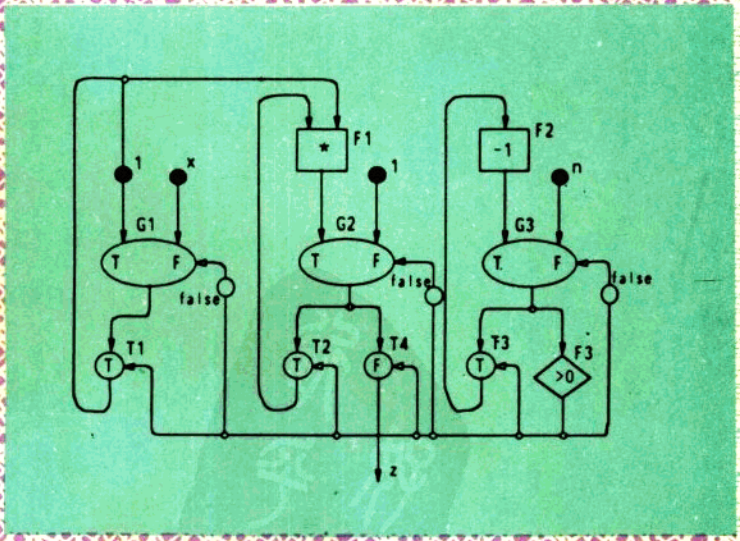
并发程序设计语言和模型

赵 宏 编著



东北大学出版社

内 容 简 介

本书以作者亲自从事的教学实践为基础，对并发程序设计的模型和语言加以介绍，并结合几个典型的并发和分布问题，给出一些模型和语言的应用实例，其目的是使读者更深刻地理解并发程序设计的概念和方法，从而可以完成对并发的描述，进而实现一个并发系统。全书共分四章：第一章介绍和描述了并发程序设计一些概念；第二章介绍了对并发系统的基本描述手段和方法；第三章介绍了五个并发程序设计的模型系统；第四章介绍了几个并发程序设计语言。之后，对于各种并发程序的模型和语言从应用场合、通信、缓冲等十多个方面进行了比较。在本书的最后，给出了一些思考题和练习题。

本书可以作为计算机软件和应用专业研究生的课程参考教材，同时也可以供计算机研究人员、计算机专业本科生参考。

前　言

随着计算机技术的不断发展，计算机应用的场合也在不断加深和拓宽。现实世界中越来越多的应用要求计算机的支持，越来越多的人和工作离不开计算机。当今的社会正向着有序化、组织化、合作化和整体化的方向发展，许多项目和工作已经不是由一方或一个组织所能解决的了，只能通过多方的合作才能完成。大到太空探索研究项目、生态平衡项目，小到日常的办公过程、工厂生产装配线上的生产活动，这些无不体现了多个人或多个实体之间有组织、有计划的合作、交互和配合。计算机技术和通信技术的发展也为这方面的工作提供了越来越好的环境和条件。例如：高速通信网络、多媒体技术、形式化描述技术、智能人机接口、分布式系统、分布式数据库技术等等。有了这些环境的支持，现实中的合作和并发活动才能在计算机中得到很好的描述和解决。

然而由于历史的原因，作为描述这些活动的程序设计一直是传统的、顺序的单个程序流程运行方式。这样，一方面使计算机的描述和实现不能真实地反映客观实际，另一方面也使所实现的程序运行起来不能有很高的效率。同时，它还使得人们的编制程序的思维一直受到局限。在这种情况下，由于没有相应的并发描述机构，为了达到同时运行的效果，就必需投入很大精力去在实现上自己构造相应的支持机制，不能把主要精力放到怎样描述所要解决的问题上去。

编写本书的目的就在于使读者从顺序程序设计的局限性中解脱出来，建立起并发程序设计的概念，了解并发程序的模型和语言的风格以及在进行并发程序设计时应该考虑的问题，掌

握并发处理语言的语法和语义以及并发程序设计的基本技巧，为以后的课题研究和实际工作打下基础。

本书主要介绍并发程序设计的模型和语言，并结合一些典型的并发和分布问题，给出一些这些模型和语言的应用实例，以期使读者更深刻地理解并发程序设计的概念和方法。全书总共分为四章：第一章是对于并发程序设计一些概念的描述，介绍了进行并发程序设计的环境、硬件支持、网络系统、描述机制等等；第二章介绍了对并发系统的描述手段和方法，引出了进行并发活动时的一些问题，描述了并发活动的一些典型问题以及在进行并发描述时的一些应该考虑的问题；第三章介绍了五个并发程序设计的模型系统，包括共享变量模型、交换函数模型、Petri网模型、数据流模型和CSP模型；第四章介绍了几个并发程序设计语言，包括并发PASCAL、分布进程、ADA语言、共享资源、PLITS和ESTELLE语言。在本书的最后，我们给出了一些思考题和练习题。

学习本课程，要求读者具有以下计算机方面的基础知识，它们是：计算机组成原理和体系结构、数据结构、程序设计语言、操作系统以及一些关于自动机、图论等知识。

本书可以作为计算机应用专业和计算机软件专业研究生的课程参考教材，同时也可以作为广大计算机研究人员、计算机专业本科生的参考之用。

本书的内容曾经由作者对计算机专业的研究生和本科生进行过多次讲授。在此基础上，作者通过多次的修改和补充写成了本书。

由于水平所限，本书中肯定有不少的缺点和错误，作者非常诚恳地希望广大读者提出批评和指正。

作　者

1995年8月于沈阳东北大学

目　录

第一章　并发程序设计概念

计算机应用环境的复杂化、多样化和随时变化的性质对计算机支持机制提出了更高的要求。对于这种工作如何支持，使用什么样的工具和手段来描述和表示这些合作关系，已经成为非常必要的研究课题。这种研究的一个重要方面就是并发程序设计的研究。在实际生产过程和生活中，许多活动是同时进行的。大到国际事务，小到每个人的日常动作，我们经常会看到或碰到多个活动同时进行的例子。在计算机应用中，要把这些活动描述出来，就要真实地反映这种同时动作的情况。然而由于历史的原因，程序设计一直是传统的、顺序的单个程序流程运行方式。这样，一方面使计算机的描述和实现不能真实地反映客观实际，另一方面也使所实现的程序运行起来不能有很高的效率。同时，它还使得人们编制程序的思维一直受到局限。在这种情况下，由于没有相应的并发描述机构，为了达到同时运行的效果，就必需投入很大精力去在实现上自己构造相应的支持机制，不能把主要精力放到怎样描述所要解决的问题上。

并发程序设计正是为解决这一问题而发展起来的一种新型的程序设计方法。它完全摈弃了在以往的程序设计方法中程序只是单一的顺序的结构的思想，以真实地反映客观实际和提高运行效率为目标，使所编制的程序中有许多部分能够同时动作。这样一来，程序也更易于为人们所理解了。应该说，这样编制程序更方便和容易了。可以用它来描述和模型化一个实际问题，尤其是较大型的分布和并发问题。同时，也可以用它对一个系统进行分析和验证，研究所设计的系统是否满足要求，是否有死锁，是否有溢出，是否能达到所期望的目的等等。进

而，还可以用它从直观的描述来直接自动生成程序代码，从而较容易地实现一个系统。

在这一章中，我们介绍一些关于并发程序设计的概念。

所谓并发程序设计（Concurrent Programming）可以解释为编制这样的程序，使其在一给定的时间内可以有几部分程序同时运行。

并发问题对于我们来说应该并不陌生。在实际生活和生产过程中，并发活动是经常发生的，我们早已经常遇到过这些情况，只是大多数人对其并不注意罢了。在计算机系统中，并发程序设计的概念完全是根据实际提出来的。在本章以下各节，我们将介绍与并发相关的问题和概念。

1.1 几个并发性活动的例子

下面，我们举几个具有并发性(Concurrency)的例子：

1．一个大工程项目的活动

一个大的工程项目（例如，建筑一幢大楼）可能本身包含了许多比较独立的更小一点的工作步。这些工作步之间有些必须要按先后顺序进行，因为这些工作的前一个工作步的完成是后一个工作步能够开始进行的先决条件；有些则可以同时进行，因为它们之间并不存在相互之间的影响作用。利用这种有些工作可以同时进行的特性，对于一个大的工程中的许多工作步的按排综合进行优化，就可以从人力、物力和时间上更节省。例如图1.1表示的是一个工程的流程。

在该图所示的工程中，任务2与任务1、3、4、5可以同时进行，任务3与任务4也可以在同时进行，而其它的活

动就必须按顺序进行了。例如，任务 5 就必须要等到任务 3 和任务 4 都完成之后才能开始执行；而任务 3 和任务 4 的开始又必须等待任务 1 的完成。我们不妨来分析一下，这个工程的各工作步之间为什么要并发进行呢？并发进行之后有些什么好处呢？这里，基本的原因有以下两个：

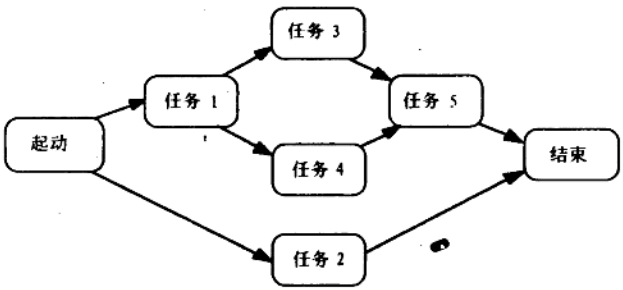


图1.1 一个大的工程中各任务之间的并发活动

(1) 有许多工作人员可以同时参加工作；

(2) 如果这些任务允许同时进行的话，整个工程时间可以相应地缩短。

我们应该记住这两条，因为在计算机系统中，同样是两条类似的原因要求我们进行并发计算。

2．计算机中一个表达式的计算

在计算机系统中，一个表达式中可能有 +、−、×、÷ 等多种运算，而这些运算不一定必须一个个地顺序进行，有些运算可以同时进行。例如，有一个表达式的形式为：(2*A)+((C-D)/3)，其运算过程如图1.2。

在这个例子中，我们可以看到“2 * A”与“C - D”是两个互不相关的计算操作。这样，它们就可以独立地（同时地）分别进行运算。否则，如果它们相互依赖的话，就必须是一个先进行，另一个则根据前一个的运算结果再继续运算。例如，上面表达式中的除法“/”，就必须在操作“C - D”完成之后才能进行，这是因为它是与“C - D”相依赖的，这里的除法操作要用到“C - D”的结果。

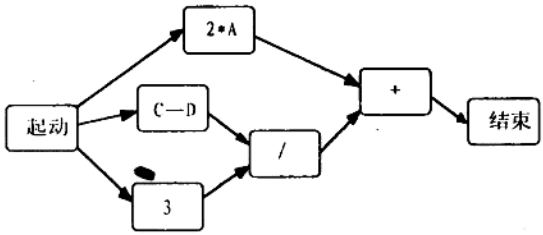


图1.2 一个表达式的运算过程

3. 计算机操作系统

计算机操作系统是一个比较复杂的管理程序，是用户使用计算机系统的接口界面。在计算机操作系统中，更要求并发活动，这是各方面的条件要求和支持的。这些因素是：

(1) 现代的计算机一般都配有许多可用的外部设备，如输入键盘、打印机、软磁盘、硬磁盘、磁带、鼠标器、数字化仪、扫描仪、绘图仪、CPU、中断机构、通信机制等。计算机操作系统要管理、控制这些设备，组成一个高效率的、用户满意的工作环境来为用户服务。

(2) 一个计算机系统包含了许多的系统软件和应用软件包，即它有许多可同时使用的工具。

(3) 计算机的操作系统被要求能够支持多道程序（Multi-programming），使许多个用户可以同时运行自己的程序，而且应该使每个用户都感觉到好象是他们自己独占机器一样。这样，操作系统就应该具有调度能力、有使有限的系统资源为多个用户共享的能力。（例如，如何分配CPU时间、如何分配有限的内存、如何分配磁带、磁盘和打印机等外设）

综上所述，我们说，一个计算机系统之所以要支持多道程序或并发程序设计的原因有二：

① 为了有效地利用各种资源

如果只进行一道作业，有很多时刻CPU是处于空闲状态的（譬如，在从键盘输入数据时，由于输入设备的速度与CPU的速度相比是很慢很慢的，CPU在大部分时间内都在等待）。另外，在单道作业中，若该作业不是很大的话，系统的大部分内存将被闲置，没有充分发挥作用。

② 提高对用户的响应速度

系统中每个用户都可以相互独立地向自己的作业发送命令，并可以尽快地得到响应。一个短作业可以与长作业同时进行，并先于长作业而完成。

1.2 有关并发的几个概念

有两个概念，我们应加以区别。这两个有关的概念是并行和并发。

并行（Parallel）：一般是指计算机中的各硬件之间的同时操作，特别是指多个CPU之间的同时计算操作。为了衡量一个机器的并行性，把机器按指令流和数据流分为四类：

SISD 单指令流单数据流

(Single-Instruction-stream, Multiple-Data-stream)

MISD 多指令流单数据流
(Multiple-Instruction-stream, Single-Data-stream)
SIMD 单指令流多数据流
(Single-Instruction-stream, Multiple-Data-stream)
MIMD 多指令流多数据流
(Multiple-Instruction-stream, Multiple-Data-stream)

在计算机体系结构发展中，现在人们正在寻求新的计算结构、以取代老的、传统的结构—“Von Neumann”结构的计算机，这种结构的计算机就是数据流计算机（Dataflow）。

为什么要提出Dataflow结构呢？

以往的计算机结构往往都过份强调指令流的作用，而忽视了数据流的作用，把数据流看成是被动的、僵死的被作用对象，完全受指令流控制。然而从实际情况中我们可以看到，数据流正是一个异常活跃的流，它不只是被动地被操作，它是事物的主要部分，完全可以由于它的活动来激活程序执行。使用这种数据流结构，可以更好地利用程序的并行性，提高效率。

Von Neumann: 数据被动地存放于存储区，指令则一次一个地（顺序地）由程序计数器控制，按事先按排好的顺序执行。

Dataflow: 数据是活动的，同步地输入二维程序并激活每一条作用于它的指令。

这里，由于有多个指令流在同时执行，而每一个指令流又有一系列的指令，故它是二维程序。

这种结构使CPU本身的并行操作更明显。例如，现在有许多大的向量计算机（其中的数据好象是一个向量或一个矩阵，计算机的操作可以同时对所有这些数据进行，体现了多数据流的概念）、流水线计算机（Pipeline，在其中，数据就好象在工厂中的一条装配线上的装配件，随着它们的流动，不同工

作点上的多个操作被激活，同时对其进行加工处理，体现了多指令流的概念），这些都属于数据流计算机。

在多处理机系统中，多个处理机之间的操作可以同时进行，这也体现了CPU之间的并行。

并发（Concurrency）：并发是指多个程序段在执行的时间上是相互重叠的，即使这种重叠只是一小部分，我们也说这些程序段之间是并发执行的。

并发和并行在概念上有一些差别，这可以用表1.1表示：

表1.1

名称（并发或并行）	软件/硬件	逻辑/物理	进程/处理机
并 发	软件上	逻辑上	进程实现
并 行	硬件上	物理上	处理机实现

并发是指进程级别上的在逻辑上的重叠执行，而并行则是指处理机级别上的在物理上的重叠执行．并发执行的程序在实际机器上运行可能并不是并行执行的。

下面这个例子也许会有助于说明以上的问题．设有一个售票处，外面有三个队列的人在排队买票．为能更说明问题，假设外面的队列与里面的售票人员是隔绝的，相互谁也看不见谁．他们之间只能通过售票的小窗口交钱和取票．在一般情况下，售票处里面也有三个售票员分别在三个对应的窗口售票，这样我们把里面的售票动作比作是并行运行，把外面的几个队列同时排队买票的动作比作是并发执行，就比较好理解我们所提出的问题了．图1.3表示了这种情况．

然而，并行和并发是不同的，从概念上讲，硬件上的是否并行运行并不直接影响在软件上是否并发执行．就拿本例来说，假设售票处里面只有一个售票员，而这个售票员的工作效

率相当高，有一套高超的售票技巧，他的效率高到可以同时对三个窗口的队列同时售票，而且能把售票动作分解，并且以相当快的频率在三个售票窗口之间进行切换．例如他可以在第一个窗口从该队列接过钱，然后马上转到第二个窗口将该窗口所要的票交给该队列队首的人，又到第三个窗口为第三个队列的人找回零钱．这之后又马上回到第一个窗口去为第一个队列的人去服务．这样，如果其服务速度足够快的话，每个被服务的人并没有感觉出他的购票动作被打断，给所有队列要买票的人的感觉是每个队列都有一个售票员在为其服务．

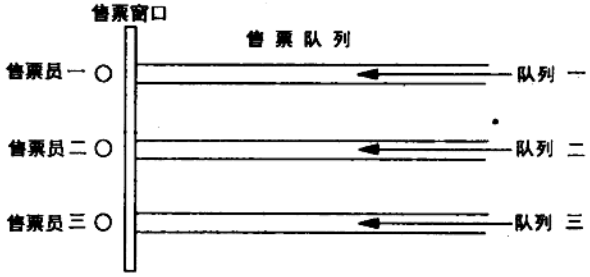


图1.3 服务员与队列及并发和并行的关系

从概念上讲，凡是能够用并发执行的程序段来完成的任务，都可以等价地由一个相应的顺序程序来完成．这种“等价代换”也为数据库中的一致性更新及安全性的后备恢复等问题的分析简化提供了手段．

然而，使用并发执行的程序，可以取得如下的好处：

① 加快了程序的执行速度．在同样时间内，程序顺序执行要比程序并发执行慢，这是显而易见的．

② 便于进行程序设计，能够较为真实地反映出客观实际的情况。因为客观世界中的活动是并发进行的，它应该用对应的并发机制去描述，才能达到自然、真实和准确的目的。

③ 可以更好地利用硬件上的并行操作来完成一个任务（提高资源利用率）。一个计算环境可能包括许多硬件设备，一个顺序程序不可能同时使用所有这些硬件设备。而在并发执行的情况下，一个并发程序段可以使用另一个并发程序段当时没有使用的硬件设备。

例如，有图1.4表示的一段程序：

在这里我们可以看到，K的产生，也即“$L * M * N$”操作的进行，必须在中间变量L、M和N三个数产生之后才能进行。而L、M和N的产生则完全是相互无关的，完全可以并发地执行，也就是重叠地或同时地完成。

```
L = I1 * I2
M = I3 * I4
N = I5 * I6
K = L * M * N
```

其并发执行的有向图如右所示。

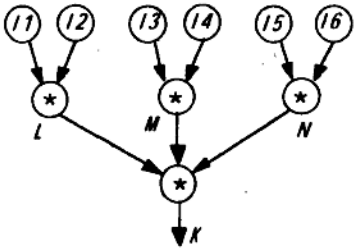


图1.4 一个并发执行的有向图

然而，我们也应该看到，程序并发执行以后，会给我们带来一些新的问题。并不是将原来顺序执行的程序段简单地同时执行就可以了，并发执行有其新的特点：

1. 并发程序丢失了顺序程序的封闭性。程序顺序执行时，相同的程序代入相同的参数来运行，所得到的结果是一样

的．然而，程序并发执行时情况就不一样了．程序并发执行时，同样一个程序，代入相同的参数，在不同时间去运行，可能会产生不同的结果．

2．并发执行使得程序和计算不再是一一对应的了．在程序顺序执行时，程序的一个运行就是一个计算，一个程序只对应一个计算．在程序并发运行时，一个程序的运行可能有多个计算，即多个进程与之对应．而且，一个程序可能同时运行多次，每次运行又可能对应多个进程．

3．程序并发执行时，各程序段之间是相互制约的．这是因为在许多情况下各程序段之间并不是相互无关的，它们之间具有这样和那样的关系．因此，它们的运行就会互相牵制、互相影响，一个程序段就不能不顾及其它程序段的情况而自己运行下去．

在计算机系统中，并发活动是大量存在的和经常发生的，而且实际情况要比以上我们举出的例子复杂得多．比如多个进程之间甚至多个处理机之间的通讯、多个任务对某一个共享资源的存取、系统资源的统一分配和回收等等．对于这种环境，如何更好地反映并行特性，怎样编写程序使得它能很好地利用硬件的并行性和使程序更好地并发执行，是非常重要的．本书的目的就是要介绍关于并发程序的有关概念和几种并发程序设计的方法，熟悉和掌握并发语言的特性．

另一个概念是软件中的并行性（Parallelism）．

在软件中，一般有两类并行性，分别叙述如下：

(1)，对于一块数据的各个部分进行共同的操作．例如，对于几个数组进行相加，其动作是相应元素进行相加，其对应Fortran程序如下：

```
DO 10 = 1,100
F(I) = A(I) + B(I) + C(I) + D(I)
```

```
10    CONTINUE
```

这个例子的并发性在于，可以把产生数组F的

```
F(1) = A(1) + B(1) + C(1) + D(1)
F(2) = A(2) + B(2) + C(2) + D(2)
... ... ... ...
F(100) = A(100) + B(100) + C(100) + D(100)
```

这诸多个相同的操作同时进行．这种并行性我们称为规则的（Regular）或操作同构的．

(2)．另一种方式是对于一系列数据进行不同的操作．例如，一个多条赋值语句的Fortran程序如下：

```
A = E - G
B = H * Z
C = E * H + F
D = E + G
```

这种可以同时执行的语句所体现的并行性是非规则的或操作异构的．

这些软件的并行性直接影响着硬件系统的设计，硬件系统应该设计得能够充分利用这些软件上的并行性，以便能使硬件的并行与软件的并行性最好地配合起来，取得最佳效果．一般认为，SIMD系统适用于第一种并行性，而对于MIMD系统来说，则较适用于第二种并行性．

为了能更好地利用软件的并行性，人们已经作了许多尝试，包括：硬件本身的创新和发展以及软件方面的努力（如多道程序的操作系统）．但是，系统结构和计算机技术的不断发展越来越向人们指出这样一个事实：即用多个处理机（每个处理机都是一个自含的、拥有自己内存的系统并通过某种方式进行通信）组成的系统来完成一个任务，要比用一台大的机器运行多道程序更经济、更可靠、效率更高和更利于发挥软件的并行性．我们要记住这句话，它将作为我们这本书所叙述的内容的基础。