第 4 章　计算机软件

4.1　软件的概念、特点和分类

4.1.1　软件的概念与特点

1．软件的概念

“软件（Software）”一名词是在 20 世纪 60 年代出现的。一般认为，软件是计算机系统中与硬件相互依存的另一个重要组成部分。从系统工程角度来看，它作为系统元素，与计算机硬件、人、数据库、过程等共同构成计算机系统。

软件由四部分组成：计算机程序、数据、文档和售后服务。其中，计算机程序是按事先设计的功能和性能要求执行的指令序列；数据是指支撑计算机程序执行的数据结构；文档是与程序开发、维护和使用有关的图文材料，它又可分为系统文档、用户文档和 Web 站点。系统文档用于描述系统的结构；用户文档针对软件产品解释如何使用系统；Web 站点用于下载系统信息；售后服务是指能够为购买计算机软件的用户提供有力的技术支持。

简单地说：**软件 = 程序+数据+文档+服务**。

更细致地探讨，“软件”可具有三层含义：一是个体含义，软件是指计算机系统中的某个程序、数据、文档以及修改服务；二是整体含义，软件是指在特定计算机系统中所有个体含义的软件的总体；三为学科含义，软件是指在开发、使用和维护前述含义下的软件所涉及的理论、原则、方法、技术所构成的学科，在这种含义下，软件也可称为软件学。

软件是用户与硬件之间的接口。要使用计算机，就必须编制程序，必须有软件。用户主要通过软件与计算机进行交互。软件在计算机系统中起指挥、管理作用。计算机系统工作与否，做什么以及如何做，都“听命于”软件。

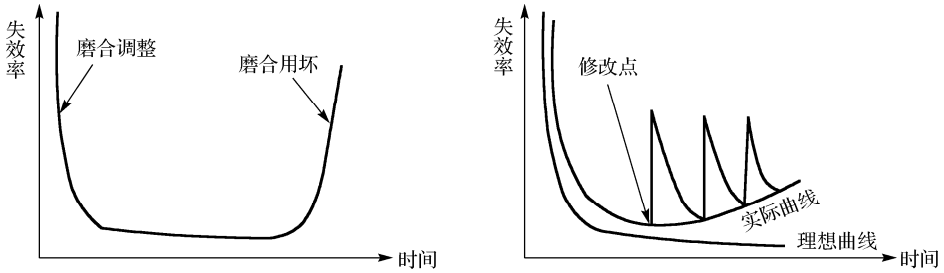
2．软件的特点

（1）软件是一种逻辑实体，而不是具体的物理实体。因而它具有抽象性。这个特点使它和计算机硬件，或是其他工程对象有着明显的差别。人们可以把它记录在介质上，但却无法看到软件的形态，而必须通过观察、分析、思考、判断，去了解它的功能、性能及其他特性。

（2）软件是开发出来的，不是制造出来的。软件没有明显的制造过程，因而软件的质量主要取决于软件的“开发”。在开发过程中，通过人们的智力活动和有效的管理，把知识与技术转化成信息产品。一旦某一软件产品开发成功，以后就可以大量地复制同一内容的副本。

（3）软件可能被废弃，但不会被用坏。在软件的运行和使用期间，没有硬件那样的机械

磨损、短路、用坏等问题。任何机械、电子设备在运行和使用中，其失效率大都遵循如图 4-1(a)所示的 U 型曲线（浴盆曲线）。而软件的情况与此不同，因为它存在失效和退化问题，所以必须要多次修改（维护）软件，如图 4-1(b)所示。当然，随着时间的不断推移，软件最终会由于不适应或需求的变化而被废弃。



(a) 硬件失效率曲线　　(b) 软件失效率曲线

图 4-1　失效率曲线

（4）以往的软件大多是定制的，而不是装配的。以前很少有类似于硬件“零部件”的软件“零部件”。即使有，也多为具有完整功能的软件产品或具有独立功能的模块，而不是“零件”或“部件”。因此，软件大多数是为用户专门“定制”的，而不是用现成的软件零部件“装配”而成的。现在，这种情况正在改变，随着面向对象技术和构件技术的迅速发展，开发出了越来越多的软件部件（或称为软件组件、软件构件），可以像硬件产品那样，实现一定程度的“即插即用”。

（5）软件的开发和运行常常受到计算机系统的限制，对计算机系统有着不同程度的依赖。软件不能完全摆脱硬件单独活动。在开发和运行中必须以硬件提供的条件为依据。而有的软件则依赖于某个操作系统。

（6）软件产品生产主要是脑力劳动，还未完全摆脱手工开发方式，大部分产品是“定做的”。

（7）软件是复杂的，人类能够创造的最复杂的产物是计算机软件。导致软件复杂性的根源是：① 软件反映了实际问题的复杂性；② 程序自身逻辑结构的复杂性；③ 软件开发常常涉及其他领域的专业知识。

（8）软件费用不断增加，软件成本相当昂贵。软件的研制工作需要投入大量的、复杂的、高强度的脑力劳动，它的成本非常高。

（9）相当多的软件工作涉及社会因素。

4.1.2　软件的分类

因为针对不同类型的工程对象，对其进行开发和维护有着不同的要求和处理方法，所以目前还找不到一个统一的严格分类标准。按照软件的作用，一般可以将软件做如下分类。

1．系统软件

系统软件能与计算机硬件紧密配合在一起，负责对整个计算机系统资源的管理、调度、监视和服务，使计算机系统各个部件、相关的软件和数据协调、高效地工作。

系统软件中的数据结构复杂，外部接口多样化，用户能够对它反复使用。它包括操作系统、程序语言处理系统、数据库系统、诊断和控制系统、网络系统、系统实用程序等。

① 操作系统　系统软件的核心，它负责对计算机系统内各种软、硬资源的管理、控制和监视。

② 程序语言处理系统　把用程序语言编写的程序变换成可在计算机上执行的程序，进而直接执行得到计算结果，完成这些功能的软件包括编译程序、解释程序和汇编程序。

③ 数据库管理系统　负责对计算机系统内全部文件、资料和数据的管理和共享。

④ 网络系统　负责对计算机系统的网络资源进行组织和管理，使得在多台独立的计算机间能进行相互的资源共享和通信。

⑤ 系统实用程序　为增强计算机系统的服务功能而提供的各种程序，如磁盘清理程序、备份程序等。

系统软件是计算机系统必不可少的一个组成部分。

2．支撑软件

支撑软件也称为工具软件，是协助用户开发软件的工具性软件，其中包括帮助程序人员开发软件产品的工具，也包括帮助管理人员控制开发的进程的工具。

支撑软件可分为纵向支撑软件和横向支撑软件：

纵向支撑软件是指支持软件生存期各个阶段特定软件工程活动所使用的软件工具，如需求分析工具、设计工具、编码工具、测试工具、维护工具等；

横向支撑软件是指支撑整个软件生存期各个活动所使用的软件工具，如项目管理工具、配置管理工具等。

20 世纪 90 年代中后期发展起来的软件开发环境以及后来开发的中间件可看成现代支撑软件的代表。软件开发环境主要包括环境数据库、各种接口软件和工具组，三者形成整体，协同支撑软件的开发和维护。

3．应用软件

应用软件是在系统软件支持下，在特定领域内开发，为特定目的服务的一类软件。

现在几乎所有的国民经济领域都使用了计算机，为这些计算机应用领域服务的应用软件种类繁多，其中商业数据处理软件是所占比例最大的一类，工程和科学计算机软件大多属于数值计算问题。

4．可复用软件

最初实现的典型可复用软件是各种标准函数库，通常是由计算机厂商提供的系统软件的一部分。

这些标准函数库中的标准函数可以不加改造，直接在新开发的程序中使用。后来可复用的范围扩展到了算法之外，数据结构也可以复用。到了 20 世纪 90 年代，作为复用的基础，可复用的范围从代码复用发展到了体系结构的复用、开发过程的复用。特别是，面向对象开发方法的核心思想就是基于复用的，为此，建立了可复用的类库、应用程序库等，其中可复用成分称为可复用构件。

在开发新的软件时，可以对已有的可复用构件稍加修改，或不加修改，通过继承复用所需的属性或服务。

4.2 软件的发展

软件的发展受到计算机应用与硬件发展的推动和制约，其发展过程大致可分为以下三个阶段。

1．程序设计阶段

从第一台计算机上的第一个程序的出现到实用的高级程序设计语言的出现以前为第一阶段（1946 年~1956 年）。

当时计算机应用领域较窄，主要是科学与工程领域的数值计算。编程所用的工具主要是机器语言或汇编语言。设计和编制程序采用个体方式，对和程序有关的文档的重要性认识不足，重视编程技巧。

特点：依赖于个人编程“技巧”，开发方法不规范，开发成本难控制，其交付的产品主要是程序。

2．程序系统阶段

从实用的高级程序设计语言出现以后到软件工程出现以前为第二阶段（1956 年~1968 年）。

随着计算机应用领域的逐步扩大，除了科学计算继续发展以外，出现了大量的数据处理问题，其性质和科学计算有明显的区别，涉及非数值数据。为了提高程序设计人员的工作效率，出现了实用的高级程序设计语言，为了充分利用计算机系统资源，又出现了操作系统。

为了适应大量数据处理问题的需要，也开始出现数据库及其管理系统。到了 20 世纪 50 年代后期，人们逐渐认识到了程序和相关文档的重要性，因而到了 20 世纪 60 年代初期，出现了软件一词融合程序及其文档为一体。

特点：软件开发是以小组的形式出现的，提交的软件产品除了程序外还有相应的使用说明书。

这个阶段，软件的复杂度迅速提高，研制周期变长，但正确性难以保证，可靠性问题也相当突出。到了 20 世纪 60 年代中期，软件的发展遇到了很多急需解决的问题，出现了人们难以控制的局面，即软件危机。

所谓的软件危机是计算机软件在它的开发和维护过程中所遇到的一系列严重问题。

① 如何开发软件，怎样满足对软件日益增长的需求；

② 如何维护数量不断膨胀的已有软件。

为了克服这一危机，进行了以下三个方面的工作：

① 提出了结构化程序设计方法；

② 提出用工程方法开发软件；

③ 从理论上探讨程序正确性和软件可靠性问题。

这一阶段的研究对象增加了并发程序，并着重研究高级程序设计语言、编译程序、操作系统以及各种支撑软件和应用软件。计算机系统的处理能力得到加强，设计和编制程序的工作方式逐步走向合作方式。

3．软件工程阶段

软件工程出现以后迄今为止为第三阶段（1968 年以来）。

由于大型软件的开发是一项工程性任务，采用个体或合作方式不仅效率低，产品可靠性差，而且很难完成。

由此，1968 年，德国人 Bauer 在北大西洋公约（北约）学术会议上提出了软件工程概念，开创了以工程的方法来管理软件的开发历程。

特点：开发方法遵循工程化的方法，按照软件生命周期分阶段开发软件，提交的软件产品除了程序之外，还有严格的文档。

从几十年来的软件工程实践表明：按工程化的原则和方法组织软件开发工作是必要的、有效的，是摆脱软件危机的一条主要出路。

表 4-1 列出了三个发展时期主要特征的对比。

表 4-1　计算机软件发展的三个时期及特点

时间 / 特点	程序设计	程序系统	软件工程
软件内容	程序	程序及说明书	程序、文档、数据
主要程序设计语言	汇编及机器语言	高级语言	软件语言*
软件工作范围	程序编写	包括设计和测试	软件生存周期
需求者	程序设计本人	少数用户	市场用户
开发软件的组织	个人	开发小组	开发小组及大中型软件开发机构
软件规模	小型	中小型	大、中、小型
决定质量的因素	个人编程技术	小组技术水平	技术水平和管理水平
开发技术和手段	子程序和程序库	结构化程序设计	数据库、开发工具、开发环境、工程化开发方法、标准和规范、网络和分布式开发、面向对象技术及软件复用
维护责任者	程序设计者	开发小组	专职维护人员
硬件特征	价格高，存储容量小，工作可靠性差	降价；速度、容量及工作可靠性有明显提高	向超高速、大容量、微型化及网络化方向发展
软件特征	完全不受重视	软件技术的发展不能满足需要，出现软件危机	开发技术有进步，但未获突破性进展，价高，未完全摆脱软件危机
* 这里软件语言包括需求定义语言、软件功能语言、软件设计语言、程序设计语言			

随着计算机技术的飞速发展，各种各样的应用软件需要在各种平台之间进行移植，或者一个平台需要支持多种应用软件和管理多种应用系统，软、硬件平台和应用系统之间需要可靠和高效的数据传递或转换，使系统的协同性得以保证。这些，都需要一种构筑于软、硬件平台之上，同时对更上层的应用软件提供支持的软件系统，而软件中间件就是在这个环境下应运而生的。

20 世纪 90 年代，中间件技术才开始迅速发展。进入 21 世纪后，或许是意识到软件通用性和用户需求个性化的矛盾过于突出， 软件平台的研究迅速发展起来。其中，业务基础平台作为一个新的软件层级尤为引人注目。业务基础平台是以业务为导向和驱动的、可快速构建应用系统的软件平台。2003 年前后，许多公司相继宣布推出自己的平台，掀起了第一轮业务基础平台热潮。2005 年，ERP 厂商再度引发“平台热”，金碟、SAP 都在此时高调推出平台战略。

现在软件领域工作的主要特点如下：

（1）应用领域的不断扩大，出现了嵌入式应用、网络软件及分布式应用和分布式软件。

（2）开发方式有个体合作方式转向工程方式，软件工程发展迅速，形成了“计算机辅助软件工程”。

（3）致力于研究软件过程本身规律，研究各种软件开发规范与模型。

（4）除了软件传统技术继续发展外，人们着重研究以智能化、自动化、集成化、并行化、开放化以及自然化为标志的软件开发新技术。

（5）注意研究软件理论，特别是探讨软件开发过程的本质。

4.3 从机器语言到高级语言

计算机语言（Computer Language）是指用于人与计算机之间通信的语言，是人与计算机之间传递信息的媒介。为了让计算机解决一个实际问题，必须事先用计算机语言编制好程序。计算机语言使人们得以和计算机之间进行交流，其种类非常多，根据程序设计语言与计算机硬件的联系程度，可以分为三类：机器语言、汇编语言和高级语言。

4.3.1 机器语言

电子计算机所使用的是由“0”和“1”组成的二进制数，二进制是计算机语言的基础。在计算机发展的早期，人们使用机器语言进行编程。计算机提供给用户的最原始的工具就是指令系统，采用二进制编码的指令编写程序，输入计算机运行就能得到预期的结果。

以计算机所能理解和执行的“0”、“1”组成的二进制编码表示的指令，称为机器指令，或称为机器码。用机器指令编写的程序称为机器语言程序，或称为目标程序，这是计算机能够直接执行的程序。机器指令的格式一般分为两个部分，如下所示：

指令格式：

操作码	操作数地址

其中，操作码指出 CPU 应执行何种操作，如加、减、乘、除等，而操作数地址指出该指令所操作（处理）的数据或者数据所在位置。

CPU 可执行的全部指令称为该 CPU 的指令系统，即它的机器语言。**应当注意，不同的机器，其指令系统是不同的，**大多数现代计算机都设计了比较庞大的指令系统，以满足用户的需求。

使用机器语言编制程序的前提是程序员必须熟悉机器的指令系统，并记住各个寄存器的功能，还要了解机器的许多细节。虽然机器语言对计算机来说是最直观的，又比较简单，不需要任何翻译就能立即执行，但由于它是用二进制形式表示的，很难阅读和理解，还容易出错，所以给编写程序带来了很大的困难，而且对于编程中出现的错误也很难迅速发现，修改就更困难了，这是机器语言的缺点。

例如，下面是一段计算 10+9+8+…+2+1 的程序（该程序用 8086/8088 的指令系统编写），表 4-2 中的第一列是存储器的地址；第二列是机器语言程序，如果不加说明，用户是很难读懂的；第三列是要说明的汇编指令。

指令系统中的指令分成许多类，如 Intel 公司的奔腾和酷睿处理器中共有 7 大类指令：数据传送类、算术运算类、逻辑运算类、移位操作类、位（位/串）操作类、控制转移类、输入/输出类等。

表 4-2　机器语言程序

存储器地址（十六进制）	机器指令（十六进制）	汇编指令	注释
13BE:0100	B80000	MOV AX,0	；立即数 0 送 AX 寄存器
13BE:0103	BB0A00	MOV BX,0A	；设 BX 寄存器为计数器
13BE:0106	11D8	ADC AX,BX	；A 与 B 相加其结果送 A
13BE:0108	4B	DEC BX	；计数器减 1
13BE:0109	75FB	JNZ 0106	；判断计数器是否为 0，如不为 0，则继续进行加法运算

每一类指令（如数据传送类、算术运算类）又按照操作数的性质（如整数还是实数）、长度（16 位、32 位、64 位、128 位等）而区分为许多不同的指令，因此 CPU 往往有数以百计的不同的指令。为解决软件兼容性问题，采用“向下兼容方式”开发新的处理器指令，即所有新处理器均保留老处理器的全部指令，同时还扩充功能更强的新指令。

4.3.2　汇编语言

为了克服机器语言的缺点，在科研人员的研究工作中很快就发明和产生了比较易于阅读和理解的汇编语言。所谓汇编语言，就是采用英文字母、符号来表示指令操作码、寄存器、数据和存储地址等，并在程序中用它们代替二进制编码数，这样编写出来的程序就称为符号语言程序或汇编语言程序。大多数情况下，一条汇编指令对应一条机器指令，少数对应几条机器指令。下面是几条汇编指令的操作符及它们代表的含义。

ADD	加法	SUB	减法
MOV	传送	MUL	无符号乘法
JMP	无条件转移	CMP	比较指令

尽管汇编语言比机器语言容易阅读理解和便于检查，但是，计算机不懂得任何文字符号，它只能接收由 0、1 组成的二进制代码程序，即目标程序。因此，有了汇编语言，就得编写和设计汇编语言翻译程序（简称汇编程序），专门负责把汇编语言书写的程序翻译成可直接执行的机器指令程序，然后再由计算机去识别和执行。一般说来，汇编程序被看成系统软件的一部分，任何一种计算机都配有只适合自己的汇编程序。汇编语言程序的执行过程如图 4-2 所示。

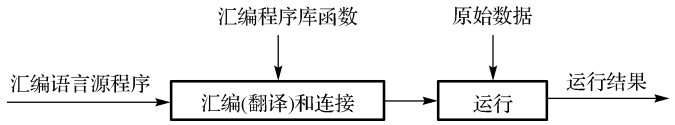


图 4-2　汇编语言程序的执行过程

汇编语言的抽象层次很低，与机器语言一样，是与具体的机器密切相关的，仍然是面向机器的语言。只有熟悉了机器的指令系统，才能灵活地编写出所需的程序。而且针对某一种机器编写出来的程序，在别的型号的机器上不一定可用，即可移植性较差。一些复杂的运算通常要用一个子程序来实现，而不能用一个语句来解决，因此用汇编语言编写程序仍然相当麻烦。尽管如此，从机器语言到汇编语言，仍然是前进了一大步。这意味着人与计算机的硬件系统不必非得使用同一种语言。程序员可以使用较适合人类思维习惯的语言。随着计算机程序设计技术的发展而出现的高级语言可以避免汇编语言的这些缺点。

4.3.3 高级语言

1. 高级语言概述

高级语言的出现是计算机编程语言的一大进步。它屏蔽了机器的细节，提高了语言的抽象层次，程序中可以采用具有一定含义的数据命名和容易理解的执行语句。这使得在书写程序时可以联系到程序所描述的具体事物，比较接近人们习惯的自然语言，是为一般人使用而设计的，处理问题采用与普通的数学语言及英语很接近的方式进行，并且不依赖于机器的结构和指令系统。如目前比较流行的语言有 C/C++、Visual Basic、Visual FoxPro、Delphi、Fortran、Pascal、Java、Python 等。使用高级语言编写的程序通常能在不同型号的机器上使用，可移植性较好。

20 世纪 60 年代末开始出现的结构化编程语言进一步提高了语言的抽象层次。结构化数据、结构化语句、数据抽象、过程抽象等概念，使程序更便于体现客观事物的结构和逻辑含义。这使得编程语言与人类的自然语言更接近。但是二者之间仍有不少差距。主要问题是程序中的数据和操作分离，不能够有效地组成与自然界中的具体事物紧密对应的程序成分。

用高级语言编写的源程序，也必须翻译成目标程序（机器码），机器才能执行。将高级语言所写的程序翻译为机器语言程序有两种程序：编译程序和解释程序。

编译程序是把高级语言程序（源程序）作为一个整体来处理的，编译后与子程序库链接，形成一个完整的可执行的机器语言程序（目标程序代码）。源程序从编译到执行的过程如图 4-3 所示。

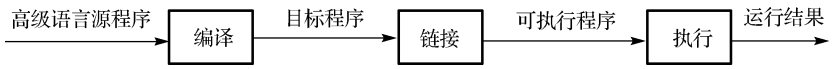


图 4-3 从编译到执行的过程

解释程序按照高级语言程序的语句书写顺序，解释一句、执行一句，最后产生运行结果，但不生成目标程序代码，解释程序结构简单、易于实现，但效率低。

高级语言语句的功能强，程序比较短，容易学习，使用方便，通用性较强，便于推广和交流。其缺点是编译程序比汇编程序复杂，而且编译出来的程序往往效率不高，其长度比有经验的程序员所编写的同样功能的汇编语言程序要长一半以上，运行时间也要长一些。因此，在实时控制系统中，有些科研人员仍然使用汇编语言进行程序设计。

2. 高级语言的特点

1）名称说明

预先说明程序中使用的对象的名称，使编译程序能检查程序中出现的名称的合法性，从而能帮助程序员发现和改正程序中的错误。

2）类型说明

通过类型说明用户定义的对象的类型，从而确定了该对象的使用方式。编译程序能够发现程序中对某个特定类型的对象使用不当的错误，因此有助于减少程序错误。

3）初始化

为减少发生错误的可能性，应该强迫程序员对程序中说明的所有变量初始化。

4）程序对象的局限性

程序设计的一般原理是，程序对象的名称应该在靠近使用它们的地方引入，并且应该只有程序中真正需要它们的那些部分才能访问它们，即局部化和信息隐蔽原理。

5）程序模块

模块有一系列优点：第一，可以构造抽象数据类型，用户可以对这种数据进行操作，而并不需要知道它们的具体表示方法；第二，可以把有关的操作归并为一组，并且以一种受控制的方式共享变量；第三，这样的模块是独立编译的方便单元。

6）循环控制结构

常见的循环控制结构有 for 语句、while-do 语句、repeat-until 语句等，在许多场合下，还需要在循环体内任意一点测试循环结束条件。

7）分支控制结构

常见的有单分支的 if 语句、双分支的 if…else 语句、多分支的 case 语句等。

8）异常处理

提供了相应的机制，从而不必为异常处理过分增加程序长度，并且可以把出现异常的信息从一个程序单元方便地传送到另一个单元。

9）独立编译

独立编译意味着能分别编译各个程序单元，然后再把它们集成为一个完整的程序。如果没有独立编译的机制，就不是适合软件工程需要的好语言。

用户在进行程序设计时，可根据实际情况选择高级语言。

4.3.4　高级语言程序设计类型

1．结构化程序设计

结构化程序设计诞生于 20 世纪 60 年代，发展到 20 世纪 80 年代，已经成为当时程序设计的主流方法，几乎为所有的程序员所接受和使用，它的产生和发展形成了现代软件工程的基础，结构化程序设计的基本思想是采用自顶向下、逐步求精的设计方法和单入口单出口的控制结构。自顶向下、逐步求精的方法，使所要解决的问题逐步细化，并最终实现由顺序、选择和循环这三种基本结构构成的描述。

一个复杂的问题可以划分为多个简单问题的组合，这样的划分包括两个方面：一是把问题细化为若干模块组成的层次结构；二是把每一个模块的功能进行进一步的细化，分解成为一个个更小的子模块，直到分解成一个个程序语句为止。这样设计的优点在于：

① 符合人们思考问题的一般规律，易于编写和维护。

② 把一个问题逐步细化，从相对简单的问题出发，以各个击破的策略，逐个解决问题，分析问题是一个从总体到局部的过程，而解决问题则是一个由局部到总体的过程。在自顶向下、逐步细化的过程中，把复杂问题分解成一个个简单问题的最基本方法就是模块化，按照功能或层次结构把一个问题划分为几个模块，然后对每个模块进行进一步细化。模块化分析便于问题的分析，同时也有利于程序设计以及软件工程中的组织与合作，按照模块作为工作划分的依据，各个模块可以独立地进行开发、测试和修改。

在 20 世纪 50、60 年代，软件人员在编程时常常大量使用 GOTO 语句，使得程序结构非

常混乱。结构化程序设计的概念最早由荷兰科学家 E.W. Dijkstra 提出。1965 年，他就指出“可以从高级语言中取消 GOTO 语句”，“程序的质量与程序中所包含的 GOTO 语句的数量成反比”。1966 年，Bohm 和 Jacopini 证明，只用顺序、选择和循环三种基本的控制结构（如图 4-4 所示）就能实现任何单入口单出口的程序。这为结构化程序设计技术奠定了理论基础。

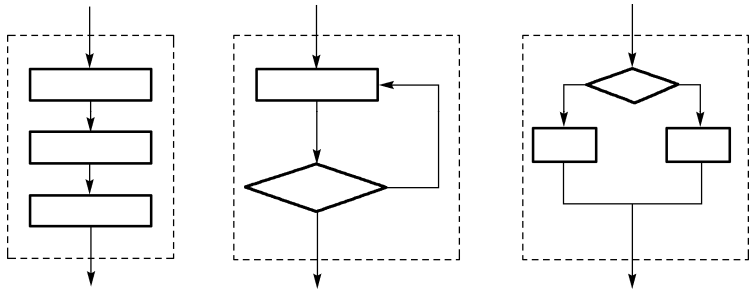


图 4-4 三种基本结构

因此，结构化程序设计主要包括两个方面：

① 在代码编写时，强调采用单入口单出口的基本控制结构（顺序、选择、循环），避免使用 GOTO 语句。这将大大改善程序的清晰度，提高程序的可读性。在结构化程序设计思想流行的 10 年中，出现了许多优秀的结构化程序设计语言，如 Pascal 语言、C 语言等，它们具有结构化的数据结构与控制结构，易读易懂易测试，容易保证程序的正确性。

② 在软件设计和实现过程中，提倡采用自顶向下和逐步细化的原则。该方法符合人类解决复杂问题的普遍规律，因此可以显著提高软件开发的成功率和生产率。

虽然结构化程序设计方法有很多优点，但是作为面向过程的设计方法，将解决问题的重点放在了如何实现过程的细节方面，把数据和对数据的操作截然分开，因而仍然有着方法本身无法克服的缺点，这样设计出来的程序，其基本形式是主模块与若干子模块的组合（如 C 语言中的 main 函数和若干子函数）。由于数据和操作代码（函数）的分离，一旦数据的格式或结构发生改变，相应的操作函数就要改写。而且对于核心数据的访问也往往得不到有效的控制。同时，如果程序进行扩充或升级改进，也需要大量修改函数。例如，当要设计一个简单的人事管理程序时，要分别设计数据结构（通常用结构体来存放人员信息，如姓名、编号、年龄、住址等）和算法（如添加、删除、查找、排序）。如果需要对数据结构进行修改，则所有相关算法也必须做相应修改。这样，程序开发的效率就难以提高，大大限制了软件产业的发展。因此出现了面向对象的程序设计方法。

2．面向对象程序设计

1）面向对象的程序设计思想

面向对象是从本质上区别于传统的结构化方法的一种新方法、新思路。它吸收了结构化程序设计的全部优点，同时又考虑到现实世界与计算机空间的关系，认为现实世界是由一系列彼此相关并且能够相互通信的实体组成，这些实体就是面向对象方法中的对象，每个对象都有自己的自然属性和行为特征，而一些对象的共性的抽象描述，就是面向对象方法中的核心——类。

面向对象的程序设计方法就是运用面向对象的观点来描述现实问题，然后再用计算机语言来描述并处理该问题的。这种描述和处理是通过类与对象实现的，是对现实问题的高度概括、分类和抽象。每个对象都具有自己的数据和相应的处理函数，整个程序是由一系列相互作用的对象来构成，不同对象之间是通过发送消息相互联系、相互作用的。

利用面向对象的程序设计方法进行软件开发，其数据和操作将很好地封装在对象中，数据的访问权限也可以得到有效的控制，数据结构和格式的改变只是局限于拥有该数据的对象和类中，因此修改也较为方便。同时，通过由现有的类进行派生，可以得到更多具有特殊功能的新类，从而实现代码的重用。这种继承和派生的机制是对已有程序的发展和改进，有利于实现软件设计的产业化。

从人类认识问题的角度来讲，结构化程序设计的方法假定在开始程序设计的时候，就对所要解决的问题有深入、彻底的认识，对程序有全面的规划，然后自顶向下、逐步细化，直到细致到可以用计算机语言描述为止，然后开始编写程序、调试、测试等。事实上，人类对问题的认识有一个逐步深入的过程，在程序设计之初，可能对问题还没有彻底的认识。因此，在程序的编写，甚至运行、测试过程中，如果有了新的认识和需要，而对其中的关键数据或算法有所改动，则有可能整个程序都会被修改。其实这种认识的过程是完全符合现实的，是必然的，这也是结构化设计无法克服的困难在哲学高度的解释。面向对象的程序设计方法就比较符合人类认识问题的客观规律。对一个具体问题进行分析、抽象，将其中的一些属性和行为抽象成相应的数据和函数，封装到一个类中，就用这个类在计算机中描述现实世界中的问题。随着编写程序或分析的不断深化，对问题有了新的理解和认识，可以通过在相应的类中加入新属性和新行为，或者可以由原来的类派生一个新的类，给这个新类加入属性和行为。例如，仍以一个简单的人事管理程序为例，可以将人员信息与对这些信息的处理方法封装在一起，构成一个“人员”类。当这个类被应用到不同的场合（如学生信息管理、教师信息管理）时，可以根据需要派生新类，并在派生的类中添加新的成员。类的派生过程，反映了人们认识问题深入程度的发展。与面向过程的程序设计方法相比，面向对象的方法为适应问题的发展而对程序进行的修改要少得多。

2）面向对象程序设计的基本特点

（1）抽象

抽象是人类认识问题的最基本手段之一。面向对象方法中的抽象是对具体问题（对象）进行概括，即忽略事物的非本质特征，只注意那些与当前目标有关的本质特征，从而抽象出一类对象的共性并加以描述。抽象的过程，就是对问题进行分析和认识的过程。对一个问题的抽象一般来讲应该包括两个方面：数据抽象和代码抽象（或称为行为抽象）。前者描述某类对象的属性或状态，也就是此类对象区别于彼类对象的特征物理量；后者描述某类对象的共同行为特征或具有的共同功能。对一个具体问题进行抽象分析的结果，是通过类来描述和实现的。例如，对人进行分析，通过对全部人类进行归纳、抽象，提取出其中的共性如姓名、性别、年龄等，它们组成了人的数据抽象部分，用 C++语言来表达，可以是：char *name，char *sex，int　age；而人类的共同行为如吃饭、行走等这些动物性行为以及工作、学习等社会性行为，构成了人的代码抽象部分，也可以用 C++语言表达：EatFood( )，Walk( )，Work( )，Study( )。

进一步来分析，如果是为一个企业开发用于人事管理的软件，这个时候来分析人，所关心的特征就不会只限于这些。除了上述人的这些共性外，这里所研究的人又有新的共性，比

如工龄、工资、工作部门、工作能力，还有他的上、下级隶属关系等。由此也可以看出，同一个研究对象，由于所研究问题的侧重点不同，就可能产生不同的抽象结果；即使对于同一个问题，解决问题的要求不同，也可能产生不同的抽象结果，这些结果的不同就表现在得出不同的抽象成员。

（2）封装

封装是面向对象方法的一个重要原则。它有两个含义：第一个含义是将抽象得到的数据成员和代码成员相结合，形成一个不可分割的整体，即**对象**，这种数据及行为的有机结合也就是封装。第二个含义称为信息隐蔽，即尽可能隐蔽对象的内部细节，通过对抽象结果的封装，将一部分行为作为对外部的接口，以便达到对数据访问权限的合理控制，把整个程序中不同部分的相互影响减少到最低限度。这种有效隐蔽和合理控制，就可以达到增强程序的安全性和简化程序编写工作的目的。

利用封装的特性，编写程序时，对于已有的成果，使用者不必了解具体的实现细节，而只需要通过外部接口，依据特定的访问规则，就可以使用这些现有的东西。在 C++中，是利用类（class）的形式来实现封装的。

（3）继承

继承是面向对象技术能够提高软件开发效率的重要原因之一。其含义是特殊类的对象拥有其一般类的全部属性与服务，称为特殊类对一般类的继承。只有继承，才可以在已有认识的基础之上有所发展，有所突破，摆脱重复分析、重复开发的困境。如 C++中就提供了类的继承机制，允许程序员在保持原有类特性的基础上，进行更具体、更详细的类的定义。这样，通过类的这种层次结构，就可以反映出认识的发展过程。新的类由原有的类产生，包含了原有类的关键特征，同时也加入了自己所特有的性质，原有的类称为基类或父类，产生的新类称为派生类。这种继承和派生的机制对程序设计的发展是极为有利的。

（4）多态

多态性是指在一般类中定义的属性和行为，被特殊类继承之后，可以具有不同的数据类型或表现出不同的行为，这使得同一属性或行为在一般类及各个特殊类中具有不同的语义。例如，一个类中有许多求两个数中最大值的行为，针对不同的数据类型，就要写很多个不同名称的函数来实现。事实上，它们的功能几乎完全相同，这时，就可以利用多态的特征，用统一的标识来完成这些功能。这样，就可以达到类的行为的再抽象，进而统一标识，减少程序中标识符的个数。在 C++中，多态是通过重载函数和虚函数等技术来实现的。

4.3.5 常用程序设计语言

通常情况下，一项任务可以使用多种编程语言来完成。当为一项任务选择语言的时候，通常有很多因素要考虑，如人的因素（编程小组的人是否对这门语言熟悉）、语言的能力（该语言是否支持所需要的一切功能，如跨平台、方便的数据库接口等），甚至还要考虑一些其他的因素，如开发这类任务的开发周期等。

有的时候，可能没有多少选择，比如通过串行口控制一个外部设备，C 语言加上汇编语言是最明智的选择，而另一些时候选择则会比较多。因此，了解一些常用的编程语言是非常有必要的。

1．BASIC 语言

BASIC 是一种易学易用的高级语言，它是 Beginner's All-Purpose Symbolic Instruction Code 的缩写，其含义是“初学者通用符号指令编码”。它是从 FORTRAN 语言简化而来的，最初是美国 Dartmouth 学院为便于教学而开发的会话语言。它自 1965 年诞生以来，其应用已远远超出教学范围，并于 1977 年开始了标准化工作。

BASIC 语言的特点是简单易学，基本 BASIC 只有 17 种语句，语法结构简单，结构分明，容易掌握；具有人机会话功能，便于程序的修改与调试，非常适合初学者学习运用。

BASIC 的主要版本有标准 BASIC、高级 BASIC、结构化 BASIC（如 QBASIC、True BASIC、Turbo BASIC）、CAREALIZER、GFA BASIC、POWER BASIC，以及在 Windows 环境下运行的 Visual BASIC。

2．FORTRAN 语言

FORTRAN 于 1954 年问世，于 1957 年由 IBM 公司正式推出，是目前仍在使用的最早的高级程序语言，在早期，FORTRAN 不便于进行结构化程序的设计和编写。

FORTRAN 是一种主要用于科学计算方面的高级语言。它是第一种被广泛使用的计算机高级语言，并且至今仍富有强大的生命力。FORTRAN 是英文 Formula Translator 的缩写，其含义是“公式翻译”，允许使用数学表达式形式的语句来编写程序。

程序分块结构是 FORTRAN 的基本特点，该语言书写紧凑，灵活方便，结构清晰，自诞生以来至今不衰，先后经历了 FORTRAN II、FORTRAN IV、FORTRAN 77、FORTRAN 90 的发展过程，现又发展了 FORTRAN 结构程序设计语言。

3．COBOL 语言

COBOL 是英文 Common Business Oriented Language 的缩写，其意为“面向商业的通用语言”。第一个 COBOL 文本于 1960 年推出，其后又修改和扩充了十几次，并逐步标准化。

COBOL 语言的特点是按层次结构来描述数据，具有完全适合现实事务处理的数据结构，具有更接近英语自然语言的程序设计风格，有较强的易读性，是世界上标准化最早的语言，通用性强。由于 COBOL 的这些特点，使其成为数据处理方面应用最为广泛的语言。

然而，用 COBOL 编写的程序不够精练，程序文本的格式规定、内容等都比较庞大，不便记忆。

4．PASCAL 语言

PASCAL 语言是系统地体现结构程序设计思想的第一种语言，既适用于数值计算，又适用于数据处理。PASCAL 语言的特点是结构清晰，便于验证程序的正确性，简洁、精致，控制结构和数据类型都十分丰富，表达力强、实现效率高、容易移植。

PASCAL 的成功在于它的以下特色：

① PASCAL 具有丰富的数据类型，有着像枚举、子界、数组、记录、集合、文件、指针等众多的用户自定义数据类型，能够用来描述复杂的数据对象，十分便于书写系统程序和应用程序。

② PASCAL 提供的语言设施体现了结构程序设计的原则，有着简明通用的语句，基本结

构少，但框架优美，功能很强；算法和数据结构采用分层构造，可自然地应用自顶向下的程序设计技术；程序可读性好，编译简单，目标代码效率较高。

5．C 语言

C 语言是 1972 年由美国的 Dennis Ritchie 设计发明的，并首次在 UNIX 操作系统的 DEC PDP-11 计算机上使用。它由早期的编程语言 BCPL（Basic Combined Programming Language）发展演变而来。在 1970 年，AT&T 贝尔实验室的 Ken Thompson 根据 BCPL 语言设计出较先进的并取名为 B 的语言， 最后导致了 C 语言的问世。C 语言功能齐全、适用范围大、良好地体现了结构化程序设计的思想，准确地说，C 语言是一种介于低级语言和高级语言间的中级语言。

6．C++语言

前面介绍的 C 语言以其简洁、紧凑，使用方便、灵活、可移植性好，有着丰富的运算符和数据类型，可以直接访问内存地址，能进行位操作，能够胜任开发操作系统的工作，生成的目标代码质量高等独有的特点风靡了全世界。但由于不支持代码重用，因此，当程序的规模达到一定程度时，程序员很难控制程序的复杂性。

1980 年，贝尔实验室的 Bjarne Stroustrup 开始对 C 语言进行改进和扩充。1983 年，正式将其命名为 C++。在经历了 3 次 C++修订后，1994 年制定了 ANSI C++ 标准草案。以后又经过不断完善，成为目前的 C++。

C++包含了整个 C，C 是建立 C++的基础。C++包括 C 的全部特征和优点，同时能够完全支持面向对象编程（OOP）。目前，在应用程序的开发之中，C++是一种相当普遍的基本程序设计语言，开发环境由 UNIX 到 Windows 都可以使用 C++。

C++对面向对象程序设计（OPP）支持以下功能：C++支持数据封装；C++类中包含私有、公有和保护成员；C++通过发送消息来处理对象之间的通信；C++允许函数名和运算符重载；C++支持继承性；C++支持动态联编。

C++是一门高效的程序设计语言，而且仍在不断发展中。美国微软公司现已推出 C#（C Sharp）语言，来代替 C++语言。

7．Java 语言

Java 是在 C++相当强大后才发展起来初具规模的。它是一种面向对象的程序设计语言，以一组对象组织程序，与 C++相比，Java 加强了 C++的功能，但去除了一些过于复杂的部分，使得 Java 语言更容易理解，并且容易学习。比如 Java 中没有指针、结构等概念，没有预处理器，程序员不用自己释放占用的内存空间，因此不会引起因内存混乱而导致的系统崩溃。

Java 语言的特点如下：语法简单，功能强大；支持分布式处理，安全性强（内置 TCP/IP、HTTP、FTP 协议类库，三级代码安全检查）；与平台无关（一次编写，到处运行）；是解释运行的，效率较高；支持多线程（用户程序并行执行）；可动态执行；具有丰富的 API 文档和类库。

根据结构组成和运行环境的不同，Java 程序可以分为两类：Java Application 和 Java Applet。简单地说，Java Application 是完整的程序，需要独立的解释器来解释运行；而 Java Applet 则是嵌在 HTML 编写的 Web 页面中的非独立程序，由 Web 浏览器内部包含的 Java 解释器来解释运行。Java Application 和 Java Applet 各自使用的场合也不相同。除此之外，还有一种混合

型应用程序（appletcation），是指在不同的主机环境中可作为不同的类型，或者是小应用程序，或者是应用程序。

用 Java 可以开发几乎所有的应用程序类型，主要包括：多平台应用程序、Web 应用程序、基于 GUI 的应用程序、面向对象的应用程序、多线程应用程序、关键任务的应用程序、分布式网络应用程序、安全性应用程序等，它是目前使用较多的面向对象的程序设计语言。

互联网的发展产生了大量的网络应用，也促成了许多新语言的产生和流行。

从名字上看，HTML（HyperText Markup Language，超文本标记语言）和 XML（eXtensible Markup Language，可扩展标记语言）都属于语言，但对于是否是真正的计算机语言还有不同的看法，因为它们都没有传统语言的基本控制结构和复杂的数据结构定义及子程序定义。标记语言的主要用途是描述网页的数据和格式。

在互联网应用中，有大量的基于解释器的脚本语言，如 VB Script、Java Script、PHP、Java servlet、JSP 等，这些脚本语言使互联网以多种姿态的动态形式，跨越不同的硬件、系统平台运行，并且其应用开发相对于传统语言还要容易一些。

8．C#语言

C#（读为“C sharp”），是一种安全的、稳定的、简单的、优雅的，是由 C 和 C++衍生出来的面向对象的编程语言。它继承了 C 和 C++的强大功能和语法风格，同时在继承了 C++的面向对象特性的同时去掉了一些它们的复杂特性（如没有宏和模版；C#不再提供对指针类型的支持，使得程序不能随便访问内存地址空间，从而更加健壮；不允许多重继承，避免了以往类层次结构中由于多重继承带来的可怕后果）。

C#综合了 VB 简单的可视化操作和 C++的高运行效率，以其强大的操作能力、优雅的语法风格、创新的语言特性和便捷的面向组件编程的支持成为.NET 开发的首选语言，体现了当今最新的程序设计技术的功能和精华。并且 C#成为了 ECMA 与 ISO 标准规范，C#看似基于 C++写成的，但又融入了其他语言如 Pascal、Java、VB 等。

C#的特点如下：完全面向对象；支持分布式；自动管理内存机制；安全性和可移植性好；指针受限使用；支持多线程。

9．Python 语言

Python 是一种解释型、面向对象、动态数据类型的高级程序设计语言。自从 20 世纪 90 年代初 Python 语言诞生至今，它逐渐被广泛应用于处理系统管理任务和 Web 编程。Python 已经成为最受欢迎的程序设计语言之一。2011 年 1 月，它被 TIOBE 编程语言排行榜评为 2010 年度语言。自从 2004 年以后，Python 的使用率是呈线性增长。

由于 Python 语言的简洁、易读以及可扩展性，在国外用 Python 做科学计算的研究机构日益增多，一些知名大学已经采用 Python 教授程序设计课程。例如麻省理工学院的计算机科学及编程导论课程就使用 Python 语言讲授。众多开源的科学计算软件包都提供了 Python 的调用接口，例如著名的计算机视觉库 OpenCV、三维可视化库 VTK、医学图像处理库 ITK。而 Python 专用的科学计算扩展库就更多了，例如如下 3 个十分经典的科学计算扩展库：NumPy、SciPy 和 Matplotlib，它们分别为 Python 提供了快速数组处理、数值运算以及绘图功能。

因此 Python 语言及其众多的扩展库所构成的开发环境十分适合工程技术、科研人员处理实验数据、制作图表，甚至开发科学计算应用程序。

4.4 操作系统

计算机软件系统包括系统软件和应用软件，系统软件其核心是操作系统。操作系统是系统软件中一个最基本的大型软件，是全面地管理计算机软件和硬件的系统程序，是用户与计算机之间的接口。

4.4.1 操作系统的概念和功能

1．操作系统的概念

操作系统是一组程序的集合，它是系统软件的基础或核心，是最基本的系统软件，其他所有软件都是建立在操作系统之上的。一方面，它直接管理和控制计算机的所有硬件和软件，使计算机系统的各部件相互协调一致地工作；另一方面，它向用户提供正确利用软硬件资源的方法和环境，使得用户能够通过操作系统充分而有效地使用计算机。

因此，操作系统是用户与计算机系统之间的接口。它好似一个不可逾越的计算机管理中心，任何用户都必须通过它才能操作和使用计算机系统的各种资源。

2．操作系统的作用

操作系统的主要作用有以下三个：

① 提高系统资源的利用率，通过对计算机系统的软、硬件资源进行合理的调度与分配，改善资源的共享和利用状况，最大限度地发挥计算机系统工作效率，即提高计算机系统在单位时间内处理任务的能力（称为系统吞吐量）。

② 提供方便友好的用户界面，通过友好的工作环境，改善用户与计算机的交互界面。如果没有操作系统这个接口软件，用户将面对一台只能识别 0、1 组成的机器代码的裸机。有了操作系统，用户才可能方便有效地同计算机打交道。

③ 提供软件开发的运行环境，在开发软件时，需要使用操作系统管理下的计算机系统，调用有关的工具软件及其他软件资源。进行一项开发时，先问是在哪种操作系统环境下开发的，当要使用某种保存在磁盘中的软件时，还要考虑在哪种操作系统支持下才能运行。因为任何一种软件并不是在任何一种系统上都可以进行的，所以操作系统也称为软件平台。

操作系统的性能在很大程度上决定了计算机系统工作的优劣。具有一定规模的计算机系统，包括中、高档微机系统，都可以配备一个或几个操作系统。

3．操作系统的功能

从资源管理的角度来看，操作系统的功能包括：作业管理、文件管理、处理机管理、存储管理和设备管理五个方面。

① 作业管理

作业是指用户请求计算机系统完成的一个独立任务，它必须经过若干个加工步骤才能完成，其中每一个加工步骤称为作业步。作业管理包括作业的调度与控制两个方面。作业调度是指在多道程序设计系统中，系统要在多个作业中按一定策略选取若干作业，为它们分配必要的共享资源，使之同时执行。

常用的作业调度策略包括：先来先服务策略，最短作业优先策略，响应比最高者优先策略，优先数策略，以及分类调度策略等。而作业控制包括：控制作业的输入，控制被选中作业的运行步骤，控制作业执行过程中的故障处理，以及控制作业执行结果的输出等。

② 文件管理

文件管理又称为文件系统，文件是一组完整的信息集合。计算机中的各种程序和数据均为计算机的软件资源，它们以文件的形式存放在外存中。操作系统对文件的管理主要包括：文件目录管理，文件存储空间的分配，为用户提供灵活方便的操作命令（如文件的按名存取等）以及实现文件共享，安全、保密等措施。

③ 处理机管理

中央处理器（CPU）是计算机的核心部件，它是决定计算机性能的最关键的部件，而处理机管理即为 CPU 管理。因此，应最大限度地提高处理器的效率。在多道程序系统中，多个程序同时执行，如何把 CPU 的时间合理地分配给各个程序是处理机管理要解决的问题，它主要解决 CPU 的分配策略、实施方法等。

CPU 管理的另一个工作是处理中断，CPU 硬件中断装置首先发现产生中断的事件，并中止现行程序的执行，再由操作系统调出处理该事件的程序进行处理。

④ 存储管理

计算机系统的内存空间分成两个区域。一个是系统区，用于存放操作系统、标准子程序和例行程序；另一个用于存放用户程序。操作系统的存储管理主要解决多道程序在内存中的分配，保证各道的程序互不冲突，并且通过虚拟内存来扩大存储空间。

⑤ 设备管理

现代计算机系统都配置了各种各样的 I/O 设备，它们的操作性能各不相同。设备管理便是用于对这类设备进行控制和管理的一组程序。它的主要任务是：设备分配，用户提出使用外部设备的请求后，由操作系统根据一定的分配策略进行统一分配，并为用户使用 I/O 设备提供简单方便的命令。输入输出操作控制，设备管理程序根据用户提出的 I/O 请求控制外部设备进行实际的输入输出操作，并完成输入输出后的善后处理。

4．操作系统的层次结构

操作系统中定义了它的内核层和它与用户之间的接口（如图 4-5 所示）。

① 操作系统的内核

在图 4-5 中，位于操作系统中心的 Kernel 被称为内核程序，也就是说 Kernel 是操作系统的核心。Kernel 的组成部分包括：管理计算机各种资源所需要的基本模块（程序）代码，这些相应功能模块可以直接操作计算机的各种资源，例如：文件管理就是属于这类功能模块的；设备驱动程序直接和设备进行通信以完成设备操作，键盘的输入就是通过操作系统的键盘驱动程序进行的，键盘驱动程序把键盘的机械性接触转换为系统可以识别的 ASCII 码并存放到内存的指定位置，供用户或其他程序使用；内存管理可在一个多任务的环境中把现有程序调入内存运行，或者将内存分为几个部分分别供几个程序使用。在不同的时间片，CPU 在不同的内存地址范围执行不同的程序。

Kernel 核心程序还包括调度（Scheduled）和控制（Dispatcher）程序，前者决定哪一个程序被执行，后者控制着为这些程序分配时间片。

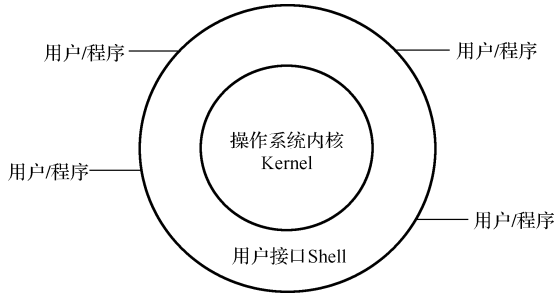


图 4-5 操作系统的内核和 Shell 结构

② 操作系统的接口 Shell

在 Kernel 和用户之间的接口部分就是 Shell 程序。Shell 最早是基于 UNIX 系统提出的概念，它是用户和 Kernel 之间的一个交互接口。早期的 Shell 为一个命令集，Shell 通过基本命令完成基本的控制操作。Shell 运行命令时，使用参数改变命令执行的方式和结果。它对用户或者程序发出的命令进行解释并将解释结果通报给 Kernel。

Shell 命令有两种方式，一种是会话式输入，会话方式表现在程序被执行过程中提供接口。另一种是命令文件方式。MS DOS 系统将 Shell 称为命令解释器（Command），在 Windows 系统中 Shell 是通过“窗口管理器”来完成这个任务的。被操作的对象如文件和程序，以图标的方式形象化地显示在屏幕上，用户通过鼠标点击图标的方式向“窗口管理器”发出命令，启动程序执行的“窗口”。

5．操作系统的启动

启动操作系统的过程是指将操作系统从外部存储设备装载到内存并开始运行的过程，Windows 操作系统的启动过程如下：

① 机器加电（或者按下 Reset）；

② CPU 自动运行 BIOS 的自检程序，测试系统各部件的工作状态是否正常；

③ CPU 自动运行 BIOS 的自举程序，从外部存储设备的引导扇区读出引导程序装入内存；

④ CPU 运行引导程序，从外部存储设备读出操作系统装入内存；

⑤ CPU 运行操作系统。

操作系统运行时内存的态势如图 4-6 所示。

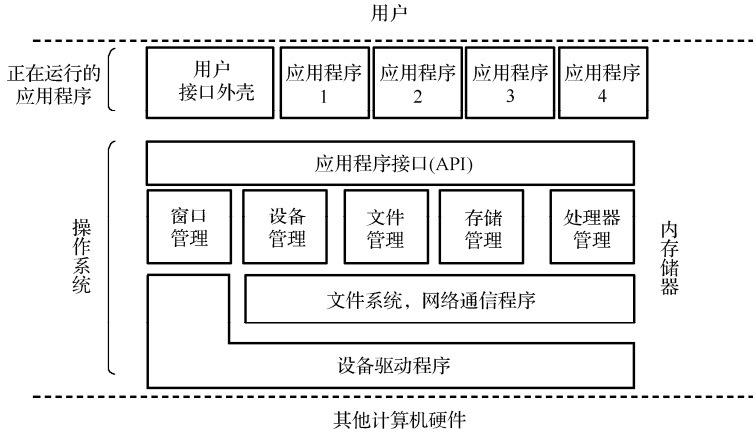


图 4-6 操作系统运行时内存的态势