第 1-2 章 导论和操作系统结构

1、 什么是操作系统（操作系统作用）？

操作系统是控制和管理计算机各种软件和硬件资源、有效地组织多道程序运行的系统软件，是用户与计算机之间的接口。

2、 **操作系统功能：**存储器管理、处理机管理、设备管理、文件管理、用户接口管理。

3、 操作系统服务？

用户界面、程序执行、I/O 操作、文件系统操作、通信、错误检测、资源分配、统计、保护和安全

4、 用户模式和内核模式的概念及作用？

现代操作系统是由硬件驱动的，由于操作系统和用户共享了计算机系统的硬件和软件，为了保证用户程序中的一个出错仅影响正在运行的程序，必须区分操作系统代码和用户定义代码的执行。

系统引导时，硬件开始处于内核模式，装入操作系统，开始在用户模式下执行用户进程。一旦出现陷阱或中断，硬件回车哦你个用户模式切换岛内和模式。之后系统会将控制交还给用户程序。用户模式和内核模式的操作提供了保护操作系统和用户程序不受错误用户程序影响的手段。

5、 操作系主要类型？

（1）多道批处理系统：用户作业成批处理，作业建立、过度、完成都自动有系统成批完成，且在计算机内存中同时存放几道相互独立的程序，使它们在管理程序控制下相互穿插运行。

（2）分时系统：系统内存若干并发程序对时间片共享使用。

（3）实时系统：计算机对于外来信息能够以足够快的速度处理、并在被控对象允许的范围内做出快速反应。

（4）分布式系统

（5）专用系统

6、操作系统的用户界面：命令解释程序、图形界面

7、系统调用：提供了操作系统提供的有效服务界面，

为什么用 API 而不是系统调用？可移植性、直观

8、系统调用类型：进程控制、文件管理、设备管理、信息维护、通信

9、系统程序：文件管理、状态信息、文件修改、程序语言支持、程序装入和执行、通信

10、操作系统结构：

简单结构：利用最小的空间提供最多的功能，

缺点：没有划分成模块，没有很好的区分接口和功能层次

分层结构：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，

优点：模块性，构造和调试的简单化，每层只能利用较低层的功能和服务，很容易扩展系统，很容易移植平台，高可靠性，高安全性，

缺点：用户空间与内核空间通信带来的性能影响

微内核：与分层方法类似，但更灵活,便于扩充操作系统。

11、虚拟机：单个计算机的硬件抽象为不同的执行部件，仿佛每个独立的执行环境都在自己的计算机上运行。

优点：第一，可以通过共享小型磁盘来共享文件。

第二，可以通过定义一个虚拟网络，每台虚拟机通过虚拟通信网络来传递消息。

12、计算机系统可以大致分为 4 个组成部分：计算机硬件，操作系统，系统程序与应用程序，

用户

13、作业的概念：批处理系统以作业为单位把程序和数据调入内存以执行

14、系统调用的概念：是操作系统提供给编程人员的唯一接口

第三章 进程

1. **进程的概念：执行中的程序**
2. **进程与程序的区别**

   程序不是进程，程序只是被动实体，而进程是活动实体，当一个可执行文件被装入内存时，一个程序才能被称为进程。

   - 进程是一个动态概念，程序是一个静态概念；
   - 进程有生命周期，有诞生有消亡，短暂的；而程序是相对长久的。
   - 进程具有并发性，而程序没有；
   - 进程是竞争计算机系统资源的基本单位，其并发性受到系统本身的制约；
   - 不同的进程可以包含同一程序，只要程序所对应的数据集不同

3. **进程状态及其转换**|||||||

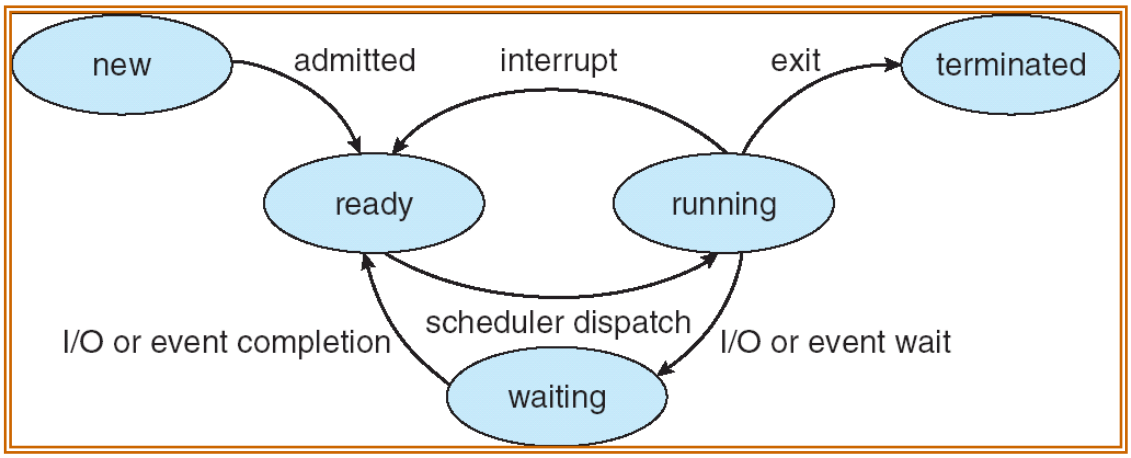
   新的：进程正在被创建

   运行：指令正在被执行

   等待：进程等待某个事件的发生

   就绪：进程等待分配处理器

   终止：进程完成执行



4. **进程控制块**

   每个进程在操作系统中用程序控制块来表示（PCB）：包括进程状态、程序计数器、CPU 寄存器、CPU 调度信息、内存管理信息、记账信息、I/O 状态信息

   PCB 用来保存程序运行期间的重要信息、进程存在的唯一标识、记录了 OS 所需的用于描述进程及控制进程所需的全部信息、进程与 PCB 是一一对应的

5. **进程调度：**选择一个可用的进程到 CPU 上执行
6. 作业队列：保存系统中所有的进程，进程进入系统会被加入作业队列
7. 就绪队列：驻留在内存中的、等待运行的程序保存在就绪队列中
8. 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
9. 调度程序：进程选择有相应的调度程序执行
10. **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（内存中进程的数量），执行的并不频繁

11. **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
12. **两者区别：**执行的频率
13. **中期调度：**中级调度主要完成虚拟内存管理相关得换入换出操作
14. **上下文切换（调度过程）：**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
15. **进程操作：**进程创建，进程终止

    进程创建：父进程创建子进程，子进程继续创建，从而形成一棵进程树

    进程终止：父进程能够中止子进程的执行
16. **父进程能够中止子进程的执行的原因：**

    ① 子进程使用了超过它所分配到的一些资源

    ② 子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
17. 对换技术、交换技术：将内存中暂时不能运行的进程，或暂时不用的数据和程序，换出到外存，以腾出足够的内存空间，把已经具备运行条件的进程，或进程需要的数据和程序，换入内存
18. **进程分类：**I/O 为主的进程、CPU 为主的进程
19. 进程终止实现

第一步：根据被终止进程的标识符，从 PCB 集合中查找对应进程控制块并读出该进程的状态；

第二步：若被终止进程正处于执行状态，则终止该进程的执行，并设置调度标志为真，用于指示该进程被终止后应重新进行调度，选择一新进程，把处理机分配给它。

第三步：若进程还有子孙进程，应将其所有子孙进程终止，以防它们成为不可控制的。

第四步：将进程所占有的全部资源释放（还给父进程或系统），释放进程控制块（若该进成为执行态，要进行进程调度）。

第五步：将被终止进程（它的 PCB）从所在队列（或链表）中移出，等待其他程序来收集相关信息。

20. 进程协作的目的:信息共享、提高运算速度、模块化、方便
21. **进程间通信基本模式：**（1）共享内存（2）消息传递
22. 消息传递通过系统调用来实现，速度慢，对于交换较少数量的数据很有用，因为不需要避免冲突。
23. 共享内存消息传递速度快，只有在建立共享内存区时需要系统调用

四． 线程

1. **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
2. 实现：将进程的资源申请和调度属性分开。即进程作为资源的申请和拥有者，但不作为调度的基本单位，这样，就产生了线程的概念。
3. **线程：**是进程中的一个实体，是独立调度和分派的基本单位。
4. **优点：**响应度高、经济、资源共享、多处理器体系结构的利用
5. 线程是 CPU 运行的一个基本单元，包括程序计数器、寄存器集、栈空间
6. 一个线程与它的对等线程共享代码段、数据段、操作系统资源
7. 传统的或重型进程等价于只有一个线程的任务
8. 线程分为内核线程和用户线程
9. **用户级线程和核心级线程的区别**

（1） 线程的调度与切换时间

用户级线程的切换通常发生在一个应用进程的多个线程之间，无须通过中断进行 OS 的内核，且切换规则也简单，因此其切换速度特别快。而核心级线程的切换时间相对比较慢。

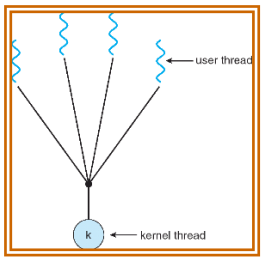
（2） 系统调用

用户级线程调用系统调用时，内核不知道用户级线程的存在，只是当作是整个进程行为，使进程等待并调度另一个进程执行，在内核完成系统调用而返回时，进程才能继续执行。而核心级线程则以线程为单位进行调度，当线程调度系统调用时，内核将其作为线程的行为，因此阻塞该线程，可以调度该进程中的其他线程执行。

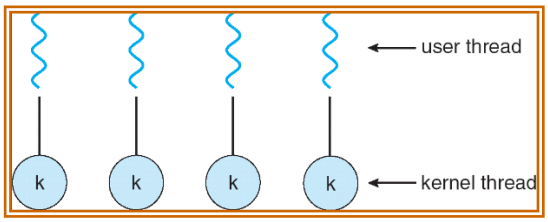
（3） 线程执行时间

如果用户设置了用户级线程，系统调用是以进程为单位进行的，但随着进程中线程数目的增加，每个线程得到的执行时间就少。而如果设置的是核心级线程，则调度以线程为单位，因此可以获得良好的执行时间。

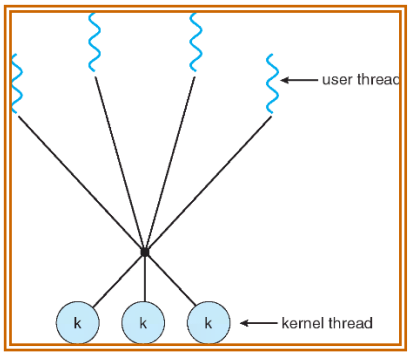
10. **多对一模型：**多个用户级线程映射到一个内核线程
    (1) 线程管理由线程库在用户空间中进行，效率比较高
    (2) 如果一个线程执行了阻塞系统调用，整个进程会阻塞
    (3) 任一时刻只有 1 个线程访问内核，并行性差（多核）



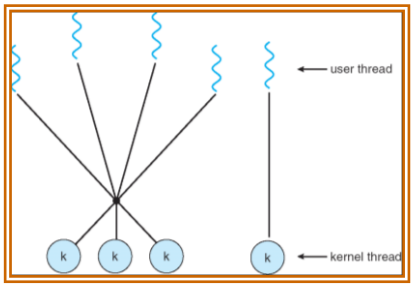
11. **一对一模型：**每个用户级线程映射到一个内核线程
    (1) 并行性高
    (2) 需要一一对应，系统开销大



12. **多对多模型：**允许多个用户级线程映射到多个内核线程上，允许操作系统创建足够多的内核线程



13. **二级模型：**跟多对多类似，也允许一个用户线程绑定到一个内核线程上



14. **线程取消：**在线程结束前终止线程的任务

15. **线程取消的两种方法：**

1 异步取消：立即终止目标线程。2 延迟取消：允许目标线程不断地检查它是否应终止，允许目标线程有机会以有序的方式终止自己

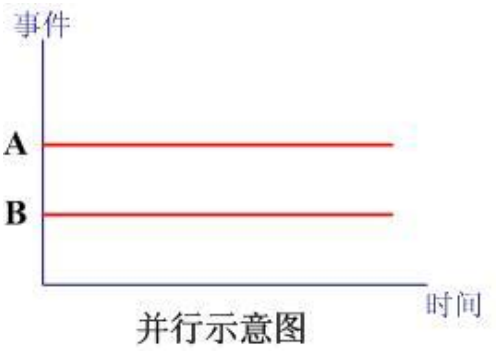
16. **并发与并行的概念及区别**

并行：同一时刻，两个事物均处于活动状态

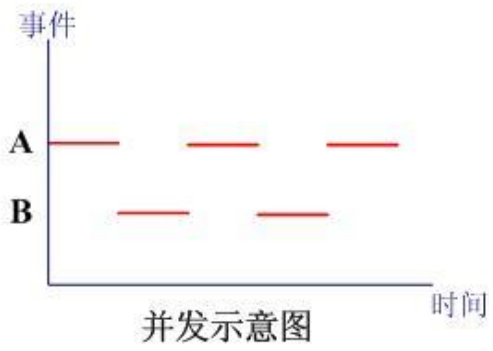
示例：CPU 中的超流水线设计和超标量设计

并发：宏观存在并行特征，微观上存在顺序性，同一时刻，只有一个事物处于活动状态

示例：分时操作系统中多个程序的同时运行



并行示意图



并发示意图

17. **线程池：**创建一定数量的线程，放到池中等待工作

18. **优点：**通常用现有线程处理请求比等待创建新的线程要快，限制了在任何时候可用线程的数量

第五章 CPU 调度

1. 进程执行由 CPU 执行和 I/O 等待周期组成。进程在这两个状态间切换。

2. **CPU 调度决策可在四中环境下发生**

A. 当一个进程从运行状态切换到等待状态。（I/O 请求）

B. 当一个进程从运行状态切换到就绪状态。（出现中断）

C. 当一个进程从邓得状态切换到就绪状态。（I/O 完成）

D. 当一个进程终止时。

当调度只发生在 A. 、D.情况下时是非抢占的。否则是抢占的。

3. **分派程序：**用来将 CPU 的控制交给由短期调度程序选择的进程

4. **分派程序功能：**

（1）切换上下文

（2）切换到用户模式

（3）切换到用户进程的合适位置，以从新启动程序

5. 一些调度算法的基础概念

吞吐量：只一个单位时间内所完成的进程的数量

周转时间：从进程提交到完成的时间段（周转时间为所有时间段之和，包括等待进入内存、在就绪队列中等待、在 CPU 上执行和 I/O 上执行）:到达就绪队列起到执行结束的时间。

等待时间：在就绪队列中所花费的时间之和

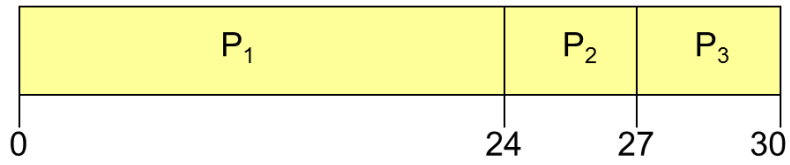
响应时间：从提交请求到产生第一响应的时间，是开始响应的时间

6. 调度算法：

A. 先来先服务（FCFS）：平均等待时间较长，是非抢占的，一旦 CPU 被分配给一个进程，该进程会保持 CPU 直到释放 CPU 为止。

Process进程	Burst Time区间时间
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3 The Gantt Chart for the schedule is:

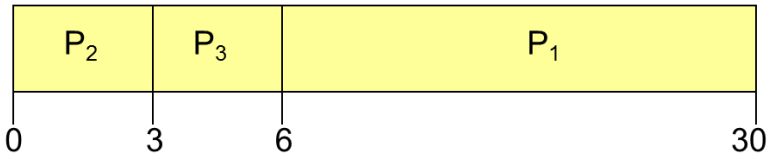


- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

Suppose that the processes arrive in the order

P_2 , P_3 , P_1

- The Gantt chart for the schedule is:



- Waiting time for P_1 = 6; P_2 = 0; P_3 = 3
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case
- *Convoy effect*（护航效果）short process behind long process
  - 所有其他进程都等待一个大进程释放CPU
  - 导致CPU和设备的使用效率变低

缺点:

1.周转时间与响应时间无法保证

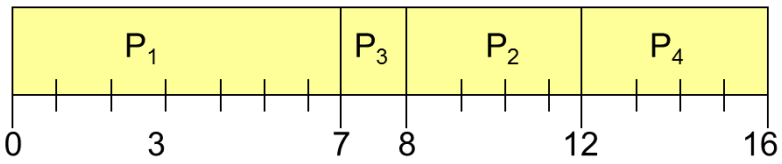
2.对短作业不利

B. 最短作业优先：将每个进程与下一个 CPU 区间段关联，分配 CPU 给具有最短区间段的进程。

最短作业优先（SJF）

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)



- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

最短剩余时间（SRT）：SRT 是针对 SJF 增加了强占机制的一种调度算法，它总是选择预期剩余时间最短的进程。只要新进程就绪，且有更短的剩余时间，调度程序就可能抢占当前正在运行的进程。

- ► SRT 不像 FCFS 偏向长进程，也不像轮转法 产生额外的中断，从而减少了开销。
- ► 必须记录过去的服务时间，从而增加了开销。
- ► 从周转时间来看，SRT 比 SJF 有更好的性能。

C. 最高响应比：在进行进程调度时，从中选择响应比高者的进程投入运行。

$$R = \frac{W + T}{\mathrm{T}} = 1 + W/T$$

D. 优先级调度：每一个进程都有一个优先号数，CPU 被分配给具有最高优先级的进程（优先号数小的优先级高）

n SJF 也是一种优先级调度算法，优先级是预测的下一个 CPU 区间时间

问题：

饥饿—低优先级进程可能永远无法执行

解决方案：

老化—随着时间增加进程优先级

E．轮转法：为每一个进程定义了一个时间片，如果一个进程在被调度程序选中之后用完了系统规定的时间片，但未完成要求的任务，则它自行释放自己所占有的 CPU 而排到就绪队列的末尾，等待下一次调度。同时，进程调度程序又去选择当前就绪队列中的第一个进程。（具体内容看书）

F．多级队列调度

G：多级反馈队列调度

7. 多处理器调度要处理的问题： 负载分配

第 6 章 进程同步

1. **临界区：**一段可能对某个共享的数据进行修改的一段代码
2. **解决临界区必须要满足的三项要求**

   （1）互斥：当有一个进程在临界区内执行的时候，任何其他的进程都不可以再进入临界区执行

   （2） 前进：空闲让进。

   （3） 有限等待：要控制进程从做出进入临界区选择到请求被允许的过程中，其他进程被允许进入该临界区的次数（避免进程等待时间过长）
3. **信号量：**信号量是一个整型值，>=0 表示系统中当前某类资源的可用数目，<0 表示系统中等待该资源的进程数。它的值只能由 P、V 原语操作所改变。

剩下的看书

第 7 章 死锁

1. **死锁产生的四个必要条件：（4 个条件同时满足会引起死锁）**

   A. 互斥，B. 占有并等待，C.非抢占，D.循环等待
2. **如果分配图无环，那么系统就没有进程死锁，如果有环，那么可能存在死锁**
3. **死锁的处理方法**

（1）可以用协议以预防或避免死锁

（2）可允许系统进入死锁状态，然后检测它，并加以回复

（3）可忽视这个问题，认为死锁不会发生

4. 死锁预防：

（1）对非共享资源，必须要有互斥条件

（2）当一个进程申请一个资源时，它不能沾有其他资源

（3）如果占有资源并申请另一个不能立即分配的资源，那么其现已分配的资源都可被抢占

（4）对所有资源类型进行完全排序，且要求每个进程按递增顺序来申请资源，当进程申请某个资源时，他必须释放掉所有较低序号的资源。

注：以上四条分别对应死锁的四个产生条件

5. 如果系统能按某个顺序为每个进程分配资源并能避免死锁，那么系统状态就是安全的，如果没有这样的顺序存在，那么系统状态就处于不安全状态。

关系：安全状态不是死锁状态，死锁状态也不是安全状态，不是所有不安全状态都能导致死锁状态。

6. 银行家算法: 看书（重点）

死锁避免是根据防止系统进入不安全状态实现的

静待资源分配破坏了占有并等待条件

资资源的按序分配可以破坏循环等待条件

银行家算法是一种死锁预防算法

第 8 章 内存管理

1. 背景知识

（1）机器指令可以用内存地址做参数，而不能用磁盘地址作参数。

（2）程序必须放入内存中的进程空间才能被执行

（3）CPU 能直接访问的存储器只有内存和处理器内的寄存器

（4）寄存器可以在一个 CPU 时钟周期内完成访问

（5）内存访问需要多个 CPU 周期

（6）设置高速缓存解决 CPU 与内存速度不匹配的问题

（7）地址绑定：

A. 编译时：如果在编译时就知道进程在内存中的的驻留地址，那么就可以生成绝对地址。

B.加载时：如果在编译时并不知道进程驻留在内存的什么地方，那么编译器就会生成可重定位代码。这种情况，最后的地址绑定会延迟到进程加载时进行。

C. 执行时：

2. **逻辑地址：**CPU 所生成的地址

3. **物理地址：**内存单元所看到的地址（即加载到内存地址寄存器中的地址）

4. **动态加载：**一个子程序只有在调用时才被加载。所有子程序都以重定位的形式保存在磁盘上。为了获得更好的内存空间使用率。

5. **静态链接：**为了程序正确执行，必须由连接装配程序把它们连接成一个可运行的目标程序

问题：花费时间，浪费空间

6. **动态链接：** 在程序开始运行时，只将主程序段装配好并调入内存，其它各段的装配是在主程序段的运行过程中逐步完成。每当需要调用一个新段时，再将这个新段装配好，并与主程序段链接。减少磁盘和内存空间的浪费。

与动态加载不同，动态链接需要操作系统的帮助。

7. **单一连续分配：**所谓单一，是指内存中只驻留一道作业。为便于地址转换，把作业连续的存放在内存中，而不是离散的存放。

内存通常分为两个区域，一个用于驻留操作系统，另一个用于用户进程。

内存映射：地址变换就是要建立虚拟地址与内存地址的关系

- 优点：方法简单，易于实现
- 缺点：仅适合于单道程序

分区管理的基本原理：给每一个内存中的进程划分一块适当大小的存储区,以连续存储各进程的程序和数据，使各进程得以并发执行。

- 实现地址重定位的方法

（1）静态地址重定位：静态地址重定位实在虚拟空间程序执行之前由装配程序完成地址映射工作。对于虚拟空间内的指令或数据来说，静态地址重定位只完成一个首地址不同的连续地址变换。它要求所有待执行的程序必须在执行之前完成它们之间的链接，否则将无法得到正确的内存地址和内存空间。

优点:容易实现，无需硬件支持。

缺点:

① 程序经地址重定位后就不能移动了，因而不能重新分配内存，不利于内存的有效利用。

② 必须占用连续的内存空间，这就难以做到程序和数据的共享。

- 动态地址重定位：动态地址重定位是在程序执行过程中，在 CPU 访问内存之前，将要访问的程序或数据地址转换成内存地址。动态重定位依靠硬件地址变换机构完成。该地址变换机构需要一个(或多个)基地址寄存器 BR 和一个(或多个)程序虚拟地址寄存器 VR

  指令或数据的内存地址 MA 与虚拟地址的关系为:

  MA=(BR)+ (VR)

- 优点：
  - 可以对内存进行非连续分配
    - 对于同一进程的各分散程序段，只要把各程序段在内存中的首地址统一存放在不同的 BR 中，则可以由地址变换机构变换得到正确的内存地址。
  - 动态重定位提供了实现虚拟存储的基础
    - 动态重定位不要求在作业执行前为所有程序分配内存，也就是说，可以部分地、动态地分配内存。
  - 有利于程序段的共享
- 缺点：
  - 需要附加的硬件支持。
  - 实现存储管理的软件算法比较复杂。

逻辑地址空间绑定到相分离的物理地址空间的概念是 OS 管理内存的核心。

保护：重定位寄存器含有最小的物理地址值，界限地址寄存器含有逻辑地址的范围值（如重定位=10400，界限=7460）当 CPU 调度选择一个程序来执行时，CPU 所产生的的每个地址都需要与这两个寄存器核对，以保证操作系统和其他用户进程不受该进程的运行影响。保护键法是一种常用的保护法

8. 分页、页表解构、分段，及其相关计算重点看书

明确非连续内存分配方法（分页机制、保护方法、共享方法等）

明确页表的结构有哪几种形式，各自的方法

明确分段管理方法

- **分页：将物理内存分为固定大小的块（称为帧），将逻辑内存分成同样大小的块（称为页），执行一个大小为 n 页的进程，要发现 n 个空闲帧并把程序装入其中，利用页表进行逻辑到物理地址的映射，会有内部碎片问题。**

  物理地址=页号所对应的帧号*帧大小（与页大小一样）+页偏移

  - **段式与页式管理的比较**

    段式管理与页式管理的地址变换机构非常相似，但两者有着概念上的根本差别。表现在：

    段是信息的逻辑单位，它是根据用户的需要划分的，因此段对用户是可见的；页是信息的物理单位，是为了管理主存的方便而划分的，对用户是透明的。

    页的大小固定不变，由系统决定。段的大小是不固定的，它由其完成的功能决定。

    段式向用户提供的是二维地址空间，页式向用户提供的是一维地址空间，其页号和页内偏移是机器硬件的功能。

    由于段是信息的逻辑单位，因此便于存贮保护和信息的共享，页的保护和共享受到限制。

第 9 章 虚拟内存（本章主要解决执行程序如何从硬盘载入内存）

1. 虚拟内存：将用户逻辑内存和物理内存区分开

虚拟内存不考虑物理存储器的大小和信息存放的实际位置，只规定每个进程中互相关连的信息的相对位置。

每个进程都拥有自己的虚拟内存。

直观的讲，它是以透明方式提供给用户一个比实际内存存大得多的地址空间，用户可在这个地址空间内编制程序，而完全不用去考虑实际内存的大小，它是逻辑上对内存容量进行扩充的一种存储器系统

- 虚拟内存可以通过如下方式实现

  按需调页

  按需调段

  实现虚拟存储技术应注意

  1.需要有相当容量的辅存以便于存放多用户作业的地址空间。

  2.要有一定容量的主存。

  3.地址变换机构

2. 按需调页：需要时调入相应的页

好处：更少的 I/O、更少的内存空间、更快的响应、更多的用户

交换程序对整个进程进行操作，而调页程序只是对进程的单个页进行操作。

调页过程：书 274

3. 页面置换算法（重点）：书 283 页

4. 帧分配最小数量：当指令完成之前出现页错误时，该指令必须重新执行，因此，必须有足够的帧来容纳所有单个指令所引用的页

- 每个进程帧的最少数量由体系结构决定，最大数量由可用物理内存的量决定

5. 帧分配算法：

固定分配：平均分配、按比例分配

优先级分配：比例分配中利用优先级而不是大小进行分配，产生缺页后，从低优先级的进程中选一个置换

6. 全局 局部分配

全局置换：从所有帧的集合中选择一个置换帧

缺点：不能控制进程的缺页率

局部置换：从自己的分配帧中选一个

缺点：不能使用其它进程不常用的内存空间

7. **系统颠簸：**如果一个进程没有足够的帧，缺页率会很高，一个进程频繁的换入换出页，造成系统颠簸。

解决方法：通过局部置换算法和优先置换算法能限制系统颠簸

8. **内存映射文件机制：**文件的内存映射可以将一磁盘块映射成内存的一页或多页，这样，一页大小的部分文件从文件系统读入物理页，以后文件的读写就按照通常的内存访问来处理，由于是通过内存操作文件而不是使用系统调用，简化了文件访问和使用。

   映射到内存中的文件进行写可能不会立即写到磁盘上的文件中

9. **内存映射到 I/O：**为了更方便的访问 I/O，将一组内存地址专门映射到设备寄存器，对这些内存地址读写就如同对设备寄存器读写。

10. **内核内存的分配方法：Buddy 系统、slad 分配**

11. **虚拟内存管理中影响性能的其他因素：**预调页、页大小、TLB 范围、反向页表、程序结构等、I/O 互锁

第 11 章 文件系统实现

1. **文件系统的两大组成部分：**一组文件、目录结构
2. 操作系统中与管理文件有关的软件和数据称为文件系统
3. 文件系统提供了在线存储和访问计算机操作系统和所有用户的程序与数据的机制。负责为用户建立、撤销、读写、修改和复制文件，还负责完成对文件的按名存取和进行存取控制。

   目标：便于查找与访问。
4. **文件的访问方法：**
   - **顺序访问：**文件中的信息被按记录的顺序依次访问。顺序访问方式基于文件的磁带模型
   - **直接访问：**允许程序快速读或者写记录，而不需要按照特定的顺序，顺序访问方式基于文件的磁带模型

5. **目录作用：**管理文件系统
6. **常用目录结构：**

单层目录：一个目录对应所有用户

- **命名问题**
  - **不允许有同名的文件**
- **分组问题：**没有分组功能

双层结构目录：将目录分为 2 级：主目录（MFD） 和用户文件目录（UFD）

- **优点：**不同用户可以为文件取相同名字、有效搜索
- 没有分组功能

树状结构目录：

优点：有效的搜索、支持分组

无环图目录：具有共享的子目录和文件

- 文件可能拥有多个绝对路径名
  - 不同的文件名指向同一个文件
- 问题
  - 磁盘利用率计算
  - 备份
  - 删除
    - 指向不存在文件的悬挂指针，甚至可能存在磁盘地址被重用

通用图目录：

7.硬链接（传统链接）

- 一个文件只有在引用计数为 0 时才会被真正删除
- 限制：仅仅限于文件，文件系统类型必须相同

8.符号链接（软链接）

- 可以跨文件系统（卷）
- 可以链接到一个目录

第 12 章 大容量存储器结构

书：393 页（重点）

关于计算虚拟地址到物理地址转换：首先求出每个分页的大小，然后看所给的虚拟地址在那个页中，俺也查询页表，得到其所对应的的帧号，用帧号乘以页大小再加上虚地址即偏移量，就得到物理地址。

- 按系统观点
  - 数据传输率
  - 传输方式
  - 共享方式
- I/O设备种类繁多，数据传输率存在很大差距
  - 低速设备：数据传输率在每秒几个到几百字节范围，常见有键盘、鼠标、语音的输入输出等设备。
  - 中速设备：数据传输率一般在每秒数千字节到万字节的范围，常见的有针式、激光打印机等。
  - 高速设备：数据传输率在每秒十万字节以上，典型的有磁带、磁盘、光盘等。

- I/O 内核子系统提供的服务：I/O 操作调度、缓冲、高速缓存、假脱机与设备预留、错误处理、保护。