计算机软件开发系列丛书

系统程序使用TURBO C

林烟桂 编著

学苑出版社

希望

计算机软件开发系列丛书

系统程序——使用 Turbo C

林烟桂　编著
郝阿朋　改编
燕卫华　审校

学苑出版社
1994

(京)新登字 151 号

内 容 提 要

本书详细讲述了计算机理论、BASIC、数据处理等方面的内容。每章均附有习题，以便读者能在学习每章后，能进行自我测验，从而进一步加深理解，达到融会贯通的学习效果。本书内容详尽，结构严谨，适合于从事计算机专业的师生学习和参考。

欲购本书的用户，请直接与北京 8721 信箱联系，邮编 100080，电话 2562329。

版 权 声 明

计算机软件开发系列丛书

系统程序——使用 Turbo C

编　　著：林烟桂
改　　编：郝阿朋
审　　校：燕卫华
责任编辑：甄国宪
出版发行：学苑出版社　　**邮政编码**：100036
社　　址：北京市海淀区万寿路西街 11 号
印　　刷：双青印刷厂
开　　本：787×1092　1/16
印　　张：28.875　　**字　　数**：676 千字
印　　数：1～5000 册
版　　次：1994 年 2 月北京第 1 版第 1 次
ISBN7－5077－0779－2/TP・11
本册定价：49.00 元

学苑版图书印、装错误可随时退换

目　录

第一章 导 论

1.1 何谓系统程序

计算机系统包括硬件及软件。硬件指计算机的物理实体，如主机、键盘、终端设备、磁盘驱动器、打印机等。软件指程序，又分为系统程序及应用程序两类，如下图 1.1。

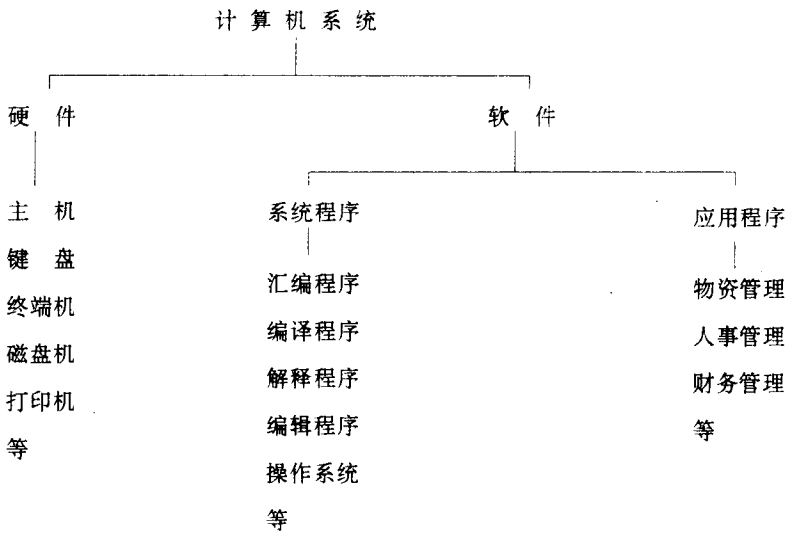

图1.1 计算机系统

系统(system)程序是由为达到一特定目标而相互作用的一组过程所组成。这里所谓过程(procedure)包括输入、存储、比较、控制以及输出等。因为系统要达到特定的目标，所以应权衡利弊，以达到最佳效果。在设计中，各过程的小目标有时会发生冲突，这时，就要以整体的目标为重，以决定取舍。

系统程序(system program)的目标主要有以下两方面：

(1) 使一般人容易使用计算机。
(2) 使一般人有效地利用计算机系统资源。

系统程序一般是购买机器时，随机携带的软件，例如：

- 操作系统 (operating system)
- 汇编程序 (assemblier)
- 解释程序 (interpreter)
- 编译程序 (compiler)
- 编辑程序 (editor)

- 调试程序（debugger）
- 连接程序（linking loader）
- 排序/合并程序（sort/merge）
- 程序库（program library）

应用程序是指计算机使用者自行开发解决问题的程序，这些应用程序使用系统程序所提供的功能。系统程序侧重于控制指挥计算机系统的资源，而应用程序偏重于解决问题，要使用计算机的系统资源一般要通过系统程序的帮助才行，所以，系统程序可以看成是应用程序设计与计算机系统的介面(interface)。如图 1.2。

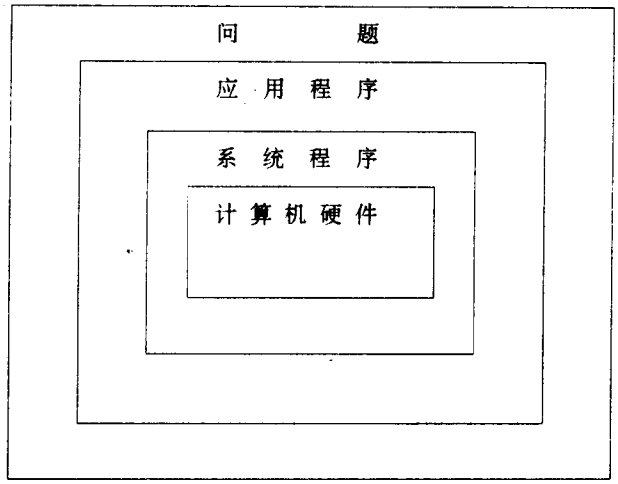

图1.2 计算机软硬件关系图

系统程序包括很多内容，举例说明如下：

· 操作系统

操作系统主要的程序有：存储管理程序，处理机(CPU)管理程序，输入输出管理程序，文件管理程序以及一些公共程序等。计算机操作系统是由一组程序所组成，其目的是为管理计算机系统资源，如中央处理机、内存、外存、显示器、键盘、打印机等，以便使使用者更容易操作计算机和更有效地利用系统资源。目前较常用的操作系统有 MS－DOS、OS/2、PC－DOS、UNIX 等。

· 汇编程序

汇编语言是由机器语言演变而成的一种符号语言，是由一些简便缩写的助记符所组成。属于低级语言。

· 解释程序

解释程序是一个解释高级语言的程序，被解释的高级语言源程序将被翻译一句执行一句，例如 BASIC。本书中的 BasicX 也属于解释程序。

· 编译程序

编译程序是将高级语言如 COBOL、PASCAL、BASIC、C 等编写的源程序转换成机器语言程序，然后，再连接执行。高级语言较为接近人们的自然语言和数学语言，容易了解、学习、维护，因此为人们所喜爱。

·编辑程序

编辑程序用来帮助使用者建立程序或数据文件。通常从键盘上键入程序或数据，然后存入磁盘或磁带，以便以后使用。编辑程序有行编辑(line editor)和全屏幕编辑(full screen editor)两种。其中，全屏幕编辑和硬件有密切关系。

·调试程序

调试程序对某一可执行的目的程序加以控制性调试，以便查找其中的错误。

·连接程序

连接程序将若干目标程序模块连接起来，成为一个可载入内存执行的程序文件。

·排序/合并

合并是指将数个数据文件依某一顺序合并成一个数据文件。排序是指某一数据文件依某一栏(或数栏)的数据顺序建立起另一新的数据文件

·库

将许多已写好的程序集合在一起，规定各程序的名称及调用方式，这种集合称为程序库。例如常用的数学库中包含有平方根函数、对数函数、三角函数、伪随机数等。

·其它

通常计算机系统软件是由计算机厂家或计算机公司所提供。由厂商提供的软件因需求不同而异，种类凡多，不胜枚举。上面说明的只是一些常用的而已。

1.2 系统程序的演变

计算机系统起源于第二次世界大战期间，第一部电子计算机是当时机械计算器(mechanical calculator)的延伸。当时有两个新概念：

(1) 想将操作机械式计算器的按键程序储存在某一机器中。

(2) 不改变硬件，只改动程序就可解决问题。

存储程序如果可行，则可用它来进行重复运算，若用人来进行这些运算是相当复杂的，而将指令存入机器内，就使重复性运算变得相当灵活、快速。在一九四四年，美国哈佛大学建造了第一部内存储式计算机，名为 Mark I，它和现在计算机一样是电子元件储存程序，并可以循环计算，但却有两大缺点，其一为指令存于低速的电子式机械继电器(relay)，其二为对每个问题，机器必需重新插线，因为指令放在硬件中，有时需要数周时间插线。

在四十年代末期，John von Neumann 先生于 ENIAC 计划中提出不改变硬件，只改变程序的观念，认为数据和程序都储存在存储器内，其表示法没有什么不同。根据此观念建造的机器，可以轻易地修改程序，因此解决一个问题就相当于产生一个程序(program)，也就是编写一连串的指令以控制机器。现在，大部分计算机系统仍沿用此观念，只是变得容量更大，速度更快了。

当时准备程序是一项相当艰难的工作，指令一般经由面板上的开关逐个地载入计算机内存，不久则演进为将程序存储在纸带上，使用者先利用面板开关键入一个小的载入程序，再利用该载入程序载入纸带上较复杂的程序。因此要花很多的时间。由于人们不容易想像机器码(machine code)，因此常发生错误，即使发生错误，也要化很长的时间来修正错误。这

种情况实在令人难以忍受，因此发展出两种重要的程序：

(1) 汇编程序。
(2) 监督程序。

汇编程序属于系统程序，用来协助使用者快速地写出指令。它解决了两个问题：

(1) 以助记符取代索然无味的机器码。
(2) 修改程序后重新汇编，会汇编成不同的机器码。

监督程序逐渐衍变成我们所熟悉的操作系统，它提供了一些简单而实用的命令(command)，例如由键盘键入一个程序，显示内存中的内容，修改指定存储器单元内容，经由外界媒体载入程序，将内存内容存到外界媒体，启动程序执行等功能。早期的外界媒体为纸带或磁带。由于监督程序的存在，使得人们比较容易使用计算机系统。

用汇编语言设计程序，其设计速率慢，容易错，不易维护。于是提供了较佳的程序语言，就是目前普遍使用的高级语言，如 PASCAL、C、COBOL 等，其编译程序可将使用者设计的高级语言程序转化为机器代码。用高级语言设计程序，其设计速率快、不易出错，且易于维护，但高级语言的一个语句常要转换成数个或数十个机器指令，因此程序会大好多倍，占用存储空间较多，执行速度较慢，这也是为求快速设计所付出的代价。

解释程序是另外一种高级语言系统，它可以不经机器码的转换，直接执行程序。解释程序直接分析每一语句，看它要做什么，然后完成它。解释程序通常是交互式的，较容易学习使用，但其执行速度较编译后的机器码程序要慢。

在改善程序开发环境方面，最主要的发展是提供了行编辑能力，这一观念早就被提出，但因那时打孔卡片的输入方法相当有效，而且也无法接受使用者占用宝贵机器时间进行编辑程序，因此在当时并没有被大量采用。一直到六十年代后期，行编辑才盛行起来。当程序变得越来越大时，人们发现需要发展一种方法，将不同人所设计的不同程序片段结合在一起，其方法是采用重定址码。汇编程序时，产生的机器码地址不予固定，即为可重定码。连接程序负责把所有有关的可重定位码，依相关顺序结合在一起，并确定其正确的地址。其实连接程序也是操作系统的一部分。操作系统主要提供程序开发的一个工作环境，它的功能包括在储存介质和内存间移动程序，为程序准备执行空间，让程序分享 CPU，管理输入输出，提供文件管理等。

早期计算机系统上监督程序的功能和今天微机上的只读存储器中的监督程序很类似。接下来的发展过程提供了良好的磁带或磁盘的单用户者批处理操作系统(batch operating system)，它们的功能和现代的 CP/M 磁盘微机操作系统类似。一直到 1972 年，软盘机，价廉物美、快速、高容量、直接存储等优点，大大地改变了人们对小型操作系统的看法。利用这种系统，人们可以快速地输入大型程序，缩短了程序开发的时间。

以上系统程序的发展目的在于使计算机更容易使用。但另一个同样重要的目标是更有效地使用计算机系统的资源，这引导着大型操作系统的发展。早在 60 年代初，研究人员在系统程序方面做了相当多的工作，当时发展的重点在多任务操作系统(multi－tasking)，也称为多道程序设计系统(multi－programming)，它是让 CPU 轮流处理多个相互独立的工作，看起来好象这些工作在同时进行。若有十个工作，则每个工作可轮流使用十分之一的时间。主

要的目的是让批处理更有效。多道程序的方式使得以计算为主和以输入输出为主的工作得以同时进行，例如计算机系统同时接受同时接受好几个小型批工作，当其中之一执行输入工作时，CPU 就启动某些特殊的硬件去完成该工作，而 CPU 交给另一个要执行计算的工作，当原先的输出输入工作完成毕时，CPU 又可交还给该工作继续进行。从硬件角度看，要支持这种处理方式，良好的中断系统(interrupt system)就非常重要了。

在 60 年代后期，商业化的分时系统(time sharing)开始出现，在这些系统里有一些功能是用来管理终端机的，使用者可以借助于终端和计算机沟通。分时系统的主要优点在于借助于改善程序开发环境来提高程序设计师的生产能力，使用者可经由终端机将程序交给计算机编译、解释或汇编，若计算机的工作负担不大，则可在短时间内得到响应结果，再利用行编辑程序修改测试的源程序，然后再传给计算机处理，而再不需要卡片机等机械媒体，因而大幅度地缩短了程序开发的时间。若计算机的负担太大(太多的终端在同时工作)，则系统反应时间就很长。

对于大型计算机系统而言，多道程序操作系统是传统系统发展的理想，将来的研究发展应着重于结合优良的硬件及软件，以创造更快、更大、更有效、更廉价的计算机系统。

1.3 系统程序的展望

经过几十年的努力，某些重要系统程序的理论和结构已经相当完备，包括编辑程序、汇编、解释、编译的程序，以及如何把中断信号和操作系统结合、磁盘管理系统的建立、单个 CPU 建立多道作业系统等均已相当成熟，但仍然有一些新领域尚待开发，例如：

- 由多个 CPU 组成的计算机操作系统。在单一晶片中包含多个 CPU，这类系统可应用在计算机繁重的工作上、或绘图、语言识别系统等。
- 并行高级语言(concurrent high level language)。这类语言，同一程序中的不同部分可同时并行地在多个或单个 CPU 上执行。
- 自动化程序设计(automatic programming)。人们一直希望有一天能经由正规描述(formal description)软件而自动产生操作系统及编译程序。同样希望经由自然语言而非正规描述方法而直接产生应用程序。
- 软件生命周期(software lifecycle)控制系统，它能指导使用者如何建立与维护他的程序，利用由上而下(top－down)的设计方式，举例说明让使用者可以设计一些低层次的逻辑程序，建立测试文件等。

在小型系统的领域里，从大系统修改而安装在小系统的趋势将会继续。将来有两种计算机将会形成主流，第一种是高性能个人电脑或工作站，这个市场定会不断增长，因为使用者喜欢独自占用一个计算机系统。并可连接区域网络或国际性网络。第二种是多用户微机系统，它们的功能已和现在的个人电脑不相上下，但价格却低的多，这类机器将被广泛使用于商业上。

习题一

一、何为系统程序？

二、解释与编译有何不同？

三、请说明 John von Neumann 在 1940 年后期提出什么观念？

第二章 语言表示法

2.1 语言表示法

每种语言均有一套语法(syntax)。符合语法为正确,否则为语法错误。描述语法最常用的方法为 EBNF(Extended Backus Naur Form)。BNF 因描述程序语言 ALGOL 60 而著名,后来扩展到对其它语言的描述。例如英文语句(sentence)可以描述为

<sentence>::=<subject><predicate>.

象 BNF 用特定符号来描述另一种语言的语言,称为元语言(metalanguage),BNF 所使用的符号有三种:

(1) ::= 定义为。
(2) | 或。
(3) { } 重复发生 0 次、1 次、2 次、…、无穷次。

后来为了描述方便起见,增加了三种,总共六种,称为 EBNF。

(4) <> 非终端符号 (no—terminal symbol)。
(5) [ ] 任选(optional)。方括号内的语法成分可选可不选。
(6) \\ 必须选一种,且只能选一种。

例 2-1:英语简单语句由下列三条语法规则所组成。

P1 <sentence>::=<subject> <predicate> 。
P2 <subject>::=lions | cats
P3 <predicate>::=cry | fly

语法规则 P1 描述一英语简单语句,定义为主语<subject>其后跟表语<predicate>最后跟句号。合乎 P1 则为正确,否则为错误。语法规则 P2 说明主语为终端符 lions 或 cats 二者选一。语法规则 P3 说明表语为终端符号 cry 或 fly 二者选一。因此英语简单句 <sentence>只有下列四种是合法的,其余为错误。

lions cry。
lions fly。
cats cry。
cats Cly。

2.2 语句分析

辨认语句是根据输入的语句,导出产生该语句的造句程序。一般称此工作为语法分析

(parsing)。其工作通常复杂而困难,有时甚至是不可能作到的。而用那一类型的造句程序来定义语言是影响分析工作是否困难的最主要原因。常用采用的是由上而下(top down)的分析方法。此法由起始符号找出产生整个语句的一系列造句程序。

例如要分析

lions cry.

是否合乎英语简单句,从<subject>起始符号集合{lions,cats} 开始,注意这里的花括号表示集合(set),而不是 EBNF 记号。集合中共有两个元素,lions、cats。由 P2 知非终端符号<subject>可以直接产生终端符号 lions。由 P3 得知非终端符号<predicate>可以直接产生终端符号 cry。所以可知 lions cry。为合法的英语简单句。以分析树(parsing tree)表示如下图 2.1。

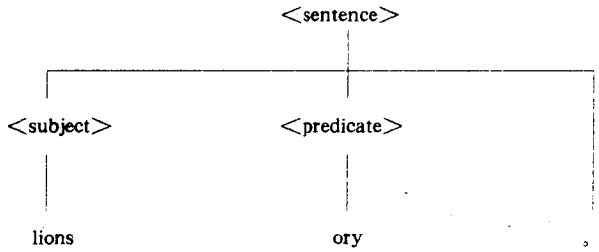

图2.1 英语简单名<sentence>的分析树

例 2-2: S 语言以 EBNF 定义如下。请分析字串 XYZ 是否合乎 S 语法。

P1 <S>::=x<A>
P2 <A>::=z | y<A>

解:

<S> XYZ
X <A> X YZ 依 P1,<S>::=X<A>
<A> YZ
Y<A> YZ 依 P2,<A>::=Y<A>
<A> Z
Z Z 依 P2,<A>::=Z

故知 XYZ 为合乎<S>语法。其分析树如图 2.2 所示。

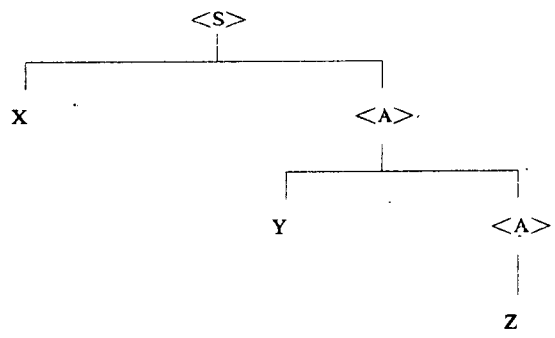

图2.2 由上而下分析XYZ是否合乎<S>语法

例 2-3：<Condition>语法以 EBNF 定义如下。请剖析字串 ab<cc 是否合乎<Condition>语法。

P1 <Condition>::=<Id>\<|=|>\<Id>
P2 <Id>::=<Alpha>{<Alpha>}
P3 <Alpha>::=a|b|c|d

解：

<Condition>		ab < cc	
<Id> < <Id>		ab < cc	依 P1
<Alpha><Alpha>	< <Id>	ab < cc	依 P2
a<Alpha>	< <Id>	ab < cc	依 P3
<Alpha>	< <Id>	b < cc	
b	< <Id>	b < cc	依 P3
	< <Id>	<cc	
	<Id>	cc	
	<Alpha><Alpha>	cc	依 P2
	c<Alpha>	cc	
	<Alpha>	c	
	c	c	依 P2

故知 ab < cc 为合乎 <Condition> 语法。其剖析树如图 2.3 所示

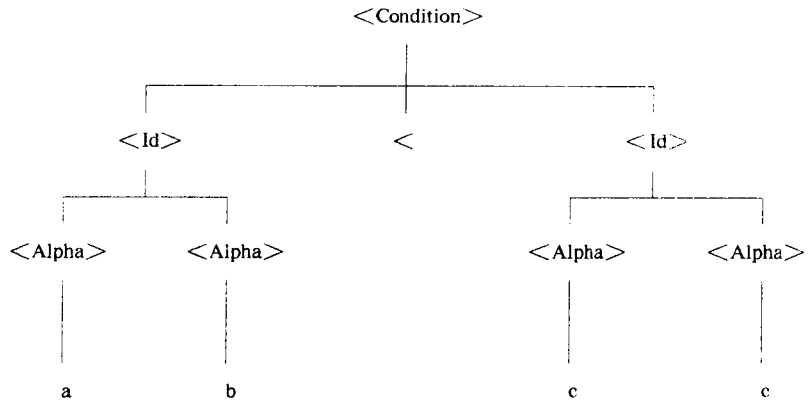

图2.3 由上而下分析ab<cc合乎<Condition>语法

例 2-4：T 语言以 EBNF 定义如下。请剖析字串 XXZ 是否合乎 T 语法。

P1 <T>::=<A>|<B>
P2 <A>::=x<A>|y
P3 <B>::=x<B>|z

解：

(1)

<T>	xxz	
<A>	xxz	依 P1，<T>::=<A>
x<A>	xxz	依 P2，<A>::=x<A>
<A>	xz	
x<A>	xz	依 P2，<A>::=x<A>

<A>	z	
?	xxz	不合乎<T>语法

(2)

<T>	xxz	
<B>	xxz	依 P1,<T>::=<B>
x<B>	xxz	依 P2,<B>::=x<B>
<B>	xz	
x<B>	xz	依 P2,<B>::=x<B>
<B>	z	
z	z	xxz 合乎<T>语法

在分析过程中,<T>的转换若选<A>,发现字串 xxz 不符合<T>语法,若选<B>则为合法。问题在于语法规则 P2 与 P3 的起始符号均为 x,因此在剖析过程中若下一个字元为 x 时,无法判断要使用那一条语法规则,导至分析错误。这种情况应该避免。在定义语法规则时若能遵循下述规定,自然能避免此种错误。

规定 A:

若有语法规则

<A>::=<A1>|<A2>|<A3>|...<An>

则所有能从 <Ai> 产生的语句的起始符号集合的交集必须是空集合。要取得 <Ai> 的起始符号集合,可用下列两个规则。

规则一

若起始符号即是终端符号,则为该终端符号的集合。即

<Ai>::=x<B>
start(<Ai>)=start(x<B>)={x}

规则二

若起始符号为非终端符号,则为所有非终端符号的起始符号的合集。

若

<A>::=<A1>|<A2>|<A3>|...|<An>

则

start(<A>)=start(<A1>) ∪ start(<A2>) ∪... ∪ start(<An>)

例 2-5: T 语言定义如下。请求出 <T> ,<A> ,<B> 的起始符号集合。

P1 <T>::=<A>|<B>
P2 <A>::=x<A>|y
P3 <B>::=x<B>|z

解:

start(<B>)=start(x<B>) ∪ start(z)
={x,z}

此处大括号表示集合

start(<A>)=start(x<A>) ∪ start(y)={x,y}