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

第六章 处理机调度

授课人：郑然

6.1.1 处理机调度的功能

- 确定数据结构
- 制定调度策略（调度原则）
- 给出调度算法
- 具体的实施处理机分派

不同类型的操作系统往往采用不同的处理机分配方法。

6.1.2 处理机调度的分层实现

只有内存中的程序才能在CPU上运行。因此，处理机的调度通常分为两层：

- **宏观上：作业调度**

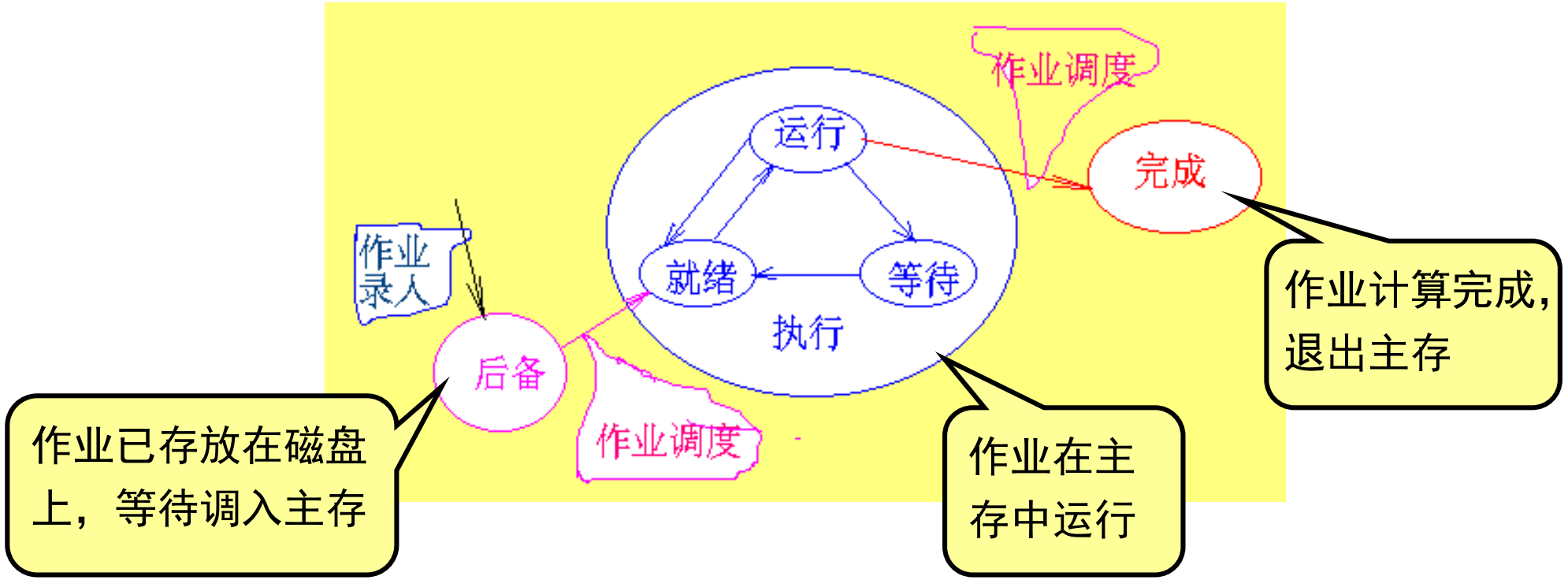
对存放在辅存设备上的大量作业，以一定的策略进行挑选，分配主存等必要的资源，建立作业对应的进程，使其投入运行。

- **微观上：进程调度**

对进入主存的所有进程，确定哪个进程在什么时候获得处理机，使用多长时间。

6.2.1 作业调度

作业调度的主要任务是完成作业从后备状态到执行状态和从执行状态到完成状态的转变。



6.2.2 作业调度的功能

- **确定数据结构**

  建立作业控制块(JCB, Job Control Block)，记录进入系统的作业情况(类型、状态、资源请求与分配等)；

- **确定调度策略与调度算法**

- **分配资源**

  为被选中的作业创建进程，并且为其申请系统资源；

- **善后处理**

  收回作业占用的全部资源，撤销作业控制块以及与该作业有关的全部进程。

6.2.3 作业控制块

- 每个作业进入系统时由系统为其建立一个作业控制块JCB(Job Control Block)
- 存放作业控制和管理信息
- 作业存在的标志

作业名	
资源要求	估计运行时间
	最迟完成时间
	要求的内存量
	要求外设的类型及台数
	要求文件量和输出量
资源使用情况	进入系统的时间
	开始运行的时间
	已运行的时间
	内存地址
	外设台号
类型	控制方式
	作业类型
优先级	
状态	

6.2.4 作业调度算法性能的衡量

作业调度算法规定从后备作业中选择作业进入系统内存的原则。

1. 确定调度算法时应考虑：

(1) 与系统的整体设计目标一致

(2) 考虑系统中各种资源的负载均匀

(3) 保证作业的执行

(4) 对一些专用资源的使用特性的考虑

- 面向用户：周转时间、响应时间、截止时间
- 面向系统：吞吐量、利用率、设备的均衡利用
- 调度算法：易于实现、执行开销

2. 调度性能的衡量

通常采用平均周转时间和带权平均周转时间衡量作业调度算法性能的好坏。

(1) 周转时间：各作业提交给计算机系统到该作业的结果返回给用户所需要的时间

- 定义：$t_i = tc_i - ts_i$

  t_i ：作业周转时间

  tc_i：作业完成时间

  ts_i：作业提交时间

- 意义：说明作业i在系统中停留的时间长短
- 平均周转时间：

$$t = \frac{1}{n}\sum_{i=1}^{n} t_i$$

(2) 带权周转时间

- 定义：一个作业的周转时间与其运行时间的比值
- 意义：说明作业i在系统中的相对等待时间

(3) 平均带权周转时间

- 精确度：高于周转时间和平均周转时间

带权周转时间：

$$w_i = \frac{t_i}{t_r}$$

t_r:作业实际运行时间

平均带权周转时间：

$$W = \frac{1}{n}\sum_{i=1}^{n} W_i$$

6.2.5 作业调度算法

1. 先来先服务调度算法(FCFS)：

- 策略：
  - 按照作业到来的先后次序进行调度
- 特点：
  - 每次选择等待时间最久的作业，而不考虑作业运行时间的长短
  - 实现简单，效率较低，在一些实际的系统和一般应用程序中应用较多

2. 短作业优先调度算法

- 策略：
  - 考虑作业的运行时间，每次总选择一个请求运行时间最小的作业调入内存（系统）。
- 特点：
  - 易实现，系统吞吐量高
  - 只考虑短作业，而没有考虑长作业的利益
  - 相对先来先服务调度算法实现要困难些，如果作业的到来顺序及运行时间不合适，会出现饿死现象

讨论两种调度算法下的周转时间与带权周转时间

作业	提交时间	执行时间	开始时间	完成时间	周转时间	带权周转时间
1	8.00	2.00				
2	8.50	0.50				
3	9.00	0.10				
4	9.50	0.20				

两种调度算法下：

作业执行顺序？

平均周转时间？

平均带权周转时间？

3. 响应比高者优先调度算法

介于这两种算法之间的一种折衷的算法。

响应比 = 响应时间 / 执行时间

= 1 + 等待时间 / 执行时间

每调度一个作业时，计算后备作业表中每个作业的响应比，挑选响应比高者投入运行。

理论上比较完备，但作业调度程序要统计作业的等待时间、用户估计的运行时间，并要作浮点运算（这是系统程序最忌讳的）浪费大量的计算时间，这是系统程序不允许的。

4. 优先数调度算法

综合考虑各方面因素（作业等待时间、运行时间、缓急程度，系统资源使用等）给每个作业设置一个优先数，调度程序总是选择一个优先数最大（或者最小）的作业调入（系统）内存。

困难在于如何综合考虑，这些因素之间的关系如何处理。

5. 均衡调度算法

一种更为理想化的调度算法，如何实现更困难，并且算法本身的开销有时会远远大于先来先服务和小作业优先调度算法的不足，这也是这两种算法被众多系统采用的最根本的原因。

6.3.1 调度/分派结构

处理机分配由调度和分派两个功能组成。

1. 调度：

组织和维护就绪进程队列。包括确定调度算法、按调度算法组织和维护就绪进程队列。（按调度原则选择进程）

2. 分派：

当处理机空闲时，从就绪队列队首中移一个PCB，并将该进程投入运行。（赋予使用处理机的权限）

6.3.2 进程调度的功能

1. 记录和保持系统中所有进程的有关情况和状态特征
2. 决定分配（处理机）策略

   调度策略的不同，组织就绪进程队列的方式也不同。

   - 先来先服务调度：就绪进程按等待时间大小的顺序排队
   - 优先数调度：就绪进程按优先数的先后次序排队

3. 实施处理机的分配和回收

6.3.3 进程调度方式

- 非剥夺方式
  - 优点：实现简单，系统开销小
  - 缺点：难以满足紧急任务的要求
- 剥夺方式
  - 优点：及时响应紧急任务
  - 缺点：增加了系统开销

实现中还可采用选择可抢占策略。

1. 进程优先数调度算法

目前操作系统广泛采用的一种进程调度算法，按照某种原则由系统（或用户、或系统与用户结合）赋予每个进程一个优先数，在处理机空闲时，进程调度程序就从就绪进程中选择一个优先数最大（或者最小）的进程占用CPU（进程从就绪状态转换成运行状态）。

采用这种调度算法的关键是如何确定进程的优先数、一个进程的优先数确定之后是固定的，还是随着该进程运行的情况的变化而变化。

优先数的确定

- 静态：
  - 进程优先数在进程创建时确定后就不再变化
    - 系统确定：（运行时间、使用资源，进程的类型）
    - 用户确定：（紧迫程度，计费与进程优先数有关）

    系统与用户结合（用户可以为本用户的进程设置优先数，但不是作调度用，系统还要根据系统情况把用户设置的进程优先数作为确定进程优先数的一个参数）
- 动态：

  系统运行过程中，根据系统的设计目标，不断调整进程的优先数，其优点是能比较客观地反映进程的实际情况和保证达到系统设计目标。

2. 循环轮转调度算法

系统的响应时间分成大小相等（或不等）的时间片。每个进程被调度到后，占用一个时间片，时间片用完后，该进程让出CPU，排在就绪队列的队尾。多个进程循环轮转。

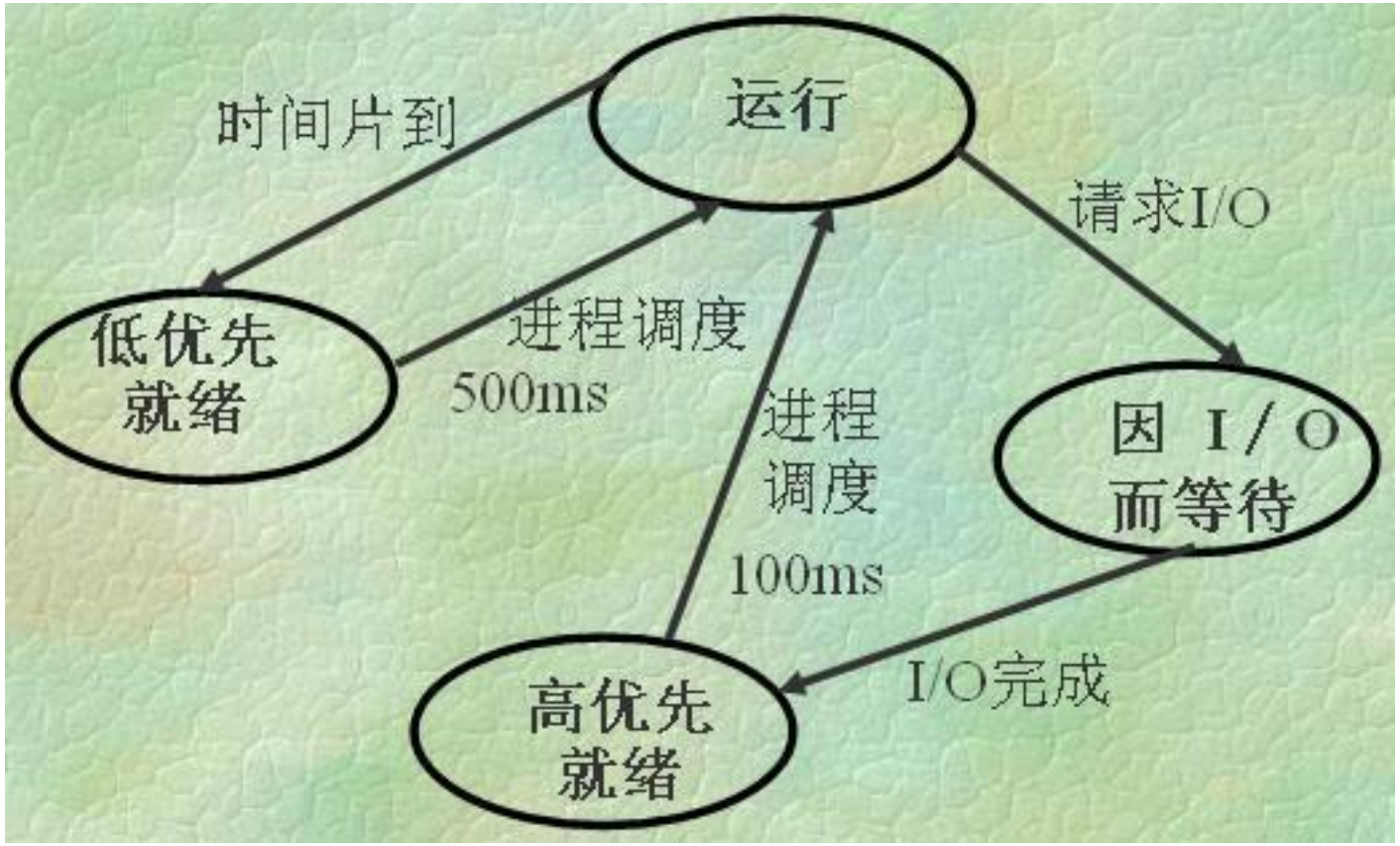
T(响应时间)=N(进程数目)*q(时间片)

- 简单循环轮转调度
  - 实现简单、系统开销小
  - 不灵活，当系统中进程较多时，系统开销变大
- 可变时间片轮转调度
  - 根据系统当前的进程数动态确定时间片的大小

3. 多级反馈算法

- 多个简单算法的综合和发展

6.3.5 调度用的进程状态变迁图



通过进程调度变迁图，可反映系统进程调度的各种特征。

1. 队列结构

- I/O等待对列——进程如果请求I/O，则进入I/O等待队列
- 低优先就绪队列——进程如果在运行中超过了分配给它的时间片，就进入低优先就绪
- 高优先就绪队列——进程从等待状态变为就绪状态时，进入高优先就绪队列

2. 进程调度算法

- 当CPU空闲时，若高优先就绪队列非空，则从高优先就绪队列中选择一个进程运行，分配时间片为100ms。

- 当CPU空闲时，若高优先就绪队列为空，则从低优先就绪队列中选择一个进程运行，分配时间片为500ms。

  优先调度与时间片调度相结合的调度策略

3. 调度效果

- 优先照顾了I/O量大的进程；
- 适当照顾了计算量大的进程。

6.4 Unix进程调度

- UNIX系统采用优先数调度算法
  - 进程有一个进程优先数p_pri
  - p_pri取值范围是 - 127 ~ 127，其值越小，进程的优先级越高

1. 优先数的确定

- 0＃进程（－100）；
- 资源请求得不到满足的进程，磁盘（－80），打印机（－20），…；
- 等待块设备I/O完成的进程(－50)；
- 等待字符设备I/O完成的进程(0～20）；
- 所有处于用户态运行进程同步（一般情况下为大于100）。

目的

充分利用系统资源，即提高系统资源的使用效率

2. 优先数的计算

（1）计算公式：

p_pri = min{127, (p_cpu/16+p_nice+PUSER)}

其中：

p_cpu 进程占用CPU的程度R_1：

$R_1 = T_u/(T_u+T_{nu})$

T_u:进程创建后占用CPU的累计值

T_{nu}：进程生成后没有占用CPU的累计值

p_nice 用户通过系统调用nice()设置的进程优先数

PUSER 常数，其值为100

大量的统计工作，
并且要做浮点运算

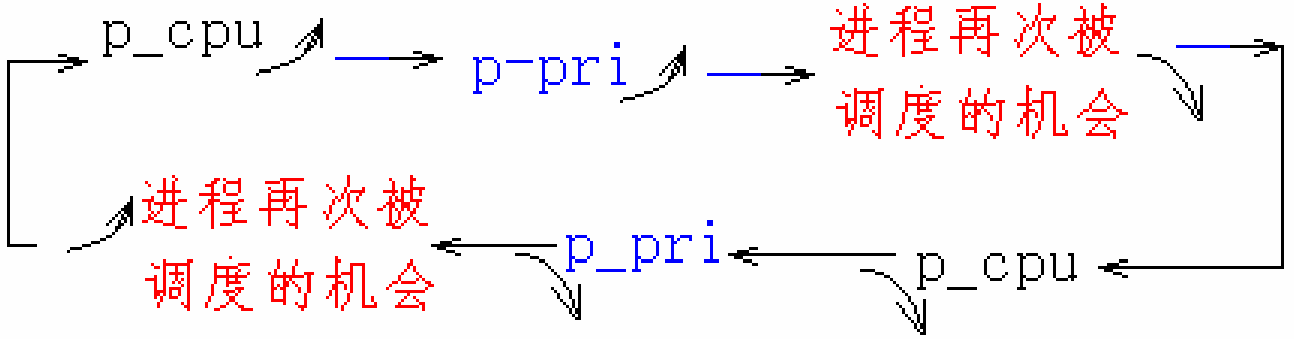
UINX系统中对p_cpu的处理：

每个时钟中断当前占用处理机的进程的 p_cpu++;

每秒钟（时钟中断）（对所有进程）：

```
p_cpu=p_cpu-10;
if (p_cpu-SCHMAG < 0 )
   p_cpu = 0;
```

其中：SCHMAG =10



这种负反馈的效果使得系统中在用户态下运行的进程能均衡地得到处理机

第六章 小结

一. 处理机的二级调度

二. 作业调度

1. 作业的状态
2. 作业控制块
3. 作业调度的功能
4. 周转时间、带权周转时间：定义、物理意义
5. 常用的作业调度算法

先来先服务 短作业优先

三. 进程调度

1. 进程调度的功能
2. 调度方式： 非剥夺方式 剥夺方式
3. 常用的进程调度算法：

优先数调度

循环轮转调度

4. 调度用的进程状态变迁图

(多种进程状态及变迁)