第二章 处理器管理

主讲人：詹永照 教授

江苏大学计算机科学与通信工程学院

总　述

- 操作系统的重要任务之一：

  ---是充分、有效地利用系统的各类资源。
- 处理器计算机系统中最宝贵的资源，如何分配处理机更合理、更高效地为各种计算服务，就是处理机管理要考虑的问题。
- 中断处理是操作系统的灵魂。因此，本章先介绍中断处理，进而介绍程序运行中的概念——进程，以及进程的控制与调度，最后介绍处理机调度更小的单位线程。

第二章 处理器管理

2.1 中断系统

2.2 进程

2.3 线程

2.1 中断系统

- 中断的定义:

  --计算机在执行期间，发生任何非寻常的或非预期的急需处理事件，使得CPU暂时中断当前程序的执行，而转去执行相应的事件处理程序，等到事件处理结束后又返回到原来被中断的程序继续执行的过程。

- 中断的使用

  --设备和CPU之间的通信：中断控制方式、DMA方式和通道方式;

  --其它特殊情况。

2.1 中 断 系 统

2.1.1 中断装置

2.1.2 中断系统的职能

2.1.3 中断的分类

2.1.4 中断的优先级

2.1 中 断 系 统

- 定义：

发现产生的事件而产生中断的硬件装置称为中断装置。

2.1 中 断 系 统

2.1.1 中断装置

2.1.2 中断系统的职能

2.1.3 中断的分类

2.1.4 中断的优先级

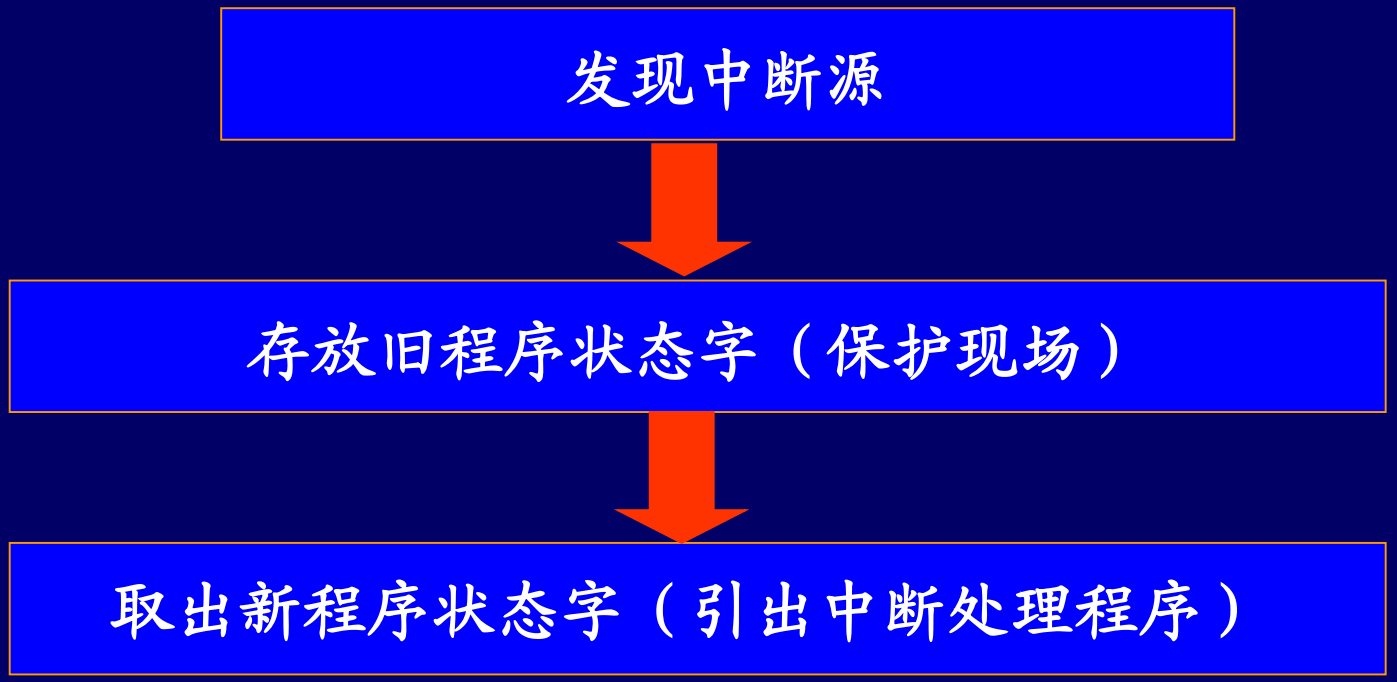
2.1.1 中断系统的职能

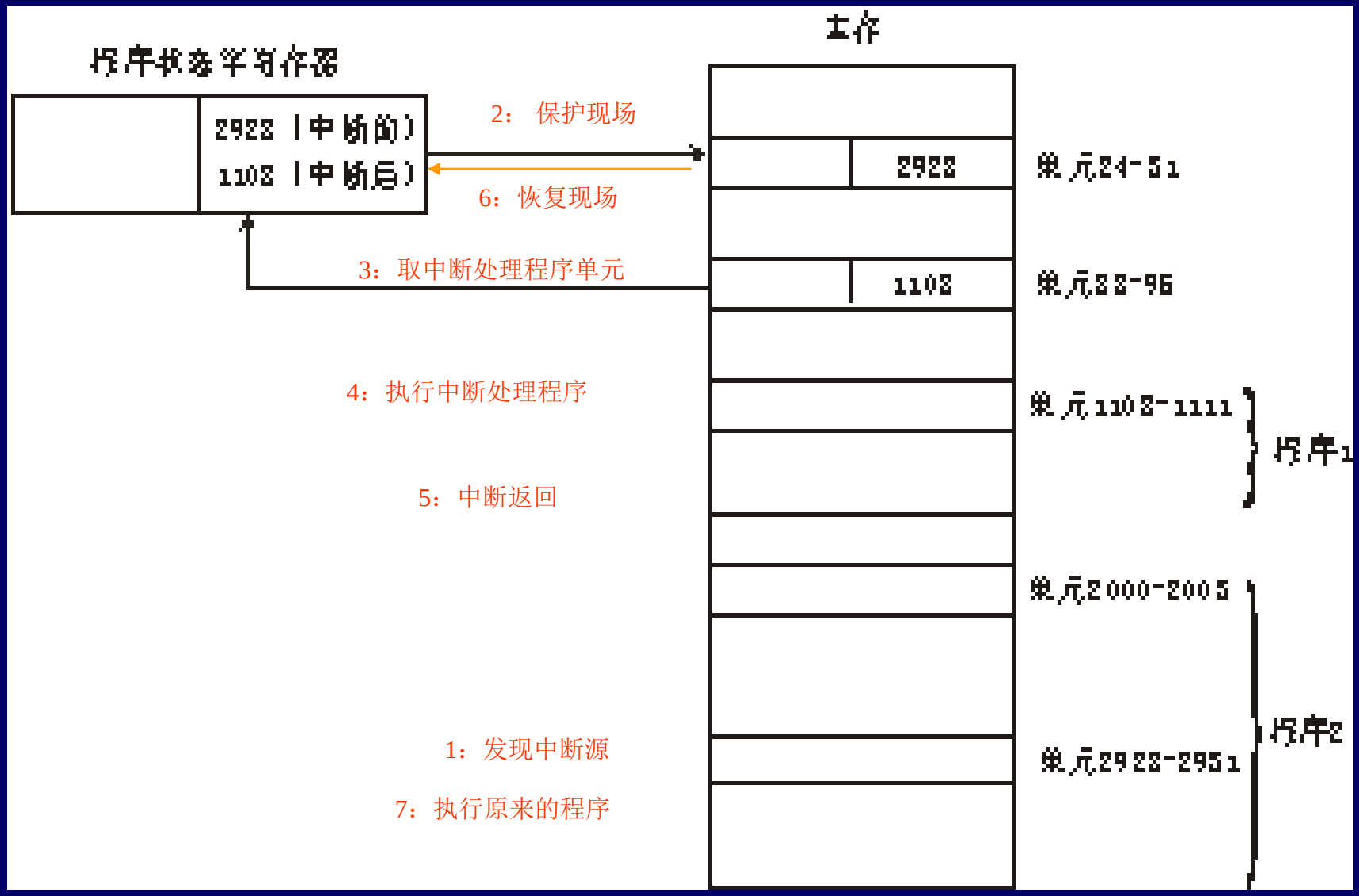
引起中断的事件

主要做以下三件事：

- 发现中断源，提出中断请求：当有多个中断源时，根据优先级，发出中断请求。
- 保护现场：保存处理器中存放程序状态字的寄存器信息，以便在中断处理结束后恢复运行。
- 启动处理事件的程序：将处理中断程序的程序状态字送入处理器的程序状态字寄存器，从而就引出了处理中断的程序。

中断处理过程示意图




程序状态字寄存器
2928（中断前）
1108（中断后）
主存
2：保护现场
6：恢复现场
3：取中断处理程序单元
4：执行中断处理程序
5：中断返回
1：发现中断源
7：执行原来的程序
2928
1108
单元24-51
单元88-96
单元1108-1111
单元2000-2005
单元2928-2951
程序1
程序2

中断处理程序

- **定义：** 处理中断事件的处理控制程序。
- **任务：** 处理中断事件，恢复正常操作。
- **主要完成的工作：**
  - 保护未被硬件保护的内容；
  - 取出中断字分析；
  - 根据中断原因转不同处理；
  - 恢复正常操作（恢复运行某一程序、重新调度另一程序运行、重新启动OS）。

2.1 中 断 系 统

2.1.1 中断装置

2.1.2 中断系统的职能

2.1.3 中断的分类

2.1.4 中断的优先级

2.1.2 中断的分类

中断主要分为软件中断和硬件中断两大类。

- 软件中断：由CPU执行某些指令引起的中断称为软件中断（亦称内部中断）。

  例：地址非法、除法出错、溢出中断等。

- 把许多执行指令过程中产生的错误或特殊情况也纳入中断处理的范围，这类中断称为异常中断，简称异常。

  异常一般分为三类：

  - 失效（Faults）
  - 陷阱（Traps）
  - 中止（Abort）

2.1.2 中断的分类

软件中断－失效：

- 产生时机：某条指令在启动之后真正执行之前被检测到异常，产生中断，且在中断服务完成后返回该条指令，重新启动并执行。

如：访问存储器时页面失效。

处理过程：

- 进入相应的中断处理程序；
- 从辅存中读取相应页面的内容到主存；
- 重新启动该指令并执行。

2.1.2 中断的分类

软件中断－陷阱:

- 产生时机: 执行产生陷阱的指令而引起的，通常也称为自愿访管中断，执行完中断处理程序回到程序的下一条指令。

自愿访管中断是实现对操作系统的功能调用。如申请主存、I/O资源等。

- 例: DOS中的INT和UNIX中Trap命令的执行。

2.1.2 中断的分类

软件中断－中止:

- 产生时机: 异常发生后无法确定造成异常指令的实际位置时，原来的程序已无法继续执行。
- 处理方式: 在此情况下原来的程序已无法继续执行，因此中断服务程序往往重新启动操作系统并重建系统表格 。
- 例: 硬件错误或系统表格中的错误（主存出错）。

2.1.2 中断的分类

- 硬件中断：由CPU或内存外部发出的中断信号而引起的中断称为硬件中断（亦称外部中断）。
- 例如I／O设备发的中断、各种定时器引起的时钟中断等等。

- 硬件中断的分类：
  - 非屏蔽中断：指不允许用户干预的中断；一旦发出中断信号后，CPU要立即响应，如I/O通道错，奇偶校验错等。
  - 可屏蔽中断：可通过每一类中断源设置一个中断屏蔽触发器，屏蔽它们的中断请求。（大多数硬件中断都是可屏蔽中断）

2.1 中 断 系 统

2.1.1 中断装置

2.1.2 中断系统的职能

2.1.3 中断的分类

2.1.4 中断的优先级

2.1.3 中断的优先级

- 优先级的引入:

  在计算机系统中，常常会遇到多个中断信号同时发生的情况，这时，CPU必须确定首先为哪一个中断信号服务，以及服务的顺序。另外，有些中断服务是不允许其它中断打断的，有些中断要优先处理，有些中断在服务期间要接受比它更需要紧急处理的中断等等。解决这些问题的方法就是为每个中断确定一个中断优先级。

2.1.3 中断的优先级

- 解决方法

  --为每个中断确定一个中断优先级。

- 定义：多个中断事件同时发生时，中断装置应按规定的顺序来响应，该顺序称为中断优先级。

- 如何实现中断优先级
  - 软件查询
  - 硬件处理

2.1.3 中断的优先级

软件查询方式：中断优先级由查询的顺序决定，先被查询的中断具有高的优先级。

--这种方法需要设置一个中断请求信号的寄存器，将各中断信号保存下来以便查询，并可对没有处理的中断请求作一个备忘录。

- 优点：可利用修改软件的方法来改变中断优先级，而不必要更改硬件。
- 不足：服务效率低，因为优先级最低的中断请求信号必须先将优先级高的中断查询一遍，若中断种类较多，有可能使低优先级的中断处理相应较慢。

2.1.3 中断的优先级

硬件处理方式：为了克服软件方式的缺点，通常采用硬件编码器来组成中断优先级电路。根据编码器的编码形式确定各个中断信号的优先级。

- 优点：响应速度快，中断能及时得到处理。
- 缺点：修改优先级困难，不灵活。

2.2 进　程

- 进程的引入：
  - 程序概念的缺点：程序的概念已不足以描述程序的并发执行，如：两个用户同时执行CC编译自己的程序，某时刻程序的执行状态难以刻画（运行、就绪、等待？）。
  - 因此，有必要引入一个能确切描述并发执行过程特点、能揭示系统动态特性的新概念——进程。

2.2.1 进程的概念

- 进程的定义（程序段+数据）
  - 能和其它程序并行执行的程序段在某数据集合上的一次运行过程，它是系统资源分配和调度的一个独立单位。
- 进程的特点：
  - 进程的基础是一个程序段，而不是整个程序。
  - 进程是程序段在一些数据上的一次运行，即在“某数据集合”上的运行。
  - 进程是一个动态的概念，它实质上是程序的一次执行过程，也就是说，进程具有动态性。
  - 进程是一个能独立运行的基本单位，具有独立性，是资源分配和调度的单位。

2.2.1 进程的概念

- 进程和程序的区别:
  ---程序是一组指令的集合，它只规定了运行活动时所要完成的功能，本身没有运行的含义，因此是一个静态的概念。
  ---进程是一段程序的一次运行活动，是一个动态概念。进程是具有产生、消亡及其“执行—暂停—执行”的活动过程。进程从产生到消亡都是具有其动态生命历程，是具有一定“生命期”的。
  ---进程和程序之间不存在一一对应关系。一个程序可以对应多个进程；反之，一个进程至少要对应一个程序，或对应多个程序，多个进程也可对应相同的程序。

2.2 进　程

2.2.1 进程的概念
2.2.2 进程的实体
2.2.3 进程状态和转换
2.2.4 进程控制
2.2.5 进程调度

2.2.2 进程的实体

1、进程的组成（3个部分）

物质基础

- 程序：它规定进程一次运行活动所需完成的功能。多个进程也可以同时对应一个程序。
- 数据集合：程序运行时需要用到的数据和开辟的工作区域构成进程一次运时的数据集合，它为某进程所专用。
- 进程控制块（PCB）：描述和标志进程的存在的数据结构（创建进程时，建立PCB；完成任务被撤销时，撤销PCB。）

2.2.2 进程的实体

PCB与进程一一对应，系统根据PCB的存在而感知进程的存在。PCB是进程存在的惟一标志。

PCB的内容随具体操作系统的不同而异，右图为多数PCB块中包含的信息。

PCB结构

name	进程标识符	标识
status	进程当前状态	说明
next	当前队列指针	
all-q-next	总链指针	
start-addr	执行程序开始地址	
priority	进程优先级	
cpustatus	CPU 现场保护区	现场
communication information	进程通信	
process-family	家族联系	
own-resource	占有资源清单	

2.2.2 进程的实体

2. PCB的组织方式（为了管理上的方便）

- 线性方式：所有的PCB组成一个数组，系统可以通过下标访问PCB 。
- 链接方式：把具有相同状态的PCB用其中的链接字按一定方式连接成一个队列。系统中一般有运行队列、就绪队列、阻塞队列等。系统设置固定单元，用以指出各队列的第一个PCB的起始地址。
- 索引方式：系统根据所有进程状态，建立N张索引表，并把各索引表在内存的首地址记录于内存中的一些专用单元中，在每个索引表中，记录具有相应状态的某个PCB在PCB表中的地址。

2.2 进　程

2.2.1 进程的概念
2.2.2 进程的实体
2.2.3 进程状态和转换
2.2.4 进程控制
2.2.5 进程调度

2.2.3 进程状态和转换

- 进程的三种基本状态
  - 按进程在执行过程中的不同时刻的不同状况把进程划分为三种基本状态。
    - 就绪状态
    - 执行状态
    - 阻塞状态

2.2.3 进程状态和转换

- 就绪状态

  ---进程已得到除CPU以外的全部资源，是一旦获得CPU就可以执行的状态。

- 执行状态

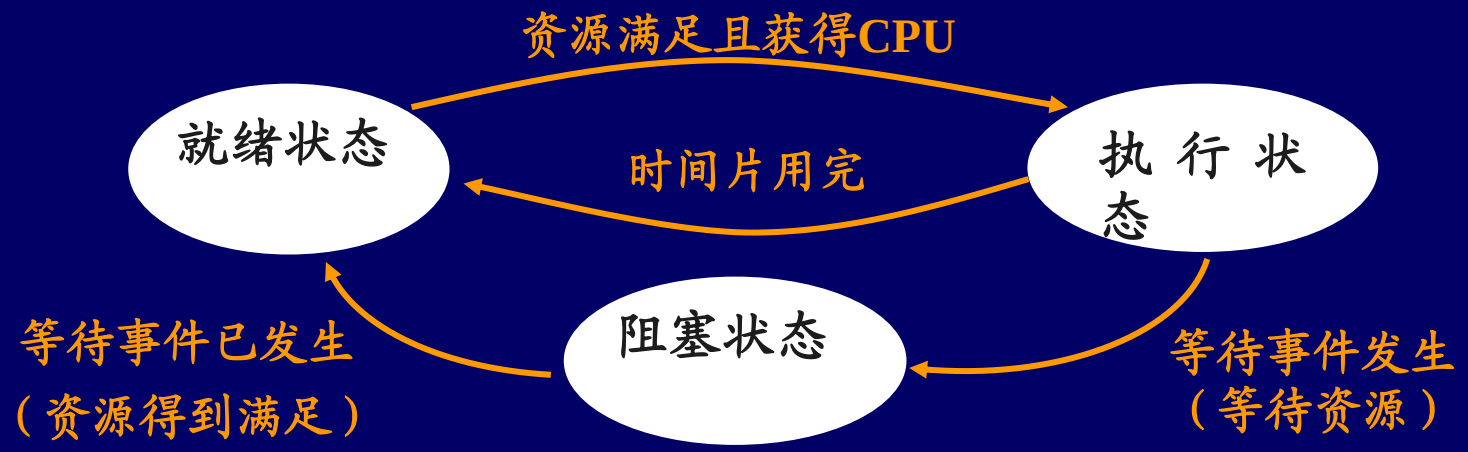
  ---进程已获得必要的资源并占有CPU，正在执行的状态。在单处理器系统中，只能有一个进程处于执行状态。在多机处理系统中则可能有多个进程处于执行状态。

- 阻塞状态（等待状态）

  ---进程因等待某一事件而暂不能执行的状态，是等待除CPU外的资源或事件。

2.2.3 进程状态和转换

◆ 进程在运行过程中，由于自身的进展情况，由于与其它进程并发执行，相互制约，也由于外界条件的变化，其状态会不断发生变化。



2.2 进　程

2.2.1 进程的概念
2.2.2 进程的实体
2.2.3 进程状态和转换
2.2.4 进程控制
2.2.5 进程调度

2.2.4 进程控制

- 进程控制的主要任务：
  - 对系统中所有进程从创建到消亡的全过程实行有效地管理和控制。
- 包括
  - 对进程状态变化加以管理和控制；
  - 创建新进程和撤消已完成任务的进程；

2.2.4 进程控制

- 进程控制包括：

  进程创建、

  进程撤消、

  进程阻塞、

  进程唤醒。

  这些操作都要对应地执行一个特殊的程序段（操作系统核心程序）称之进程控制原语（一种特殊的系统调用）。同时系统也通过系统调用给用户提供进程控制的功能。

2.2.4 进程控制

- OS的内核中，用于进程控制的原语
  - 创建原语
  - 撤销原语

对应进程的建立和撤消

  - 阻塞原语
  - 唤醒原语

对应进程状态转换
使进程在等待和就绪态之间转换

2.2 进　　程

2.2.1 进程的概念
2.2.2 进程的实体
2.2.3 进程状态和转换
2.2.4 进程控制
2.2.5 进程调度

2.2.5 进程调度

- 调度的层次
- 调度的方式
- 进程调度算法

2.2.5 进程调度

调度的层次（共分三级）

- 高级调度

  ——也叫宏调度或作业调度，其主要功能是按照某种原则从外存上的后备作业中选一个或几个进入内存，并为其运行做好有关准备工作。系统一旦接纳了一个作业，便将它变为一个或一组进程，为它们分配必要的资源，并挂到就绪对列上。

批处理系统中用，分时系统不用

2.2.5 进程调度

- 中级调度：负责内外存之间的进程对换，以解决内存紧张的问题，即它将内存中处于等待状态的某些进程调到外存对换区，以腾出内存空间，再将外存对换区中已具备运行条件的进程重新调入内存准备运行。

- 低级调度：又称微调度或进程调度。它决定就绪队列中哪个进程将获得处理器，并将处理器分配给该进程，让其真正运行。执行进程调度功能的程序是进程调度程序，进程调度程序的执行频率很高，所以它必须常驻内存。进程调度是操作系统中最基本的调度，在批处理和分时系统中都必须配置它。

2.2.5 进程调度

- 调度的层次
- 调度的方式
- 进程调度算法

2.2.5 进程调度

调度的方式：指把CPU分配给进程后，它能占用多长时间。通常有两种方式：剥夺式和非剥夺式。

- 剥夺式（抢占式）：又称抢占式。当一个进程正在执行时，系统可以基于某种原则强行将CPU的控制权从当前进程转给其它进程。（这些原则如：优先级原则、短进程原则、时间片原则等）
- 非剥夺式（非抢占式）：又称非抢占式。在该方式下，进程对处理器的控制权具有自主性，除非该进程主动出让CPU控制权，否则其它进程不可能有机会运行。
  - 优点：简单，系统开销小。
  - 缺点：不能及时处理紧急任务。

2.2.5 进程调度

- 调度的层次
- 调度的方式
- 进程调度算法

2.2.5 进程调度

进程调度算法：进程调度在选择就绪进程投入运行时，可能会发现有多个进程同时处于就绪状态，因此它应按一定的算法选择一个进程，以便把CPU分配给它，这个算法就是进程调度算法。

- 调度算法直接影响到操作系统的适用环境和工作效率；
- 对不同的系统及系统目标，有不同的调度算法。

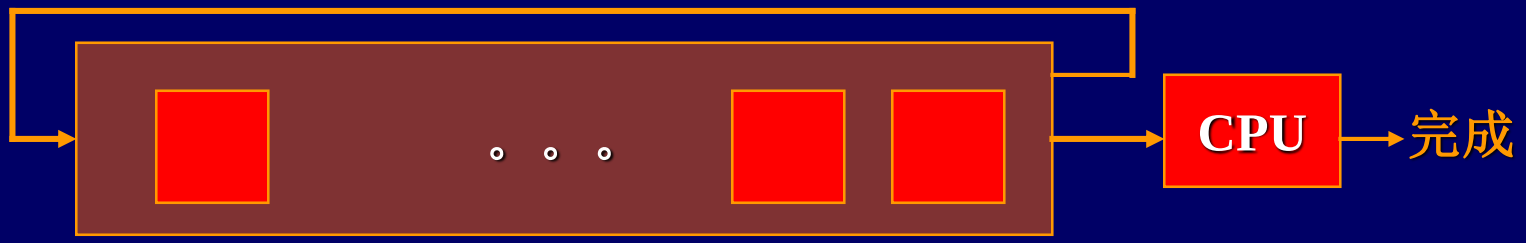
2.2.5 进程调度

常用的算法

- 时间片轮转法（适合于分时系统）

  --把CPU按时间片、按顺序赋予就绪队列中的每一个进程。若某个进程在规定时间片内未执行完毕，也必须释放CPU，并把CPU分配给下一个进程。对于未完成执行的进程，释放CPU后回到就绪队列的末尾排队，等待下一轮时间片。这样一次又一次地执行，一次又一次地等待，直到该进程的任务完成。

  --时间片轮转调度算法特别适合于分时系统中使用。



2.2.5 进程调度

- 优先级调度（适用于批处理系统和实时系统 ）

---把处理器分配给就绪队列中具有最高优先级的进程。该算法的关键在于如何确定进程的优先级，常用的有如下两种方法来确定进程的优先级:

--静态优先级：在进程创建时即被确定的，不变化

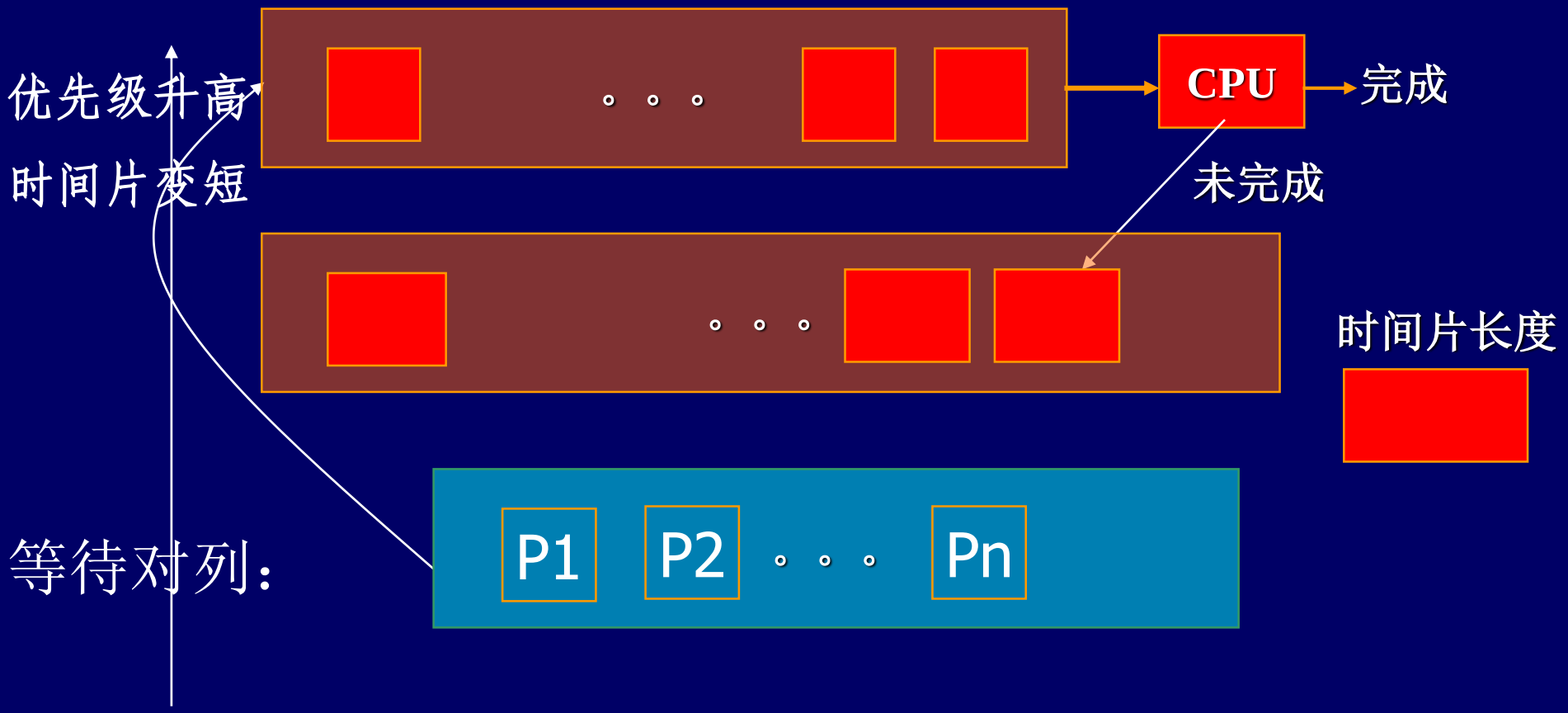
优先级确定原则：进程类型，对资源的需求，用户要求

--动态优先级：是指在进程的执行期间，按某种原则不断修改进程的优先级，优先级一般随进程的等待时间，占用CPU的时间的变化而变化。

2.2.5 进程调度

- 多重队列轮换法
  - ——多重队列轮换法就是把时间片轮转法中的单就绪队列改为双就绪队列或多就绪队列，并赋给每个队列不同的优先权。进程调度首先调度高优先权队列中的进程占用CPU并执行，当高优先权队列中已无就绪进程时，才能去调度低优先权队列中的就绪进程。

2.2.5 进程调度



2.2.5 进程调度

特点

- 每个就绪队列按“先来先服务”的原则获得CPU;
- 进程用完时间片后未完成，排入下一级就绪队列中;
- 阻塞的进程转为就绪状态时，安排在优先级高的就绪队列中;
- 调度总是先调度优先级高的队列中的进程。

第二章 处理器管理

2.1 中断系统

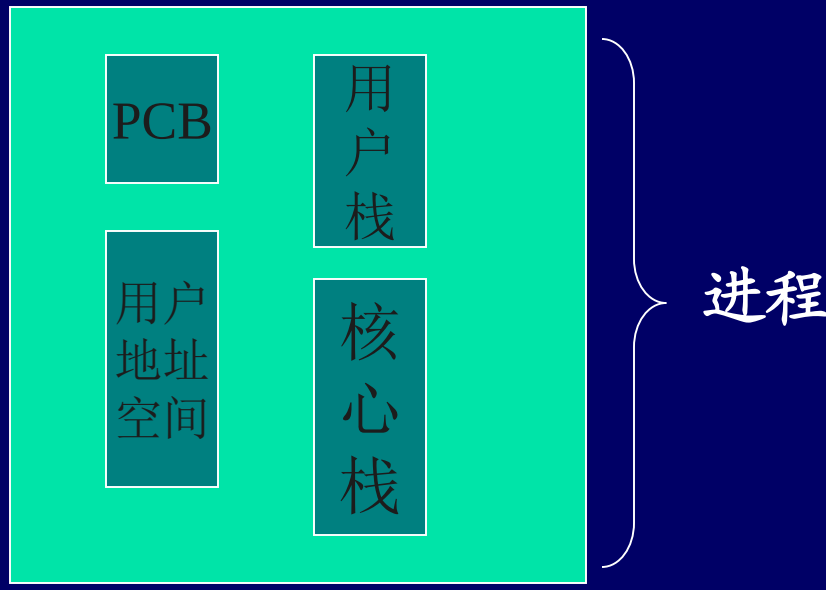
2.2 进程

2.3 线程

2.3 线　程

- 操作系统引入进程后，程序并发执行时，系统必须为进程的创建、撤消和切换中付出较大的时空开销。系统中所设置的进程数目不宜太多，进程切换的频率也不宜过高，因此它也限制了程序的并发执行程度。
- 在引入线程的操作系统中，线程是进程中的一个实体，它是比进程更小的能独立运行的基本单位，因此，在为其创建、撤消和切换所需付出的开销也就更小，因而能显著地提高程序执行的并行程度和系统吞吐量。
- 线程的定义：线程是指进程内的一个执行单元（有的书中称线程为轻型进程），也是进程内的可调度实体。在多线程操作系统中，通常在一个进程中包括多个线程，每个线程都作为利用CPU的基本单位，是具有最小开销的实体。

2.3 线　程



- 线程的属性
  - 四部分组成:
    - 惟一的标识符、状态寄存器
    - 两个栈、私用存储器
  - 是进程中的一个实体，是被系统独立调度和分派的基本单位。本身很小，调度开销小。
  - 线程可以创建和撤消另一个线程；同一个进程中的线程可以并发执行；不同进程中的线程也能并发执行。

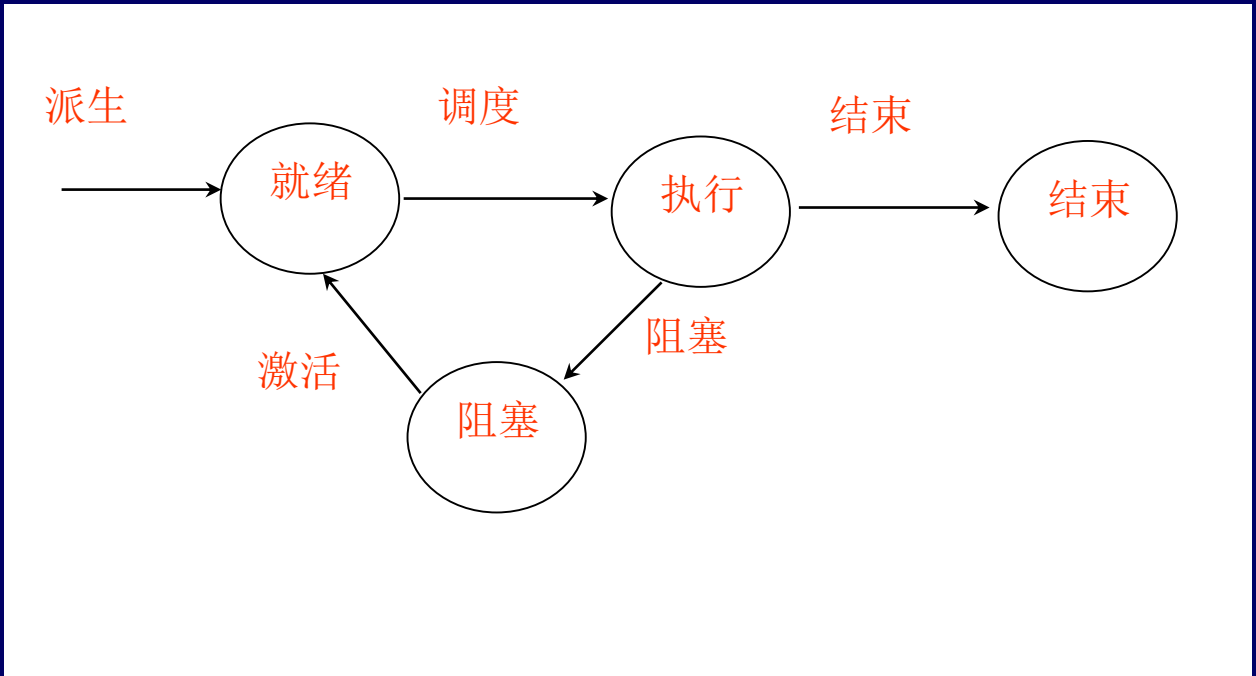
2.3 线　程

线程和进程的关系

- 从地址空间来看：线程是进程内的一个执行单元；进程至少有一个线程；它们共享进程的地址空间；而进程有自己独立的地址空间；
- 从资源拥有情况来看：进程是资源分配和拥有的单位；而线程是同一个进程内的线程共享进程的资源；
- 从CPU调度单位方面来看：进程不是处理器调度的基本单位，而线程是处理器调度的基本单位；
- 从并发性角度来看：进程可并发执行，线程也可并发执行；
- 从状态角度看：进程和线程同样有三态转换、创建和终止。

2.3 线　　程

- 线程的状态与操作



2.3 线　程

线程典型应用场合

- 前后台并行工作场合 :如,表处理。
- 异步处理工作场合 :如,设置一个备份线程，它每隔一分钟把RAM缓冲区的数据和信息写入磁盘 。
- 需要加快执行速度的场合 :如,一个线程在计算一批数据时另一个线程可以从设备上输入下一批数据，从而加快进程的执行速度。
- 组织复杂工作的程序 :如,多个不同的任务需要处理，多线程机制可方便程序的设计和组织，且可提高整个系统效率。
- 同时有多个用户服务请求的场合:如,文件服务器。

2.3 线　程

- 线程实现两种方式：
  - 1. 内核支持线程的实现
  - 2. 用户级线程的实现
- 1. 内核支持线程的实现
  - 由OS内核管理：创建、撤消和切换。
    内核提供相应的系统调用和API，用户程序可以创建、执行和撤消线程。
  - 内核中保留了一个线程控制块（TCB）。
  - 调度算法
    - 和进程类似，抢占式和非抢占式两种；
    - 时间片轮转法、优先权等算法。

2.3 线　程

- 2. 用户级线程的实现
  - 仅存在于用户级中，创建、撤消和切换与内核无关，内核不知道线程的存在；
  - 在用户空间实现的，运行在中间系统之上，中间系统提供线程库管理调度线程。