目录

[进程同步 1](#_Toc503479424)

[死锁四个必要条件 1](#_Toc503479425)

[线程和进程 2](#_Toc503479426)

[进程7状态 2](#_Toc503479427)

[进程与线程 2](#_Toc503479428)

[两种线程 3](#_Toc503479429)

[内存管理 3](#_Toc503479430)

[全局描述符表 3](#_Toc503479431)

[局部描述符表 4](#_Toc503479432)

[任务状态段 4](#_Toc503479433)

[分页分段的区别 6](#_Toc503479434)

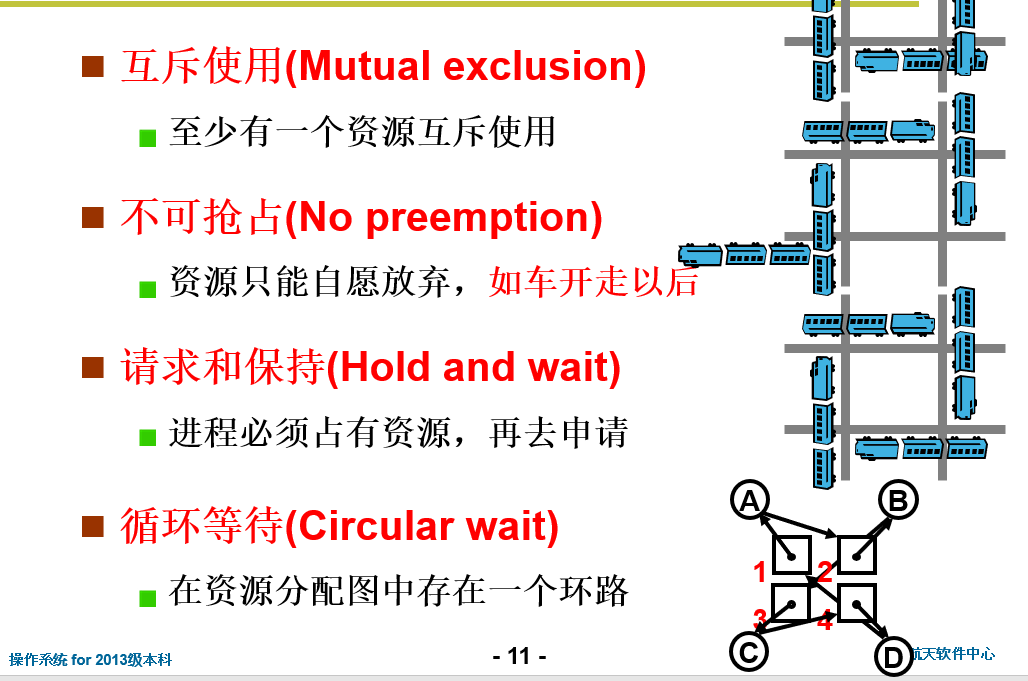
[分段存储方式的优缺点 6](#_Toc503479435)

[磁盘 7](#_Toc503479436)

[磁盘访问延迟 7](#_Toc503479437)

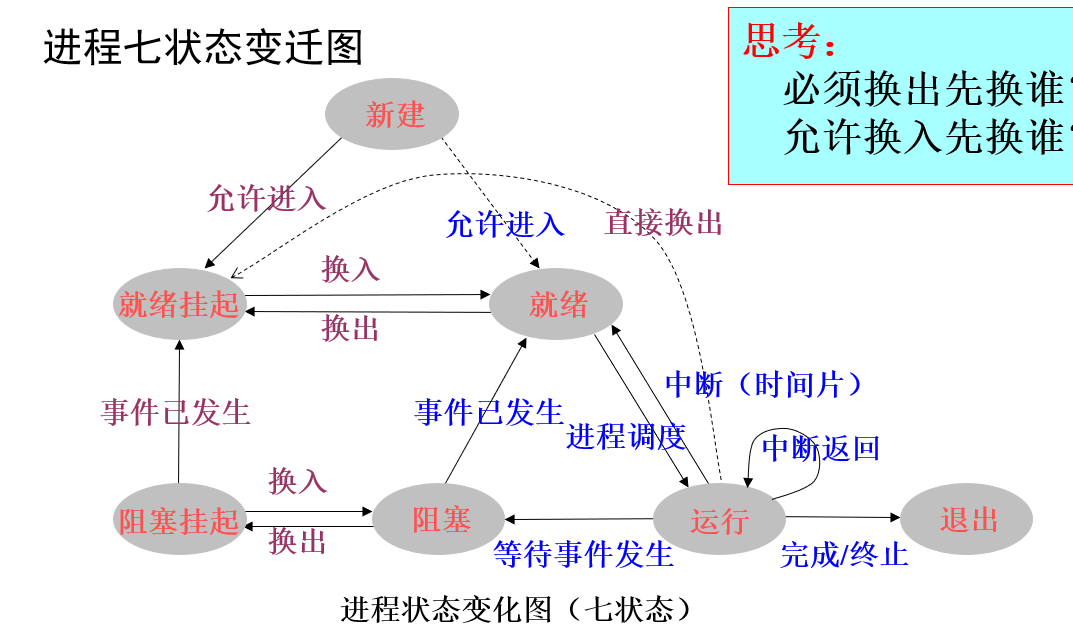
# 进程同步

## 死锁四个必要条件



# 线程和进程

## 进程7状态



## 进程与线程

* 进程 ── 重量级任务  
  ① 每一进程占用独立的地址空间。  
   ⇒ 此处的地址空间包括代码、数据及其他资源。  
  ② 进程间的通信开销较大且受到许多限制。  
   ⇒IPC， 对象（或函数）接口、通信协议、...  
  ③ 进程间的切换开销也较大。  
   ⇒ 上下文切换（***context switch***），地址空间切换。  
   ⇒ 上下文包括代码、数据、堆栈、处理器状态、资源、...
* 线程 ── 轻量级任务  
  ① 多个线程共享进程的地址空间（代码、数据、其他资源等）。  
   ⇒ 线程也需要自己的资源，如程序计数器、寄存器组、调用栈等。  
  ② 线程间的通信开销较少且比较简单。  
   ⇒ 因为共享而减少了需要通信的内容。  
   ⇒ 但也因为充分共享，需要用户对共享资源进行保护。  
  ③ 线程间的切换开销也较小。  
   ⇒ 只需保存每一线程的程序计数器、寄存器组、堆栈等空间。  
   ⇒ 不必切换地址空间，从而成本大为降低（通常约1/10）。

## 两种线程

用户级线程和内核级线程的区别

（1）内核支持线程是OS内核可感知的，而用户级线程是OS内核不可感知的。

（2）用户级线程的创建、撤消和调度不需要OS内核的支持，是在语言（如Java）这一级处理的；而内核支持线程的创建、撤消和调度都需OS内核提供支持，而且与进程的创建、撤消和调度大体是相同的。

（3）用户级线程执行系统调用指令时将导致其所属进程被中断，而内核支持线程执行系统调用指令时，只导致该线程被中断。

（4）在只有用户级线程的系统内，CPU调度还是以进程为单位，处于运行状态的进程中的多个线程，由用户程序控制线程的轮换运行；在有内核支持线程的系统内，CPU调度则以线程为单位，由OS的线程调度程序负责线程的调度。

（5）用户级线程的程序实体是运行在用户态下的程序，而内核支持线程的程序实体则是可以运行在任何状态下的程序。

内核线程的优点：

当有多个处理机时，一个进程的多个线程可以同时执行。

内核线程的缺点：

由内核进行调度，需要系统调用。

用户线程的优点：

（1）线程的调度不需要内核直接参与，控制简单，用户灵活性大。

（2）可以在不支持线程的操作系统中实现。

（3）创建和销毁线程、线程切换代价等线程管理的代价比内核线程少得多。

（4）允许每个进程定制自己的调度算法，线程管理比较灵活。这就是必须自己写

管理程序，与内核线程的区别

（5）线程能够利用的表空间和堆栈空间比内核级线程多。

（6）同一进程中只能同时有一个线程在运行，如果有一个线程使用了系统调用而

阻塞，那么整个进程都会被挂起。另外，页面失效也会产生同样的问题。

用户线程缺点：

资源调度按照进程进行，多个处理机下，同一个进程中的线程只能在同一个处理机下分时复用。一个线程阻塞时，引起进程的阻塞。

# 内存管理

## 全局描述符表

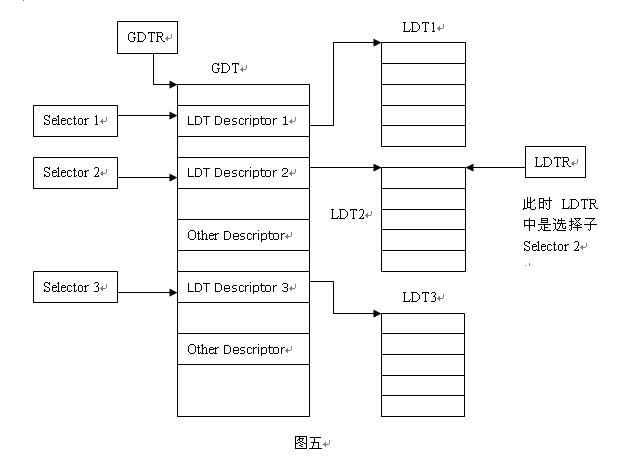
* GDT指向LDT\TSS段描述符表入口地址
* GDT记录OS使用的内存区域的段信息
* GDT对所有进程均可用
* 整个系统只定义一个GDT，其中存放有允许多个进程共同访问的那些段的描述符。
* GDT位置由CPU的GDTR寄存器指出。

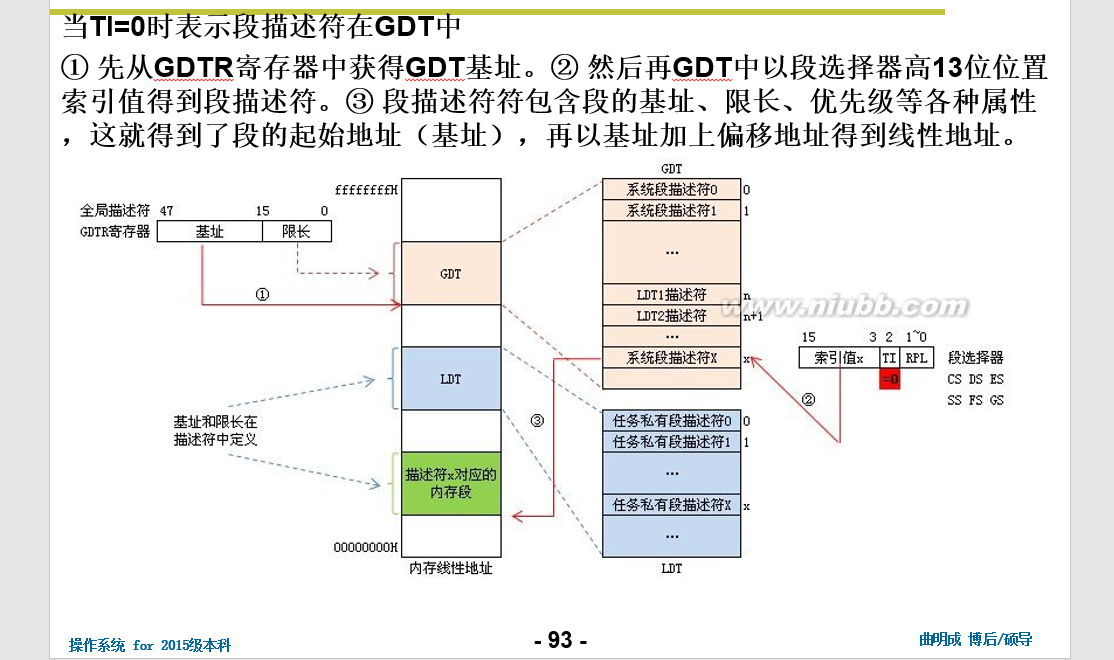
## 局部描述符表

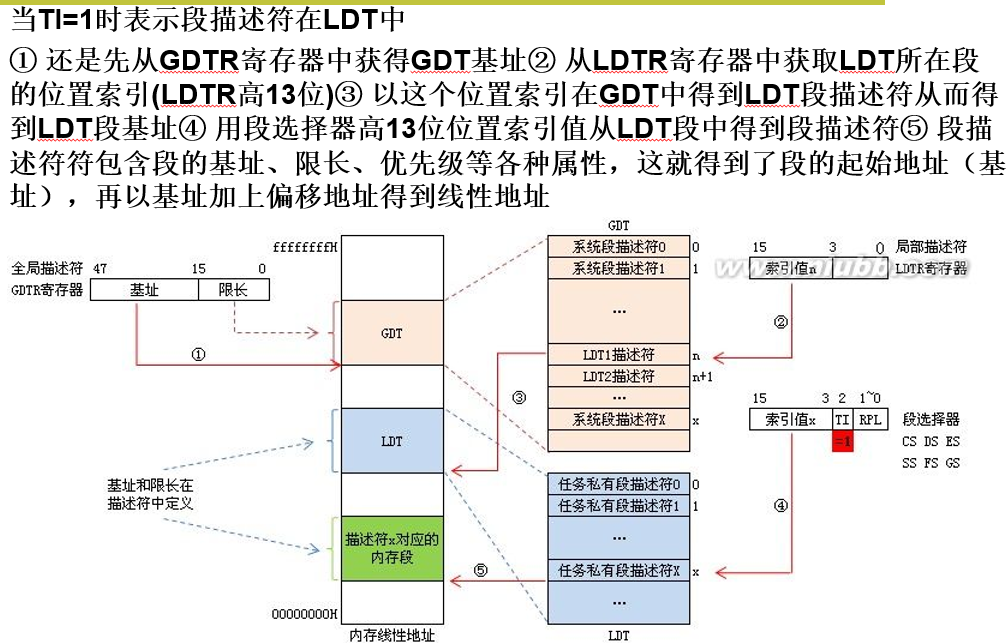
* 每个进程的局部空间的所有段描述符集中存放在一张表中，这张表为该进程的局部描述符表（LDT）。
* 每个进程的LDT也将它看成是一个段,因此也需要一个段描述符来给出其在内存的基址、长度等信息，称为LDT描述符，存放在GDT中。
* 当前正在执行的进程，其在GDT中的LDT描述符位置，由CPU的LDTR寄存器指出。

## 任务状态段

* **每个任务都有一个任务状态段TSS，描述任务状态段的描述符称为TSS描述符**
* **所有任务的TSS描述符均被存放在GDT中。**
* **系统可通过TR寄存器在GDT中找到正在执行的任务的TSS描述符，从而找到相应任务的TSS。**
* **任务状态段TSS中包含：**
* **1.任务的CPU现场（通用寄存器、段选择寄存器、指令指针、标志寄存器等）；**
* **2.特权级分别为0、1、2时的堆栈段选择符和栈顶指针；**
* **3.该任务被调用时，前一个任务TSS的返回连接选择符；**
* **4.I/O允许位图等。**







## 分页分段的区别

分页和分段有许多相似之处,比如两者都不要求作业连续存放.但在概念上两者完全不同,主要表现在以下几个方面:   
(1)页是信息的物理单位,分页是为了实现非连续分配,以便解决内存碎片问题,或者说分页是由于系统管理的需要.段是信息的逻辑单位,它含有一组意义相对完整的信息,分段的目的是为了更好地实现共享,满足用户的需要.   
(2)页的大小固定,由系统确定,将逻辑地址划分为页号和页内地址是由机器硬件实现的.而段的长度却不固定,决定于用户所编写的程序,通常由编译程序在对源程序进行编译时根据信息的性质来划分.   
(3)分页的作业地址空间是一维的.分段的地址空间是二维的.

## 分段存储方式的优缺点

分页对程序员而言是不可见的，而分段通常对程序员而言是可见的，因而分段为组织程序和数据提供了方便。与页式虚拟存储器相比，段式虚拟存储器有许多优点：

(1)    段的逻辑独立性使其易于编译、管理、修改和保护，也便于多道程序共享。

(2)    段长可以根据需要动态改变，允许自由调度，以便有效利用主存空间。

(3)    方便编程，分段共享，分段保护，动态链接，动态增长

 因为段的长度不固定，段式虚拟存储器也有一些缺点：

(1)    主存空间分配比较麻烦。

(2)    容易在段间留下许多碎片，造成存储空间利用率降低。

(3)    由于段长不一定是2的整数次幂，因而不能简单地像分页方式那样用虚拟地址和实存地址的最低若干二进制位作为段内地址，并与段号进行直接拼接，必须用加法操作通过段起址与段内地址的求和运算得到物理地址。因此，段式存储管理比页式存储管理方式需要更多的硬件支持。

# 磁盘

## 磁盘访问延迟

**磁盘访问延迟 = 队列时间 + 控制器时间 +   
 寻道时间 + 旋转时间 + 传输时间**