操 作 系 统（0601009）

河海大学“课程思政”示范课程

授课教师：	张鹏程、陆佳民
Email：	pchzhang@hhu.edu.cn
QQ：	185319755、1284762490
办公室：	勤学楼4515、4121

回顾

- 1、每个线程具有哪些内容？
- 2、在多线程结构中进程和线程分别承担什么？
- 3、线程属性？
- 4、进程和线程分别封装了什么？
- 5、线程的组织方式有哪三种？具体阐述
- 6、线程的实现有哪两种？
- 7、内核级线程？优缺点
- 8、用户级线程？优缺点
- 9、混合式线程？

2.6 处理器调度

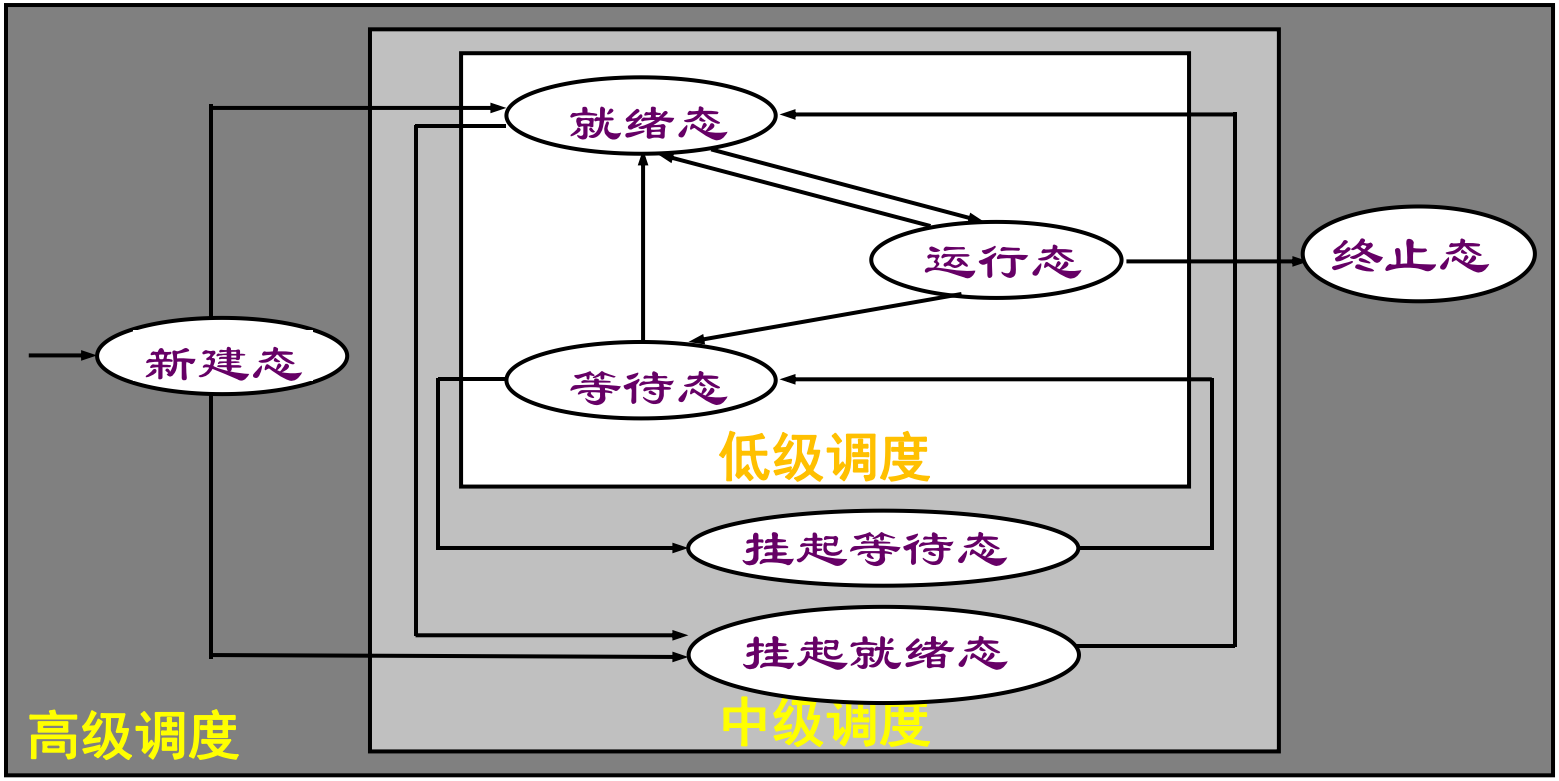
- 处理器调度的层次
- 选择调度算法的原则
- 作业的管理与调度
- 低级调度功能和类型
- 作业调度和低级调度算法

2.6 处理器调度

- 处理器调度的层次
- 选择调度算法的原则
- 作业的管理与调度
- 低级调度功能和类型
- 作业调度和低级调度算法

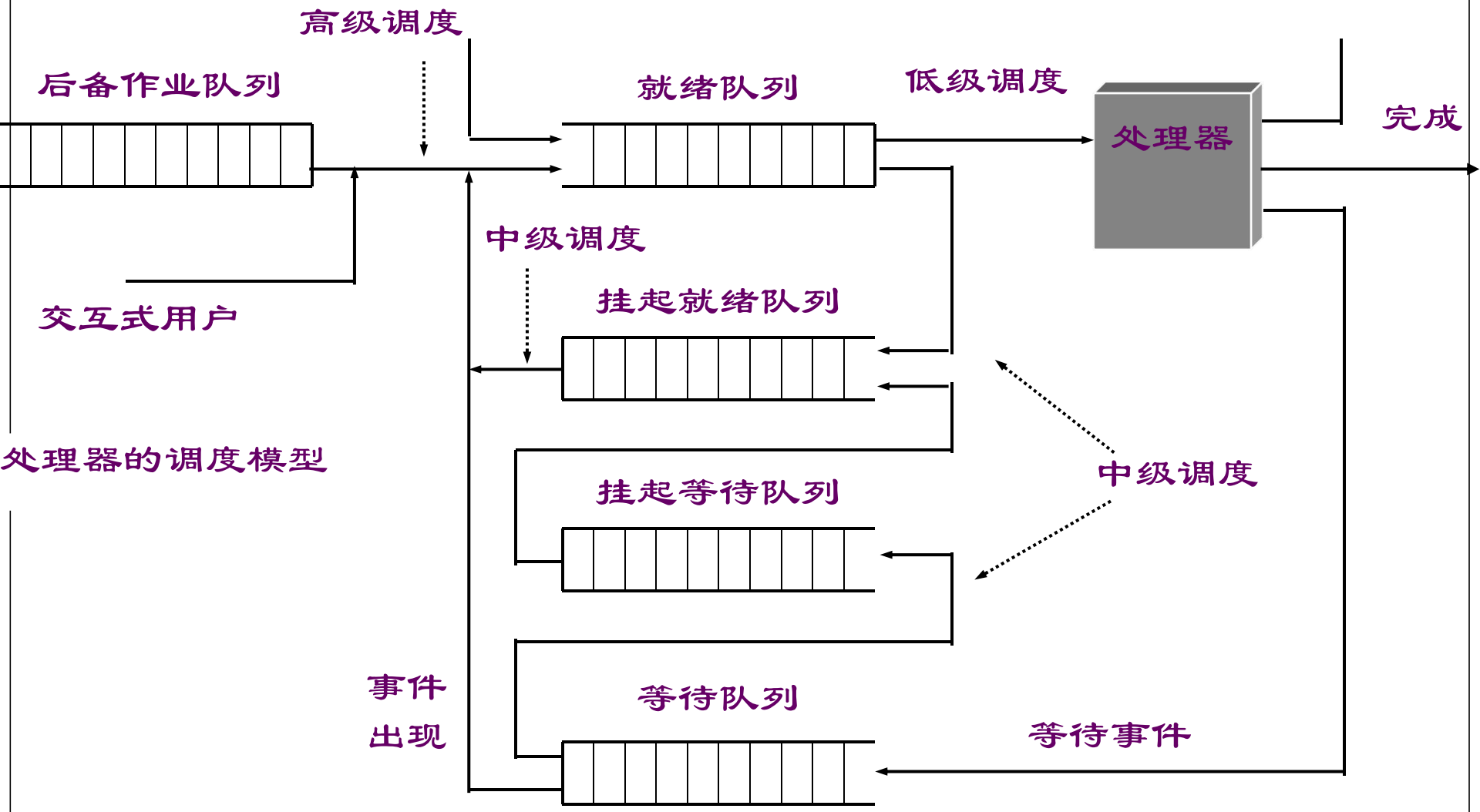
2.6.1 处理器调度的层次

- 用户作业生命周期：

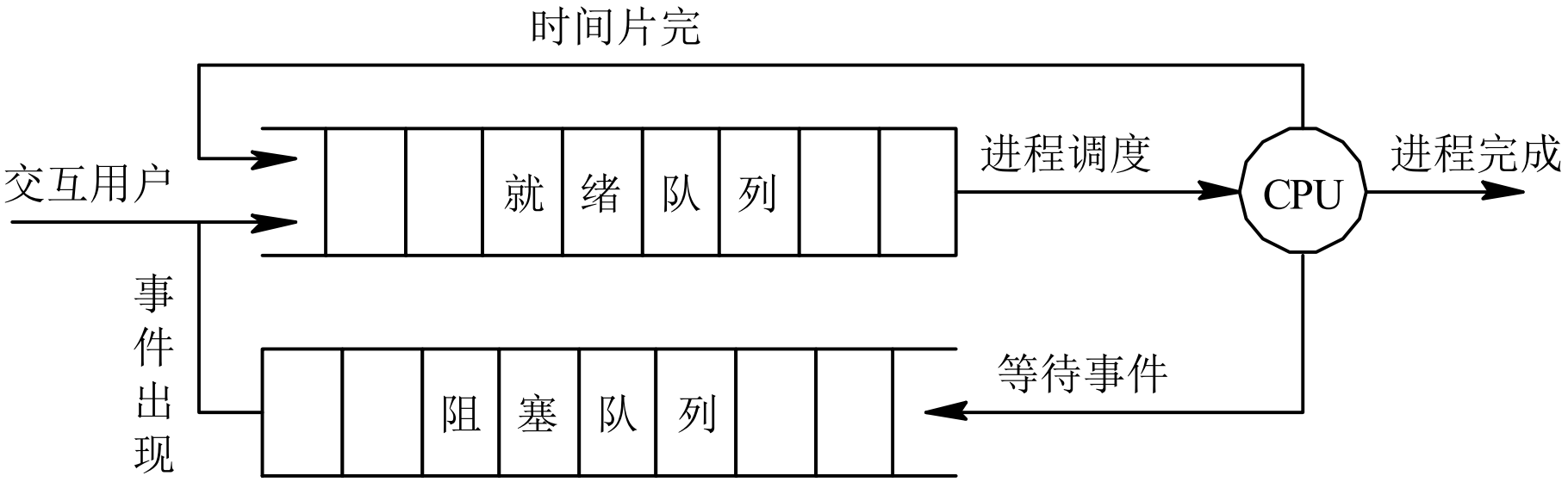


2.6.1 处理器调度的层次

- 处理器调度可分为三个级别：
  - 高级调度（作业调度、长程调度）
  - 中级调度（内存调度）
  - 低级调度
- 低级调度是各类操作系统中必须具有的功能
- 在纯粹的分时或实时操作系统，作业被直接调入内存，因此通常不需要高级调度
- 在分时系统或具有虚拟存储器的操作系统中，专门引进了中级调度，控制进程在内存和外存间的对换


高级调度
后备作业队列
就绪队列
低级调度
处理器
完成
中级调度
交互式用户
挂起就绪队列
处理器的调度模型
中级调度
挂起等待队列
事件
出现
等待队列
等待事件

三级调度模型



两级调度模型

2.6.1：总结

- 高级调度发生在新进程的创建中，它决定一个进程能否被创建，或者创建后能否被置成就绪状态
- 中级调度反映到进程状态上就是挂起和解除挂起，它根据系统的当前负荷情况决定停留在主存中进程数
- 低级调度决定哪一个就绪进程占有CPU

2.6 处理器调度

- 处理器调度的层次
- 选择调度算法的原则
- 作业的管理与调度
- 低级调度功能和类型
- 作业调度和低级调度算法

2.6.2 选择调度算法的原则

- 任何层次的处理器调度，都由操作系统的调度程序实施，调度程序（scheduler）所使用的算法称为调度算法
- “效益优先，兼顾公平原则”
- 调度算法设计通常应考虑如下原则/目标：
  1. 资源利用率
  2. 吞吐率
  3. 公平性
  4. 响应时间
  5. 周转时间
  6. 截至时间的保证
  7. 优先权原则

2.6.2：原则（1）：资源利用率

CPU利用率 =

CPU有效工作时间/CPU总的运行时间

CPU总的运行时间 =

CPU有效工作时间+CPU空闲等待时间

2.6.2：原则（1）附

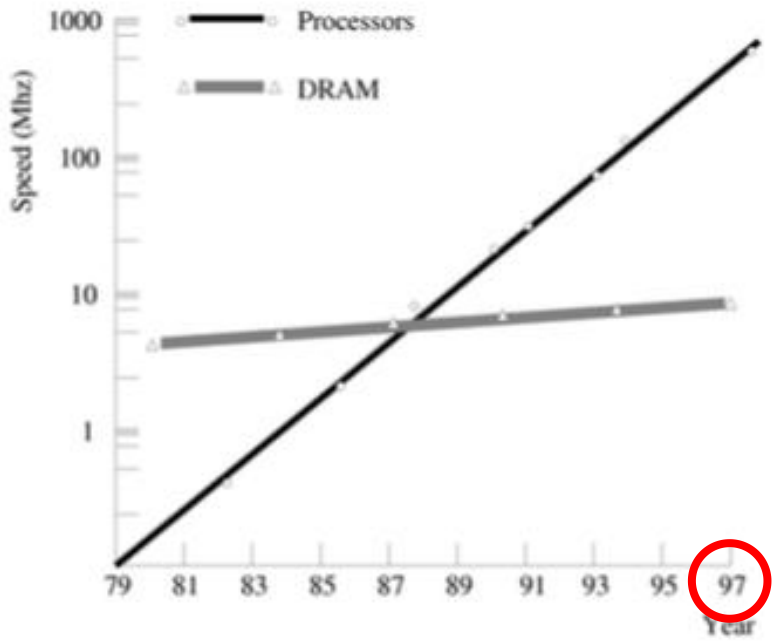


Fig. 1. Trends in DRAM and CPU speed.

Intel i7-4770K: 3.5 GHz

DDR3: 1600 MHz

[1] S. Manegold, P. Boncz, and M. Kersten, “Optimizing main-memory join on modern hardware,” *TKDE*, 2002.

2.6.2：原则（2）：吞吐率

- 单位时间内CPU处理的作业数
  - 长作业多：吞吐率低
  - 短作业多：吞吐率高

2.6.2：原则（3）：公平性

- 确保每个用户每个进程获得合理的CPU份额或其他资源份额，不会出现饿死情况

2.6.2：原则（4）：响应时间

- 交互式进程从提交一个请求(命令)到接收到响应之间的时间间隔称响应时间
- 包括三部分时间：
  - 请求信息传到处理机的时间
  - 处理机处理请求信息的时间
  - 形成的响应回送到终端的时间

使响应时间尽可能的短，是分时系统衡量调度性能的一个重要指标

2.6.2：原则（5）：作业周转时间

- 作业周转时间：批处理用户从作业提交给系统开始，到作业完成为止的时间间隔
- 如果作业 i 提交给系统的时刻是 t_s，完成时刻是 t_f，该作业的周转时间 t_i 为：

$$t_i = t_f - t_s$$

- 实际上，它是作业在系统里的等待时间与运行时间之和

使作业周转时间或平均作业周转时间尽可能短，是批处理系统衡量调度性能的一个重要指标

2.6.2：原则（5）续

- 周转时间通常包括四部分时间：
  - 作业在外存后备队列上等待调度的时间
  - 进程在就绪队列上等待进程调度的时间
  - 进程在处理器上执行的时间
  - 进程等待I/O操作完成的时间
- 平均作业周转时间：

$$T = \sum_{i=1}^{n} t_i / n$$

衡量不同调度算法对相同作业流的调度性能

2.6.2：原则（5）：带权周转时间

- 带权周转时间：如果作业 i 的周转时间为 t_i，所需运行时间为 t_k，则该作业的带权周转时间为：

$$w_i = t_i / t_k$$

- t_i 是等待时间与运行时间之和，故带权周转时间总大于1
- 平均作业带权周转时间：

$$W = \sum_{i=1}^{n} w_i / n$$

衡量相同调度算法对不同作业流的调度性能

2.6.2：原则（6）：截止时间保证

- 截止时间：指某个任务必须开始执行的最迟时间或必须完成的最迟时间
- 是评价实时系统调度性能的一个重要指标

2.6.2：原则（7）：优先权

- 实时、分时、批处理均可遵循的原则，让一些紧急的任务尽快被执行

2.6 处理器调度

- 处理器调度的层次
- 选择调度算法的原则
- 作业的管理与调度
- 低级调度功能和类型
- 作业调度和低级调度算法

2.6.3 作业的管理与调度

1、作业和进程的关系

作业是用户提交给操作系统计算的一个独立任务。每个作业必须经过若干相对独立，且相互关联的顺序加工步骤才能得到结果。其中每个加工步骤称为一个作业步。

进程是已提交完毕并选中运行的作业的执行实体，也是为完成作业任务向系统申请和分配资源的基本单位，它处于运行、就绪、等待等多个状态的变化之中，在CPU上推进。最终完成应用程序任务。

2.6.3 作业的管理与调度

1、作业和进程的关系

作业是任务实体，进程是完成任务的执行实体。没有作业任务，进程无事可做，没有进程，作业任务无法完成。作业的概念更多的用于批处理操作系统中啊，而进程则用于各种多道程序设计系统。

2.6.3 作业的管理与调度

- 批处理作业
- 交互式作业

2.6.3：批处理作业

- 批处理作业的输入
- 批处理作业的建立
- 批处理作业的调度
  - 选择作业
  - 分配资源
  - 创建进程
  - 作业控制
  - 后续处理

2.6.3：交互式作业

- 交互作业的组织、提交和控制与批处理作业的差别：
  - 生命周期，由用户决定
  - 作业情况和资源需求通过具体命令来提交和控制
  - 输入一条/一组命令，则创建一个/若干进程来完成

2.6.3：交互式作业（续）

- 具体的键盘命令：
  - 作业控制类
  - 资源申请类
  - 文件操作类
  - 目录操作类
  - 设备控制类

2.6 处理器调度

- 处理器调度的层次
- 选择调度算法的原则
- 作业的管理与调度
- 低级调度功能和类型
- 作业调度和低级调度算法

2.6.4 低级调度功能

- 调度程序担负两项任务
  - 调度：确定就绪进程/线程使用处理器的次序
  - 分派：确定如何时分复用CPU

2.6.4：低级调度的引发事件

- 创建新进程/线程
- 进程/线程终止
- 进程/线程阻塞自己
- 进程/线程运行过程中发生中断或异常
- 进程/线程执行系统调用
- 进程/线程请求的I/O操作完成
- 进程/线程耗尽时间片
- 进程/线程改变优先级

附：CPU-I/O Burst Cycle

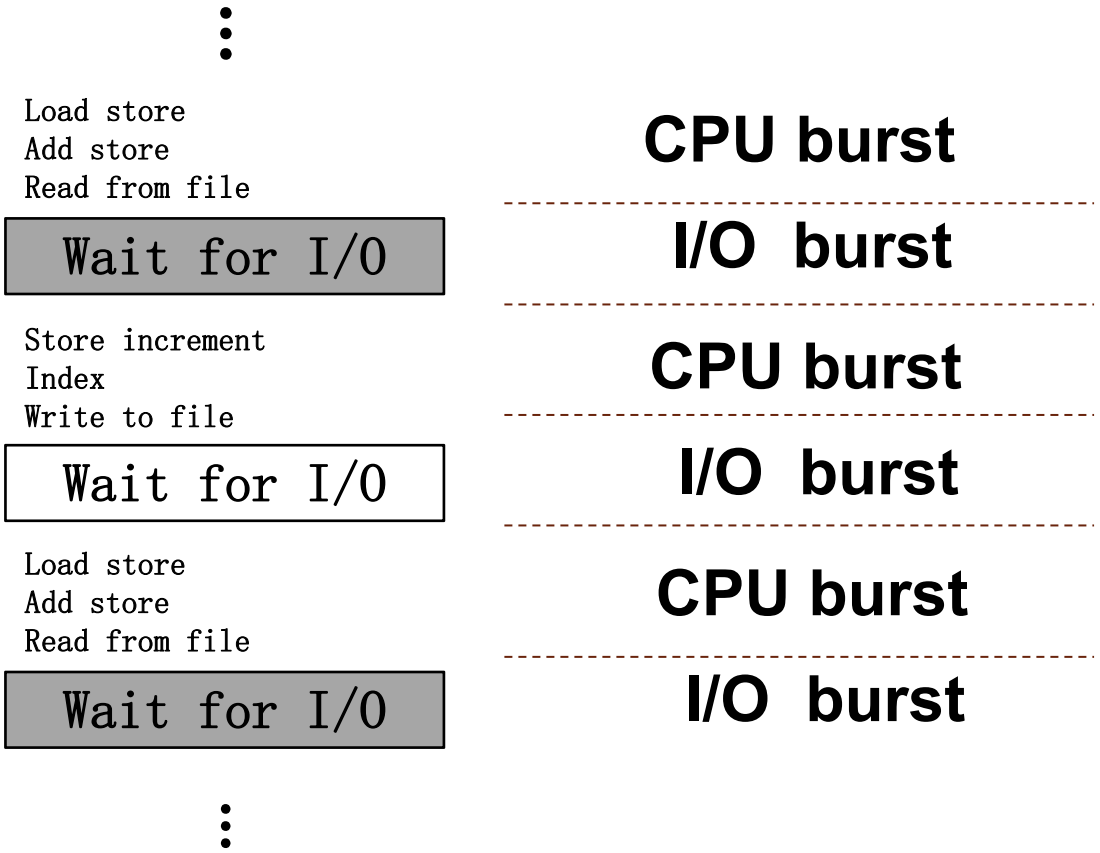


Figure 5.1
Alternating sequence of CPU and I/O bursts
"OSC" P154

2.6.4：低级调度功能组成

1. 队列管理程序
2. 分派程序（dispatcher）
    1. 当前进程/线程出让资源
    2. 装入分派程序（1st）
    3. 新进程的上下文切换（2rd）
3. 上下文切换程序(context switcher)

发生两次上下文切换

利用多组上下文寄存器，加快切换效率

2.6.4：低级调度类型

- 剥夺方式（preemptive），抢占式：
  - 当一个进程正在处理器上执行时，系统可以根据规定的原则剥夺分配给它的处理器，而把处理器分配给其他进程使用
  - 有两种常见的剥夺原则：高优先级剥夺原则和时间片剥夺原则，前者高优先级进程或线程可以剥夺低优先级进程或线程运行，后者当运行进程时间用完后被剥夺处理器

2.6.4：低级调度类型

- 非剥夺方式（nonpreemptive），非抢占式：
  - 一旦某个进程或线程开始执行后便不再出让处理器，除非该进程或线程运行结束或发生了某个事件不能继续执行

抢占式调度 高于 非抢占式调度 的开销
- 调度程序自身运行开销
- 内外存程序、数据的对换开销

提高用户体验：
- 内核关键进程是非抢占的
- 用户进程是抢占的

回顾

- 1、处理器有哪三个层次的调度？
- 2、什么叫高级调度？
- 3、什么叫中级调度？
- 4、什么叫低级调度？
- 5、处理器调度算法的选择原则？分别定义
- 6、作业和进程的关系？
- 7、批处理作业的管理和调度步骤？
- 8、交互式作业的管理和调度步骤？
- 9、调度程序担负两项任务？
- 10、低级调度的类型？

2.6 处理器调度

- 处理器调度的层次
- 选择调度算法的原则
- 作业的管理与调度
- 低级调度功能和类型
- 作业调度和低级调度算法

2.6.5 作业调度和低级调度算法

- 作业调度算法
- 低级调度算法：面向进程 /线程

- 某些算法可以通用
- 两级调度也可以采用不同的调度算法

2.6.5 作业调度和低级调度算法

1. FCFS：先来先服务算法
2. SJF：最短作业优先算法
3. SRTF：最短剩余时间优先算法
4. HRRF：响应比高者优先算法
5. PS：优先级调度算法(Priority Scheduling)
6. RR：轮转调度算法
7. MLFQ：多级反馈队列调度算法
8. LS：彩票调度算法(Lottery Scheduling)

2.6.5/1： FCFS(First Come First Served)

- 策略：按照作业进入系统的先后次序来挑选作业，先进入系统的作业优先被挑选。 这是一种非剥夺式算法
- 效果：算法容易实现，效率不高，只顾及作业等候时间，没考虑作业要求服务时间的长短，不利于短作业而优待了长作业。有利于CPU繁忙型作业而不利于I/O繁忙型作业

2.6.5/1: FCFS Example

作业	所需CPU时间	运行时间
1	28	28
2	9	37
3	3	40

平均作业周转时间 $T = (28 + 37 + 40)/3 = 35\ ms$

改变作业顺序会怎样？

2.6.5/1: FCFS Example

作业	所需CPU时间	运行时间
2	9	9
1	28	37
3	3	40

平均作业周转时间 $T = (9 + 37 + 40)/3 = 29\ ms$

2.6.5/1: FCFS Example

作业	所需CPU时间	运行时间
3	3	3
2	9	12
1	28	40

平均作业周转时间 $T = (3 + 12 + 40)/3 = 18\ ms$

FCFS调度算法的平均作业周转时间与作业提交的顺序有关

2.6.5/1: FCFS Example

平均带权作业周转时间

1，2，3：$W = (28/28 + 37/9 + 40/3)/3 \approx 6$

2，1，3：$W = (9/9 + 37/28 + 40/3)/3 \approx 5.2$

3，2，1：$W = (3/3 + 12/9 + 40/28)/3 \approx 1.24$

平均带权作业周转时间 反映了不同作业流对调度算法的影响。

越接近1，则表示算法对当前作业流的响应及时。

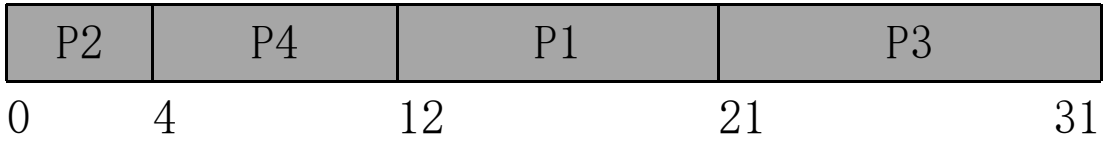
2.6.5/2: SJF (Shortest Job First)

- 策略：最短作业优先算法以进入系统的作业所要求的CPU时间为标准，总选取估计计算时间最短的作业投入运行。属于非抢占式算法
- 效果：算法易于实现，效率不高。
  - 需要预先知道作业所需的CPU时间。这个时间只能靠估计，估计值很难精确，若估计过低，系统可能提前终止该作业。
  - 忽视了作业等待时间，会出现饥饿现象
  - 由于缺少剥夺机制，对分时、实时处理很不理想。

2.6.5/2: SJF Example

作业	所需CPU时间
1	9
2	4
3	10
4	8

假设，四个作业同时到达系统，并进行调度



SJF下，执行顺序为 2，4，1，3

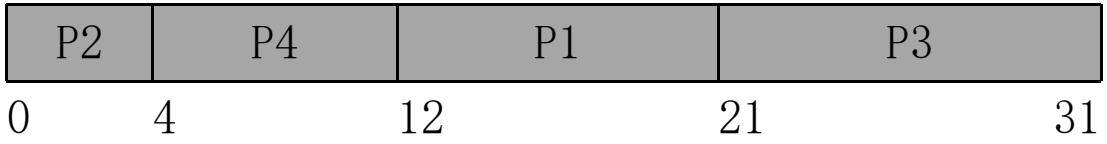
$$T = (4 + 12 + 21 + 31)/4 = 17$$

$$W = (4/4 + 12/8 + 21/9 + 13/10)/4 = 1.98$$

2.6.5/2: SJF Example

作业	所需CPU时间
1	9
2	4
3	10
4	8

假设，四个作业同时到达系统，并进行调度



SJF下，执行顺序为 2，4，1，3

$$T = (4 + 12 + 21 + 31)/4 = 17$$

$$W = (4/4 + 12/8 + 21/9 + 13/10)/4 = 1.98$$

FCFS 下?

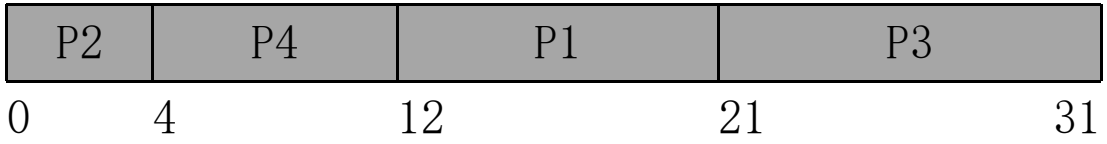
T = 19

W = 2.61

2.6.5/2: SJF Example

作业	所需CPU时间
1	9
2	4
3	10
4	8

假设，四个作业同时到达系统，并进行调度



SJF下，执行顺序为 2，4，1，3

$T = 17$

$W = 1.98$

- ✓ SJF的调度性优于FCFS
- ✓ 实现SJF需要知道作业所需运行时间

FCFS下?

$T = 19$

$W = 2.61$

2.6.5/2: SJF Time Estimation

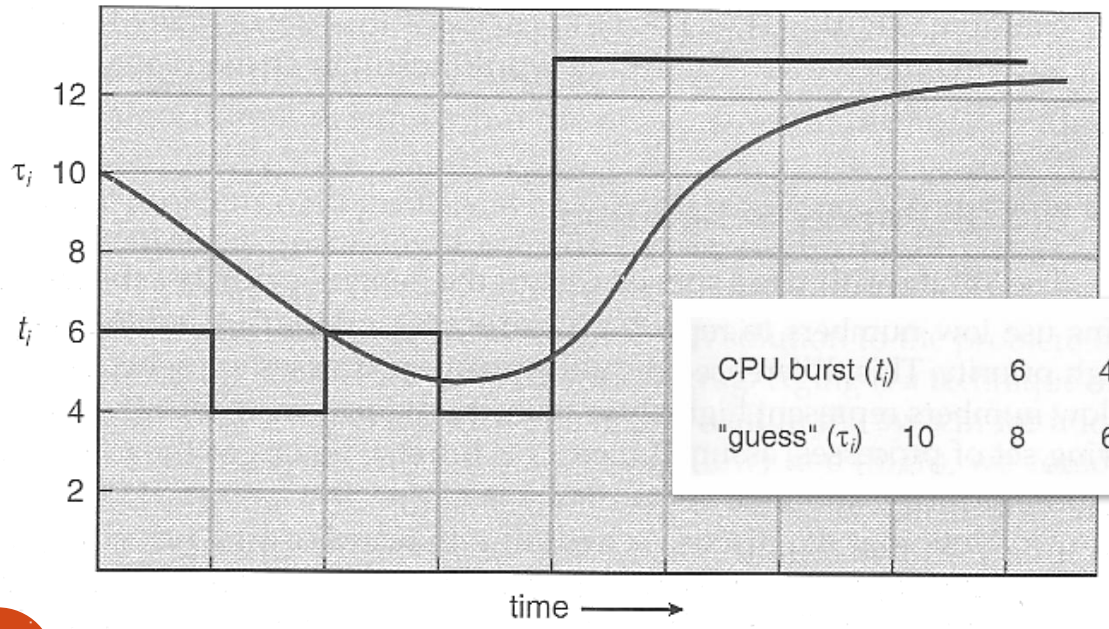
$$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$$

- ✓ 兼顾最近信息与历史信息
- ✓ 历史越久远，对估算值影响越小

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\alpha t_{n-1} + \ldots + (1-\alpha)^j \alpha t_{n-j} + \ldots + (1-\alpha)^{n+1} \alpha \tau_0$$

2.6.5/2: SJF Time Estimation

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\alpha t_{n-1} + \ldots + (1-\alpha)^j \alpha t_{n-j}$$
$$+ \ldots + (1-\alpha)^{n+1} \alpha \tau_0$$



CPU burst (t_i)	6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Figure 5.3
Prediction of the length of the next CPU burst
"OSC" P161

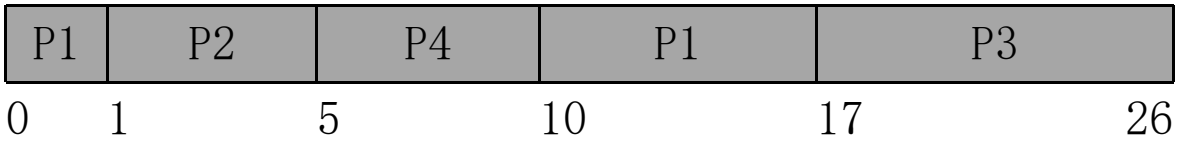
2.6.5/3： SRTF (Shortest Remaining Time First)

- 最短剩余时间优先算法是把SJF算法改为**抢占式**的调度算法。
- 策略：当一个作业正在执行时，一个新作业进入就绪状态，如果新作业需要的CPU时间比当前正在执行的作业剩余下来还需的CPU时间短，SRTF强行赶走当前正在执行作业。
- 效果：确保新的短作业/进程能够很快获得服务。
- 此算法不但适用于作业调度，同样也适用于进程调度 。

2.6.5/3: SRTF Example

作业	到达时间	所需CPU时间
1	0	8
2	1	4
3	2	9
4	3	5

SRTF:



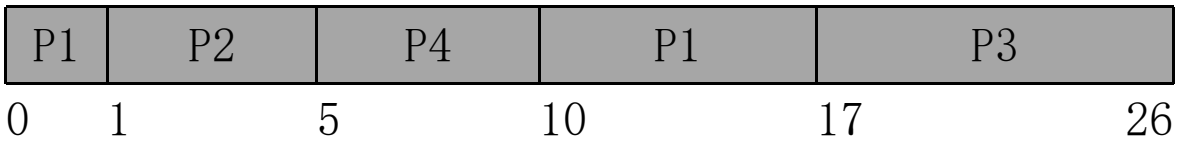
$$wa = ((10-1)+(1-1)+(17-2)+(5-3))/4 = 6.5$$

SJF:

2.6.5/3: SRTF Example

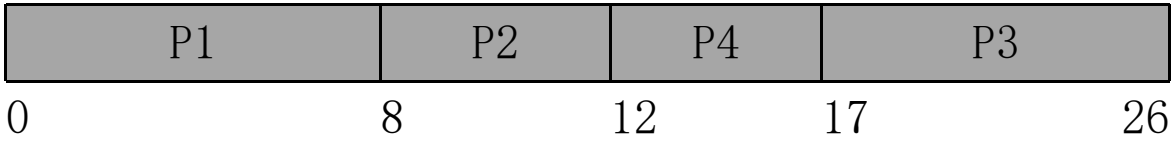
作业	到达时间	所需CPU时间
1	0	8
2	1	4
3	2	9
4	3	5

SRTF:



$$wa = ((10-1)+(1-1)+(17-2)+(5-3))/4 = 6.5$$

SJF:

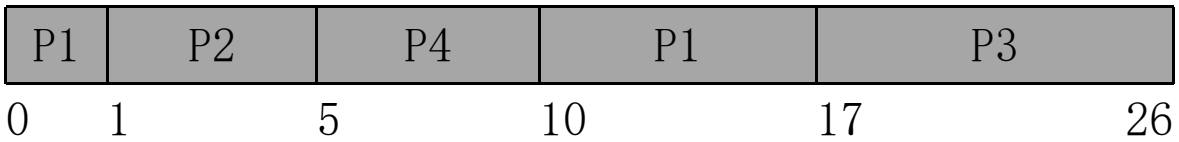


$$wa = ((0-0)+(8-1)+(12-3)+(17-2))/4 = 7.75$$

2.6.5/3: SRTF Example

作业	到达时间	所需CPU时间
1	0	8
2	1	4
3	2	9
4	3	5

SRTF:



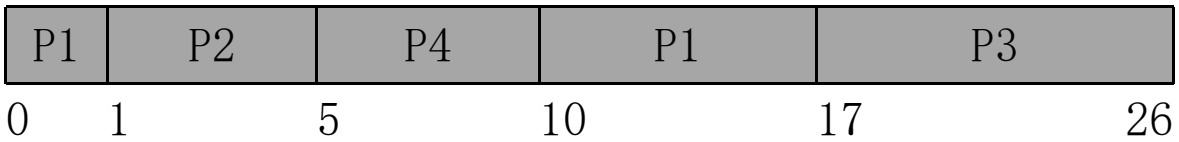
$$T = ((17-0)+(5-1)+(26-2)+(10-3))/4 = 13$$

SJF:

2.6.5/3: SRTF Example

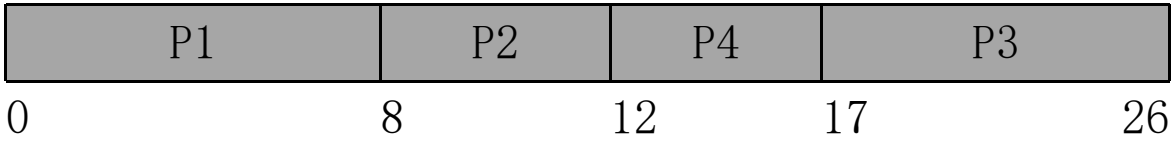
作业	到达时间	所需CPU时间
1	0	8
2	1	4
3	2	9
4	3	5

SRTF:



$$T = ((17-0)+(5-1)+(26-2)+(10-3))/4 = 13$$

SJF:



$$T = ((8-0)+(12-1)+(26-2)+(17-3))/4 = 14.25$$

2.6.5/3： SRTF (Shortest Remaining Time First)

- 采用SRTF后，等待和平均周转时间都有较为明显的降低，对短作业非常有利。但是SRTF也存在不容忽视的缺点：
  - 该算法对长作业不利，可能导致长作业长期饥饿
  - 该算法考虑作业的紧迫程度，但不能保证紧迫作业会被及时执行（剩余作业时间不反应作业的紧迫程度）
  - 由于作业的长短只是根据用户所提供的估计执行时间而定的，用户会有意无意缩短时间

2.6.5/4: HRRF (Highest Response Ratio First)

- FCFS与SJF算法都是片面的调度算法
  - FCFS只考虑作业等候时间而忽视了作业的计算时间
  - SJF只考虑用户估计的作业计算时间而忽视了作业等待时间
- 将两者结合起来，既考虑作业的等待时间也考虑作业的运行时间，这就是得到了非抢占式的高响应比优先算法

2.6.5/4: HRRF (Highest Response Ratio First)

- 响应比：作业进入系统后的等待时间与估计计算时间之和称作该作业的响应时间，作业的响应时间除以作业估计计算时间称作响应比

响应比 = 1 + 已等待时间/估计运行时间

2.6.5/4: HRRF (Highest Response Ratio First)

- 响应比：作业进入系统后的等待时间与估计计算时间之和称作该作业的响应时间，作业的响应时间除以作业估计计算时间称作响应比

响应比 = 1 + 已等待时间/估计运行时间

如果作业等待时间相同，要求服务的时间越短则优先权越高也就是说短作业容易得到较高响应比

2.6.5/4: HRRF (Highest Response Ratio First)

- 响应比：作业进入系统后的等待时间与估计计算时间之和称作该作业的响应时间，作业的响应时间除以作业估计计算时间称作响应比

饥饿现象不会发生

响应比 = 1 + 已等待时间/估计运行时间

如果服务时间相同，则优先权取决于等待时间，待长作业等待时间足够长后，也将获得足够高的响应比

2.6.5/4: HRRF Example

作业	到达时间	所需CPU时间
1	0	20
2	5	15
3	10	5
4	15	10

SJF:
(1, 3, 4, 2)

$$T = (20 + (25 - 10) + (35 - 15) + (50 - 5))/4 = 25$$

$$W = (20/20 + 15/5 + 20/10 + 45/15)/4 = 2.25$$

FCFS:
(1, 2, 3, 4)

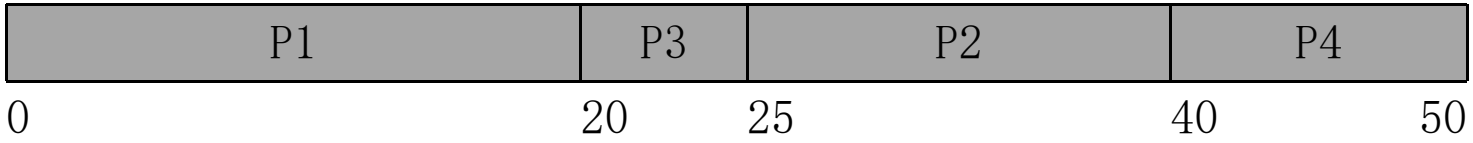
$$T = (20 + (35 - 5) + (40 - 10) + (50 - 15))/4 = 28.75$$

$$W = (20/20 + 30/15 + 30/5 + 35/10)/4 = 3.125$$

2.6.5/4: HRRF Example

- 开始只有作业1，被选中执行时间20
- 作业1执行完毕，响应比依次为1+15/15、1+10/5、1+5/10，作业3被选中
- 作业3执行完毕，响应比依次为1+20/15、1+10/10，作业2被选中
- 作业2执行完毕，作业4被选中，执行时间10

作业	到达时间	所需CPU时间
1	0	20
2	5	15
3	10	5
4	15	10



$$T = (20 + (25 - 10) + (40 - 5) + (50 - 15))/4 = 26.25$$

$$W = (20/20 + 15/5 + 35/15 + 35/10)/4 = 2.46$$

2.6.5/4: HRRF vs. SJF

- HRRF 性能介于 SJF 和 FCFS 算法之间
- 避免了长作业的极端等待（优于SJF, SRTF）
- 每次计算各道作业的响应比会有一定的时间开销，需要估计期待的服务时间，性能比SJF略差

2.6.5/5：PS (Priority Scheduling)

- 优先级调度算法是根据确定的优先级来选取作业，每次总是选择优先数高的作业
- 规定用户作业优先级
  - 外部指定法：用户自己提出作业的优先级
  - 内部指定法：系统综合考虑有关因素来确定用户作业的优先级
- 优先级确定方法：
  - 静态优先级：进程/线程创建时确定，且生命周期中不改变
  - 动态优先级：随时间而改变

2.6.5/5：PS （续）

- 静态优先数法中优先数的确定依据
  - 重要算题程序的进程优先数大，这样有利于用户更早得到结果
  - 交互式用户的进程优先数大，这样有利于终端用户的响应时间等等

实现简单，但会产生饥饿现象

2.6.5/5：PS （续）

- 动态优先数法
  - 在创建一个进程时，根据进程类型和资源使用情况确定一个优先数，当进程耗尽时间片或重新被调度时，再次计算并调整所有进程的优先数
  - 两种原则：
    - 根据进程占有CPU时间多少来决定，当进程占有CPU时间愈长，那么，在它被阻塞之后再次获得调度的优先级就越低，反之，进程获得调度的可能性越大（短进程的优先级高）
    - 根据进程等待CPU时间多少来决定，当进程在就绪队列中等待时间愈长，那么，在它被阻塞之后再次获得调度的优先级就越高，反之，进程获得调度的可能性越小（久进程的优先级高）

2.6.5/5：PS （续）

- 实现方式
  - 剥夺式 ：SRTF（静态优先）
  - 非剥夺式：SJF（静态优先），HRRF（动态优先）

2.6.5/6: RR (Round-Robin)

- 策略：调度程序每次把CPU分配给就绪队列首进程使用一个时间片，例如100ms，就绪队列中的每个进程轮流地运行一个时间片。当这个时间片结束时，强迫一个进程让出处理器，让它排列到就绪队列的尾部，等候下一轮调度（FCFS/SJF/…）
- 实现原理：实现这种调度要使用一个间隔时钟。当一个进程开始运行时，就将时间片的值置入间隔时钟内，当发生间隔时钟中断时，中断处理程序就通知处理器调度进行处理器的切换工作

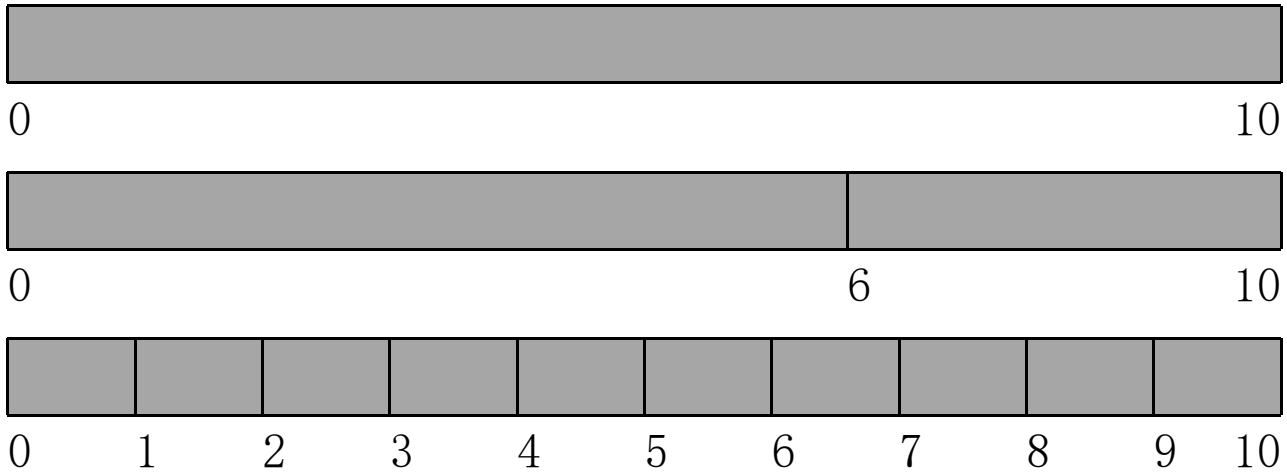
2.6.5/6: RR (Round-Robin)

- 策略：调度程序每次把CPU分配给就绪队列首进程使用一个时间片，例如100ms，就绪队列中的每个进程轮流地运行一个时间片。[illegible]个时间片结束时，强迫一个进程让出处理器[illegible]轮调度（FCF[illegible]
- 实现原理：[illegible]进程开始运行[illegible]发生间隔时钟[illegible]进行处理器的[illegible]

- **基本轮转法：**它要求每个进程轮流运行一个相同的时间片
- **改进的轮转法：**对于不同的进程给以不同的时间片；时间片的长短可以动态地修改等等

2.6.5/6：RR （续）

相同进程（处理时间为10ms）在不同时间片划分下



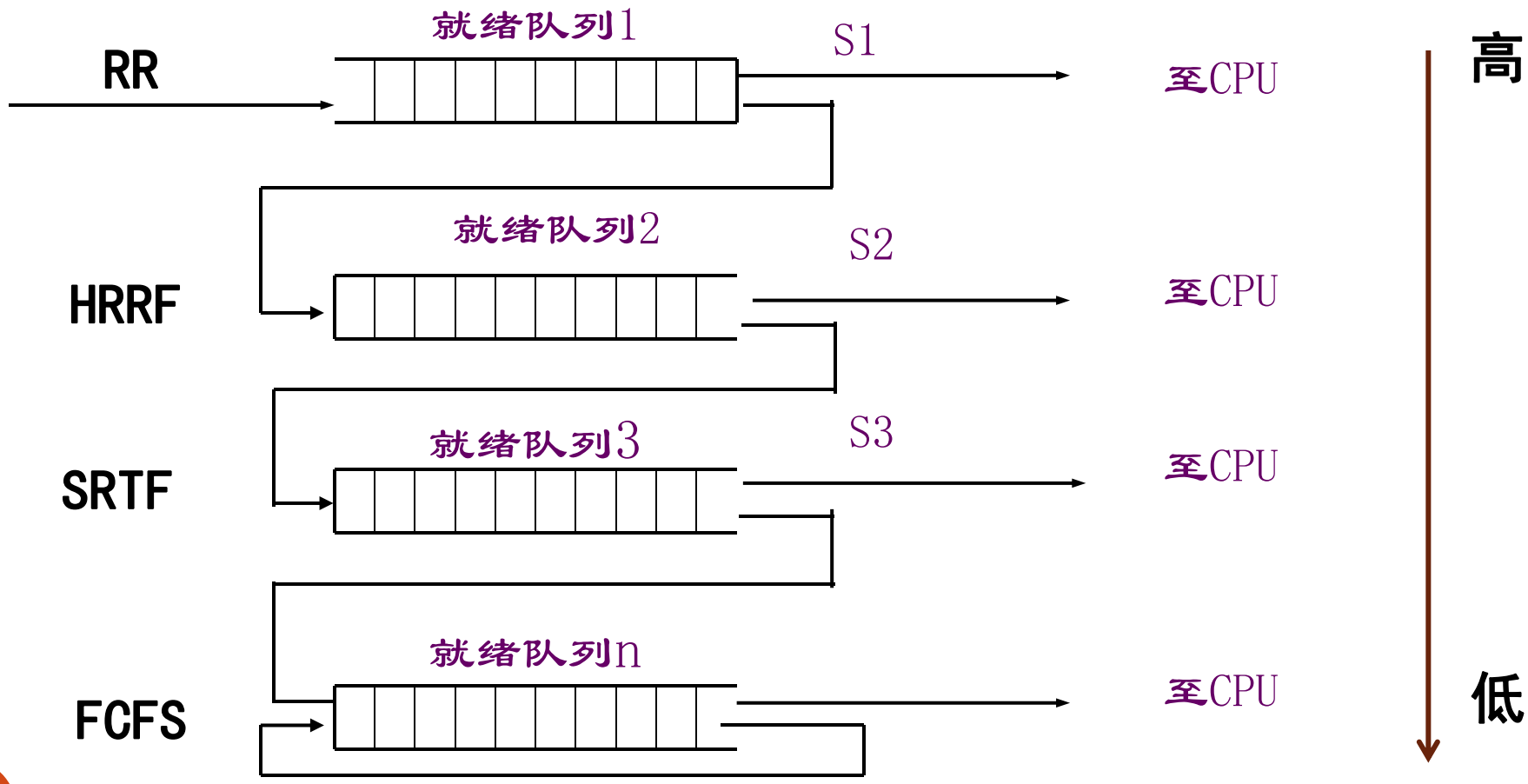
时间片	上下文切换
12	0
6	1
1	9

RR是一种剥夺式调度，系统耗费在进程切换上的开销比较大，这个开销与时间片的大小很有关系

2.6.5/7: MLFQ (Multi-Level Feedback Queue)

- 多级反馈队列调度（反馈循环队列或多队列策略）
  - (1)主要思想：是将就绪进程分为两级或多级，系统相应建立两个或多个就绪进程队列，较高优先级的队列一般分配给较短的时间片。处理器调度每次先从高级就绪进程队列中选取可占有处理器的进程，只有在选不到时，才从较低级的就绪进程队列中选取。同一队列中的进程按先来先服务原则排队。开始工作时，每当一个新进程进入内存后，首先进入高优先级队列等候调度，若能在该级队列的一个时间片内执行完成，则撤离系统，否则进入低一级的队列等候调度，队列级别越低，时间片就越大，低优先级队列中的进程获得调度时运行的时间就长一些
  - 多级反馈队列如图所示：

2.6.5/7: MLFQ (Multi-Level Feedback Queue)



2.6.5/7：MLFQ（续）

- 效果：性能较好，能满足各类用户的需要
  - 对分时交互式作业：通常可在最高优先级队列规定的一个时间片内完成，响应时间快
  - 对于长批处理型作业：可以从高到低在各优先级队列中运行一个时间片直到在某个级别队列中执行完毕或者在最后一个队列中经过若干个时间片执行完毕，决不会发生长批处理型作业长期得不到调度的情况

例子（1）

- 假设一个系统中有5个进程，它们的到达时间和服务时间如表所示，忽略I/O以及其他开销时间，若分别按先来先服务、非抢占式及抢占的短进程优先、时间片轮转、多级反馈队列（FB，第i级的队列的时间片为2的i-1次幂）以及立即抢占的多级反馈队列（FB，第i级的队列的时间片为2的i-1次幂）调度算法进行CPU调度，请给出各进程的完成时间、周转时间、带权周转时间、平均周转时间和平均带权周转时间

进程	到达时间	服务时间
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

	进程	A	B	C	D	E	平均
FCFS	完成时间	3	9	13	18	20	
	周转时间	3	7	9	12	12	8.6
	带权周转时间	1	1.17	2.25	2.4	6	2.56
SPF非抢占	完成时间	3	9	15	20	11	
	周转时间	3	7	11	14	3	7.6
	带权周转时间	1	1.17	2.75	2.8	1.5	1.84
SPF抢占	完成时间	3	15	8	20	10	
	周转时间	3	13	4	14	2	7.2
	带权周转时间	1	2.16	1	2.8	1	1.59
RR（q=1）	完成时间	4	18	17	20	15	
	周转时间	4	16	13	14	7	10.8
	带权周转时间	1.33	2.67	3.25	2.8	3.5	2.71
FB非抢占	完成时间	3	17	18	20	14	
	周转时间	3	15	14	14	6	10.4
	带权周转时间	1	2.5	3.5	2.8	3	2.56
FB抢占	完成时间	4	18	15	20	16	
	周转时间	4	16	11	14	8	10.6
	带权周转时间	1.33	2.67	2.75	2.8	4	2.87

某多道程序设计系统供用户使用的主存为 100K，磁带机 2 台，打印机 1 台。采用可变分区内存管理，采用静态方式分配外围设备，忽略用户作业 I/O 时间。现有作业序列如下：

作业号	进入输入井时间	运行时间	主存需求量	磁带需求	打印机需求
1	8:00	25 分钟	15K	1	1
2	8:20	10 分钟	30K	0	1
3	8:20	20 分钟	60K	1	0
4	8:30	20 分钟	20K	1	0
5	8:35	15 分钟	10K	1	1

作业调度采用 FCFS 策略，优先分配主存低地址区且不准移动已在主存的作业，在主存中的各作业平分 CPU 时间。现求：(1)作业被调度的先后次序?(2)全部作业运行结束的时间?(3)作业平均周转时间为多少?(4)最大作业周转时间为多少?

答：(1)作业调度选择的作业次序为：作业 1、作业 3、作业 4、作业 2 和作业 5。

(2)全部作业运行结束的时间 9:30。

(3)周转时间：作业 1 为 30 分钟、作业 2 为 55 分钟、作业 3 为 40 分钟、作业 4 为 40 分钟和作业 5 为 55 分钟。

(4)平均作业周转时间=44 分钟。

(5))最大作业周转时间为 55 分钟。

分析：本题综合测试了作业调度、进程调度、及对外设的竞争、主存的竞争。

8:00 作业 1 到达，占有资源并调入主存运行。

8:20 作业 2 和 3 同时到达，但作业 2 因分不到打印机，只能在后备队列等待。作业 3 资源满足，可进主存运行，并与作业 1 平分 CPU 时间。

8:30 作业 1 在 8:30 结束，释放磁带与打印机。但作业 2 仍不能执行，因不能移动而没有 30KB 的空闲区，继续等待。作业 4 在 8:30 到达，并进入主存执行，与作业 3 分享 CPU。

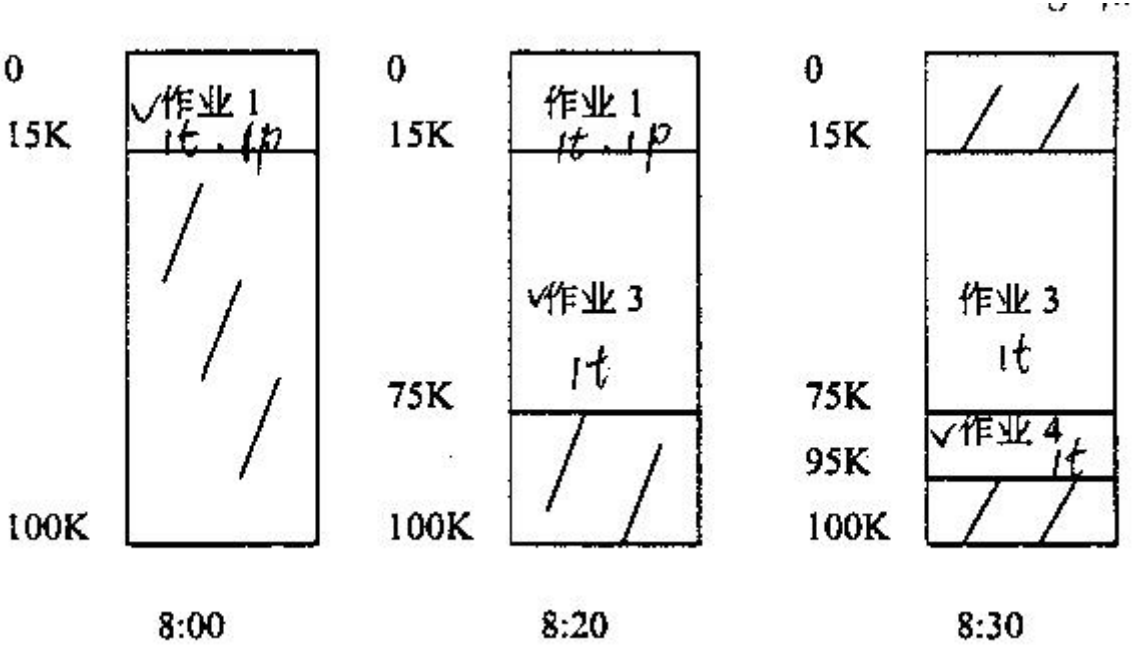
8:35 作业 5 到达，因分不到磁带机/打印机，只能在后备队列等待。

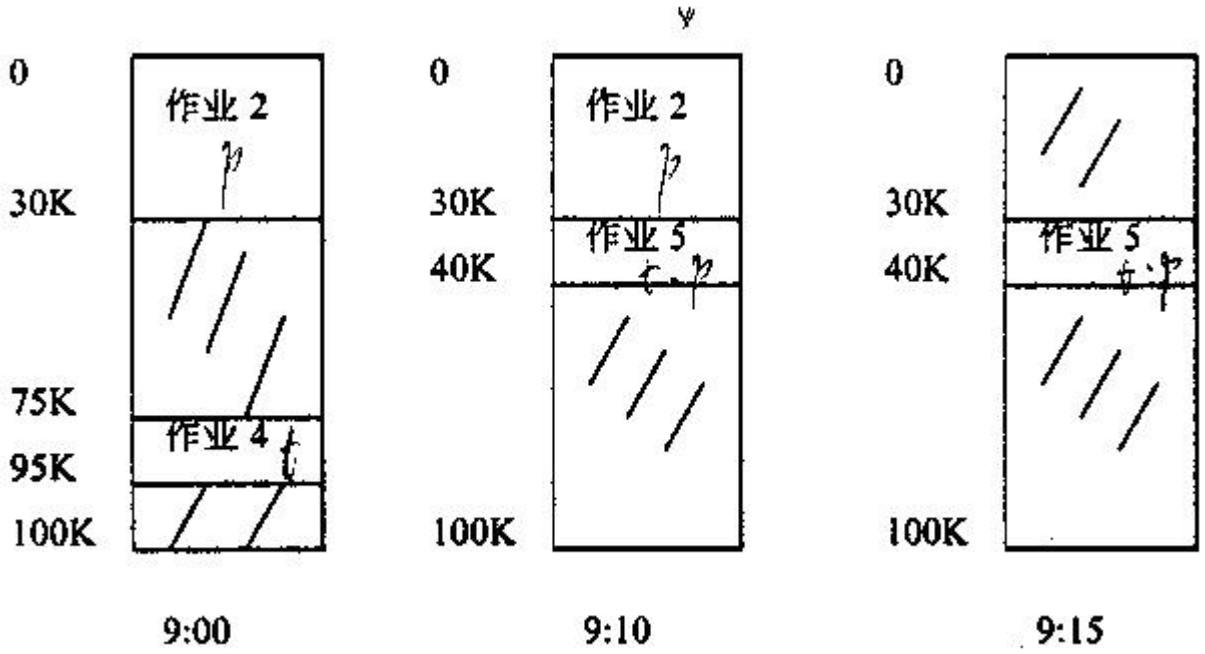
9:00 作业 3 运行结束，释放磁带机。此时作业 2 的主存及打印机均可满足，投入运行。作业 5 到达时间晚，只能等待。

9:10 作业 4 运行结束，作业 5 因分不到打印机，只能在后备队列继续等待。

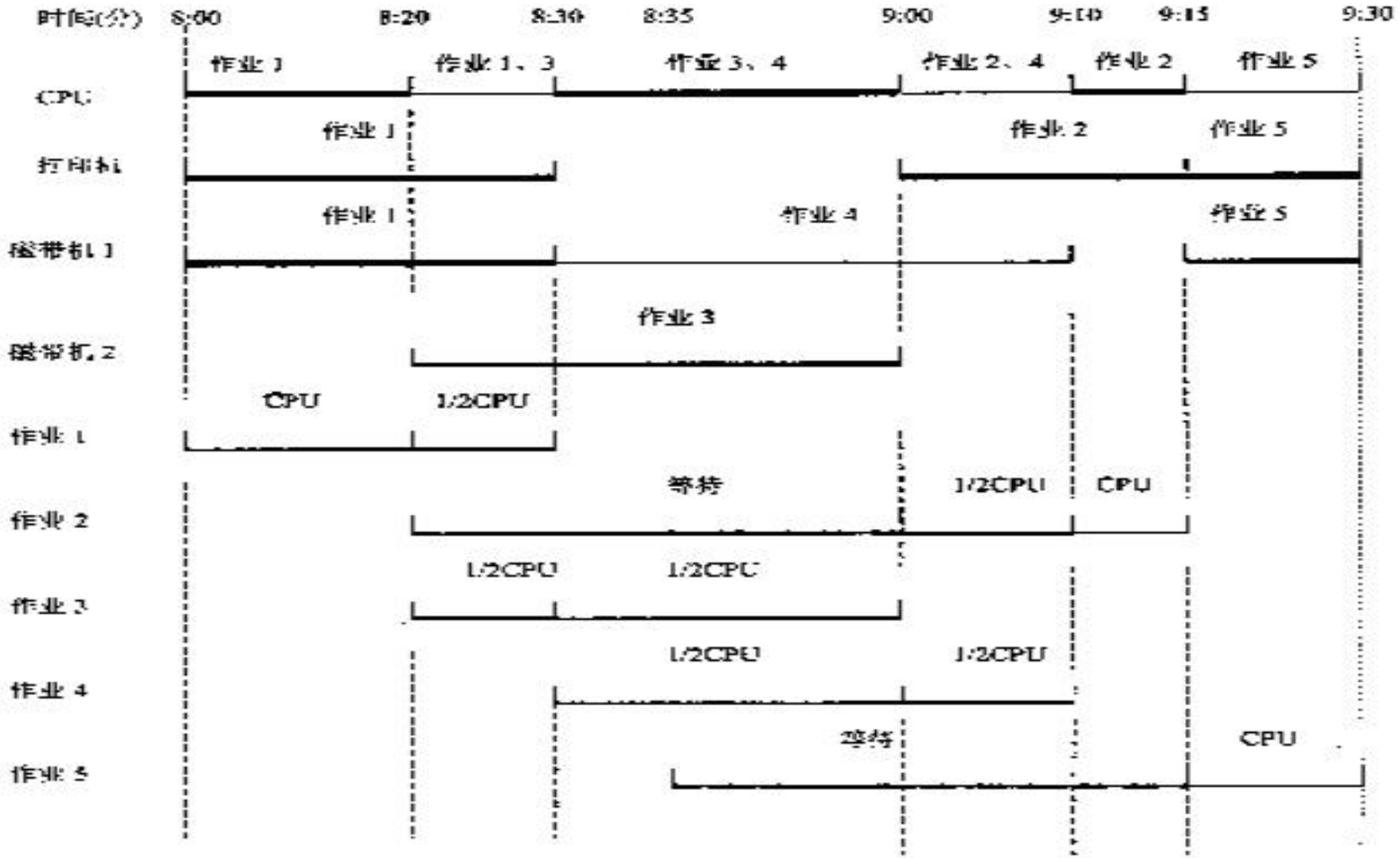
9:15 作业 2 运行结束，作业 5 投入运行。

9:30 作业全部执行结束。


0
15K
作业 1
100K
8:00
0
15K
作业 1
作业 3
75K
100K
8:20
0
15K
作业 3
75K
作业 4
95K
100K
8:30


0
30K
75K
95K
100K
作业 2
作业 4
9:00
0
30K
40K
100K
作业 2
作业 5
9:10
0
30K
40K
100K
作业 5
9:15

《操作系统教程》（第三版）CH2 应用题参考答案



作业

- 二、应用题
  - 7题
  - 11题
  - 18题
  - 20题
  - 25题
  - 28题
  - 32题