操作系统：处理机调度

1. 分级调度
2. 作业调度
3. 进程调度
4. 调度算法
5. 算法评价
6. 实时系统调度算法

处理机调度

处理机调度管理的目的：

是对 CPU 资源进行合理的分配使用，以提高处理机利用率，并使各用户公平地得到处理机资源。这里的主要问题是**处理机调度算法**和调度算法特征分析。

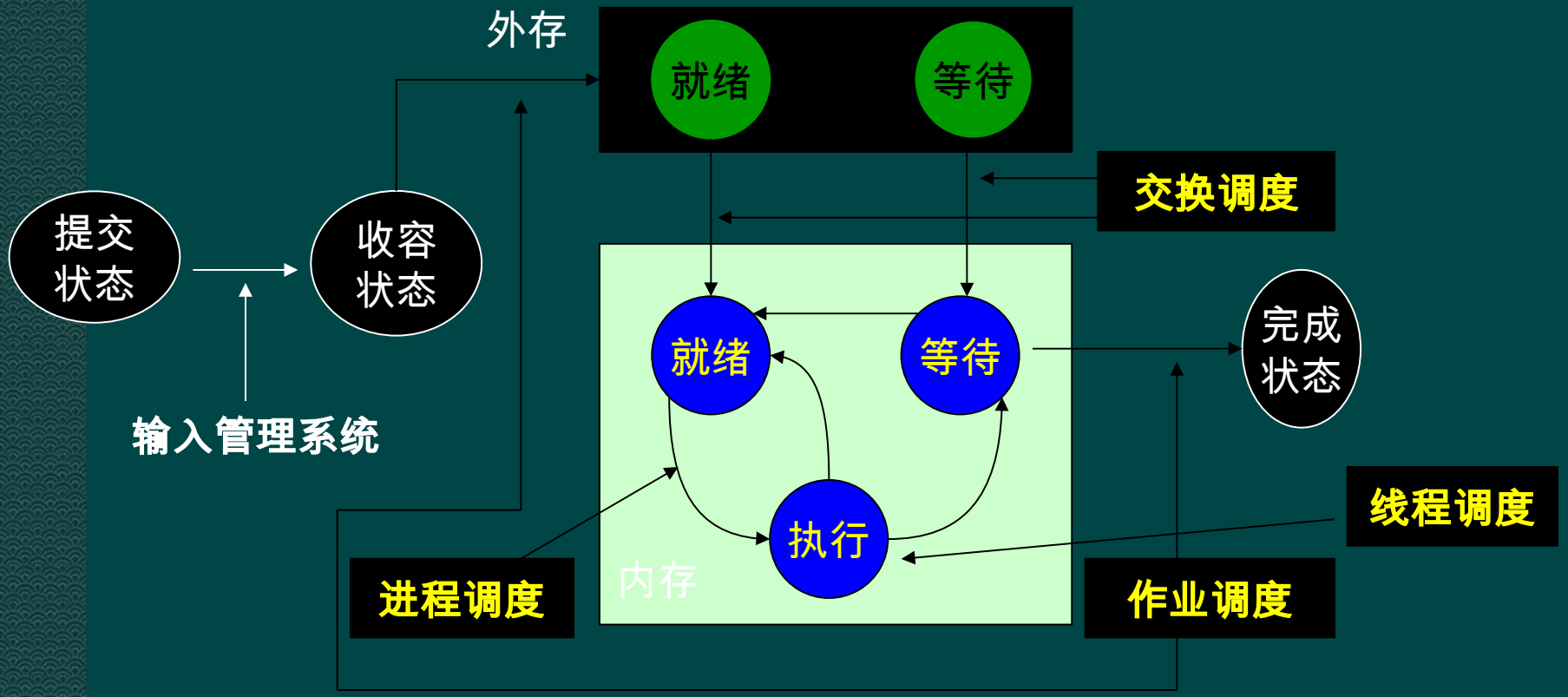
4.1分级调度

- 衡量调度策略的常用指标：
  - **周转时间**：作业提交计算机到返回用户的时间。
  - **吞吐率**：在给定的时间内，计算机系统完成的总工作量。
  - **响应时间**：用户发送指令给计算机到计算机返回结果给用户的时间。
  - **设备利用率**：输入输出设备的使用情况。

4.1分级调度

- **4.1.1 作业的状态及其转换**

一个作业从提交给计算机系统到执行结束退出系统，一般经历**提交、收容、执行和完成**等四个状态。



4.1分级调度

4.1.2调度的层次

- 作业调度：又称为“宏观调度”、“高级调度”。
- 交换调度：又称为“中级调度”。
- 进程调度：又称为“微观调度”、“低级调度”。按照某种策略和方法选取一个处于就绪状态的进程占用处理机。
- 线程调度：进程内调度 -- 多个并发执行线程 .

4.1.3 作业与进程的关系

- 作业是用户向计算机提交的任务实体，例如一次计算、一个控制过程等。反过来，进程则是计算机为了完成用户任务实体而设置的执行实体，是系统分配资源的基本单位。
- 一个作业总是由一个以上的多个进程组成的。

- 作业分解：
  - 首先，系统必须为一个作业创建一个根进程。
  - 然后，在执行作业控制语句时，根据任务要求，系统或根进程为其创建相应的子进程。
  - 为各子进程分配资源并调度子进程执行以完成作业要求的任务。

4.2 作业调度

421 作业调度功能

- 记录系统中各作业的状况；
- 从后备队列中挑选出一部分作业没入执行；
- 为被选中作业做好执行前的准备工作；
- 在作业执行结束时做善后处理工作。

422作业调度目标：

- 对所有作业应该是公平合理的；
- 应使设备有高的利用率；
- 每天执行尽可能多的作业；
- 有快的响应时间。

4.2 作业调度

4.2.3 作业性能衡量

1. 周转时间

作业i的周转时间 Ti 为　　Ti=Tei-Tsi

其中 Tei 为作业的完成时间， Tsi 为作业提交的时间。

一个作业的周转时间说明了该作业在系统内停留的时间，包含两部分：**等待时间** 和 **执行时间** ，即：

Ti=Twi+Tri

这里， Twi 指作业由后备状态到执行状态的等待时间。

2. 带权周转时间

带权周转时间是作业周转时间与作业执行时间的比：

Wi=Ti/Tri

注意：对于分时系统，仅仅用周转时间或带权周转时间来衡量调度是不够的，还应保证有用户能够容忍的响应时间。

4.3 进程调度

4.3.1 进程调度功能

- 记录所有进程的运行状况（静态和动态）；
- 当进程出让 CPU 或调度程序剥夺执行状态进程占用的 CPU 时，选择适当的进程分派 CPU .

- 完成进程上下文切换
  - 检查是否可以进行上下文切换 (时机);
  - 保留被切换出 CPU 的进程上下文 (准备);
  - 装配被选择运行进程的上下文 (切换).

- **4.3.2 进程调度的时机**
  - **引起进程调度的原因有以下几类 ：**
    - 进程**执行完毕**；
    - 进程自己调用**阻塞原语**进入睡眠状态；
    - 进程调用了 **P 原语操作**，因资源不足而被阻塞或调用了 **v 原语操作**激活了等待资源的进程队列；
    - 进程提出 **I/O 请求后被阻塞**；
    - **时间片**已经用完；
    - 系统**调用返回**用户进程时，可看作系统进程执行完毕，从而可调度一新的用户进程执行。

  当 CPU 执行方式是可剥夺时，还有：

    - 就绪队列中的某进程的**优先级变得高于当前进程**的优先级，也将引发进程调度。

4.3 进程调度

5.3.4 进程调度性能评价

- **进程调度性能衡量方法分为：定形和定量两种**
  - 定形衡量
    - **调度的可靠性；**
    - **调度的简洁性 .**
  - 定量评价
    - **CPU 的利用率评价、进程在就绪队列中的等待时间与执行时间之比等 ；**
    - **模拟或测试系统响应时间方法来评估 .**

4.4 调度算法

- 调度算法种类：
  - **先来先服务**
  - **轮转算法**
  - **多级反馈轮转算法**
  - **优先级法**
  - **最短作业优先法**
  - **最高响应比优先法**

4.4 调度算法（ FCFS ）

◈ 先来先服务（ FCFS, First Come First Serve ）

最简单的调度算法，按先后顺序进行调度

算法：

- 按照作业提交或进程变为**就绪状态的先后次序，分派 CPU**
- 当前作业或进程占用 CPU ，直到**执行完或阻塞，才出让 CPU**（非抢占方式）。
- 在作业或进程唤醒后（如 I/O 完成），并不立即恢复执行，通常等到当前作业或**进程出让 CPU** 。

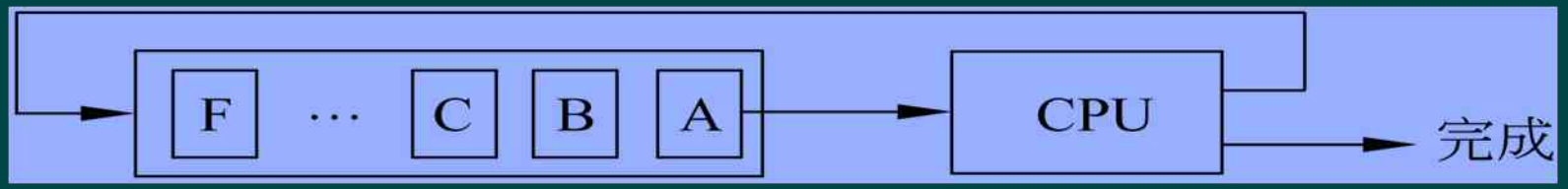
特点：

- 比较**有利于长作业**，而不利于短作业。
- **有利于 CPU 繁忙的作业**，而不利于 I/O 繁忙的作业。

4.4 调度算法（轮转法）

- 轮转法：让每个进程在就绪队列中等待时间与享受服务时间成正比例
  - 将系统中所有的就绪进程按照 FCFS 原则，排成一个队列。
  - 每次调度时将 CPU 分派给队首进程，让其执行一个时间片。时间片的长度从几个 ms 到几百 ms 。
  - 在一个时间片结束时，发生时钟中断。调度程序据此暂停当前进程的执行，将其送到就绪队列的末尾，并通过上下文切换执行当前的队首进程。
  - 进程可以未使用完一个时间片，就出让 CPU （如阻塞）。

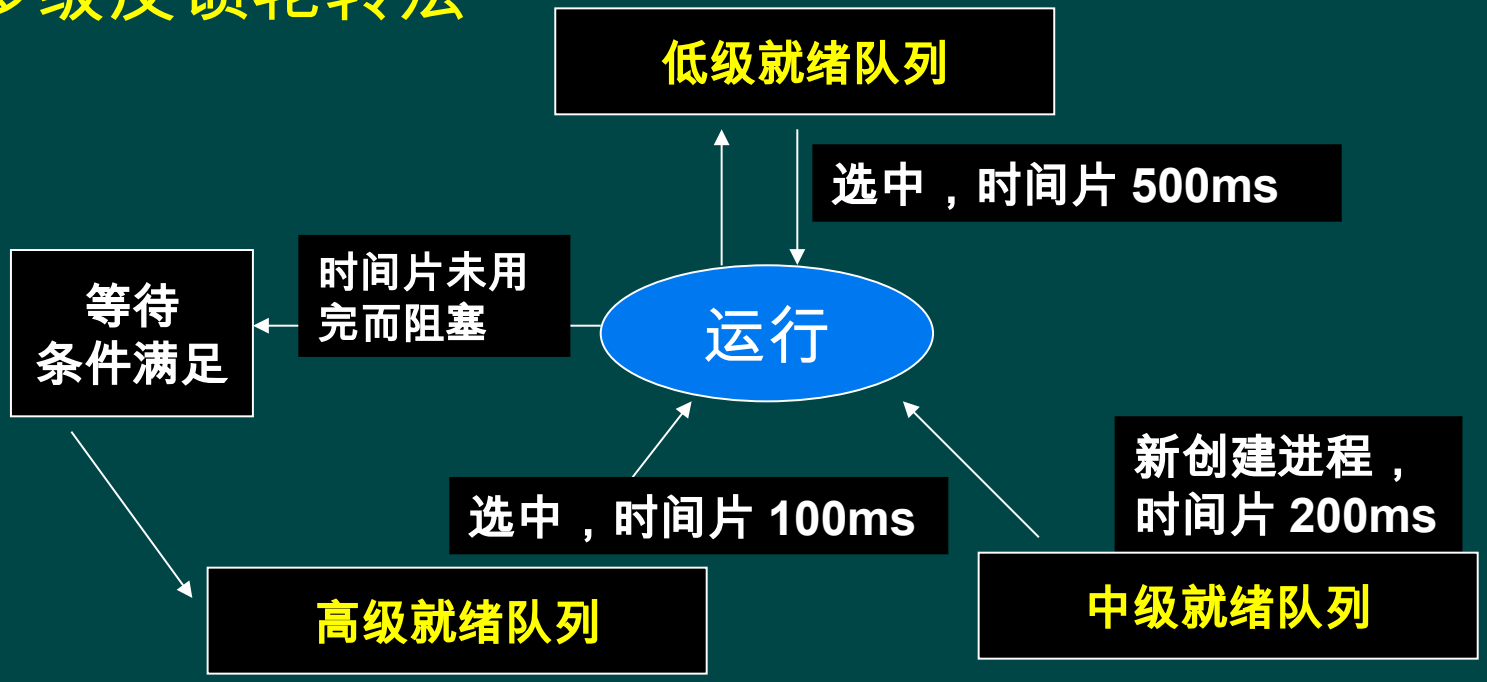
4.4 调度算法（轮转法）



- 时间片长度变化的影响
  - 过长 - >退化为 FCFS 算法，进程在 1 个时片内执行完。
  - 过短 - >用户的一次请求需要多个时间片才能处理完，上下文切换次数增加，加重系统开销。
- 对响应时间要求：T(响应时间)=N(进程数目)*q(时间片)
- 就绪进程的数目：数目越多，时间片越小。
- 系统的处理能力：应当使用户输入通常在一个时间片内能处理完，否则使响应时间、平均周转时间和平均带权周转时间延长。

4.4 调度算法（多级反馈轮转法）

- 多级反馈轮转法



- 同级就绪队列按轮转算法
- 先服务高级队列，高级队列空再服务低级别队列

4.4 调度算法（优先级法）

- 优先级法
  - 系统或用户按某种原则为作业或进程指定一个优先级来表示作业或进程享有的调度优先。
    - 静态优先级：优先级在生存周期中不变。
      - 作业调度优先级（按作业级别或缴费多少）；
      - 进程调度优先级（根据进程类型： I/O , CPU, 均衡型等）.
    - 动态优先级：优先级随着作业或进程的执行过程不断的变化。
      - 根据进程占用 CPU 时间的长短来决定；
      - 根据就绪进程等待 CPU 的时间长短来决定。

        优先级 P=a*t (t 为等待时间）

4.4 调度算法（最短作业优先法）

- 最短作业优先法（ Shortest Job First ）
  - **算法：**
    - **对预计执行时间短的作业（进程）优先分派处理机。通常后来的短作业不抢先正在执行的作业。**
  - **优点：**
    - **比 FCFS 改善平均周转时间和平均带权周转时间，缩短作业的等待时间；**
    - **提高系统的吞吐量；**
  - **缺点：**
    - **对长作业非常不利，可能长时间得不到执行**
    - **未能依据作业的紧迫程度来划分执行的优先级；**
    - **难以准确估计作业（进程）的执行时间，从而影响调度性能。**

4.4 调度算法（6）

- 最高响应比优先法（ HRN ）
  - 最高响应比优先法（ Highest Response_ratio Next ）调度策略同时考虑每个作业的等待时间长短和估计需要的执行时间长短，从中选出响应比最高的作业投入执行 。
  - 响应比 R 定义如下： $R = (W+T)/T = 1+W/T$

    T: 作业估计需要的执行时间

    W: 作业就绪队列中等待时间
  - HRN 算法是 FCFS 和 SJF 之间的折中算法。

4.5 算法评价

4.5.1 FCFS 方式的调度性能分析

设处理机或系统资源为服务器，一个进程为享受该服务器服务的顾客。当这些顾客按 FCFS 方式排队享受服务的系统模型如图 4.7 。

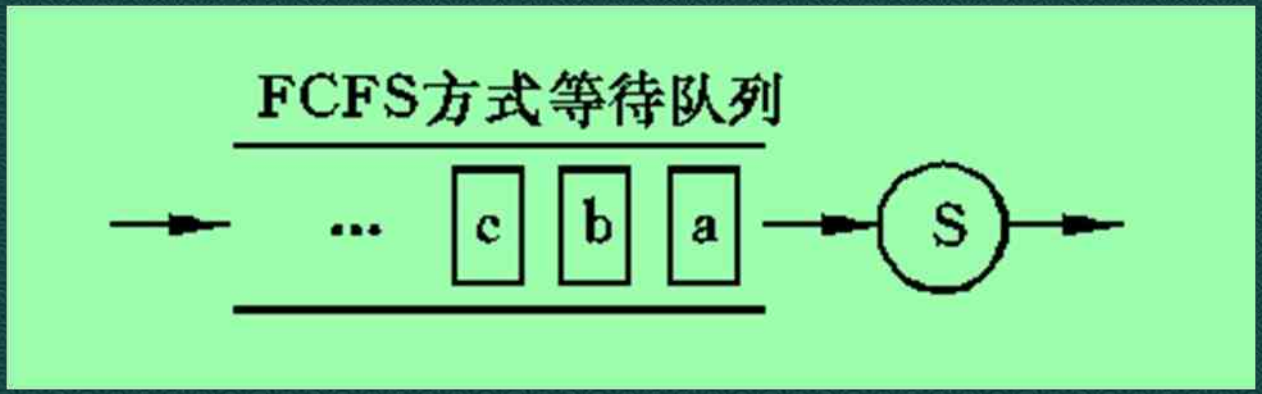


图 4.7 FCFS 方式的评价模型

这里，假定该系统模型中只有一个服务器S。

设新顾客到达等待队列的时间与系统的当前状态、以前的顾客到达时间都无关，也就是新顾客到达系统的时间是服从泊松分布的。设λ为到达率，则在单位时间内x个顾客到达的概率为：

$$P(x)=e^{-\lambda}\lambda^{x}/x!$$

单位时间内顾客到达的期望值，即算术平均值为：

$$E(x)=\sum_{x=0}^{\infty}xP(x)=e^{-\lambda}\sum_{x=1}^{\infty}\lambda^{x}/(x-1)!=\lambda e^{-\lambda}\sum_{y=0}^{\infty}\lambda^{x}/y!$$

（ y=x-1 ）

由于

$$\sum_{x=0}^{\infty}\lambda^{x}/x!=e^{\lambda}$$

所以，E(x) = λ

也即单位时间内顾客到达的平均值等于其到达率。

设服务器S为顾客提供服务的概率也服从泊松分布，且μ为服务率，则单位时间内x 个顾客被服务的概率是

$$P(x)=e^{-\mu}\mu^{x}/x!$$

同理，单位时间内被服务顾客个数的算术平均值等于其服务率μ。

将单位时间换成任意时间t ，可得到在已知时间t内x 个顾客到达的概率为：

$$P(x(t))=e^{\lambda t}(\lambda t)^{x}/x!$$

在t时间内，一个顾客也不到达的概率为：

$$P(0)=e\text{-}\lambda t$$

从而，t时间内至少到达一个顾客的概率为：

$$P(x(t)>0)=1\text{-}e\text{-}\lambda t$$

如果把时间 t 换成固定的时间间隔 τ ，则在任何时间间隔 τ 内至少有一个到达发生的概率仍为 1-e-λt 。利用马尔可夫性质，可以简化顾客和服务的排队模型。

由于服务器的服务概率也服从泊松分布，也可推出它在 τ 时间间隔内至少为一个顾客服务的概率为 1-e-μt 。服务器的服务特性也是满足马尔可夫性质的。

由于　　$P(x(t)>0)=1-e^{-\lambda t}$

其密度函数　　$P(x(t)>0)=\lambda e^{-\lambda t}$

t 的期望值等于

$E(t)\int_{\infty}^{0} t\lambda e^{-\lambda t}dt=-te^{-\lambda t}|_{\infty}^{0}+\int_{\infty}^{0} e^{-\lambda t}=1/\lambda$

即两个连续到达的顾客之间的平均时间间隔为 1 / λ 。同理，可得服务器服务时间的平均值为 1 / μ 。显然，只有当　1 / μ < 1 / λ　，也就是　λ < μ 时，系统才是稳定的。否则，等待服务队列将无限增长。

设 S i 为系统的一个状态，表示等待服务的等待队列中有 i-1 个顾客，服务器中有 1 个顾客存在。由概率密度函数可知，在 dt 时间内 1 个新顾客到达的概率是：

P(dt 时间内 1 个顾客到达) = λe-λdtdt

将上式做多项式展开得： λe-λdtdt = λdt + O(dt2)

同理可得，在 dt 时间内服务器为 1 个顾客提供服务的概率是：

P(dt 时间内 1 个顾客离开) = μdt + O(dt2)

省略以上两式的二次方项，在 dt 时间内 1 个顾客到达或离开的概率为：

P(dt 时间内 1 个顾客到达)=λdt

P(dt 时间内 1 个顾客离开)=μdt

对于 i = 0 时，有： P(dt 时间内 1 个顾客离开) = 0

在 dt 时间内，顾客一个也不到达和顾客一个也不离开的概率为：

1-P(dt 时间内 1 个顾客到达) -P(dt 时间内 1 个顾客离开)
-P(dt 时间内 2 个以上顾客到达)-P(dt 时间内 2 个以上顾客离开)
。

由于上式的后两项实际上是 dt 的二次方以上的函数，从而上式可合并为：

P(dt 时间内不发生变化) = 1-(λ + μ) dt-O(dt2)(i > 0)

或 P(dt 时间内不发生变化) = 1-λdt-O(dt2) (i = 0)

忽略二次方项，可得状态变化图如图 4.8。

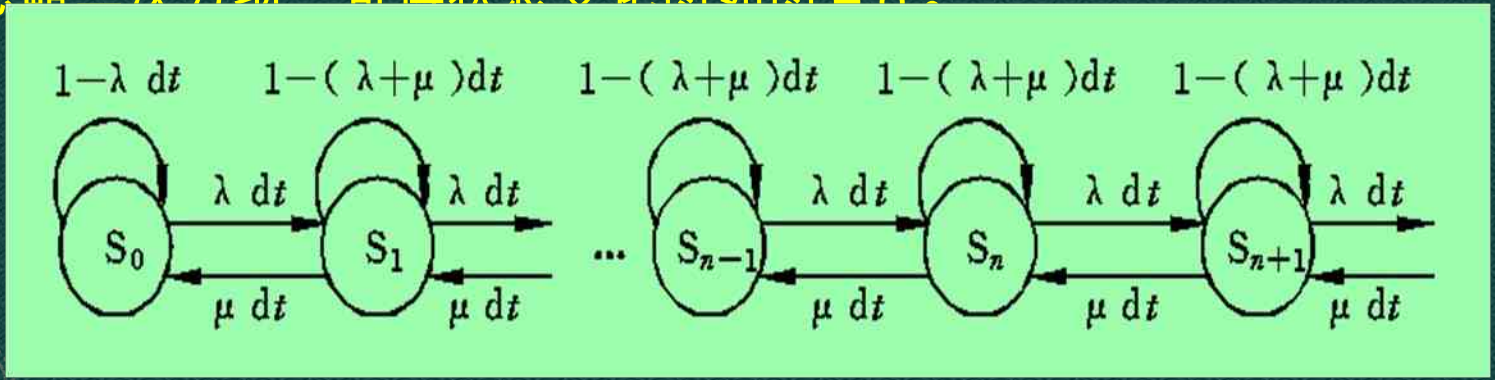


图 4.8 状态转换图

设系统在 t+dt 时间内处于状态 S i 的概率为 Pi(t+dt) ，则由状态转换图有：

$$Pi(t + dt) = [\ 1-(\lambda + \mu)\ dt\ ]\ Pi(t) + \lambda dt\ Pi-1(t) + \mu dt\ Pi+1(t)$$

式中 i≥1 。对于 i = 0 时，有：

$$P0(t + dt) = (1-\lambda dt)P0(t) + \mu dt\ P1(t)$$

当系统到达稳定状态时，上述概率将会趋于一个常量， Pi(t + dt) 的导数为 0 ，即：

$$Pi'(t + dt) = -(\lambda + \mu)Pi(t) + \lambda Pi-1\ (t) + \mu Pi+1\ (t) = 0$$

$$P0'(t + dt) = -\lambda P0(t) + \mu P1(t) = 0$$

即： $P1 = (\lambda / \mu)\ P0(t)$

$$(\lambda + \mu)\ Pi(t) = \lambda Pi-1\ (t) + \mu Pi+1\ (t)$$

令 $\lambda / \mu = \rho$ ，采用代入法可得：

$$Pi(t) = \rho i P0(t)$$

另外，系统运行中的任一时刻总是处于图 4.8 所示状态图中的任一状态中，从而有：

$$\sum_{i=0}^{\infty} P_i = 1$$

由此可得：

$$\sum_{i=0}^{\infty} \rho^i P_0(t) = 1$$

由于在 ρ < 1 时有

$$\sum_{i=0}^{\infty} \rho^i = 1/(1-\rho)$$

从而有： P0(t) = 1-ρ

即稳态条件下，系统内无顾客概率为 1-ρ = (μ-λ) / μ，而系统内有顾客的概率为 λ / μ。系统内顾客的算术平均值是：

$$n = E(i) = \sum_{i=0}^{\infty} iP_i(t) = \sum_{i=0}^{\infty} (1-\rho)^i \rho^i = \rho(1-\rho)$$

把上述按 FCFS 方式排列和调度，并只有一个服务器的系统称为 M/M/1 系统。第一个字母 M 表示顾客到达时间间隔是指数分布，具有马尔可夫性质，第二个字母 M 表示从服务器离开的顾客的时间间隔服从指数分布，具有马尔可夫性质，第三个字母 1 表示只有一个服务器。

设响应时间 R 为从顾客到达等待队列后开始到离开服务器的时间，则在系统进入稳定状态之后，系统中的顾客数 n 和平均响应时间之间存在一个非常简单的关系，即 $n=\lambda R$ ，λ 为顾客的平均到达率。该公式称为 Little 结果（ Little's result ），是一个在系统分析中广泛使用的公式。

利用 Little 结果可求出 M/M/1 系统的平均响应时间：

$R = n / \lambda = \rho / \lambda(1 / (1-\rho)) = 1 / (\mu(1-\rho))$

平均响应时间和 ρ 的关系图如图 4.9 。

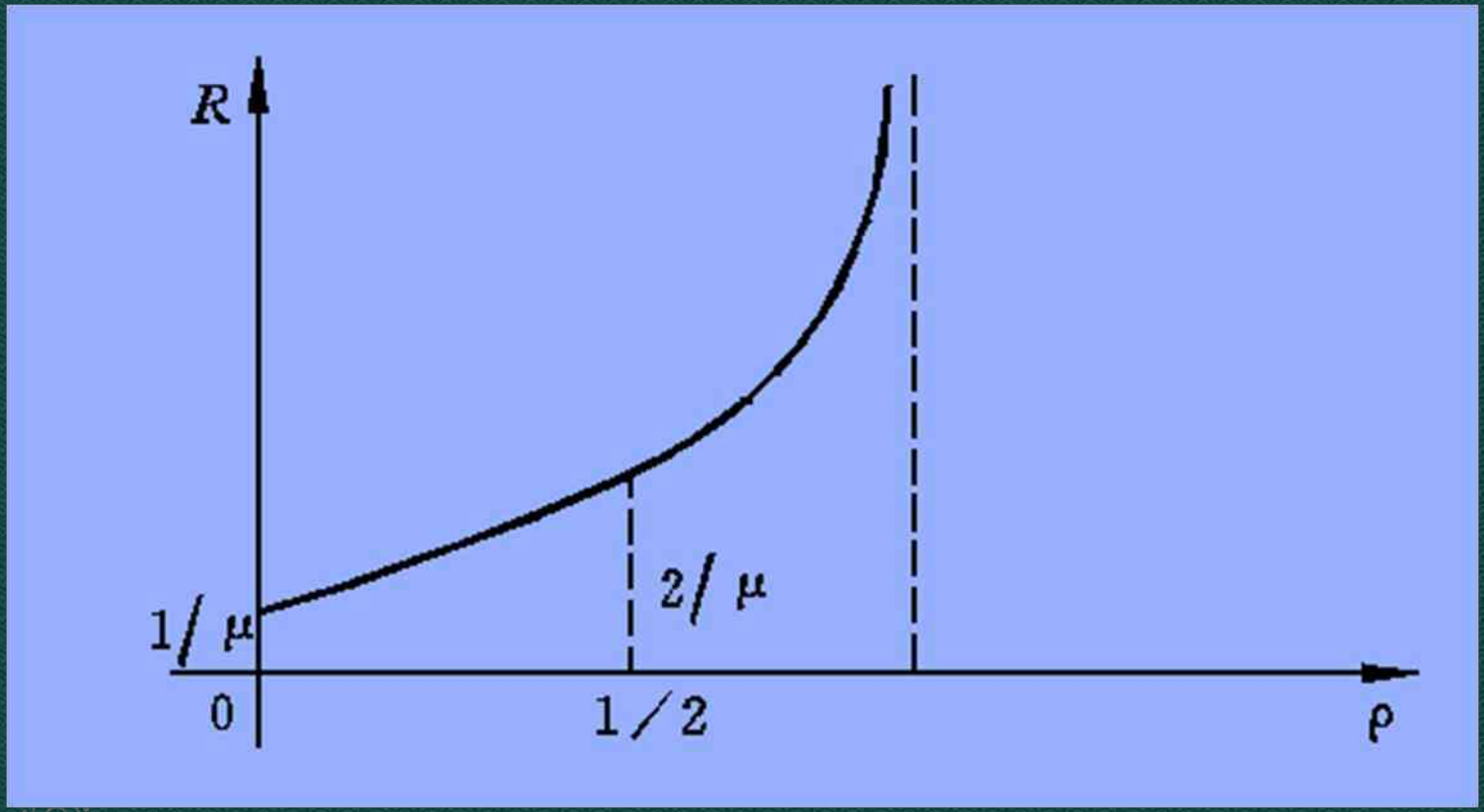


图 4.9

由图 4.9 可以看出， M/M/1 系统的服务性能是由 ρ 决定的。如果 ρ 趋近 1 ，即 λ 趋近 μ ，则响应时间急剧增大，系统性能变差。而当 ρ 小于 1/2 时，等待队列中为空的可能性较大，因为平均响应时间小于平均服务时间的 2 倍。

下面再来看看 M/M/1 系统对短作业或短进程的影响。

设短作业的到达率和服务率分别为（ λ1 ， μ1 ），长作业的到达率和服务率为（ λ2 ， μ2 ），且二者都服从泊松分布。二者的合成仍是泊松过程，其到达率 λ = λ1 + λ2 ，平均服务时间为：

1 / μ = λ1 / (λ1 + λ2)*1 / μ1 + λ2 / (λ1 + λ2)*1 / μ2 = 1 / (λ1 + λ2)*(λ1 / μ1 + λ2 / μ2)

从而有：

$$\lambda / \mu = \lambda 1 / \mu 1 + \lambda 2 / \mu 2 = \rho 1 + \rho 2$$

短作业的服务时间 $1 / \mu 1$ 远小于长作业的服务时间 $1 / \mu 2$。FCFS 调度策略时有响应时间 R 为：

$$R = 1 / \lambda^{*}\rho / (1-\rho)$$

其中，$\lambda = \lambda 1 + \lambda 2$，$\rho = \rho 1 + \rho 2$

由于 λ 和 ρ 中包含了 λ1，λ2，μ1 和 μ2，所有作业的平均响应时间相同，从而，短作业在系统中的驻留平均时间与长作业的驻留平均时间相同，这对短作业是不利的。

4.5.2 轮转法调度性能评价

设时间片为 q ，服务时间平均值为 1 / μ ，每个顾客平均需要 k 个时间片。从而有， 1 / μ = k*q 。如果某个顾客刚好需要 k 个时间片的服务时间，则这个顾客就会到达等待队列 k 次（ k > 1 ）。

对于短作业， 1 / μ1 = k1*q ，而对于长作业， 1 / μ2 = k2*q 。设新顾客的到达率等于 λ = λ1 + λ2 ，服务时间：

$$1 / \mu = (\lambda 1 / \mu)\, k1*q + (\lambda 2 / \mu)\, k2*q$$
$$= (1 / \lambda)(\lambda 1 / \mu 1 + \lambda 2 / \mu 2)$$

从而有： λ / μ = ρ = ρ1 + ρ2

这看上去好像在平均响应时间方面，轮转法和FCFS 调度方式上变化不大。但事实上，在等待队列中享受过 k 次时间片服务的顾客的响应时间是：

$$R(k) = \rho / (\lambda(1-\rho))$$

$$= 1 / (\mu(1-\rho)) = k*q / (1-\rho)$$

也就是说，响应时间与服务时间成正比。从而，所需服务时间短的顾客的响应时间将会小于所需服务时间长的顾客的响应时间。因此，轮转法在响应时间上要优于 FCFS 调度方式。

4.5.3 线性优先级法的调度性能

线性优先级调度策略（ SRR ）是介于 FCFS 方式和轮转法之间的一种调度策略。 SRR 方式把新到达的顾客首先送入等待室休息一段时间后，再送到等待服务队列。（如图 4.10 ）

设顾客到达等待室的到达率为 λ ，到达等待队列的到达率为 λ′ 。 λ′ 依赖于等待室的到达率 λ 和线性参数 r ，其中 r = 1-b / a 。由 4.4 节可知，有 a > b ，且 a 和 b 分别为等待室内顾客和等待队列中顾客优先级的线性增加系数。

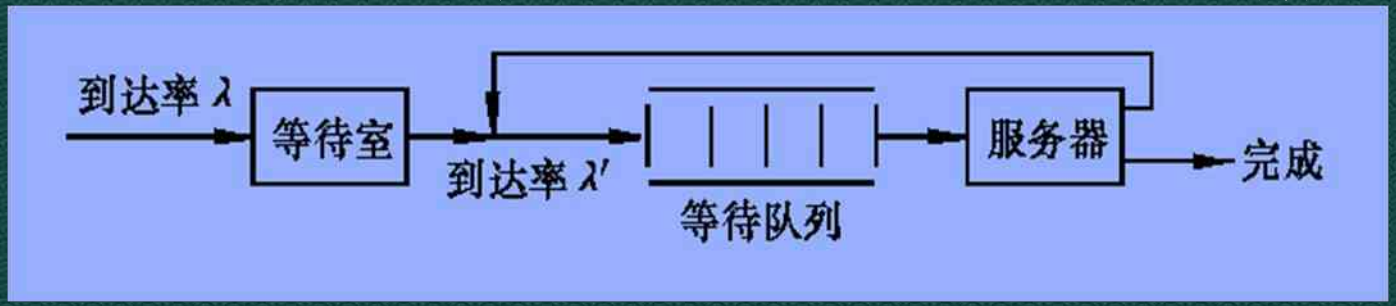


图 4.10 线性优先级调度的评价模型

设 t1 和 t2 分别为两顾客接连到达等待室的时间，则有：

$$1 / \lambda = t2 - t1$$

设 t 1′和 t 2′分别为两顾客从等待室接连到达等待队列的时间，则有： $1 / \lambda' = t2' - t1'$

由图 4.6 可知，有： $(t2-t1) / (t2'-t1') = r$

即： $\lambda' / \lambda = r$ ， $\lambda' = r\lambda$

由于 r < 1 ，所以等待时以 r 的比率滞留到达的顾客。

线性优先级调度系统的响应时间是R s*r 。且R s*r 由等待时的平均等待时间R d 和进入等待队列后得到服务的平均响应时间R s 之和组成。由对轮转法的分析，则有： $R s = 1 / (\mu-\lambda)$

$$r* R s = 1 / (\mu-\lambda)$$

从而：$R_d = r^* R_s - R_s = 1/(\mu-\lambda) - 1/(\mu-\lambda)$

$= (\lambda-\lambda)/(\mu-\lambda)(\mu-\lambda)$

另外，由轮转法可知：$R_{s(k)} = kq/(1-\rho)$

其中，$\rho = \lambda/\mu$。从而有：

$R_{sr(k)} = R_d + R_{s(k)}$

$= (\lambda-\lambda)/(\mu-\lambda)(\mu-\lambda) + kq/(1-\rho)$

$= 1/(\mu-\lambda) - (1-kq\mu)/(\mu-\lambda)$

其中 k 是一个顾客享受服务的时间片的次数，q 是时间片常数。

FCFS、SRR 以及轮转法三种调度平均响应时间：

FCFS：$R_{fc(k)} = 1/(\mu-\lambda)$

轮转法：$R_{rr(k)} = kq\mu/(\mu-\lambda)$

SRR：$R_{sr(k)} = 1/(\mu-\lambda) - (1-kq\mu)/(\mu-\lambda)$

如果 kq = 1 / μ ，即顾客的服务时间由其平均服务时间相等的话，则R sr(k) 中第二项变为 0 ，上述R fc(k) = R rr(k) = R sr(k) 。当然，对于需要服务时间短的顾客来说，有 kq < 1 / μ ，而对于需要服务时间长的顾客来说，则有 kq > 1 / μ 。从而，对于服务时间短的顾客其响应时间：

R rr < R sr < R fc

而对于服务时间长的顾客来说，其响应时间则为：

R fc < R sr < R rr

上面只是从响应时间的角度对几种常见的调度策略进行了评价分析。除了响应时间之外， CPU 利用率也是评价调度性能的另一个标准。

4.5 算法评价

μ: **服务率**（单位时间里 x 个顾客被服务的概率）
λ：**到达率**（单位时间里 x 个顾客到达的概率）

- FCFS 平均响应时间：

  $R_{fc}(\kappa) = 1/(\mu-\lambda)$

- 轮转法平均响应时间：

  $R_{rr}(\kappa) = \kappa q\mu/(\mu-\lambda)$

- SRR 平均响应时间：

  $R_{sr}(\kappa) = 1/(\mu-\lambda) - (1-\kappa q\mu)/(\mu-\lambda)$

- 对于服务时间短顾客：

  $R_{rr} < R_{sr} < R_{fc}$

- 对于服务时间长顾客：

  $R_{fc} < R_{sr} < R_{rr}$

κ: 平均每个顾客的作业需要的时间片个数（完成）
q：时间片长度
Rfc:FCFS 平均响应时间
Rsr: 线性优先级法平均响应时间
Rrr: 轮转法平均响应时间

4.6 实时系统调度方法

4.6.1 实时系统特点

- 把给定的任务，按所**要求的时限**调配到相应的设备上去处理完成
- 实时系统中处理外部事件分为：
  - 硬实时任务（时限绝对满足）
  - 软实时任务（时限略微可延迟）
- 实时操作系统基本特点：
  - 有限等待时间
  - 有限响应时间
  - 用户控制
  - 可靠性高
  - 系统出错处理能力强

4.6 实时系统调度方法

4.6.1 实时系统的特点

- 实时系统要求实时操作系统具有下述能力：
  - 很快的进程或线程**切换速度**
    - 进程或线程切换速度是实时系统设计的核心。
  - 快速的**外部中断响应**能力
  - 基于优先级的**抢先式**调度策略
    - 基于优先级的**固定点抢先式调度策略**；
    - 基于优先级的**随时抢先式调度策略**。

4.6 实时系统调度方法

4.6.2 实时调度算法的分类

- 静态表格驱动类
  - 最早时限优先法：调度时限最早的任务执行。
- 静态优先级驱动抢先式调度算法类
  - 频率单调调度算法：静态优先级抢占式调度算法。
- 动态计划调度算法类

  执行前排出计划，根据中间状态随时做调整。
- 尽力而为调度算法类

  根据经验给任务指定优先级。

4.6 实时系统调度方法

- 4.6.3 时限调度算法与频率单调调度算法
  - **时限调度算法**
    - 按用户的**时限要求顺序设置优先级**，优先级高者占据处理机。该算法需要的输入信息：
      - **任务就绪时间或事件到达时间**
      - **开始时限**
      - **完成时限**
      - **处理时间**
      - **资源需求**
      - **优先级**
  - **频率单调调度算法**
    - 基本原理是频率越低（**周期越长**）的任务的优先级越低。

下面是用时限调度算法调度周期性实时任务的例子：

设实时系统从两个不同的数据源 DA 和 D B 周期性地收集数据并进行处理，其中 DA 的时限要求以 30 ms 为周期， DB 的时限要求以 75 ms 为周期。设 DA 所需处理时限为 15 ms ， DB 所需处理时限为 38 ms ，则与 DA 和 DB 有关进程的事件发生时限（就绪时限）、执行时限以及结束时限如图 4.11 所示。

进 程	事件发生时限	执行时限	结束时限
DA(1)	0	15	30
DA(2)	30	15	60
DA(3)	60	15	90
…	…	…	…
DB(1)	0	38	75
DB(2)	75	38	150
DB(3)	150	38	225
…	…	…	…

图 4.11 周期性任务的预计发生、执行与结束时限

如果使用时限调度算法，并按最近结束时限优先级最高的方法进行排列，可以给出图 4.11 所示各进程的调度顺序和相对时间。（如图 4.12 ）

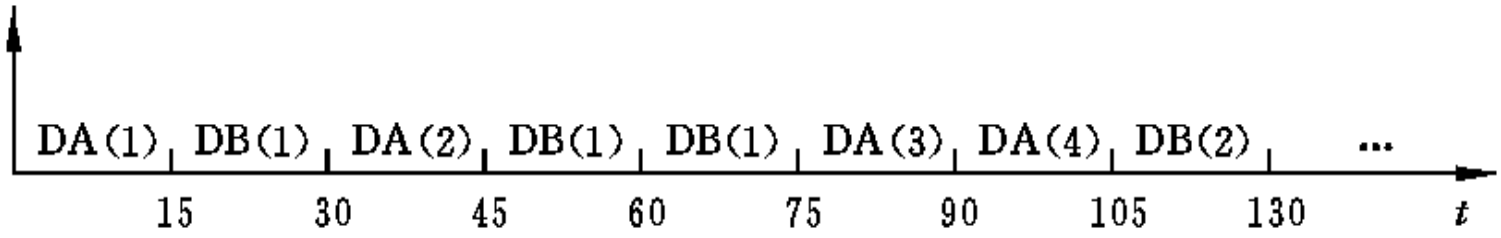


图 4.12 时限调度算法给出的调度顺序

如图 4.12 所示，在开始时，进程 DA（ 1 ）和 DB（ 1 ）的结束时限比较， DA（ 1 ）的结束时限最近，从而调度进程 DA（ 1 ）执行。 DA（ 1 ）的实际结束时间为 15 ，小于 30 的时限要求。紧接着，进程 DB（ 1 ）被调度执行。在执行到时间为 30ms 时，进程 DA（ 2 ）进入就绪状态。由于 DA（ 2 ）的结束时限为 60 ，近于 DB（ 1 ）的结束时限 75 ，从而 DB（ 1 ）被 DA（ 2 ）抢先。 DA（ 2 ）的实际结束时间为 45 ，小于要求时限 60 。在 DA（ 2 ）结束之后， DB（ 1 ）再次占有处理机继续执行，当 DB（ 1 ）执行到时间为 60ms 时，进程 DA（ 3 ）进入就绪状态。但是，由于 DA（ 3 ）的结束时限为 90 ，远于 DB（ 1 ）的结束时限 75 ，从而 DB（ 1 ）继续执行。

时限调度算法也可以用于非周期性任务调度。频率单调调度算法是一种被广泛用于多周期性实时处理的调度算法。

频率单调调度算法的基本原理是频率越低（周期越长）的任务的优先级越低。

设任务周期为 T ，任务的执行时间为 C ，则使用频率单调调度算法的必要条件是 C≤ T 。

已经证明，对于 n(n≥1) 个周期的不同任务来说，设每个周期为 T_i ，其相应任务的执行时间为 C_i ，则使用频率单调调度算法的充分条件是

$$\frac{C_1}{T_1}+\frac{C_2}{T_2}+\cdots+\frac{C_i}{T_i}+\cdots+\frac{C_n}{T_n}\le n(2^{\frac{1}{n}}-1)$$

例如，对于由 3 个周期组成的实时任务序列来说，其执行时间与周期之比应是：

$$\frac{C_1}{T_1}+\frac{C_2}{T_2}+\frac{C_3}{T_3}\le 3(2^{\frac{1}{3}}-1)=0.799$$

如果进程执行时间与周期比之和大于 $n(2^{1/n}-1)$ ，则用户所要求的时限无法保证。

小 结

- 本章以 CPU 管理为核心，讨论管理、控制用户进程执行的方法。主要包括：
  - 作业与进程的关系
  - 作业调度策略与算法
  - 进程调度策略与算法
  - 实时调度系统

比较常用的有 RR(轮转) 法、 FCFS(先来先服务) 法、优先级法和 SRR(线性优先级) 法等。

轮转法主要用于分时系统，它具有较好的响应时间，且对每个进程来说都具有较好的公平性。

FCFS 法不利于执行时间短的进程。

SRR 法是介于 FCFS 法和 RR 法之间的一种进程调度方法。