CHILL

用户手册

CCITT第XI研究组编　寇化栋 译　郭肇德 校

人民邮电出版社

CHILL用户手册

CCITT第XI研究组编

寇化栋　译

郭肇德　校

人民邮电出版社

内 容 提 要

本书详细介绍了CCITT推荐的通信用高级程序设计语言CHILL的全部语法及描述。在编写上采用由浅入深、循序渐进的方法，特别注意以大量的例子对各个CHILL概念进行解释。

本书可供从事程控交换软件的设计人员、维护人员、CHILL语言的编译者及有关人员使用。

CHILL用户手册

CCITT第XI研究组编

寇化栋 译

郭肇德 校

*

人民邮电出版社出版发行

北京东长安街27号

北京振华胶印厂印刷

新华书店总店科技发行所经销

*

开本：787×1092 1/16　　1991年4月　第一版

印张：15　页数：120　　1991年4月北京第1次印刷

字数：366 千字　插页：2　印数：1—2 500　册

ISBN 7-115-04448-1/TP·057

定价：7.15 元

译 者 序

本书最初是根据CHILL bulletin 1984年第1期《CHILL User's manual》专辑翻译的，后来又对照“国际电信联盟”的正式文件进行了修改。这本书在章节结构的安排、描述方法等方面与建议Z.200非常相似。建议Z.200的对象主要是从事语言编译的技术人员，本书的对象是广大的CHILL使用者。它兼顾了完整、严格和通俗三方面的要求，不仅给出了CHILL语言的全部语法及描述，同时还以大量的例子和深入浅出的叙述对各个CHILL概念进行解释。对于初学CHILL的人来说，比较难理解和掌握的是过程的各种属性、并发执行、名字的作用域和可见性。本书专门用第7章、第8章和第9章分别对它们进行介绍和解释，尤其在概念上容易搞错的地方，用笔虽然不多，但起到了画龙点睛的作用。本书确实是学习和使用CHILL语言的好参考书。

名词的翻译是一个难题。人民邮电出版社电信图书编辑部的编辑同志们做了一件有益的工作。他们广泛征求意见，整理了一份“CHILL译名对照表”。对于本书中的名词，基本上采用这张“对照表”上的译名。但是对GOTO、IF、FOR、WHILE这些国内已经流行的名词，就象现在没有人将序号A、B、C、D译成甲、乙、丙、丁一样，译者没有把它们译成汉字，而用原文。至于DO仍采用原文是因为在CHILL中DO动作不仅有循环的含义，还有作为语句括号的作用，将其译为循环动作会引起概念上的错误。

非常感谢我的老师郭肇德教授。不仅是因为他校对了本书的译文，更主要的是在带领我们从事程控交换机研究的初期，他就指导我注意收集CHILL的资料，开展对CHILL的研究。现在我们对CHILL的研究逐步深入，已研制成功具有并发功能的CHILL编译程序。所取得的每一个成果，都倾注着他的指导和关怀。

由于本人水平有限，译文中定有错误，敬请诸位指正。

通信工程学院

寇化栋

目　录

1 引　言

1.1 概　述

随着存储程序控制（SPC）交换系统使用的日益增多，程序设计正在成为电信技术的一个重要部分。早在20世纪60年代后期，CCITT开始调查SPC程序设计时就已认识到这个前景。在1973—1976研究期内，CCITT着手从事一项特殊的工作——研制用于SPC的系列语言：CHILL、SDL和MML。

当时，SPC程序设计中所面临的大多数问题都已经很清楚了，并且已经开发了或正在开发多种程序设计语言用于解决这些问题。人们很自然会提出这样一个问题：已经有如此丰富的用于SPC程序设计的语言族，为什么CCITT还要着手研制另外一种程序设计语言呢？理由主要有两点：

- 要提供一种面向SPC系统的所有生产厂家和全部用户的标准语言。
- 弥补已有的程序设计语言用于SPC系统的不足，将某些语言独有的特性合并在一种语言中。

遍布全世界的各大电信团体促进了语言标准化的发展。大多数电信设备的生产厂家和用户都以某种形式与软件打交道。因此在软件领域，标准化所带来的好处也将是巨大的。

CHILL集20世纪70年代所研制的各种性能最好的高级程序设计语言之大成，确实反映语言技术的当今水平，可应用于广阔的领域。

CHILL的研制工作始于1975年，到1979年底产生了该语言的待批草案。1980年11月CCITT全体会议批准了CHILL定义，形成CCITT建议Z.200。

在CCITT1980—1984研究期内，CHILL定义得到了改进和充实。

除了建议Z.200外，还产生了下述文件：

1. **CHILL用户手册**，即本书。这个手册以Z.200 1980年版为基础。建议Z.200是面向编译结构的，而用户手册却是面向用户来描述CHILL语言。

2. **CHILL概述***。这是一本CCITT手册，它浅显地介绍了CHILL语言。

3. **CHILL的形式定义**。这也是一本CCITT手册，它使用数学符号以严格的形式化的方式描述CHILL语言。

在研制的初期，列出了对CHILL的几项要求。CHILL至少应该适于下述应用领域：

- 呼叫处理
- 测试与维护
- 操作系统
- 联机与脱机支援

* Introduction to CHILL一书已由人民邮电出版社翻译出版，书名为CHILL概述。——编注。

- 实现人机语言（MML）
- 验收测试。

尽管这张表是根据交换系统的需求列出的，但它涉及的范围却非常广泛。事实上，这张表包括了计算机科学中被称为系统程序设计的大部分领域。因此，实际上CHILL是一种用于系统程序设计的高级语言。

CHILL并不打算为上面提到的每个应用领域提供特殊的语言结构，但是它有通用的基础结构，为具体的应用领域提供了一些适用的可能手段。CHILL必须考虑到在曾经编制过的软件中SPC系统软件是最大、最复杂的软件中的一种，并且还必须考虑到SPC系统软件将来会更趋庞大和更加复杂。因此CHILL必须是用于大系统的，而不是用于小系统的程序设计语言。为了满足上述需求，CHILL必须：

- 提高可靠性
- 能产生高效的目标代码
- 灵活且功能强
- 利于模块化和结构化程序设计
- 不依赖于机器
- 容易学习和使用。

作为一种语言，CHILL是独立于机器的。然而在具体实现时，可以包含一些由具体实现所定义的语言成分。包含这类成分的程序一般是不可移植的。

在设计CHILL时，最基本的考虑是可靠性。因为SPC系统如此庞大和复杂，使用现行的软件工程技术再也无法满足其严格的可靠性要求，因此强调可靠性已成为SPC系统的最根本的要求。所构造的CHILL既可以实现被动可靠性又可以实现主动可靠性。被动可靠性通过设计语言的结构来实现，首先它可以避免引入错误。语言的功能强且结构灵活，可以自然地构造系统的模型，并且有助于结构化、模块化程序设计。主动可靠性是通过提供大量的一致性检查来完成的，主要的手段是可见性控制和模式控制。为了确保可靠性而又不致于使效率的损失大到不可接受的程度，许多检验可以静态地进行。只有少数被称为CHILL的动态语义条件的语言规则不能静态地确定。违反动态语义规则将引起运行异常。

本章以下各节根据Z.200（1980）给出了CHILL语言的简介。关于Z.200（1984）的内容将在1.13节中介绍。

1.2 语言简介

CHILL程序由三个部分组成：

- **数据对象**的描述，
- 在数据对象上执行的**动作**的描述，以及
- **程序结构**，即建立程序要素以形成一个整体的方法。

数据对象由**说明**语句和**定义**语句来描述。数据对象可以是**值**或者**单元**，单元可以存放值。**动作**由**动作语句**来描述。动作语句描述了在数据对象上进行的操作，以及把值存入数据单元或者从数据单元中检索值的次序。**程序结构要素**确定了程序的结构以及通过程序结构确定程序的数据对象的生存期和可见性。

对于在给定上下文中数据对象的使用，CHILL提供了充分的静态检查。

以下各节简述了各种CHILL概念。每一节是相应章的简述，详细的概念描述在相应章中给出。

1.3 模式与类别

CHILL的基本数据对象是**值**和可存放值的**单元**。值附有类别，而单元附有**模式**。模式和类别确定了单元和值的性质，这些性质由编译程序来检验（模式类似于其它程序设计语言(如Pascal）的类型)。值的类别确定了可以包含这个值的单元的模式。单元的模式确定了被存放在其中的值的集合，以及其它与之相关的性质，如它的大小，它的内部结构和它的访问方式。

模式可以在程序中显式地指明。CHILL提供了后面将提到的一组预定义模式。此外，CHILL语言还允许使用模式定义语句来建立新的、也可能更复杂的模式。

用于顺序执行程序设计的预定义模式，按以下范畴分类：

- **离散模式**：整数模式、字符模式、布尔模式、集合（符号的）模式和它们的范围模式。
- **幂集模式**：某些离散模式的元素的子集。
- **引用模式**：用于对单元引用的受限引用模式、自由引用模式和行模式。
- **组合模式**：串模式、数组模式和结构模式。
- **过程模式**：过程被看作可处理的数据对象。

除了用于顺序执行的程序设计的模式外，CHILL还提供了下述模式，以用于并发执行的程序设计：

- **同步模式**：用于进程间同步和通信的事件模式和缓冲区模式。
- **实例模式**：用于标识进程。

另一类在CHILL中没有显式标记的模式范畴（即程序不可说明这一类对象）是所谓的**动态模式**。与其它模式相比，这些模式提供了灵活性并且使得经济地分配寄存器成为可能。动态模式是某些性质只能动态地确定的一种模式。

1.4 单元及其访问

在程序中通过**单元说明语句**来建立单元。单元说明语句也定义了一个**名字**，当存值或者取值而必须访问所建立的单元时，就使用这个名字。单元也可作为某些特定语言成分的结果(例如**DO动作**中的循环计数器)隐式地建立。最后，单元可以用动态的方法**产生**（GETSTACK内部子程序)。另外可以通过单元等同说明或者有基说明来引入另外一些名字，用来访问在别的地方建立的单元。

单元可以是**可引用的**，这意味着对于这个单元存在一个对应的**引用值**。单元也可以是**组合的**，这意味着这个单元是由一个或多个可以被分别访问的子单元所组成。访问子单元的方法是：对于串和数组是**求子串**、**求下标**和**求片**，对于结构则是**选择**。单元可以是**只读的**。这

意味着在接收一个初值后，此单元的内容将得到保护(静态地)，其内容不得修改。只能通过访问这样的单元从其中得到值，而不能存放一个新的值进去。

若附到一个单元上的模式是动态的，则该单元称为**动态模式单元**。

单元除了它的模式外，还有下述性质：

- **可引用性**：单元是否存在引用值；
- **存储类别**：是否静态地分配单元；
- **区域性**：该单元是否在某个区域内被说明。

1.5 值及其操作

值是在其上定义了特定操作的基本对象。一个值或者是一个（CHILL）**有定义的值**，或者是一个（在CHILL意义上无定义）**未定义的值**。未定义值的使用导致程序处于（在CHILL意义上）无定义的情况，此时认为这个程序是错误的。本书指明了这些情况。

CHILL允许在需要值的上下文处使用单元，这时，访问单元就可以取得其所包含的值。

值附有**类别**，类别用于相容性检查。

一个值可以是**字面值**，在此情况下它标志一个与实现无关的、在编译时已知的离散值。值可以是**常量**，这时它总提供同一个值，即它只需要计算一次。**字面值**和**常量**值都假定是在运行之前就计算好了的，因此不可能产生运行时异常。值可以是**区域的**，在此情况下它能以某种方式引用**区域**单元。值可以是**组合的**，即由一个或多个子值组成。

用户可以通过**同义定义语句**给常量值定义名字。

1.6 动　　作

动作构成CHILL程序的算法部分。

赋值动作把（计算好的）值存入一个或多个单元中。**过程调用**调用一个过程，**内部子程序调用**调用一个内部子程序（内部子程序是这样的过程，其定义不是用CHILL语言书写的，且有较通用的参数和结果返回方法)。为了从过程调用返回和（或）建立过程调用的结果，要使用**结果动作**和**返回动作**。

为了控制顺序动作流，CHILL提供了下列控制动作流：

- **IF动作** 用于两叉分支；
- **情况动作** 用于多叉分支。分支的选择可以基于多个值，类似于判决表；
- **DO动作** 用于重复或作为括号；
- **出口动作** 用于以结构化方式离开括号动作；
- **引发动作** 用于引发特定的异常；
- **GOTO动作** 用于无条件地转移到加了标号的程序点。

动作和数据语句可以组合起来构成一个模块或begin-end程序块，它们构成一个（复合）动作。

为了控制并发动作流，CHILL提供了**启动**、**停止**、**延迟**、**继续**、**发送**、**延迟情况**和**接收**

情况等动作，或对接收表达式进行计算。

1.7 过　程

过程是一个（可以带参数的）子程序。

过程有一个唯一的定义，它确定了这个过程的功能和怎样应用这个过程。过程既可以返回一个值（值过程）、一个单元（单元过程），也可以不提供结果。第一种情况的过程调用出现在要求值的地方，第二种情况的过程调用出现在要求单元的地方，最后一种情况的过程仅仅是一个动作，只能通过过程调用动作对其调用。

为了建立调用环境和过程定义环境之间的联系，提供了参数传递机制。

CHILL中的过程是一种数据对象。这意味着过程（由定义的描述符内部表示的）可以被当作值进行操作，例如可以将它们存入单元，或者作为参数进行传送。

1.8 并发执行

CHILL允许各程序单位并发执行。进程是并发执行的单位。启动动作创建指定进程定义的一个新进程。则可认为这个进程与启动它的进程并发执行。CHILL允许具有相同或不同定义的一个或多个进程在同一时刻是活跃的。停止动作使得执行该动作的进程终止执行。

每个进程总是处于两种状态之一：即活跃或者延迟。从活跃状态到延迟状态的转化称为进程的延迟，从延迟状态到活跃状态的转化称为进程的重新激活。对事件执行延迟动作，对缓冲区域信号执行接收动作，或者对缓冲区执行发送动作都能使执行这些动作的进程被延迟。对事件执行继续动作，对缓冲区或信号执行发送动作，或者对缓冲区执行接收动作，都能使被延迟的进程重新活跃。

缓冲区和事件都是受限使用的单元，发送、接收和接收情况等操作是定义在缓冲区上的；延迟、延迟情况和继续等操作是定义在事件上的。缓冲区是进程间同步和传递信息的手段。事件只用作进程之间的同步。信号由信号定义语句定义。它们表示一些用于组合或分解进程之间传递值表的功能。提供的发送动作和接收情况动作，适用于同步和诸值之间的通信。信号被定义为“持续的”，即发送动作发送的值表一直是有效的，直到接收情况动作接收。

区域是一种特殊的模块，它用于实现对定义在区域内临界过程的互斥访问。这种过程可以用来访问由几个进程共享的区域单元。

1.9 程序结构、可见性和生存期

程序结构语句确定了程序的结构，即它们定义了在程序上下文中定义的对象的有效性。

程序结构语句是begin-end分程序、模块、过程、进程和区域。程序结构语句提供了控制单元生存期和名字的可见性的手段。

单元的生存期是指单元在程序内存在的时间。单元可以（在单元说明中）显式地说明，

或者（通过调用内部子程序GETSTACK）**产生**，也可以作为语言**构造成分**的使用结果而**隐式地说明**或**产生**。

若一个名字可以在程序的某一点上使用，则称此名字在该点是**可见的**。名字的**作用域**包括所有可见的点，即该名字在那里可以识别它所标志的对象。

Begin-end分程序既确定了名字的可见性，又确定了单元的生存期。过程定义和进程定义的作用与Begin-end分程序相同。

模块用于限制名字的可见性，以防止它们被非法使用。利用**移出语句**和**移入语句**，可以控制名字在程序不同部分的可见性。

在控制名字的可见性方面，**区域**的作用与模块相同。

完整的CHILL**程序**是一系列模块或区域，并认为这些模块或区域由一个（假想的）进程定义所包含。这个最外层进程由系统启动，程序在系统的控制下执行。

1.10 异常处理

CHILL的动态语义条件是那些（与上下文有关的）不能静态地确定的条件。违反动态语义规则将引起运行时**异常**。

异常也可以由执行**引发动作**引起，还可能由于执行**断言动作**而有条件地引起。当在某给定的程序点出现了异常，则控制将转到此异常相应的处理程序(如果存在的话)。若没有显式地规定处理程序，控制将转到由实现定义的异常处理程序（见1.11节）；否则，程序是错误的。

大多数异常都有名字。这个名字可以是CHILL定义的异常名字，或者是由实现所定义的异常名字，或者是程序定义的异常名字。对于某些动态错误（例如未定义值的使用）可能很难产生异常。当出现这些错误时，则认为程序是错误的。

1.11 实现任选

CHILL允许有**实现定义的整数模式**、**实现定义的内部子程序**、**实现定义的进程定义**和**实现定义的异常处理程序**。

实现定义的整数模式必须用实现定义的模式名字来标志。这个名字应该用CHILL中没有规定的新模式定义语句作出定义。在CHILL的语法和语义规则的范围内，允许把现有CHILL定义的算术运算扩充到实现定义的整数模式中去。实现定义的整数模式的例子是**长整数**和**短整数**。

内部子程序是其定义未在CHILL中规定的过程。它可以有比CHILL程序更通用的参数传递和结果返回规则。

内部进程名字是其定义未在CHILL中规定的进程名字。CHILL进程可以和实现定义的进程结合或者启动这些进程。

实现定义的异常处理是附属于假想最外层进程定义的处理程序。若在出现某异常后控制转移到该处理程序处，此时由实现决定应做什么动作。

1.12 相容性规则摘要

在一个程序上下文各个地方都出现单元与（或）值。就所有这些情况而论，对它们的性质提出了某些特定的要求。因为单元和值的性质主要由它们的模式和类别确定，所以要根据相容性规则来描述这些要求。关于这些相容性规则的准确的、详细的叙述，读者可参考建议Z. 200（1980）中的第九章。本手册第12章给出了这些规则的摘要。

1.13 Z. 200 1984年版本

在CCITT 1981—1984研究期内，CHILL又扩充了新的内容。这个期间的工作也包括对Z. 200中某些部分进行改写，但不影响现存的合法的Z. 200（1980）程序的有效性。这个用户手册是根据Z. 200 1980年版编写的，并且根据所谓勘误校正表（COM XI 301，1983年8月，见第19章）进行了修改。下面给出Z. 200(1984)对CHILL所作的最重要的扩充：

- 为改进动态存储分配(使用“堆”的概念)，给出了新的内部子程序。
- 设置了输入输出机制，包括一个灵活的、可扩充的模型来处理“外部对象”，并且把这些与CHILL表示法联系起来。
- 便于用多种方法对CHILL程序进行分块编译的结构。包括：

— 提供由用户来控制识别名字的新的名字规则。

— 程序块的静态性质的规定：**说明模块**、**说明区域**和**上下文**。

— 指明在其它地方寻找程序块的机制。

在下一个研究期（1985—1988）内，预期会对CHILL进行扩充，实现对浮点数值的操作以及处理时间和超时的结构。