華中科技大學
计算机科学与技术学院
School of Computer Science & Technology, HUST

操作系统原理

第4章 进程及进程管理

华中科技大学计算机学院 谢美意

目 录

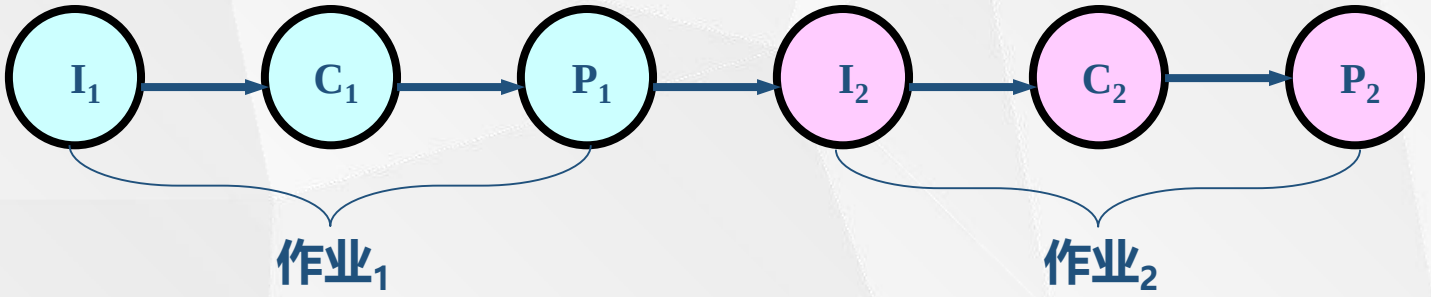
CONTENT

華中科技大學
计算机科学与技术学院
School of Computer Science & Technology, HUST

进程的引入

顺序程序

- 程序的一次执行过程称为一个计算，它由许多简单操作所组成。
- 一个计算的若干操作必须按照严格的先后次序顺序地执行，这个计算过程就是程序的顺序执行过程。



I：输入操作；C：计算操作；P：输出操作

单道系统的工作情况

顺序程序的特点

- **顺序性**

  处理机的操作严格按照程序所规定的顺序执行。

- **封闭性**

  程序一旦开始执行，其计算结果不受外界因素的影响。

- **可再现性**

  程序执行的结果与它的执行速度无关 (即与时间无关)，而只与初始条件有关。

并发程序

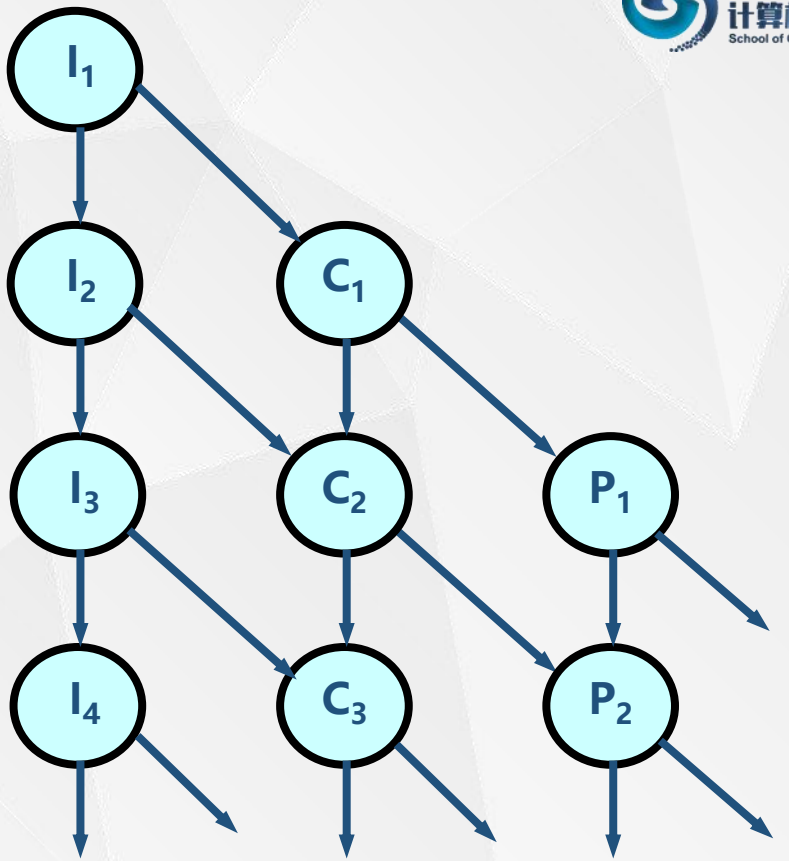
若干个程序段同时在系统中运行，这些程序段的执行在时间上是重叠的，一个程序段的执行尚未结束，另一个程序段的执行已经开始，即使这种重叠是很小的一部分，也称这几个程序段是并发执行的。

并发执行的描述方法

cobegin

S_1；S_2；…；S_n；

coend



多道系统的工作情况

- 哪些程序段的执行必须是顺序的？为什么？
- 哪些程序段的执行可以是并行的？为什么？

并发程序的特点

① 程序与计算不再一一对应

例1：

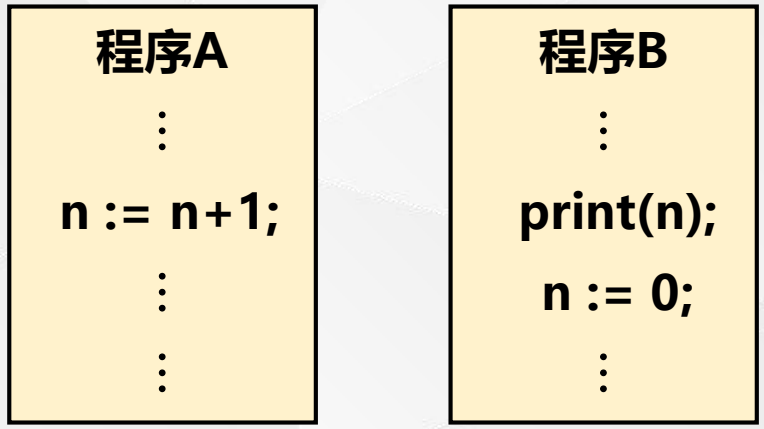
例2：

② 程序并发执行的相互制约

- 间接的相互制约关系 —— 资源共享
- 直接的相互制约关系 —— 公共变量

③ 失去程序的封闭性和可再现性

若一个程序的执行可以改变另一个程序的变量，那么，后者的输出就可能有赖于各程序执行的相对速度，即失去了程序的封闭性特点。



程序A的语句n :=n+1与程序B的两个语句的关系	之前	之后	之间
n的赋值	10	10	10
打印的结果	11	10	10
n的最终赋值	0	1	0

程序并发执行时的结果与各并发程序的相对速度有关，即给定相同的初始条件，若不加以控制，也可能得到不同的结果，此为**与时间有关的错误**。

进程概念

进程（Process）定义

进程是指一个具有一定独立功能的程序关于某个数据集合的一次运行活动。

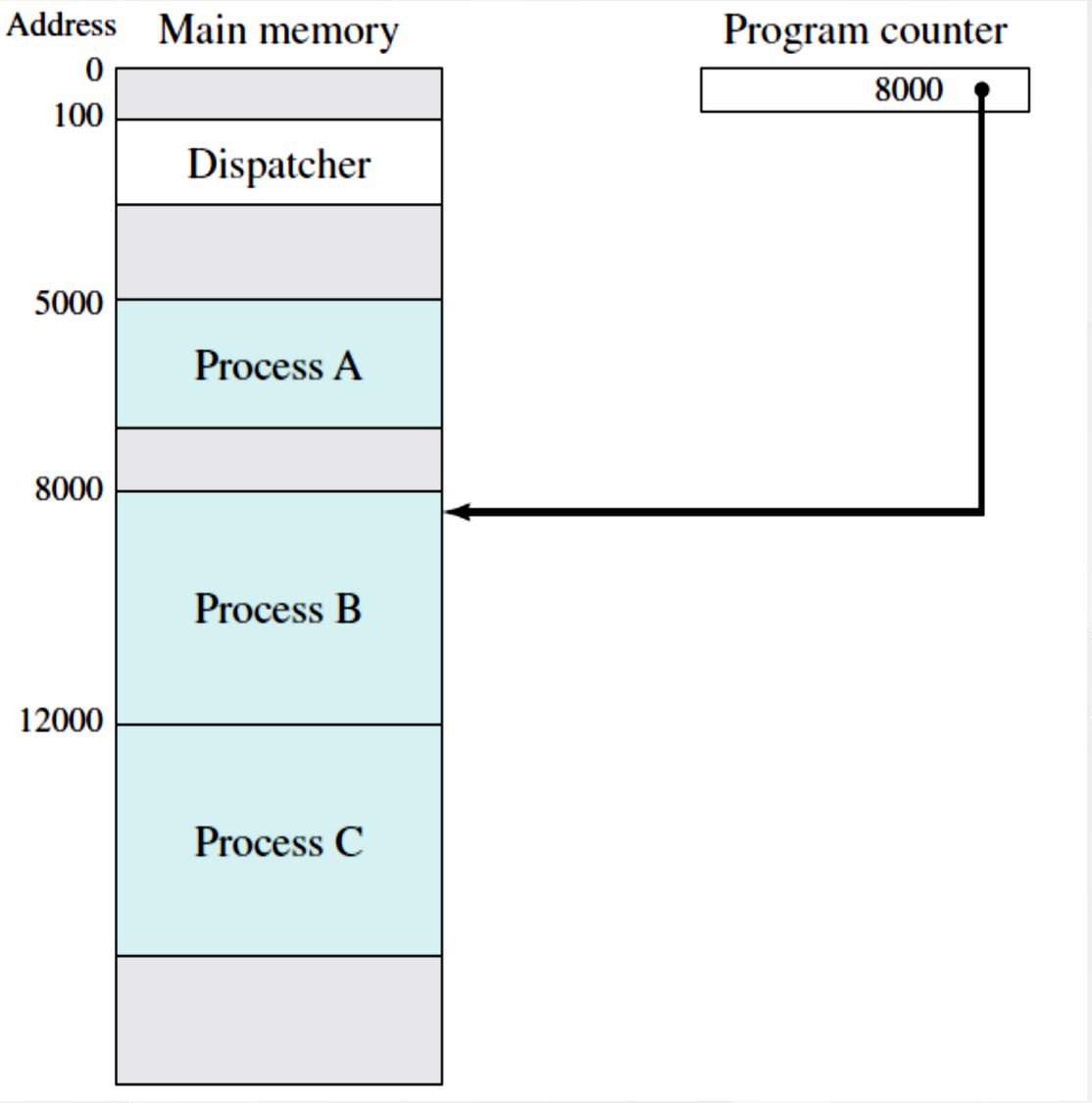
进程与程序的区别

① 程序是静态的概念，进程是动态的概念；

② 进程是一个独立运行的活动单位；

③ 进程是竞争系统资源的基本单位；

④ 一个程序可以对应多个进程，一个进程至少包含一个程序。



1	5000	27	12004
2	5001	28	12005
3	5002	----------Time-out	
4	5003	29	100
5	5004	30	101
6	5005	31	102
----------Time-out		32	103
7	100	33	104
8	101	34	105
9	102	35	5006
10	]103	36	5007
11	]104	37	5008
12	105	38	5009
13	8000	39	5010
14	8001	40	5011
15	8002	----------Time-out	
16	8003	41	100
----------I/O request		42	101
17	100	43	102
18	101	44	103
19	102	45	104
20	103	46	105
21	104	47	12006
22	105	48	12007
23	12000	49	12008
24	12001	50	12009
25	12002	51	12010
26	12003	52	12011
		----------Time-out	

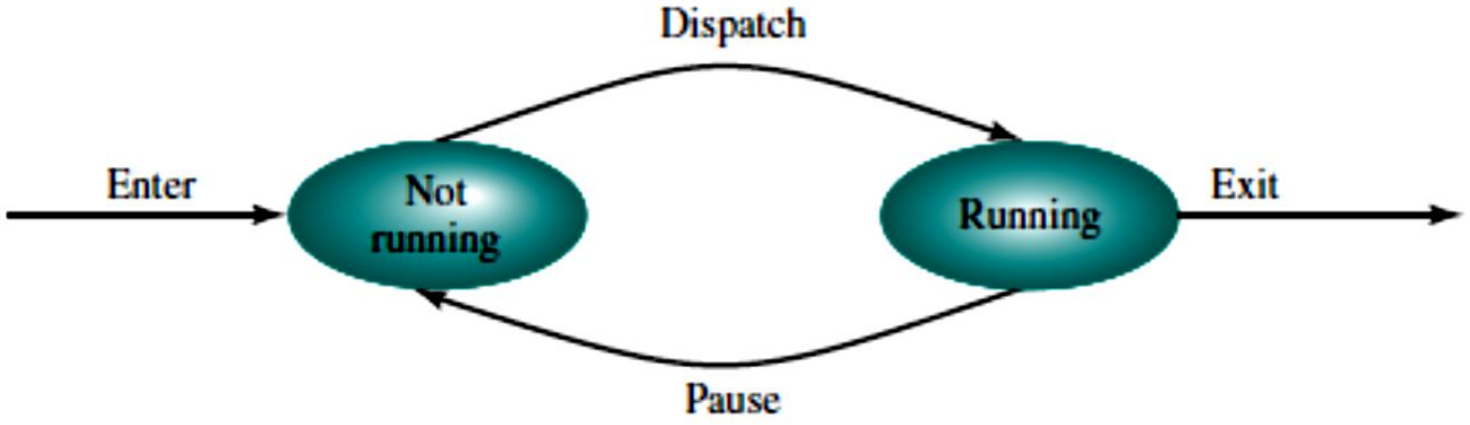
内存中有A、B、C三个进程
和操作系统调度程序

5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

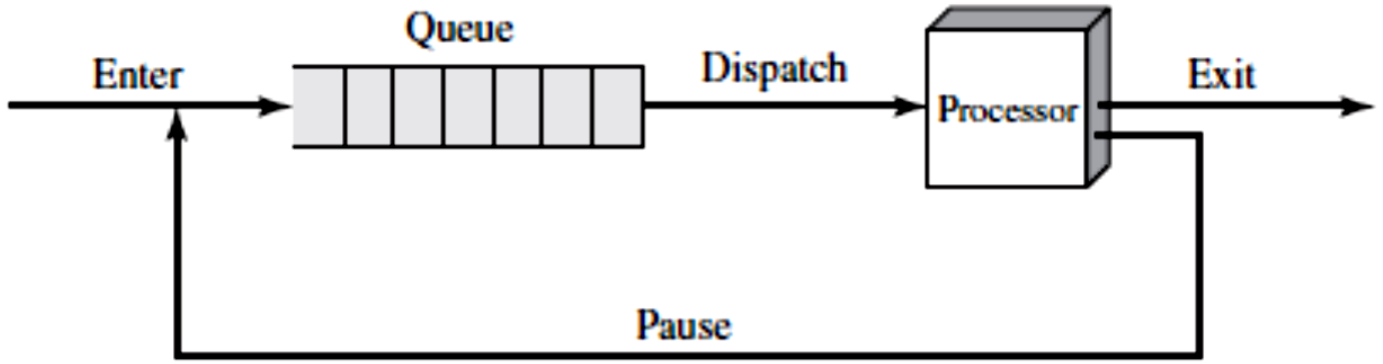
(a) Trace of process A (b) Trace of process B (c) Trace of process C

某一时刻A、B、C三个进程的执行轨迹

进程的状态



(a) State transition diagram



(b) Queueing diagram

进程的两个基本状态

进程的状态

进程的三个基本状态

① 运行状态(running)

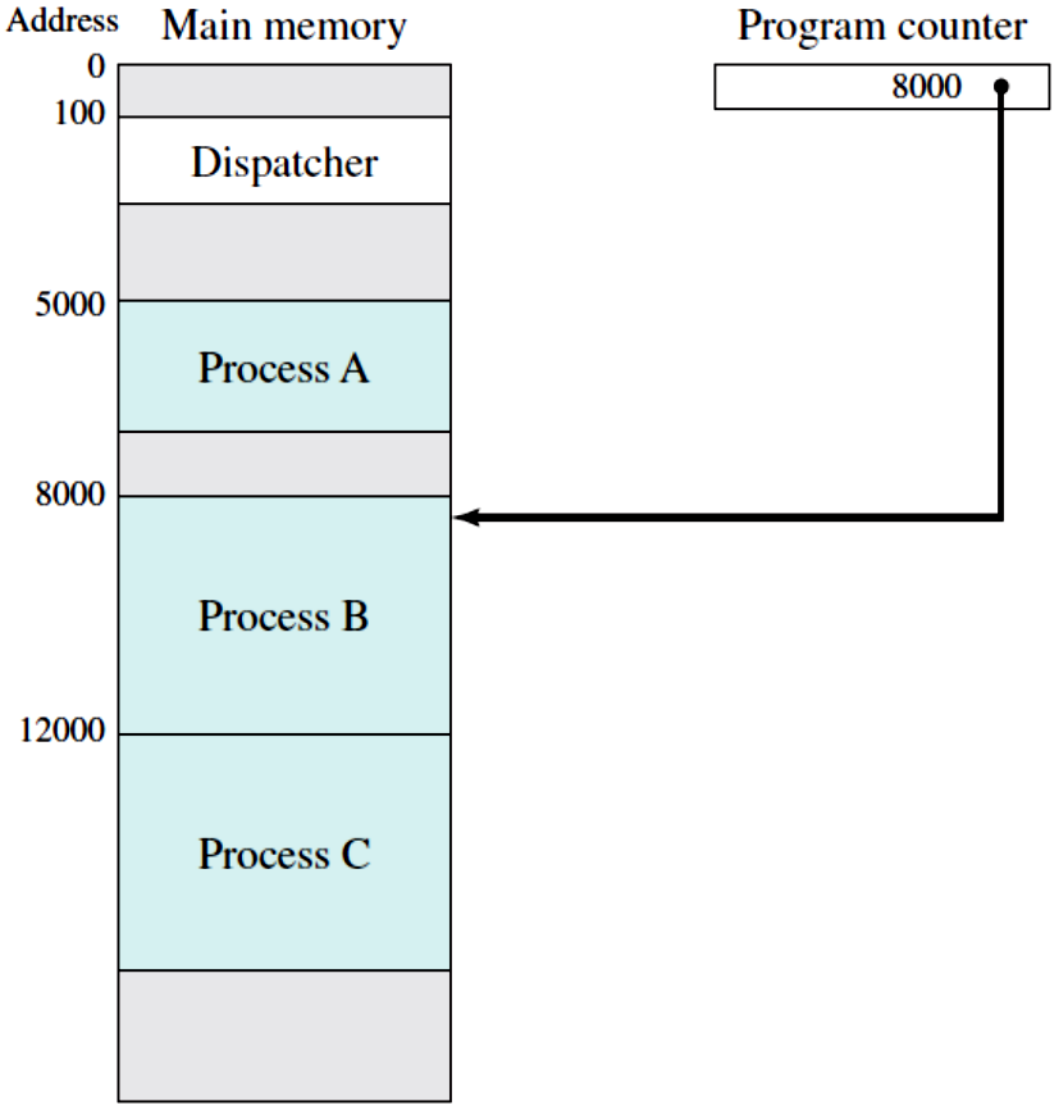
该进程已获得运行所必需的资源，它的程序正在处理机上执行。

② 等待状态(wait)

进程正等待着某一事件的发生而暂时停止执行。这时，即使给它CPU控制权，它也无法执行。

③ 就绪状态(ready)

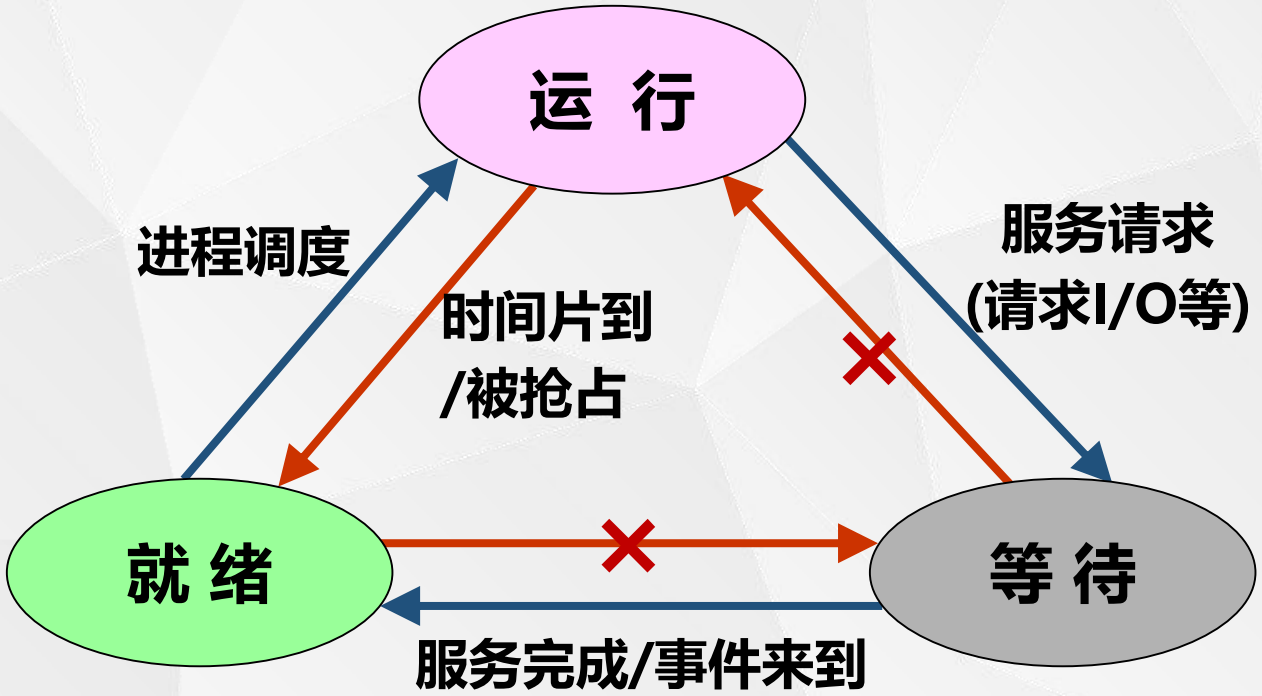
进程已获得除CPU之外的运行所必需的资源，一旦得到CPU控制权，立即可以运行。



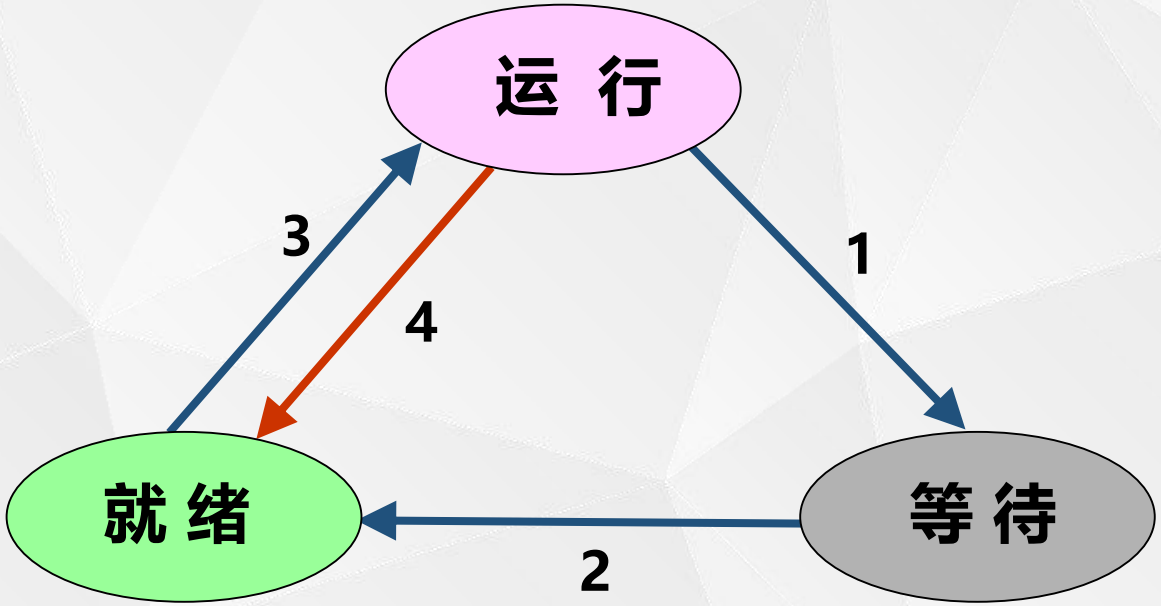
1	5000	27	12004
2	5001	28	12005
3	5002	----------Time-out	
4	5003	29	100
5	5004	30	101
6	5005	31	102
----------Time-out		32	103
7	100	33	104
8	101	34	105
9	102	35	5006
10	]103	36	5007
11	]104	37	5008
12	105	38	5009
13	8000	39	5010
14	8001	40	5011
15	8002	----------Time-out	
16	8003	41	100
----------I/O request		42	101
17	100	43	102
18	101	44	103
19	102	45	104
20	103	46	105
21	104	47	12006
22	105	48	12007
23	12000	49	12008
24	12001	50	12009
25	12002	51	12010
26	12003	52	12011
		----------Time-out	

在红色箭头指向的这个时刻，A、B、C三个进程分别处于什么状态?

进程状态变迁图



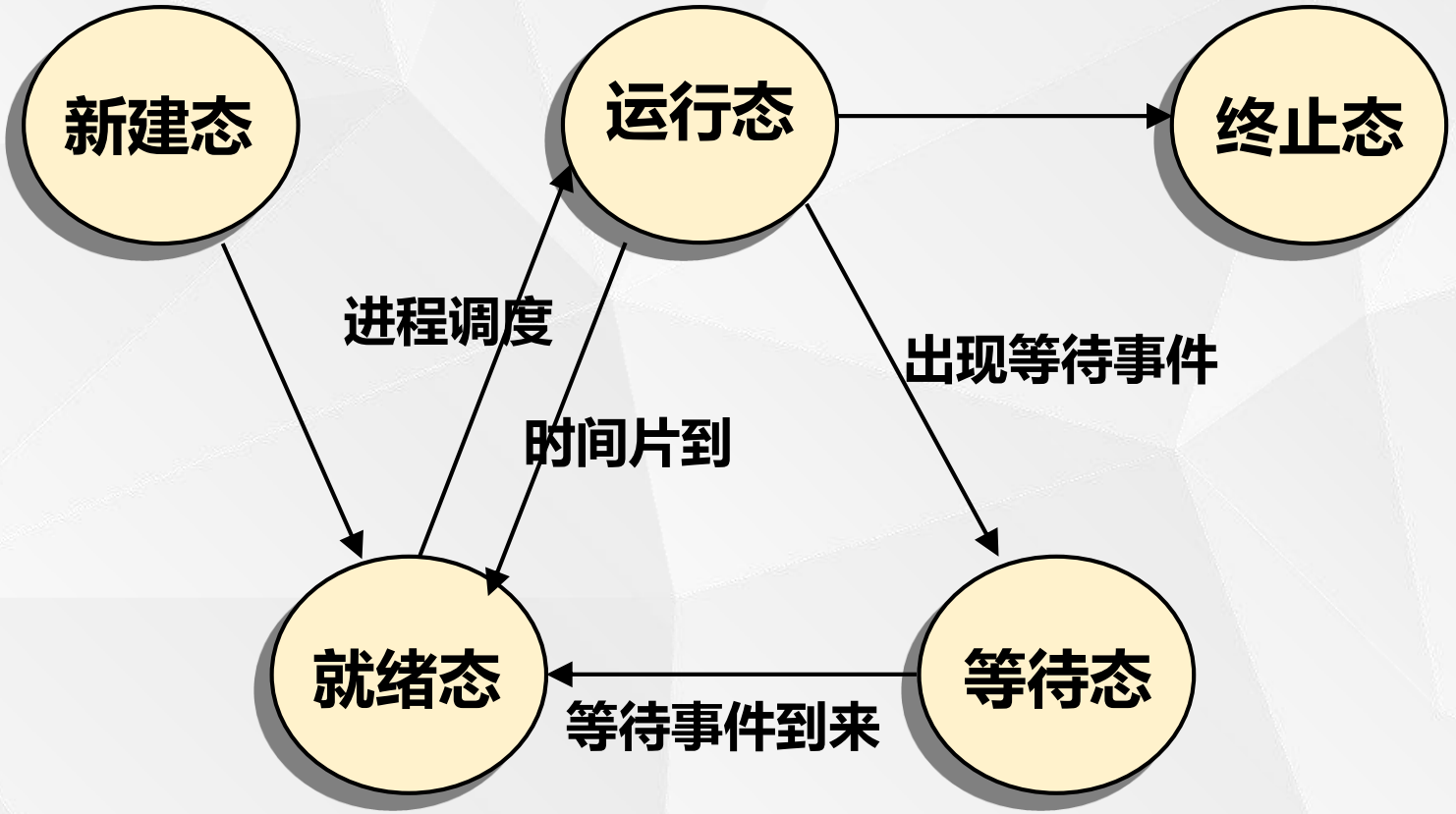
因果变迁



变迁1——> 变迁3，是否会发生？需要什么条件？

变迁4——> 变迁3，是否会发生？需要什么条件？

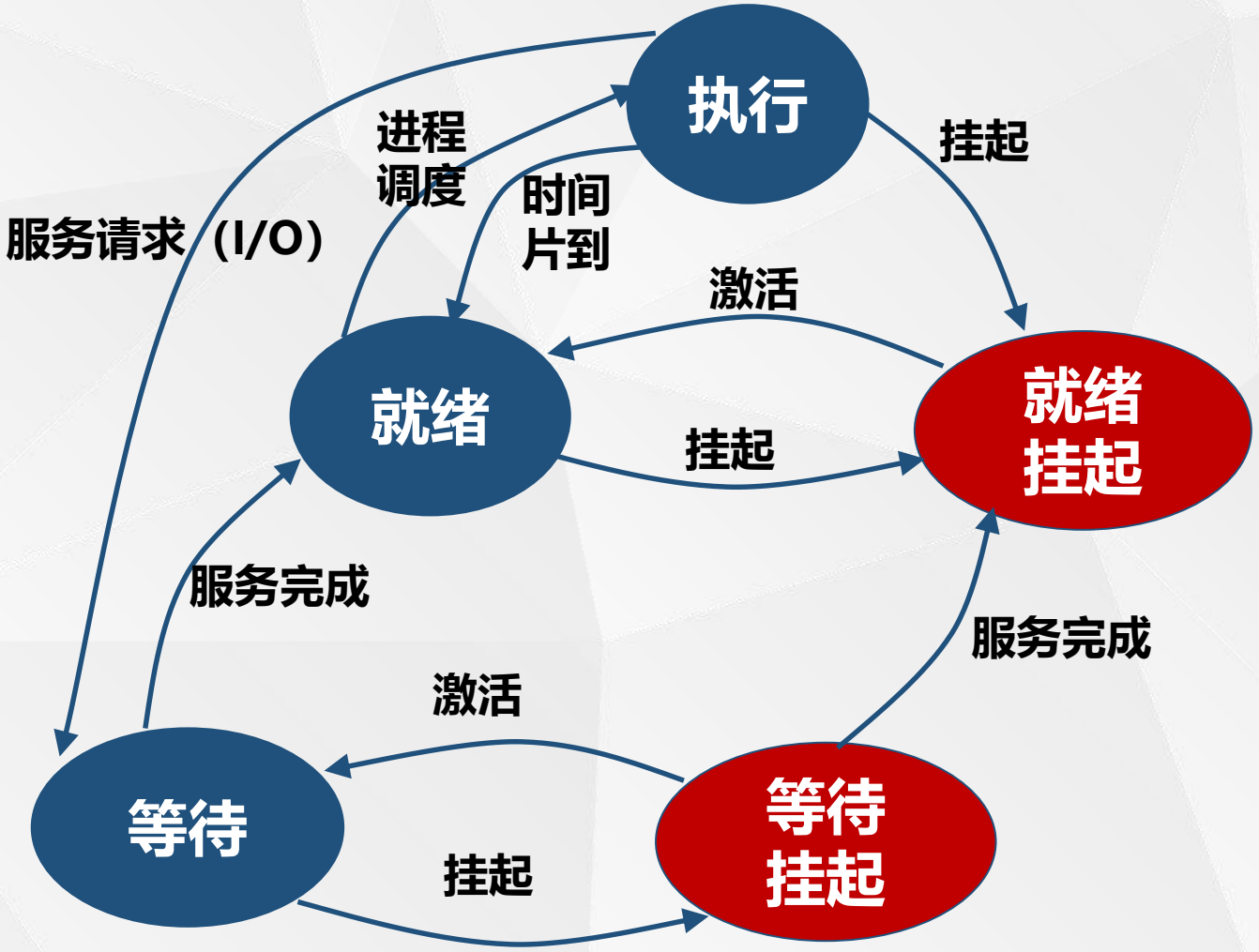
进程的五个状态及其转换




```
main()
{
    sleep(2);
    return();
}
```

讨论：此程序执行过程中的进程状态变迁。

具有挂起状态的进程状态变迁图



挂起：把内存中的进程换出到磁盘中，以便为其他进程让出空间。

- 等待→等待挂起
- 等待挂起→等待
- 等待挂起→就绪挂起
- 就绪挂起→就绪
- 就绪→就绪挂起
- 执行→就绪挂起

进程描述

- **进程控制块**

  描述进程与其他进程、系统资源的关系以及进程在各个不同时期所处的状态的数据结构，称为进程控制块 (process control block，PCB)。

- **进程的组成（进程映像）**

进程控制块

程序与数据

① 程序与数据

描述进程本身所应完成的功能。

② 进程控制块PCB

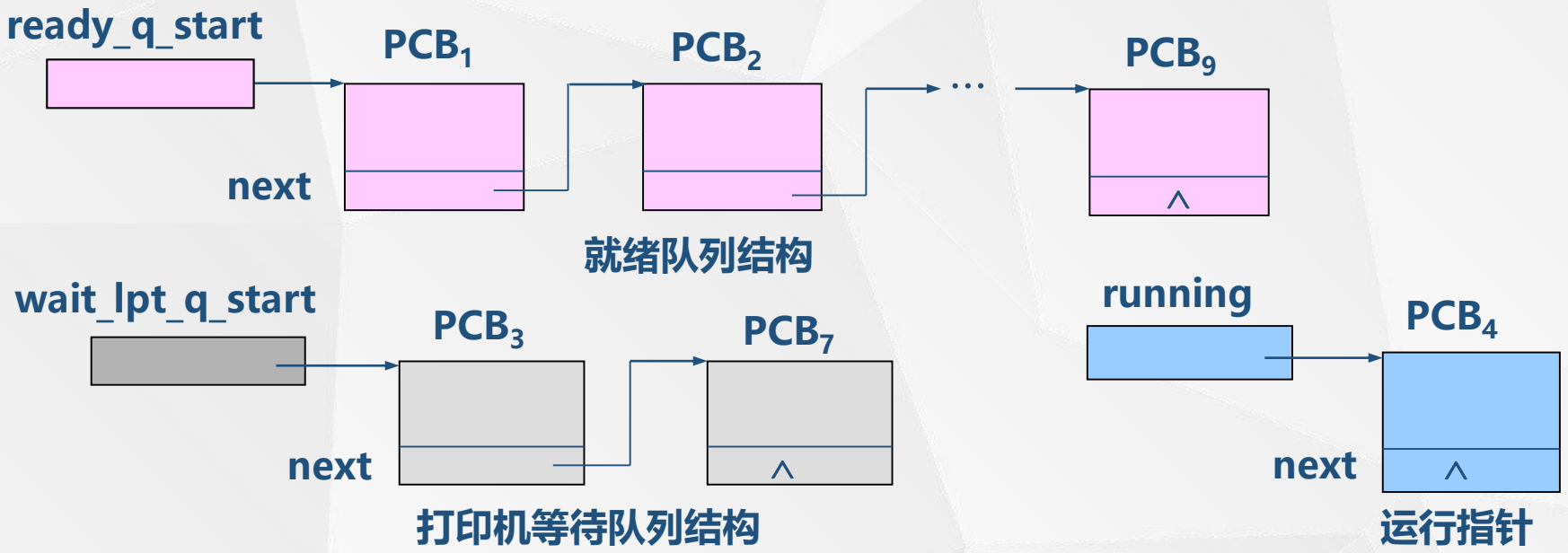
进程的动态特征，该进程与其他进程和系统资源的关系。

进程控制块的主要内容

① **进程标识符**　　进程符号名或内部 id号

② **进程当前状态**　　本进程目前处于何种状态

Q：大量的进程如何组织?



进程控制块的主要内容 （续）

③ 当前队列指针next

该项登记了处于同一状态的下一个进程的 PCB地址。

④ 进程优先级

反映了进程要求CPU的紧迫程度。

⑤ CPU现场保护区

当进程由于某种原因释放处理机时，CPU现场信息被保存在PCB的该区域中。

⑥ 通信信息

进程间进行通信时所记录的有关信息。

⑦ 家族联系

指明本进程与家族的联系

⑧ 占有资源清单

- Linux中的PCB结构叫task_struct，每个进程都把自己的信息放在一个task_struct结构里， task_struct包含了这些内容：
  - **标示符**： 描述本进程的唯一标示符，用来区别其他进程。
  - **状态**： 任务状态，退出代码，退出信号等。
  - **优先级**： 相对于其他进程的优先级。
  - **程序计数器**： 程序中即将被执行的下一条指令的地址。
  - **内存指针**： 包括程序代码和进程相关数据的指针，还有和其他进程共享的内存块的指针
  - **上下文数据**： 进程执行时处理器的寄存器中的数据。
  - **I/O状态信息**：包括显示的I/O请求，分配给进程的I / O设备和被进程使用的文件列表。
  - **记账信息**： 可能包括处理器时间总和，使用的时钟数总和，时间限制，记账号等。
- task_struct的定义可以在include/linux/sched.h里找到。所有运行在系统里的进程都以task_struct链表的形式存在内核里。

进程控制

1. 进程控制的概念

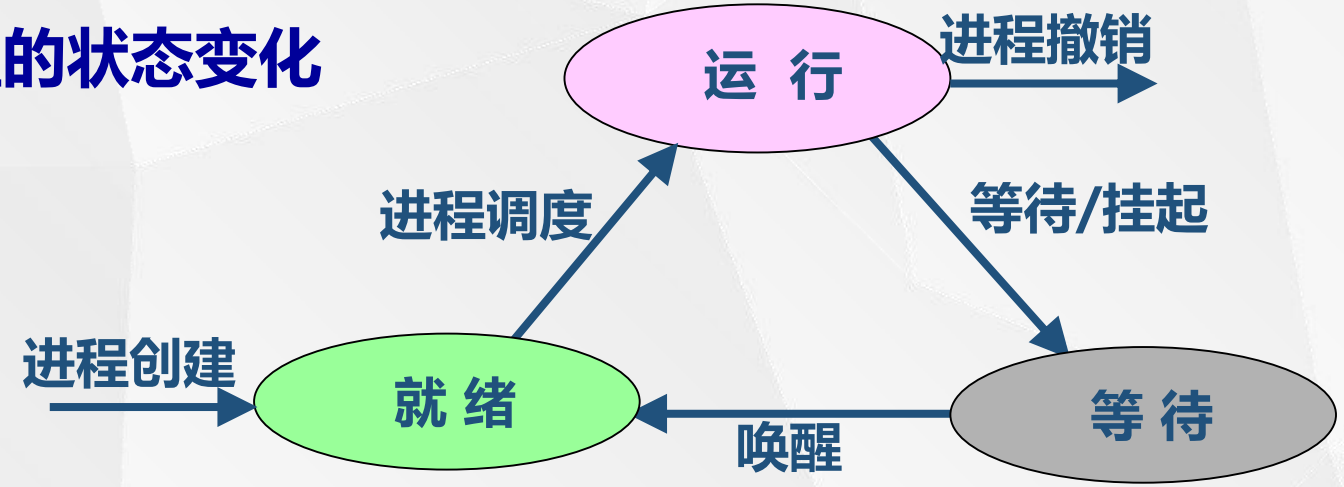
(1) 进程控制的职责

对系统中的进程实施有效的管理，负责进程状态的改变。

① 常用的进程控制原语（什么叫原语？）

创建原语、撤消原语、等待原语、唤醒原语

② 进程的状态变化



2 进程创建

① 进程创建原语的形式

create (name，priority)

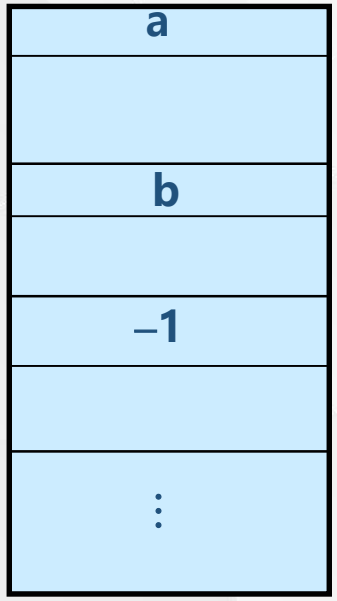
- name为被创建进程的标识符
- priority为进程优先级

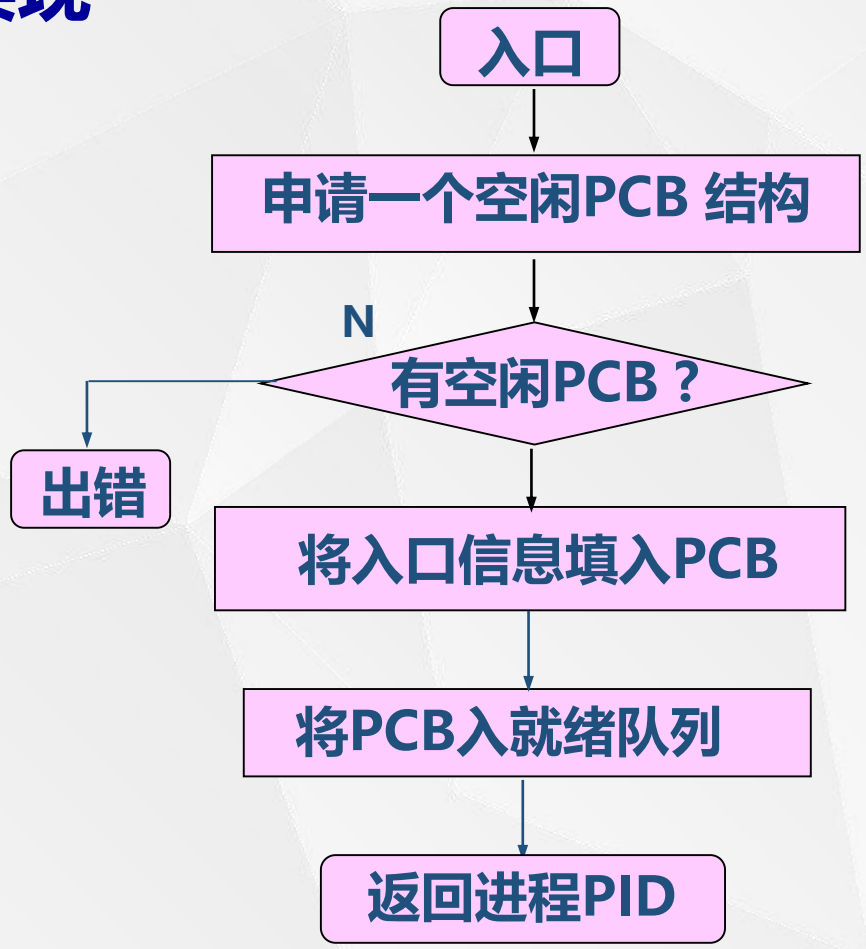
② 进程创建原语的功能

创建一个具有指定标识符的进程，建立进程的PCB结构。

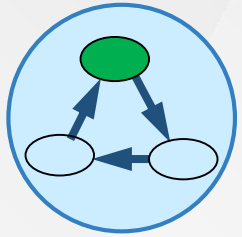
③ 进程创建原语的实现

- PCB池

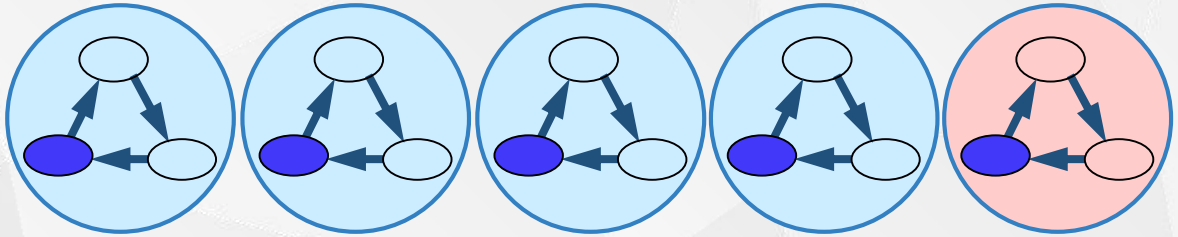




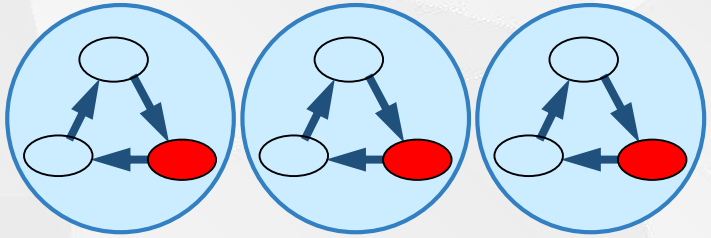
运行指针



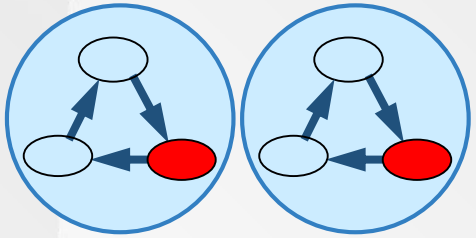
就绪队列



等待队列1



等待队列2



创建进程的典型原因

新建一个批处理任务	在一个批处理作业流中，当操作系统完成了上一个作业，从作业流中读出下一个作业时。
交互式登录	用户通过一个交互式终端登录到某个系统中时。
操作系统创建，以提供某种服务	现代操作系统往往需要创建某种服务（如打印服务）以辅助用户进程完成其工作。
被已经存在的进程所创建	例如为了提高多核系统的资源利用率，某用户进程创建一个子进程。

Linux——fork()

Linux用fork()创建一个子进程，它从父进程继承整个进程的地址空间，包括：**进程上下文、进程堆栈、内存信息、打开的文件描述符、信号控制设置、进程优先级、进程组号、当前工作目录、根目录、资源限制、控制终端**等。

① 为新进程分配一个新的PCB结构；

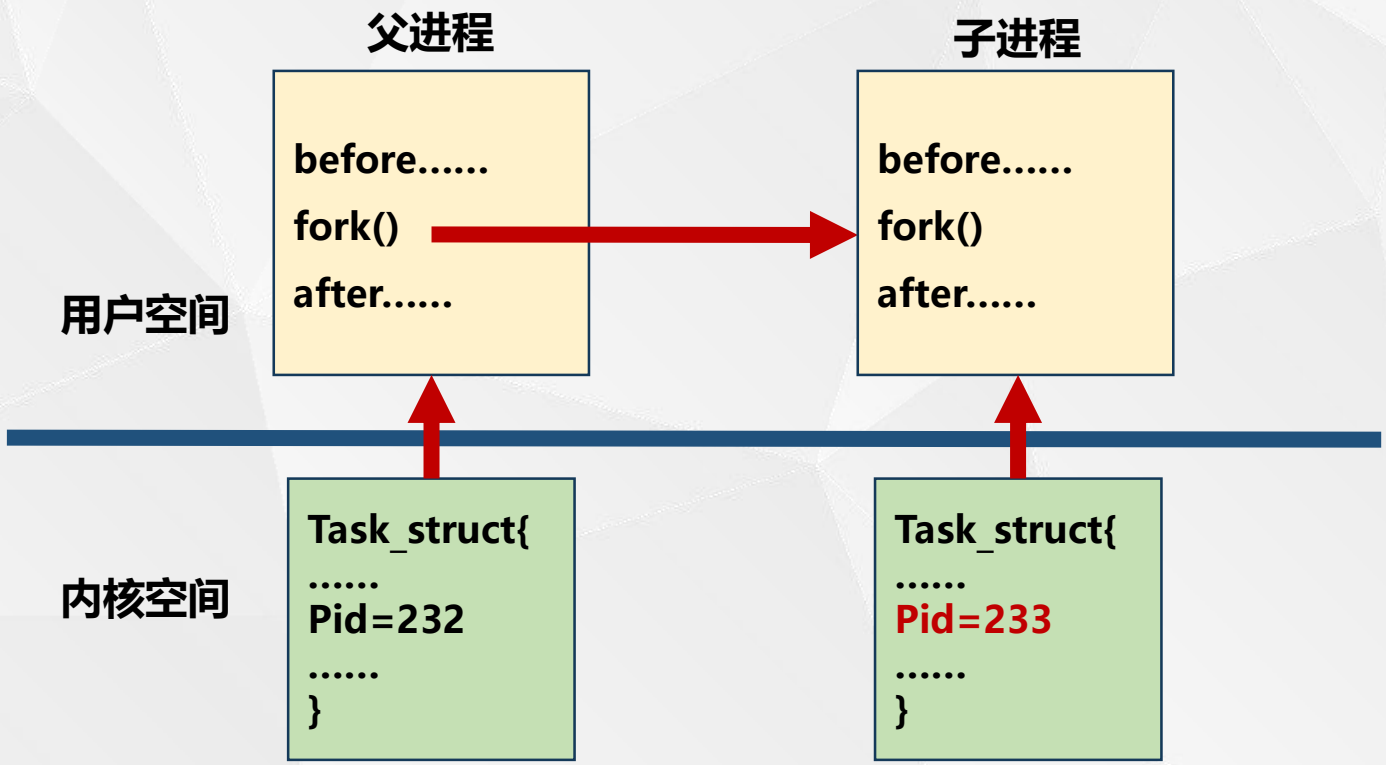
② 为子进程赋一个唯一的进程标识号 (PID)；

③ 为子进程做一个父进程上下文的逻辑副本。**这意味着父子进程将执行完全相同的代码。**

④ 增加与该进程相关联的文件表和索引节点表的引用数。这就意味着父进程打开的文件子进程可以继续使用。

⑤ **对父进程返回子进程的进程号，对子进程返回零。**

Linux——fork()


父进程
子进程
before……
fork()
after……
before……
fork()
after……
用户空间
内核空间
Task_struct{
……
Pid=232
……
}
Task_struct{
……
Pid=233
……
}