第5章 进程调度

5.2 调度算法

5.2 调度算法

- CPU调度程序将执行调度算法从就绪队列中选择进程执行。
- 有多种不同的调度算法。调度算法的优劣影响整个系统的性能。
- 本部分的算法有些适合作业调度，有些适合进程调度，有些适用于两者。

（1） 先来先服务调度算法

- 先来先服务(First-Come First-Served，FCFS)调度算法既可用于作业调度，也可用于进程调度。
- 在作业调度中：从后备作业队列中选择一个或多个最先进入该队列的作业（等待时间最长的队列），将它们调入内存，为它们分配资源，创建进程，然后放入就绪队列。
- 进程调度中：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，使之投入运行。该进程一直运行到完成或因等待某一事件而阻塞时才释放处理机。

先来先服务调度算法例

- 设有4道作业，它们的提交时间及执行时间如下表，若按先来先服务调度算法进行调度，试计算4个作业的平均周转时间和平均带权周转时间。（时间单位：小时，以十进制计算）。

作业	提交时间	估计运行时间
1	10	2
2	10.2	1
3	10.4	0.5
4	10.5	0.3

作业周转时间及带权周转时间的计算

- 平均周转时间T=(2.0 + 2.8 + 3.1 + 3.3)/4=2.8
- 平均带权周转时间W=(1 + 2.8 + 6.2 + 11)/4=5.25

作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
1	10	2	10	12	2	1
2	10.2	1	12	13	2.8	2.8
3	10.4	0.5	13	13.5	3.1	6.2
4	10.5	0.3	13.5	13.8	3.3	11

先来先服务算法特点

- FCFS是非抢占式算法
  - 一旦分配，一直保持，直到释放CPU为止，即程序终止或者请求I/O
- 优点
  - 算法简单，易于实现，
- 缺点
  - 不利于短作业：只顾忌了作业等待时间，而未考虑作业要求的服务时间
  - 不利于I/O密集型（I/O-bound）作业：Convoy Effect
  - 不利于分时系统：每一个作业都需要一定时间，可能导致分配时间过长

（2） 短作业优先调度算法

- 短作业优先（Shortest-job-first，SJF）调度算法同时适合作业和进程调度

- 在作业调度中，从后备队列中选择一个或多个估计运行时间最短的作业，将它们调入内存运行。

- 在进程调度中，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，使之投入运行。该进程一直运行到完成或因等待某一事件而阻塞时才释放处理机。

短作业优先调度算法例

- 平均周转时间　T=(2.0+1.8+2.4+3.6)/4=2.45
- 平均带权周转时间　W=(1+6+4.8+3.6)/4=3.85

作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
1	10	2	10	12	2	1
2	10.2	1	12.8	13.8	3.6	3.6
3	10.4	0.5	12.3	12.8	2.4	4.8
4	10.5	0.3	12	12.3	1.8	6

短作业优先调度算法的特点

- 短作业优先调度是非抢占式算法
- 算法调度性能较好

如上例中，	先来先服务	短作业优先
平均周转时间	2.8	2.45
平均带权周转时间	5.25	3.85

  - **可以证明，当一批作业同时到达时，最短作业优先调度算法，是最佳算法，能获得最短平均周转时间。**
- 但对长作业不利，未考虑作业的紧迫程度，运行时间为估计。
  - 当SFJ用于低级调度时：
    - 需要计算的是进程在下一个CPU周期长度，而不是整个进程的用时长度。
    - 实则为“最短下一个CPU用时优先”算法(shortest next CPU burst)

下一个CPU时间的长度

- SJF的困难：如何知道下一个CPU区间的长度?
  - 作业调度:
    - 用户提交作业所指定的进程时间极限作为长度
  - 进程调度:
    - 计算下一个CPU区间长度的近似值，通过历史值预测下一个CPU区间
    - t_n:最近第n个CPU区间的实际执行时间
    - τ_{n+1}:第$n+1$个CPU区间的预测时间
    - α，权重参数，$0 \leq \alpha \leq 1$,
    - 定义：$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$
    - 注意：α=0? α=1? $\alpha = \frac{1}{2}$?

最短剩余时间优先调度算法

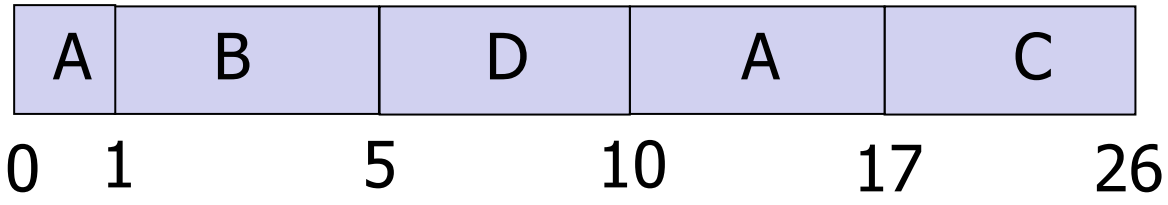
- 最短作业优先是非抢占式的，当应用到低级调度，可以改造为抢占式的。

- 抢占式的最短进程优先调度算法也称为最短剩余时间优先（Shortest-remaining-time-first，SRTF）调度算法，即当一个新进程进入就绪队列时，若其需要的运行时间比当前运行进程的剩余时间短，则它将抢占CPU。

最短剩余时间优先算法例

- 调度：

进程	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
A	0	8	0	17	17	2.125
B	1	4	1	5	4	1
C	2	9	17	26	24	2.67
D	3	5	5	10	7	1.4



最短剩余时间优先算法例（续）

- 平均周转时间　T=(17 + 4 + 24 + 7)/4=13
- 平均带权周转时间 W=(2.125 + 1 + 2.67 + 1.4)/4=1.8

进程	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
A	0	8	0	17	17	2.125
B	1	4	1	5	4	1
C	2	9	17	26	24	2.67
D	3	5	5	10	7	1.4

（3） 优先级调度算法

- 优先级 （Priority）表示进程的重要性及运行优先性，优先级调度（Highest Priority First, HPF ）总是选择就绪队列中优先级最高的进程执行。
  - 用于作业调度，就是选择优先级最高的作业进入内存。
- 优先级相同按照FCFS调度。
- 优先级通常用优先数来衡量。在某些系统中，优先数越大优先级越高；而在另一些系统中，优先数越大优先级越小。

按调度方式对优先级调度算法分类

- 非抢占式优先级调度算法：
  - 系统一旦将处理机分配给就绪队列中优先级最高的进程后，该进程便一直运行下去，直到完成或因发生某事件使该进程放弃处理机时，系统才将处理机分配给另一个更高优先级的进程。
  - 优先级体现：非抢占算法只是将新的进程，按照优先级，加到就绪队列的头部

- 抢占式优先级调度算法：
  - 将处理机分配给优先级最高的进程，使之运行。在进程运行过程中，一旦出现了另一个优先级更高的进程时，进程调度程序就停止原运行进程，而将处理机分配给新出现的高优先级进程。
  - 优先级体现：优先级高的进程只要就绪，就抢占当前进程

优先级的类型

- 优先级分为两种：
  - 静态优先级
  - 动态优先级

静态优先级

- 静态优先级是在创建进程时确定的，确定之后在整个进程运行期间不再改变。
- 确定依据有：
  - 进程类型：系统，用户
  - 进程对资源的需求：执行时间，资源数量
  - 用户要求：紧迫程度
- 特点：简单易行，系统开销小，但不精确。存在饥饿现象。

动态优先级

- 动态优先级是指在创建进程时，根据进程的特点及相关情况确定一个优先级，在进程运行过程中再根据情况的变化调整优先级。

- 确定原则有：占用CPU时间，等待时间。
  - 例：优先数=CPU使用时间/2+基本优先数

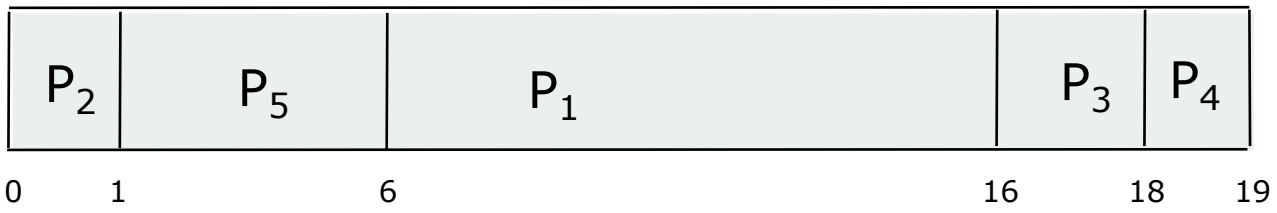
优先级调度算法的问题

- 无穷阻塞/饥饿
  - 超负载计算机系统中，低优先级进程会无穷等待CPU
  - 例：1973年关闭的MIT IBM7094，发现了1967年提交的低优先级进程
- 解决方案：老化aging
  - 对低优先级进程，随着等待时间，逐渐增加其优先级
  - 例：进程优先值为0-127（值越低优先级越高），以15分钟递减优先值（-1），则从127开始，不超过32小时，可以老化为优先级值为0

优先级调度算法例1

有一组进程，它们在0时刻按P_1 - P_5的顺序到达，各自的执行时间和优先数（优先数越小优先级越高）如下。请给出非抢占优先级调度的调度结果，并计算平均等待时间。

进程	运行时间	优先数
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2



- 平均等待时间= 8.2

优先级调度算法例2

进程	运行时间	优先数	到达时间
P_1	10	3	0
P_2	1	1	1
P_3	2	4	2
P_4	1	5	3
P_5	5	2	4

如果进程的到达时间不同，请给出抢占优先级调度的调度结果，并计算平均等待时间。

（4） 时间片轮转调度算法

- 时间片轮转法（Round-robin, RR）：
- 专门为分时系统设计的进程调度算法。
- 系统将所有就绪进程按到达时间的先后次序排成一个队列，每次调度时把CPU分配给队首进程，并令其执行一个时间片。
- 当时间片用完时，停止该进程的执行，将它送至就绪队列末尾等待下一次执行，然后再把处理机分配给就绪队列中的新队首进程。
- 如此不断循环，直至完成为止。

时间片轮转算法例

- 设有A、B、C、D、E五个进程，其到达时间和执行时间如下表，采用时间片轮转调度算法，当时间片大小为1和4时，试计算其平均周转时间和平均带权周转时间。（设：这里到达指的是已经进入就绪队列，即新到达进程，排在这一时刻，其他刚进入（因时间片到而转入）就绪队列的进程前面）

作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
A	0	3				
B	1	6				
C	2	4				
D	3	5				
E	4	2				

时间片大小为1

- A、B、C、D、E要求运行时间依次为3、6、4、5、2，到达时间依次为0、1、2、3、4。

0：A运行；
1：B运行，A等待；
2：A运行，CB等待；
3：C运行，BDA等待；
4：B运行，DAEC等待；
5：D运行，AECB等待；
6：A运行，ECBD等待；
7：E运行，CBD等待；
8：C运行，BDE等待；
9：B运行，DEC等待；
10：D运行，ECB等待；
11：E运行，CBD等待；
12：C运行，BD等待；
13：B运行，DC等待；
14：D运行，CB等待；
15：C运行，BD等待；
16：B运行，D等待；
17：D运行，B等待；
18：B运行，D等待；
19：D运行，

时间片为1的周转时间

- 平均周转时间　T=(7 + 18 + 14 + 17+8)/5=12.8
- 平均带权周转时间 W=(2.33 + 3 + 3.5 + 3.4+4)/5=3.246

作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
A	0	3	0	7	7	2.33
B	1	6	1	19	18	3
C	2	4	3	16	14	3.5
D	3	5	5	20	17	3.4
E	4	2	7	12	8	4

时间片大小为4

- A、B、C、D、E要求运行时间 依次为3、6、4、5、2，到达时间依次为0、1、2、3、4。

0：A运行，BCD依次到达；
3：B运行，CD等待，后E到达；
7：C运行，DEB等待；
11：D运行，EB等待；
15：E运行，BD等待；
17：B运行，D等待；
19：D运行

时间片为4的周转时间

- 平均周转时间　T=(3＋18＋9＋17+13)/5=12
- 平均带权周转时间 W=(1＋3＋2.25＋3.4+6.5)/5=3.23

作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
A	0	3	0	3	3	1
B	1	6	3	19	18	3
C	2	4	7	11	9	2.25
D	3	5	11	20	17	3.4
E	4	2	15	17	13	6.5

时间片大小的选择

① 若时间片太大，所有进程都能在一个时间片内完成，则时间片轮转算法退化为先来先服务；

② 若时间片太小，则进程调度频繁，会有大量时间用于上下文切换，系统开销增加。

因此时间片大小选择应适当。

确定时间片大小应考虑的因素

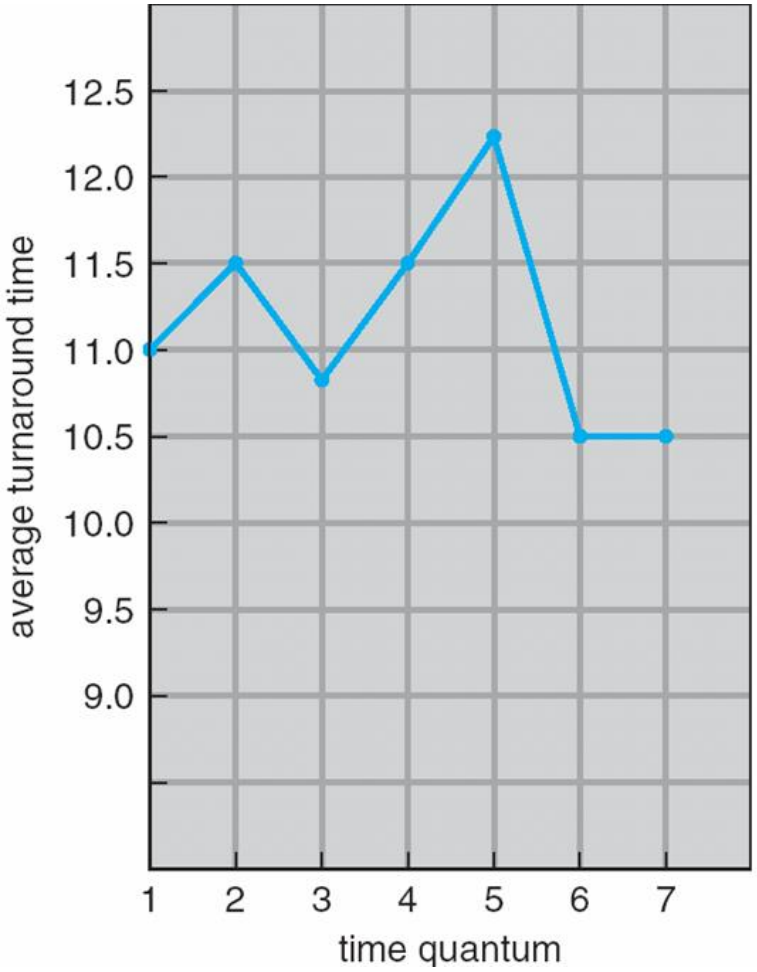
① 系统对响应时间的要求：
- 响应时间=时间片*进程数。进程数一定，则时间片与系统响应时间成正比。

② 就绪队列中的进程数目
- 当响应时间要求固定，时间片与就绪进程数成反比。

③ 系统处理能力：
- 人所能承受的响应时间一定，系统速度快则时间片可增长。

④ 上下文切换时间：
- 一般上下文切换时间(10us)为时间片(10-100ms)的很小一部分

平均周转时间和时间片的关系

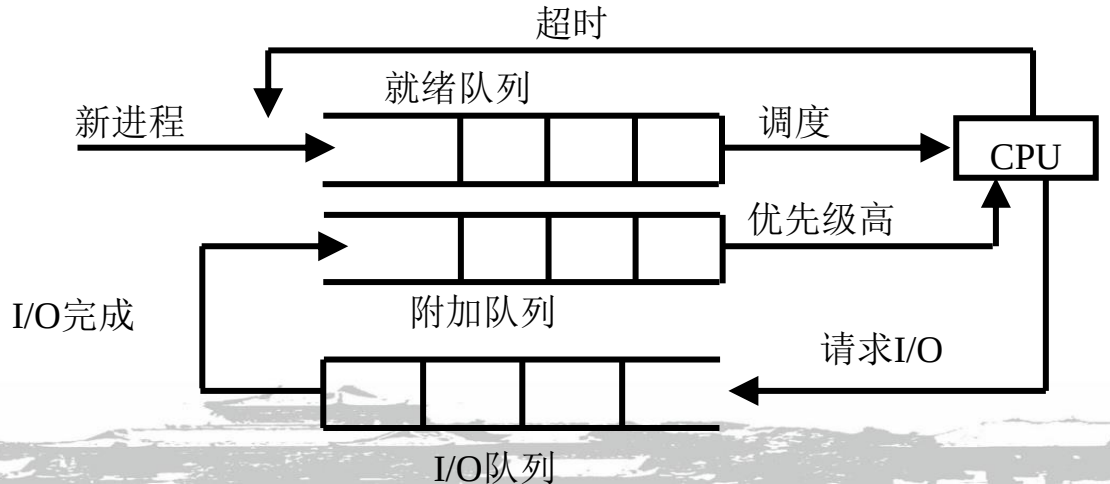


process	time
P_1	6
P_2	3
P_3	1
P_4	7

根据经验：80%的cpu区间应该小于时间片

时间片轮转算法的特点及改进

- RR算法特点：
  - 对偏重I/O的进程不公平
  - 改进为虚拟时间片轮转算法。
- 虚拟时间片轮转算法：
  - 新进程基于FCFS进入就绪队列，进程用完时间片后也进入就绪队列
  - 进程因I/O阻塞进入I/O队列，I/O完成时进程进入附加队列，附加队列的优先级高于就绪队列
  - 当进程从附加队列被调度时，其运行时间不超过上次发生中断时剩余的时间。



（5） 高响应比优先调度算法

- 先来先服务的局限性：片面考虑作业等待时间，忽视计算时间
- 短作业优先的局限性：片面考虑计算时间，忽视作业等待时间
- 高响应比优先调度（Highest Response Ratio First，HRRF）算法是对短作业优先调度算法和先来先服务调度算法的一种综合。
- 响应比的定义：

响应比 = 作业响应时间/估计运行时间

其中： 响应时间=作业等待时间+估计运行时间

因此：

响应比 = 1 + 作业等待时间/估计运行时间

最高响应比优先调度算法思想

- 在每次调度作业运行时，先计算后备作业队列中每个作业的响应比，然后挑选响应比最高者投入运行。
- 特点：
  - 有利于短作业——等待时间相同，短作业优先，
  - 考虑等待时间——运行时间相同，等待时间长的作业优先运行。
  - 主要用于作业调度，非抢占调度

最高响应比优先算法例

- 设有A、B、C、D、E五个进程，其到达时间分别为0、1、2、3、4，要求运行时间依次为3、6、4、5、2，采用最高响应比优先调度算法，试计算其平均周转时间和平均带权周转时间。
  - 调度顺序：

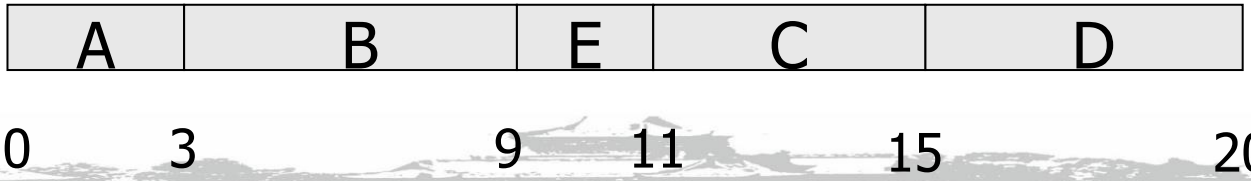
    0：A运行，BCD依次到达；

    3：$r_B=1+2/6$，$r_C=1+1/4$，$r_D=1$；B先运行。

    9：$r_C=1+7/4$，$r_D=1+6/5$，$r_E=1+5/2$；E先运行。

    11：$r_C=1+9/4$，$r_D=1+8/5$；C先运行。

  - 由此可知作业的运行顺序为A、B、E、C、D。



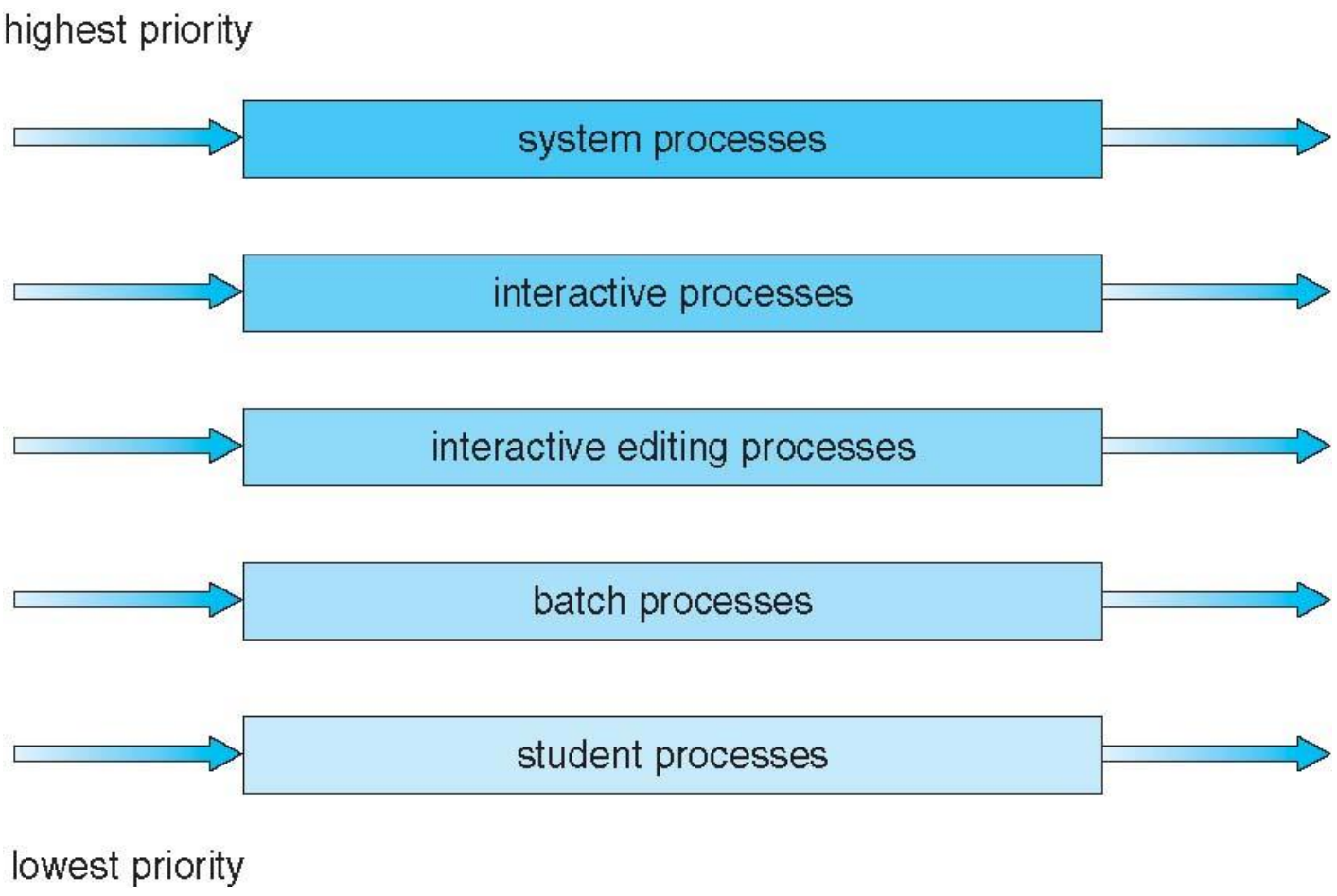
周转时间的计算

- 平均周转时间　T=(3＋8＋13＋17+7)/5=9.6
- 平均带权周转时间 W=(1＋1.33＋3.25＋3.4+3.5)/5=2.496

作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
A	0	3	0	3	3	1
B	1	6	3	9	8	1.33
C	2	4	11	15	13	3.25
D	3	5	15	20	17	3.4
E	4	2	9	11	7	3.5

（6） 多级队列调度算法

- 多级队列调度算法（Multilevel Queue Scheduling）思想：根据作业性质或类型不同，将进程就绪队列分为多个，每个队列采用不同的调度算法。
- 例如：
  - 终端型作业（交互）为前台作业，批处理作业为后台作业。
  - 前台采用时间片轮转算法，后台采用先来先服务
  - 前台作业的优先级高。
  - 高优先级进程可以抢占低优先级进程
- 缺点：
  - 进程进入系统被永久分配到某个队列中
  - 虽然调度开销低，但是不灵活


highest priority
system processes
interactive processes
interactive editing processes
batch processes
student processes
lowest priority

（7） 多级反馈队列调度算法（1）

- 多级队列调度中，进程所在的队列不变，而多级反馈队列（Multilevel Feedback Queue，MLFQ）的不同在于，进程可以变换队列排队。
- 思想：
  - 设置多个就绪队列，并为每个队列赋予不同的优先级。第1个队列的优先级最高，第2队列次之，其余队列的优先级逐次降低。
  - 每个队列中进程执行的时间片大小也各不相同，进程所在队列的**优先级越高**，其相应的**时间片就越短**。

多级反馈队列调度算法（2）

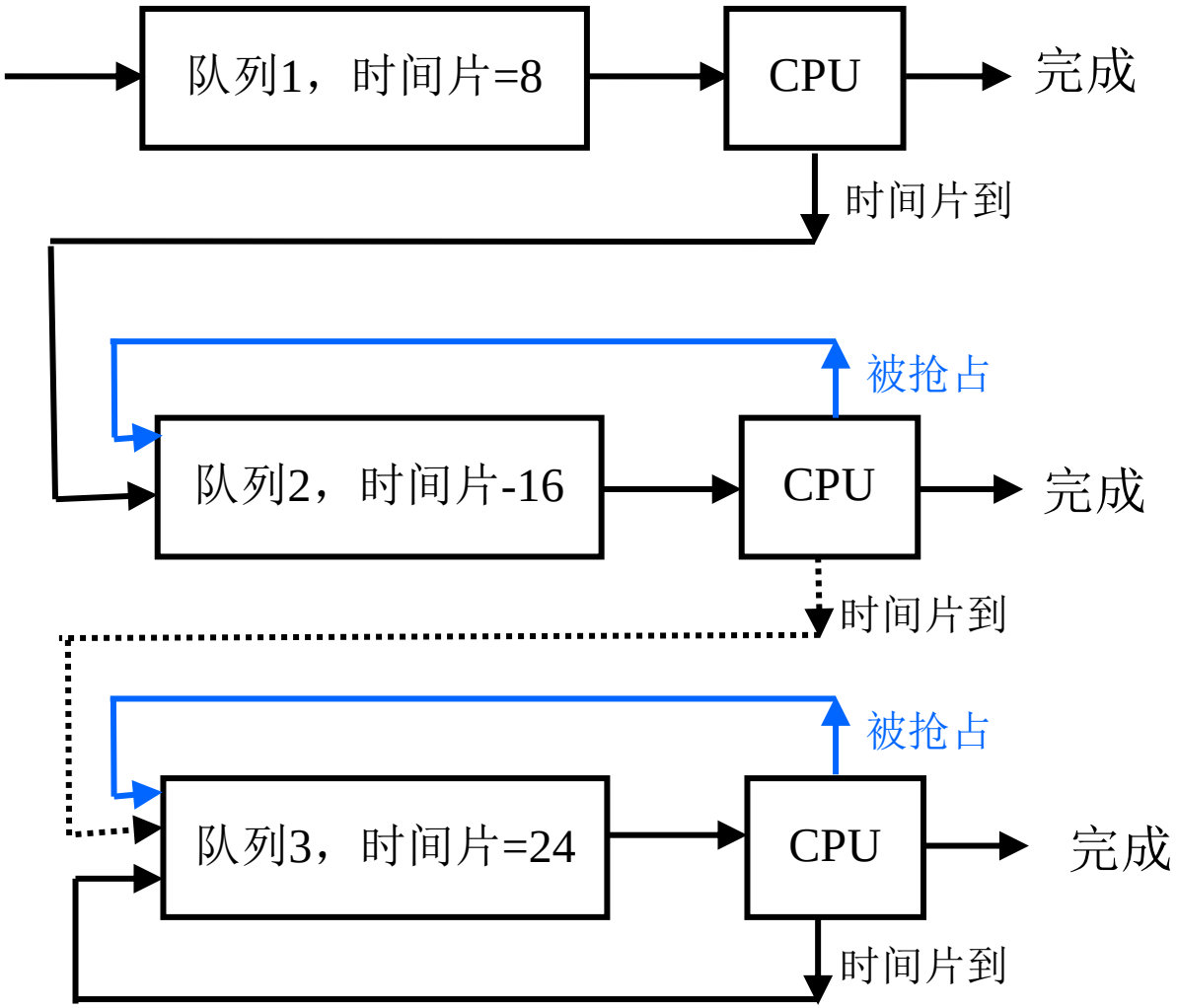
- 当一个新进程进入系统时，首先将它放入第1个队列的末尾，按**先来先服务**的原则排队等待调度。
- 当轮到该进程执行时，如能在此时间片内完成，便可准备撤离系统；
- 如果它在一个时间片结束时尚未完成，调度程序便**将该进程转入第2队列的末尾**，再同样地按FCFS原则等待调度执行。
- 如此下去，最后一个队列中使用某种调度算法。

多级反馈队列调度算法（3）

- 仅当第1个队列为空时，调度程序才调度第2队列中的进程运行；仅当第1个至第（i - 1）个队列均为空时，才会调度第i个队列中的进程运行。

- 当处理机正在为第i个队列中的某进程服务时，若又有**新进程**进入优先级较高的队列中，则此时新进程将**抢占**正在运行进程的处理机，即由调度程序把正在执行进程放回**第i个队列末尾**，重新将处理机分配给新进程。

多级反馈队列调度算法例子



多级反馈队列调度算法的特点

- 特点
  - 允许进程在队列之间移动
  - 根据CPU区间的特点区分进程，如果进程使用过多的CPU时间，就会被移动到低优先级队列
- 多级反馈队列是非常复杂调度程序，通过配置参数来定义最佳的调度程序
  - 队列数量
  - 每个队列的调度算法
  - 用以确定何时升级到最高优先级队列的方法
  - 用以确定何时降级到最低优先级的方法
  - 用以确定进程在需要服务时将会进入哪个队列的方法

多级反馈队列调度算法的性能

- 多级反馈队列调度算法能较好满足各类用户的需求：
  - 终端型用户：大多能在一个时间片内完成，响应时间较短；
  - 短批处理作业用户：能在前几个队列完成，周转时间较短；
  - 长批处理作业用户：依次在1～n队列中运行，不会长时间得不到处理。

多级反馈队列调度算法的问题

- 问题：MLFQ会导致饥饿
  - 当一个长作业进入系统，最终必将移入优先级最低的就绪队列，则如果有大量短作业形成稳定工作流，则长作业陷入饥饿。
  - 解决办法：提升低优先级队列中等待时间过长的进程优先级。

多级反馈队列调度算法例

- 设有A、B、C、D、E五个进程，其到达时间分别为0、1、3、4、5，要求运行时间依次为3、8、4、5、7，采用多级反馈队列调度算法，系统中共有3个队列，其时间片依次为1、2和4，最后一级采用时间片轮转调度，试计算其平均周转时间和平均带权周转时间。

调度分析

- A、B、C、D、E到达时间依次为0、1、3、4、5，要求运行时间依次为3、8、4、5、7

0：A运行；
1：B运行，A等待；
2：A运行，B等待；
3：C运行，BA等待；
4：D运行，BAC等待；
5：E运行，BACD等待；
6：BB运行，ACDE等待；
8：A运行，CDE等待；B等待
9：CC运行，DE等待；B等待
11：DD运行，E等待；BC等待
13：EE运行，BCD等待；
15：BBBB运行，CDE等待；
19：C运行，DEB等待；
20：DD运行，EB等待；
22：EEEE运行，B等待；
26：B运行。

周转时间的计算

- 平均周转时间　T=(9＋26＋17＋18+21)/5=18.25
- 平均带权周转时间 W=(3＋3.25＋4.25＋3.6+3)/5=3.42

作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
A	0	3	0	9	9	3
B	1	8	1	27	26	3.25
C	3	4	3	20	17	4.25
D	4	5	4	22	18	3.6
E	5	7	5	26	21	3

（8）公平分享调度算法

- 公平分享调度（Fair-share Scheduling）算法
  - 动机：
    - 采用时间片轮转(Round-Robin)的情况下，**多个用户，如果所拥有的进程数不同，则用户拥有CPU的比率不同**。这样造成某些用户的响应比过低，不利于公平原则。
  - 解决：
    - 对于不同用户，**基于进程组来分配CPU时间**，其实现思想是对系统中的每个用户赋予某种权值，根据用户权值大小，按比例分配处理机时间。

UNIX中公平分享调度实现

- UNIX基于优先权调度，优先数越大优先权越低。对进程组k中进程j的优先数计算公式如下：
  - $CPU_j(i)=CPU_j(i-1)/2$；进程j在时间段i之前使用的CPU时间累计衰减
  - $GCPU_k(i)=GCPU_k(i-1)/2$；进程组k在时间段i之前使用的CPU时间累计衰减
  - $P_j(i)=Base_j+ CPU_j(i)/2+ GCPU_k(i)/4W_k$；进程j在时间段i的优先数，$Base_j$ 为进程j的基本优先数，W_k为进程组k的权值。

UNIX中公平分享调度实现

- 下例中有用户组1，拥有进程A；用户组2拥有进程B，C，组权重为1/2，即W_k为0.5，Base为60。相应的优先数计算公式为：
  - $CPU_j(i)=CPU_j(i-1)/2$;
  - $GCPU_k(i)=GCPU_k(i-1)/2$;
  - $P_j(i)=60+ CPU_j(i)/2+ GCPU_k(i)/2$

公平分享调度例

	进程A(Group1)			进程B (Group2)			进程C(Group2)		
时间	优先数	计数	组	优先数	计数	组	优先数	计数	组
0	60	0	0	60	0	0	60	0	0
		1	1						
		2	2						
		…	…						
		60	60						
1	90	30	30	60	0	0	60	0	0
					1	1			1
					2	2			2
					…	…			…
					60	60			60

公平分享调度例(续1)

	进程A(Group1)			进程B (Group2)			进程C(Group2)		
时间	优先数	计数	组	优先数	计数	组	优先数	计数	组
2	74	15	15	90	30	30	75	0	30
		16	16						
		17	17						
		…	…						
		75	75						
3	96	37	37	74	15	15	67	0	15
						16		1	16
						17		2	17
						…		…	…
						75		60	75

公平分享调度例(续2)

	进程A(Group1)			进程B (Group2)			进程C(Group2)		
时间	优先数	计数	组	优先数	计数	组	优先数	计数	组
4	78	18	18	81	7	37	93	30	37
		19	19						
		20	20						
		…	…						
		78	78						
5	98	39	39	70	3	18	76	15	18
					4	19			
					…	…			

公平分享调度例(续3)

- 以上A、B、C调度次序为：

  ABACAB……
- 用户组1与用户组2拥有CPU各为50%，故实现基于用户的公平调度。

几种调度算法的比较

	FCFS	SJF	SRTF	RR	HPF	HRRF	MLFQ
选择方式	FCFS	预计最短	剩余最短	常数	优先数	高响应比	多条件
调度模式	非抢占	非抢占	抢占	抢占	抢占、非抢占	非抢占	抢占
吞吐量	——	高	高	时间片小会导致吞吐量低	不强调	高	——
响应时间	可能很慢，特别是进程执行时间差别大时	为短进程提供好的响应时间	提供好的响应时间	为短进程提供好的响应时间	提供好的响应时间	提供好的响应时间	——
开销	小	可能较大	可能较大	小	——	可能较大	可能较大
对进程影响	短进程不利，I/O密集不利	长进程不利	长进程不利	公平	区分对待	平衡	平衡
饥饿	无	可能	可能	无	可能	无	可能

课堂练习

- 1. 设有P1、P2、P3、P4、P5五个进程，按照 P1~P5次序在时刻0同时到达， P1~P5要求运行时间和优先数如图。

进程	运行时间	优先数
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

分别采用FCFS、SJF、优先级调度（非抢先，优先数小优先级高）、RR（时间片为1）四种调度算法，计算各进程的周转时间。

课堂练习

- 2.有一个具有两道作业的批处理系统，采用两级调度处理作业，即，系统首先利用作业调度将作业装入内存并分配资源，后利用进程调度分配CPU。作业调度采用短作业优先的调度算法，进程调度采用以优先数为基础的抢占式调度算法，在下表所示的作业序列，作业优先数即为进程优先数，优先数越小优先级越高。

作业名	到达时间	估计运行时间	优先数
A	10:00	40分	5
B	10:20	30分	3
C	10:30	50分	4
D	10:50	20分	6

(1) 列出所有作业进入内存时间及结束时间。

(2) 计算平均周转时间。

课堂练习

- 3. 进程P1和P2在同一时刻先后进入就绪队列：
  - 进程P1总共需要进行8秒的计算工作，每完成两秒计算后会进行1秒的I/O工作，第4次I/O完成后，P1结束。
  - 进程P2总共进行20秒的计算工作。

请分别分析按照最短剩余时间有限优先调度、时间片轮转调度（时间片长4秒）的进程执行情况，并计算进程周转时间。