

第二章

并发进程

方 钰

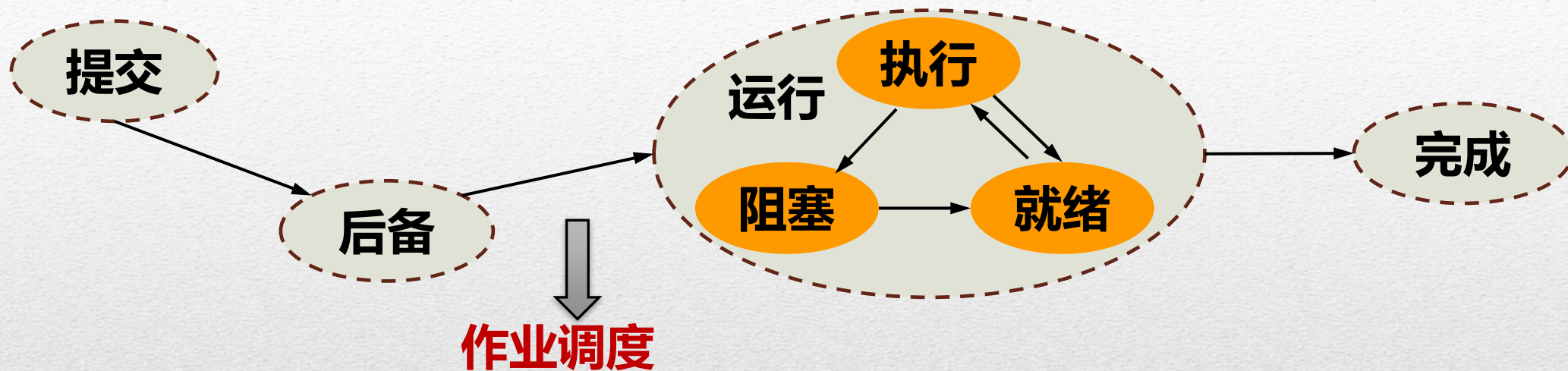


主要内容

- 2.1 **进程基本概念与调度控制**
- 2.2 **UNIX的进程**
- 2.3 **中断的基本概念及UNIX中断处理**
- 2.4 **进程通信机制**

作业调度

作业经历提交、后备、运行、完成四个状态



- 接纳多少个作业：取决于系统的**多道程序度**（系统规模和运行速度）
- 接纳哪些作业：取决于具体的**调度算法**

作业调度过程：

1. 按照一定的策略选取若干作业调入内存
2. 创建进程，分配资源
3. 如果创建成功，将新创建的进程插入就绪队列

程序并发执行带来的问题.....

 资源共享  各种程序活动的相互依赖与制约

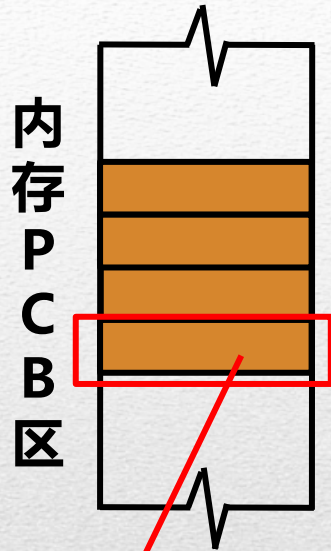
为了解决程序并发执行带来的问题：



一组数据与指令代码的集合

结构特征
代码段、数据段、堆
栈段、**进程控制块**

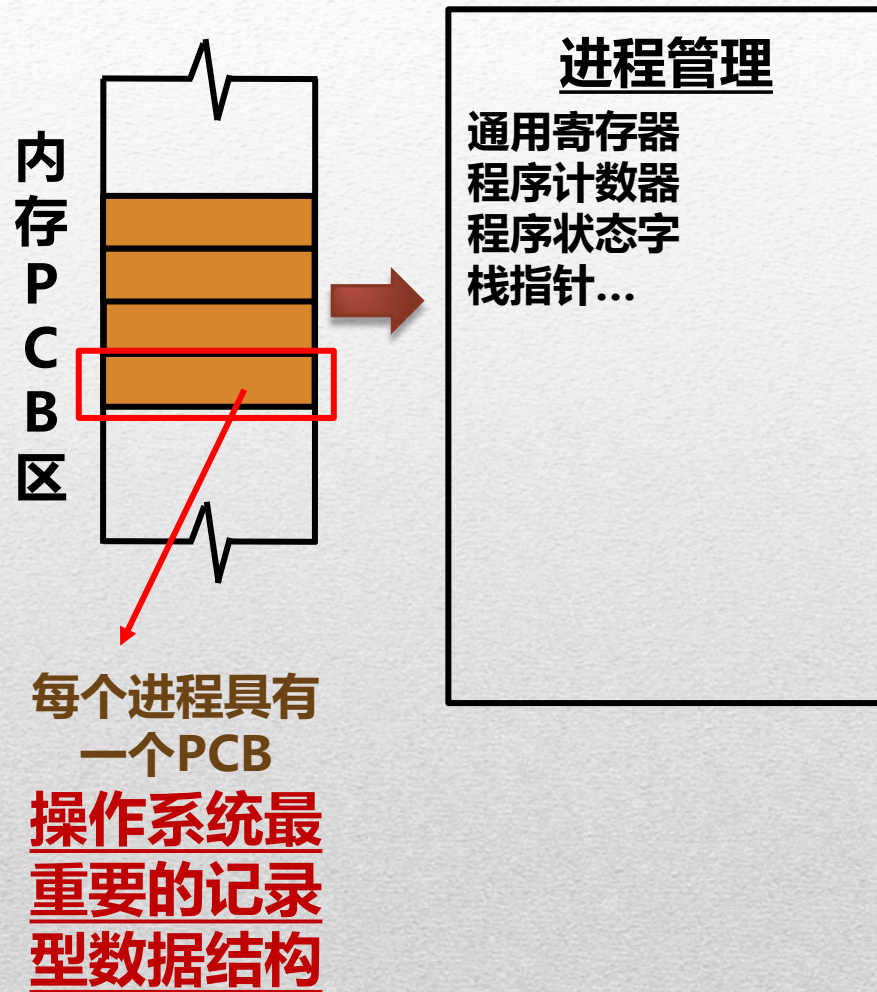
进程控制块 (Process Control Block, PCB)



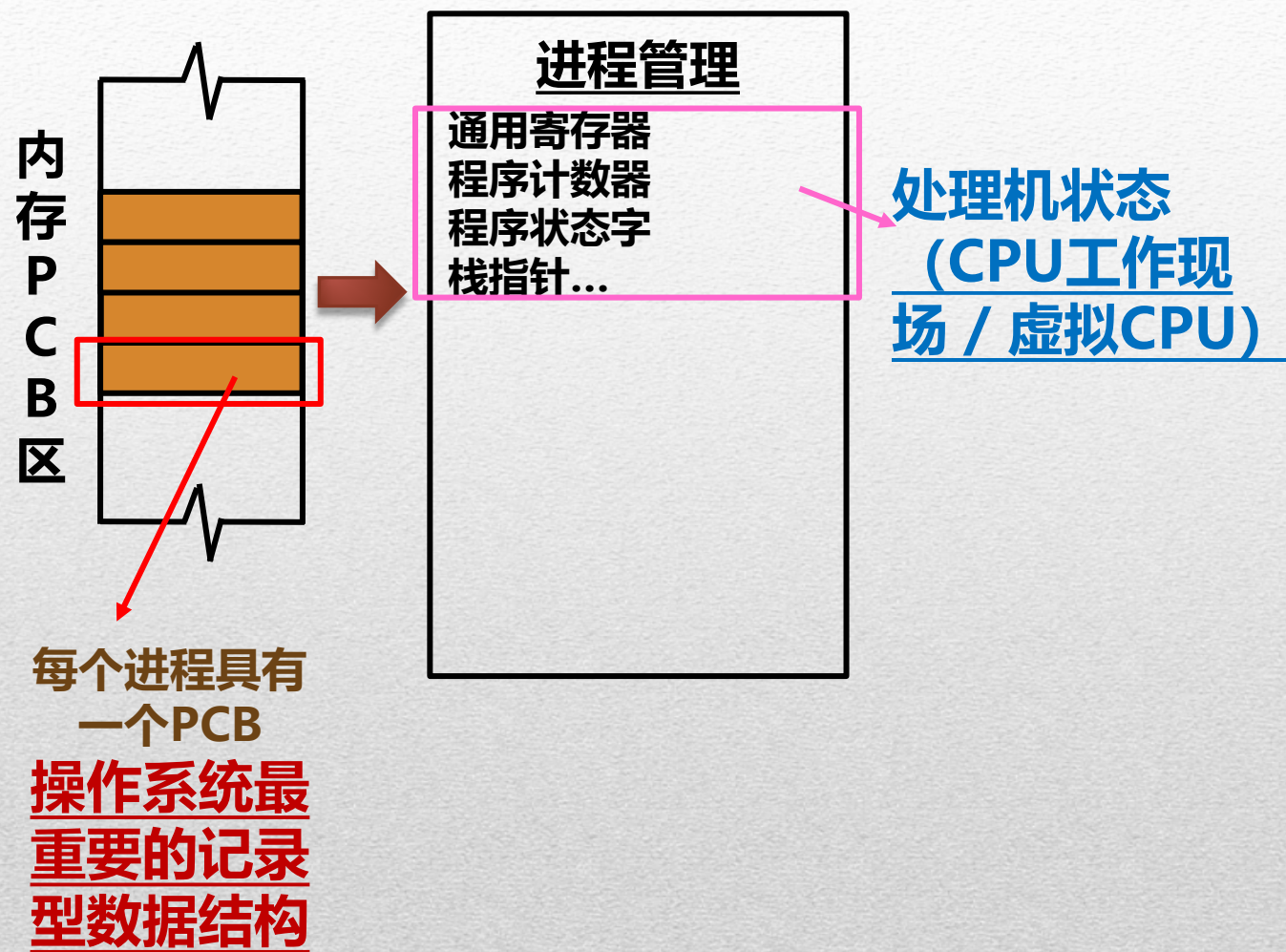
每个进程具有
一个PCB

操作系统最
重要的记录
型数据结构

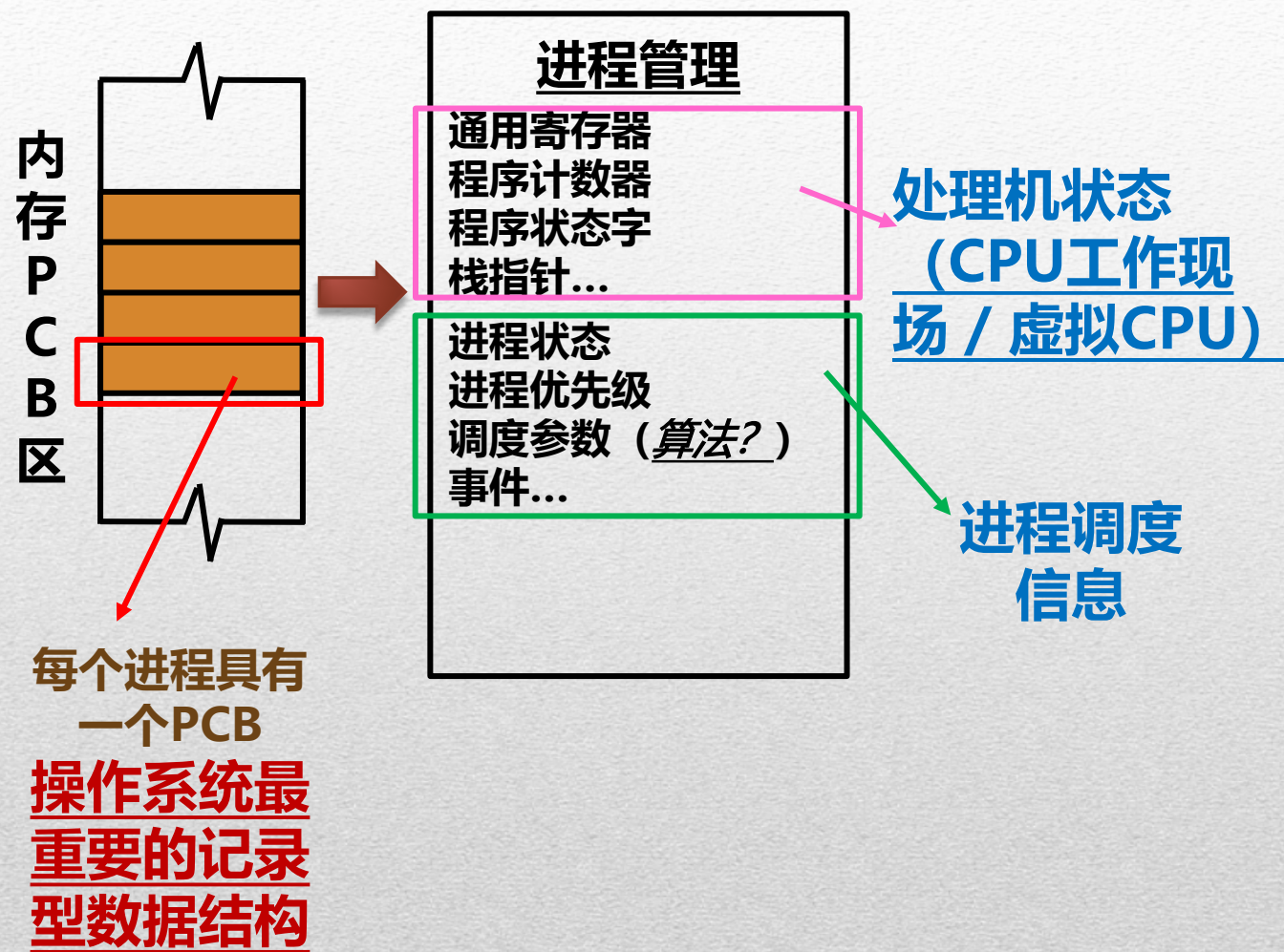
进程控制块 (Process Control Block, PCB)



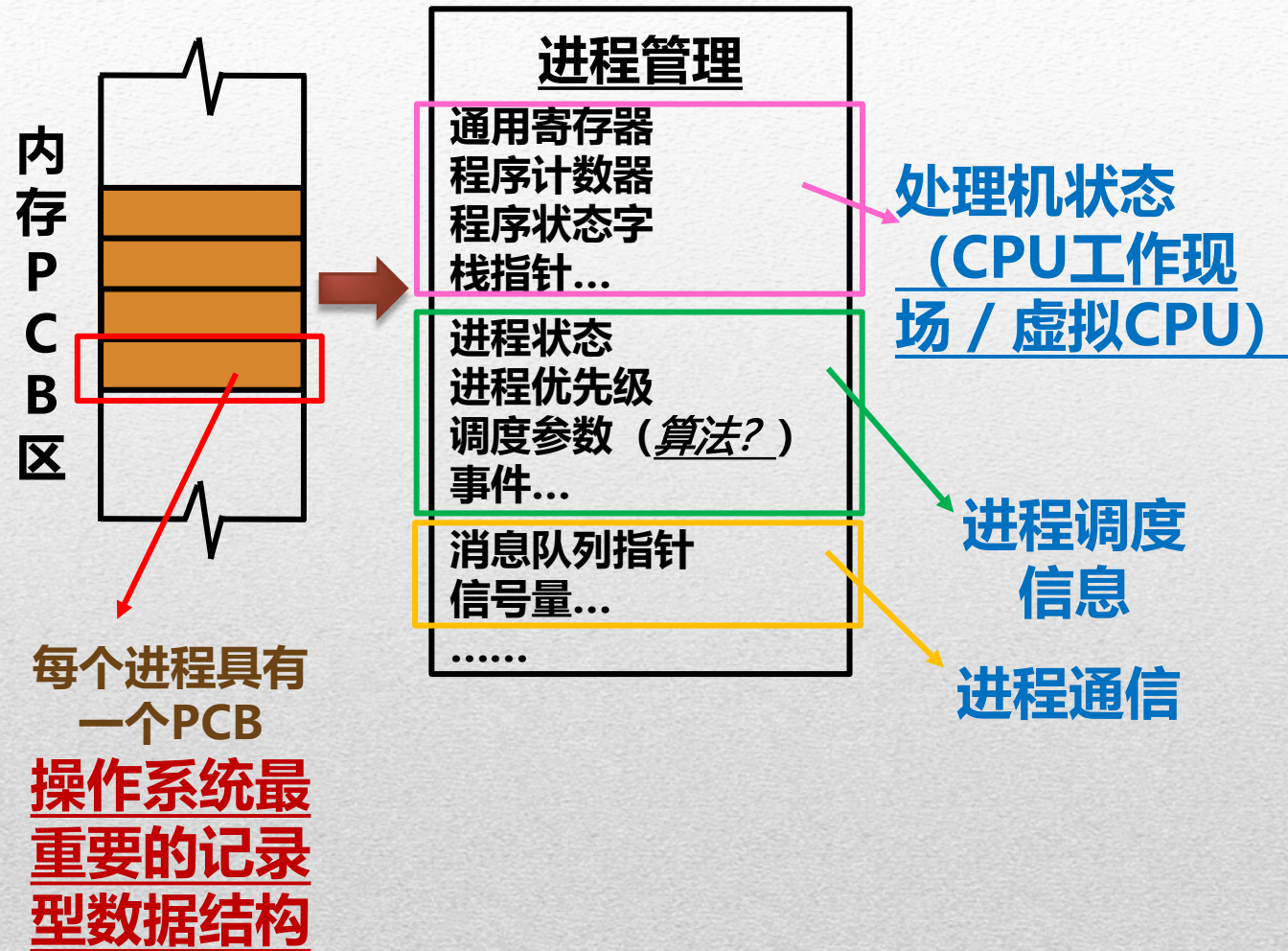
进程控制块 (Process Control Block, PCB)



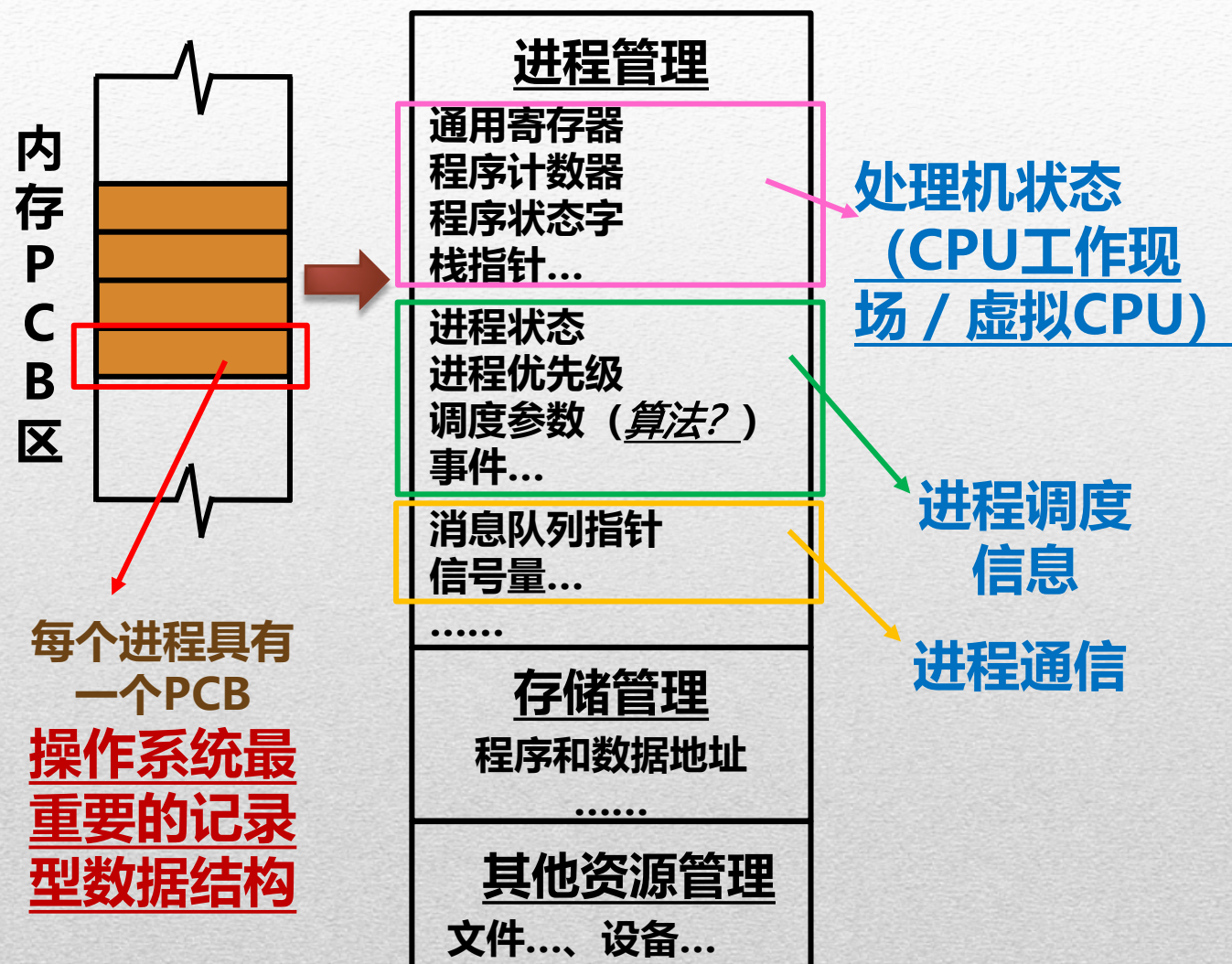
进程控制块 (Process Control Block, PCB)



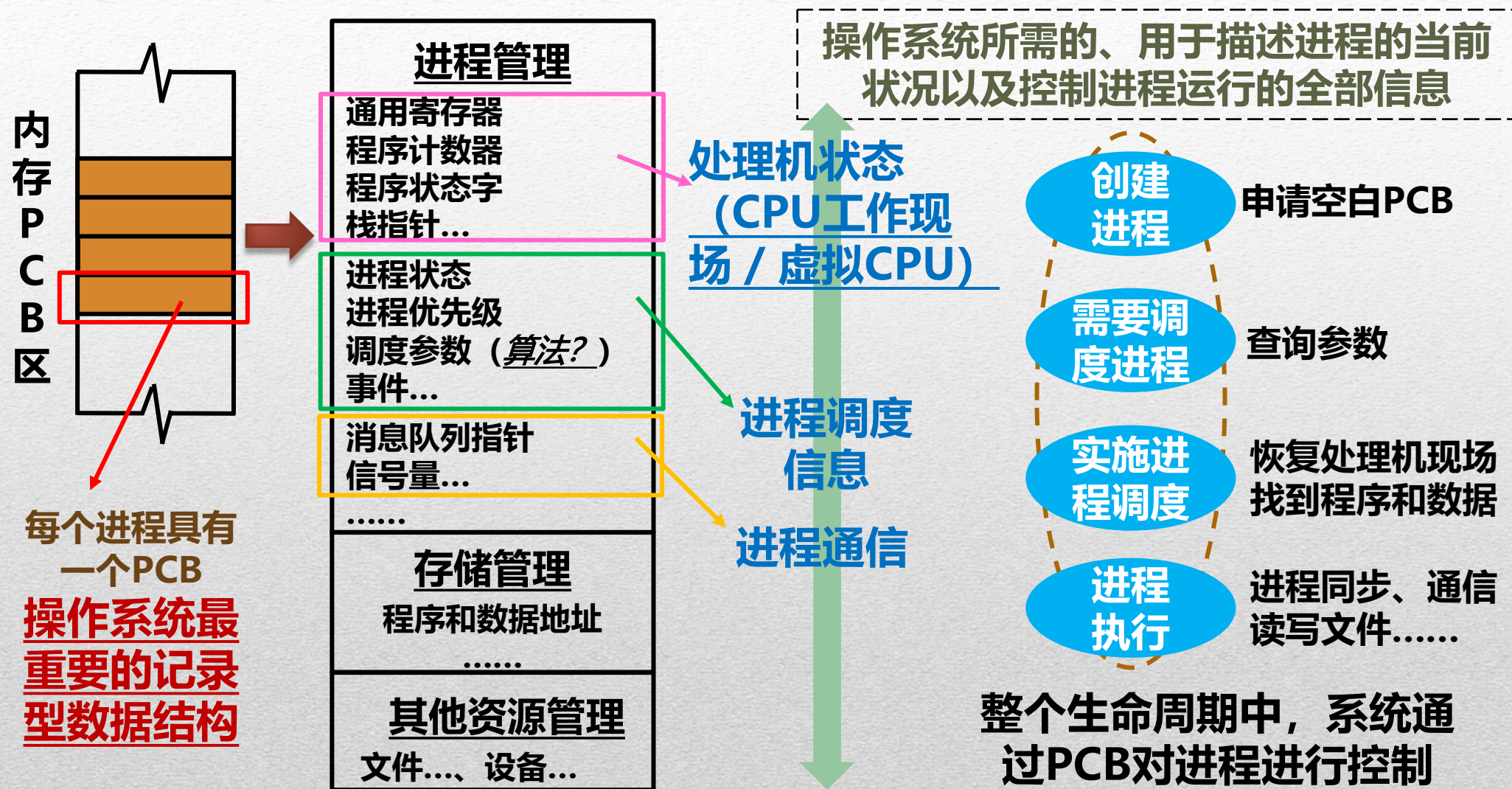
进程控制块 (Process Control Block, PCB)



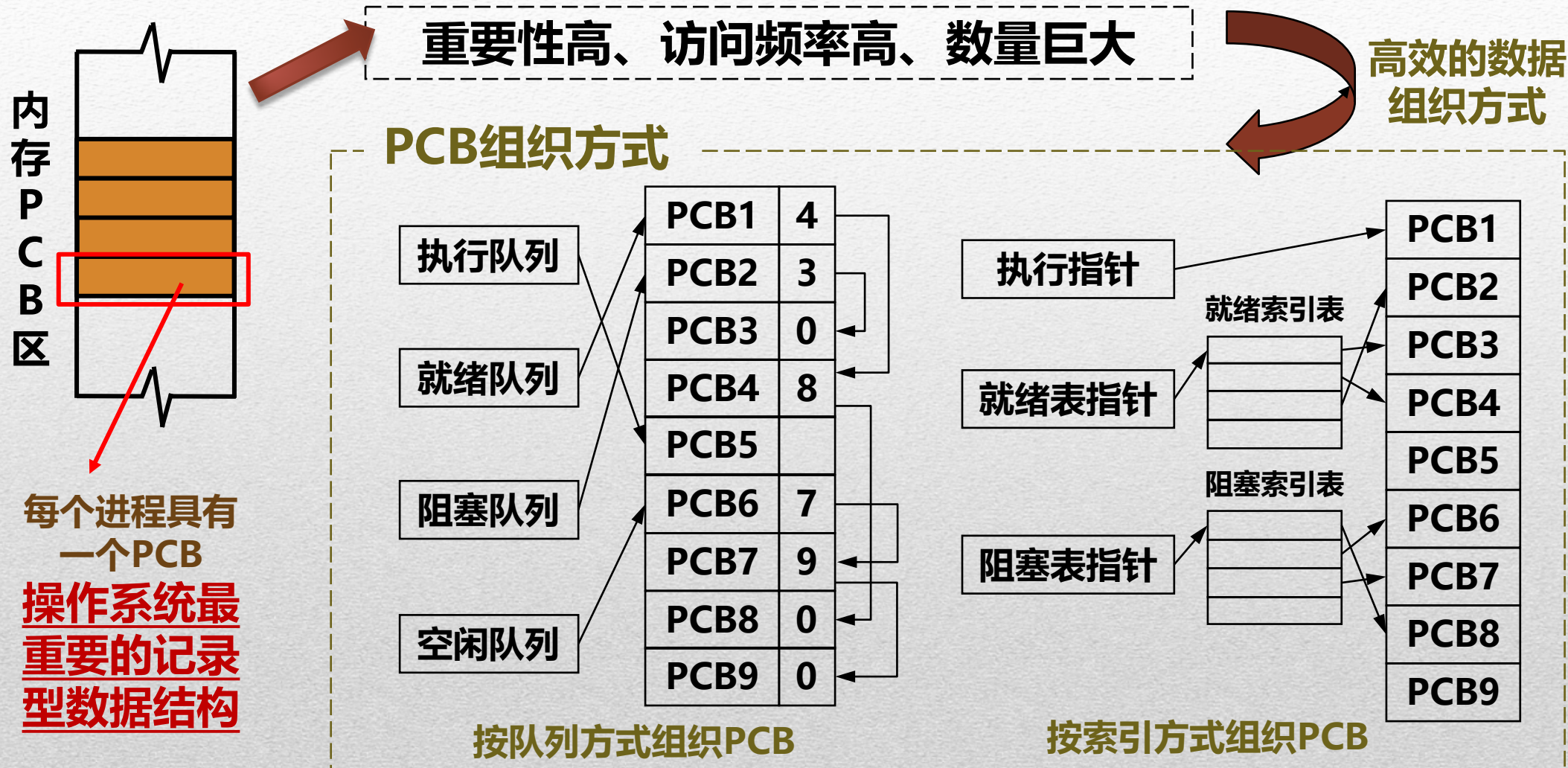
进程控制块 (Process Control Block, PCB)



进程控制块 (Process Control Block, PCB)



进程控制块 (Process Control Block, PCB)



程序并发执行带来的问题.....

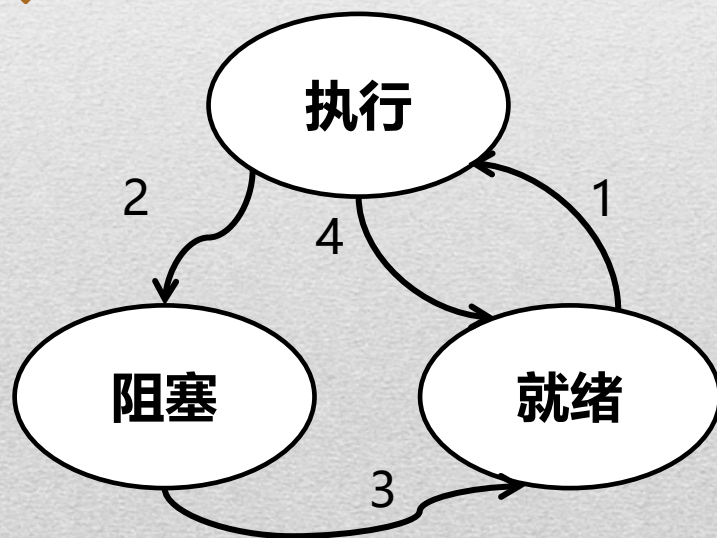
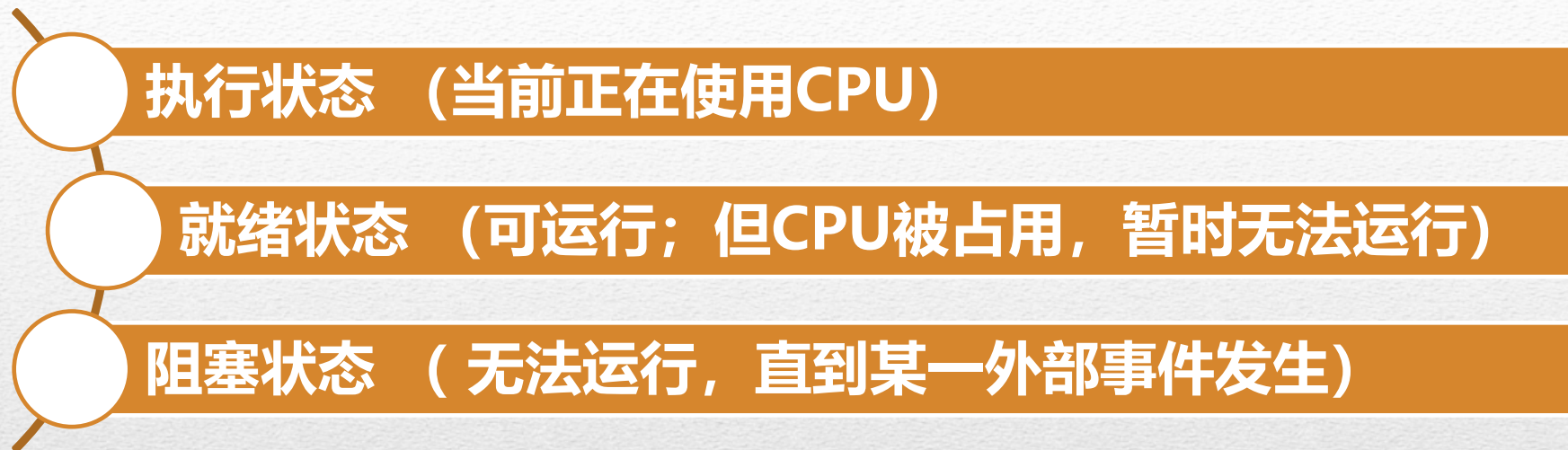
 资源共享  各种程序活动的相互依赖与制约

为了解决程序并发执行带来的问题：



一组数据与指令代码的集合	结构特征 代码段、数据段、堆栈段、 进程控制块
静态的 存放在某种介质上	动态性 ，具有生命周期 “ 由创建而产生，由调度而执行，由撤销而消亡 ”

进程的三种调度状态



1. 进程被调度
2. 进程由于等待某种外部事件被阻塞
3. 等待的外部事件发生被唤醒
4. 将CPU让给另一个进程

排队等待叫号
(就绪状态, 等待调度)



就绪状态



执行状态



排队等待叫号
(就绪状态, 等待调度)



就绪状态

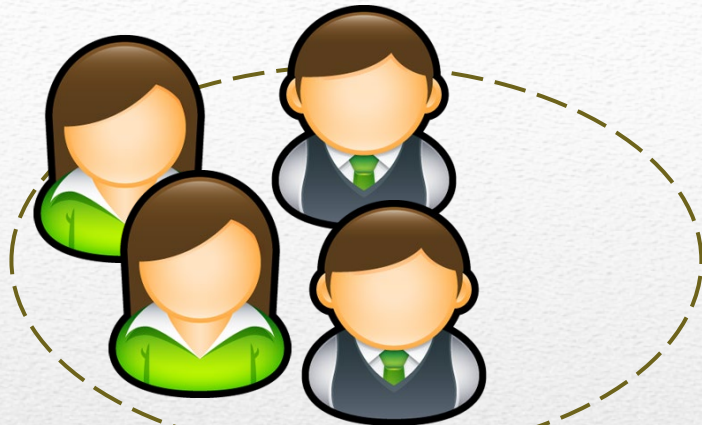
① 就诊 (分配CPU, 进程执行)



执行状态

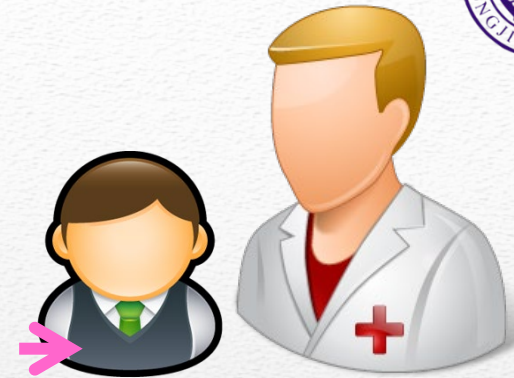


排队等待叫号
(就绪状态, 等待调度)



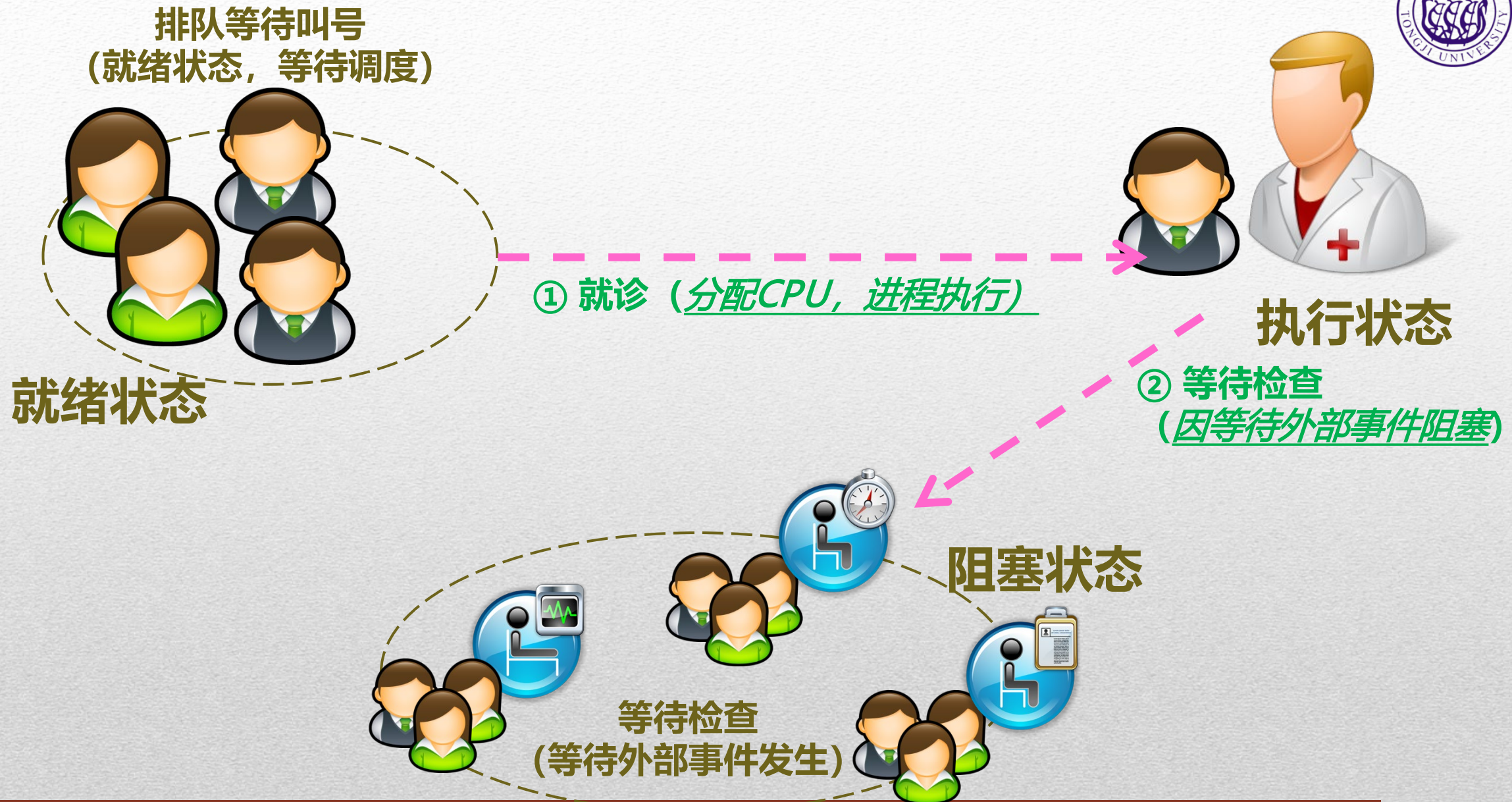
就绪状态

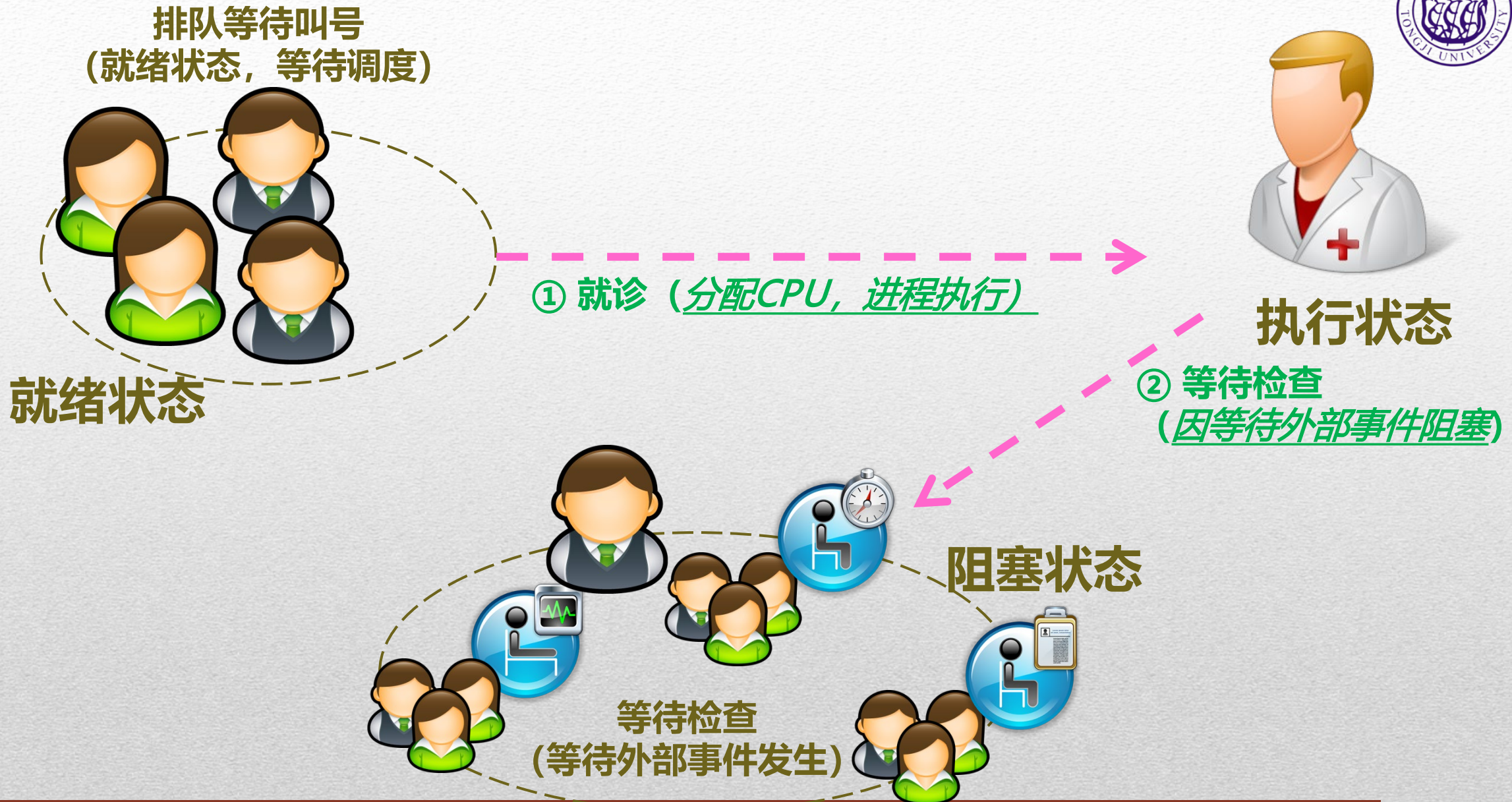
① 就诊 (分配CPU, 进程执行)

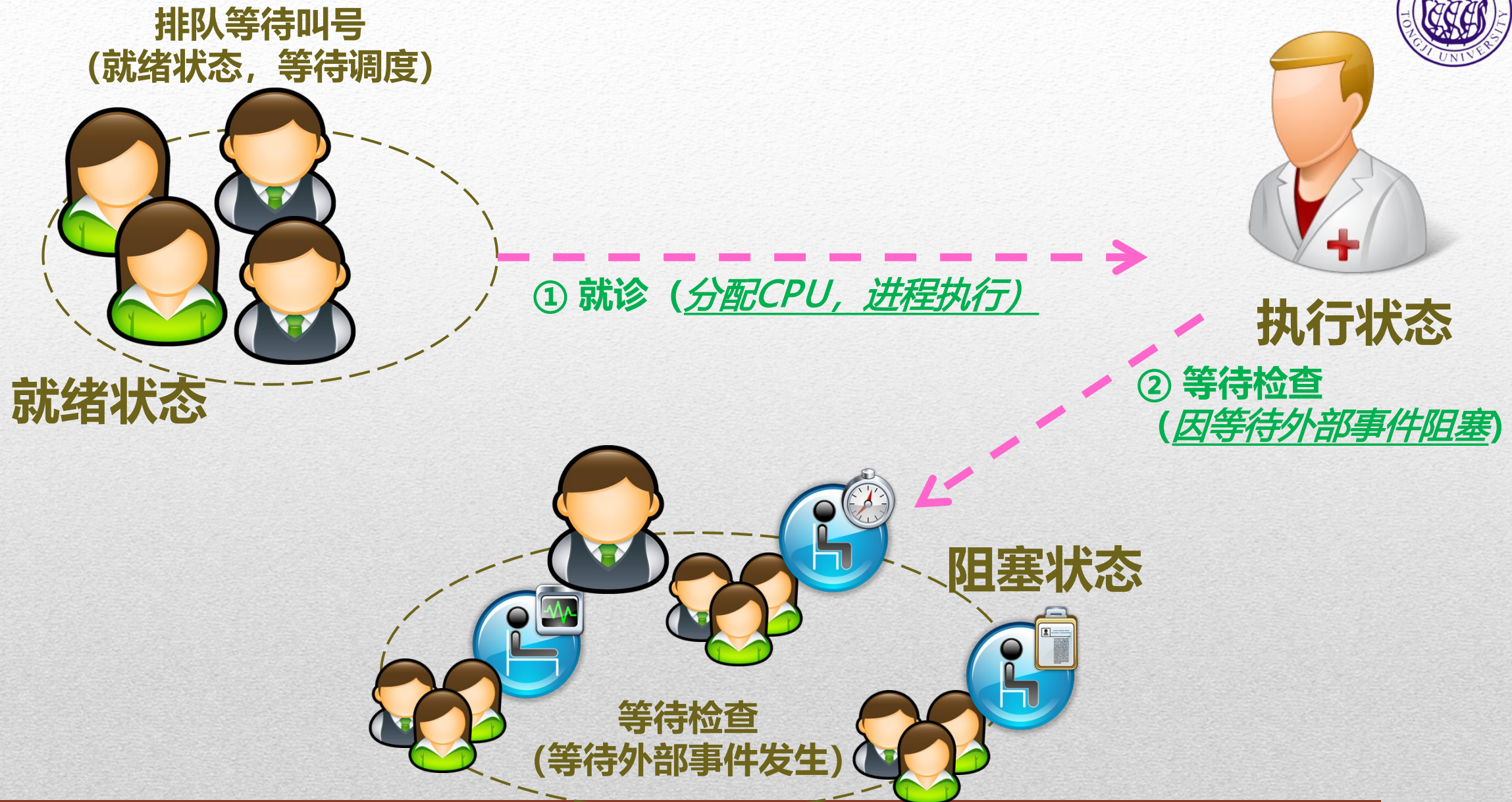


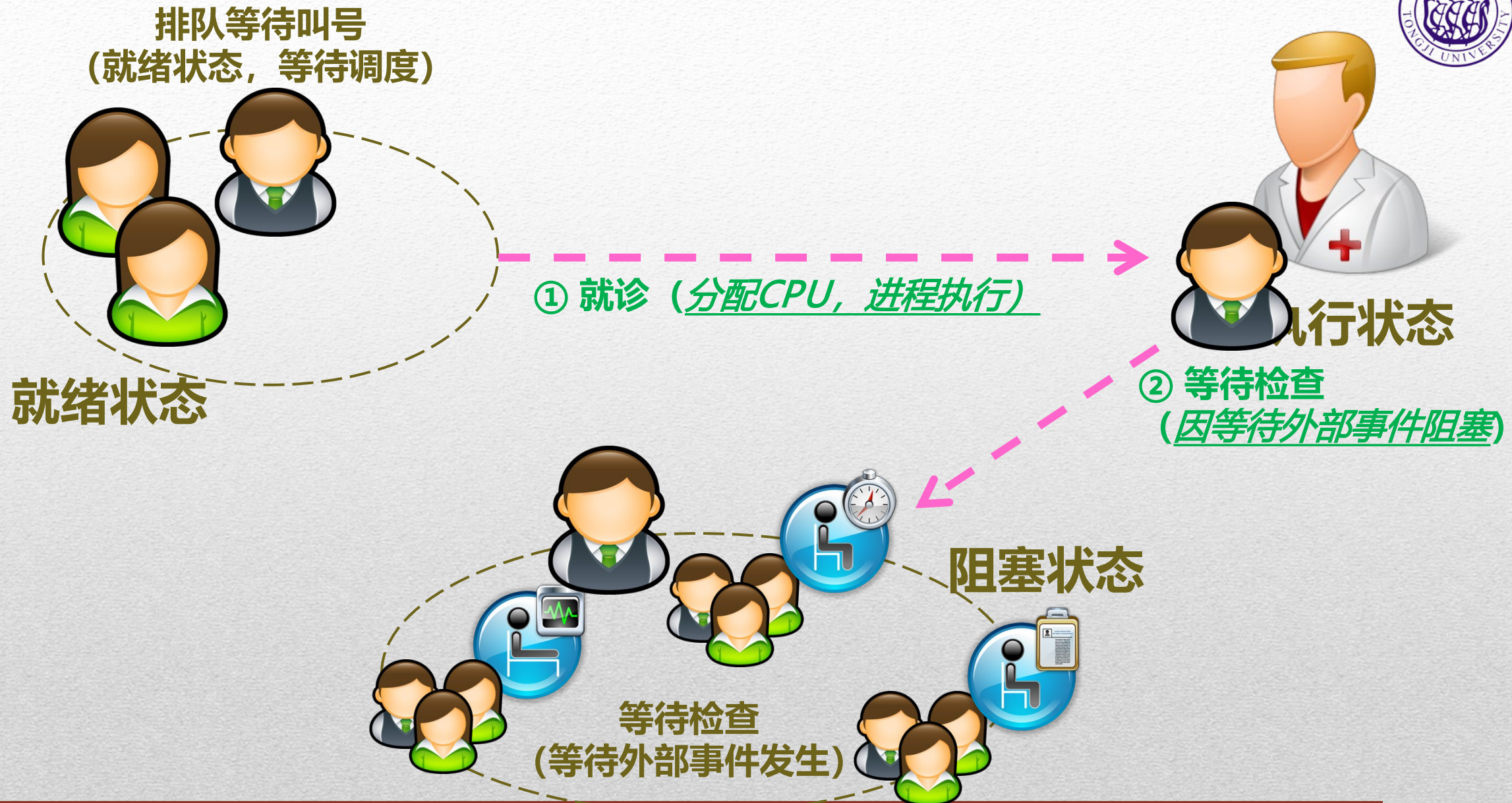
执行状态

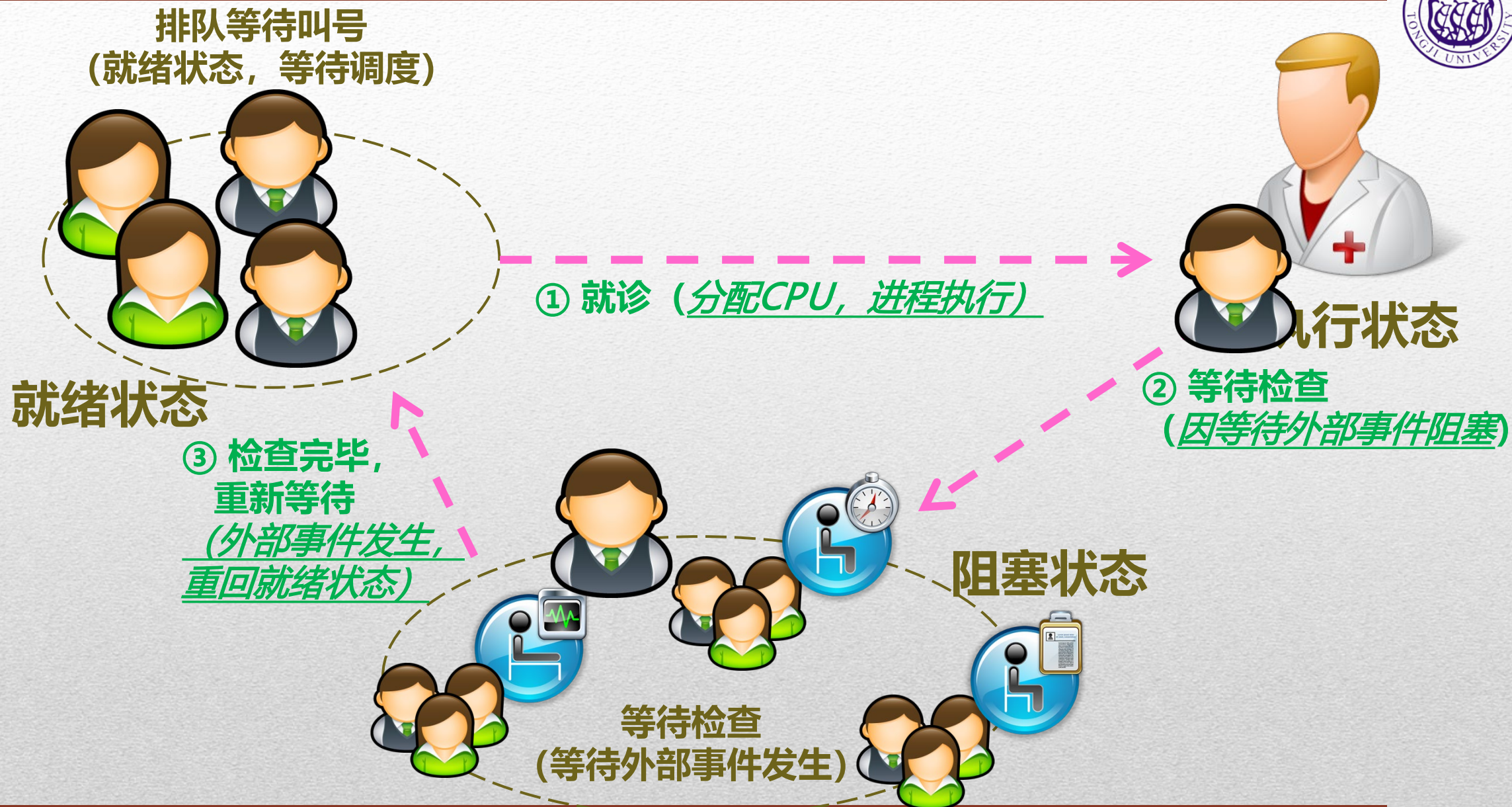


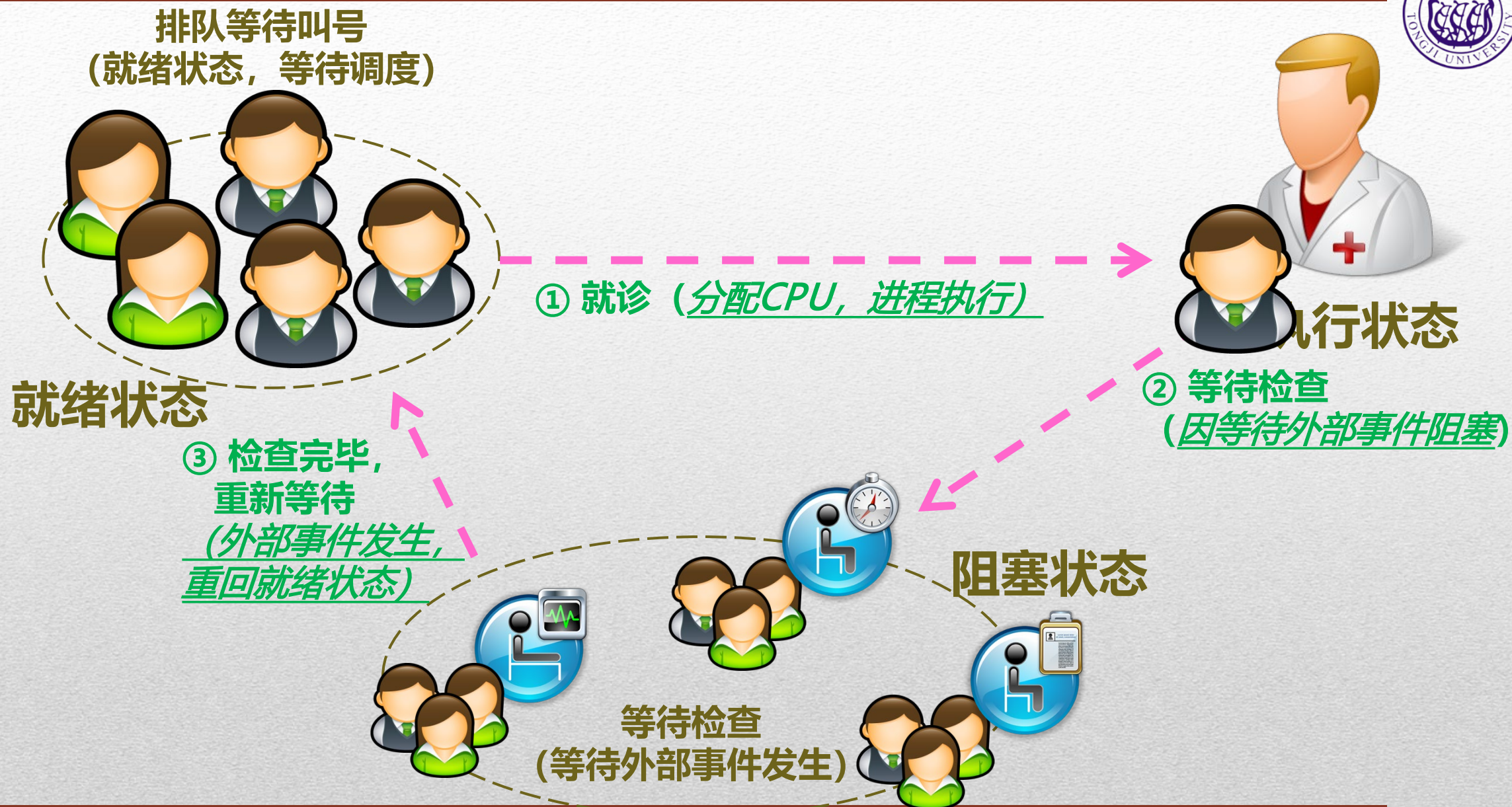


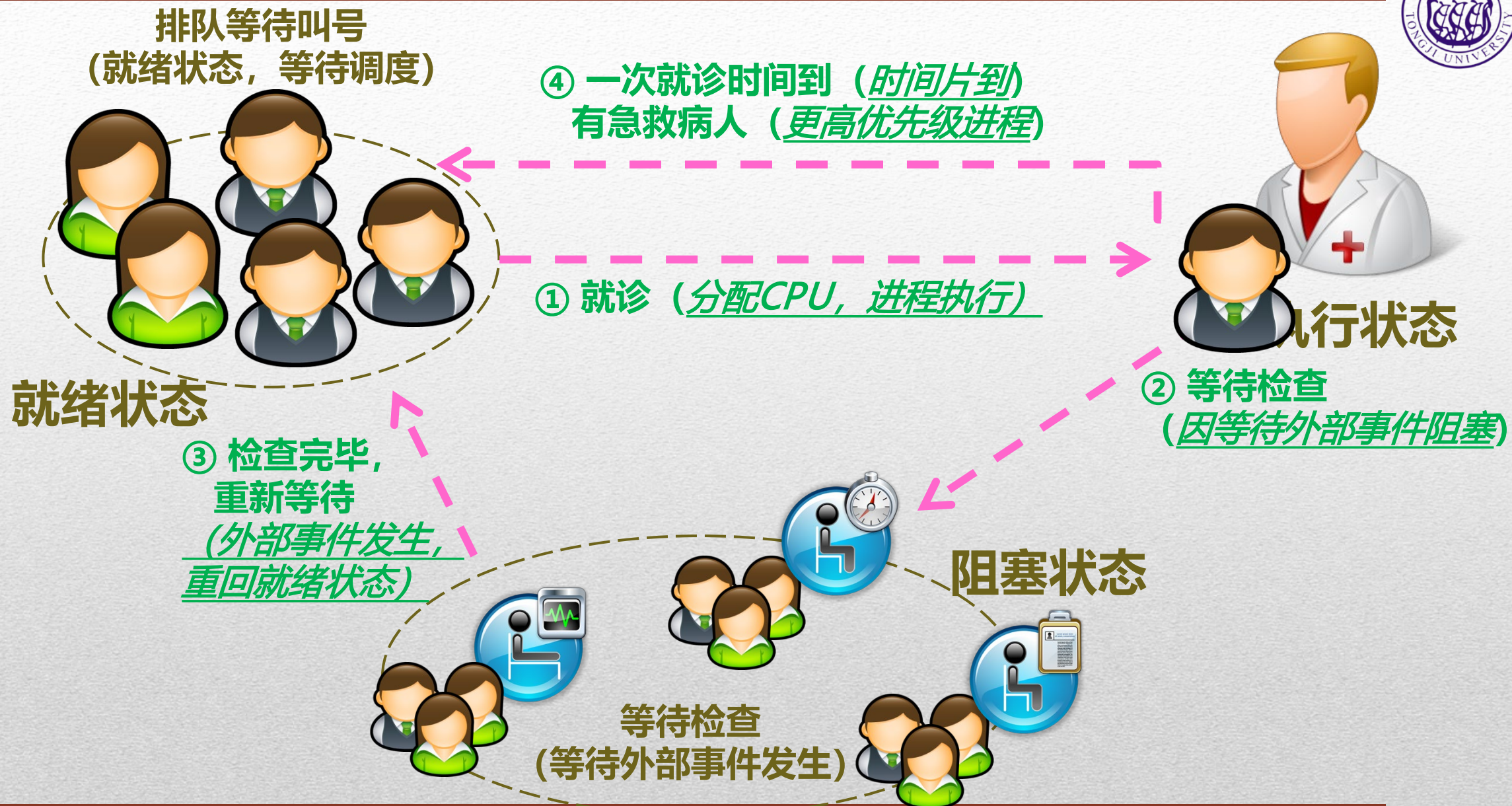




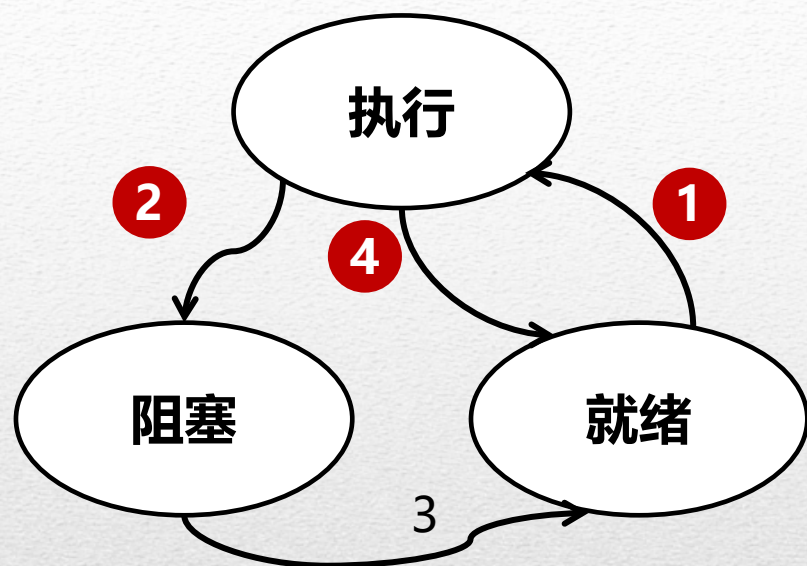








进程调度状态的控制



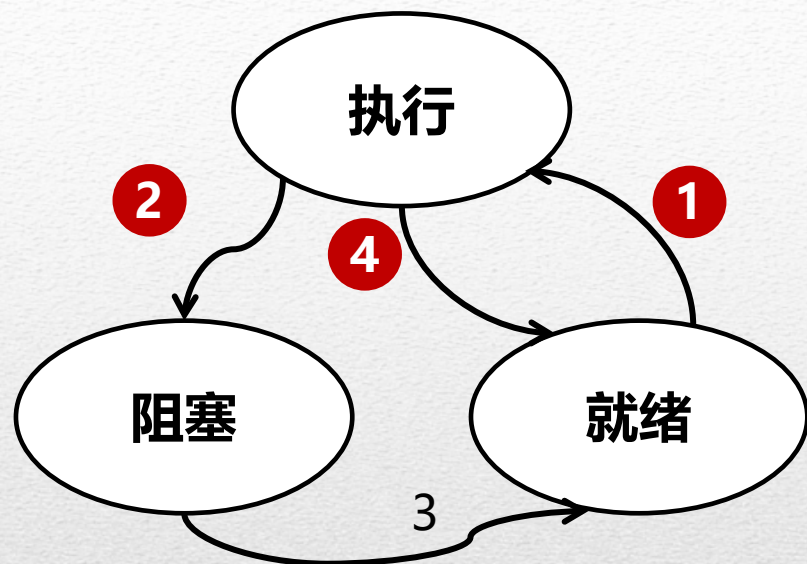
进程 “下台” / “上台”

引起进程切换调度的事件：

(不同的调度方式会不同)

1. 进程时间片到
2. 有更高优先级的进程就绪
3. 进程阻塞，无法继续执行

进程调度状态的控制



进程“下台” / “上台”

引起进程切换调度的事件:

(不同的调度方式会不同)

1. 进程时间片到
2. 有更高优先级的进程就绪
3. 进程阻塞, 无法继续执行

抢占式/剥夺式调度

现运行进程暂停, PCB中的调度状态

4 “执行” → “就绪”

非抢占式/进程主动放弃

现运行进程暂停, PCB中的调度状态

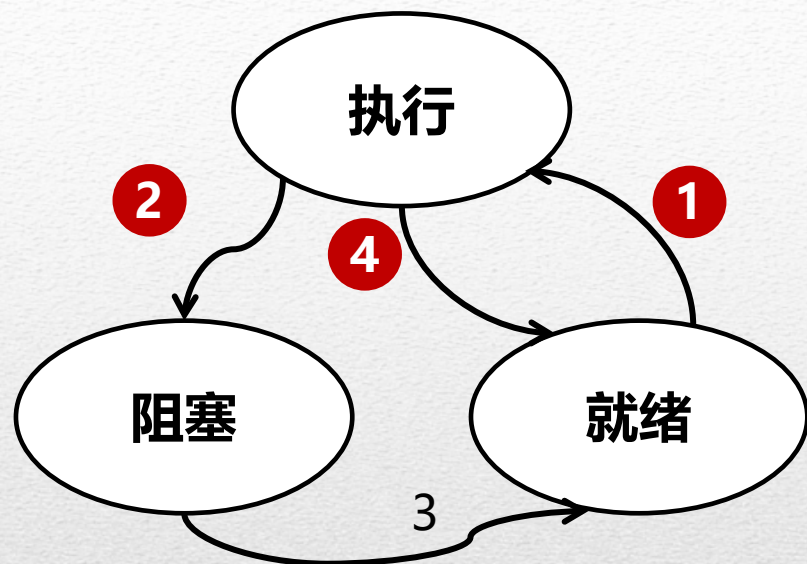
2 “执行” → “阻塞”

不同的调度方式

实时性高
但开销大

实现简单, 开销小
难于满足紧急任务
的需求

进程调度状态的控制



进程“下台” / “上台”

引起进程切换调度的事件:

(不同的调度方式会不同)

1. 进程时间片到
2. 有更高优先级的进程就绪
3. 进程阻塞, 无法继续执行

执行进程切换调度 (由**调度程序**完成):

1. **保存**现执行进程**工作现场**信息在其PCB中
2. **选择**另一个**就绪进程**, 状态“**就绪**” → “**执行**” ①
3. 用该进程PCB中的工作现场信息**恢复现场**

进程的上下文切换

“下台” 进程未来某时刻会被调度程序重新选中而“上台”

抢占式/剥夺式调度

现运行进程暂停, PCB中的调度状态

④ “**执行**” → “**就绪**”

非抢占式/进程主动放弃

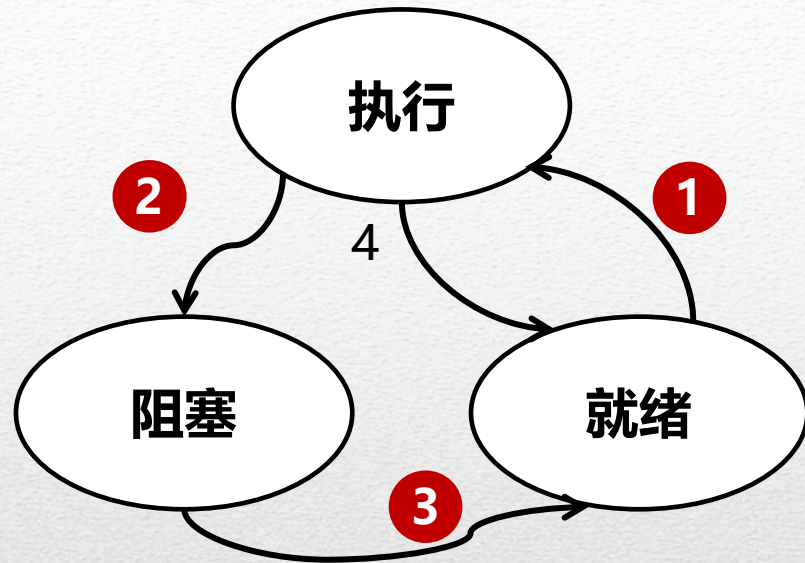
现运行进程暂停, PCB中的调度状态

② “**执行**” → “**阻塞**”



- 调度算法
- 调度时机
- 调度过程
- 进程死锁

进程调度状态的控制



进程的阻塞与唤醒

引起进程阻塞的事件:

1. 请求系统服务
 2. 启动某个操作
 3. 无新工作可做
 4.
- (进程无法再继续执行下去)

进程**阻塞过程** (由**阻塞程序**完成):

1. 立即停止执行
2. PCB中的进程状态 “**执行**” → “**阻塞**”
3. PCB进入阻塞队列
4. 由**调度程序**完成进程**切换调度**



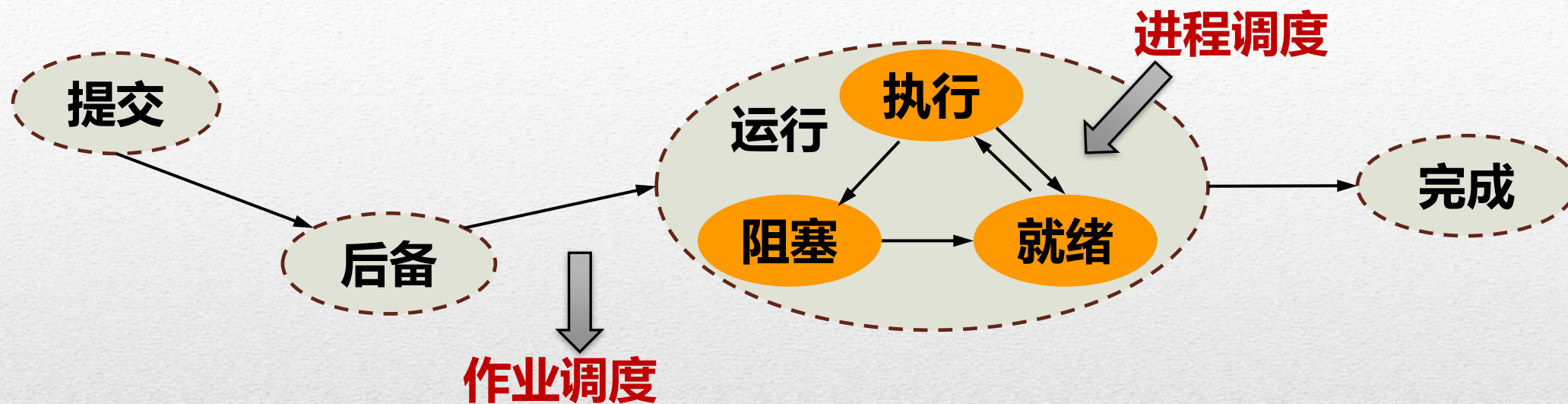
进程不能永远“睡觉”，必须在某个时间被唤醒，两个过程必须成对出现

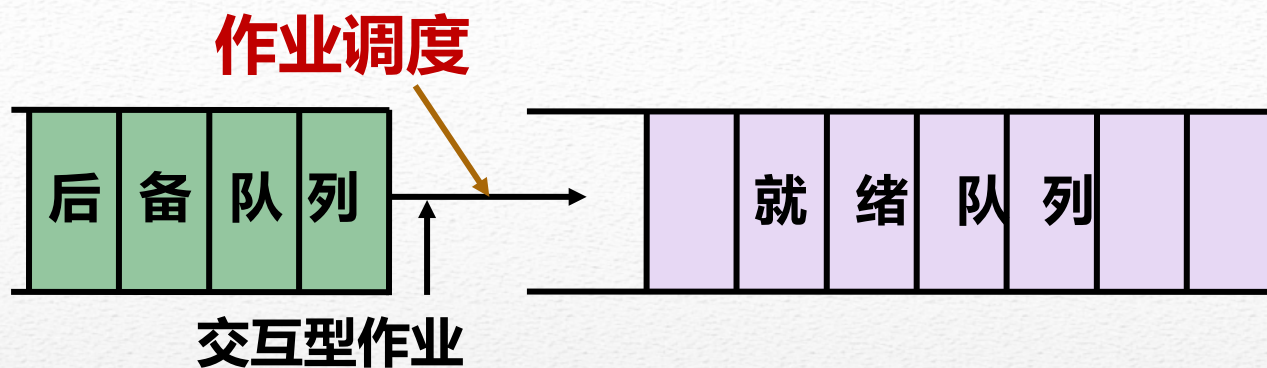
进程**唤醒过程** (由**唤醒程序**完成):

1. 将PCB从阻塞队列中移出
2. PCB中的进程调度状态 “**阻塞**” → “**就绪**”
3. 由调度算法决定是否**切换调度**

作业调度

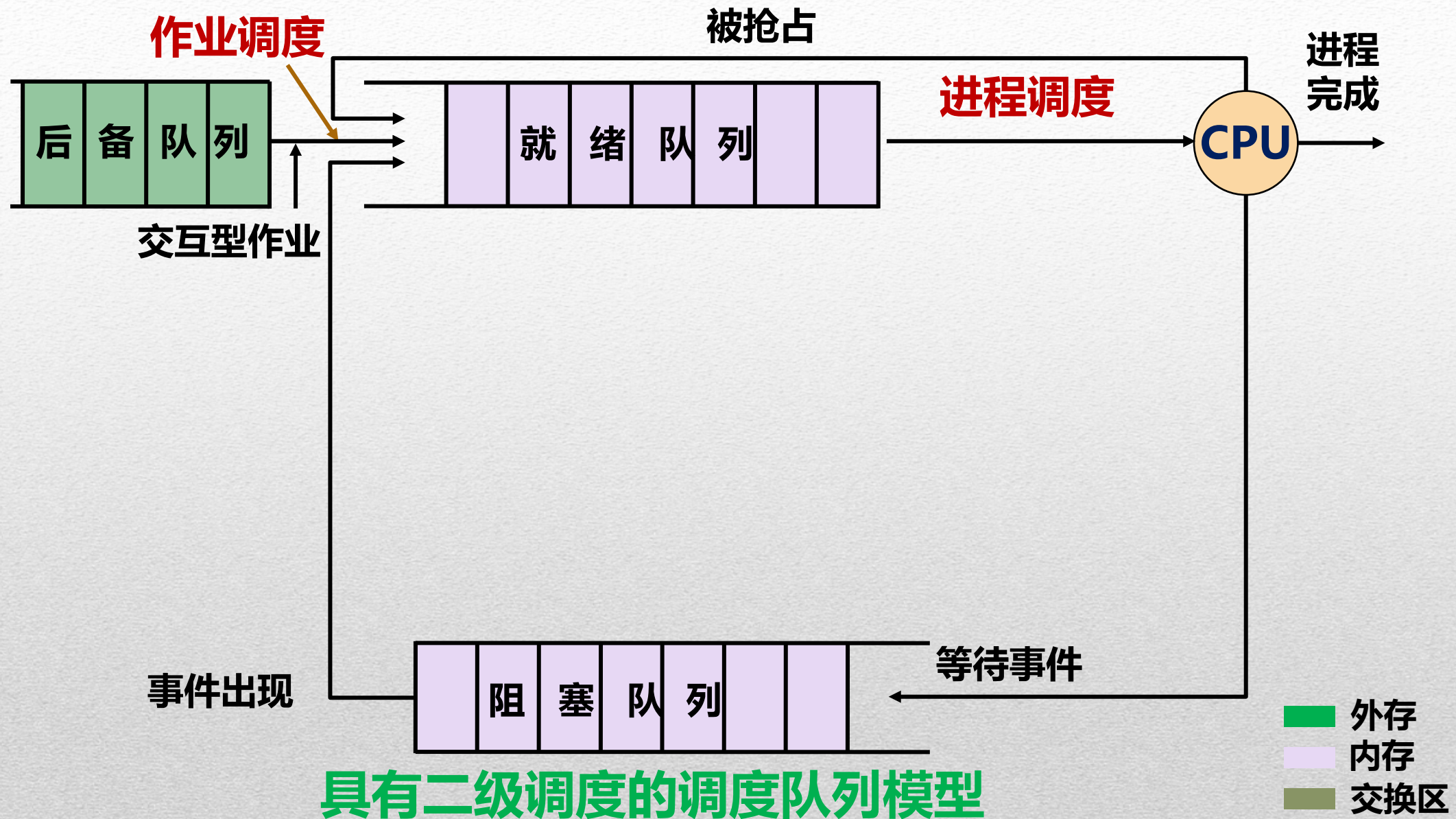
作业经历提交、后备、运行、完成四个状态





具有一级调度的调度队列模型

- 外存
- 内存
- 交换区



处理机调度要考虑的因素：

1. 调度本身的开销
2. 相关数据结构的维护
3. 各种不同的调度对象：I/O密集型，CPU密集型，长作业，短作业



选择调度方式和调度算法的若干准则

1. 周转时间短——批处理型作业

$$\begin{aligned}\text{周转时间} &= \text{作业完成时间} - \text{作业提交时间} \\ &= \text{作业运行时间} + \text{作业等待（后备）时间}\end{aligned}$$

2. 响应速度快——交互型作业

3. 高优先权优先

4. 截至时间保证——实时型作业



调度算法

1. 先来先服务算法 (FIFO)

作业调度

从后备队列中选择一个或多个最先进入该队列的作业，将它们调入内存、为它们分配资源、创建进程，放入就绪队列。

进程调度

从就绪队列中选择一个最先进入该队列的进程，为之分配处理机。该进程一直运行到完成或阻塞时才放弃处理机。

非抢占

优点：

非抢占式，实现简单。

缺点：

1. 长作业（进程）有利，短作业（进程）吃亏；
2. CPU密集型作业（进程）有利，I/O密集型作业（进程）吃亏。



调度算法

2. 短作业/进程优先 (SJ/PF)

周转时间
响应时间
优先权

作业调度

从后备队列中选择一个或多个估计运行时间最短的作业，将它们调入内存、为它们分配资源、创建进程，放入就绪队列。

进程调度

从就绪队列中选择一个估计运行时间最短的进程，为之分配处理机。该进程一直运行到完成或阻塞时才放弃处理机。

非抢占

优点:

当所有作业/进程同时到达时，平均周转时间最短

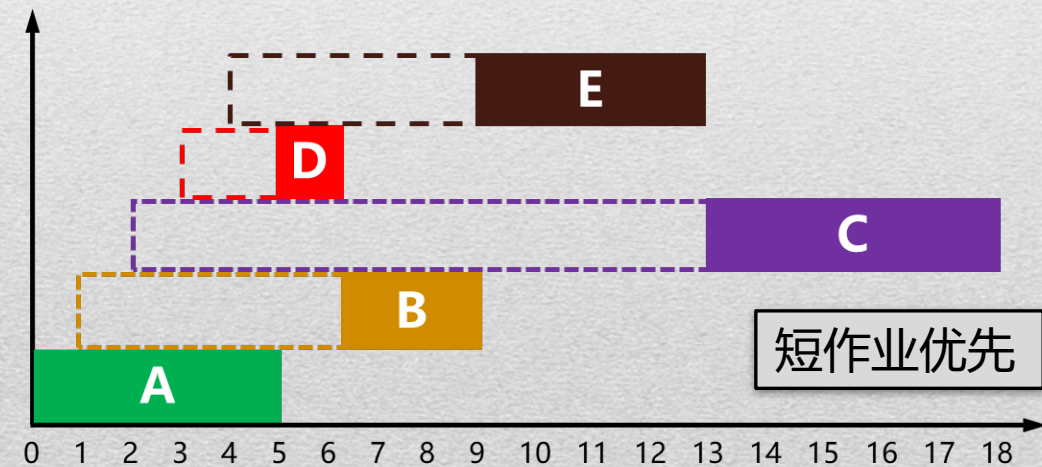
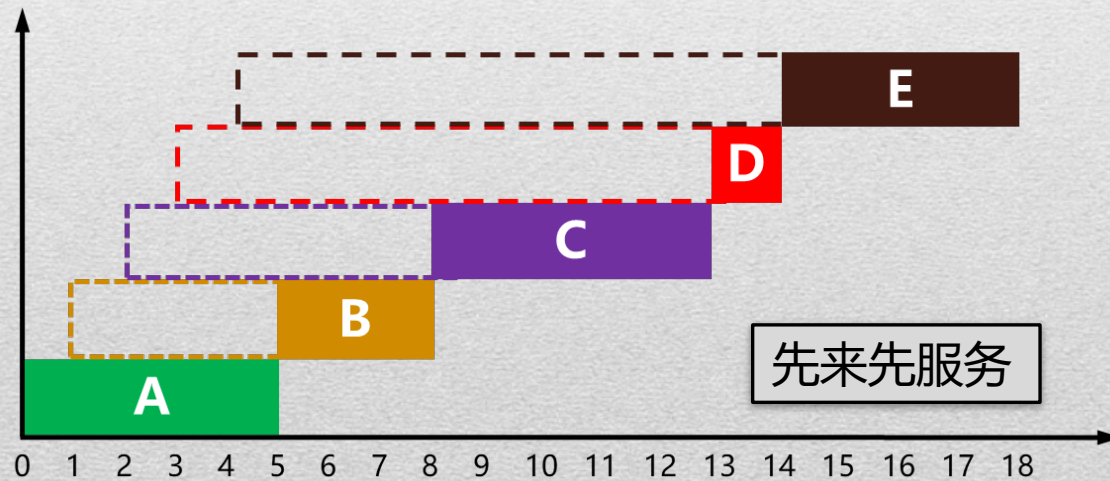
缺点:

1. 短作业（进程）有利，长作业（进程）吃亏；
2. 未考虑作业（进程）的紧迫程度。



进程调度算法周转时间比较

进程名	A	B	C	D	E	平均
到达时间	0	1	2	3	4	
服务时间	5	3	5	1	4	
先来先服务	5	7	11	11	14	9.6
短作业优先	5	8	16	<u>3</u>	9	8.2





调度算法

2. 短作业/进程优先 (SJ/PF)

■ 周转时间
■ 响应时间
■ 优先权

作业调度

从后备队列中选择一个或多个估计运行时间最短的作业，将它们调入内存、为它们分配资源、创建进程，放入就绪队列。

进程调度

从就绪队列中选择一个估计运行时间最短的进程，为之分配处理机。该进程一直运行到完成或阻塞时才放弃处理机。

非抢占

优点：

当所有作业/进程同时到达时，平均周转时间最短

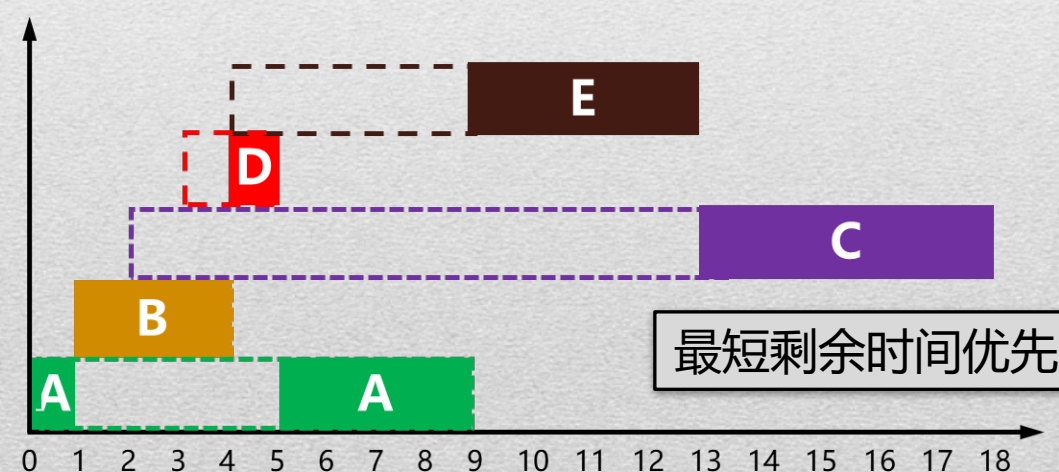
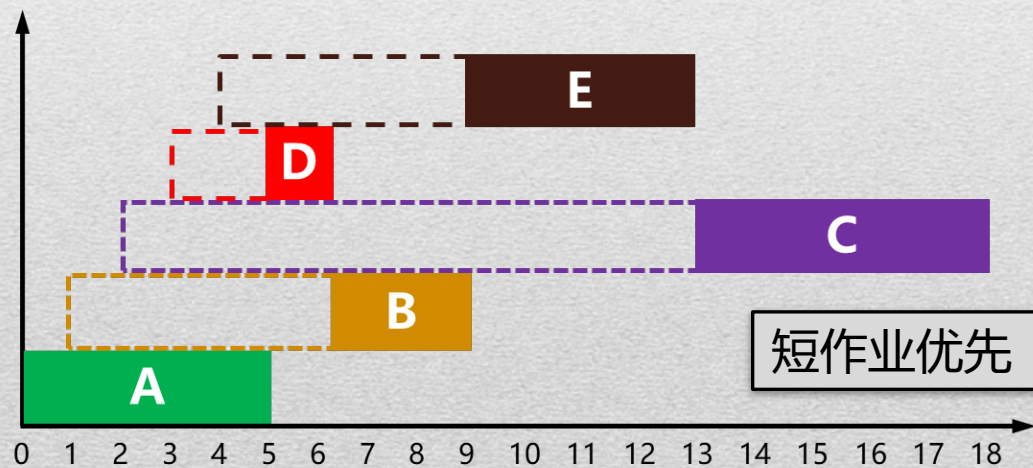
缺点：

1. 短作业（进程）有利，长作业（进程）吃亏；
2. 未考虑作业（进程）的紧迫程度。

抢占式改进：
最短剩余时间优先

进程调度算法周转时间比较

进程名	A	B	C	D	E	平均
到达时间	0	1	2	3	4	
服务时间	5	3	5	1	4	
先来先服务	5	7	11	11	14	9.6
短作业优先	5	8	16	<u>3</u>	9	8.2
最短剩余时间优先	9	3	16	<u>2</u>	9	7.8





调度算法

4. 时间片轮转调度算法 (RR)

交互式系统 (分时系统)
中的进程调度



时间片大小将
影响系统性能

时间片到



进程调度

每个进程轮流使用CPU固定时间片后将CPU让给其它进程，自己进入就绪队列等待下一轮调度。

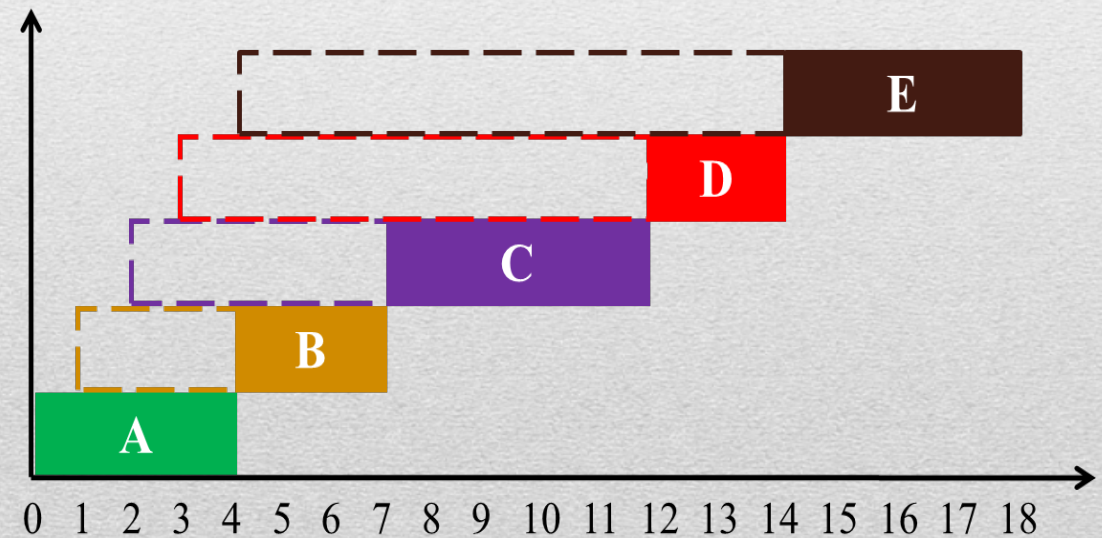
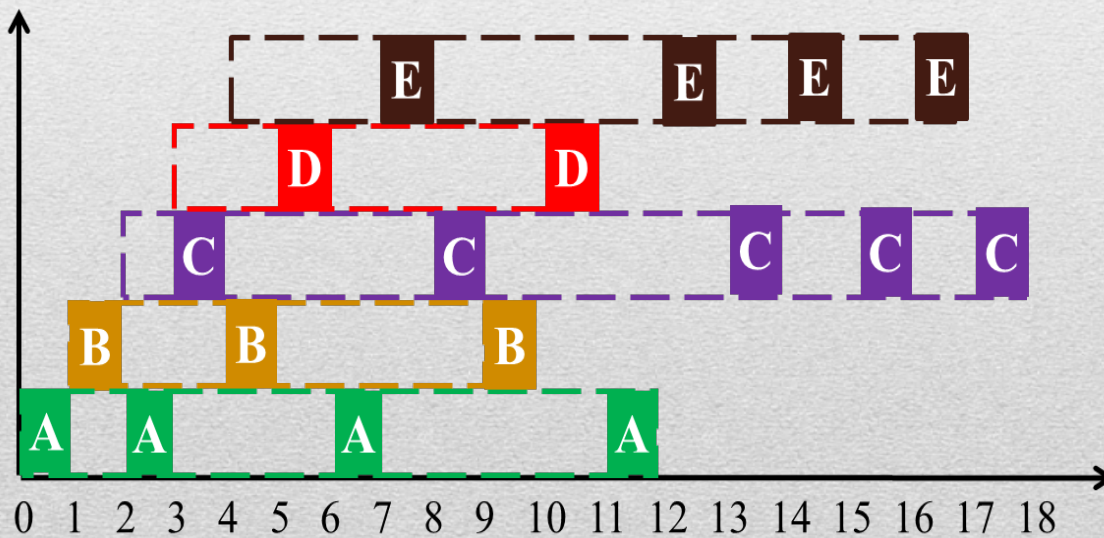
抢占

优点：各进程能够比较均衡地共享使用处理机。

缺点：系统的效率与时间片的设置密切相关。时间片过大，与用户的交互性就差；时间片过小，进程间切换过于频繁，一个进程需要轮转多次才能到达终点，系统开销就会增大。

时间片选取对调度性能的影响

进程名	A	B	C	D	E	平均
到达时间	0	1	2	3	4	
服务时间	4	3	5	2	4	
周转时间 (Q=1)	12	9	16	8	13	11.6
周转时间 (Q=5)	4	6	10	11	14	9





调度算法

3. 最高优先级优先调度算法 (FPF)

■ 周转时间
■ 响应时间
■ 优先权

作业调度

从后备队列中选择一个或多个**优先权最高**的作业，将它们调入内存、分配资源、创建进程，放入就绪队列。

进程调度

从就绪队列中选择一个**优先权最高**的进程，为之分配处理机。

多数操作系统中采用的进程调度算法

非抢占

进程一直运行到完成；或发生某事件而阻塞，系统将处理机分配给另一个优先级最高的进程。
批处理系统、要求不高的实时系统

抢占式

进程执行中，调度程序可随时停止其执行，将处理机分配给新出现的优先级更高的进程。
实时系统、交互系统



调度算法

3. 最高优先权优先调度算法 (FPF)

周转时间
响应时间
优先权

静态优先权

创建进程时确定，且进程的整个运行期间保持不变。

进程类型：系统进程 > 用户进程

资源需求：需求少的 > 需求多的

优点：简单，不需要维护优先权。
缺点：高者恒高，不断地有高者转为就绪的话，低者将会“饥饿”。

例：某多道程序系统采用**抢占式静态优先级进程调度算法**。某段时间内有A、B、C三个进程，优先级 $C > B > A$ 。就绪时刻、计算与I/O所需时间如下表所示：


进程	进程就绪时刻	计算时间	I/O操作时间	计算时间
A	0ms	15ms	10ms	5ms
B	5ms	25ms	15ms	10ms
C	10ms	5ms	20ms	10ms

若采用多道方式运行，给出这三个进程运行完成总共所需的时间（忽略进行系统调度所需时间），采用多道方式运行比采用单道方式运行节省多少时间。

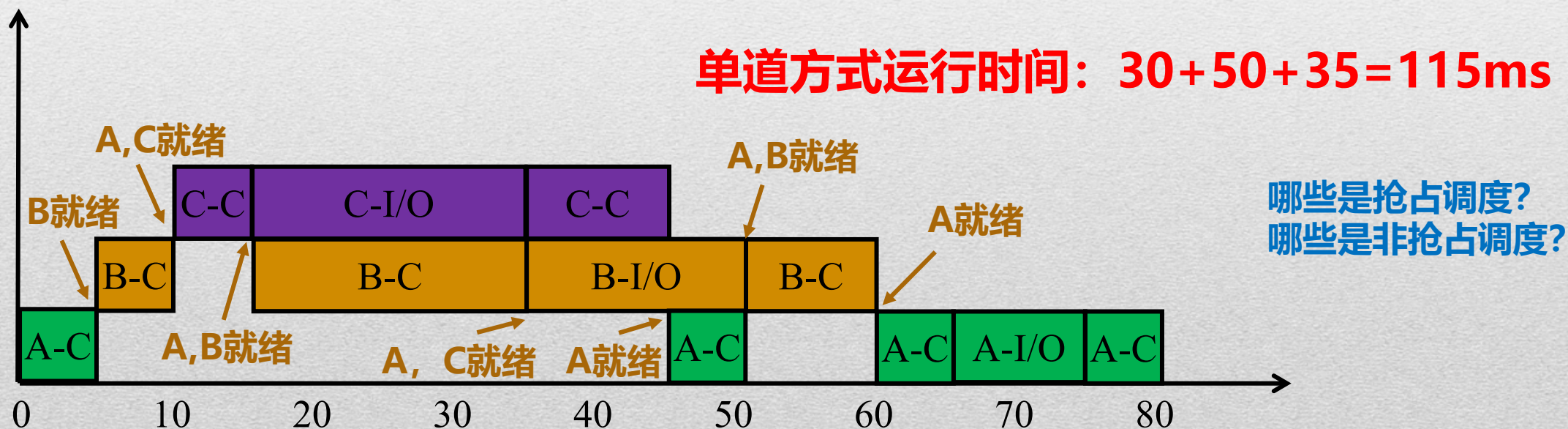


调度算法

3. 最高优先权优先调度算法 (FPF)

 周转时间
 响应时间
 优先权

进程	进程就绪时刻	计算时间	I/O操作时间	计算时间
A	0ms	15ms	10ms	5ms
B	5ms	25ms	15ms	10ms
C	10ms	5ms	20ms	10ms





调度算法

3. 最高优先权优先调度算法 (FPF)

■ 周转时间
■ 响应时间
■ 优先权

动态优先权

系统为刚生成的进程赋初始优先权，之后根据进程的行为动态调整优先权的值

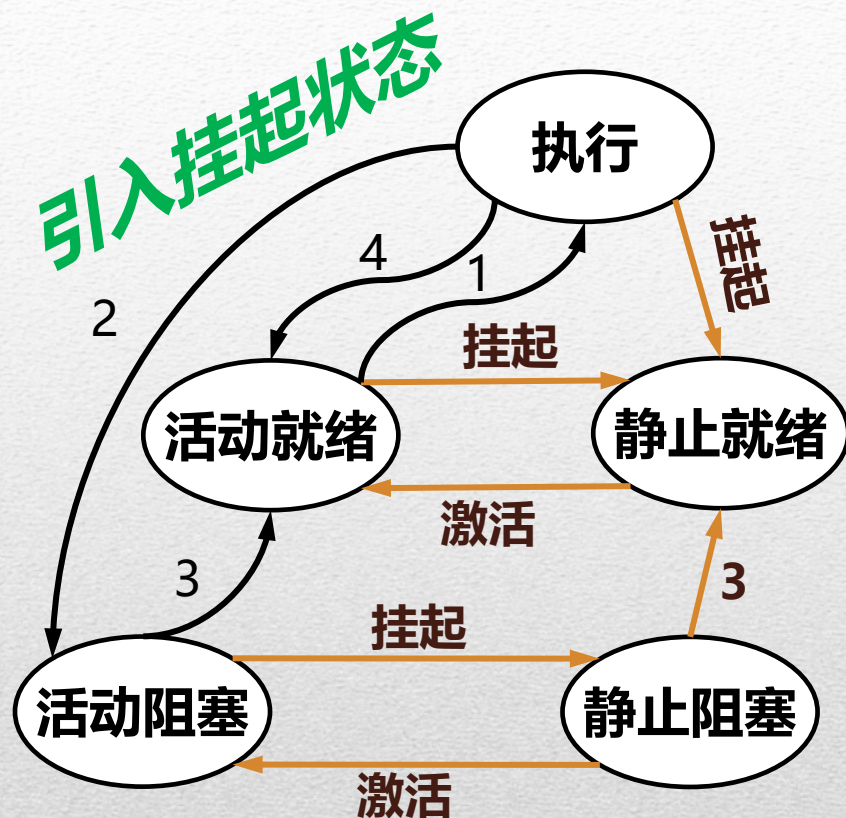
防止高者恒高，
低者“饥饿”。

如何调整？

正在使用CPU的进程优先级下降；很久未得到CPU的进程优先级升高；I/O密集型进程拥有较高的优先权

何时调整？
间隔太短，开销太大；间隔太长，不能及时反映变化
一般选择调度、时钟中断、陷入等时机调整

进程调度状态的控制



引起进程挂起的事件:

终端用户请求 父进程请求
操作系统负荷调节

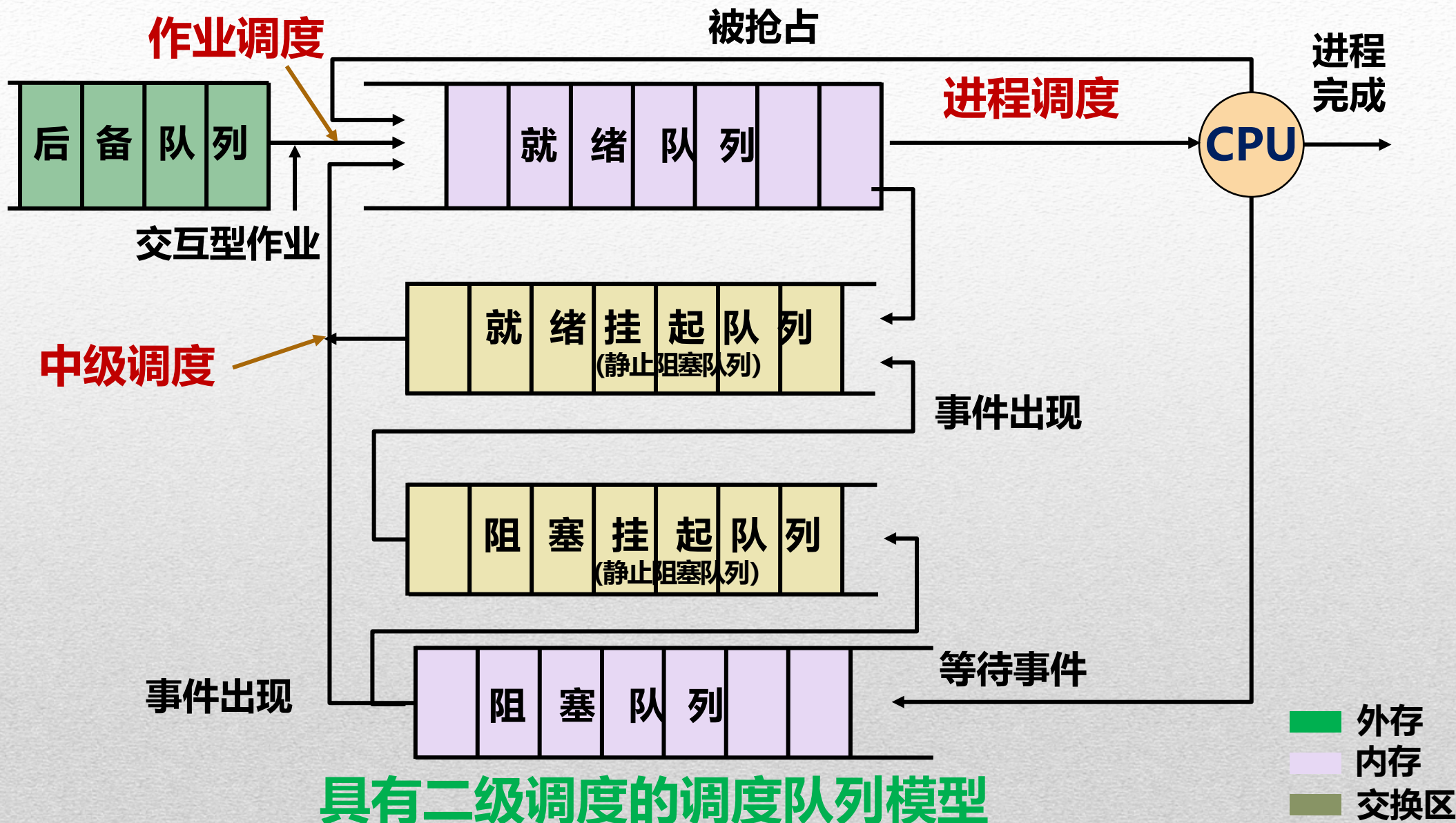
进程**挂起**过程:

1. 若为当前执行进程: 立即停止执行, PCB中的进程状态“执行” → “静止就绪”, 调度程序进行**切换调度**
2. 若非当前执行进程: PCB中的进程状态“活动就绪” → “静止就绪” / “活动阻塞” → “静止阻塞”

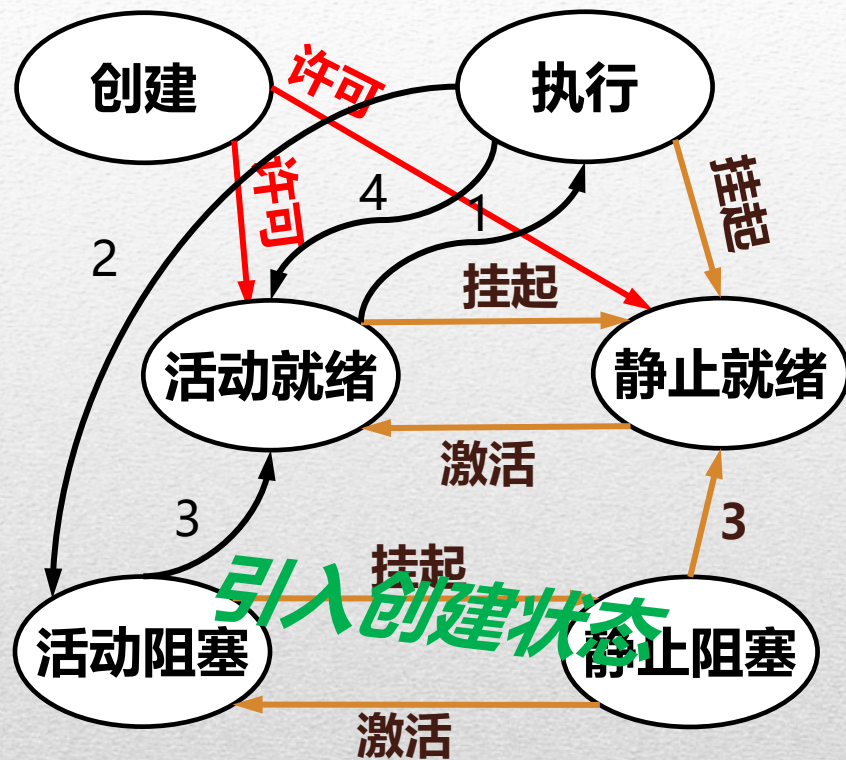
进程**激活**过程:

1. PCB中的进程状态“静止就绪” → “活动就绪” / “静止阻塞” → “活动阻塞”
2. 若转入“活动就绪”, 则PCB进入就绪队列, 由调度算法决定是否**切换调度**

两个过程也必须成对出现



进程调度状态的控制



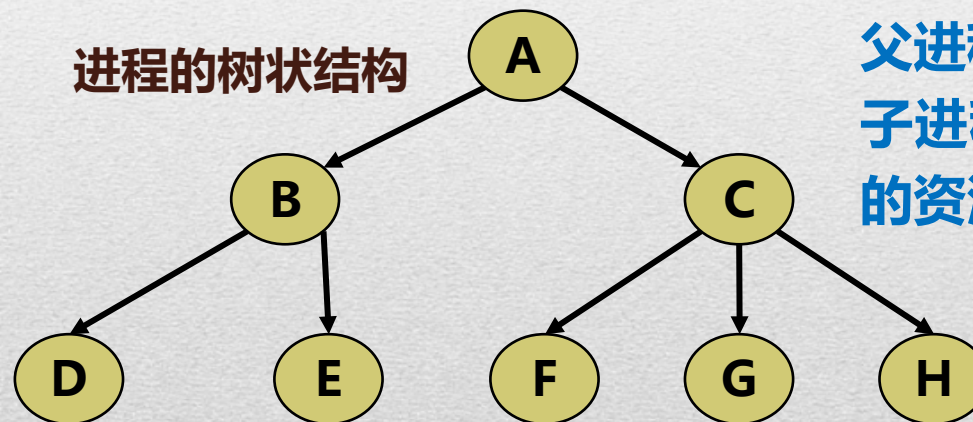
引起进程创建的事件:

用户登录
提供服务

作业调度
应用请求

进程**创建**过程:

1. 申请空白PCB
2. 为进程分配资源 (内存空间)
3. PCB初始化 (标识、处理机状态、进程调度信息)
4. 进入就绪队列 (活动? 静止?)

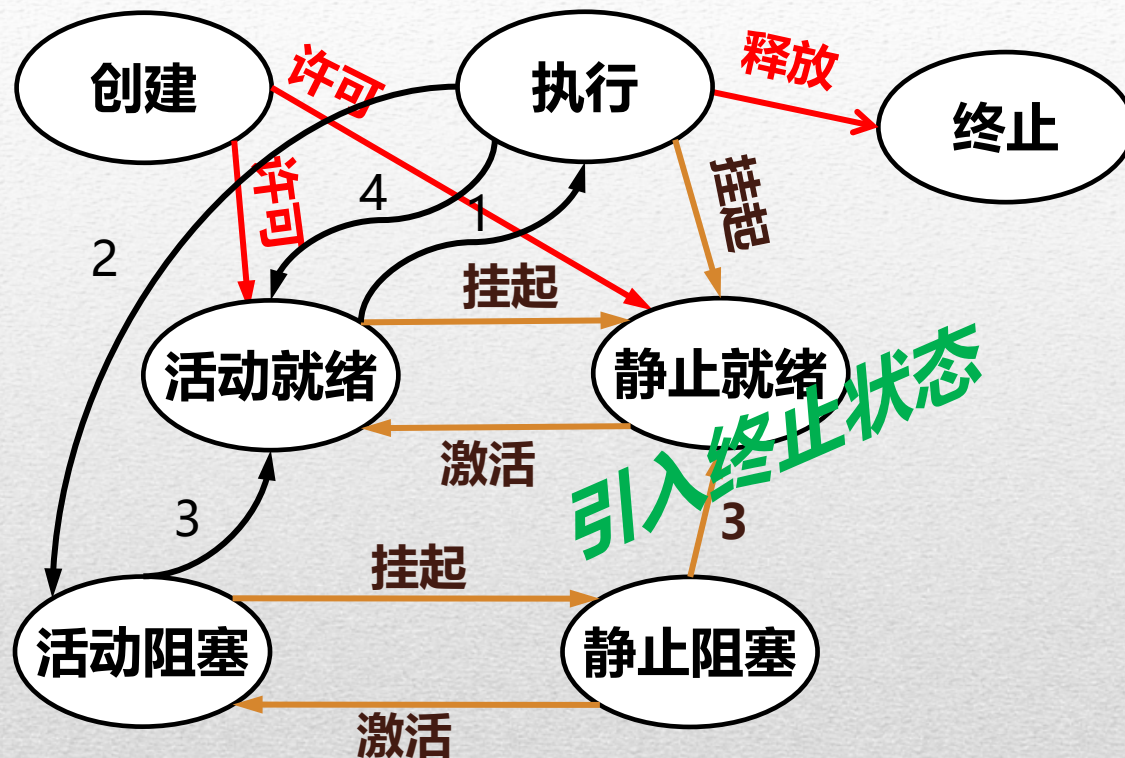


进程的树状结构

父进程创建子进程
子进程拥有父进程的
资源

子进程撤销时, 资源归还父进程
父进程撤销时, 撤销所有子进程

进程调度状态的控制



引起进程终止的事件:

正常结束 异常结束
外界干预 (人为、父进程)

进程终止过程:

1. 从PCB中读出该进程的状态
2. 立即终止该进程的执行
3. 终止其所有子孙进程
4. 释放全部资源
5. 移除该进程PCB
6. 进程切换调度

只有当删除进程PCB后，进程才彻底消亡



进程

动态性，具有生命周期
“由创建而产生，由调度而执行，由撤销而消亡”

Very Important!

程序并发执行带来的问题.....

 资源共享  各种程序活动的相互依赖与制约

为了解决程序并发执行带来的问题：



一组数据与指令代码的集合	结构特征 代码段、数据段、堆栈段、 进程控制块
静态的 存放在某种介质上	动态性 ，具有生命周期 “由创建而产生，由调度而执行，由撤销而消亡”

- 多个进程实体可同时存在于内存中并发执行
- 独立运行、独立分配资源和独立接受调度的基本单位
- 按不可预知（异步）的速度向前推进

进程是程序的一次运行过程!!!



本节小结:

- 1 程序与进程的区别与联系
- 2 进程的调度状态及状态转换

请阅读教材：29页 ~ 47页 (2.1节 ~ 2.3节)



E01：并发进程（进程基本概念与进程调度）