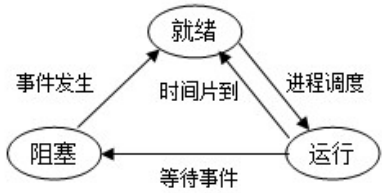


# 第一章

1. 操作系统设计目标：方便性、有效性、便于设计实现维护。
2. 引入多道程序系统的原因：提高 CPU 的利用率。  
特点：在主存同时存放多个作业，使之同时处于运行状态，共享系统中的各种资源。
3. 操作系统基本功能：处理机管理、存储器管理、设备管理、文件管理。
4. 批处理系统特点：吞吐量大、资源利用率高、无法交互、平均周转时间长。  
分时系统特点：同时性、独立性、交互性、及时性。  
实时系统特点：实时性、可靠性、确定性。
5. 衡量 OS 的性能指标：资源利用率、吞吐量、周转时间。
6. 对称多处理：操作系统和用户程序可安排在任何一个处理机上运行，各处理机共享主存和各种 I/O 设备。
7. 操作系统的特性：并发性、共享性、虚拟性、异步性。
8. CPU 工作状态：核心态（操作系统内核程序）、用户态（用户程序）。  
用户态到核心态的转换由硬件完成。  
核心态到用户态的转换由内核程序执行后完成。
9. 系统调用：内核向用户提供的，用来运行系统内核子程序的接口。  
特权指令执行时，CPU 处于核心态。
10. 用户与操作系统的接口：操作接口（命令语言或窗口界面）、  
编程接口（系统调用）。

## 第二、三章

1. 程序顺序执行的特点：串行性、封闭性、可再现性。
2. 进程的四大特性：动态性、独立性、并发性、结构性。
3. 进程控制块的组成部分：进程标识符、状态+调度+存储器管理信息、使用的资源信息、CPU 现场保护区、记账信息、进程间家族关系、进程的链接指针。
4. 进程基本状态：运行态、阻塞态、就绪态。
5. 进程控制：是指系统使用一些具有特定功能的程序段来创建、撤消进程，以及完成进程各状态之间的转换。

```
graph TD; 就绪((就绪)) -- "进程调度" --> 运行((运行)); 运行 -- "时间片到" --> 就绪; 运行 -- "等待事件" --> 阻塞((阻塞)); 阻塞 -- "事件发生" --> 就绪;
```
6. 进程调度的功能：记录系统中各进程的执行状况、选择就绪进程占有 CPU、进行进程上下文的切换。  
方式：非抢先/非剥夺方式（批处理）、抢先/剥夺方式（分时、实时）。  
时机：
  - ①现行进程完成或错误终止；
  - ②提出 I/O 请求，等待 I/O 完成；
  - ③时间片用完或更高优先级进程就绪；
  - ④执行了某种原语操作。
7. 进程调度的算法：先来先服务、最短作业优先、响应比高者优先、优先级调度法、轮转法、多级反馈队列轮转法。
8. 系统对线程的支持：用户级线程、核心级线程、两级组合。
9. 并发进程关系：互斥、同步、前序。
10. 临界区四个准则：互斥使用、让权等待、有空让进、有限等待。
11. 解决进程互斥的办法：开关中断、加锁开锁、软件方法、信号量与 PV 操作。
12. 进程高级通信机制：消息缓冲、信箱、管道、共享主存区等。
13. 死锁产生的必要条件：互斥、保持和等待、非剥夺、循环等待。
14. 解决死锁的方法：忽略、死锁的预防、死锁的避免、死锁的检测和恢复。

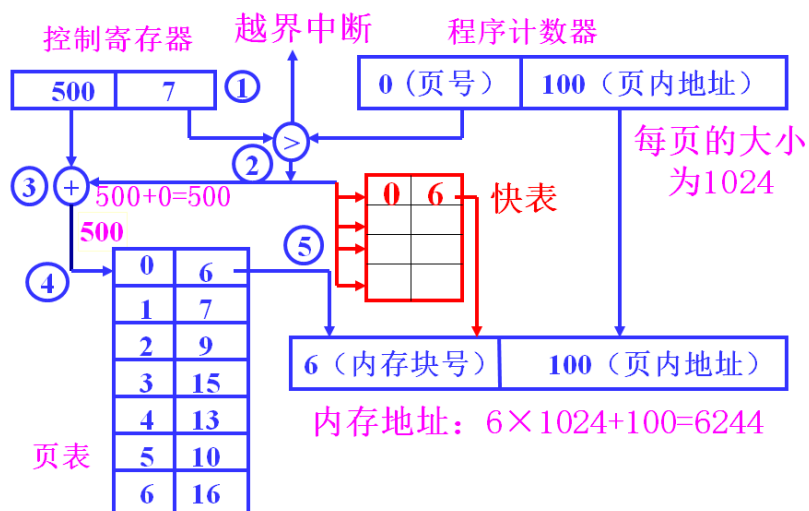
## 第四章

1. 存储器管理的功能：存储器分配、地址转换或重定位、存储器保护、存储器扩充、存储器共享。
2. 地址重定位：把程序地址空间的逻辑地址转换为存储空间的物理地址。  
分类：静态重定位、动态重定位。
3. 内存划分为：用户空间、操作系统空间。存储器针对用户空间进行管理。
4. 存储保护的目地：防止地址越界、正确进行存取。
5. 可变式分区管理分区使用的数据结构：分区说明表、空闲区链表。  
分配算法：首次适应法、最佳适应法、最坏适应法。  
存储器保护方式：动态重定位、基址+限长寄存器。
6. 覆盖的特点：小主存可运行大进程。  
交换的特点：打破了一个程序一旦进入主存，便一直运行到结束的限制。
7. 页表的作用：记录逻辑页与主存块的映射关系。
8. 段式与页式管理的主要区别：

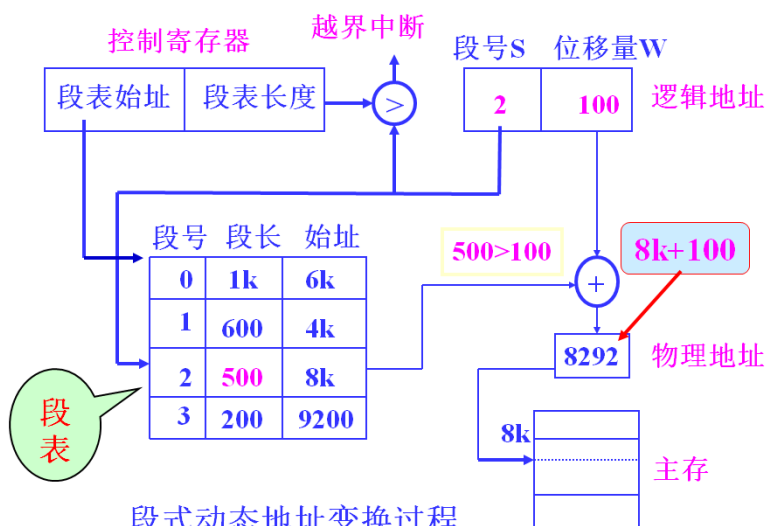
	段式管理	页式管理
划分方式	由用户划分	由硬件划分
大小	不固定	固定
地址空间维数	二维	一维
动态扩充	允许	不允许
碎片	有	无
链接方式	动态链接	静态链接

9. 虚拟存储器：系统构造的非常大的地址空间（可大于主存+辅存容量）。
10. 有效位（状态位）：指示某页是否在主存。  
修改位：指示该页调入主存后是否被修改过。  
访问位（引用位）：指示该页最近是否被访问过。
11. 页面置换算法：最佳置换算法（无法实现）、先进先出置换算法（抖动、Belady异常）、最近最少使用的页面置换算法（复杂）、时钟页面置换算法。
12. 程序访问的局部性：空间局部性、时间局部性。

13. 多级页表结构：页表在内存不必连续存放，且要访问页时才建立页表。
14. 页的共享需要一个专门数据结构。



使用快表后的地址变换过程



段式动态地址变换过程

## 第五章

1. 文件：存储在外部存储器上的具有符号名的相关信息的集合。  
文件系统：操作系统中管理文件的软件机构。
2. 文件系统的功能：管理文件存储器、实现按名存取、具有灵活多样的文件结构和存取方法、提供操作命令、保证安全性、共享。
3. UNIX 系统中的文件分类：普通文件、目录文件、特别文件。
4. 文件目录可分为：一级目录、二级目录、多级目录。
5. 文件控制块：包含了文件的说明信息和管理控制信息。
6. 文件的逻辑结构分为：无结构的字节流式文件、有结构的记录式文件。  
存取方法分为：顺序存取、直接存取（随机存取）。
7. 文件的物理结构：
  - ①连续文件（顺序文件）：文件目录表；
  - ②链接文件：用链接指针链接；
  - ③索引文件：为每个文件建立一张索引表；
  - ④索引顺序文件：Windows 的 MFT（主控文件表）。
8. 对存储空间的管理方法：空白文件目录、空闲块链表、位映像表/位示图。
9. 多级目录的好处：便于文件分类、查找速度快、实现文件共享。
10. 文件的操作命令：创建、删除、打开、关闭、读写、追加、随机存取、得到/设置文件属性、重命名文件。
11. 存取控制表：记录各个域对对象的存取权限。
12. 存储器映射文件：将文件映射到进程地址空间的一个区域，返回虚拟地址，仅当需要对文件存取时，才传输实际的数据。

## 第六章

1. I/O 设备分类：字符设备、块设备、网络通信设备、时钟。
2. I/O 数据传输的控制方式：程序查询、中断、DMA、通道控制。
3. 设备类型：独占设备、共享设备、虚拟设备。  
虚拟设备：在一类设备上模拟另一类设备的 I/O 技术。  
虚拟设备的实现技术：Spooling（假脱机技术），利用可共享磁盘的一部分空间，来模拟独占的 I/O 设备（以空间换时间）。
4. I/O 软件的分层：用户 I/O 接口、独立于设备的软件、设备驱动程序、中断处理程序、I/O 硬件。
5. 设备独立性：指系统设备无论如何改变，用户的程序都不受影响。
6. 一个特定磁盘上的信息如何进行编址：盘面号、磁道号和扇区号（或柱面号、磁头号和扇区号）。
7. 要将磁盘上一个块的信息传输到主存需要系统花费哪些时间：寻道时间、旋转延迟时间、读/写传输时间。
8. 常用磁盘调度算法：先来先服务、最短寻道时间优先、扫描法（SCAN, C\_SCAN, LOOK, C\_LOOK）。

## 第七章

1. Linux 进程的状态：运行、等待（可/不可中断）、暂停、跟踪、僵死、死亡。
2. Linux 进程控制块（`task_struct`）部分结构：
  - `thread_info`：当前进程基本信息；
  - `mm_struct`：指向进程的虚拟内存描述符；
  - `fs_struct`：指向文件系统信息；
  - `files_struct`：指向该进程打开文件信息；
  - `signal_struct`：所接收的信号；
  - `dentry`：指向目录结构的指针。
3. Linux 创建进程的函数：
  - `fork()`：子进程拷贝父进程的数据段、代码段。
  - `clone()`：实现对多线程应用程序的支持，共享进程在内核的数据结构。
  - `vfork()`：产生一个新的子进程，与父进程共享数据段。
4. 0 号进程：所有进程的祖先进程，每个 CPU 都有一个 0 号进程。
  - 1 号进程：由 0 号进程创建，负责完成内核的初始化工作，以及创建和监控在操作系统外层执行的所有用户态进程。
  - 系统会强迫所有的孤儿进程成为 1 号进程的子进程。
5. Linux 进程切换的过程（只发生在核心态）：
  - ①切换页目录表以安装一个新的地址空间；
  - ②切换核心栈和硬件上下文。
6. Linux 进程调度：采用可抢先式的动态优先级调度。
  - 进程调度时机：
    - ①出现更高优先级的实时进程；
    - ②进程执行阻塞原语；
    - ③进程停止运行或被杀死；
    - ④进程自愿放弃处理机；
    - ⑤进程用完时间片。

## 第八章

1. Linux 进程地址空间的管理：32 位机的地址空间为 4GB，前 3G 是私有地址空间，后 1G 是进程的公有地址空间（内核虚空间）。

内核 1GB 虚空间中的前 896MB 用来映射物理内存的前 896MB，后 128MB 的虚空间实现对超过 896MB 的物理内存的映射。

2. Linux 虚拟内存区域描述符 `vm_area_struct` 部分结构：

`vm_mm`：虚拟内存描述符；

`vm_start`：起始地址；

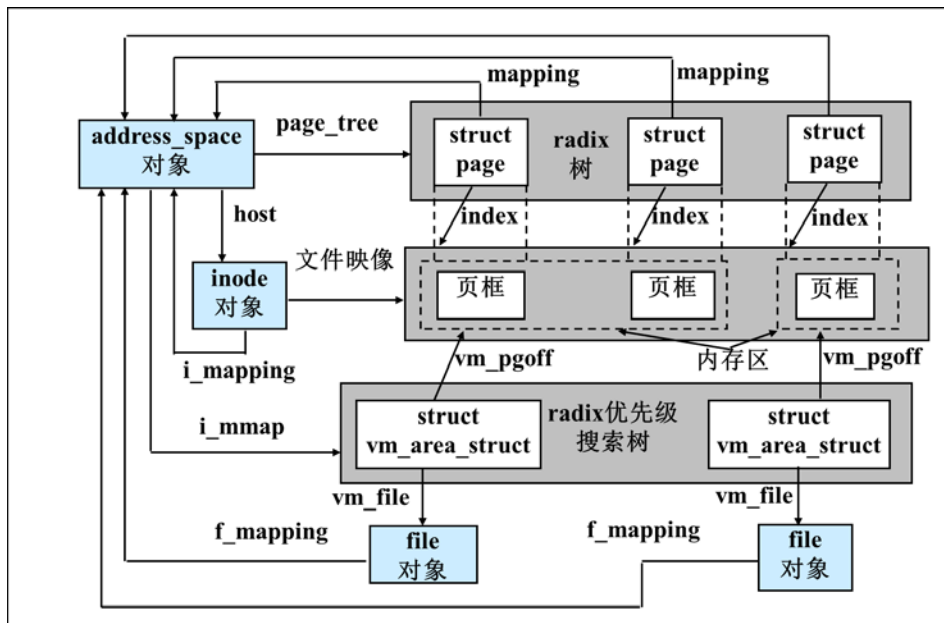
`vm_end`：结束地址；

`vm_next`：单链表；

`vm_rb`：红黑树；

`vm_file`：映射文件时指向文件对象；

`prio_tree_node`：映射文件时，用此结构构造 radix 优先级搜索树。



3. Linux 虚拟内存描述符 `mm_struct` 部分结构：

`mm_rb`：指向红-黑树的根；

`*pgd`：指向页目录表；

`mm_users`：次使用计数器；

`mm_count`：主使用计数器；

`mmlist`：双向链表；

`start_code, end_code`：可执行代码所占用的地址区间。

4. Linux 管理物理内存页框的数据结构 `page`：

`unsigned long flags`：页框状态标志；



\_count: 页框的引用计数;  
 \_mapcount: 对应的页表项数目;  
 private: 由伙伴系统使用;  
 \*mapping: 页高速缓存;  
 index: 在页高速缓存中以页为单位偏移;  
 lru: 链入活动页框链表或非活动;  
 \*virtual: 页框所映射的内核虚地址。

5. Linux 的内存管理区 zone:

ZONE\_DMA: 低于 16MB 的常规内存页框。

ZONE\_NORMAL: 高于 16MB 且低于 896MB 的常规内存页框。

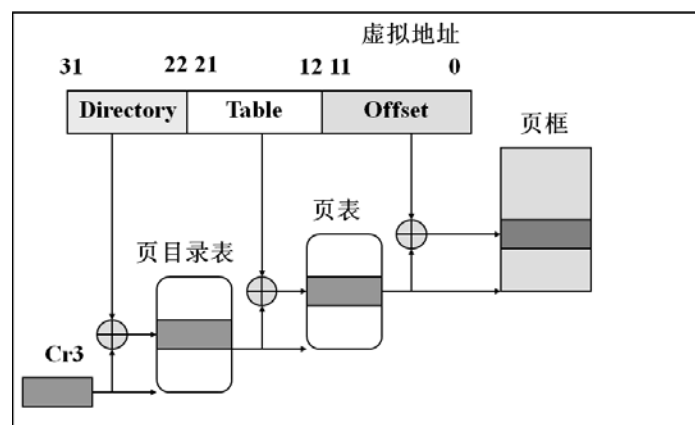
ZONE\_HIGHMEM: 高于 896MB 的物理页框。

6. 伙伴系统: 管理连续的空闲内存页框, 分成 11 个链表 (1, 2, 4……1024)。

以页框为单位, 适合于对大块内存的分配请求。

7. slab 分配器: 为小内存区分配内存, 每个 slab 包含了若干个同类型的对象。

8. Linux 地址转换: 32 位处理机普遍采用二级页表模式, 为每个进程分配一个页目录表, 页表一直推迟到访问页时才建立。

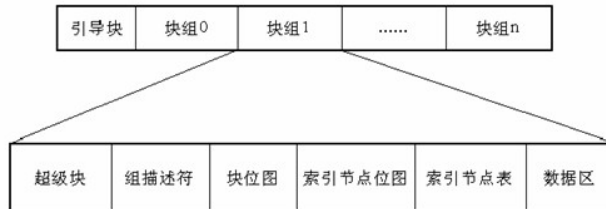


9. 所缺页的存放处:

- ①该页从未被进程访问过, 且没有相应的内存映射;
- ②该页已被进程访问过, 但其内容被临时保存到磁盘交换区上;
- ③该页在非活动页框链表中;
- ④该页正在由其它进程进行 I/O 传输过程中。

# 第九、十章

## 1. Ext2 文件卷的布局:



超级块：存放整个文件卷的资源管理信息。

索引节点：存放文件的管理控制信息（m 块）。

2. Linux 系统把一般的文件目录项分成简单目录项（包含文件名+索引节点号）和索引节点（一个）两部分，前者提高了文件目录的检索速度，后者实现了多条路径共享文件，减少信息冗余。

## 3. 索引节点 ext2\_inode 部分结构:

i\_mode: 文件类型和访问权限;

i\_uid: 拥有者标识符;

i\_size: 以字节为单位的文件长度;

i\_gid: 组标识符;

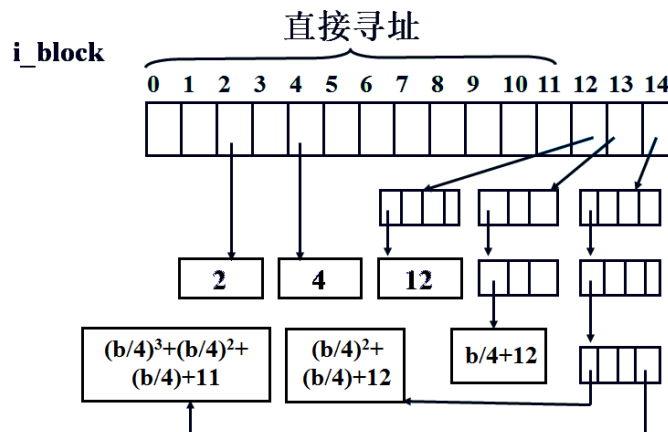
i\_links\_count: 硬链接计数;

i\_block[15]: 索引表,  $15 \times 4B$ ;

i\_blocks: 数据块数;

i\_file\_acl: 访问控制表 ACL。

## 4. 索引表结构:



1-11 块：给出文件最初的 12 个逻辑块号对应的物理块号；

12 块：一次间接索引块， $b$  是盘块大小，每个逻辑块号占 4B；

13 块：二次间接索引块；

14 块：三次间接索引块。

5. 硬链接：即指向文件索引节点的指针，每添加一个硬链接，硬链接计数+1。

符号链接：即一个字符串，内容为指向一个文件/目录的一条路径。

区别：符号链接不与文件索引节点链接，不改变硬链接计数，可跨文件系统。

6. Linux 磁盘管理原则：

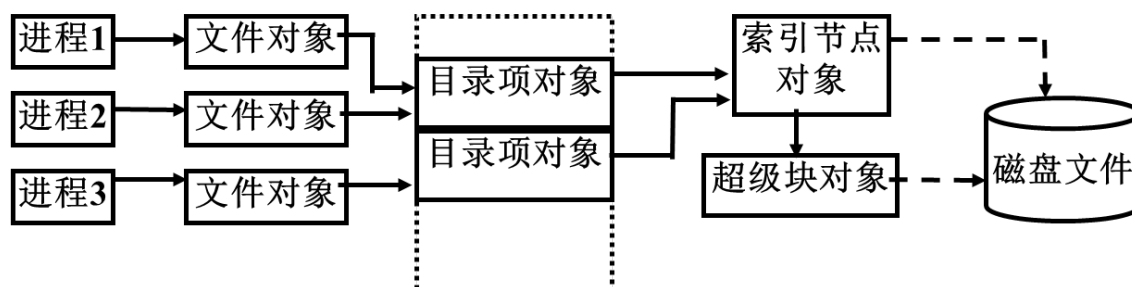
①文件的数据块和其索引节点尽量在同一个块组中；

②文件和它的目录项尽量在同一个块组中；

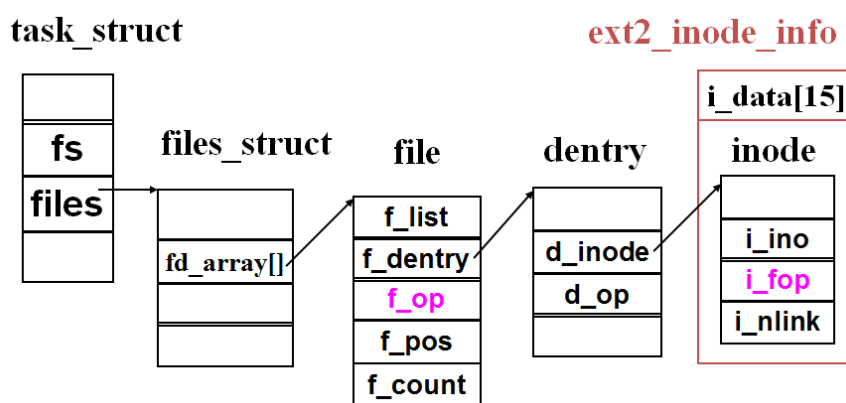
③父目录和子目录尽量在同一个块组中；

④每个文件的数据块尽量连续存放。

7. VFS 的四个主要对象：超级块对象、索引节点对象、目录项对象、文件对象。



8. 进程打开一个磁盘文件要涉及的数据结构：



9. 文件对象 file 部分结构：

list\_head f\_list：文件对象链表；

\*f\_dentry：指向目录项对象；

f\_count：该对象的引用计数；

**f\_pos:** 文件的当前读写位置;

**\*f\_mapping:** 映射;

目录项对象 **dent** 部分结构:

**d\_count:** 引用计数;

**\*d\_ino:** 指向文件的 **ino**;

**\*d\_parent:** 指向父目录项对象;

**d\_alias:** 属于同一 **ino** 的 **dent** 链表。

索引节点对象 **ino** 部分结构:

**i\_sb\_list:** 同一超级块的索引节点链表;

**i\_ino:** 磁盘索引节点号;

**i\_count:** 该对象的引用计数;

**i\_nlink:** 硬链接计数。

**tast\_struct** 部分结构:

**\*fs:** 指向文件系统信息;

**\*files:** 指向进程打开文件信息。

**files\_struct** 部分结构:

**\*\*fd:** 指向文件对象指针数组;

**\*fd\_array[]:** 文件对象指针数组。

10. 安装表作用: 保存安装点与被安装的文件系统信息。

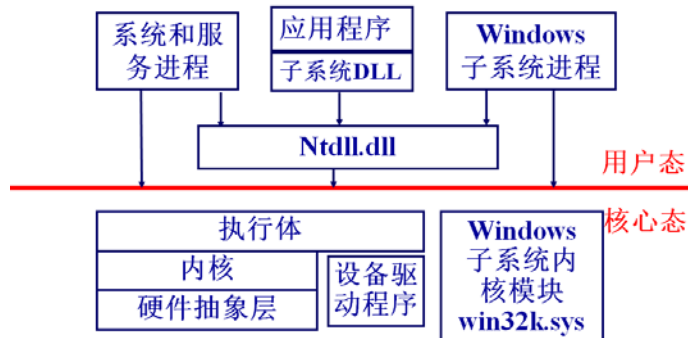
**vfsmount** 部分结构:

**\*mnt\_mountpoint:** 指向安装点的目录项对象;

**\*mnt\_root:** 指向被安装文件系统的根目录。

# 第十四章

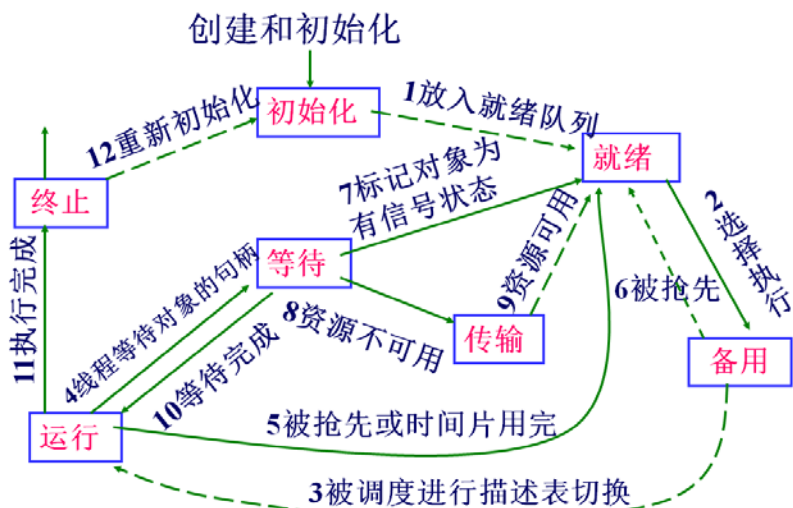
## 1. Windows 体系结构：



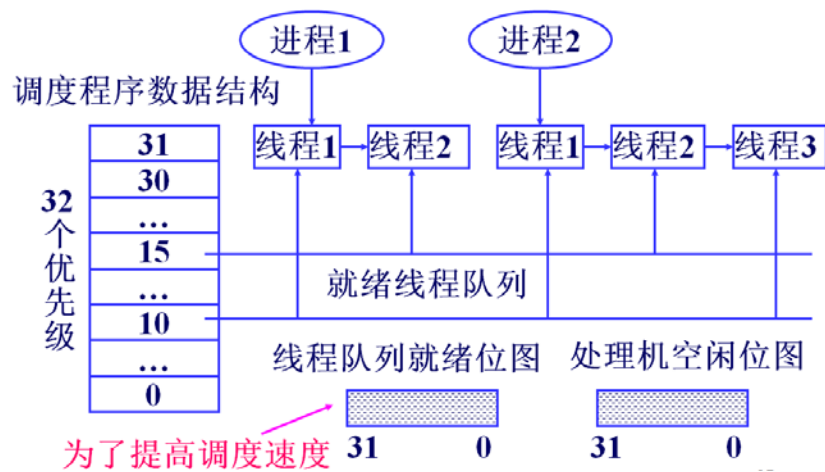
2. Windows 硬件抽象层作用：使内核、设备驱动程序和执行体免受特殊硬件平台差异的影响，增强系统可移植性。
3. Windows 系统机制：陷阱调度、执行体对象管理器、同步（自旋锁、内核调度程序对象）、本地过程调用 LPC。
4. DPC：用来执行不紧急的任务，当 IRQL 降低到 DPC 级别以下时，DPC 中断就产生。产生后依次执行 DPC 队列中的每个例程，直至 DPC 队列为空。  
APC：为用户程序和系统代码提供了一种在特定用户线程环境中执行代码的方法。每个线程都有自己的 APC 队列，APC 队列由内核管理。
5. Windows 的两种类型对象：执行体对象、内核对象。  
执行体对象实现执行体组件：进程管理器、内存管理器、I/O 管理等。  
内核对象：由内核实现的一个初级对象集，仅供执行体使用。  
一个执行体对象可以包含一个或多个内核对象。
6. Windows 多处理机系统提供的同步/互斥机制：
  - ①内核引入自旋锁，实现多处理机互斥机制。
  - ②内核以内核对象的形式给执行体提供其他同步机构，称为“调度程序对象”。
7. Windows 的调度程序对象包括：进程对象、线程对象、事件对象、信号量对象、互斥体对象、可等待的定时器对象及文件对象等。  
每个同步对象都有“有信号”或“无信号”两种状态。
8. Windows 线程实现等待同步对象操作的方法：内核将进程挂起并把它状态设为等待态，直到所等待的调度程序对象句柄由无信号变为有信号状态。

# 第十五章

1. Windows 管理进程和线程的数据结构：执行体进程块 EPROCESS、执行体线程块 ETHREAD、内核进程块 KPROCESS、内核线程块 KTHREAD。
2. 进程对象的部分基本属性：
  - ①内核进程块：包含 Windows 内核调度线程所必须的信息；
  - ②进程 ID：包括进程唯一标识、父进程标识、映像名、进程所在窗口位置；
  - ③访问令牌：用户登录时的系统安全认证；
  - ④存储器管理信息：用 VAD 和指向工作集列表的指针描述；
  - ⑤对象句柄列表：记录进程创建和打开的所有对象；
  - ⑥异常/调试程序窗口：进程异常/调试时的内部通信通道；
3. CreateProcess 主要流程：
  - ①打开 exe 文件，创建区域对象，建立映像与内存之间的映射关系；
  - ②创建执行体进程对象及主线程；
  - ③通知 Win32 子系统，完成一系列的初始化。
4. Windows 线程的状态：就绪状态、备用状态、运行状态、等待状态、传输状态、终止状态、初始化状态。



5. Windows 线程调度：基于优先级的抢先式的多处理机调度系统，优先级相同时按时间片轮转。相关的数据结构：32 个就绪线程队列、32 位线程就绪队列位图、32 位处理机空闲位图。



#### 6. 线程优先级的提升时机：

- ①I/O 操作完成；
- ②信号量或事件等待结束；
- ③前台进程中的线程完成一个等待操作；
- ④由于窗口活动而唤醒图形用户接口线程；
- ⑤线程处于就绪状态超过一定时间，仍未能进入运行状态(处理器饥饿)。

# 第十六章

1. Windows 进程地址空间的布局：32 位的地址空间上，允许每个用户进程占有 4G 虚存空间。低 2G 为私有地址空间，高 2G 为进程公用的操作系统空间。
2. Windows 存储器管理的两个数据结构：

VAD：采用平衡二叉树结构来管理进程私有地址空间，当线程要求分配虚存时，为它建立 VAD 结构。进程页表的构建一直推迟到访问页时才建立。

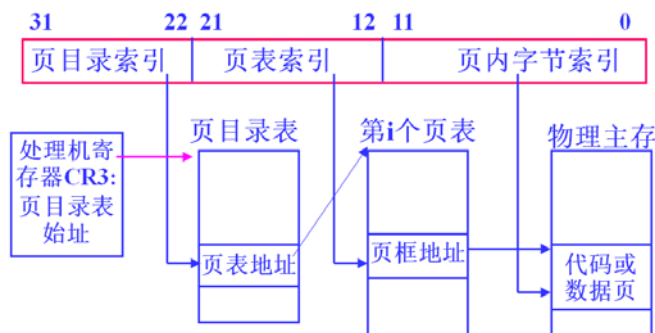
区域对象：是一个可被多个进程共享的存储区,可被多个进程打开。利用区域对象可映射磁盘上的文件（包括页文件），访问这个文件就像访问主存中的一个大数据组，而不需要使用文件的读/写操作。

3. 区域对象的作用：
  - ①利用区域对象将一个可执行文件装入主存；
  - ②使用区域对象可将一个大于进程地址空间的文件映射到进程地址空间；
  - ③缓存管理器利用区域对象访问一个被缓存文件中的数据。
4. Windows 进程私有地址空间的页可能是空闲的（未被使用），或被保留（已预留虚存，还没分配物理主存），或被提交（已分配物理主存或交换区）。
5. Windows 采用二级页表结构，页表和页目录表的结构相同。



6. Windows 的虚拟地址变换过程：

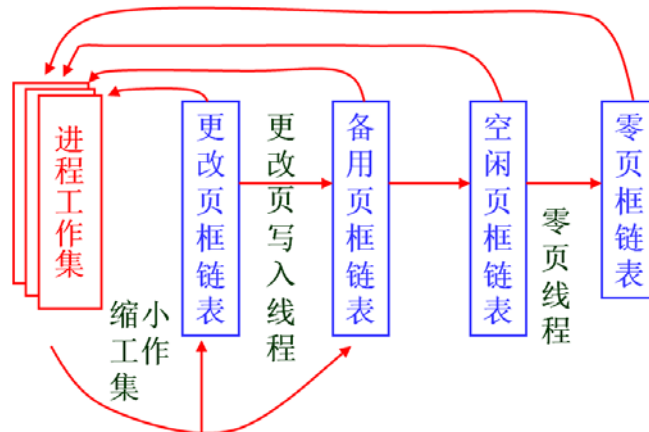
32位的地址被分解为三部分：





7. Windows 管理物理内存的数据结构：页框数据库。页框的 8 种状态：活动、转换、备用、更改、更改不写入、空闲、零初始化、坏页框。

### 页框的状态转换图



8. 原型页表项：区域对象的页表，虚拟页式中用来实现多进程共享页。
9. Windows 采用的页置换策略：
- 多处理器系统：局部先进先出置换策略。
- 单处理器系统：时钟页面置换算法。

# 第十七章

1. Windows 所支持的文件系统：FAT12、FAT16、FAT32、NTFS。

2. NTFS 文件卷结构：分区引导扇区、主控文件表 MFT、文件数据区。

MFT 的作用：MFT 是 NTFS 卷的管理控制中心，包含了卷上所有的文件、目录及空闲未用盘簇的管理信息。小文件或小目录的所有属性常驻在 MFT 中，大文件会由 NTFS 分配一个与 MFT 分开的区域，用来存储属性值。

3. NTFS 文件的物理结构：索引顺序结构。

4. 管理文件的目录结构采用 B-树（也有说法是 B+树）。