经典教材《计算机操作系统》**最新版**

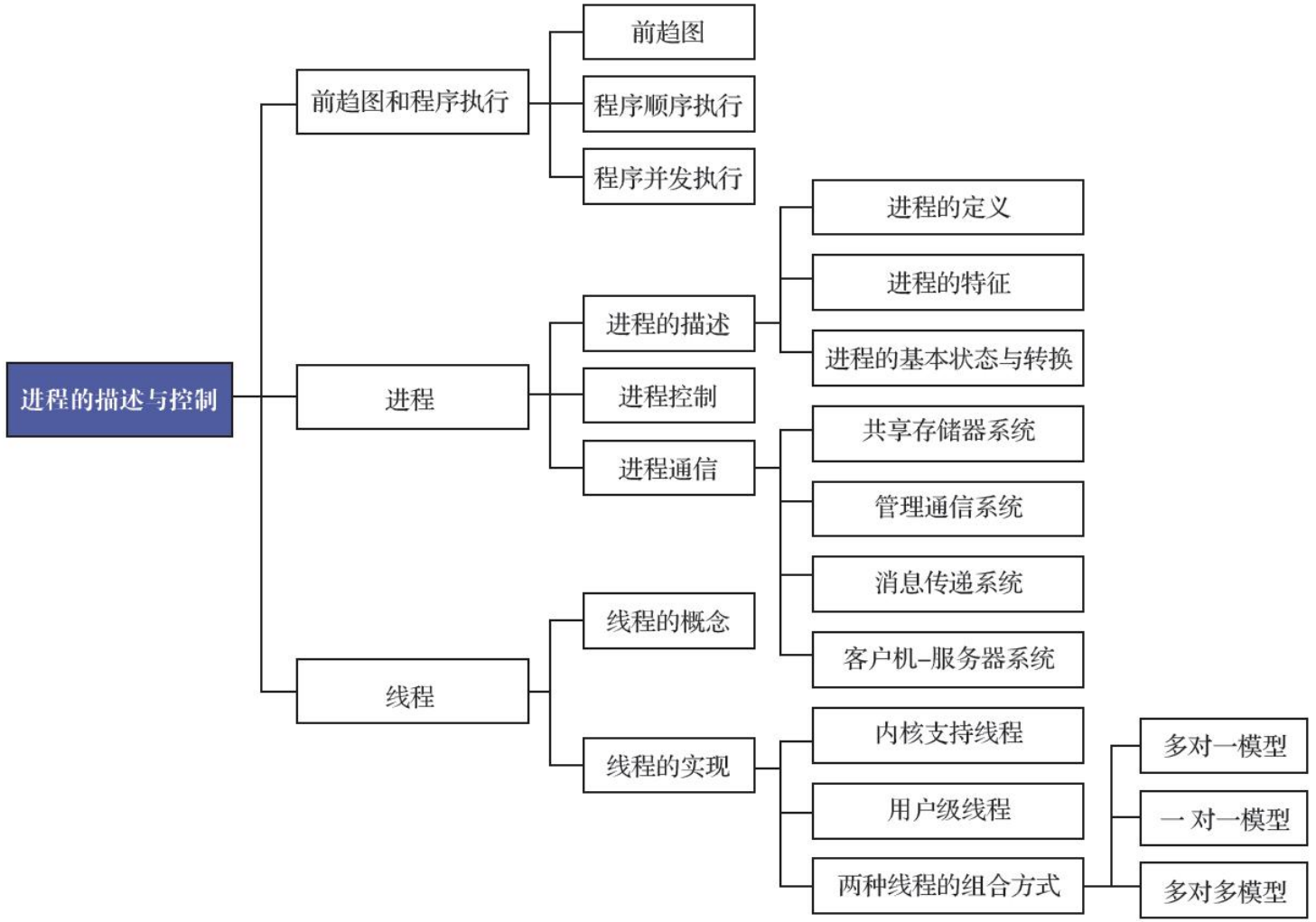
第2章 进程的描述与控制

主讲教师：李灵慧

人民邮电出版社
POSTS & TELECOM PRESS

第2章知识导图

第1章	操作系统引论
第2章	**进程的描述与控制**
第3章	处理机调度与死锁
第4章	进程同步
第5章	存储器管理
第6章	虚拟存储器
第7章	输入/输出系统
第8章	文件管理
第9章	磁盘存储器管理
第10章	多处理机操作系统
第11章	虚拟化和云计算
第12章	保护和安全



内容导航：

第2章 进程的描述与控制

前驱图是什么

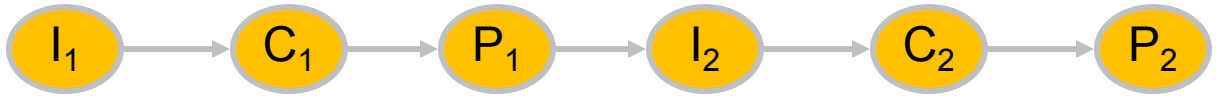
程序顺序执行（早期操作系统）

- 一个较大的程序通常都由若干个程序段组成
- 程序在执行时，必须按照某种先后次序逐个执行，仅当前一操作执行完后，才能执行后继操作。

前趋图 – 程序执行过程

- 有向无循环图，用于描述进程之间执行的先后顺序
- 结点表示进程或程序段，有向边表示前趋关系

程序抽象：输入I、计算C、输出P，输入是计算的前驱，输入完才能计算，计算是输出的前驱



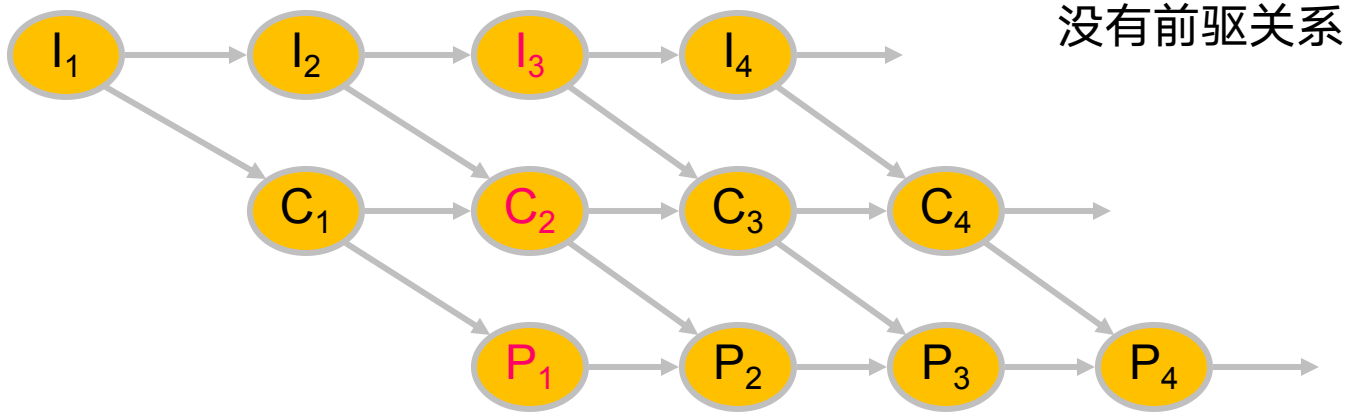
程序顺序执行时的前趋图

前趋关系：$I_i \rightarrow C_i \rightarrow P_i$

程序执行

程序并发执行

- 采用多道程序技术，将多个程序同时装入内存，使之并发运行。



I2和C1可以并发执行，二者没有前驱关系

程序并发执行时的前趋图

前趋关系：$I_i \rightarrow C_i$，$I_i \rightarrow I_{i+1}$，$C_i \rightarrow P_i$，$C_i \rightarrow C_{i+1}$，$P_i \rightarrow P_{i+1}$

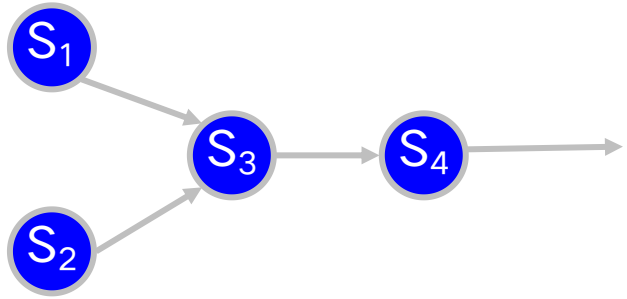
例：程序段如下：

S_1：　a:=x+2

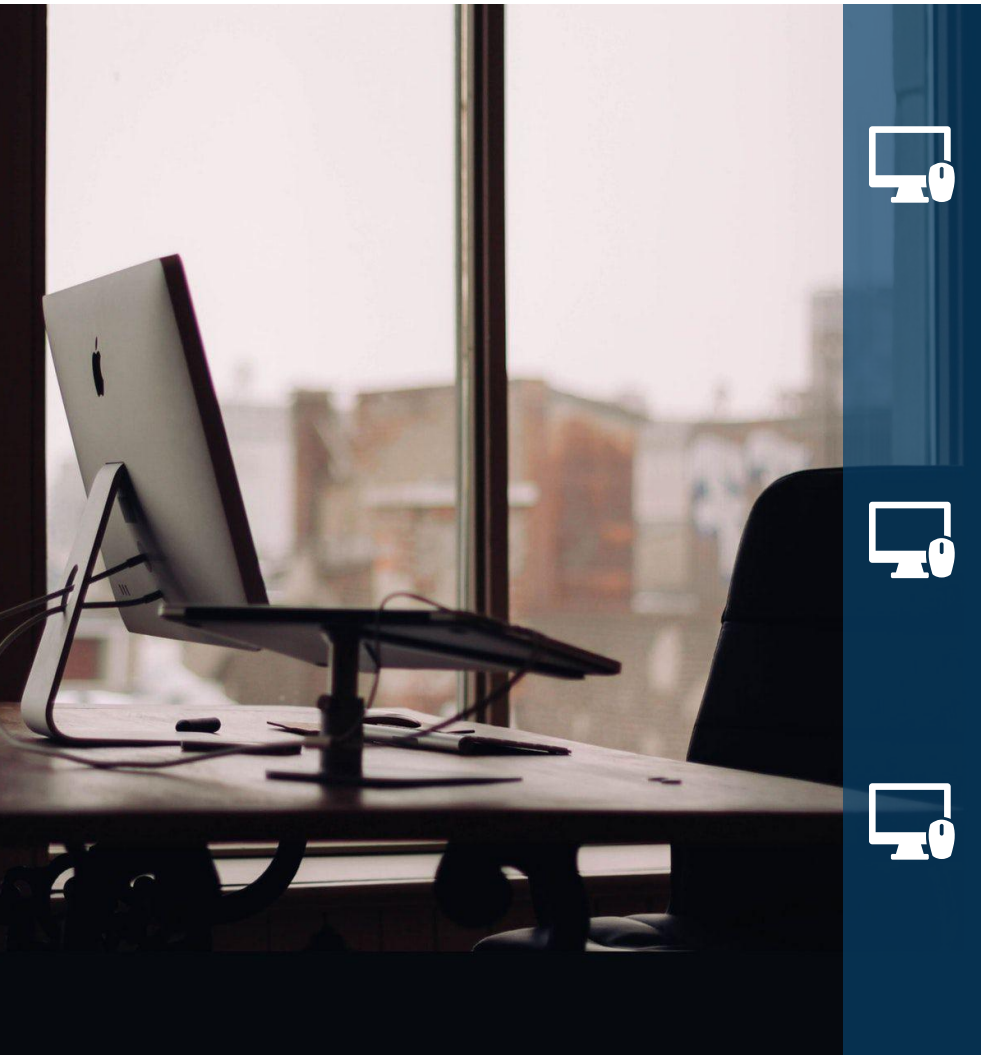
S_2：　b:=y+4

S_3：　c:=a+b

S_4：　d:=c+b



程序并发执行的特征



间断性

- 并发程序之间相互制约关系，程序走走停停。
- 执行——暂停执行——执行。

失去封闭性

- 多个程序共享全机资源。
- 执行状态受外界因素影响，程序的执行失去了封闭性。

不可再现性

- 程序经过多次执行后，虽然其执行时的环境和初始条件都相同，但得到的结果却各不相同。

不可再现性的例子

循环程序A

…

N:=N+1;

…

循环程序B

…

print(N);

N:=0;

赋值

循环程序A和B共享一个变量N，他们以不同的速度运行（并发执行）。(假定某时刻变量N的值为n）。

- N:=N+1　在print(N)和N:=0之前，此时得到的N值分别为n+1,n+1,0
- N:=N+1　在print(N)和N:=0之后，此时得到的N值分别为n, 0 , 1
- N:=N+1　在print(N)和N:=0之间，此时得到的N值分别为n,n+1,0

结论：从最后一个值来看，N的值失去了可再现性。

内容导航：

第2章 进程的描述与控制

进程的定义和特征

几种典型定义

- 进程是程序的一次执行。
- 进程是一个程序及其数据在处理机上顺序执行时所发生的活动。
- 进程是程序在一个数据集合上运行的过程，它是系统进行资源分配和调度的一个独立单位。

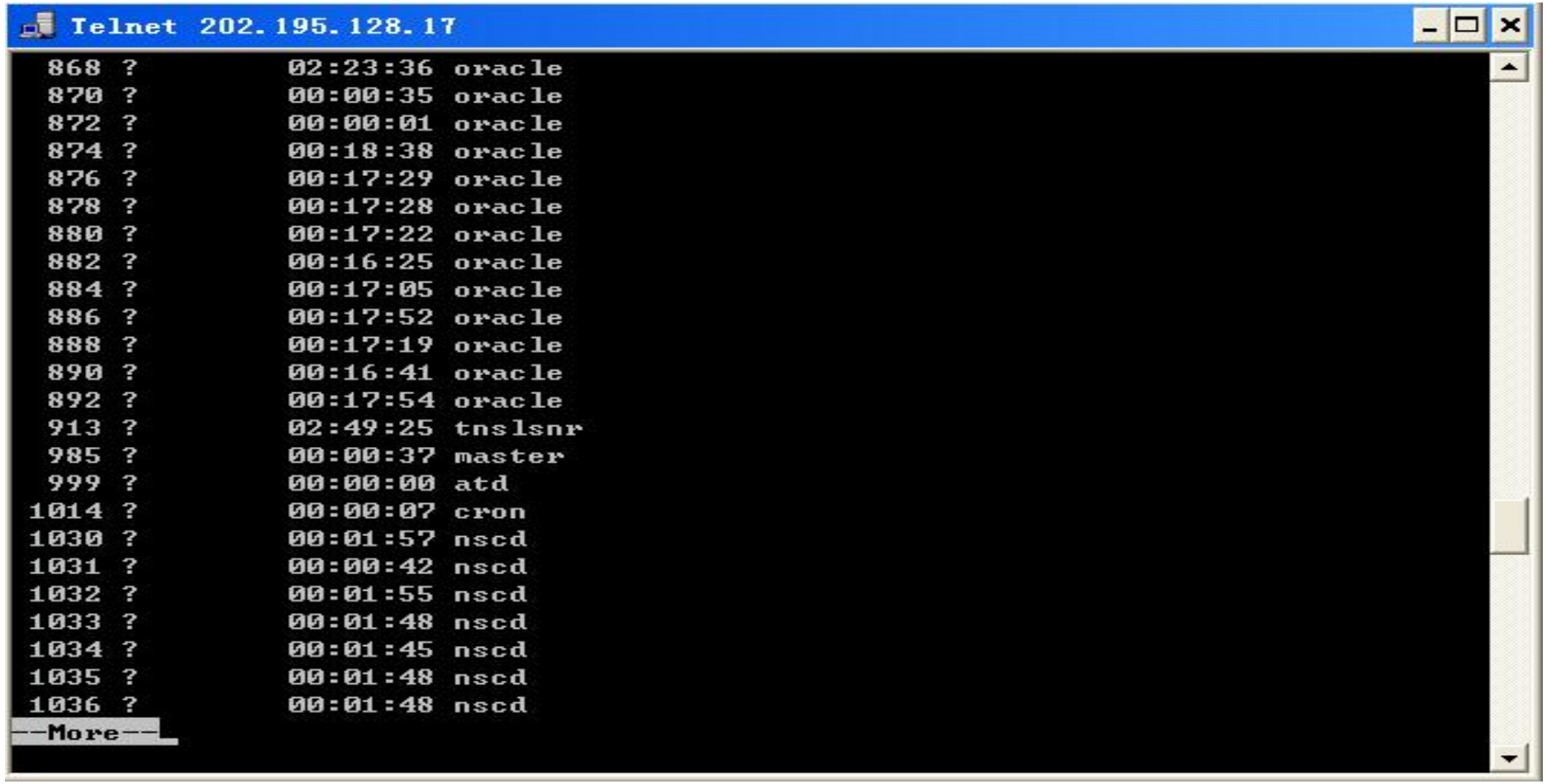
进程定义（本课程）：

- 进程是进程实体的运行过程，是系统进行资源分配和调度的一个独立单位。进程是程序的执行过程，又是CPU和资源调度的单位。

进程控制块(process control block, PCB)

- 专门的数据结构，与进程一一对应。有一个进程就有一个PCB,管理进程就是管理PCB。

进程例子： Suse Linux



进程例子： Windows

进程的特征

动态性
（最基本的特征）

- 生命期
- 程序的执行过程，创建、执行、消亡过程

并发性
(Concurrence)

- 多个进程共存于内存中，一段时间内同时运行

独立性

- 进程实体是一个能独立运行的基本单位
- 是系统中独立获得资源和独立调度的基本单位

异步性

- 按各自独立的、不可预知的速度向前推进

OS的基本特征并发性、共享性、虚拟性、异步性

进程和程序的区别

进程是程序的一个实例，
是程序的一次执行。

进程是活动的，
生命周期的；
程序是静态的，
是一段代码。

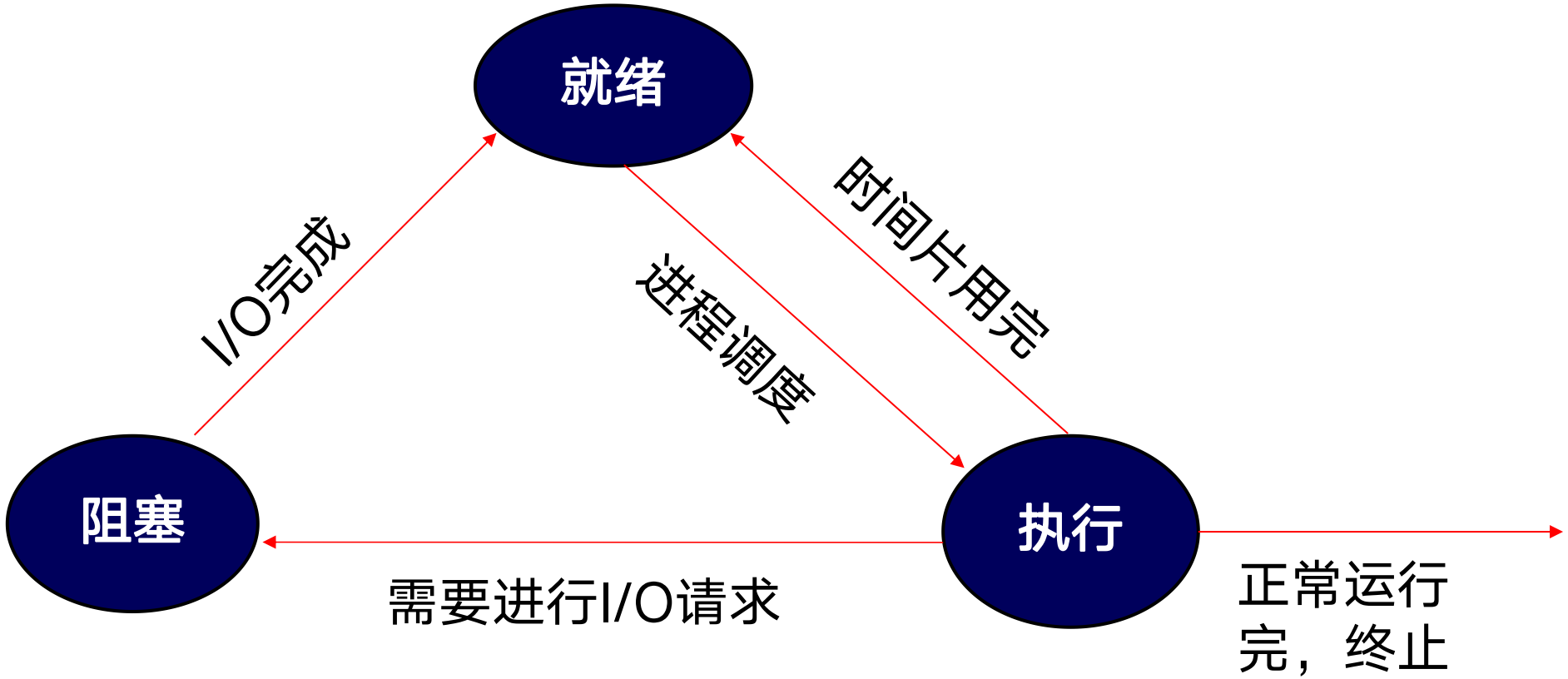
程序是进程的代码部分，
进程还包括数据、PCB等。

进程在内存中，
程序在外存中。

进程的状态及转换

- 就绪状态：完事具备，只欠“东风（等待CPU调度）”，一个就绪队列
  - 一个较大的程序通常都由若干个程序段组成
  - 程序在执行时，必须按照某种先后次序逐个执行，仅当前一操作执行完后，才能执行后继操作。
- 执行状态：已获得CPU，正在执行的状态
  - 单处理机：一个进程处于执行状态
  - 多处理机：多个进程处于执行状态
- 阻塞状态：进程等待某些事件发生的状态
  - 正在执行的进程由于发生某事件而暂时无法继续执行的状态
  - 典型事件：请求I/O、申请缓冲空间
  - 根据阻塞原因，设置多个阻塞队列

进程状态及其转换



3个基本状态以及转换关系

随堂讨论

表 4.2　　跟踪进程状态：CPU 和 I/O

时间	Process0	Process1	注
1	运行	就绪	
2	运行	就绪	
3	运行	就绪	Process0 发起 I/O
4	阻塞	运行	Process0 被阻塞
5	阻塞	运行	所以 Process1 运行
6	阻塞	运行	
7	就绪	运行	I/O 完成
8	就绪	运行	Process1 现在完成
9	运行	—	
10	运行	—	Process0 现在完成

随堂讨论

1. 问题1：为什么不能从阻塞态变为运行态呢？

2. 问题2：为什么不能从就绪态变为阻塞态呢？

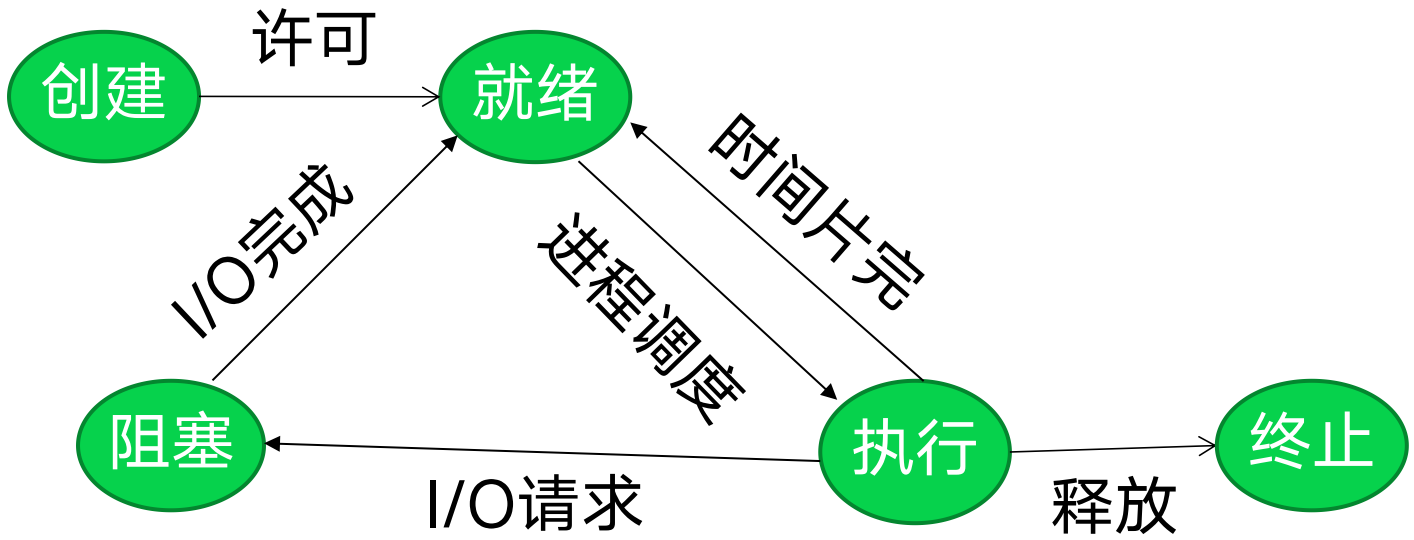
创建状态和终止状态

01 创建状态

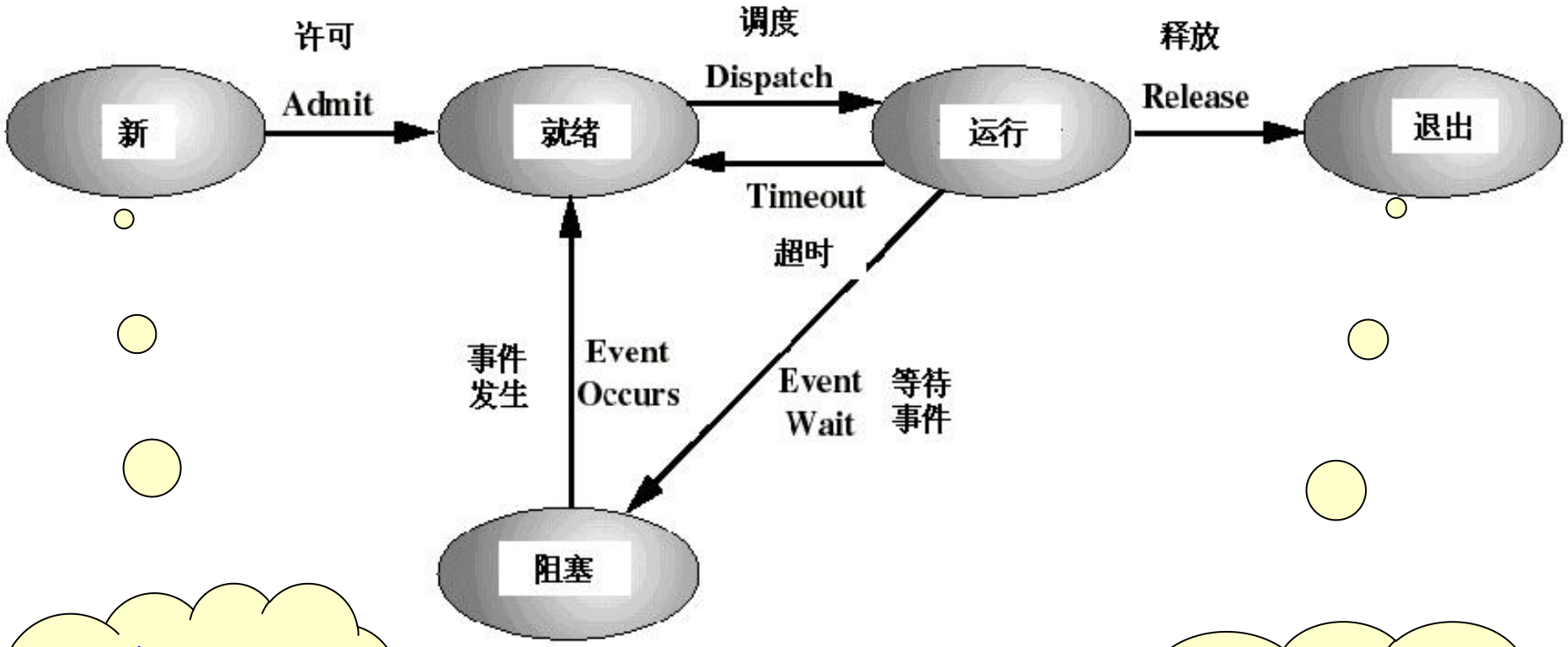
- 申请一个空白PCB；填写PCB；分配资源；设置就绪状态插入就绪队列

02 终止状态

- 等待OS善后，等待资源回收；
- 收回PCB



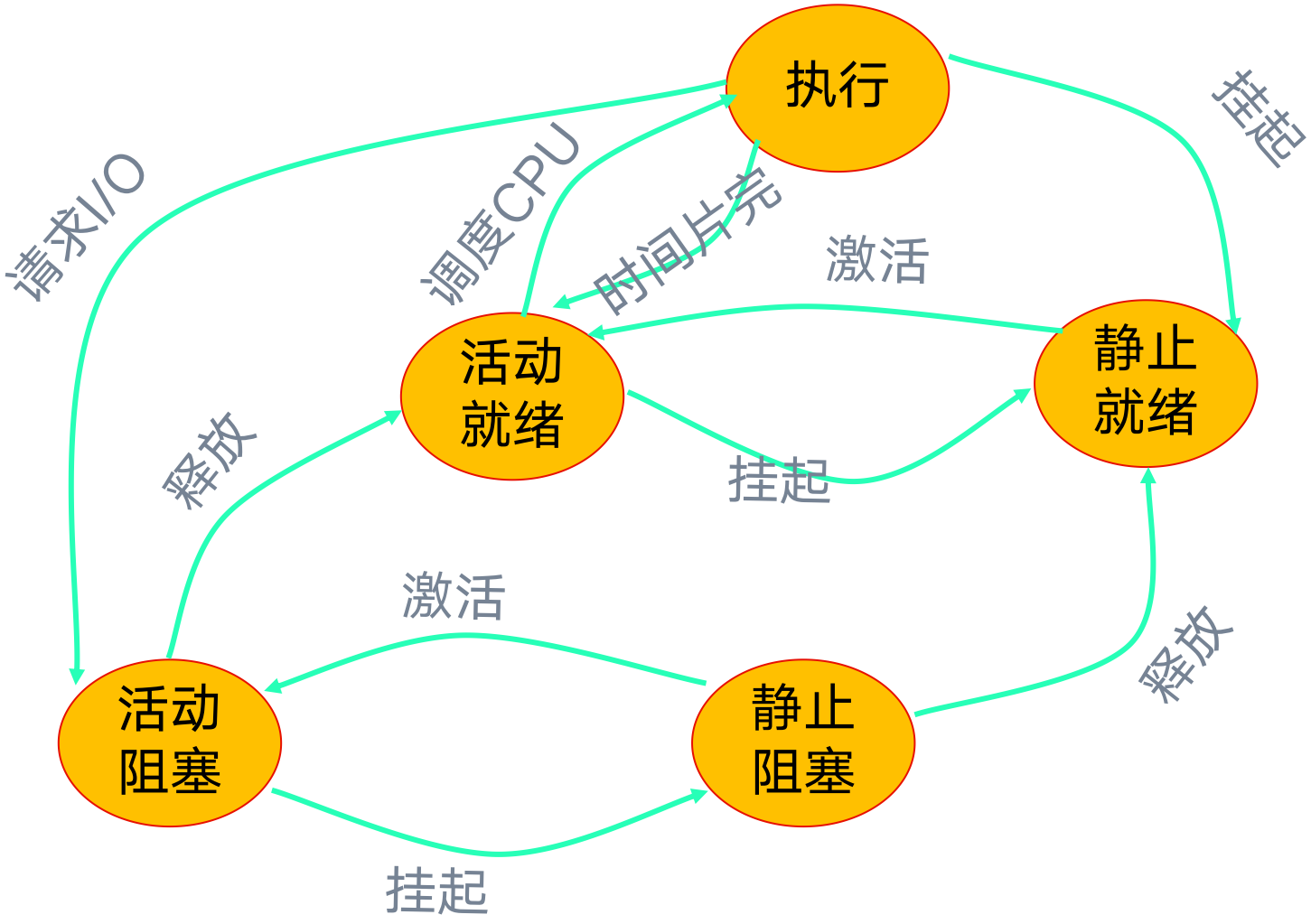
进程状态的转换



是一个进程刚刚建立，但还未将它送入就绪队列时的状态

一个进程已经正常结束或异常结束。

引入挂起状态的状态转换



进程控制块PCB(1)

PCB是进程的一部分，是操作系统中最重要的记录型数据结构，是进程存在的唯一标志，常驻内存。

PCB的作用：

- 作为独立运行基本单位的标志；
- 能实现间断性运行方式；
- 提供进程管理所需要的信息；
- 提供进程调度所需要的信息；
- 实现与其他进程的同步与通信。

进程控制块PCB(2)

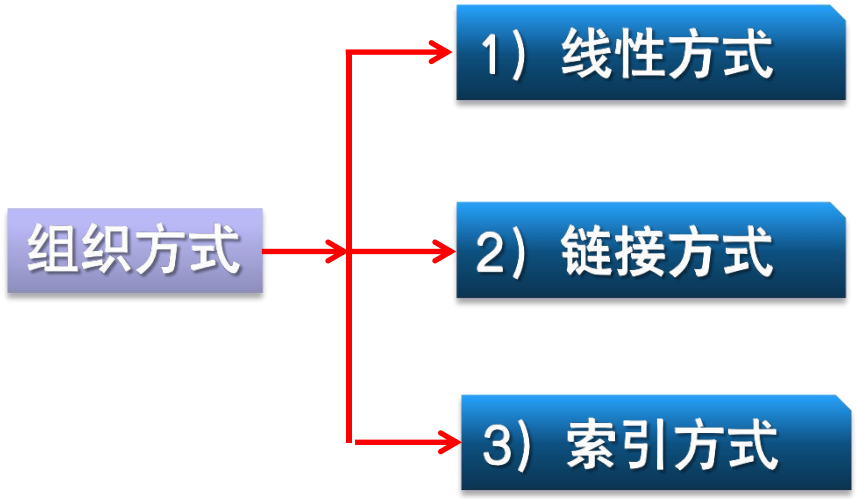
PCB的信息

- 进程标识符，例如PID
- 处理机状态：现场信息，寄存器信息
- 进程调度信息：进程状态、进程优先级
- 进程控制信息：程序和数据地址等

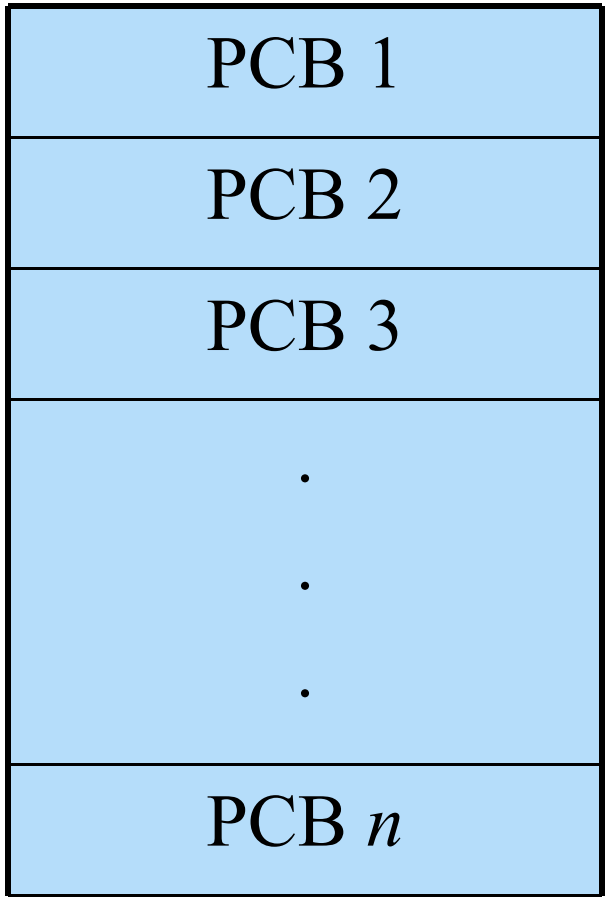
进程

思考：系统中可能拥有数十个、数百个乃至数千个PCB，如何组织？

进程



PCB线性组织方式

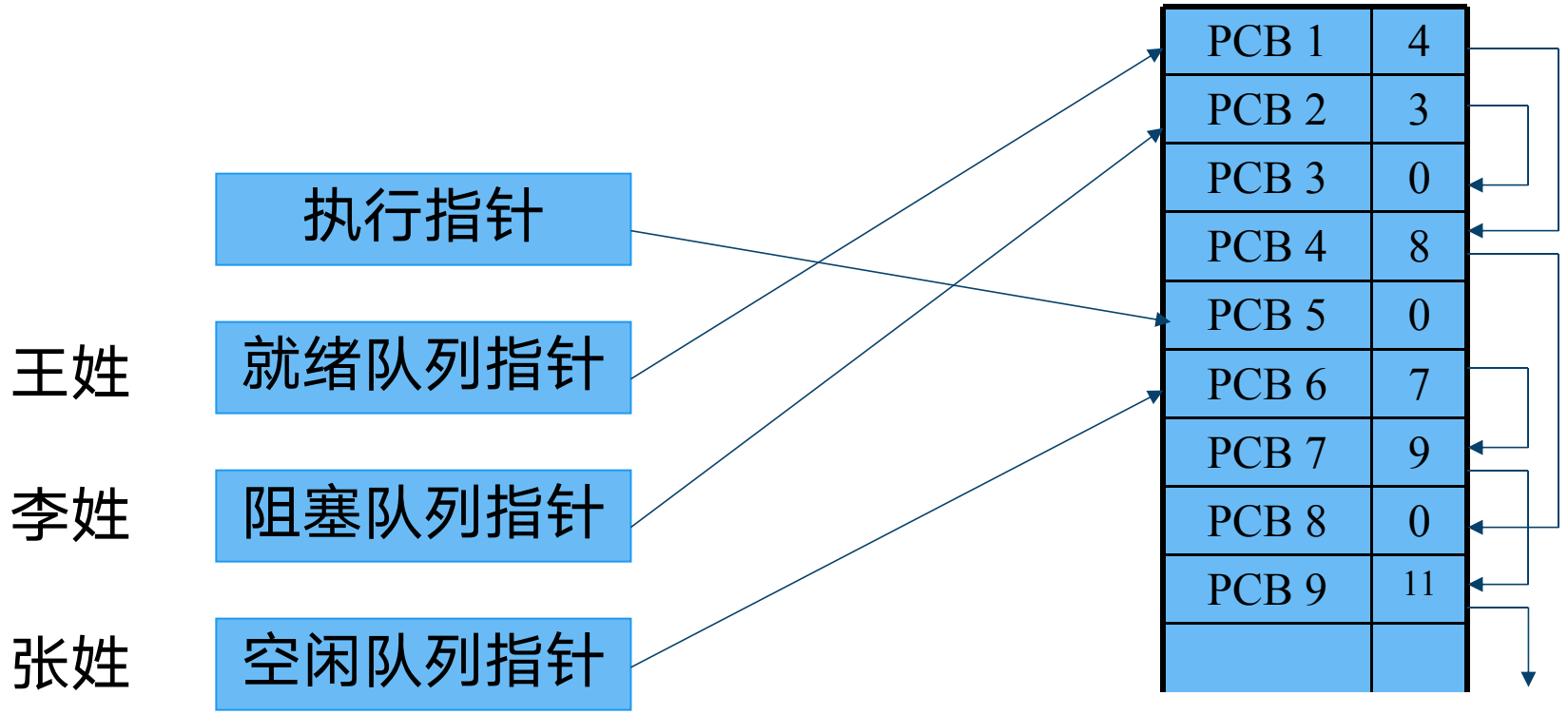


班级里所有的同学个人信息表

（个人信息）

1. 所有PCB组织在一张线形表，表的首址放在内存专用区域
2. 实现简单（列一个表格就可以）
3. 查找效率低（想想每次签到，找自己的名字）

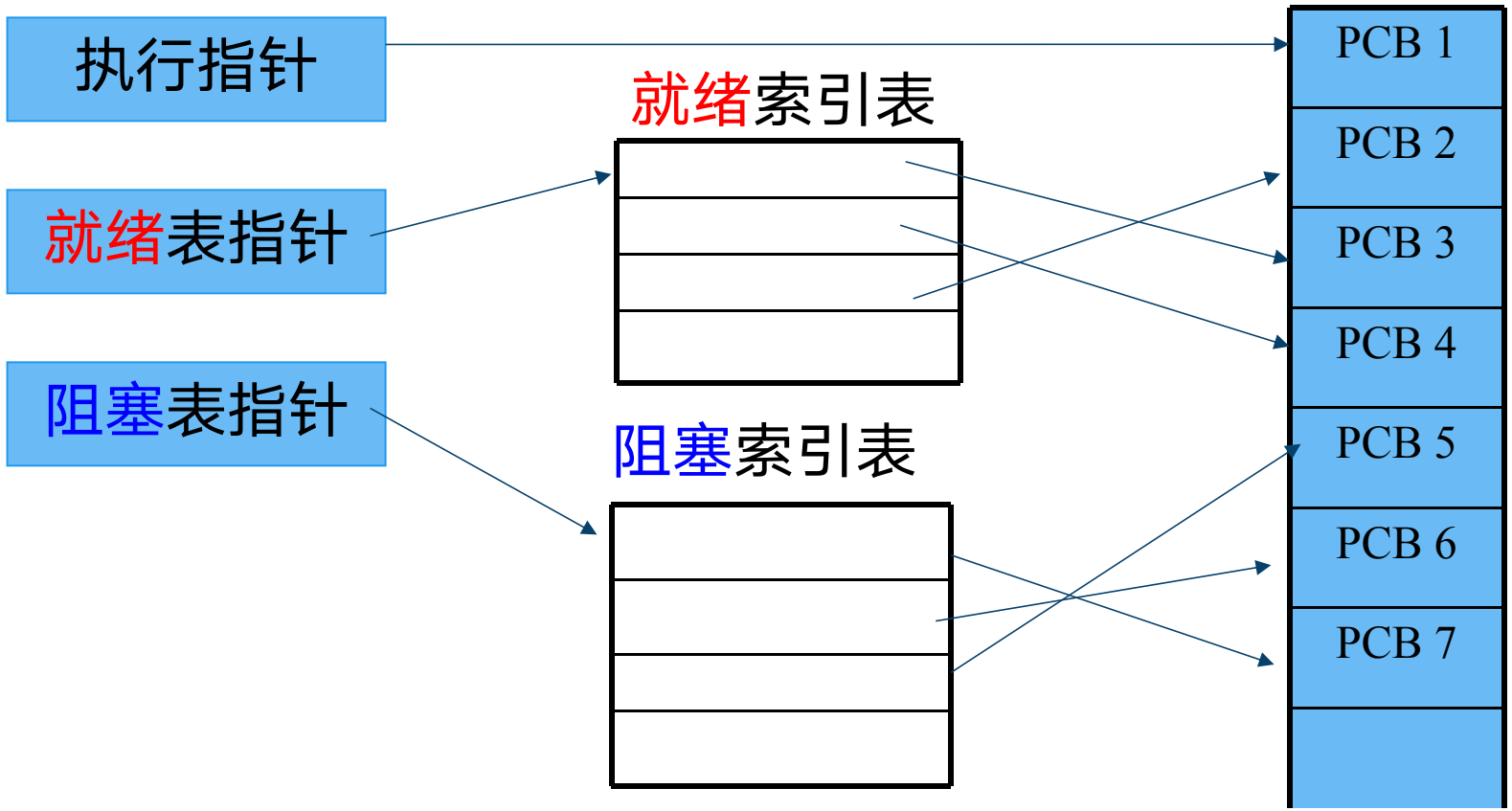
PCB链接组织方式



实现复杂（前一个同学要记住后一个同学的所有个人信息）

查找效率较高（超链接），最起码比一个一个查表快

PCB索引组织方式



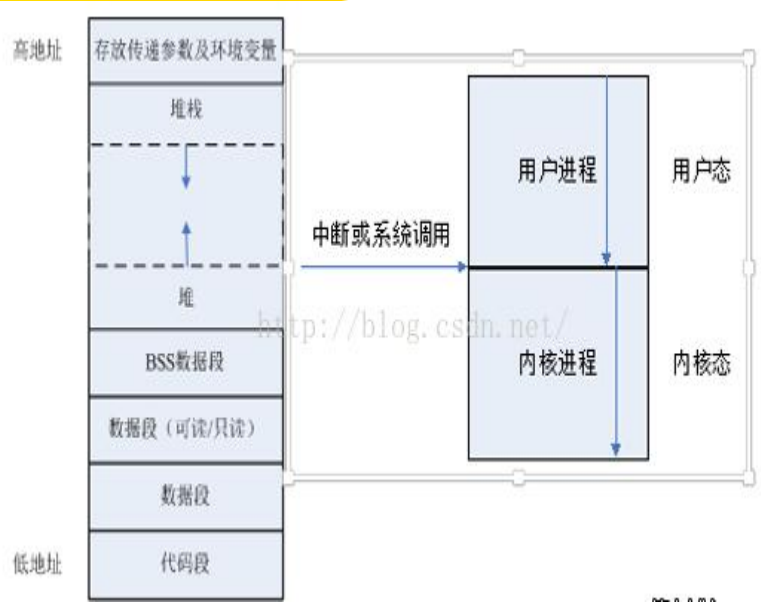
综合线性和链接组织的优缺点

内容导航：

第2章 进程的描述与控制

进程控制

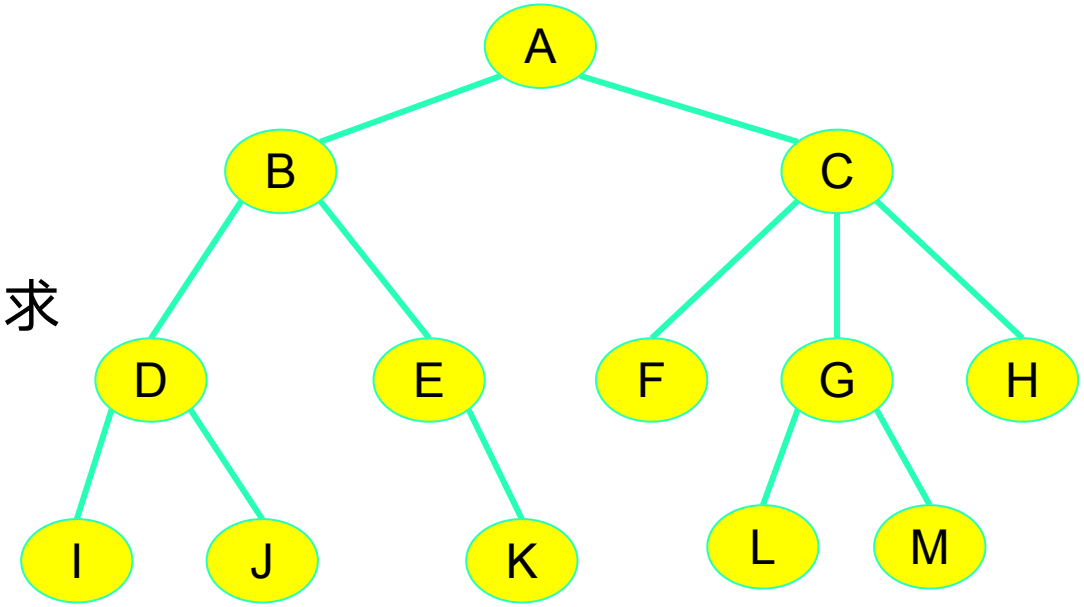
- 进程管理最基本的功能
- 一般由OS内核中的原语实现
- 包括：



进程创建　进程终止　进程阻塞与唤醒　进程挂起与激活

进程创建

- 进程具有层次结构
- 引起进程创建的事件
  - 用户登录、作业调度、提供服务、应用请求
- 进程图
  - 描述进程家族关系的有向树

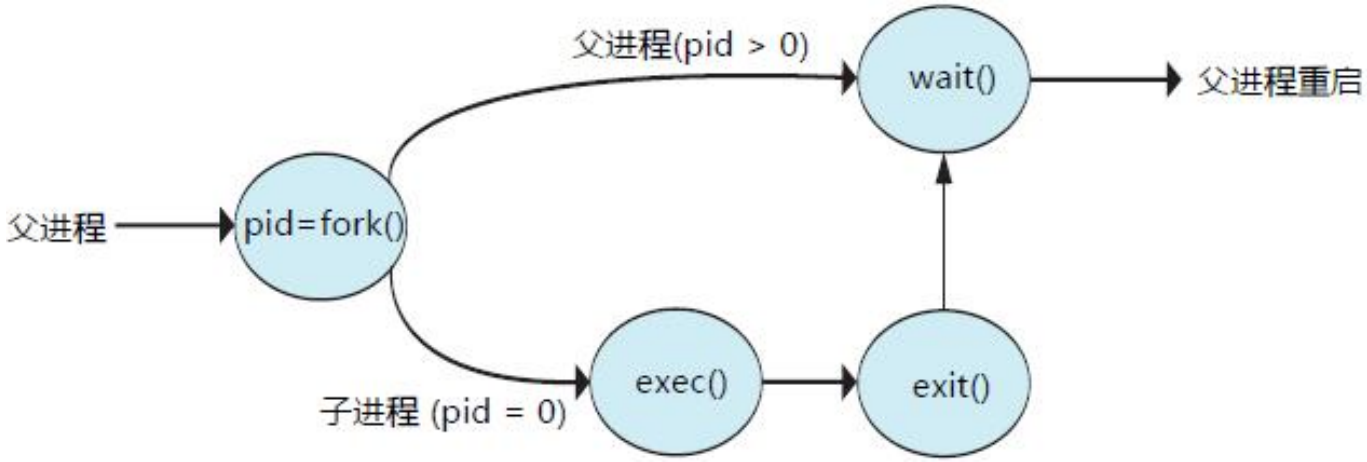


- 进程创建过程：

① 申请空白PCB；　② 分配所需资源；　③ 初始化PCB；　④ 插入就绪队列。

进程创建-UNIX实例

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
int main()
{
    pid_t pid;
    /* fork a child process */
    pid = fork();
    if (pid < 0) { /* error occurred */\
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { /* child process */
        execlp("/bin/ls",  "ls",  NULL);
    }
    else { /* parent process */
        /* parent will wait for the child to complete */
        wait(NULL);
        printf("Child Complete");
    }
    return 0;
}
```



进程终止

引起进程终止的事件：正常结束、异常结束、外界干预

进程的终止过程：

1. 根据被终止进程的标识符，从PCB集合中检索出该进程的PCB，从中读出该进程的状态；
2. 若被终止进程正处于执行状态，应立即终止该进程的执行，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；
3. 若该进程还有子孙进程，还应将其所有子进程予以终止；
4. 将该进程所拥有的全部资源，或者归还给其父进程或系统；
5. 将被终止进程（PCB）从所在队列中移去。

进程的阻塞与唤醒

引起进程阻塞和唤醒的事件

- 向系统请求共享资源失败；等待某种操作的完成；新数据尚未到达；等待新任务的到达。

进程阻塞过程

- 阻塞原语Block()。
- 进程的阻塞是进程自身的一种主动行为。
- 具体过程：停止执行；状态由执行改为阻塞；将PCB插入阻塞队列。

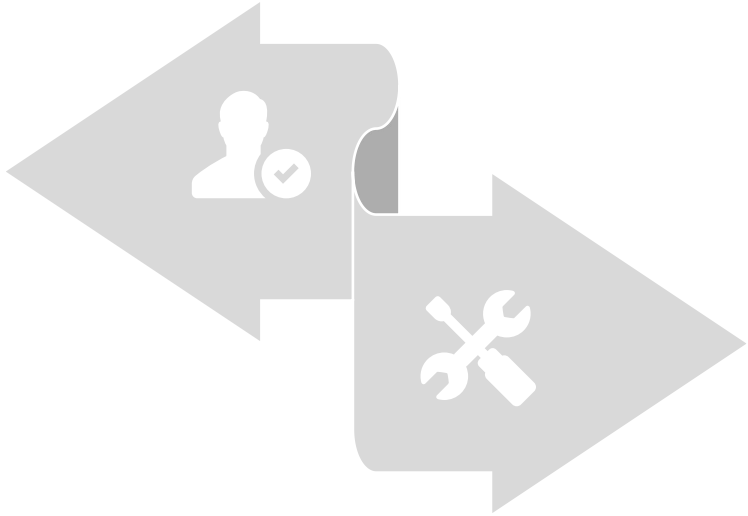
进程唤醒过程

- 唤醒原语Wakeup()。
- 具体过程：从阻塞队列中移出；状态由阻塞改为就绪；将PCB插入就绪队列。
- 必须成对使用Block和Wakeup原语。

进程的挂起与激活

进程的挂起

- Suspend()原语
- 执行过程



进程的激活过程

- Active()原语
- 执行过程

第二次作业

简答题

1 2 3 4 5 6 7 8

9 10 11 12 13 14 15 16

17 18 19 20

综合应用题

21 22

标黄色为本次作业