Microsoft C6.0、Quick C2.5

高级C程序设计技巧与捷径

陈 星　贾若芊
李 红　辛 俭

本书是缩短学习高级C语言途径的特殊工具集专家编程的技巧和实例

北京希望高级电脑公司

前　言

这本书有两个相关目的：一是教会读者C语言，二是展示如何使用C语言在PC机上编写各种各样的程序（PC机是泛指IBMPC，XT，AT，PS/2系统和其它厂家生产的兼容机）。

为什么这本书以Microsoft C为基础呢？因为Microsoft C编译器为PC机最广泛使用的专业化软件开发工具。它被认为是该领域的标准软件。它的最新名称为Microsoft专业开发系统（PDS），在该书中我们也这样引用。Quick C不但继承了许多PDS的优良特性，而且是PDS的简化、快捷的翻版。这本书对PDS和Quick C均适用。

本书针对的对象

本书适用于想在PC环境下学习C语言编程的每一个人。无论你是一个软件开发员，一名专业编程人员，学生或者只想编编程序的人，这本书也可作作普通学校、大学和企业中的计算机科学教程。

这本书的独到之处：许多介绍C语言的书是基于UNIX环境或理论性文章的，而忽视了运行该语言的机器。而该书采取了不同的方式：它联系PC机和MSDOS（或OS/2）操作系统教授C语言，这种方法具有以下优点：

第一，把目标集中于特定的硬件上使得学习语言更容易、更有趣，在UNIX环境下运行的C语言一般不具有标准化的图形处理或其它与硬件相关的辅助设备，因此编程举例只集中于text-based interation。但在PC领域内，正好相反，我们可利用诸如特定的正文符号、光标的编程控制和彩色图形等特性。

此外，当我们面向C语言更复杂的一面时，C语言的结果可通过与PC机上的实际运行相联系，便可解释得更清楚，虽然C语言结构从某些方面看具有一定的理论性和神秘性，如远指针和联合。

最后，如果你的目标是编写PC系列机上的特定程序，那么在PC环境下学习C语言，结合例子使用PC机硬件的特性将会给你编制实用程序一个良好开端。

在实际动手中学习

我们已经在该书中学习了hands-on方法。我们相信自己亲自去试试是学习的最好的方法，因此所举的例子都是安全可运行的程序。我们给出了实例程序的运行结果，这样你可确定一下你键入的程序运行是否正常。在本书开头只给出了很短的程序，在书末给出了运行稍长些和更复杂些有技巧性的程序。除了要教会你使用语言之外，这些例子还可充作你自己的技能实践和发展的开端。

图示与练习

我们试图使这本书具有“可视性”。许多编程概念用图示可进行透彻明了的解释，所以在任何可用图示的地方，我们都用图示方法对正文进行辅助说明。

每章包括了一系列问题，用以测试你对本章范围内概念的理解程度，还有一些编程练习以帮助你理解如何把概念应用于实际工作中。所有问题和练习的答案均可在书后找到。

为什么使用C语言

在前十年间,C语言已成为在PC机上编写各种程序的占绝对优势的语言。为什么？因为C语言在编程语言中是独一无二的，它在象BASIC和PASCAL这类高级语言一样为用户提供方便的同时还可象汇编语言那样允许用户紧密控制硬件和辅助外围设备，许多可在PC上用汇编语言执行的操作也可用C语言完成，而且更加便利。

当然，C语言还有其它的优点。一个较好的C语言编译器（特别是Microsoft编译器）可产生令人惊奇的快的代码。这种代码如此高效，以至于比再用汇编语言重写来提高速度有明显的提高。C语言还是一个具有优良结构的语言；利用其文法，可以容易地编写programs that are modular，因此也容易理解和维护。C语言还包括许多帮助编制大型的或复杂的程序的而设置的性能。最后一点，C语言比其它许多语言更“轻便”、程序可很容易地从一个系统传送到另一个系统。

读此书之前你应知道什么

在读此书之前，你应具有使用象BASIC和PASCAL一类高级语言的经验。把C语言作为你学习的第一门计算机语言当然可以，但C语言更适合于有经验的编程人员。它的语法——至少这是首先要看的——不象BASIC那样的好，而且它包括一些对初学编程的人来说较难理解的概念。尽管如此，本书几乎没有考虑你从前有没有经验。一切概念都是从最基本开始讲起。

你应该熟悉MS-DOS操作系统（IBM称作PC-DOS）或OS/2。你应理解目录树结构，并能在目录树上搜索。你还应能够创建、拷贝和删除文件和运行命令行程序。

Microsoft编译器系列

Microsoft具有四种不同的C语言开发platforms，PDS是最晚的产品。PDS具有两种开发platforms。第一个是窗口开发环境，叫作 programmer's WorkBench（PWB）（编程者工作台）。它利用窗口和菜单形式提供了一个直观形象的界面。所有的开发处理目标，其中包括编辑、编译、连接、运行、纠错和帮助信息都在这个环境中统一成一整体。这不但简化了程序开发，而且比那些每一步不同的开发都调用独立的命令行程序的老办法易学。

在PDS第二个开发控制台中，仍采用了这种命令行方式。从命令行执行中一个叫做CL的程序，就可编译和连接你的程序，它和Microsoft编译器老版本方式一样。实际上，PWB通过调用CL进行编译和连接，然而它对用户来说是不可见的。

Quick C也带有两个开发控制台：一个形象直观的界面，虽然该界面性能少些，但仍很象PDS中的PWB；另一个是叫作QCL的命令行编译器。

PDS和Quick C在处理源代码的方式上基本相同。如果能在PDS中运行的程序，在Quick C中也能运行，反之亦然（我们还将列举出一些小的例外）。

我们将解释如何使用PWB、Quick C以及两者的命令行形式。

该书基于Microsoft C专业开发系统（PDS）6.0版本和Quick C2.5版本。但书中列举的例子也可在这两种编译器上更低一些的版本上运行，当然也可在未来更高级的Microsoft编译器控制台上运行。

实际上，该书中大部分内容也适用于Microsoft系列之外的其它的编译器，有些部分是Microsoft编译器所特有的，这包括第一章中设置过程的讨论和第十二章中关于Microsoft图形函数的材料。

使用这本书所需设备

在这一段里，我们概述了运行PDS和Quick C以及编译本书中的例子所需要的机器设备。

硬件

你应使用一台带有8088、8086、80286、80386或80486微处理器的IBM PC，XT，AT，PS/2或其兼容机。对于PDS，Microsoft用80286、80386或80486。

你的机器还要具备至少10MB硬盘。如果你要使用PDS，你将需要8MB盘空间。Quick C至少需要2.7MB。驱动器的速度也是一项因素。凭我们的经验，如果你打算使用建于PDS中的编程员工作台，一台快速的硬盘驱动器——就象置于80386、80486一级机器的一样，将是非常理想的。

PDS要求至少512K内存，而Quick C需要448K。这样大的容量对于使用这本书来讲已足够了，但对于更大的课题项目来讲还是不够的，PDS要1M，Quick C要512K。如果你使用的是OS/2，你则需要至少3M，最好是4M内存。

PDS需要一台1.2M 5¼ 英寸的磁盘驱动器，或一台720K3½ 时的磁盘驱动器，以装载distribution disk（它不是分配为360K软盘）。Quick C可装配360K5¼时磁盘。

对于该书的大部分内容来讲，单色的显示器也可工作良好。但第十一、十二章是关于彩色图形的内容，这需要彩色图形适配器和彩色显示器。任何一种普通的适配器均适用：CGA、EGA或VGA。一些例子需要EGA或VGA，当然这样的图形比CGA更美观(CGA将很快过时）。

最后，你可能还需要一台打印机，用以打印程序清单或记录程序输出，然而这不是绝对必要的。

软件

你需要MS-DOS3.0以上版本支持PDS，或2.1以上的版本支持 Quick C。比这些低的MS-DOS版本不能工作。

若你使用OS/2，则需要1.1以上的版本支持PDS的运行。在OS/2（“DOS”）的功能范围内可运行PDS和Quick C以及本书中的所有例子程序。PDS也可以作为OS/2的一个字符型可用程序来运行——实际上，Microsoft的编译器是用于OS/2的最受欢迎的软件开发控制台。Quick C不可作为OS/2的保护型应用程序来运行。

早期的Microsoft优化编译器（PDS先驱）要求用户自行设置编辑器或字处理器来进行源代码的编写工作。但是现在，PDS中有PWB，它内部都包含有一个已建编辑器。如果你不喜欢这编辑器，或者你在使用另一个编辑器，那么可以把它合并到 PWB 中，或者只独立地使用它。Quick C中也包含有自己的编辑器，该编辑器与PDS中的相似。

选择开发环境

你需要决定使用什么硬件和软件进行C语言程序开发，下面是一些选择内容：

直观的界面或命令行

直观友好的PWB和Quick C的界面比使用命令行编译器方便得多，因为这样所有的操作均可通过菜单被激活。命令行方式提供了极大的灵活性和有效性，也是大型、复杂的开发环境所必不可少的。与PWB相比，它也快得多。但是，为了学习C语言，你可能想要一个PWB或Quick C的直观界面。

PDS或者Quick C

为了配合本书的阅读，应该使用PDS还是Quick C？PCS是PC上使用最广泛的专业化软件开发工具。它产生高度优化代码，可以适应大型、复杂的程序，内部包括有极优良的运行时间库，具有各种广泛的用途。还可以编译在OS/2中运行的程序。

Quick C则不会产生如此高度优化的代码，缺少许多建于Microsoft C中的可选项，这中间包括有编译OS/2中的程序的功能。

Microsoft C 和 Quick C 都可编译本书中的例子程序，也包括有最终目标（general-purpose）编译器。它们均允许你在开发环境内编写，编译，运行和调试你的程序。如果你是一名学生，Quick C将能很好地满足你的需要。如果你正在一个专业化开发环境内工作，你也许想要使用Microsoft编译器，这样你就有条件进行各种各样程序的开发。有些编程员将两种控制台Quick C用于初级开发，而POS用于最后的编译。

MS-DOS或OS/2

这本书主要是针对在MS-DOS环境下使用。但是，正如我们已提到的，Microsoft编译器和Quick C都可在OS/2兼容逻辑框（Compatibility box）内运行，并且这本书中的例子程序也在其兼容逻辑框内运行。这样，即使在OS/2中，你也可以很好地使用这本书。

如果你所感兴趣的是在OS/2保护型模式下而不是在兼容逻辑内进行C语言编程，这本书仍不失为一本有价值的介绍性书籍。但是，一些技术，尤其是将在第十和章十一中讨论的直接进入图形存贮的问题，不和OS/2的保护型模式兼容。我们建议：在你把范围扩大到OS/2的保护型模式编程范围之前，应使用兼容逻辑框来运行书中的内容。

C语言的演变发展

C语言是在70年代初期由Dennis Ritchie在贝尔实验室研制成功的（它的前身是叫作 B 的一种语言。）在1979年Brian Kernighan和Dennis Ritchue发表了一本书——C 编程语言，其中提出了至今仍在延用的定义。在过去的几年里，一个C 语言的新定义从美国国家标准协会诞生（the American National Standards Institute）：ANSI C 标准。这个标准被认可为C语言的权威性标准。多数编译器包括PDS和Quick C，已经被修正为与 ANSI C 兼容的形式（也还有少数例外情况）。

“传统”的Kernghan和Ritchie的C语言与ANSI C之前的变化可能不会给初学者带来多大的影响。ANSI C中最明显的变化是原型（proto typing）的出现——它是规范的函数说明。

本书已进行了修订，以反映C语言的新ANSI方式。但是，你也可以运行在ANSI C 产生之前所写的程序。在有关的地方，我们将会提到在C语言旧形式中的方式方法。

本书的内容安排

本书的前七章囊括了C语言的基础知识。在第一章中，你会写出一个非常简单的程序，编译它，连接它，并且运行它。这会给你形成一个概念：C语言是什么样子，如何去使用它。第二章讨论了在你过渡到较大的程序前的一些初步知识：变量，简单输入／输出语句和运算符，如（=）和（+）。还用了例子程序针对上述不同的讨论点进行说明。第三章和第四章包括了C语言结构的基本知识：循环和条件判断。第五章中描述了函数，第六章讲了数组，第七章解释了指针是C语言中最广泛使用的一种结构。C语言比其它语言更多地用到了指针。

在第八章中，我们的注意力由C语言转向PC硬件系统，我们更仔细地看了看键盘和显示

器的工作运行情况，以及C是如何得到这些设备的控制权的。第九章讨论了C语言的结构和联合的构成方式的好处，并显示了如何使用联合得到建于PC硬件中的重要例行程序：只读存储器BASIC输入/输出系统ROM BIOS)。第十章中讲到了单色显示器，在第十一章中说明如何使用ROM BIOS和直接进入图形存贮区来形成彩色图形。第十二章中描述了Microsoft的图形库。

第十三章中包括了文件以及它们在PC环境中的磁盘和固定磁盘系统方面的用途。第十四章中探讨了各种各样的问题，这些有助于更大型的、更复杂的程序的开发，第十五章讨论了变量的一些优点而结束全书。

我们还提供了一些附加内容来帮助你学习C语言并使这本书成为一本易学的参考书。第一个附录总结了主要的C语言语法结构。比如说，假如你忘记了如何在switch语句中使用冒号和分号的话，你可以在此查到。在附录B中包括有许多不适合放在本书主要部分的大型复杂程序。这些程序给你一个概念，那就是多么大的程序都可以用较小的程序基本构件组成，还说明了一些新的编程技术。附录C中说明了16进制数系统。十六进制在书中多处用到并且有助于透彻了解PC系统。在附录D和E中列出了参考书目和一张用于PC的字符代码。附录F中描述了PWB的CodeView debugger调试程序的操作。附录G中说明了基于Quick C中的调试性能，附录H中进一步讨论了用于PWB和Quick C中的编辑器，附录I总结了第十二章中所讲述的Microsoft图形函数。

书中印刷字体的规定

本书的正文说明部分中，所有C语言的关键字，如if，while和switch，都用黑体字表示，以使它们与这些词的普通用法区分开来。类似的，所有的C语言库函数，如printf( )，gets ( ) 和lseek()，也用黑体字，区别于用户所写的函数。变量名也用黑体字，使它们与普通正文区分开。

在文章中用到字符串常量的地方都用双引号引住，和在C语言程序中的形式一样，如"Good morning" 和 "Fatal error"。字符常量则用单引号引住，同样是依照C语言的规定：如'a'和'b'是字符。

通常由单字符组成的运算符将用括号括住说明：如（+）和（/）。

键盘键，如〔Ctrl〕和〔Return〕，用方括号括住。

程序名没有用黑体字，但名字包括有扩展名，如myprog.c、your prog.c和my prog.obj中的.c和.obj。

本版中的新内容

本版书进行了修订以反映最新知识：Microsoft C和Quick C。它与 Microsoft C6.0（又称作专业化开发系统或PDS）以及Quich C2.5兼容。描述编程员工作台（PWB）——Microsoft C6.0中的新内容的资料加入书中。有关处理MicrosoftC和Quick C中的编辑和调试问题的正文部分和附加部分都进行了进一步修订。

同样，反映新的Microsoft C特性的材料添加进书中——较重要的是图形库函数的出现。其它的增补内容包括有based指针，long double（长双型）类型，微存贮模式和新优化运算等。

最后，对一些常见的小问题和错误进行更正。这并不意味着本书没有错误了。如果你发现任何一个你认为在今后出版中应修正的地方，我们将会感谢你对问题的提出。

你一定会喜欢这本书

本书内容覆盖范围广——从简单的单行程序到复杂的图形和文件管理应用程序。我们力图使这样大范围的讨论显得容易明白，因此开始时速度慢，逐渐过渡到更多的尖锐地有挑战性的概念上。我们相信我们已成功地做到了这一点，并希望您能喜欢这本书。

目　录

第一章 起 步

本章包括几项内容。首先，我们将对在计算机上建立C语言编译器（无论是专业化开发系统（PDS）还是Quick C）的过程进行几项说明；然后，将描述你新建立的编译系统将要干些什么，即编辑—编译—连接—运行处理。还要简要地介绍用上述处理过程建立的一些文件和目录。最后，我们将脱离上述的一切，直接进入C语言本身：你将学习写第一个C语言程序，学习一些C语言编程的一般结构，特别是关于printf（）函数。

1.1 建立系统

PDS和Quick C两者都使用同一种自动安装程序，以组织和建立那些组成系统的文件。运行这个程序很简单。在A驱动器内插入设置盘（installation disk）驱动并执行SETUP。接着该程序问你一系列的问题。在屏文件（on-screen document）（或随编译器的使用手册）提供对问题的解释说明。

需要耐心

建立执行程序所涉及的许多问题和动作在你还没学C语言编程之前是不大明白的，别泄气。你不需要知道SETUP启动编程的具体细节。在本章中，我们将简单地谈谈它的一些机能，其余的部分将会随着你对C语言不断深入地研究而逐渐清楚了。

缺省应答（Default Answer）

在许多情况下，对那些正是你所需要的东西进行缺省应答，例如，按〔Enter〕键而不敲入任何东西）。这将使系统具有一定的编译程序构形，这样就会编译该书中的程序实例了。

例如，关于存贮模式问题的缺省应答将产生一个小型存贮模式的组合库（combined library），而不会产生其它模式的库（如中型、大型模式等等）。既然本书中的所有程序都使用小型存贮模式，则应当进行缺省应答。

Microsoft图形库文件

当你被要求回答是否需要包含在组合库中的库文件GRAPH.LIB 和 PGCHART.LIB时，你所进行的缺省应答有些特殊。该缺省应答是No（“N”）。但是，如果你要调用将在第十二章中讲到的Microsoft图形函数时，你的组合库中应同时包括GRAPH.LIB（用于一般的图形函数）和PGCHART.LIB（用于条形、饼形和其它种类的图形）。这样，你可能要对有关这些文件的提问回答Yes（“Y”）。

使用这些图形文件的不足之处是：它们会扩大你的组合库文件，这将占用更多的硬盘空间。如果你不缺少硬盘空间的话，这不会成为主要问题。

支持鼠标器

如果你有鼠标器，则对于已建程序中的相应问题无论如何都要回答Yes。虽然你不需要鼠标器操纵PDS或Quick C，但在许多场合，它是一个很有效的输入设备，还能加快开发处理进程。

1.2 编辑—编译—连接—运行处理

既然已建立了PDS或Quick C系统，你可能想马上开始进行编程。请耐心地再看几页，在这几页中，我们提供了程序开发过程中所发生的情况的基础知识，并引导你简要地看看由已建立的处理过程产生的文件和目录。

用已编译的语言开发一个程序至少要四步，编辑（或称为写）程序，编译程序，连接程序和运行程序。另外一步——调试（debugging）也是必不可少的，到后面我们再来讨论它。

编辑

你用人类可理解的词和符号来书写一个计算机程序，这是开发周期中的“编辑”部分。在PDS和Quick C中，你把程序直接敲入屏幕上的一个窗口中，就象在一个字处理器中一样，你可以把最后得到的结果正文作为一个独立的文件存贮起来。这通常被称为“源文件”它是一个ASCII文件；你可通过DOS中的TYPE命令读这个文件。

编译

你不能直接运行源文件。为了在计算机上运行，源文件必须被翻译成计算机中的中央处理单元（80×86微处理器）所能接受的二进制数的形式。这些二进制数被称作“机器语言”，因为计算机可识别它们。编译器的工作就是把人可读的程序形式翻译成机器可读的程序形式。编译器是一个独立的程序（实际上是一系列程序），它可以从PDS或Quick C中得到，或直接从命令行中得到。

连接

如果编译之后过程就结束了，那么问题就简单了，我们只要运行由编译产生的文件就行了。然而，在大多数已编译的语言中还包括有另外一步：一个被称作“链接”（linking）的处理过程。为什么需要链接呢？

一个原因是，许多已编译的语言（C语言是其中主要的例子）带有库例行程序，它们会被添加进你的程序当中。这些是编译器的生产者所编写的，用以执行不同的任务——从输入／输出到复杂的数学函数。它有可能提供这些函数的源代码文件并要求用户在编译自己的程序的同时，对它们进行编译。但是，更有效的方法是提供一例行程序的已完成编译的机器码形式，用户仅需要把它的程序的机器码形式与之相连即可。

在建立处理过程中创建的“组合库文件”包括了大多数编程中需要的例行程序。在链接时，来自库文件的例行程序与你机器语言形式的程序相结合。

另一个要进行链接的原因是：也许你不想同时编译你全部的程序。你可能编写了一个大型程序，但只对其中一部分进行调试和运行，而其余的部分仍有待于开发。因为编译一个程序要耗费相当长的时间（从几秒钟到几分钟，随程序的大小不等），你也许只想重新编译那些正在开发的程序部分。因此，你需要一个方法将已编译的文件链接而产生一个完整的程序。我们将在第十四章进一步讲述对程序文件进行单独分开编译的问题。

基于上述原因，编译器产生出一种中间媒介型的文件，叫作“目标文件”（object file）。连接器紧接着把所有需要的目标文件连接到一起，形成一个最终的可执行的程序。编译和链接过程的关系在图1—1中进行了简要描述。

图1—1适用于PDS。Quick C与此类似，但无BINB目录，因此LINK.EXE在BIN目

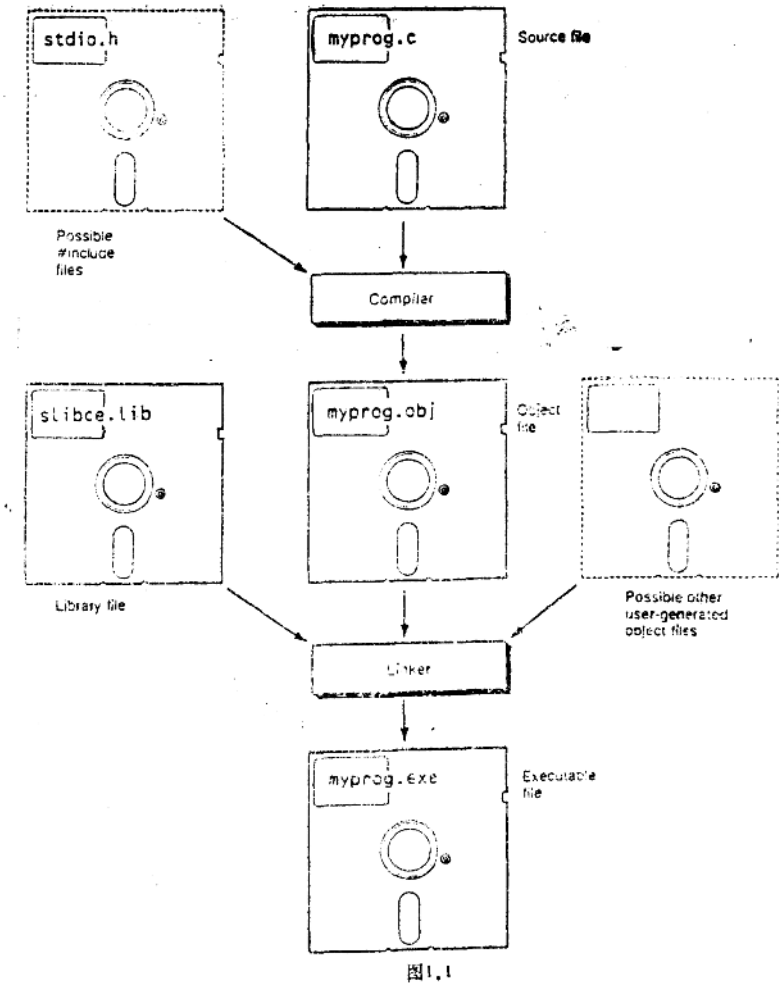

图1.1

录下。

程序文件名规则

多数版本的C语言要求你给键入的源文件加上.C文件扩展名。中间媒介——目标文件由编译器自动加上.obj扩展名，最后的可执行文件由连接器加上扩展名.exe。（系统也产生其它类型的文件，这里暂且忽略。）操作系统识别带有扩展名.exe的文件为可执行文件，这样你就可以运行它们或干别的事情，只需在DOS提示符下键入文件名。

正文编辑器产生.C源文件，然后到达编译器，产生.obj目标文件，然后到达连接器，形成.exe可执行文件。

1.3 目录和文件

如果在装入PDS或Quick C之后，你查看你的硬盘目录，你会看到一个新的目录——一个根

目录的子目录已经产生。在PDS中，它叫作C600（或一个类似的名字），在Quick C中叫作QC2（或一个类似的名字）。

在这些目录下，你会发现大量的其它的子目录。你也许会被由你的建立程序设置的文件和目录的数目震住了。这些文件干什么用？为什么有这么多的文件？这些东西太多了，我们不能详细地探讨，只能简要地涉及其中的一部分。

可执行文件

一些目录中包括你的系统要操作的可执行文件。这其中最重要的是 BIN 子目录。（"BIN"代表"二进制"（binary），是对可执行文件的另一种约定形式）。

PDS的中心程序是PWB.COM。PWB代表"编程员工作台"。（"Programmer's WorkBench"）。这是一个可在其中写程序和开发程序的窗口环境。Quick C 的中心程序是QC.EXE，它可产生相似的窗口环境。无论是PWB.COM还是QC.EXE，都在BIN 目录下。这些程序是你在启动程序时从命令行调用的。当要求这些程序去完成不同的事情时，它们将轮流调用其它的程序。

编译器是那些从PWB或QC调出的程序之一，它也在BIN目录下。编译器实际上包含有几个程序（C1.EXE，C2.EXE等等），它们控制编译进程的不同步骤。

另一个重要的程序是连接器——LINK.EXE。在Quick C中，该程序在BIN目录下；但在PDS中，它在另一个可执行文件的目录下——BINB。该目录下的程序既可以在"实际"（real）模式（普通的MS-DOS模式）也可以在OS/2的"保护型"模式下运行。（如果你用OS/2保护模式编译程序，你会发现第三个可执行文件的目录——BINP。它包括一个独立的保护型模式编译器，同其它的OS/2特定程序一样）。

另一个在BIN目录下重要的程序是命令行开发系统。它在 PDS 中叫作 CL.EXE，在 Quick C中叫作QCL.EXE。我们将在下面进一步阐述有关命令行程序开发的问题。

库文件

我们已经注意到了存在一个通常命名为SLIBCE.LIB的"组合库文件"。该文件放在BIN目录下，它内部包含有执行一系列特定任务的例行程序。C语言中这些库文件的数量和种类极其丰富。许多在其它语言中键入语言定义的处理过程（如输入／输出），在C语言中却使用库函数来处理。正如我们已提到的，库函数是由连接器自动加到你的程序上的已编译了的例行程序。LIB目录还可以包含其它文件，但SLIBCE.LIB是最重要的。

请注意，当你需要从库内调用一例行程序时，并不是所有的已编译好的库文件都加到你的程序上。如果这样，将会使每个程序都变得很大。实际上，只有那些真正需要的库文件的组件链接到你的程序上。连接器的工作之一是从库文件中抽取正确的软件组件。

已编译了的库文件可以不止一个。如果在建立过程中，你请求用其它存贮模式而不是小型模式（缺省值）进行编译，则在你的LIB库内将存贮有其它已编译了的库文件。例如，若你请求中型存贮模式程序，则将产生文件 MLIBCE.LIB。在本书中只需要小型存贮模式。

已编译了的库文件名也与你所选择的浮点任选项有关，即你想要如何执行浮点数学运算。如果你要求进行模拟仿真（是浮点任选项缺省值），那么库名用"E"结束（"E"代表"emnlatow仿真"）。在模拟仿真模式中，浮点运算由软件执行。但如果你设置了 8087 或其它数学协处理器芯片，并在建立过程中要求使用它，则库名以"7"结束，浮点运算由硬件来完成。

如果你编译保护型模式的程序（只有在PDS中可能用到），你会找到象 SLIBCE.LIB 这样的库名，其中“P”代表保护型模式（protected mode）。

包含文件（Include File）

对包含文件很难解释，除非你已真正用C语言编过程序并看到了是如何使用它的，但下面简要的介绍给你一个粗略的概念。

正如我们刚刚讨论的那样，由链接器连接到你的程序上以执行输入／输出或其它任务的库例行程序存贮在一个组合库文件内。但是编译器可能也需要关于这些例行程序的信息。这些信息也许包括调用库函数时要用到的常量和结构的定义及原型或函数的规范化说明。

你可以把这个信息敲入你的源文件，但是让编译者们把它们统统组装入一个单独的文件内会更便利些。这个文件只用一条语句便可插入你的程序中。这样的文件叫作“头文件”（header files）或包含文件（Include files，译注：以后译作Include文件）。它们都有.h扩展名。例如，许多程序包含有文件STDIO.H。STDIO.H中有关于标准输入／输出操作的各种各样的定义。类似地；如果你想用高等数学库例行程序，你应包括进文件MATH.H，这其中有关于这些函数的信息。

在你的程序中，包含有预处理指令#include一行，它可使整个头文件加到你的程序上，这同用字处理程序把一个文件装配进另一个文件的方式极其相似。

所有的include文件包含在一个子目录下，更确切地说，是在INCLUDE目录下。

随着对书中内容不断地学习，我们会涉及到更多的include文件，特别是在第5章。但是现在，记住：它们和你用编译器形成的文件一样是文本文件，在它被编译之前可以和你的程序组合。如果你好奇的话，可以用你的编辑器验证一下这些程序，或用TYPE命令看看其中到底有什么。（注意千万别进行修改）。

程序文件

最后，还有你自己要形成的文件：你键入PWB或Quick C中的文件源代码，和由开发系统产生的.obj，.exe和其它文件。你也许可能想为这些文件创立一个独立的目录，你是在这个目录下调用PWB或QC的。

在你的系统中，还有一些目录和文件我们尚未提到。例如，PDS中的HELP目录包括有你请求帮助时显示的正文文件。还有文件和例子程序的目录。但是我们已谈到了对你理解开发过程有重要意义的文件。

更多的文件

C语言中的文件和目录系统比BASIC或Pascal复杂。所有这些复杂性都是必要的吗？一般来讲，复杂性的设计是为了更有效地使用C语言。起初，你也许会觉得找到理解系统的路很困难，但一旦你了解了它，你定会欣赏它的价值。

例如，创建适用于特定开发需要的库文件可节省存贮空间，并使链接过程效率更高。如果你不需要大型的存贮模式的例行库程序，那么这些例行程序就不存入你的盘中。

类似地，使用独立的头文件意味着你不需要那些在你的程序中用不上的库例行程序的信息。这缩短了编译时间，因为编译器用不着去读许多不需要的信息。

1.4 建立编程环境

习惯上的DOS系统的管理安排可使你在用户系统中查找所需的文件以及存取应用程序的

工作方便化。这是PDS的情况，但Quick C与此很相似。

设置符号名

在一个目录下的程序对于另一个目录下的程序来说是不可见的。但是对于PWB(或QC)编译器，连接器和开发系统的其它程序来讲，能便利地寻找所需的各种文件是很重要的（如库文件，include文件和帮助文件）。通过在操作系统环境内设置符号名便做到了这一点，它是一个任何请求均可到达的公共区域。这些符号名与寻找文件时使用的路径名联在一起。

例如，拥有库文件的目录的符号名是LIB。假设库文件在目录C:\MSOFT\LIB下。（我们将用目录名MSOFT代替你所使用的建立过程产生的任何一个名字）。此时环境内有以下字符串：

```
LIB=C:\MSOFT\LIB
```

一个寻找库文件的操作（如连接器）是在环境中找到LIB目录并取出相应的路径名。紧接着便可得到库文件。

执行DOS命令SET，可把字符串放入环境内，例如。要设置下面的字符串，可在DOS提示符下敲入下列内容：

```
C>SET LIB=C:\MSOFT\LIB
```

设置路径（PATH）

如果要在 PROG 子目录下创建程序，我们希望只键入文件名就可在这个目录下取得PWB.COM。或QC.EXE 文件，即使它们是在 BIN（或BINB）目录下。为了做到这一点，我们使用了 DOS 命令——PATH。该命令可使 DOS 在指定目录下搜索当前目录下（PROG）找不到的文件。在PDS中，PWB.COM在BIN目录下，而连接器在 BINB 目录下，因此我们可通过在环境中设置下列语句的方式引导DOS去那儿寻找文件：

```
PATH=C:\MSOFT\BIN; C:\MSOFT\BINB
```

这条PATH语句与符号名的设置方式相同，它使用SET命令：

```
SET PATH=C:\MSOFT\BIN; C:\MSOFT\BINB
```

实际上，在PATH中PDS请求另一个目录。有些操作需要在HELP目录下执行程序，因此路径名C:\MSOFT\HELP也可成为PATH命令的一部分。

在Quick C中，由于连接器在BIN目录下，则上述的PATH语句的第二部分不需要。

设置AUTOEXEC.BAT文件

在每次启动系统时，把那些PATH和SET命令都敲入DOS是十分不方便的。幸运的是，我们可以通过把所有的命令都放入一个批处理文件的方式进行自动处理。若该批处理文件是AUTOEXEC.BAT，那么当我们引导计算机时，它就可以运行。我们甚至不用过多地去想它，因此对于上述情况，AUTOEXEC.BAT文件应包括下列行：

```
SET PATH=C:\MSOFT\BIN; C:\MSOFT\BINB; C:\MSOFT\HELP
SET INCLUDE=C:\INCLUDE
SET LIB=C:\LIB
CD \MSOFT\PROG
```

最后一行把我们立即引入要产生源文件的PROG子目录下。

上面所讲的编程环境在图1-2中进行了简要的描述：

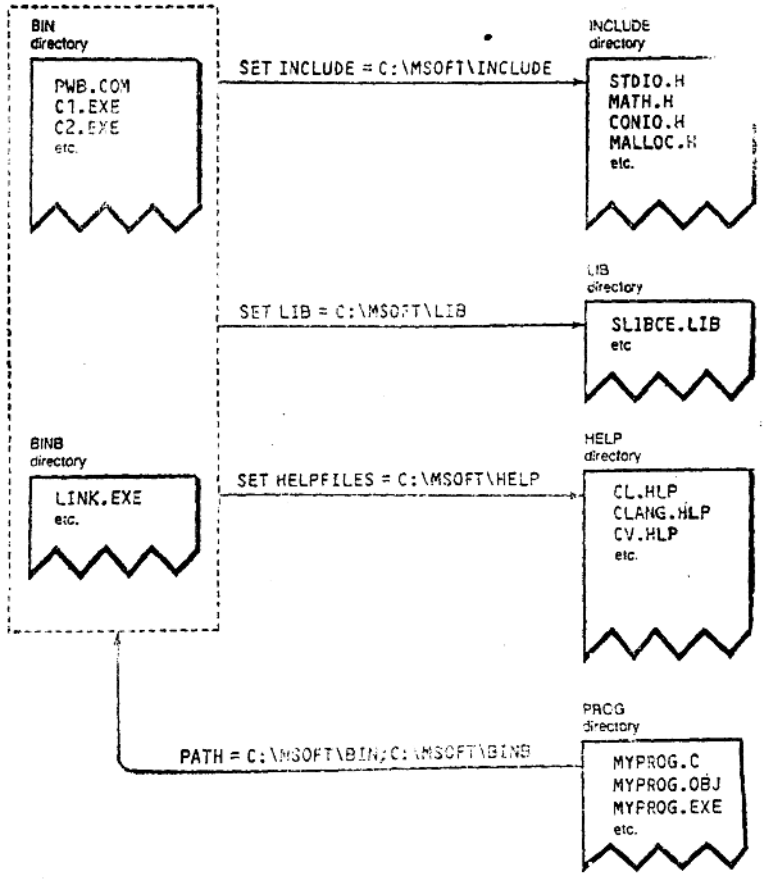

图1.2

PDS和Quick C都能提供一个带有所需要的DOS命令的文件来帮助你创建一个相应的AUTOEXEC.BAT文件，该文件叫作 NEW-VARS.BAT，它由设置程序（the setup program）放置于BIN目录下。如果你还没有建AUTOEXEC.BAT文件，那么你可以给这个文件重新起名，并将它拷贝到根目录下，或者使用你的程序编辑器把 NEW-VARS.BAT 中的行合并进一个已存在的AUTOEXEC.BAT文件中。

设置CONFIG.SYS和ANSI.SYS文件

当一个MS-DOS或PC DOS系统被引导时，操作系统首先寻找一个叫作CONFIG.SYS的文件。该文件通知DOS关于系统配置的各种情况，其中包括一次可打开多少文件，可得到多少数据缓冲区。PDS和Quick C均要求DOS能够打开20个文件，至少有10个缓冲区（缓冲区是存贮器内的临时存贮区）用来保持输入／输出数据。这样，在根目录下应有一个包括下列行的CONFIG.SYS文件：

```
files=20
buffes=10
```