) 删除文件A。
(c) 写入有8块的文件C。
(d) 删除文件B。

25、位图的开头如下：

(a) After writing file B: 1111 1111 1111 0000
(b) After deleting file A: 1000 0001 1111 0000
(c) After writing file C: 1111 1111 1111 1100
(d) After deleting file B: 1111 1110 0000 1100

35. 考虑一个有10个数据块的磁盘，这些数据块从块14到23。有两个文件在这个磁盘上：f1和f2。这个目录结构显示f1和f2的第一个数据块分别为22和16。给定FAT表项如下，哪些数据块被分配给f1和f2？
(14,18); (15,17); (16,23); (17,21); (18,20); (19,15); (20,−1); (21,−1); (22,19); (23,14)
在上面的符号中，(x, y)表示存储在表项x中的值指向数据块y。

35、分配给 F1 的区块有：22、19、15、17、21。
分配给 f2 的区块有：16、23、14、18、20。

38. 给定磁盘块大小为4KB，块指针地址值为4字节，使用10个直接地址（direct address）和一个间接块（indirect block）可以访问的最大文件大小是多少字节？

38、间接块可以容纳 1024 个地址。在 10 个直接地址中，总共有 1034 个地址。由于每个指向一个 4kb 的磁盘块，所以最大的文件是 4235264 字节。

40. 一个UNIX系统使用4KB磁盘块和4字节磁盘地址。如果每个i节点中有10个直接表项以及一个一次间接块、一个二次间接块和一个三次间接块，那么文件最大为多大？

40、i 节点可容纳 10 个指针。单个间接块包含 1024 个指针。双间接块适用于 1024^2 个指针。三个间接块适用于 1024^3 个指针。加上这些，我们得到的最大文件大小为 1074791434 块，大约是 16.06 GB。

第五章

11. 以下各项工作是在四个 I/O 软件层的哪一层完成的？

a) 为一个磁盘读操作计算磁道、扇区、磁头。

b) 向设备寄存器写命令。

c) 检查用户是否允许使用设备。

d) 将二进制整数转换成 ASCII 码以便打印。

答：

（a）设备驱动程序。（操作系统调用驱动程序间接地控制硬件）

（b）设备驱动程序。

（c）设备无关的软件。(处理上层的系统调用，设备的保护，权限确认，差错处理，设备分配与回收，数据缓冲区管理，建立逻辑设备名与物理设备名的映射关系

（d）用户级软件。(对用户提供接口）

17. 一个7200rpm的磁盘的磁道寻道时间为1msec，该磁盘相邻柱面起始位置的偏移角度是多少？磁盘的每个磁道包含200个扇区，每个扇区大小为512B。

磁盘以 120 转/秒的速度旋转，因此一次旋转需要 1000/120 毫秒。每次轮换 200 个扇区，扇区时间是这个数字的 1/200 或 5/120=1/24 期间毫秒 1 毫秒搜索，24 个扇区通过头部下方。因此，圆柱体的倾斜度应该是 24。

18. 一个磁盘的转速为7200rpm，一个柱面上有500个扇区，每个扇区大小为512B。读入一个扇区需要多少时间？

在 7200 转/分时，每秒有 120 转，所以我的旋转大约需要

8.33 毫秒。除以 500 我们得到的扇区时间约为 16.67μs

19. 计算上题所述磁盘的最大数据传输率。

每秒有 120 次旋转。在其中一个过程中，500×512 字节在头部下面传递。所以磁盘每转一圈可以读取 256000 字节或 30720000 字节/秒。

31. 磁盘请求以柱面10、22、20、2、40、6和38的次序进入磁盘驱动器。寻道时每个柱面移动需要6ms，以下各算法所需的寻道时间是多少？
(a) 先来先服务。
(b) 最近柱面优先。
(c) 电梯算法（初始向上移动）。
在各情形下，假设磁臂起始于柱面20。

(a) FCFS:10+12+2+18+38+34+32 = 146 柱面 = 876 ms（先到先得）

(b) SSF: 0+2+12+4+4+36+2 = 60 柱面 = 360 ms（每次最小移臂）

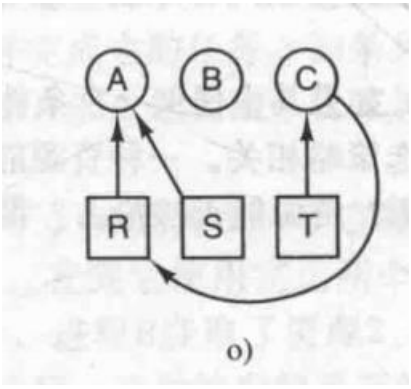
(c) 电梯算法：0+2+16+2+30+4+4 = 58 柱面 = 348 ms（一个方向）

32. 调度磁盘请求的电梯算法的一个微小更改是总是沿相同的方向扫描。在什么方面这一更改的算法优于电梯算法？

在最坏的情况下，电梯算法需要两个完整的磁盘扫描才能完成一个读/写请求服务，而改进算法最多只需要一个完整的磁盘扫描。

第六章

10. 考虑图6-4。假设在图6-4o中，C请求S而不是请求R，这是否会导致死锁？假设它同时请求S和R，情况又如何？



o)

任何变化都不会导致僵局。在这两种情况下都没有循环等待。

14. 考虑系统的如下状态，有四个进程P1、P2、P3和P4，以及五种类型的资源RS1、RS2、RS3、RS4和RS5。

C =

0	1	1	1	2
0	1	0	1	0
0	0	0	0	1
2	1	0	0	0

R =

1	1	0	2	1
0	1	0	2	1
0	2	0	3	1
0	2	1	1	0

E = (24144)

A = (01021)

使用6.4.2节中描述的死锁检测算法来说明该系统中存在一个死锁。并识别在死锁中的进程。

首先，一组未标记的进程，p=（p1 p2 p3 p4）

R1 不小于或等于 A

R2 小于 A；标记 p2；A=（0 2 0 3 1）；p=（p1 p3 p4）

R1 不小于或等于 A

R3 等于 a；标记 p3；A=（0 2 0 3 2）；p=（p1 p4）

R1 不小于或等于 R4 不小于或等于 A

因此，过程 p1 和 p4 保持未标记状态。他们陷入僵局。

17. 图6-8中的所有轨道都是水平的或者垂直的。你能否设想一种情况，使得同样存在斜轨迹的可能？

如果系统有两个或多个 CPU，两个或多个进程可以并行运行，从而导致斜轨迹。

（多 cpu 可以并行执行程序，所以它可以不需要调度，同时运行）

21. 仔细考察图6-11b，如果D再多请求1个单位，会导致安全状态还是不安全状态？如果换成C提出同样请求，情形会怎样？

D 的请求是不安全的，但 C 的请求是安全的。

22. 某一系统有两个进程和三个相同的资源。每个进程最多需要两个资源。这种情况下有没有可能发生死锁？为什么？

系统无死锁。假设每个进程都有一个资源。有一个资源是空闲的。任何一个进程都可以请求并获取它，在这种情况下，它可以完成并释放这两个资源。因此，死锁是不可能的。

23. 重新考虑上一题，假设现在共有p个进程，每个进程最多需要m个资源，并且有r个资源可用。什么样的条件可以保证死锁不会发生？

如果一个进程有 m 个资源，它就可以完成，并且不会陷入死锁。因此，最坏的情况是，每个进。程都有 m-1 资源，并且需要另一个资源。如果还有一个资源剩余，那么一个进程可以完成并释放其所有资源，让其余的资源也完成。因此，避免死锁的条件是 r>= p（m -1）+1。

不会的题：

20. 一个计算机使用32位的虚拟地址，4KB大小的页面。程序和数据都位于最低的页面(0~4095)，栈位于最高的页面。如果使用传统（一级）分页，页表中需要多少个表项？如果使用两级分页，每部分有10位，需要多少个页表项？

1 对于一级页面表，需要 2 32/2 12 或 1 百万页。因此，页表必须有 1 百万个条目。对于两级分页，主页表有 1K 个条目，每个条目都指向第二页表。只使用其中两个。因此，总共只需要三个页表条目，一个在顶层表中，一个在底层表中。

30. 考虑对于两个进程P0和P1的互斥问题的解决方案。假设变量初始值为0。P0的代码如下：

```
/* Other code*/
while(turn !=0){}/*Do nothing and wait/*/
Critical Section/*...*/
turn=0;
/* Other code*/
```

P1的代码是将上述代码中的0替换为1。该方法是否能处理互斥问题中所有可能的情形？

答：该解决方案满足互斥，因为两个过程都不可能在其关键部分。 也就是说，当 turn 为 0 时，P0 可以执行其临界区，但不能执行 P1。 同样，当 turn 为 1.但是，这假定 P0 必须先运行。 如果 P1 产生某些东西并将其放入缓冲区，那么当 P0 可以进入其临界区时，它会发现缓冲区为空并阻塞。 而且，该解决方案需要严格交替两个过程，这是不希望的。

24. 空闲磁盘空间可用空闲块表或位图来跟踪。假设磁盘地址需要D位，一个磁盘有B个块，其中有F个空闲。在什么条件下，空闲块表采用的空间少于位图？设D为16位，请计算空闲磁盘空间的百分比。

位图需要 B 位。自由列表需要 df 位。如果 d f<b，则空闲列表需要较少的位。或者，如果 f/b<1/d，则空闲列表较短，其中 f/b 是空闲块的分数。对于 16 位磁盘地址，如果可用磁盘的 6%或更少，则可用列表较短