# 第三章 进程的描述和控制

1. **进程的概念** 背诵

   - 正在执行的程序
   - 计算机上运行的程序的实例
   - 能够申请在处理器上运行的实体
   - 一个具有以下活动特征的活动单元：一组**指令序列的执行**、一个当前状态和相关的**系统资源集合**

2. **进程的要素**

   - *Identifier* 标识符，进程的唯一标识 **ID**
   - *State* **状态**
   - *priority* **优先级**
   - *Program counter* 程序计数器，即将被执行的下一条指令的地址 **pc**
   - *Memory pointers* 内存指针
   - *Context data* 上下文数据，进程执行时处理器的存储器中的数据等
   - *IO status informa-io* 状态信息，显式的IO请求，分配给进程的*IO设备*和被进程使用的*文件列表*
   - *Accounting information* 审计信息，处理器时间总和，使用的时钟数总和，时间限制，审计号

3. **进程控制块 Process Control Block**

   i. **进程标识**

      - 标识符

        > 存储在进程控制块中的数字标识号

   ii. **处理器状态信息**

      - 用户可见寄存器
      - 控制和状态寄存器
      - 堆栈指针

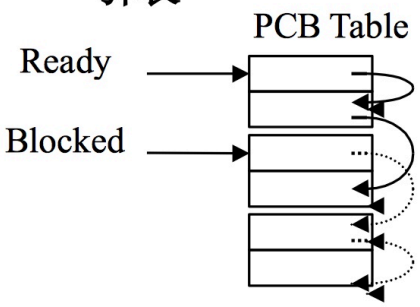
   iii. **进程控制信息**

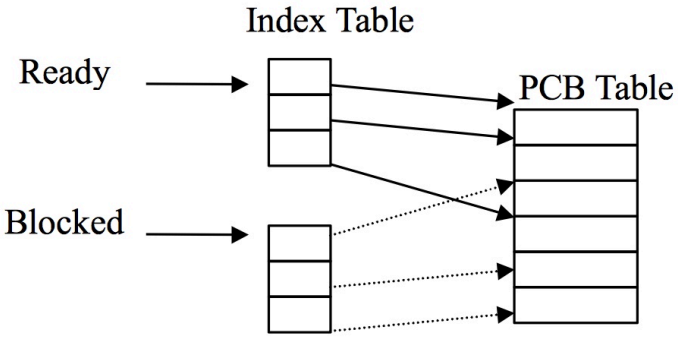
      - 调度和状态信息

        - *进程状态*
        - *优先级*
        - *调度相关信息*
        - *事件*
    - 数据结构
    - 进程间通信
    - 进程特权
    - 存储空间
    - 资源使用权和使用情况

4. **PCB的组织方式**

    - **链表**：同一状态的进程其PCB成一链表，多个状态对应多个不同的链表
    - **索引表**：同一状态的进程归入一个index表，多个状态对应多个不同的索引表
        - **各状态的进行形成不同的索引表：就绪索引表、阻塞索引表**




5. **进程跟踪记录**

> Program Counter 负责

    - 进程运行的指令序列
    - 将处理器从一个进程切换到另一个进程

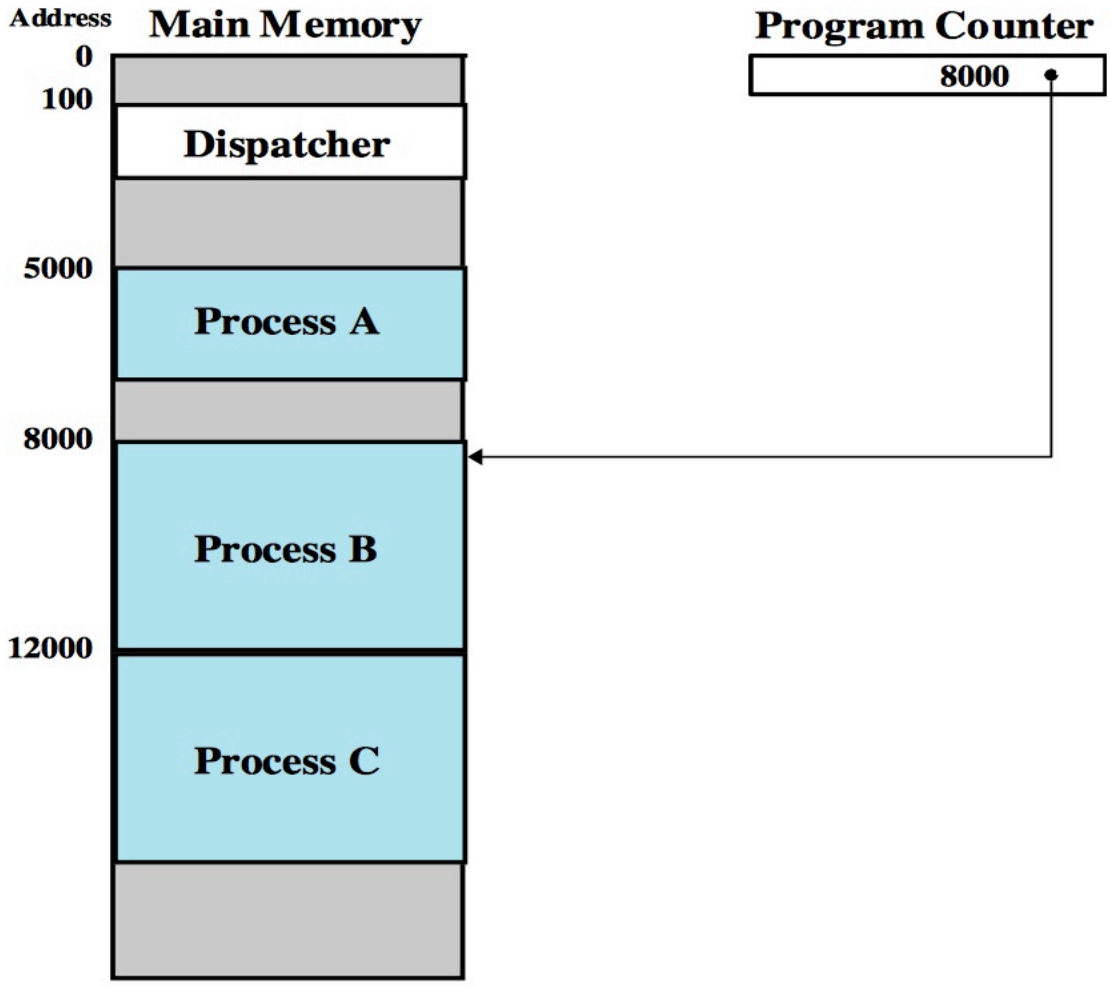



**Figure 3.2 Snapshot of Example Execution (Figure 3.4) at Instruction Cycle 13**

6. **五状态模型** 重点

   - **运行态 Running**：占用处理机资源；处于此状态的进程的数目小于等于 CPU 的数目。
   - **就绪态 Ready**：进程已获得除处理机外的所需资源，等待分配处理机资源；只要分配 CPU 就可执行。
   - **阻塞态 Blocked**：由于进程等待某种条件（如 I/O 操作或进程同步），在条件满足之前无法继续执行。 该事件发生前即使把处理机分配给该进程，也无法运行。
   - **创建态 New**：进程刚创建，**但还不能运行** (一种可能的原因是 OS 对并发进程数的限制)； 如：分配和建立 PCB 表项（可能有数目限制）、建立资源表格（如打开文件表）并分配资源，加载程序并建立地址空间表。
   - **结束态 Exit**：进程已结束运行，**回收除 PCB 之外的其他资源，并让其他进程从 PCB 中收集有关信息**（如记帐，将退出码 exit code 传递给父进程）。

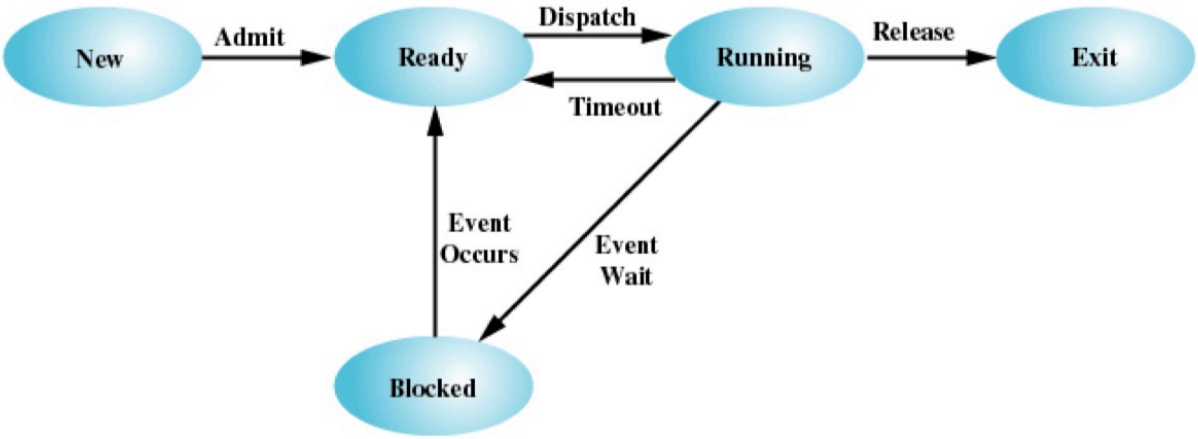
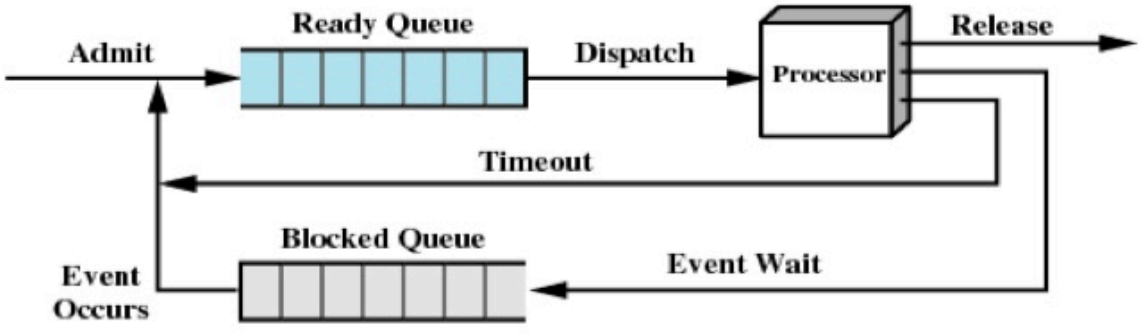


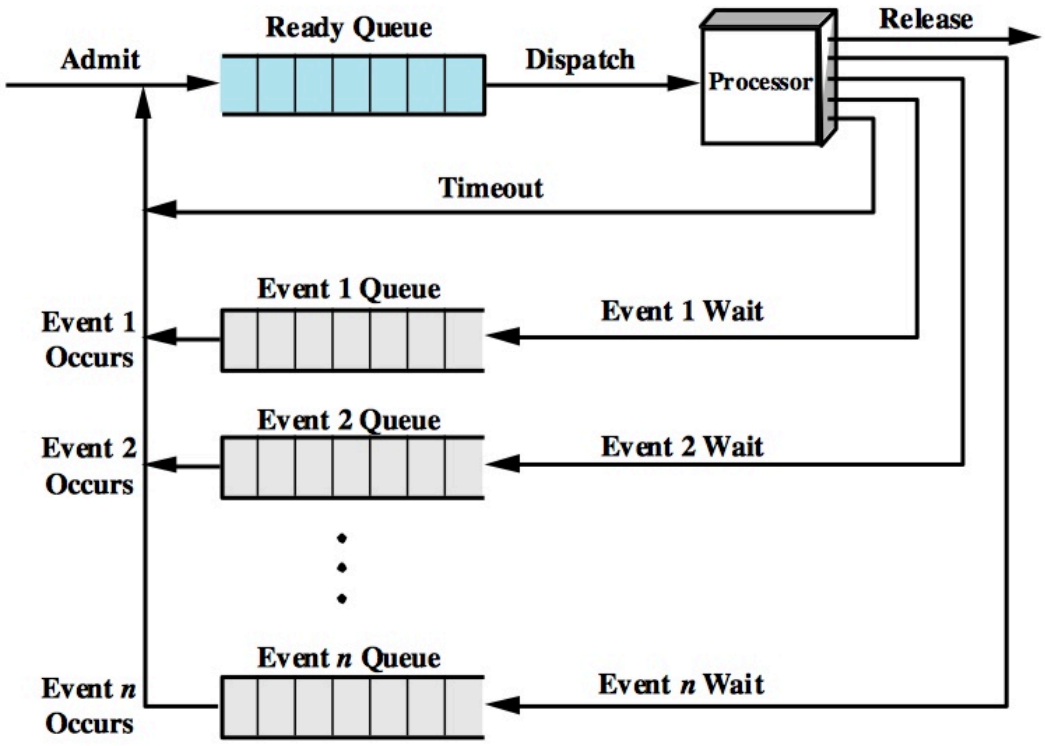

**Figure 3.6 Five-State Process Model**

# 使用两个队列




**(a) Single blocked queue**

# 多个阻塞队列




**(b) Multiple blocked queues**

7. **五状态切换**

   - **创建新进程**：创建一个新进程，以运行一个程序。可能的原因为：用户登录、OS创建以提供某项服务、批处理作业。

     > New

   - **收容** (*Admit*, 也称为提交 )： *收容一个新进程，进入就绪状态*。由于性能、内存、进程总数等原因，系统会限制并发进程总数。

     > New -> Ready

   - **调度运行** (*Dispatch*)： 从就绪进程表中选择一个进程，进入运行状态；

     > Ready -> Running

   - **释放* (*Release*)： 由于进程完成或失败而中止进程运行，进入结束状态；

     - **运行到结束**：分为正常退出 Exit 和异常退出 abort （执行超时或内存不够，非法指令或地址，I/O 失败，被其他进程所终止）
     - **就绪或阻塞到结束**：可能的原因有：父进程可在任何时间中止子进程；

> Running -> Exit

- **超时**（*Timeout*）：由于用完时间片或高优先进程就绪等导致进程暂停运行；

> Running -> Ready

- **事件等待**（*Event Wait*）：进程要求的事件未出现而进入阻塞；可能的原因包括：申请系统服务或资源、通信、 I/O 操作等；

> Running -> Block

- **事件出现**（*Event Occurs*）：进程等待的事件出现；如：操作完成、申请成功等；

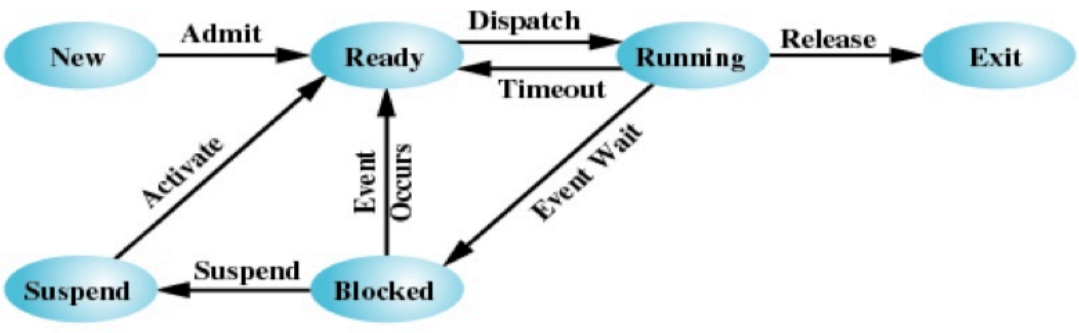
> Block -> Ready

8. **挂起**（*Suspend*） 重点

> 进程优先级的引入，使得一些低优先级进程可能等待较长时间，从而被**对换至外存**。这样做的目的是：

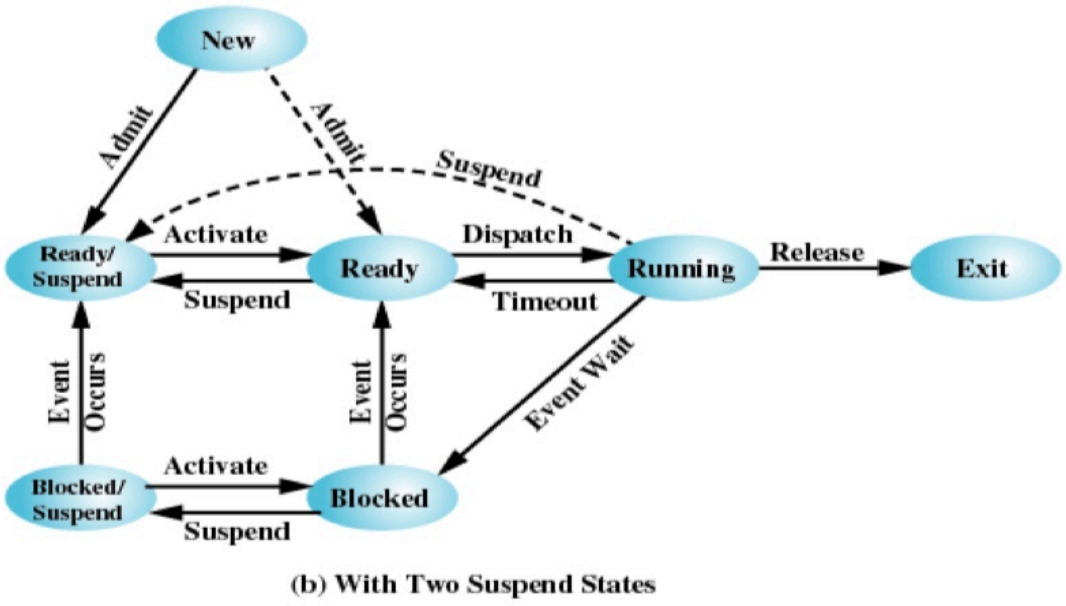
- 提高**处理机效率**：就绪进程表为空时，要提交新进程，以提高处理机效率；
- 为运行进程**提供足够内存**：资源紧张时，暂停某些进程，如：CPU繁忙（或实时任务执行），内存紧张
- **用于调试**：在调试时，挂起被调试进程（从而对其地址空间进行读写）

# 单挂起状态模型




(a) With One Suspend State

# 双挂起状态模型




(b) With Two Suspend States

**Figure 3.9 Process State Transition Diagram with Suspend States**

9. **挂起模型的状态 New**

   - **就绪状态** (Ready)： 进程在内存且可立即进入运行状态；
   - **阻塞状态** (Blocked)： 进程在内存并等待某事件的出现；
   - **阻塞挂起状态** （ Blocked, suspend ）： **进程在外存并等待某事件的出现；**
   - **就绪挂起状态** （ Ready, suspend ）： **进程在外存，但只要进入内存，即可运行；**

10. **挂起模型间的转换**

    - **挂起**：把一个进程从内存转到外存；可能有以下几种情况：

      - **阻塞到阻塞挂起**：没有进程处于就绪状态或就绪进程要求更多内存资源时，会进行这种转换，以提交新进程或运行就绪进程；

        > Block -> Blocked/Suspend

      - **就绪到就绪挂起**：当有高优先级阻塞（系统认为会很快就绪的）进程和低优先级就绪进程时，系统会选择挂起低优先级就绪进程；

> Ready -> Ready/Suspend

- **运行到就绪挂起**：对抢先式分时系统，当有高优先级阻塞挂起进程因事件出现而进入就绪挂起时，系统可能会把运行进程转到就绪挂起状态；

> Running -> Ready/Suspend

- **激活**（Activate）：

> **把一个进程从外存转到内存；可能有以下几种情况：**

- **就绪挂起到就绪**：没有就绪进程或挂起就绪进程优先级高于就绪进程时，会进行这种转换；

> Ready/Suspend -> Ready

- **阻塞挂起到阻塞**：当一个进程释放足够内存时，系统会把一个高优先级阻塞挂起（系统认为会很快出现所等待的事件）进程；

> Block/Suspend -> Block

- **事件出现**（Event Occurs）：进程等待的事件出现；如：操作完成、申请成功等；可能的情况有：

- **阻塞到就绪**：针对内存进程的事件出现；

> Block -> Ready

- **阻塞挂起到就绪挂起**：针对外存进程的事件出现；

> Block/Suspend -> Ready/Suspend

- **收容** (Admit)：收容一个新进程，进入就绪状态或就绪挂起状态。进入就绪挂起的原因是系统希望保持一个大的就绪进程表（挂起和非挂起）；

> New -> Ready
> New -> Ready/Suspend

11. **操作系统控制结构**

- 每个进程和资源的当前状态
- 方法：**构造并且维护所管理的每个实体的信息表**
- 一般说操作系统维护四种信息表：

> 内存表、IO表、文件表和进程表

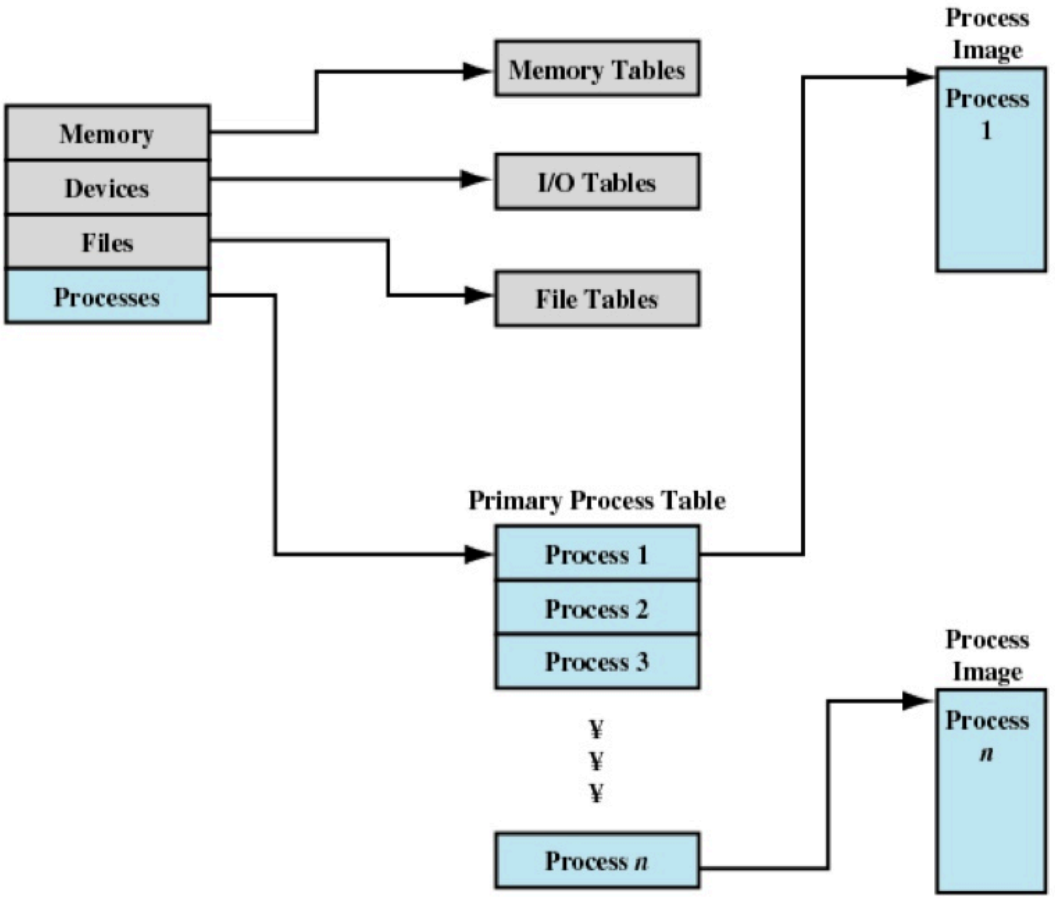



**Figure 3.11 General Structure of Operating System Control Tables**

12. **进程上下文**

> 进程上下文是对**进程执行活动全过程**的**静态描述**。进程上下文由进程的用户地址空间内容、硬件寄存器内容及与该进程相关的**核心数据结构**组成。

- **用户级上下文**：进程的用户地址空间（包括用户栈各层次），包括用户正文段、用户数据段和用户栈

  > 数据 All address

- **寄存器级上下文**：程序寄存器、处理机状态寄存器、栈指针、通用寄存器的值

  > cpu所需

- **系统级上下文**

  - 静态部分（PCB 和资源表格）

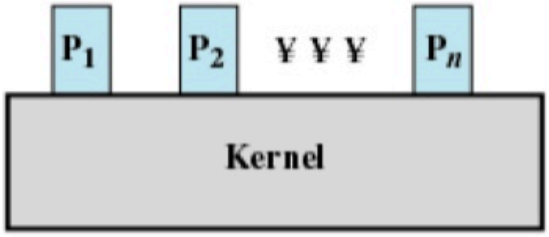
        - 动态部分：核心栈（核心过程的栈结构，不同进程在调用相同核心过程时有不同核心栈）

13. **进程表**

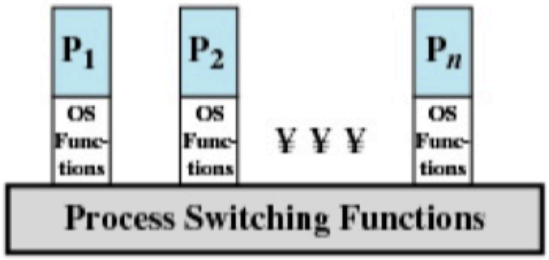
    - 两类重要信息
        - 进程的确切位置
        - 进程的各种属性（PCB中的）
    - **进程映像：操作系统里面实际的进程**

14. 操作系统的运行模式

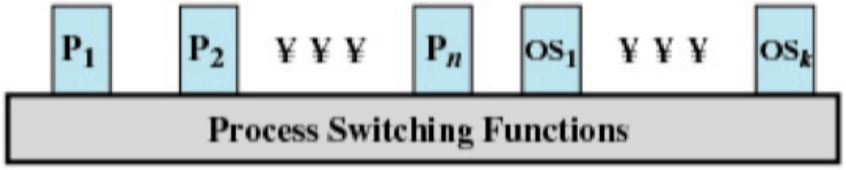
    i. *无进程内核*
    ii. *在用进程内执行*
    iii. *基于进程的操作系统*




(a) Separate kernel




(b) OS functions execute within user processes




(c) OS functions execute as separate processes

**Figure 3.15 Relationship Between Operating System and User Processes**