第一章：

1. 资源分配器：可用来解决 CPU 时间，内存空间，文件存储空间,IO 设备等问题。
2. 操作系统是一直运行在计算机上的程序，通常称为“内核”，其他程序则为系统程序和应用程序。
3. 引导程序：将操作系统如何装入电脑，并运行程序。当打开电源或重启时运行。
4. 操作系统是**中断驱动**的，事件的发生通常通过硬件或软件中断来表示，必须保存所有的中断程序地址。
5. 通过中断向量，中断将控制转移到合适的中断处理程序。中断向量表包含所有的中断处理子程序的地址。
6. **陷阱**是一种软件中断，由程序运行错误或用户请求触发。
7. 操作系统通过存储寄存器以及程序计数器 PC 的值保存 CPU 的状态。
8. 同步 I/O：I/O 完成后控制权返还给用户进程。
9. 多道程序：通过多组织作业（编码或数据）使 CPU 总有一个作业可执行，从而提高了 CPU 的利用率。
10. 分时系统（或多任务）切换频率很高，用户可以在程序运行期间与之交互，创建了一种可交互式计算。

    响应时间小于 1 秒
    每个用户在内存中至少有一个程序——进程
    如果同时多个作业可以执行——CPU 调度
    如果进程不适合留在内存中了，可以利用交换技术换入和换出
    虚拟内存：允许执行的进程不必完全放在内存中

11. **异常或陷阱源于软件错误或特殊请求。**
12. **双模式：OS 对自己和其它系统元件的保护，分为用户模式和内核模式，一旦出现陷阱或中断，硬件会从用户模式切换到内核模式。P17**
13. **执行中的程序叫进程，进程执行需要资源 ，进程终止需要归还可重用资源**

第二章：

1. **操作系统为用户、进程和其他系统提供的服务**
   - **（1）.用户界面：命令行界面（CLI），图形用户界面（GUI），批界面**
   - **（2）.程序执行：系统能将程序装入内存并运行，能够结束进程，包括正常或不正常结束**
   - **（3）.io 操作：运行程序可以请求，这些 I/O 可能涉及文件或设备**
   - **（4）.文件系统操作：程序需要读写文件和目录，需要创建和删除文件、搜索文件、列出文件信息，需要提供允许和拒绝等访问管理**
   - **（5）.通信：进程需要同本机或网络上其他计算机的进程之间交换信息，通信可以通过共享内存或消息传递（消息包通过操作系统在进程间移动）来实现**
   - **（6）.错误检测：操作系统需要知道可能出现的错误**
   - **（7）.资源分配：当同时有多个用户或多个作业运行时，系统必须为每一个分配资源**
   - **（8）.统计：记录哪些用户使用了多少和什么类型的资源**
   - **（9）.保护和安全 ：**
     - **1）保护：确保所有对系统资源的访问是受控的**
     - **2）安全：系统对外界的安全性需要用户认证，也包括保护外部设备不受非法访问**

2. **系统调用：提供了操作系统提供的有效服务界面**
3. **应用程序接口（API）：一系列适用于应用程序员的函数，包括传递给每个函数的参数及**

器返回的程序员想得到的值。

4. 系统调用的参数传递：1）通过寄存器传递参数 2）参数存在内存的块或表中 3）参数通过程序放在或压入堆栈中（内存块和堆栈的方法都不限制传递参数的数量）

5. 系统调用类型：1）进程控制 2）文件管理 3）设备管理 4）信息维护

5）通信:提供了在进程、用户和计算机系统之间创建虚拟连接的机制

6. 系统程序：系统程序提供了一个方便的环境，以开发程序和执行程序

1）文件管理状态信息 2)文件修改 3）程序语言支持 4）程序装入和执行 5)通信 6)应用程序

7. 大多数用户看到的操作系统是由系统程序决定的，而不是真正的系统调用

8. 策略和机制的区分：机制决定如何做，策略决定做什么。

9. MS-DOS：利用最小的空间提供最多的功能 但没有划分成模块，没有很好的区分接口和功能层次

10. UNIX 操作系统：

系统程序

内核： 1）系统调用接口下以及物理硬件之上的全部

2） 提供文件系统、CPU 调度、内存管理、其它操作系统功能。一层组合了大量功能

11. 分层方法：操纵系统被分成若干层（级），最低层（0 层）是硬件，最高层（N 层）是用户接口

缺点：模块性，每层只能利用较低层的功能和服务

12. 微内核：把内核空间的东西尽可能移到用户空间 ，在用户模块间利用消息传递来通信

优点：1）很容易扩展系统 2）很容易移植平台 3）高可靠性 4）高安全性

缺点：用户空间与内核空间通信带来的性能影响

13. 模块：1.很多现在操作系统实现了模块化的内核 1）面向对象方法 2）内核的核心部件是分开的 3）通过接口通信 4）可动态的加载给内核

2.与分层方法类似，但更灵活

14. 虚拟机：指通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统。

1）分层方法逻辑可延伸为虚拟机，它将硬件和操纵系统都看作硬件

2）虚拟机为下层硬件提供统一接口

3）操作系统造成一种幻觉，好像每个进程都有自己的（虚拟）内存和自己的处理机

15. 系统启动：1. 操作系统必须为硬件所使用

2.小块代码（引导装载程序）：定位内核，将它装入内存，并开始执行

第三章：

1.进程和程序的区别与联系

进程	程序
动态	静态
暂时	永久
并发	——
PCB（进程控制块）	——
多个	一个
一个	多个

2. 进程控制块（PCB）：1）★进程控制状态：新的，就绪，等待，运行，终止（P73）
2) 程序计数器 3）CPU 寄存器 4）CPU 调度信息
3.进程的概念：文本段（代码段），程序计数器，堆栈，数据段
4.进程的特征：1 结构特征：程序段、数据段、进程控制块 PCB 2★动态性：最基本的特征，进程是动态产生，动态消亡的；进程在其生命周期内，在三种基本状态之间转换（就绪、等待和执行） 3 并发性：任何进程都可以同其他进程一起向前推进 4 独立性：进程是一个能独立运行的基本单位，同时也是系统中独立获得资源和独立调度的基本单位。5 异步性：每个进程都以其相对独立的、不可预知的速度向前推进
5.上下文切换：当 CPU 切换到另一个进程的时候，系统需要保存老进程的状态，并且加载新进程的状态，上下文切换的时间是系统的额外开销，切换时系统不做任何有用的工作，时间与硬件支持密切相关
6.进程调度队列：
1）作业队列：保存系统中所有的进程。
2）就绪队列：保存就绪的，等待运行的进程。
3)设备队列：等待 I/O 设备的进程。每个设备都有自己的设备队列。
4)程在不同的队列间移动。
7.调度程序：
1）长期调度（作业调度）：选择一个进程进入内存的就绪队列，控制多道程序设计的程度
2）短期调度（CPU 调度）：从就绪队列中选择一个进程，并为之分配 CPU
3）区别：执行频率，短期调度频繁，执行快
4）中期调度：能讲进程从内存（或从 CPU 竞争）中移除，从而降低多道程序设计的程度。
8.原语：创建，终止，阻塞，唤醒，挂起
9.进程创建：
1）资源共享：
1 父子进程共享所有资源 2 子进程共享部分父进程的资源 3 父子进程不共享任何资源
2）执行：
1 子进程和父进程并发执行 2 父进程等待直到子进程终止
3）地址空间
1 子进程是父进程的复制品 2 子进程装入另一个新程序
10.进程终止：
父进程能够中止子进程的执行：
1）子进程使用了超过它所分配到的一些资源
2）进程的任务不再需要
3）如果父进程结束了，一些操作系统不允许子进程继续执行（级联终止）
11.进程阻塞：
引发事件：1 请求系统服务 2 启动某种操作 3 新数据尚未到达 4 无新工作可做
12.进程协作：好处：信息共享，加快计算，模块化，方便
13.协作进程需要一种进程间通信机制（IPC）来允许进程相互交换数据与信息，有两种基本模式：1）共享内存（不要共享变量） 2）消息传递。
14.生产者消费者问题：无限缓冲，有限缓冲
15.管道：主要用于不同进程之间的通信。

第四章

1.进程的两个基本属性：资源的拥有者，调度单位

2.系统必须完成的操作：创建进程、撤销进程、进程切换。

缺点：时间空间开销大，限制并发程度的提高

3.线程的引入目的：减少进程切换和创建开销，提高执行效率和节省资源

4.线程：是 CPU 使用的基本单位，由线程 ID，程序计数器，寄存器集合和栈组成。

5.一个传统重量级的进程只有单个控制线程。

6.多线程的好处：1）响应度高 2）资源共享 3）经济 4）多处理体系结构的充分利用

7.多线程模型：

1） 多对一：一旦线程阻塞，进程就阻塞

2） 一对一：内核负担大，切换频繁

3） 多对多（二级模型）

8．线程取消：在线程结束前终止线程任务

1）异步取消：立即终止目标线程

2）延迟取消

9．线程池：创建一定数量的线程，放到池中等待工作

优点：

1）通常用现有线程处理请求比等待创建新的线程要快

2） 限制了在任何时候可用线程的数量

第五章

1.调度准则：1）CPU 利用率 2）吞吐量：一个时间单位元内完成进程的数量，3)周转时间：等待时间和执行时间 4）等待时间：进程在就绪队列中等待时间之和。 5）响应时间

2.带权周转时间:使用带权周转时间的概念。带权周转时间是\周转时间与执行时间的比：平均带权周转时间（1/n）$\sum_1^n$ wi

3 调度算法：1）先到先服务调度（FIFO）缺点：周转时间与响应时间无法保证，对短作业不利，对于那些执行时间较短的作业或进程来说，如果它们在某些执行时间很长的作业或进程之后到达，则它们将等待很长时间。

2） 最短作业优先调度(SJF)（非强占，抢占）缺点：对长作业不利，短作业时间估计的不准确性,影响调度的公平性

3）优先级调度(PSA) 缺点：低优先级进程可能永远无法执行

4）轮转法调度(RR) 缺点：时间片过短时,上下文切换造成系统开销大时间片过长时,对短作业不公平（详细看 PPT）

第六章

1.协同进程的相互作用

1）直接作用：进程间的相互联系是有意识安排的， 进程各自的执行结果互为对方的执行条件。

2）间接作用：进程间要通过某种中介发生联系，是无意识安排的

2.进程同步：指系统中多个进程中发生的事件存在某种时序关系，需要相互合作，共同完成一项任务。具体的说，一个进程运行到某一点时要求另一伙伴进程为它提供消息，在未获得消息之前，该进程处于等待状态，获得消息后被唤醒进入就绪态。

3.进程互斥：由于各进程要求共享资源，而有些资源需要互斥使用，因此各进程间竞争使用这些资源。（无序）。

4.竞争条件（race condition）：多个进程并发访问和操作同一数据且执行结果与访问发生的特定顺序有关

5.临界资源：两个或两个以上的进程不能同时使用的资源为临界资源。临界资源可能是一些独占设备，如打印机、磁带机等；也可能是一些共享变量、表格、链表、文件等。

6.临界区：每个进程中访问临界资源的那段代码称为临界区。

7.临界区三个属性：1）互斥：一个进程在临界区执行，其他进程不得进入临界区(无空等待)

2）前进：有空让进，多中选一

3）有限等待：从一个进程做出进入临界区的请求，到该请求允许为止，其他进程（比该进程优先级高）允许进入临界区的次数有上界

8.临界区解决方法：

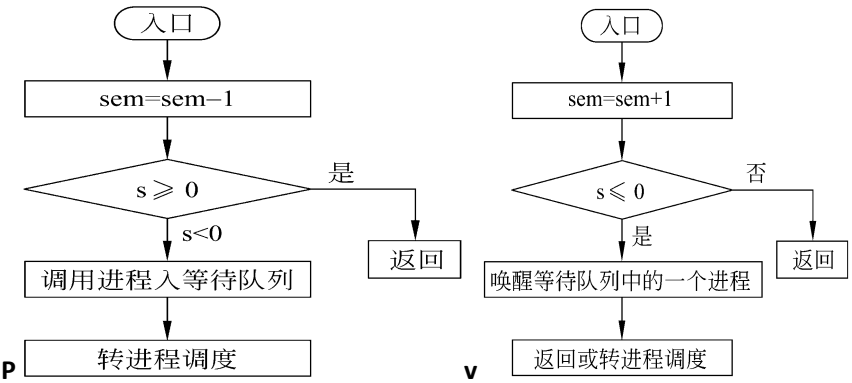
1）软件：基本思路是在进入区检查和设置一些标志，如果已有进程在临界区，则在进入区通过循环检查进行等待；在退出区修改标志。不足：忙等待，实现过于复杂，需要高的编程技巧

2）硬件：一个进程在进入临界区之前必须得到锁，而在其退出临界区时解放锁。可以简化编程任务且提高系统效率。

******信号量******

8.信号量：整形变量，初值不能为负数，必须置一次且只能一次，只能通过两个原子操作（P,V 操作）改变信号量的值。

9.P，V 原语：1）对于一种互斥或者同步关系，用一个整型变量来描述　2）当信号量大于 0 时，说明“环境安全”，可继续运行　3）当信号量小于 0 时，说明“互斥等待”，只能阻塞　4）推广到一般情况，信号量可解决复杂的同步互斥问题



例：独木桥问题，假定有如下独木桥问题：过桥时，同一方向的行人可连续过桥，当某一方有人过桥时，另一方向的行人必须等待；当某一方向无人过桥时，另一方向的行人可以过桥。试用信号量机制解决。

需要设置几个信号量？分别是互斥信号量还是同步信号量？初值设为多少？并说明设置它们的意义。

答案：将独木桥的两个方向分别标记为 A 和 B。用整型变量 countA 和 countB 分别表示 A、B 方向上已在独木桥上的行人数。初值为 0。需要设置三个初值都为 1 的互斥信号量：SA 用来实现对 countA 的互斥访问，SB 用来实现对 countB 的互斥访问，mutex 用来实现对独木桥的互斥使用。

A 方向行人过桥：

```
Begin
  P(SA);
```

B 方向行人过桥：

```
Begin
   P(SB);
```

```
    countA=countA+1;
    if (countA= =1)
        P(mutex);
  V(SA);
  过桥；
  P(SA);
    countA=countA-1;
    if(countA= =0)
        V(mutex);
  V(SA);
End
```

```
countB=countB+1;
if (countB= =1)
        P(mutex);
 V(SB);
 过桥；
 P(SB)
   countB=countB-1;
   if(countB==0)
        V(mutex);
 V(SB);
End
```

生产者消费者问题（同步问题）：n 个缓冲区、m 个生产者和 k 个消费者，允许出入缓冲区
i=0,j=0,S1(空着的个数)=n, S2（产品个数）=0， mutex=1

```
P:
While(true){
  生产产品；
  P(S1);
  P(mutex);
  往Buffer[i]放产品；
  i=(i+1)%n;
 V(mutex);
  V(S2);
}
```

```
Q:
While(true){
  P(S2);
  P(mutex);
  从Buffer[j]取产品；
  j=(j+1)%n;
  V(mutex);
  V(S1);
  消费产品；
}
```

读者写者问题:

```
Reader:
While(true){
P(rw_mutex);
P(r_mutex);
```

```
Writer：
While(true){
P(rw_mutex);
 P(mutex)；
```

```
r_cnt++ ;
if(r_cnt==1) P (mutex);
V(r_mutex);
V(rw_mutex);
Read();
P(r_mutxe);
R_cnt--;
If(r_cnt==0) V(mutex);
V(r_mutex);
}
```

```
write();
 V(mutex);
 V(rw_mutxe);
  };
```

10. 管程：一次只有一个进程。