PASCAL

语　言

IBMPC 丛书

11

PC 丛书编辑部

目　　录

第一章 IBM 个人计算机 Pascal 导引

本书描述IBM个人计算机有关Pascal的程序设计，在以后的几章里，将逐步引进两个Pascal的功能和灵活性。IBM Pascal是对ISO（国际标准化组织）标准Pascal的扩充，它着重在机器一级程序设计所"需要的系统"特点，你可以认为这些扩充是一种高级汇编语言，UCSD Pascal也是对ISO标准Pascal的扩充，它侧重在应用程序的需要。你可以用UCSD Pascal编制扩充的算术计算，作图，和屏幕编辑程序。

可惜，对这两种Pascal的这种简单地区分，总的来说过于简单化。这两种语言都可以用于系统程序设计，并且加上一些汇编语言程序后，它们都可以用于应用程序。为了真正地理解这些不同，你必须读其他章节。为了较快地得到全貌，我们从宏观上视察一下这两个语言。

IBM PASCAL

IBM Pascal是由Microsoft有限公司研制的，它是一个高效的优化代码编译程序，把相当大的精力集中在使编出的机器语言运行速度快，这种提高效率的一个后果是把源程序翻译成机器语言需要三次扫描源程序。IBM Pascal程序被翻译成在IBM DOS操作系统控制下执行的机器指令。结果是机器语言程序，它可以同其他机器语言程序结合在一起。源程序一旦被翻译之后，IBM Pascal程序同其他任何机器语言程序没有区别。

不仅IBM Pascal编译程序对源程序进行三次扫描，而且它还需要128K的内存。如果你还没有购买具有这么多容量的机器的话，你应该记住这个内存需要量。

IBM Pascal目标程序（源程序的机器语言版本）通常需要大约25K的运行支持程序。

运行支持程序是一个子程序，当你的程序运行时它驻留在内存中，它协助你的程序进行输入和输出（I/O），算术运算及其他工作。

UCSD Pascal

UCSD是加里福尼亚大学圣地亚哥分校在Kenneth Bowles博士指导下研制的。后来，与SofTech微机系统签订合同作为商品出售，原来的版本逐渐发展到IV版本，它可运行在IBM个人计算机及其他许多微型机上。

UCSD Pascal语言的一个较强的功能是：它是被称为P_系统的可移植操作系统的一部分。这个P_系统包含一个管理磁盘文件的文件服务程序，一个构造程序的编辑程序以及把汇编语言程序，FORTRAN程序和UCSD Pascal程序翻译成为可执行指令的各种翻译程序。

然而，获得可移植性是有代价的，因为在P_系统中程序被翻译成P_代码而不是机器语言。P_代码程序是一个包含抽象机器语言指令（称作P_代码指令）的程序。这些P_代码指令远比实际的机器语言指令紧凑，但是它们的执行速度很慢。整个P_系统将执行在64K的机器上，这样，它只需要64K内存。P_系统解释P_代码程序很像BASIC解释程序

直接执行BASIC指令一样。P—代码解释程序有时称作P—代码模拟程序，模拟P—代码机器。P—代码模拟程序是机器语言程序，P—系统运行UCSD Pascal程序比运行翻译成的机器指令要慢的多。这样，可移植性的代价就是降低了执行速度。

UCSD P—系统的IV版本包含一个实际代码生成程序，它可以把绝大部分P—代码指令翻译成等价的机器语言指令。由于这种翻译不是百分之百的完成，你必须仍然在P—系统下运行你的程序。然而，这种实际代码翻译的优点是提高运行速度，又把失去的大部分(不是全部）执行速度收回来了，但是结果程序相当庞大。这样，你必须作出决定对于你来说什么是重要的：是程序的大小还是程序的运行速度。(你可以有选择地把子程序翻译成实际代码）。

UCSD Pascal编译程序用一次扫描进行翻译。这意味着你可以期望相当快的翻译，P—代码程序需要很少的运行支持，因为运行支持程序建在P—代码模拟程序之中。

可移植的P—代码程序可以运行在任何具有P—系统软件的微机上。对于软件开发者，这意味着程序一旦为IBM个人计算机写出，它也将运行在其他一些微机上。

系统需要量的比较

表1.1 总结了UCSD和IBM Pascal系统需要量之间的差别，这是一个很概括的比较，但是它的确告诉你在这两种情况中需要哪种硬件。

表1.1 两个Pascal的系统需要量

	IBM	UCSD
编译需要的软盘	3	1
手　册	1	2(仅语言)
所需内存	128K	64K
提供的磁盘驱动	2	2
支持的打印机台数	最多2台	最多1台
显　示	黑白或彩色	黑白或彩色
操作系统	DOS	P—系统

IBM Pascal需要三片软盘，每遍扫描一片。由于UCSD版本是单遍扫描编译程序，所以它只需要一片软盘。两个版本都可以运行在一台软盘系统上（特别是双面驱动），但是不提倡这样作。

注意主存需要量的不同。UCSD P—系统是很紧凑的，但是你的Pascal程序可能很大。所以，在两种情况中不论哪一种情况你都可能需要128K主存。在从两个Pascal版本中选择一种版本时不用考虑这种内存的不同。你将很快会发现除最普通的程序外对所有的程序，128K是需要的。

两个Pascal都支持黑白显示和彩色转换器。然而，UCSD Pascal包含海龟图（turtlegraphics），一个用来作彩色和黑白图形的程序库。如果你打算作图，最好选择UCSD-Pascal语言。

此为试读，需要完整PDF请访问：www.ertongbook.com

对于DOS写的程序可容易地转换成其他在DOS控制下的IBM个人计算机程序。然而，对于P—系统写的程序可以在任何其他P—系统控制下的计算机上运行，（允许你随同你的程序一起出售P—系统）

如果你打算作系统级的程序设计并且你要机器语言目标程序作为你的最后结果，那么你最好选择IBM Pascal系统。你必须记住你正在用可移植性换取实现的消耗。

语言特点比较

标准（ISO）Pascal同Niklaus Wirth发明的原始语言是很相似的。设计这个原始语言的目的是为了帮助学生学习适当的程序设计技巧并且在各种计算机上容易实现。在向着现代结构程序设计语言跨进的过程中这些优越性很久以来已被人们忘记。多数Pascal版本超出了教学用的标准。他们企图作职业程序员要作的一切。IBM Pascal和UCSD Pascal也不例外。表1.2总结了这两个Pascal的全部特点。

这两个语言都把符号串作为内部建立的数据类型。IBM Pascal实际支持两类符号串：L__串（长度串）和简单串。L__串可以改变长度，而简单串则不能。UCSD串和IBM的L__串是类似的。

两个语言都支持处理串的固有函数。所谓固有函数是内部建立的函数。然而，IBM Pascal是有点混淆，因为它有L__串和简单串两种数据类型。在后面几章中很详细地叙述了这些情况。

在UCSD Pascal中，通过一个预先翻译的程序库支持作图和声音再生，如果你计划编写音乐，游戏或图形设计方面的应用程序，那么你应该使用UCSD Pascal。

表1.2 两个Pascal的特点

	IBM	UCSD
串	有	有
作　图	没有	有
声　音	没有	有
程序单位(分别编译)	有	有
模块(分别编译)	有	无
并行处理	无	有
程序链	无	有
直接文件	有	有
系统程序设计	有	无
压缩数据	无	有
初　值	有	无
结构常量	有	无
控制break	有	有
控制cycle	有	无
情况otherwise	有	无
过程作为实参	有	无

IBM Pascal不直接支持屏幕游标控制。但是在本书中包含一个实现这操作的机器语

言程序。另一方面，UCSD语言为了游标控制而包含固有函数GOTOXY。

两个语言都支持分别编译的子程序。程序单位是一族过程、函数、和数据，它们可以分别进行编译，存贮在磁盘上而后把它们联接到其他程序上。当你编写大系统时，程序单位可能是有用的。IBM Pascal模块也可以被分别编译。它们类似于程序单位。

UCSD Pascal为你提供了一种把大块程序通过动态复盖块进入内存的方法。这意味着你可以运行不太大的程序以适合一块内存。如果你打算编写很大的程序，这个特点可能是所要考虑的重要一点。如果确实如此，那么UCSD Pascal语言可能是你的唯一选择。

UCSD Pascal还提供了并行进程。并行进程是一个同其他程序一起执行的程序。两个程序都在短脉冲中执行——首先一个执行，然后另一个执行。例如，你可能要写一个程序，它将在另外一个程序作文字处理时（在其他文件上）打印某文件的内容。这打印程序称作假脱机程序（Spooer），并且在文字处理程序执行的时间之间按短脉冲执行。假脱机程序和文字加工程序是（两个）并行进程的系统部分。由于它为你提供了一种书写并行程序的途径，UCSD Pascal对于系统程序设计可能要比IBM Pascal好些。事实上，如果你要学习更多的计算机系统并行性，使用UCSD Pascal。

两个语言对于文件处理都超出了ISO标准。直接存取文件包含这样的数据，即可以在对盘的一次“搜索（Seek）”中直接存取。这个特点在许多数据处理应用中是重要的。因为它的重要性，本书以整章的篇幅讨论这个问题。

IBM Pascal在对系统程序设计方面扩充了ISO Pascal。特别是，IBM Pascal含有位级数据类型，byte和word，它允许你在内存中存取二进制编码单元，其他数据类型允许你控制8088处理机的内存段寄存器，等等。这些特点破坏了Pascal的强类型特点，使得你可以在内存中作混合类型操作。

UCSD Pascal允许你把数据压缩到尽可能小的内存空间，这种压缩属性使得数据按最有效的方式压挤到内存中。另一方面，IBM Pascal对数据不进行压缩。记住IBM Pascal的目的是要求快速和高效。如果你想压缩数据，那么你必须使用系统级数据类型自己进行压缩。

UCSD Pascal提供了一个出口过程，它允许你在过程或程序到达终点前跳出。这是一个“提前终止”固有过程，它在结构程序设计中是有用的。

如果你习惯于使用goto语句，那么你将对IBM Pascal中的各种出口感兴趣。现把这些出口汇总如下：

break	在完成整个一遍之前离开循环或离开判定路线。
Cycle	跳到循环的终点，然后在这循环的开始处开始下一个循环
情况otherwise	如果在Case语句中所有的选择失败，那么作这语句的otherwise子句。

语言限制

表1.3 汇总了一些IBM Pascal和UCSD Pascal很重要的限制。

UCSD Pascal有长整数的限定形式，它允许你处理整数直到36个数字。另一方面，IBM Pascal提供了称作word的长整数，其范围是0到65,535。

表1.3 两个Pascal的限制

	IBM	UCSD
最大整数	—32,767..32,767	—32,797..32,767
字	0..65,535	—
长字	—	直到36个数字
最大实数	7位数字E±38	16位数字E±308
串长	0..255和1..32,767字符	1..255字
数组大小	1..32,766字节	1..32,766字节
集合大小	—	1..4080个元素
标识符	识别到31个字符	识别到8个字符

UCSD Pascal在两个可供选择的模拟器之上运行，SYSTEM4模拟器支持16位实数。SYSTEM2模拟器和IBM Pascal语言支持7位数字实数。

IBM Pascal L—串可以达到255个字符长，定长串可以容纳到32,767个字符。记录，集合，和枚举标量的最大范围在IBM Pascal语言中没有定义。

无论哪一种语言你可以使用几乎是任意长度的标识符（只要在一行内可以写下）。UCSD只查看头8个字符，而IBM Pascal语言要求头31个字符必须是唯一的。在两个语言中都允许下线字符，而在UCSD Pascal系统中把它忽略掉。例如，REC—NUM被UCSD Pascal处理成RECNUM。

当你深入学习IBM个人计算机Pascal程序设计时你将会迁到每种语言所独有的其他限制。一般来说，如果你要求执行速度或系统级程序设计，并且可移植性不太重要，那么，IBM Pascal语言可能是比较好的选择。相反，如果你要求可移植性，容易使用，作图，或声音，并且你不太多的考虑执行速度或达到内存的位级，那么UCSD Pascal对你来说可能是较好的选择。

本书将为两种程序设计语言提供一些例子，语言中差异很大的部分将在同一章的不同节中讨论，这样便于你选择你所要的语言。

自学问题

1. P—代码模拟程序是什么？
2. 什么是ISO？
3. 什么是P—系统？
4. 何谓编译程序的遍？
5. 什么是可移植性的代价？
6. 什么是实际代码生成程序？
7. 为什么推荐使用两个磁盘驱动？
8. 什么是L—串？
9. 什么是海龟图（turtlegraphics）？
10. 分别编译意味着什么？

自学问题解答

1. 直接解释P—代码指令的机器语言程序。
2. 国际标准化组织——为我们建立标准的委员会。
3. 一个庞大程序，它包含文件服务程序，屏幕编辑程序，几种语言的编译程序，和P—代码模拟程序。
4. 翻译程序对程序扫描一遍，每次它读整个程序。
5. 可移植程序比特定机器程序运行速度慢
6. 它是一个程序，它把P—代码语句转换成某个机器的机器语言语句。
7. 你可以用一个驱动进行，但用单个驱动太慢了，并且UCSD系统不允许你编译很大的程序。
8. 长度串—可以改变长度的串。
9. 用来在屏幕上作图的程序库，一个虚构的海龟来回移动，在它后面拖着一把彩刷。
10. 它意味着这个程序在其他时间进行编译，分别进行编译的程序单位就是可以对它进行编译并且保存在磁盘文件中的程序，之后，当较大的程序需要这个程序单位时，不需要对它重新编译就可与大程序联在一起。

第二章　如何使用Pascal编译程序

使用 IBM Pascal编译程序

IBM Pascal编译程序是一组驻留在三个软盘上的程序，这三个软盘的名字和功能如下：

PAS1　第一遍扫描 Pascal 程序，其目的是把诸语句转换成低效的机器语言。

PAS2　第二遍是扫描从 PAS1 得到的机器语言，其目的是对低效的指令进行优化（使它们短些、快些）

PAS3　第三遍是扫描从PAS2得到的优化代码，其目的是把运行支持程序并入你的程序中，输出程序是可以执行的程序。

新磁盘的格式化

在第一次使用这三个软盘之前，你应该为三个原始 Pascal 软盘建立三个工作拷贝。

- 第 1 步，使用下面的命令对三个新盘格式化（把DOS盘放在左边驱动A，把新盘放在右边的驱动B），使用／S命令把系统各磁道复制到新盘上。

```
A>FORMAT B: /S
```

用Y回答计算机提出的问题“Do you Want to format another (Y/N) ?”以便格式化全部三个新盘。

- 第2步，放PAS1盘到驱动A并且把它全部复制到驱动B上的格式化了的空盘上。

```
A>COPY A: *.* B:
```

这种复制要花费一些时间，完成之后在荧光屏上将显示复制的文件个数，现在你在另一个盘上有了 PAS1 的复制品。

- 第 3 步，放 DOS 系统盘到驱动A（把新拷贝留在驱动B），把行编辑程序复制到B。

```
A> COPY EDLIN.COM B:
```

现在，从驱动B把盘拿出并给它标上PAS1-Copy将来你需要这个盘翻译你的程序。

- 第 4 步，把原始的PAS2盘放到驱动A，并且把另一个格式化了的空盘放在驱动 B，进行复制。

```
A>Copy A: *.* B:
```

拿出B中的盘并标上 PAS2—Copy，把它放好以留后用。

- 第 5 步，把原始PASCAL LIB盘放在驱动A，另一个（第三个）格式化的空盘放在驱动B并作复制。

A>Copy A:*.*B:

留新复制盘在驱动B上。

·第6步，把DOS 放回驱动A，从它复制 LINK.COM 程序。

A> Copy LINK.EXE B:

现在从驱动B中取出盘并标上 PASCAL.LIB-Copy。这是为编译你的程序所需要的第3个且是最后一个磁盘。

你可以把三个原始盘放在安全的地方而使用三个复制盘。现在，在PAS1-Copy 盘上有行编辑程序（EDLIN）和 PAS1 程序。PAS2-Copy 盘用来加工PAS1的输出文件，PASCAL.LIB-Copy盘有LINK程序以及你的程序所需要的预翻译的库程序。使用LINK程序把库程序和你的程序合并在一起。你将总是需要作这个工作，因为有很多输入输出，算术运算等等，Pascal 翻译程序要调用这些程序帮助它把你的程序翻译成机器语言。这些被称作运行支持，或执行支持程序，因为在你的程序运行过程中需要它们。

所有这些复制和准备作完之后，你可能以为你已经可以翻译 Pascal 程序了。遗憾的是，你还需要作一步。下两节将告诉你如何建立批量文件，它将很容易地完成其他操作。

IBM 翻译程序如何工作

三个Pascal翻译程序盘工作如下：开始使用 PAS1-Copy 盘并把它放在左边的驱动A上，把含有你的Pascal程序的盘放在右边的驱动B上。为了方便起见，让我们称你的程序为 YOURPROG.PAS。程序不能用 Pascal保留字（参看附录A和B）取名；换句话说，你可以给你的程序取你喜欢的任何名字，只要它有.PAS作为它的扩充即可。

PAS1-Copy取YOURPROG.PAS 作为它的输入并且把它翻译成PASIBF.SYM(它包含YOURPROG.PAS所用的变量表）和PASIBF.BIN(它包含与YOURPROG.PAS等价的机器语言)。你可以有选择地请求其他文件，如 YOURPROG.LST，它含有你的 Pascal 源程序的列表，或者YOURPROG.OBJ，它含有和你的程序等价的目标代码。这些文件是任选的。所以这里我们将不涉及它们。参看图2.1的信息流程图。

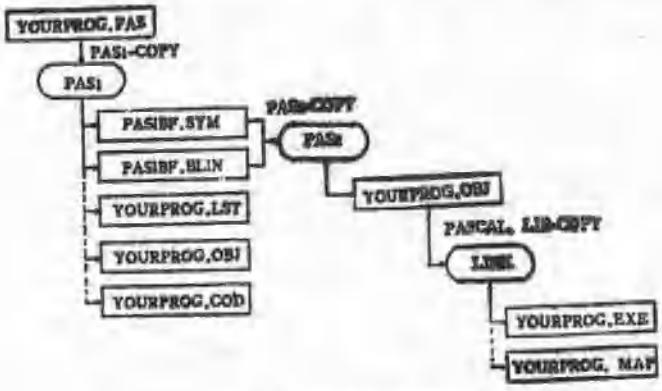

图 2.1 如何翻译你的程序

此为试读，需要完整PDF请访问：www.ertongbook.com

PAS2_Copy（在第2个盘上)取PAS1_Copy产生的两个文件作为它的输入并且优化在PAS1BF.BIN中的机器语言。除一件事外它就可以运行了；即它还没有同库程序合在一起，这依赖于进行输入输出，算术运算等等。

YOURPROG.OBJ是LINK程序的输入(同PASCAL.LIB程序一起)。LINK程序产生一个可执行的程序，称作YOURPROG.EXE，只要打入它的名字就可执行，正像命令：

```
A>YOURPROG
```

把计算机的控制传递给YOURPROG。

显然，这个过程是冗长而又厌烦的，为了回避它，你将需要下一节要描述的批量文件。为了方便你应该把这批文件放在PAS1—COPY盘上。

用于Pascal编译的方便批量文件

在键盘上用 Copy 命令打进批量文件。当进入最后一行时，在输入的最后一行的第一列打CTRL+Z便终止输入。参看图2.2。

这文件应称作 PASCAL 或者某个等价的名字像 COMPILE 或者 TRANS，并且作为.BAT文件(例如PASCAL.BAT)存贮，它假定80列屏幕模式，使用这批量文件如下：

```
A>PASCAL  YOURPROG
```

这导致按三步翻译YOURPROG.PAS.不产生列表。

```
A>PASCAL  YOURPROG  LPT1:
A>PASCAL  YOURPROG  CON:
```

这导致按三步翻译 YOURPROG.PAS 并且把编译程序列表或者送到打印机（LPT1：）或者送到萤光屏（CON：）。包括错误信息。

图 2.2用于自动翻译Pascal程序的PASCAL.BAT

```
a: mode 80
Rem Compile, Link, and run a Pascal Program
Rem PASCAL SOURCE LISTING is command line
Pause. Put PAS2 in drive A, and your Program diskette in B.
B:
A: PAS1 %1,,%2;
Pause. Put PAS2 in drive A, and leave your Program diskette in B.
A: PAS2
rename %1.obj result.obj
Pause.Put PASCAL.LIB in A, and leave your Program diskette in B.
a: link b: linkparm
erase result.obj
erase %1.exe
rename result.exe %1.exe
Pause. Running your program, now.
%1
```

PASCAL．BAT文件使用另外一个文件叫LINKPARM，应把它放在PASCAL．LIB—Copy中，使用Dos的Copy命令来作这一工作。LINKPARM文件应包含三个字，随后跟着7个空行，正像这里所给出的。

```
result
result
nul
(空行)
(空行)
(空行)
(空行)
(空行)
(空行)
(空行)
```

只要按ENTER键不打字符就可键入每一空行。这些空行和头三个字是对 LINK 程序所提问题的回答。这样，LINKPARM 自动回答重复的问题从而节省了你的时间。

PASCAL.BAT 文件将抹去前面已编译的相同名字的程序。第一次使用它，不存在以前已编译的程序，这样，将显示错误信息 BAD FILE NAME 简单地忽略它。

正像图2.2所给的那样，键入批量文件信息，并且开始直接地使用它，当你获得更多的 Pascal 系统的经验和知识之后，你可以修改它或者全部拿掉。然而，在多数情况下，它将为你在本书中遇到的问题服务。

为冒风险

现在你已为使用 IBM Pascal 编译程序作了准备。然而，如果你对这章的第一部分还没有讨论的一些文件难以理解，那么在继续学习之前你可以对它们研究一下。

使用 DOS 的 TYPE 或COPY命令探察一下 PAS1 _Copy盘。称作PASKEY，FILKQQ.INC,FILUQQ.INC,和ENTX6S的文件装着有趣的信息。例如,如果你使用命令

```
A>TYPE  PASKEY
```

显示 PASKEY，并且使用组合键 CTRL+NUMLOCK 来冻结屏幕（停止卷动），那么，你将看到预先说明（保留）的键字和过程表，这些过程和函数由 Pascal 编译程序自己使用，而你也可以使用它们。

FILKQQ 和 FILUQQ 文件装有用来作机器级操作的“include 文件”信息。例如，PTYUQQ程序让你直接地从 Pascal 程序输出字符到萤光屏。这些预先说明的程序是高级的性质，并且多数程序员是不感兴趣的。然而，它们确实为爱冒风险的读者提供了Pascal 系统附加信息。

使用UCSD PASCAL编译程序

UCSD Pascal 编译程序放在 UCSD P_系统中。为了全面地了解P-系统，你应

该需要详细地学习操作系统，文件管理，编辑程序等等，这里代之以学习一个如何使你的程序进入系统，进行编译然后执行的简单方法。

P—系统的简短巡视

UCSD P—系统存贮在六个盘上，但是我们将涉及磁盘格式化，进入程序，编译和运行三个盘。

找出标有SYSTEM4的盘并且把它（标示面向上）放进IBM个人计算机左边的驱动中。复位或打开机器，等待30—40秒以使机器开始执行 SYSTEM4 上的程序。当P—系统已准备好，你将会看到命令指南提示行和如图2.3所示的开始标识。

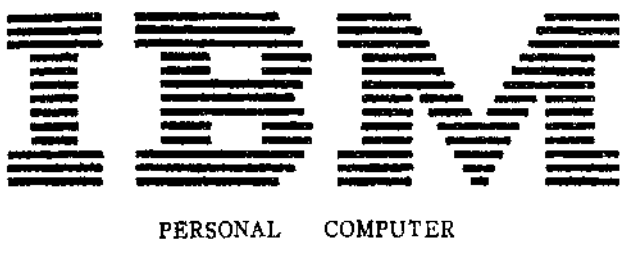

图 2.3 P—系统的命令行和开始标识

命令指南是位于萤光屏顶部的一行提示。注意，如何用单个字母显示每个命令，这个字母的后面跟着一个括弧，它把单个字母与这命令的剩余部分分开。如果你按 E（编 辑），R（运行）等等，将激活相应的子系统。为了更加熟悉这个系统，按F（文 件管理）并且你将在萤光屏顶上得到这文件子系统的提示行（参看图2.4）。这个文件提示行 也 使 用 单 个 字母，后面跟着左括号，再后是这个命令的部分拼写。

为了列出目录，接L键并且当问你盘名时，按*键表示系统驱动，这将引起显示盘 目 录如图2.4所示。列出的目录给出了盘的名字（SYSTEM4: ）和这个盘中所有文件的名字，包括它们的大小（以512个字符一块）以及文件建立的日子。它还告诉你已经用了多少块磁盘空间，多少块没有使用以及最大可用块的大小。

文件名字是由文件标识符，后跟一个句号，再跟一个文件扩充符或类型所组成，系统文件都以标识符 SYSTEM 开始。这里给出图2.4中所列出文件的简明描述。

图 2.4 文件服务系统的响应行和SYSTEM盘的内容

```
Filer:    G(er, S(ave, W(hat, N(ew, L(dir R(em, C(hng, T(rans, D(ate,?    (C.11)
SYSTEM 4:
SYSTEM.PASCAL   123    9-Jan-82
SYSTEM.MISCINFO   1    9-Jan-82
SYSTEM.INTERP    28    9-Mar-82
SYSTEM.FILER     32    19-Jul-81
```

```
SYSTEM. EDITOR      49    19-Jul-81
SYSTEM. STARTUP      2    29-Oct-81
SYSTEM. LOGO         1    29-Oct-81
SYSTEM. SYNTAX      14    22-May-81
SYSTEM. LIBRARY     34     9-Jan-82
SPOOLER.CODE         2    27-Jul-81
MYPROG. TEXT         4     9-Jan-82
MYPROG. CODE         2     9-Jan-82
12/12/files<listed/in-dir>,805  blocks  used,  15  unused,15  in  largest
```

SYSTEM.PASCAL　这是运行整个系统的P—系统操作系统。由这个程序“调用”一切别的程序。

SYSTEM.INTERP　这是解释P—代码的P—系统模拟程序。由于所有的程序都生成P—代码，所以需要这个解释程序执行任何其他程序。事实上，SYSTEM.PASCAL中的大部分是P—代码，所以这系统本身在模拟程序上运行。

SYSTEM.MISCINFO　这是包含IBM个人计算机信息的数据文件。这是“特性”模块它为P—系统描述机器，使可移植的P—系统对它正在其上运行的机器是特殊的。

SYSTEM.FILER　这是当你按F键时所执行的程序。它管理磁盘文件。

SYSTEM.EDITOR　这是屏幕编辑。用它写进 Pascal 程序，当处在操作命令清单模式时按键E来激活它。

SYSEM.STARTUP　这是P—系统启动之后，你要立即执行的程序。它是显示图2.3标识的程序。

SYSTEM.LOGO　它是图2.3显示的大的IBM标识。

SYSTEM.SYNTAX　它是包含Pascal编译程序错误信息的数据文件。无论何时只要程序不能正确编译时，就显示一个或多个这样的错误信息。

SYSTEM.LIBRARY　它是为作图、声音等预先翻译的程序库。

SPOOLER.CODE　这是当计算机正作其他什么事情时，运行行式打印机的并行程序。当编辑其他文件时，你可以使用这个程序把文件从打印机卸出。

MYPROG.TGXT　这是一个 Pascal 源程序的例子。用E（编辑）命令产生这类文件。在下一节，你将学习如何生成源程序。

MYPROG.CODE　这是一个Pascal目标（P-代码）程序例子。它是通过编译MYPROG.TEXT得到的输出。你可以执行.CODE文件。

首次上机

首先，你要复制全部系统盘并且要格式化空盘用来写你的程序。这两种情况你都需要格式化新盘。

把 SYSTEM4 盘放在左驱动器，UTILITY 盘放在右驱动上。复位或接通电源，当

你看到图2.3的系统标识时，用下面命令执行格式化程序：

```
X
Execute What file? #5:FORMAT
```

驱动#5：是右驱动（#4：是左驱动）。格式化程序叫作FORMAT。#5：FORMAT命令致使在盘#5：上的程序执行。5是被格式化的盘驱动号。取出UTILITY盘并且插入一新（空）盘，按ENTER并且等待着新盘被格式化。对于每一个你要格式化的盘，重复这个格式化步骤。

当你作完之后，退出格式化程序；你将回到命令级。按F（文件管理）并显示文件服务提示行。按T（传输），复制所有的系统盘。

把被复制的盘放在驱动#4：，放要格式化的空盘在驱动#5：，等待每个盘被复制。按Z（除去）后面跟着#5:给你的程序盘一个盘号。这个除去（zap）程序将问你问题，当问你新名字时，譬如回答PROG：，要保证包括冒号。

现在你已有了所有系统盘的复制品和称作PROG:*的格式化的空盘，你的下一步工作是写程序并且把它存贮在PROG：的文件中，然后你可编译并运行它。

运行你的程序

如果你有一台带有单面驱动的计算机，你不能得到系统盘上的编译程序。对这种情况你必须把SYSTEM.COMPILER复制到你的程序盘上（PROG：）。使用4字节文本编译程序（它在其他某个盘上），你可使用文件服务的T命令来作这种复制。

现在，在驱动#4：有系统盘，你的PROG：盘在驱动#5：，你已完成了准备工作。你必须编辑（E命令）新程序，然后对它进行编译（C命令），最后执行（X命令）这程序。

```
>Edit:
No Workfile is Present.  File? (<ent> for no file)
```

图 2.5 进入编辑程序时的响应行

为了编辑按E键，你将看到图2.5所示的编辑程序响应行。按ENTER键开始一个新程序；这时将出现图2.6所示的编辑程序响应行。单字母命令工作和你已看到的单字母命令是一样的。

为了插入按I键，将显示图2.7所示的插入响应行，写入图2.7所示的简单程序。注意，编辑程序如何为你自动地缩进，要想退回缩进的空格，你必须使用backspace键。当你打

```
>Edit: A(djst C(py D(iet F(ind I(nsrt J(mp K(ol R(plc Q(uit X(ch Z(ap (E.7n)
```

图 2.6 编辑程序为你开始写进正文做好准备

完这简单程序时，按CTRL+C从插入模式退出来。CTRL+C组合键接受你的插入，而ESC键使它丢失！

使用箭头键在你已键入的正文中移动游标。为了更熟悉编辑程序，你可用D（删除）

和I（插入）命令模式作试验。你可以用F和R命令检索和替换符号。试试看！

```
>Insert: Text {<bs> a char, <del> a line} {<etx> accepts, <esp> escapes}
Program SIMPLE;
const
YES = 'Y';
NO = 'N';
var
  ANSWER : char;
begin
  writeln('Enter Y or N:');
  readln(ANSWER);
end.
```

图 2.7 用插入命令I键入程序

当你准备保存你的程序并返回到命令清单级时，按Q便可退出。将出现图2.8所示的响应。按W键将你的文件写到盘上并且你将看到图2.9所示的响应。如果你要把你的程序保存在叫PROG: 的盘上，使用这个名字作为输出文件名字的一部分，例如，在图2.9中，用下列名字代替MYPROG:

PROG: MYPROG

这引起程序存贮在称为PROG: 的盘上。图2.10为你显示了当保存MYPROG时发生的情况。然后你可按E退出编辑程序。

```
>Quit:
U(pdate the workfile and leave
E(xit without updating
R(eturn to the editor without updating
W(rite to a file name and return
```

图 2.8 如何退出编辑程序

```
>Quit:
Name of output file (<ent> to return)—→MYPROG
```

图 2.9 写文件MYPROG举例

```
>Quit:
writing……
your file is 147 bytes long.
Do you want to E(xit from or R(eturn to the editor?
```

图 2.10 保存MYPROG之后从编辑程序中退出

*为了给盘取名为PROG: ，使用程序DISKFORMAT对空盘初始化，然后使用文件服务命令Z来命名格式化的盘为PROG: ，在UTILITY盘上可以找到DISKFORMAT程序。

此为试读，需要完整PDF请访问：www.ertongbook.com