第二章

并发进程

方　钰

主 要 内 容

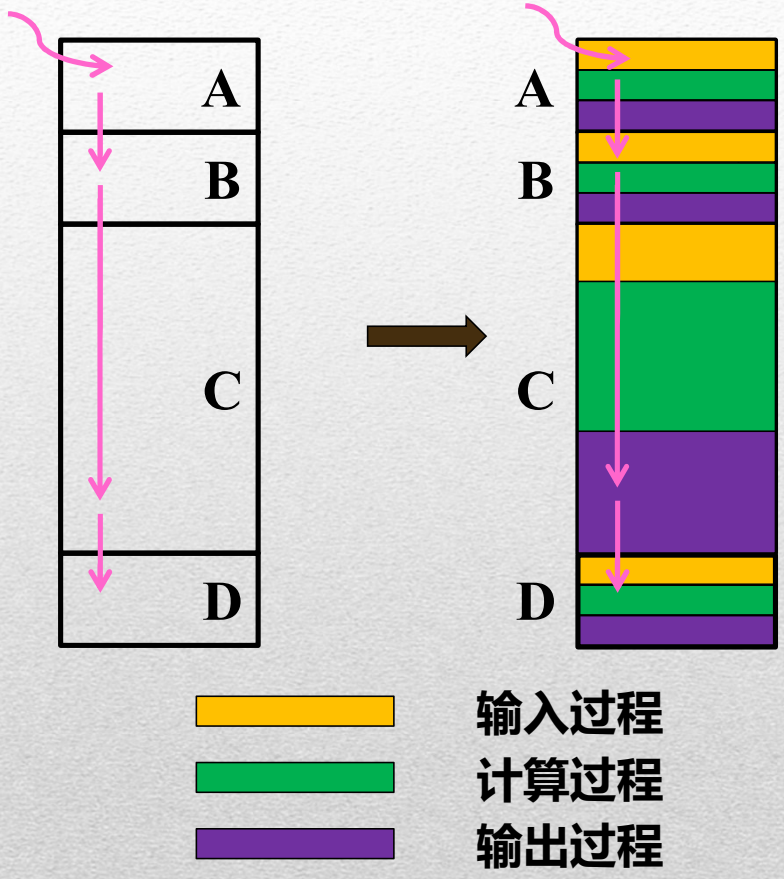
2.1 进程基本概念

2.2 UNIX进程

2.3 UNIX中断

2.4 进程通信

程序的顺序执行与并发执行



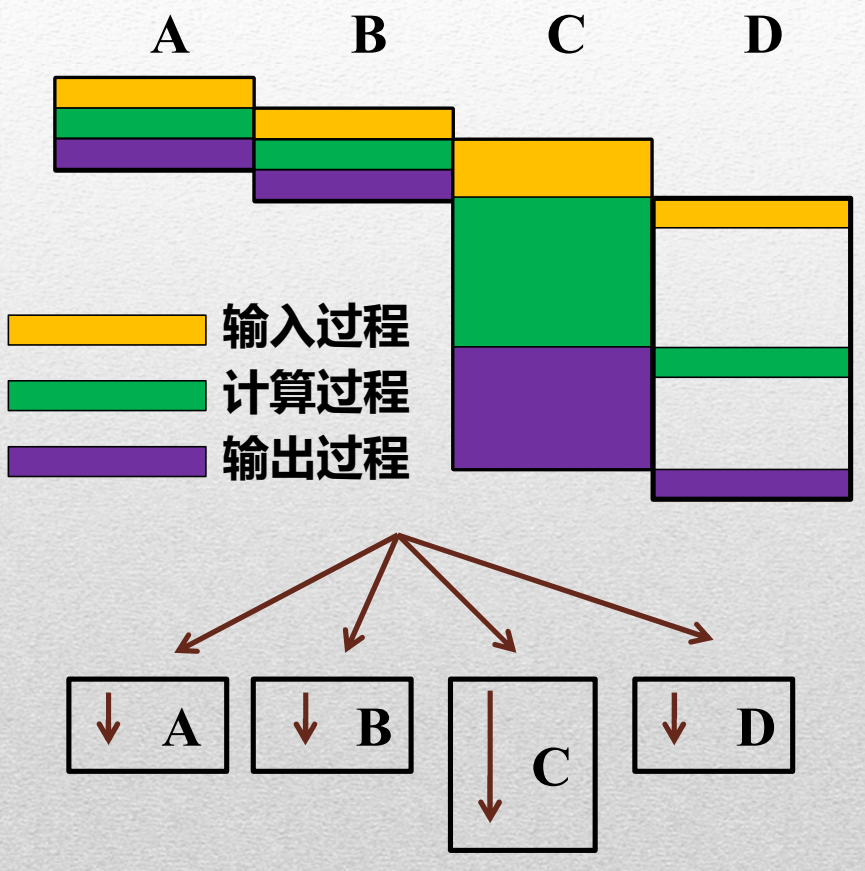
单道批处理系统

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 程序执行时独占全机，结果不受外界影响。

可再现性 只要执行时的环境和初始条件相同，结果即相同。

程序的顺序执行与并发执行



多道批处理系统

特征：

间断性：相互制约导致并发程序具有“执行—暂停—执行”这种间断性的活动规律。

开放性：多个程序共享系统中的资源。

不可再现性：结果与并发程序的执行速度有关。

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：


程序

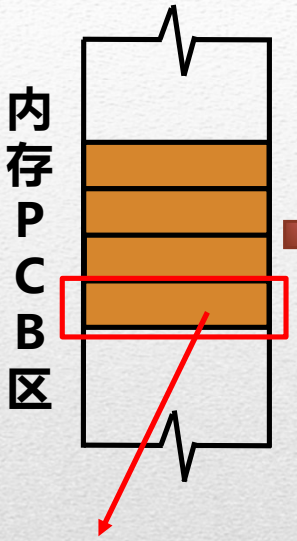

进程

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

进程管理

通用寄存器
程序计数器
程序状态字
栈指针...

进程状态
进程优先级
调度参数（*算法?*）
事件...

消息队列指针
信号量...
.....

存储管理

程序和数据地址
.....

其他资源管理

文件...、设备...

操作系统所需的、用于描述进程的当前状况以及控制进程运行的全部信息

处理机状态（CPU工作现场 / 虚拟CPU）

进程调度信息

进程通信

- 创建进程：申请空白PCB
- 需要调度进程：查询参数
- 实施进程调度：恢复处理机现场，找到程序和数据
- 进程执行：进程同步、通信，读写文件.....

整个生命周期中，系统通过PCB对进程进行控制

PCB是进程存在的唯一标志

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

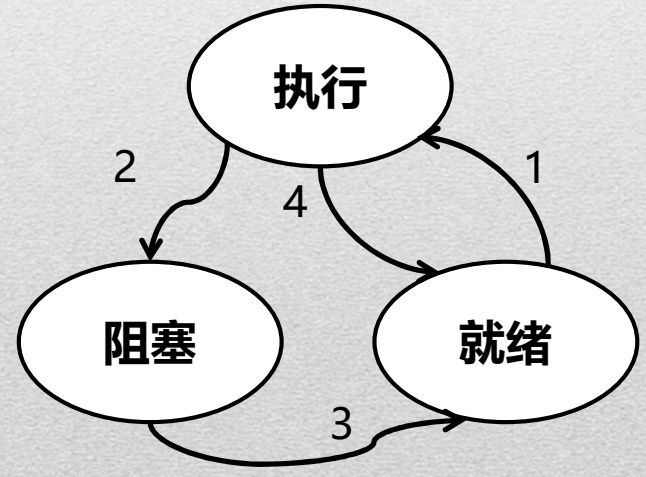
为了解决程序并发执行带来的问题：


程序 → 
进程

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

- 执行状态 （当前正在使用CPU）
- 就绪状态 （可运行；但CPU被占用，暂时无法运行）
- 阻塞状态 （ 无法运行，直到某一外部事件发生）



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

阻塞状态

等待检查
（等待外部事件发生）

排队等待叫号
（就绪状态，等待调度）

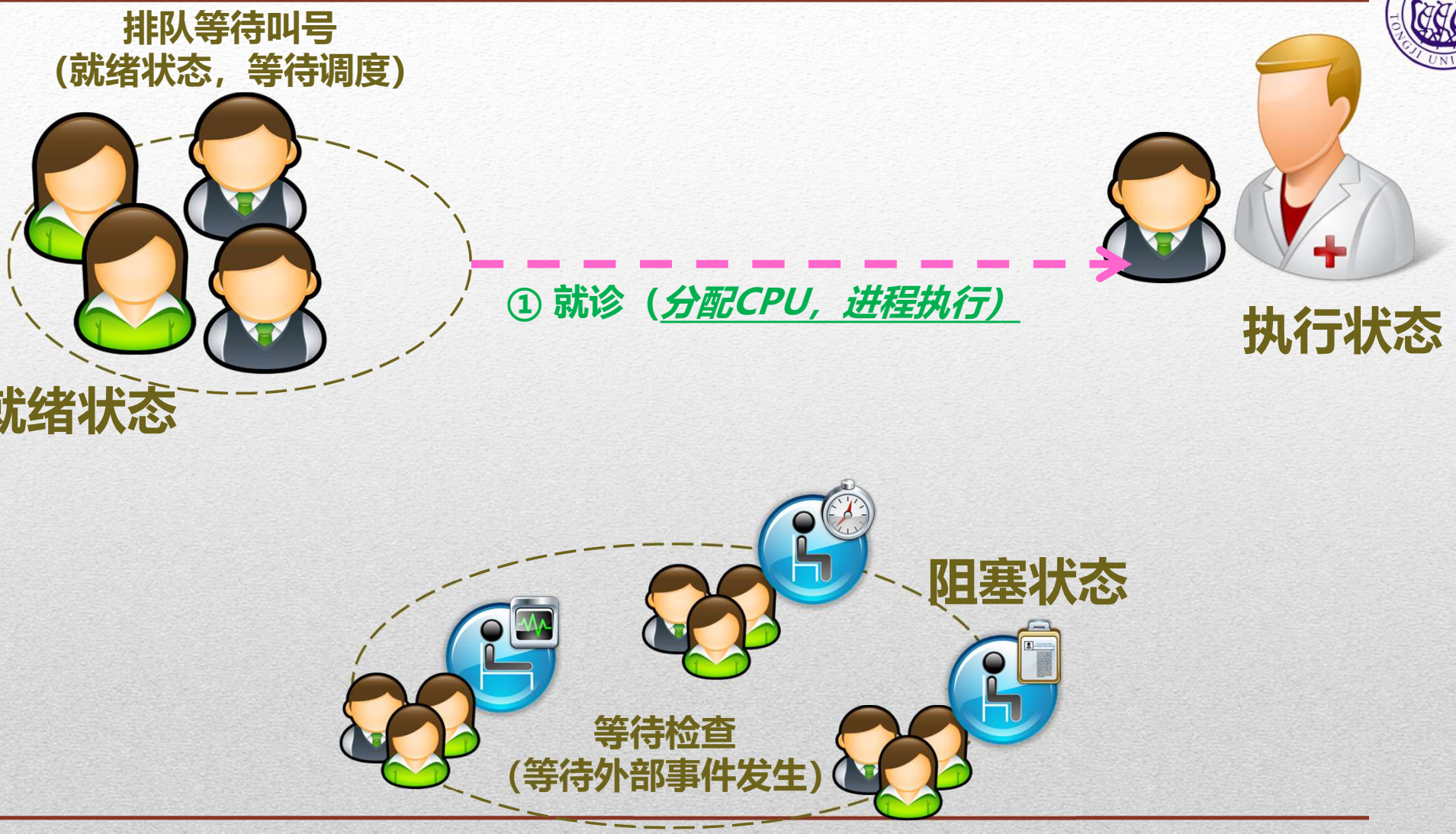
就绪状态

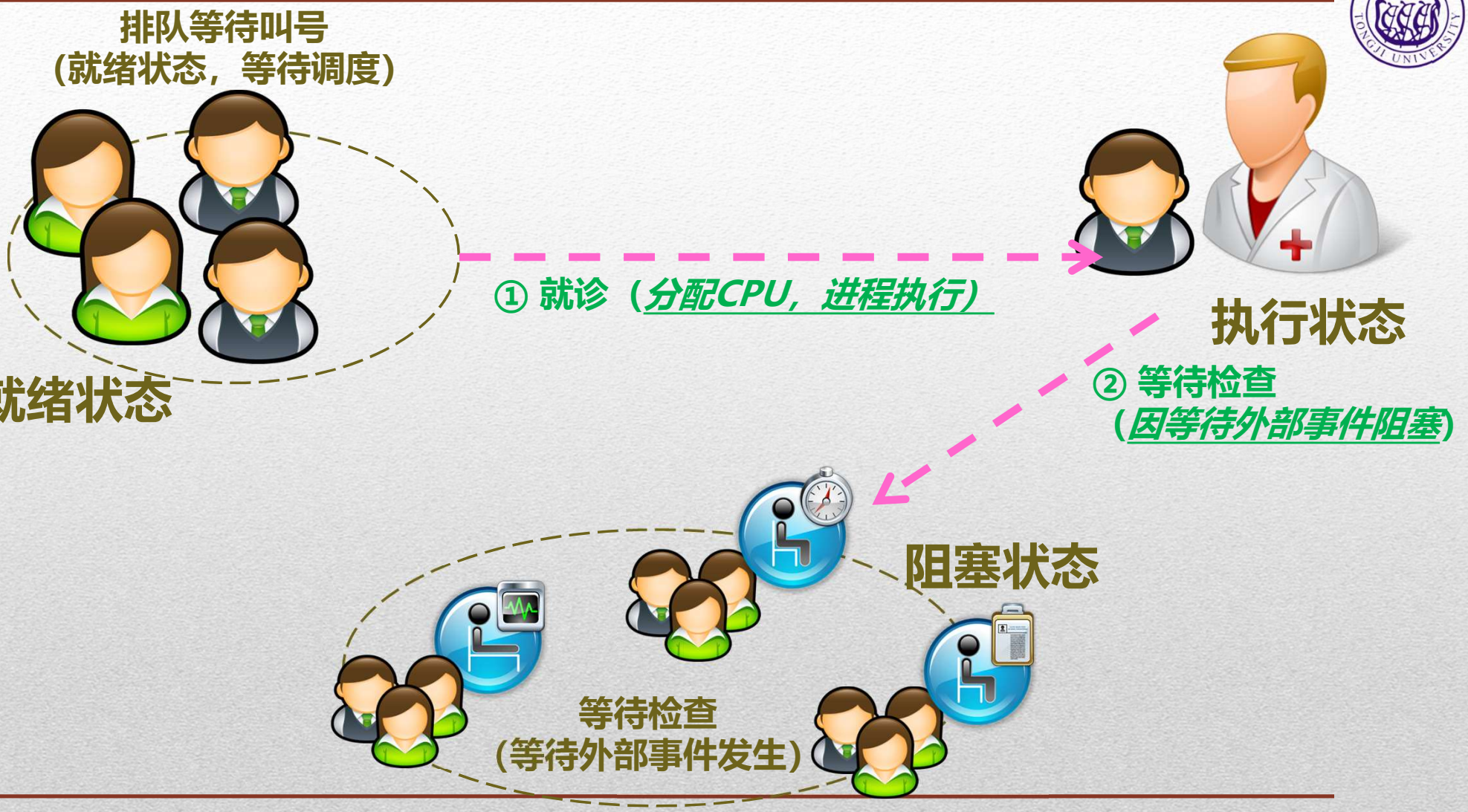
① 就诊（*分配CPU，进程执行*）

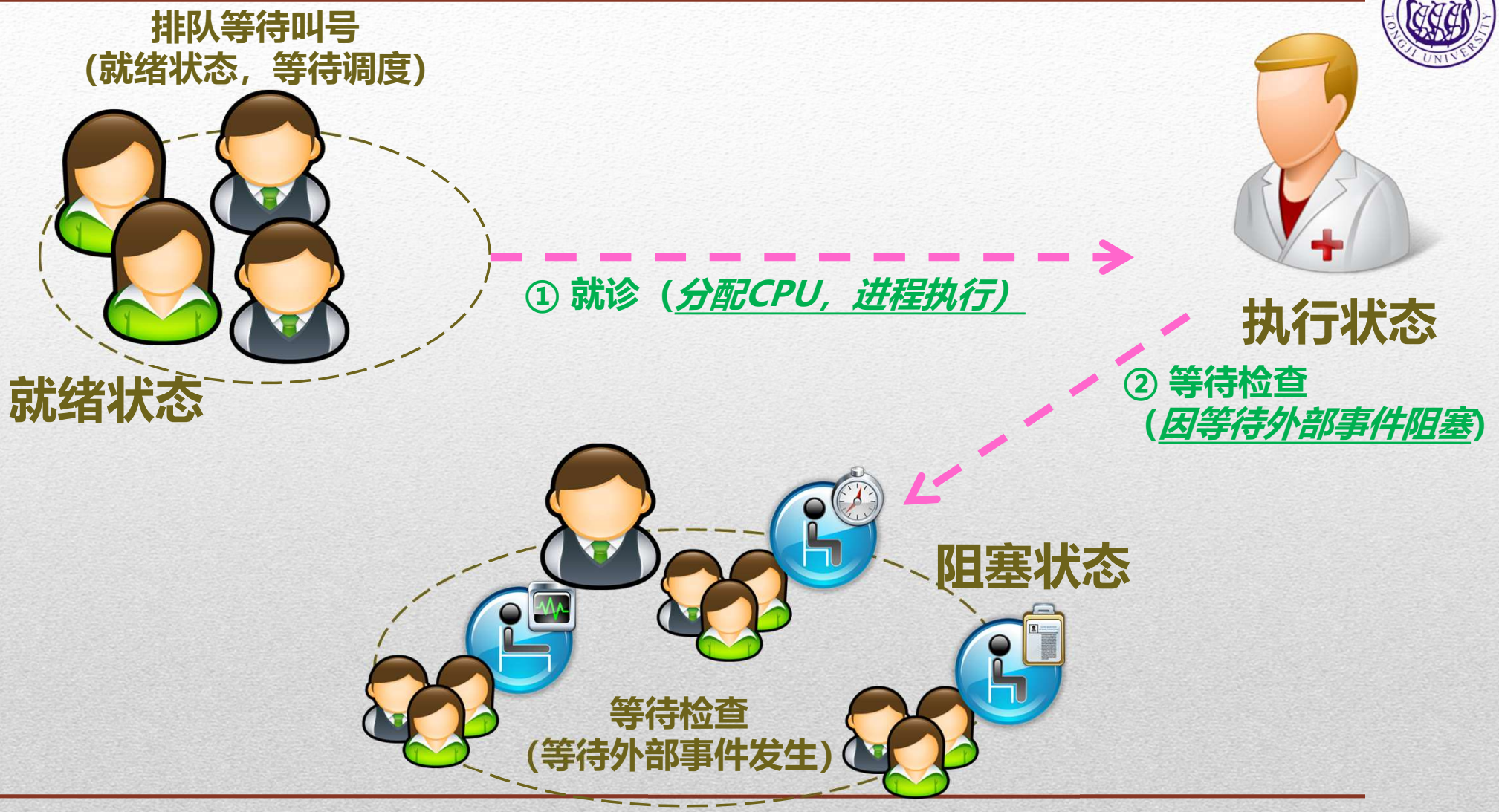
执行状态

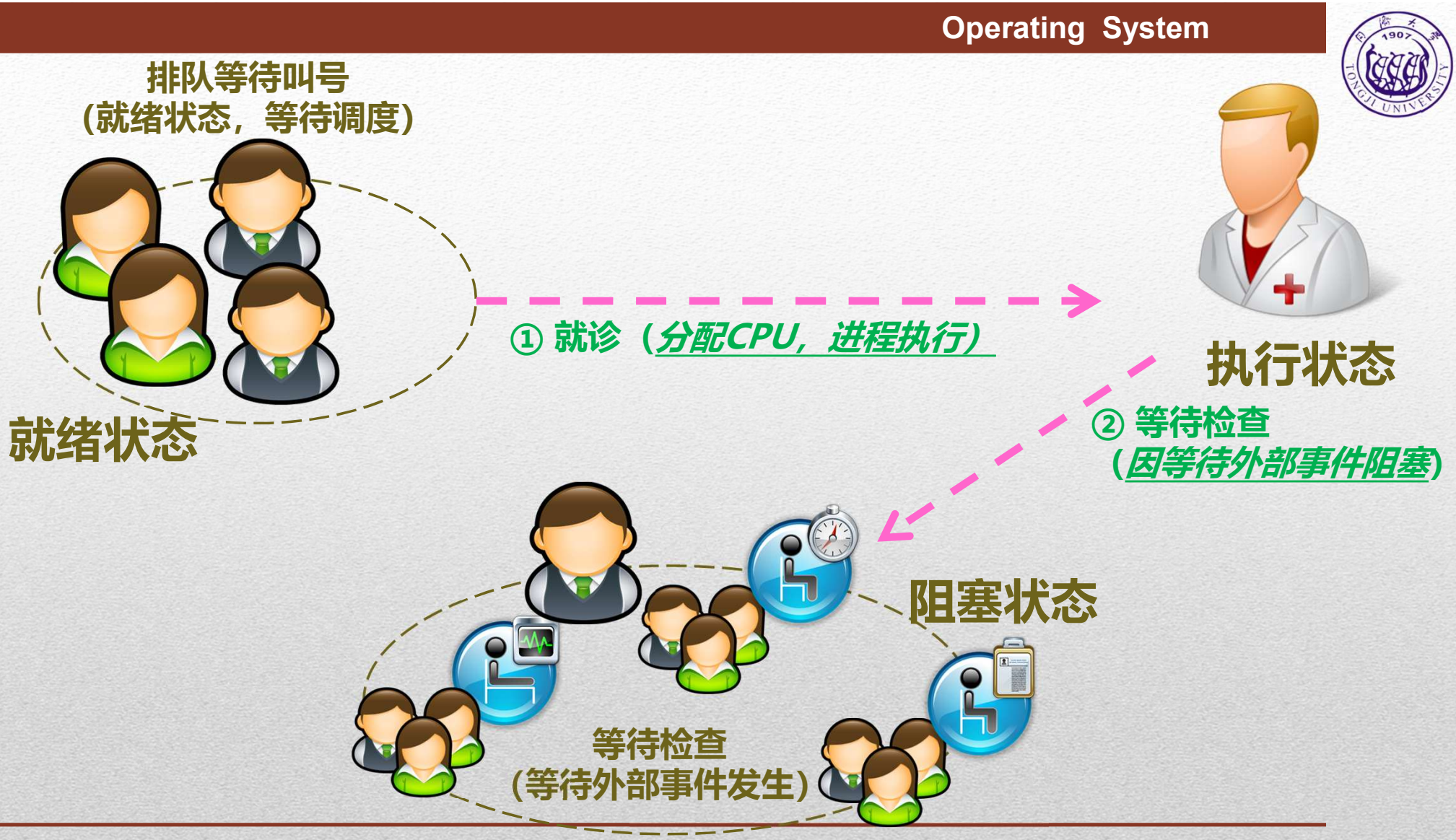
阻塞状态

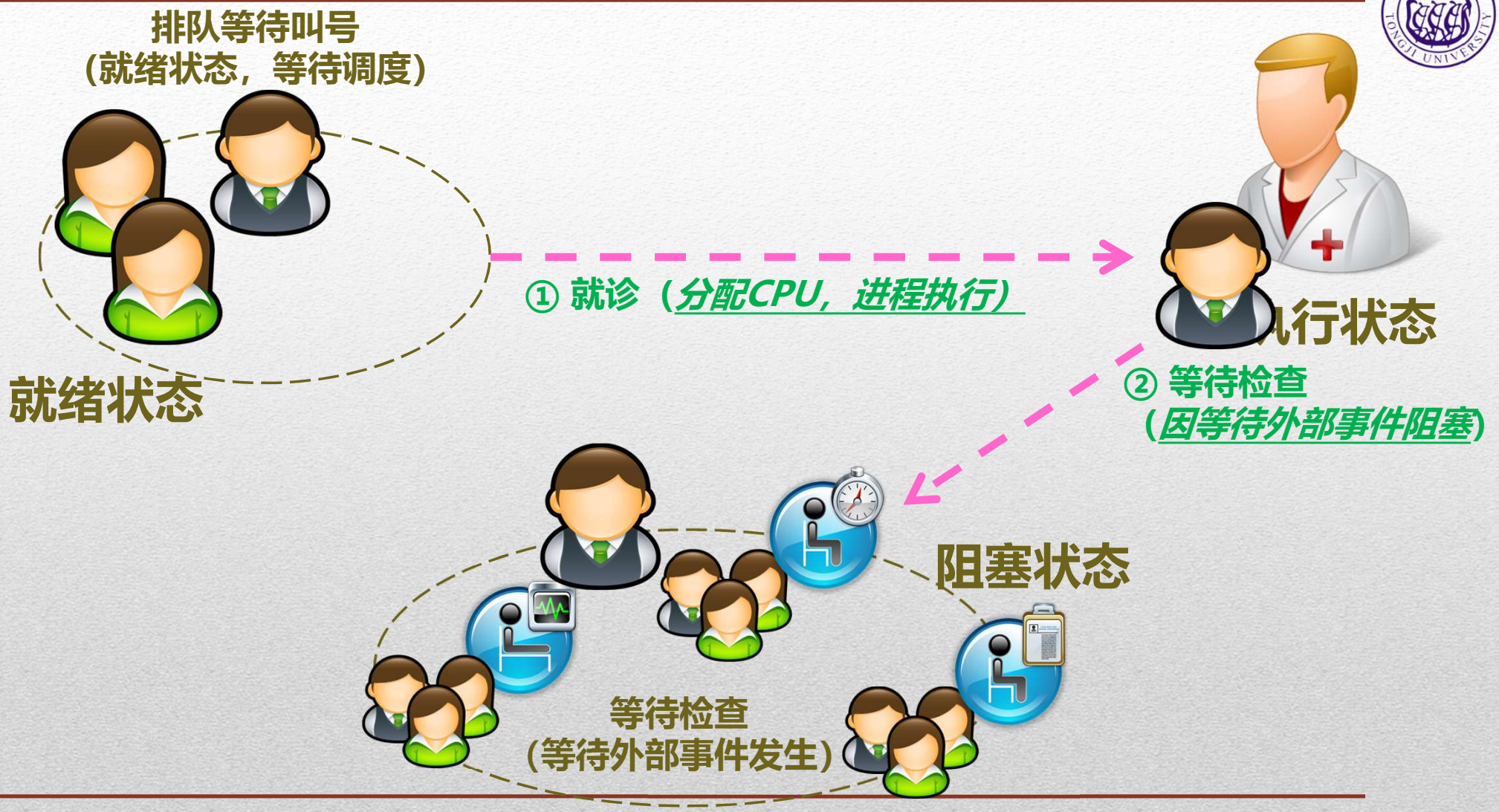
等待检查
（等待外部事件发生）

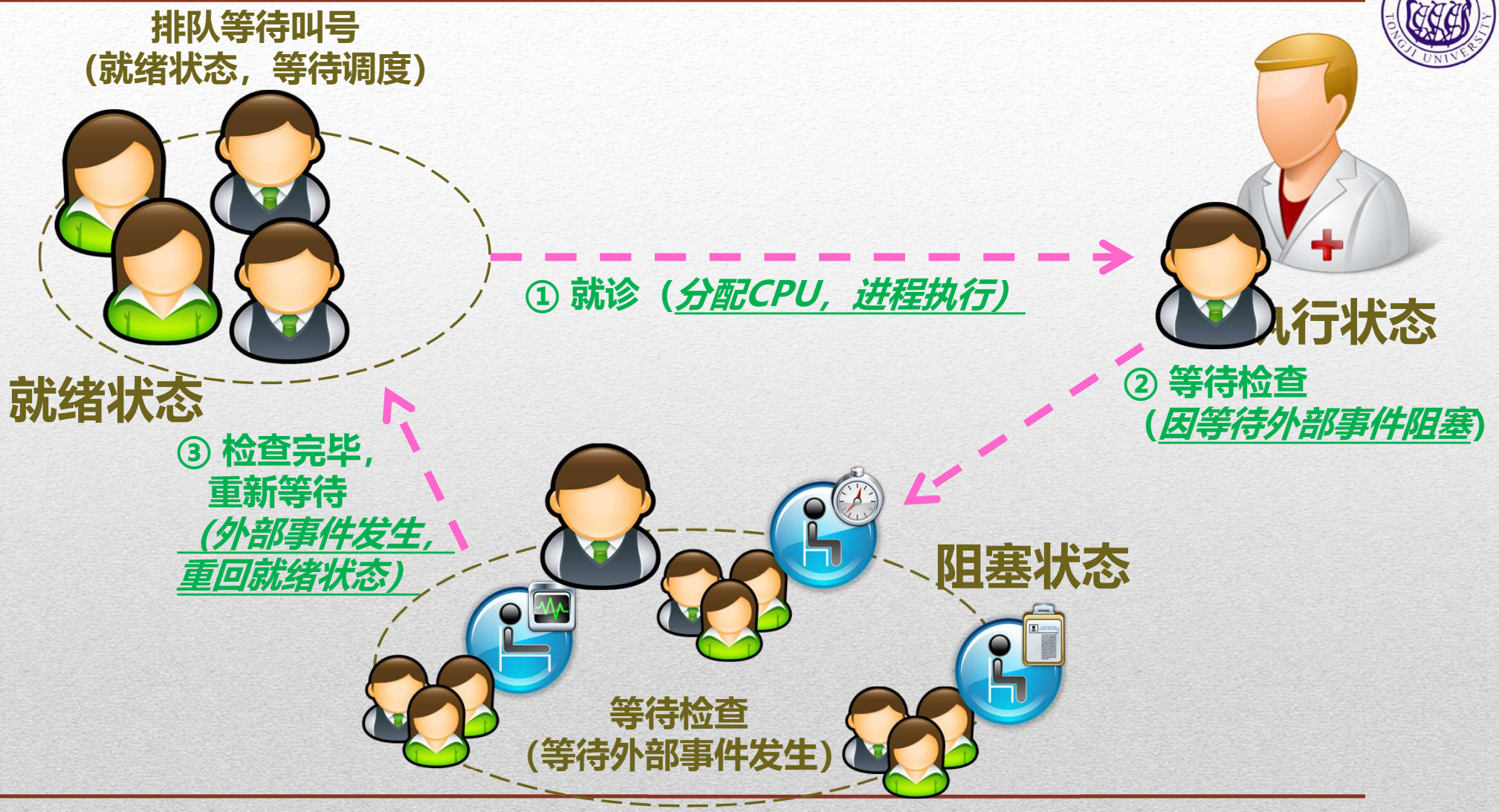

排队等待叫号
（就绪状态，等待调度）
就绪状态
① 就诊（分配CPU，进程执行）
执行状态
阻塞状态
等待检查
（等待外部事件发生）

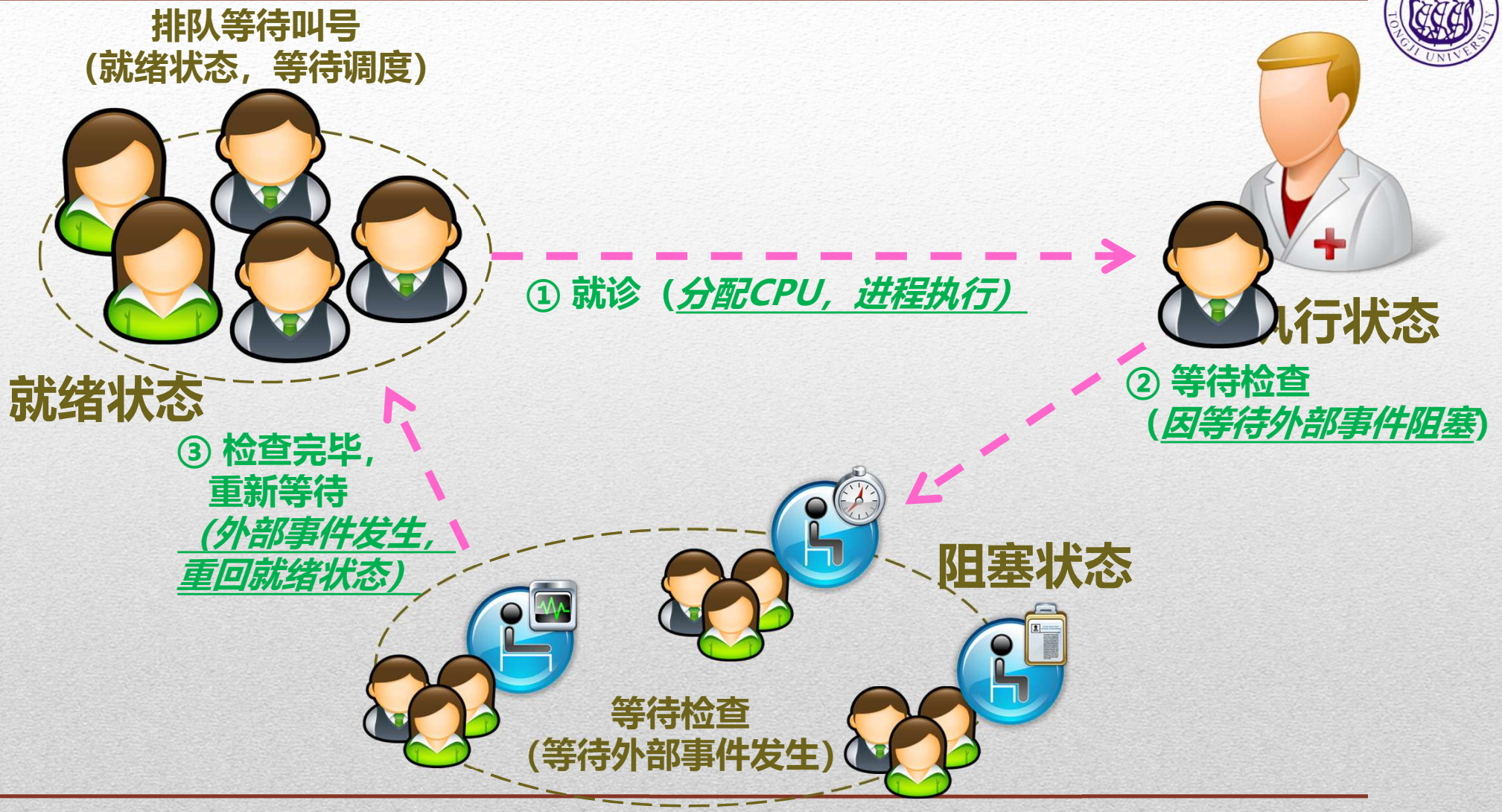

排队等待叫号
（就绪状态，等待调度）
就绪状态
① 就诊（分配CPU，进程执行）
执行状态
② 等待检查
（因等待外部事件阻塞）
阻塞状态
等待检查
（等待外部事件发生）

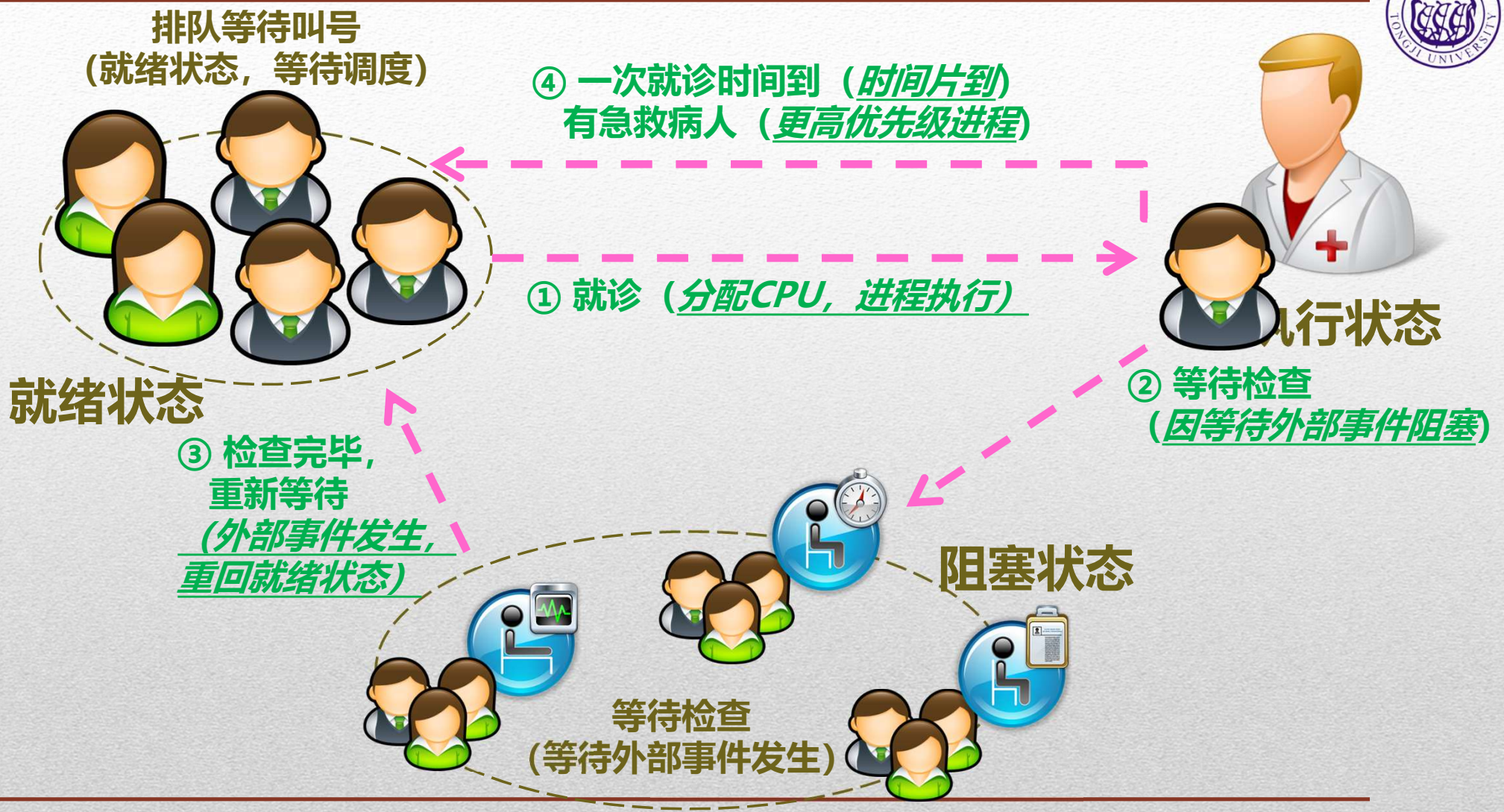

排队等待叫号
（就绪状态，等待调度）
就绪状态
① 就诊（分配CPU，进程执行）
执行状态
② 等待检查
（因等待外部事件阻塞）
阻塞状态
等待检查
（等待外部事件发生）


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
② 等待检查
（因等待外部事件阻塞）
阻塞状态
等待检查
（等待外部事件发生）

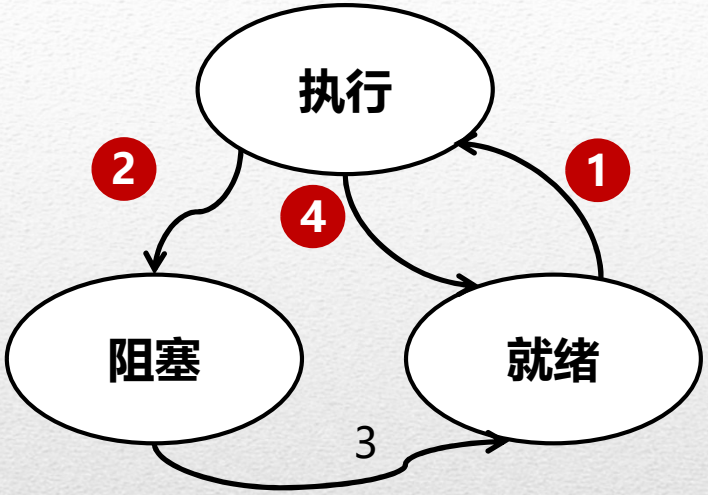

排队等待叫号
（就绪状态，等待调度）
就绪状态
① 就诊（分配CPU，进程执行）
执行状态
② 等待检查
（因等待外部事件阻塞）
阻塞状态
等待检查
（等待外部事件发生）


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
② 等待检查
（因等待外部事件阻塞）
③ 检查完毕，
重新等待
（外部事件发生，
重回就绪状态）
阻塞状态
等待检查
（等待外部事件发生）


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
② 等待检查
（因等待外部事件阻塞）
③ 检查完毕，
重新等待
（外部事件发生，
重回就绪状态）
阻塞状态
等待检查
（等待外部事件发生）


排队等待叫号
（就绪状态，等待调度）
④ 一次就诊时间到（时间片到）
有急救病人（更高优先级进程）
① 就诊（分配CPU，进程执行）
执行状态
② 等待检查
（因等待外部事件阻塞）
就绪状态
③ 检查完毕，
重新等待
（外部事件发生，
重回就绪状态）
阻塞状态
等待检查
（等待外部事件发生）

进程调度状态的控制



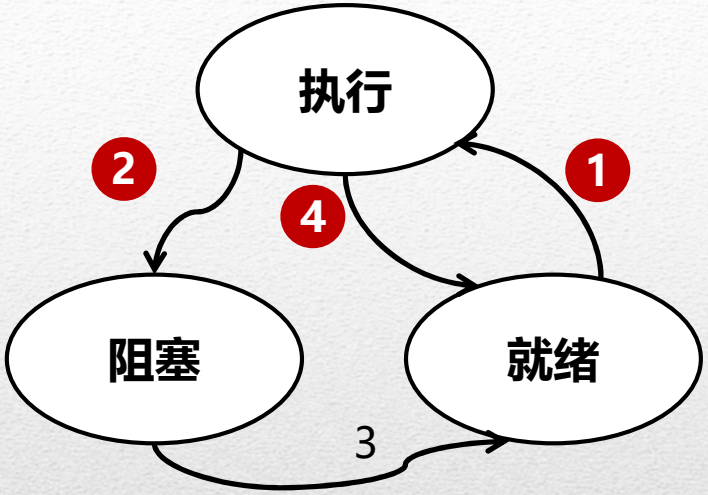
进程“下台”/“上台”

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

调度算法的选择需要考虑的因素

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

指标

作业完成时间 – 作业提交时间，或：
作业运行时间 + 作业等待时间

1. 对批处理型作业：周转时间短
2. 对交互型作业：响应速度快

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”

实时性高
但开销大

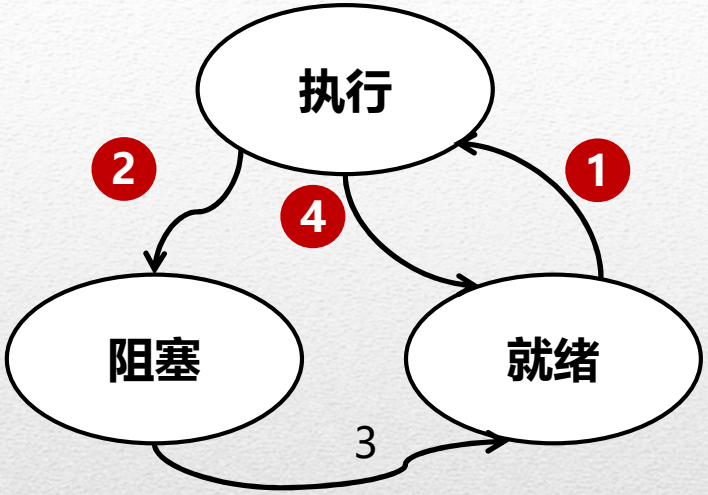
非抢占式/进程主动放弃

现运行进程暂停，PCB中的调度状态

2 “执行” → “阻塞”

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

经典的进程调度算法

先来先服务算法（FIFO）

从所有就绪进程中选择**一个最先就绪**的进程，为之分配处理机。该进程**一直运行到完成或发生某事件而阻塞才放弃处理机**。

非抢占

优点：

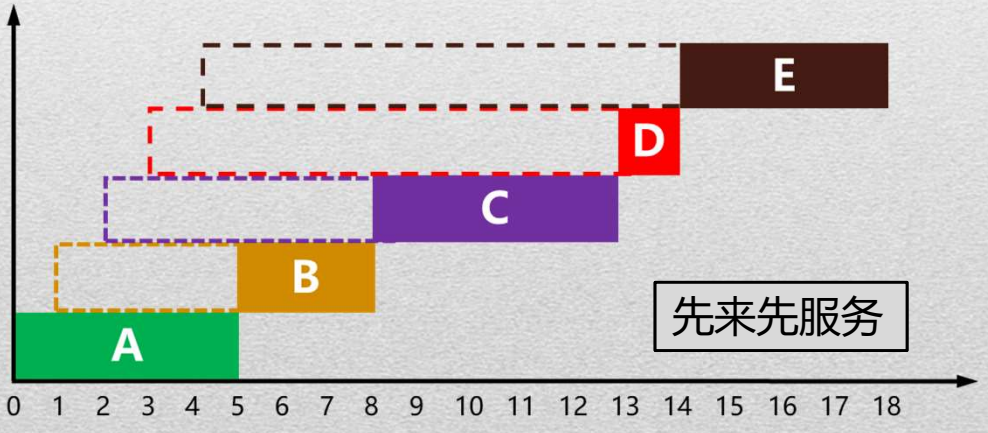
非抢占式，实现简单。

缺点：

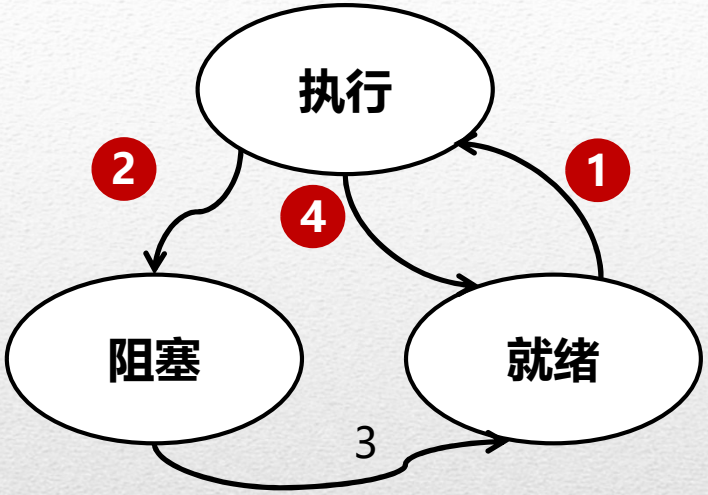
1. 长进程有利，短进程吃亏；
2. CPU密集型进程有利，I/O密集型进程吃亏。

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



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

经典的进程调度算法

短进程优先

从所有就绪进程中选择**一个估计运行时间最短**的进程，为之分配处理机。该进程**一直运行到完成或发生某事件而阻塞才放弃处理机**。

非抢占

优点：

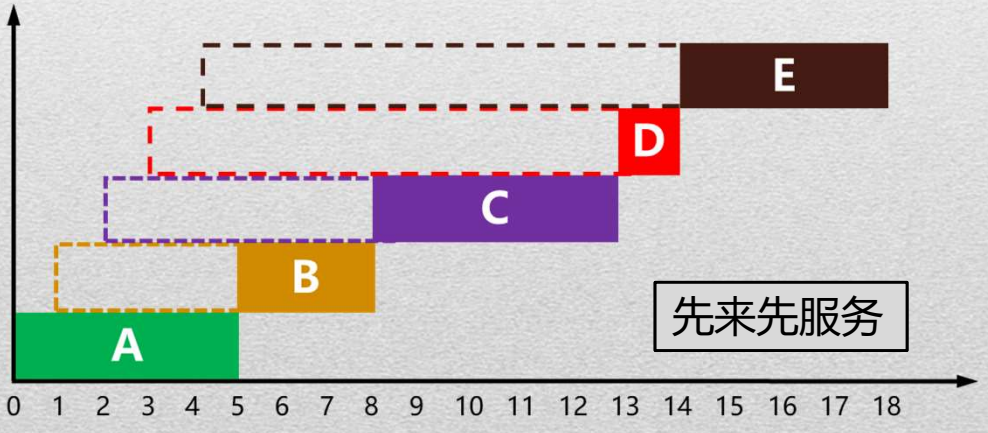
当所有进程同时到达时，平均周转时间最短。

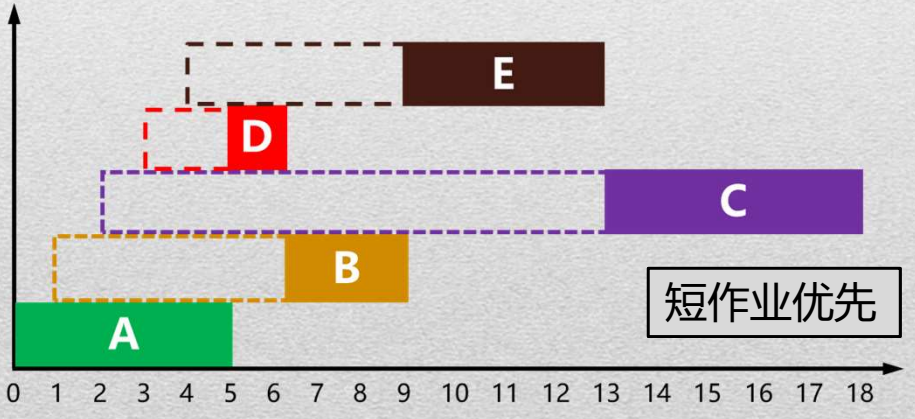
缺点：

1. 短进程有利，长进程吃亏；
2. 未考虑进程的紧迫程度。

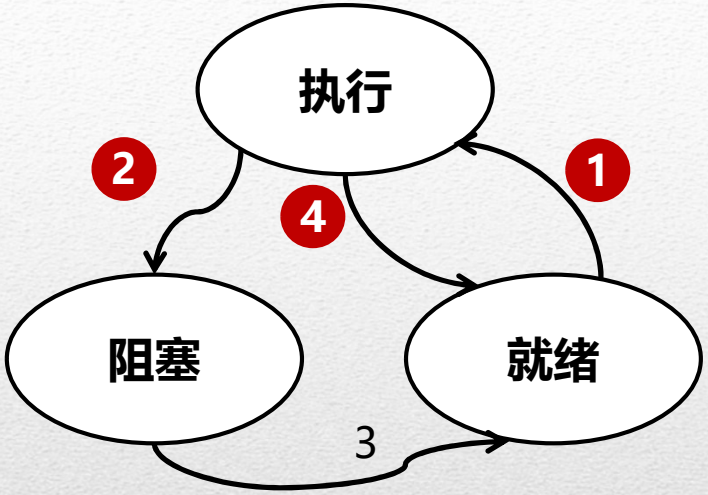
进程调度算法周转时间比较

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	3	9	8.2





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

经典的进程调度算法

短进程优先

从所有就绪进程中选择**一个估计运行时间最短**的进程，为之分配处理机。该进程**一直运行到完成或发生某事件而阻塞才放弃处理机**。

非抢占

优点：

当所有进程同时到达时，平均

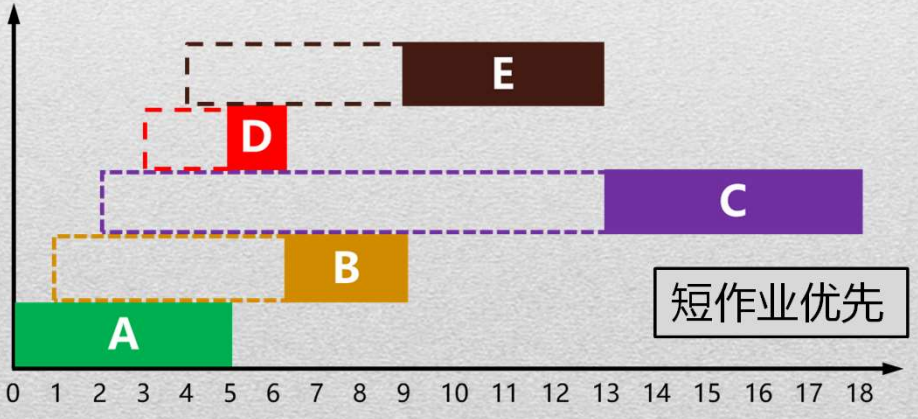
缺点：

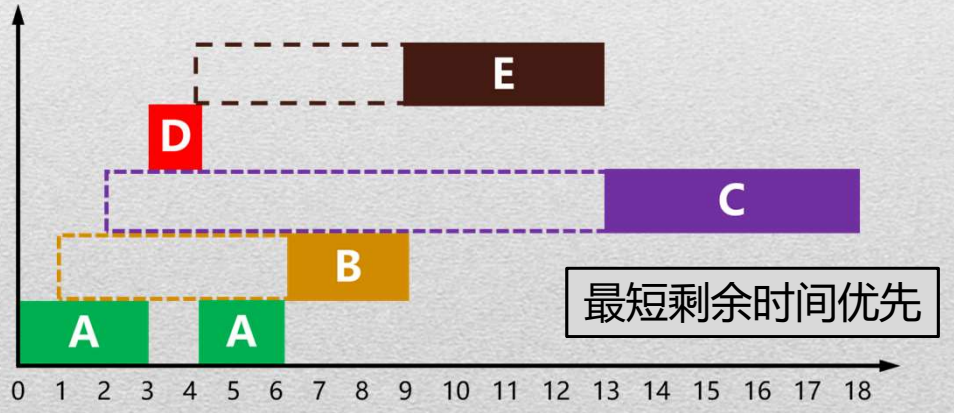
1. 短进程有利，长进程吃亏；
2. 未考虑进程的紧迫程度。

抢占式改进：
最短剩余时间优先

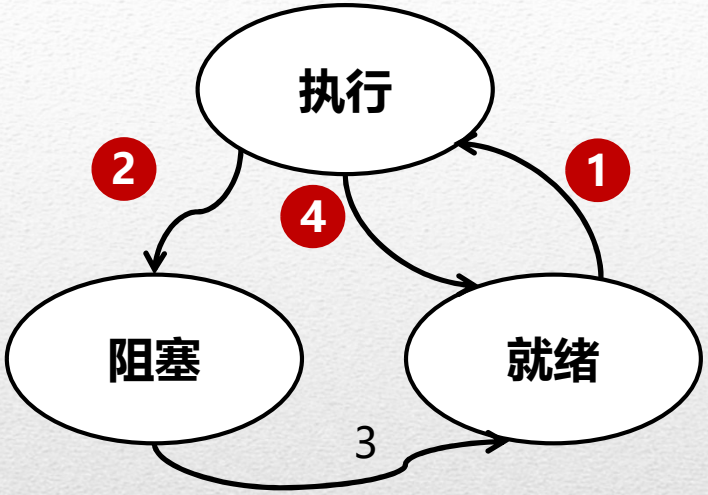
进程调度算法周转时间比较

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	3	9	8.2
最短剩余时间优先	6	8	16	1	9	8





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

经典的进程调度算法

时间片轮转调度算法

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。

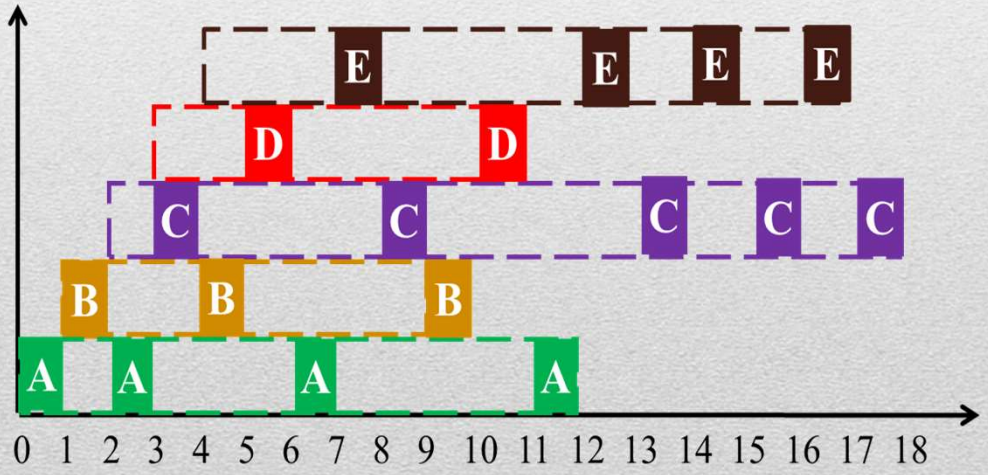
抢占

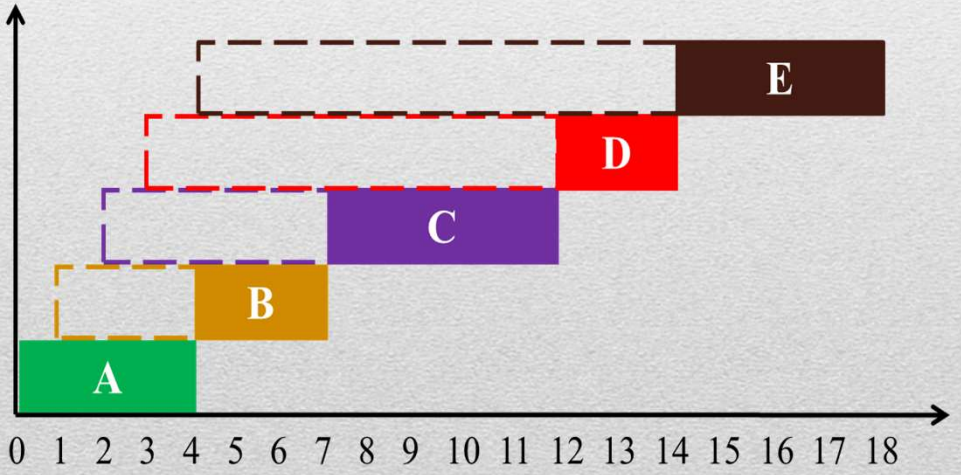
优点：各进程能够比较均衡地共享使用处理机

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。

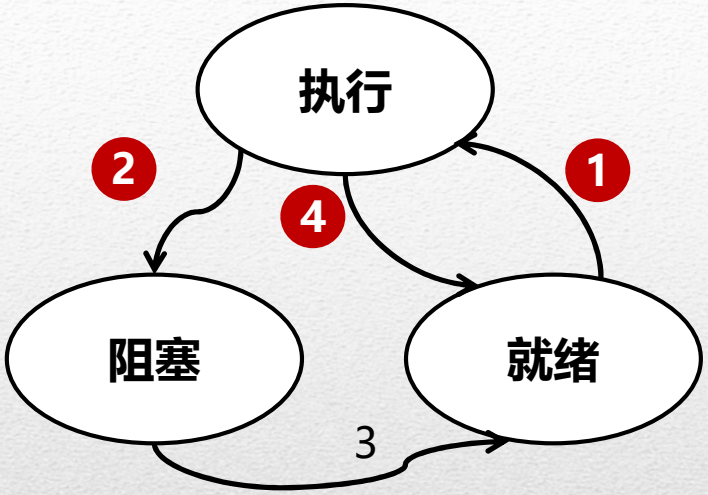
时间片选取对调度性能的影响

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





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

经典的进程调度算法

最高优先权优先调度算法

抢占或非抢占

静态优先权

创建进程时确定的，且在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，低者可能会“饥饿”。

动态优先权

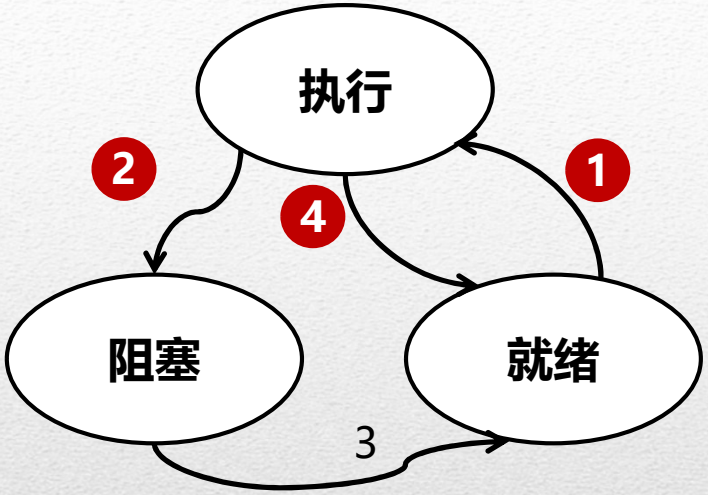
系统为刚生成的进程赋初始优先权，之后根据进程的行为动态调整优先权的值

抢占

算法的关键

优先权怎么调整？何时调整？

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”**

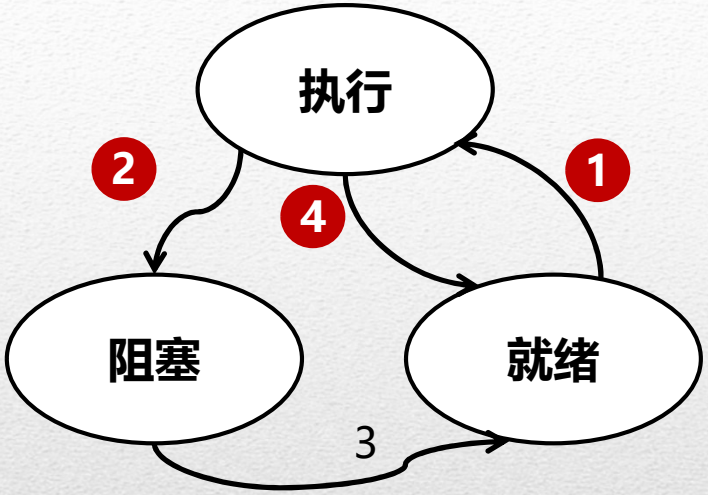
非抢占式/进程主动放弃

现运行进程暂停，PCB中的调度状态

2 **“执行”→“阻塞”**

确定了调度方式和调度算法之后，需要实现具体的调度过程。

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

然后，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. 在就绪队列中选择另一个就绪进程
3. 其状态“就绪”→“执行” 1
4. 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”

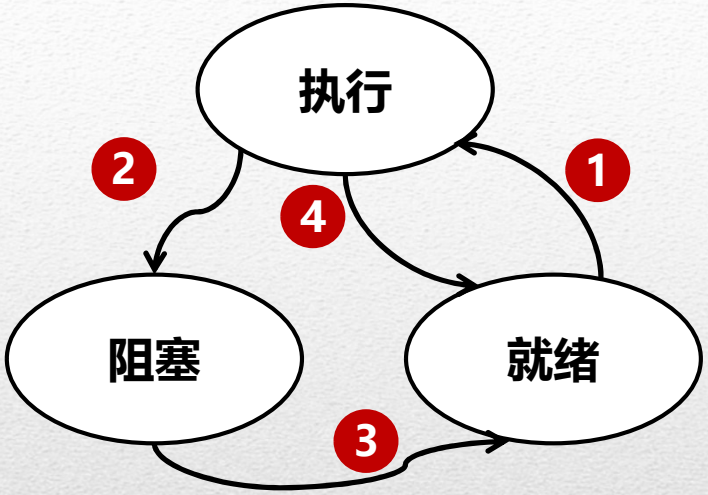
非抢占式/进程主动放弃

现运行进程暂停，PCB中的调度状态

2 “执行”→“阻塞”

确定了调度方式和调度算法之后，需要实现具体的调度过程。

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）：

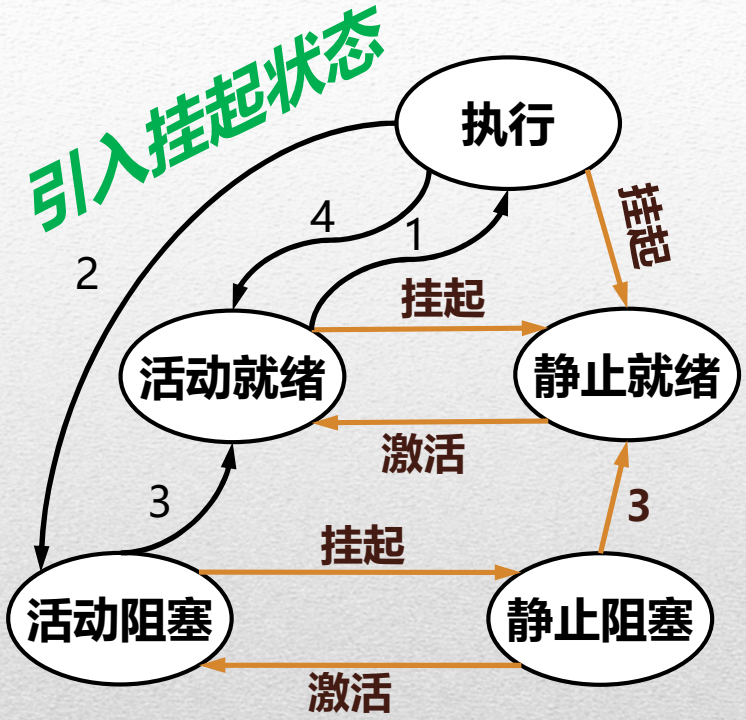
1. 立即停止执行
2. PCB中的进程状态“执行”→“阻塞” ②
3. PCB进入阻塞队列
4. 系统调度程序完成进程切换调度 ①

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
3. 由调度算法决定是否切换调度

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**：

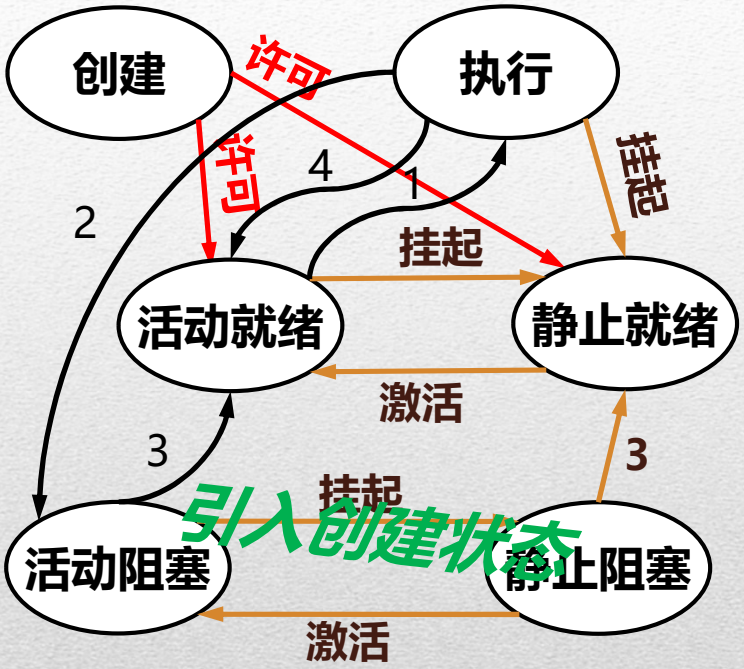
1. 若为当前执行进程：立即停止执行，PCB中的进程状态“执行”→“静止就绪”，调度程序进行**切换调度**
2. 若非当前执行进程：　PCB中的进程状态“活动就绪”→“静止就绪”/“活动阻塞”　→“静止阻塞”

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：

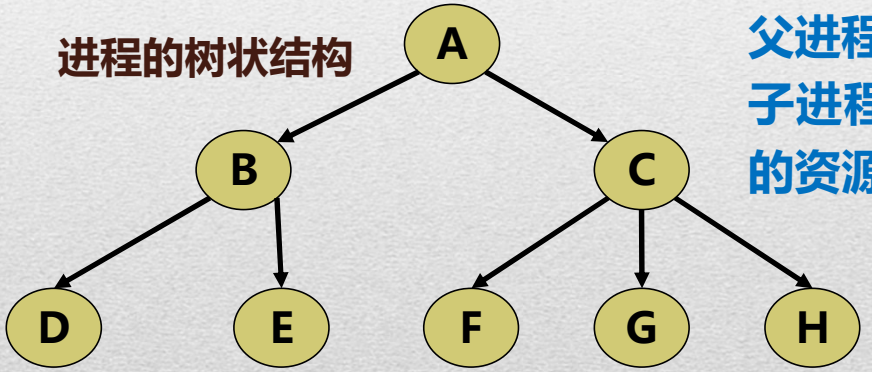
用户登录　　作业调度

提供服务　　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



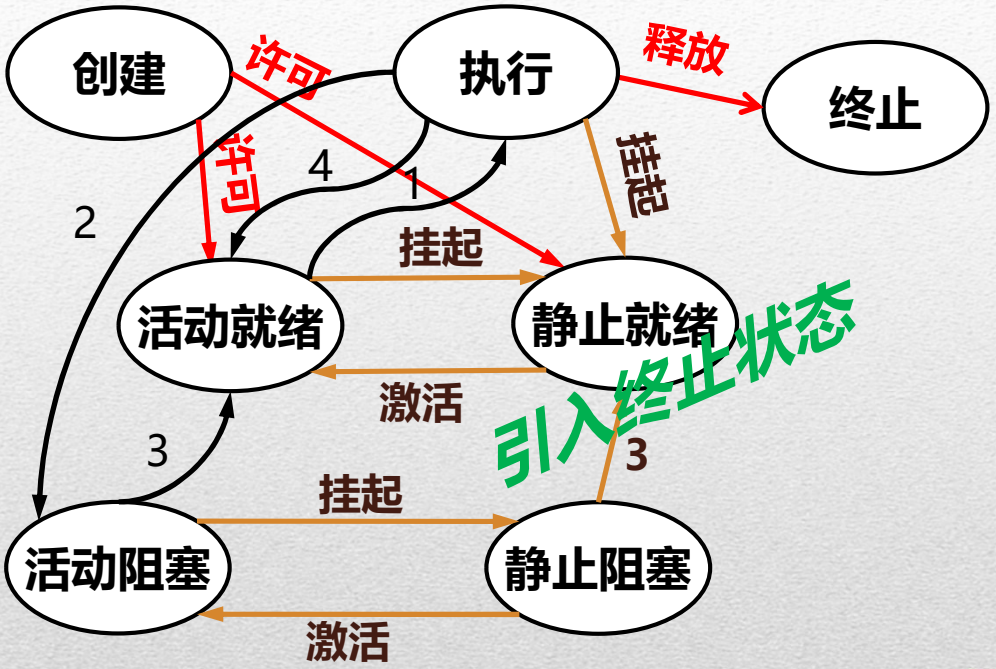
父进程创建子进程

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

PCB是进程存在的唯一标志

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：


程序


进程

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”

进程是程序的一次运行过程!!!

- 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

Very Important!

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

3 经典的进程调度算法

E01：并发进程（进程基本概念）