第三章 处理机调度与死锁

一、教学目的要求：

1．掌握多级调度的层次
2．掌握作业调度的功能
3．掌握进程调度的功能
4．掌握几种常用的进程和作业调度算法
5．理解 Belady 相象
6．了解调度算法评价和实时系统调度算法
7．了解 UNIX 常用调度命令
8．掌握死锁的概念
9．掌握银行家算法

二、内容分析：

1．概述：

1）各级调度之间的关系，以及每级调度的功能

2）通过实例来讲调度算法，并分析每种调度算法的特点

2．教学重点：

1）分级调度的层次

2）作业调度的功能与实现

3）进程调度的功能与实现

4）调度算法

5）死锁

6）银行家算法

3．教学难点：

1）调度算法

2）银行家算法

作业和进程的关系

1. 作业是向计算机提交的任务实体；进程是执行任务的执行实体
2. 作业对应一个或多个进程，反过来不成立
3. 进程出现在作业的运行状态

分级调度

一般来说，作业从进入系统到最后完成，可能要经历三级调度：高级调度、中级调度和低级调度。

（1）高级调度：又称作业调度。其主要功能是根据一定算法，从输入的一批作业中选出若干个作业，分配必要的资源，如内存、外设等。

（2）中级调度：为了使内存中同时存放进程数目不至于太多，有时需要把某些进程从内存中移到外存上，以减少多道程序的数目，为此设立了中级调度，特别在采用虚拟存储技术的系统或分时系统中，往往增加中级调度这一级。

（3）低级调度：又称进程调度，其主要功能是根据一定的算法将 CPU 分派给就绪队列中的一个进程，进程调度是操作系统中最基本的一种调度。

作业调度

1. 作业控制块（JCB），它记录作业的有关信息。

2. 作业调度的功能

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换。

进程调度

1．进程调度的功能和时机

进程调度的主要功能是：①保护现场→②挑选进程→③恢复现场。

进程调度时机：任务完成后；等待资源时；运行到时了；发现重调标志。

2．两级调度模型

作业调度和进程调度是 CPU 主要的两级调度，二者关系如下图示：

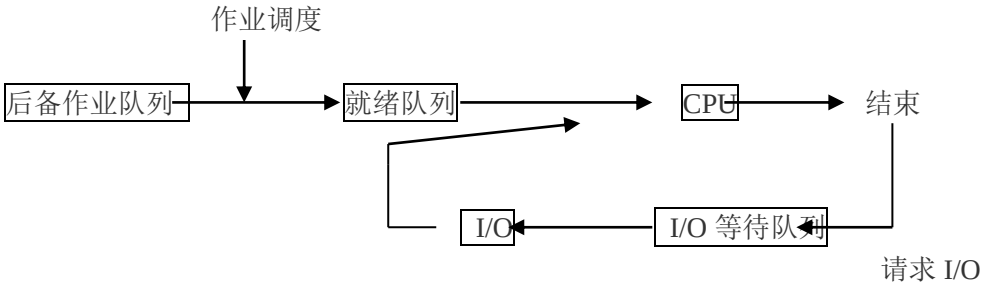


图 3-1　两级调度简化队列图

从图中可以看出，作业调度是宏观调度，而进程调度是微观调度。

3．作业调度和进程调度的区别：

（1）作业调度为进程活动做准备，进程调度使进程活动起来；

（2）作业调度次数少，进程调度频率高；

（3）有的系统不设作业调度，但进程调度不可少。

调度性能的评价

1．选择调度算法时应考虑的主要因素（作一般了解）

2．调度性能评价准则

常用的评价准则包括：

（1）CPU 利用率: 一般 CPU 的利用率在 40%-90%；

（2）吞吐量：表示在单位时间内 CPU 完成作业的数量；

（3）周转时间：从作业提交到作业完成的时间间隔。要求掌握作业的周转时间、平均周转时间、平均带权周转时间的概念并能进行计算；

（4）绪等待时间：作业在就绪队列中所花费的时间；

（5）响应时间：从提交第一个请求到产生第一个响应所用的时间。

调度算法

1．先来先服务（FCFS）法

它的实现思想就是“排队买票”的办法，是最简单的一种调度算法。

2．时间片轮转法

系统把所有就绪进程按先入先出的原则排成一个队列。新来的进程加到就绪队列末尾，每当执行进程调度时，进程调度程序总是选出就绪队列的队首进程，让它在 CPU 上运行一个时间片的时间。时间片是一个小的时间单位。

3．优先服务

先办就是优先处理，表明急事、重要的事，有最高的优先级。

一般有两种不同的处理方式:

（1）非抢占式优先级法，即：“你打完电话，他再打电话。”

（2）抢占式优先级法，即：“不等你说完，接过话筒就打电话。”

进程优先级的两种确定方式：静态方式和动态方式。

（1）静态优先级是在创建进程时就确定下来，而且在整个运行期间保持不变。

（2）动态优先级是随着进程的推进而不断改变的。

4． 其它调度算法简介（作一般了解）

UNIX 常用调度命令及命令执行过程

1． UNIX 系统中的进程调度

UNIX 系统的进程调度采用多级反馈队列轮转法。

2． UNIX 常用调度命令

（1）nohup 命令

功能：是以忽略挂起和退出的方式执行指定的命令。

（2）at 命令

at 命令允许指定命令执行的时间。

（3）batch 命令

它提交的作业的优先级比 at 命令提交的作业的优先级低。

（4）jobs 命令

功能：用来显示当前 shell 下正在运行哪些作业（即后台作业）。

（5）fg 命令

功能：把指定的后台作业移到前台。

（6）bg 命令

功能：可以把前台进程换到后台执行。

3．shell 命令执行过程（略）

死锁问题

1．死锁产生的原因以及必要条件

死锁产生的原因 ：

当某个进程提出申请资源后，使得有关进程在无外力协助下，永远分配不到必需的资源而无法继续运行，这就产生了一种特殊的现象——死锁。

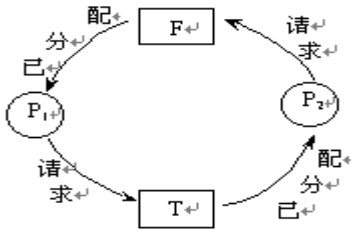
在许多实时应用中，比如计算机控制运输和监视系统方面，死锁问题也极为重要。

2．死锁产生的例子

例一：我们先来看一个申请不同类型资源的死锁例子，假定有两个进程 P_1 和 P_2 都要修改文件 F，修改时都需要一条暂时存放信息的磁带，而只有一台磁带机 T 可用。又假定由于某种原因，在进行修改之前，P_2 需要一暂存磁带（例如为了修改，要重新组织输入数据）。设 F 和 T 都是可重用资源，它们分别表示允许更新文件和允许使用磁带机。于是 P_1 和 P_2。可有如下形式：

```
进程 P1                          进程 P2
  :                                :
申请文件 F                       申请磁带机 T
r1: 申请磁带机 T                 ...
  ...                            r2: 申请文件 F
释放磁带机 T                       ...
  释放文件 F                       释放文件 F
  ...                            释放磁带机 T
  ...
```

*分析：*从上面的申请-释放过程可以看出，进程 P_1 和 P_2 有可能“同时”分别到达 r_1 和 r_2 处，例如，P_2 首先得到 T，然后 P_1 得到 F，接着 P_1 到达 r_1，最后 P_2 到达 r_2，此时，若 P_1 继续运行，则占有 F 的进程 P_1 将阻塞在 T 上，若 P_2 继续运行，则占有 T 的进程 P_2 将阻塞在 F 上，如果 P_2 不能前进，则 P_1



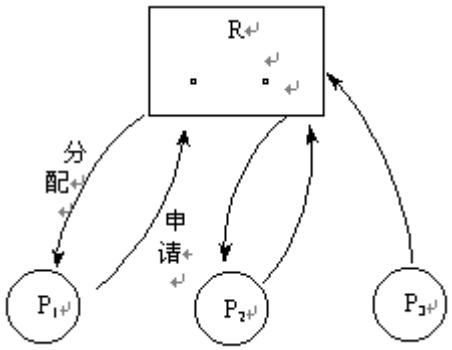
也不能继续下去，反之亦然。我们说这两个进程处在死锁状态。

例二：现在我们再来看一个关于相同类型资源共享的死锁例子，假设有一类可再使用资源

```
申请一页（或扇区）
  ...
申请一页（或扇区）
  ...
释放一页（或扇区）
  ...
释放一页（或扇区）
  ...
```

R，例如主存或外存，它包含有 m 个页面或扇区，由 n 个进程 P_1，P_2…，P_n（$2 \leqslant m \leqslant n$）

共享。假定每个进程按右图顺序申请和释放页面（或扇区）：*分析：*这里每次申请和释放只涉及 R 的一个分配单元（页或扇区）。因此，当把所有单元全部分配完毕时，便很容易发生死锁；占有 R 的单元的所有进程（前 m 个进程）会永远阻塞在第二次申请上，而有些进程（n~m 个进程）类似地会阻塞在它们的第一次申请上，在图 3.12 中说明了 n＝3，m＝2 时这种系统的状态，这类死锁是相当普遍的。



3．产生死锁有四个必要条件：

（1）互斥条件。

（2）不剥夺条件。

（3）请求和保持条件。

（4）环路等待条件。

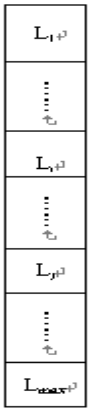
4．预防死锁

（1）破坏“请求与保持条件”

这种方法的基本思想是：每个进程在运行之前，必须预先提出自己所要使用的全部资源，调度程序在该进程所需要的资源未得到满足之前，不让它们投入运行，并且当资源一旦分配给某个进程之后，那么在该进程的整个运行期间相应资源一直被它占有，这就破坏了产生死锁的请求与保持条件。

（2）破坏环路条件

这种方法的基本思想是：对系统提供的每一项资源，由系统设计者将它们按类型进行线性排队，并赋予不同的序号。例如，设卡片输入机为 1，打印机为 2，磁带机为 3，磁盘机为 4，……。所有的进程都只能严格地按照编号递增（或递减）的次序去请求资源，亦即，只有低编号的资源要求满足后，才能对高编号资源提出要求；释放资源时，应按编号递减的次序进行。由此可以看出，对资源请求采取了这种限制之后，所形成的进程—资源图不可能再产生环路。如图 3.13 所示 .



（3）资源受控动态分配

为了避免死锁发生，操作系统必须根据预先掌握的关于资源用法的信息控制资源分配，使得共同进展路径的下一步不致于进入危险区，即只要有产生死锁的可能性，就避免把一种资源分配给一个进程。

5．发现死锁

假定系统有 n 个进程 P_1，P_2，…，P_n 和 P_m 种类型资源 R_l，R_2，…，R_m。.建立资源分配表 S 和进程等待表 W，分别如表 3.1 和表 3.2 所示，其中 a_{ij} 表示分配给进程 P_i 的资源 R_j 的数目，b_{ij} 表示进程 P_i 请求资源 R_j 的数目。另外为每一个进程设置一个等待资源计数器 C_1，C_2，…，C_n，它们表示引起相应进程被阻塞的资源数目，将末阻塞的进程组成一个表 L（或队列）。

表 3-1 资源分配表

资源 / 进程	R_1	R_2	…	R_i	…	R_m
P_1	a_{11}	a_{12}	…	a_{1i}	…	a_{1m}
P_2	a_{21}	a_{22}	…	a_{2i}	…	a_{2m}
…	…	…	…	…	…	…
P_i	a_{i1}	a_{i2}	…	a_{ii}	…	a_{im}
…	…	…	…	…	…	…
P_n	a_{n1}	a_{n2}	…	a_{ni}	…	a_{nm}

表 3-2 进程等待表

资源 / 进程	P_1	P_2	…	P_i	…	P_m
R_1	b_{11}	b_{12}	…	b_{1i}	…	b_{1m}
R_2	b_{21}	b_{22}	…	b_{2i}	…	b_{2m}
…	…	…	…	…	…	…
R_i	b_{i1}	b_{i2}	…	b_{ii}	…	b_{im}
…	…	…	…	…	…	…
R_n	b_{n1}	b_{n2}	…	b_{ni}	…	b_{nm}

其发现死锁的算法如下：（1）把未阻塞（$C_i=0$）的进程 P_i 记录在 L 表中（其全部资源请求已得到满足的进程）。（2）从 L 表中选择一进程，根据资源分配表 S 释放分配给该进程的所有资源。（3）由进程等待表 W 依次检查和修改需要该进程释放资源的每一个进程的等待计数器 C_j。（4）若 $C_j=0$，则表示该进程所请求的资源已得到满足，不再阻塞，将 P_j 记入 L 表中。（5）再从 L 表中选取另一进程，重复上述操作。（6）若所有的进程都记入 L 表中，则系统初始状态为非死锁状态，否则为死锁状态。

6．解除死锁

（1）资源剥夺法

① 还原算法。即恢复计算结果和状态。

② 建立检查点主要是用来恢复分配前的状态。

（2）撤消进程法

① 程序的优先数，即被撤消进程的优先数。

② 作业类的外部代价

③ 运行代价

三、本章小结：

CPU 调度是操作系统最核心的调度，它根据算法选择合适的进程，并把 CPU 分配给该进程使用。

在操作系统中最主要的队列有两类：一类是 I/O 请求队列，一类是就绪队列。

处理机调度分为三级。作业调度的基本功能是选择有权竞争 CPU 的作业；CPU 调度（即进程调度）是从就绪队列中选择一个进程，并把 CPU 分配给它；中级调度往往实现进程的挂起和进程映象的对换。处理机调度可由系统进程来实现。

确立调度策略是件复杂的工作，往往要兼顾多种因素的影响，通常评估性能时都要考虑的几个指标：CPU 利用率，吞吐量，周转时间，等待时间，响应时间。

先来先服务（FCFS）是最简单的调度算法，但可能导致作业等待很长时间。轮转法（RR）对于分时系统更为合适。

优先级算法：只是简单地把 CPU 分给优先级最高的进程，它可是抢占式也可以是非抢占式的。它是抢占式算法，而 FCFS 是非抢占式的。

优先级算法：只是简单地把 CPU 分给优先级最高的进程，它可是抢占式也可以是非抢占式的。它是抢占式算法，而 FCFS 是非抢占式的。

其它的常用调度算法还有：短作业优先法、最短剩余时间优先法、多级队列法、多级反馈队列法。

多级队列算法允许对不同类型的作业使用不同的算法，最常用的方法是对前台队列采用轮转法调度，对后台队列采用 FCFS 方法调度，反馈法允许一个作业从一个队列移到另一个队列。

对于 UNIX 系统中进程管理办法，采用两级调度：中级调度（即对换进程，它解决内存分配）和低级调度（解决 CPU 分配）。进程调度采用多级反馈队列轮转算法。

Shell 解释程序的工作过程基本上是读入命令行、分析命令行和构成命令树，创建子进程来执行命令树等步骤。

本章还介绍了 UNIX 系统中常用调度命令。