



程序设计基础 (C99)

CHENGXU SHEJI JICHU

姜 沐◎著



东南大学出版社
SOUTHEAST UNIVERSITY PRESS

程序设计基础(C99)

姜 沐 著

 东南大学出版社
SOUTHEAST UNIVERSITY PRESS

· 南京 ·

图书在版编目(CIP)数据

程序设计基础:C99/姜沐著. —南京:东南大学出版社,2015.9

ISBN 978-7-5641-6056-2

I. ①程… II. ①姜… III. ①C语言-程序设计
IV. ①TP312

中国版本图书馆 CIP 数据核字(2015)第 234239 号

程序设计基础(C99)

出版发行 东南大学出版社
社 址 南京市玄武区四牌楼 2 号
网 址 <http://www.seupress.com>
出 版 人 江建中
责任编辑 夏莉莉

经 销 全国各地新华书店
印 刷 南京京新印刷厂
开 本 787 mm×1092 mm 1/16
印 张 26
字 数 540 千字
版 印 次 2015 年 9 月第 1 版 2015 年 9 月第 1 次印刷
书 号 ISBN 978-7-5641-6056-2
定 价 53.00 元

本社图书若有印装质量问题,请直接与营销部联系。电话(传真):025-83791830

目 录

第 1 章 预备知识	1
1.1 什么是编程	1
1.1.1 计算机如何工作	1
1.1.2 内存中的程序来自何处	4
1.1.3 如何制作可执行文件	4
1.2 怎样用 C 语言编程	7
1.2.1 学习 C 语言编程的条件	7
1.2.2 编写最简单的 C 程序	8
1.3 输出字符序列	12
1.3.1 输出指定图案	12
1.3.2 printf() 函数的简单用法	12
1.3.3 编写代码及测试	13
1.3.4 一些特殊字符的输出	14
1.4 C 语言的“字母”和“单词”	16
1.4.1 C 语言的“字母”	16
1.4.2 C 语言的“单词”	17
第 2 章 整数类型及其五则运算	21
2.1 整数常量	21
2.1.1 输出 123	21
2.1.2 整数常量的写法	21
2.1.3 用 printf() 函数输出整数值	22
2.1.4 整数常量的局限	23
2.2 整数类型的五则运算	23
2.2.1 加法运算	23
2.2.2 减法运算	24
2.2.3 乘法运算	24
2.2.4 关于除法的两个运算	27
2.2.5 数据类型	29
2.3 让程序“记忆”数据——变量	30
2.3.1 填数问题	30

2.3.2 用变量解决问题	35
2.4 其他整数类型	38
2.4.1 unsigned 类型	39
2.4.2 short、long 和 long long 类型	40
2.4.3 字符类型	42
2.5 卡片问题	44
2.6 运行时输入数据——调用 scanf() 函数	48
2.6.1 scanf() 函数概述	48
2.6.2 scanf() 函数的应用	49
2.6.3 scanf() 函数注意事项	50
2.6.4 良好的风格	51
第3章 运算符、表达式及语句	52
3.1 运算符的基本概念	52
3.1.1 运算符	52
3.1.2 操作数	52
3.1.3 运算符的种类	52
3.2 表达式(Expression)	53
3.2.1 表达式的种类	53
3.2.2 认识表达式	53
3.2.3 表达式的作用与副效应	54
3.2.4 表达式的左值与右值	57
3.3 运算符的优先级和结合性	58
3.3.1 从 sizeof 说起	58
3.3.2 优先级的概念	58
3.3.3 结合性的概念	59
3.4 语句(Statement)	60
3.4.1 什么是语句	60
3.4.2 什么不是语句	62
3.4.3 关于语句的误区	62
3.5 算法与数据结构	62
3.5.1 算法及其特性	63
3.5.2 分橘子问题	63
3.5.3 算法的优化	65
3.5.4 什么是数据结构	66
3.5.5 找对手问题	66
3.5.6 大数相加	67
3.5.7 算法的表示	70

第4章 选择与判断	72
4.1 if 语句.....	72
4.1.1 语法形式	72
4.1.2 判断奇偶问题	73
4.1.3 防患未然	74
4.1.4 对 if 语句的详细说明	76
4.2 判等运算、关系运算及逻辑运算	78
4.2.1 判等运算	78
4.2.2 关系运算	82
4.2.3 逻辑运算	89
4.3 if-else 语句.....	95
4.3.1 代码回顾	95
4.3.2 if-else 语句的语法形式.....	95
4.3.3 倒水问题	97
4.4 难解的嵌套	103
4.4.1 自我折磨式写法	103
4.4.2 可读性和良好的习惯	104
4.4.3 减少嵌套	110
4.4.4 另一种风格	113
4.5 条件表达式	116
4.6 多项选择——switch 语句	117
4.6.1 先乘电梯再走下楼	117
4.6.2 先乘电梯再跳楼	121
4.6.3 switch 语句的一般形式	125
第5章 循 环	126
5.1 while 语句	126
5.1.1 while 语句的语法要点	126
5.1.2 良好风格	127
5.1.3 次数确定的循环	127
5.1.4 次数不定的循环	128
5.1.5 逗号表达式及其应用	133
5.2 do-while 语句	134
5.2.1 统计字符数目问题	134
5.2.2 do-while 语句的语法要点	135
5.2.3 良好风格	135
5.2.4 求逆序数问题	136
5.3 for 语句	137

5.3.1	语法要点	137
5.3.2	++、--	139
5.3.3	Fibonacci 数列	140
5.3.4	复杂的循环	141
5.3.5	正确书写表达式	141
5.4	不规则的循环及对循环的修整	142
5.4.1	循环语句中的 break 语句	142
5.4.2	continue 语句	146
5.5	循环的嵌套与穷举法	147
5.5.1	循环的嵌套	147
5.5.2	穷举法	152
5.6	goto 语句	155
5.7	浮点类型	156
5.7.1	浮点类型数据的存储模型	156
5.7.2	float 类型、double 类型与 long double 类型	156
5.7.3	double 类型常量的写法	157
5.7.4	浮点类型数据的运算	157
5.7.5	浮点类型数据的输出	158
5.7.6	浮点类型数据的输入	159
5.8	近似计算	159
5.8.1	求调和级数和	159
5.8.2	误用浮点类型数据	161
5.8.3	求一元二次方程的根	162
5.8.4	求立方根(迭代和逼近)	163
5.8.5	求 sin 函数值(通项计算)	164
第 6 章	函数及结构化程序设计	167
6.1	函数的调用、声明和定义	167
6.1.1	从初学者常犯的一个错误谈起	167
6.1.2	“()”运算符	168
6.1.3	函数类型声明	169
6.1.4	函数定义	170
6.1.5	return 语句	173
6.2	结构化程序设计	173
6.2.1	限制使用 goto 语句	173
6.2.2	自顶向下	174
6.2.3	逐步细化	174
6.2.4	模块化	174

6.2.5 求调和级数的和	174
6.3 递归	181
6.3.1 什么是递归	181
6.3.2 Hanoi 塔问题	187
6.3.3 间接递归	190
6.4 局部变量的作用域及生存期	191
6.4.1 作用域(Scope)	191
6.4.2 作用域重叠问题	192
6.4.3 对局部变量的进一步修饰	192
第7章 指向数据对象的指针	196
7.1 传值调用的局限	196
7.1.1 约分问题	196
7.1.2 对错误的分析	197
7.2 什么是指针	198
7.2.1 指针是一类数据类型的统称	198
7.2.2 指针是派生数据类型	198
7.2.3 指针是一类数据的泛称	199
7.2.4 指针专用的类型说明符——“*”	199
7.2.5 指针的分类	199
7.3 指向数据对象的指针	200
7.3.1 什么是数据对象	200
7.3.2 “&”运算符	200
7.3.3 数据指针变量的定义	202
7.3.4 指针的赋值运算	203
7.3.5 “*”运算符	204
7.4 指针的应用与误用	206
7.4.1 指针有什么用	206
7.4.2 重新构造约分函数	209
7.4.3 对指针的误用及预防	210
7.4.4 再求分橘子问题	210
第8章 数组与指针	213
8.1 使用数组	213
8.1.1 老式的解决办法	213
8.1.2 定义数组	213
8.1.3 如何称呼数组中的各个数据对象	214
8.1.4 完整的演示	214
8.2 深入理解数组	215

8.2.1	数组是一种数据类型	215
8.2.2	数组定义的含义	216
8.2.3	数组名是什么	216
8.2.4	一维数组元素的引用	217
8.2.5	数组元素引用是一个表达式	217
8.2.6	数组名的值	218
8.2.7	重复一遍	218
8.3	熟练应用一维数组	219
8.3.1	一维数组的遍历	219
8.3.2	翻卡片问题	220
8.3.3	筛法	222
8.3.4	一维数组的初始化	224
8.4	数组与函数	225
8.4.1	数组名的值和类型	225
8.4.2	对应的形参	225
8.4.3	调用原理	226
8.4.4	不可以只有数组名这一个实参	227
8.4.5	const 关键字	228
8.4.6	排序问题	231
8.5	一维数组与指针	233
8.5.1	数组名是什么	233
8.5.2	数据指针与整数的加减法	237
8.5.3	数据指针与整数的加法的应用举例	238
8.5.4	数据指针的其他运算	241
8.5.5	与数组名对应的形参	243
8.5.6	指向数组的指针	245
8.6	使用指针	246
8.6.1	通过指针操作数组	246
8.6.2	返回指针的函数	249
8.7	二维数组	251
8.7.1	二维数组的定义	251
8.7.2	二维数组元素的初始化	253
8.7.3	二维数组元素的引用和遍历	253
8.7.4	向函数传递二维数组	255
8.8	高维数组名的性质	257
8.8.1	高维数组名是指针	257
8.8.2	高维数组名是内存	259

8.8.3	<code>a[0]</code> 或 <code>*a</code> 的含义	260
8.8.4	数组与指针关系的总结	261
第9章	字符串、字符数组及指向字符的指针	263
9.1	字符串文字量	263
9.1.1	字符串文字量的定义	263
9.1.2	字符串文字量的性质	264
9.2	字符串的输入与存储	266
9.2.1	为输入的字符串准备存储空间	266
9.2.2	<code>puts()</code> 函数	267
9.2.3	字符数组的初始化	267
9.3	字符串操作的应用	267
9.3.1	求字符串长度	267
9.3.2	比较两个字符串的大小	269
9.3.3	<code>scanf()</code> 函数中的转换	271
9.3.4	字符处理库函数	275
9.4	常用的字符串函数	276
9.4.1	字符串处理库函数	276
9.4.2	<code>sscanf()</code> 与 <code>sprintf()</code> 函数	278
9.4.3	<code>restrict</code> 关键字(C99)及 <code>memcpy()</code> 函数集	278
9.4.4	字符串转换函数	279
9.5	<code>main()</code> 函数的参数	280
9.5.1	指向指针的指针	280
9.5.2	<code>main()</code> 函数的第二种写法	280
9.6	枚举类型	282
第10章	结构体与共用体	285
10.1	结构体	285
10.1.1	从一个简单例题说起	285
10.1.2	声明结构体的类型	286
10.1.3	定义结构体变量	287
10.1.4	结构体数据的基本运算	288
10.1.5	结构体变量赋初值及成员值的输入问题	290
10.1.6	结构体类型的常量(C99)	291
10.1.7	一个不太专业的技巧	293
10.1.8	结构体的其他定义方式及无名的结构体	294
10.2	结构体与指针	295
10.2.1	类型问题	295
10.2.2	通过指针读写结构体的成员	296

10.3 共用体	297
10.3.1 概述	297
10.3.2 对 double 类型的解析	299
10.4 位运算	301
10.4.1 位运算符	301
10.5 位段	308
10.5.1 位段概述	308
10.5.2 如何定义位段	308
10.5.3 位段的性质	309
10.5.4 对齐等问题	309
第 11 章 数据类型的深入讨论	310
11.1 其他类型的指针	310
11.1.1 指向函数的指针	310
11.1.2 指向虚无的指针	316
11.2 复杂数据类型的构造方法和解读	317
11.2.1 复杂数据类型的构造方法	317
11.2.2 复杂数据类型的解读	321
11.2.3 添乱的 const 等类型限定符	324
11.3 更自由地使用内存	325
11.3.1 100!=?	325
11.3.2 初级的办法	325
11.3.3 使用动态分配内存函数	328
11.3.4 改进的方法	329
11.3.5 用链表解决问题	331
11.4 typedef	335
第 12 章 程序组织与编译预处理	337
12.1 编译预处理简介	337
12.1.1 预处理的一般特点	337
12.1.2 预处理的几个阶段	338
12.2 文件包含	338
12.3 宏定义与宏替换	339
12.3.1 类似对象的宏	340
12.3.2 类似函数的宏	340
12.3.3 拼接单词	342
12.4 预处理命令的其他话题	343
12.4.1 再谈宏	343
12.4.2 其他编译预处理命令	345

12.5 使用外部变量	347
12.5.1 外部变量	347
12.5.2 static 函数	350
第 13 章 程序的输入与输出	351
13.1 面向文件的输入与输出	351
13.1.1 把程序输出写入文件	351
13.1.2 C 程序怎样读文件	356
13.1.4 fprintf()与 printf()函数的等效性	362
13.2 文件、流、FILE 及 FILE *	363
13.2.1 文件	363
13.2.2 流(stream)	364
13.2.3 FILE 结构体	364
13.2.4 FILE *	364
13.2.5 文本流和二进制流	365
13.2.6 自动打开的流	366
13.2.7 EOF	366
13.2.8 其他几个用于文本文件的 I/O 函数	366
13.3 二进制文件的读写	367
13.3.1 二进制流	367
13.3.2 用 fwrite()函数写二进制文件	367
13.3.3 用 fread()函数读二进制文件	369
13.3.4 feof()函数和 ferror()函数	370
13.3.5 讨论	372
13.4 定位问题	372
13.4.1 ftell()函数	372
13.4.2 fseek()函数	372
13.4.3 rewind()函数	373
13.4.4 fgetpos()函数和 fsetpos()函数	373
第 14 章 标准库简介	374
14.1 使用标准库的一些常识	374
14.1.1 标准头与标准头文件	374
14.1.2 使用库的禁忌	375
14.1.3 并存的宏与函数	376
14.1.4 函数定义域问题	376
14.2 对语言的补充	376
14.2.1 标准定义头文件 stddef.h	377
14.2.2 iso646.h	377

14.2.3	limits.h 和 float.h	378
14.2.4	stdarg.h	378
14.2.5	stdbool.h(C99)	378
14.2.6	stdint.h(C99)	379
14.3	stdio.h	380
14.3.1	数据类型	380
14.3.2	宏	380
14.3.3	函数	381
14.4	通用函数头文件:stdlib.h	383
14.4.1	数值转换函数	383
14.4.2	伪随机数序列生成函数	384
14.4.3	内存管理函数	384
14.4.4	环境通信函数	384
14.4.5	查找与排序函数	388
14.4.6	整数算术函数	389
14.4.7	多字节、宽字节字符和字符串转换函数	389
14.5	string.h	389
14.6	数值计算	389
14.6.1	math.h(C89)	390
14.6.2	math.h(C99)	391
14.6.3	complex.h(C99)	393
14.6.4	tgmath.h(C99)	393
14.6.5	fenv.h(C99)	393
附录 A	C 语言的关键字	395
附录 B	C 语言的数据类型	396
附录 C	ASCII 表	397
附录 D	C 语言的运算符	398
附录 E	Dev-C++ 使用简介	399
附录 F	VC++6.0 的使用	401
参考文献		404

第1章 预备知识

有限的手段,无限的运用。

1.1 什么是编程

编程(Programming)就是编写计算机程序(Program)。那么,什么是程序?
要理解“程序”这个概念,必须首先了解计算机工作的基本原理。

1.1.1 计算机如何工作

1. 计算机的组成

通常我们所看到的计算机叫做 PC(Personal Computer)机,它一般至少由键盘、显示器、主机等部分组成。

机箱上的按钮用于开机或关机。开机启动操作系统之后,就可以通过键盘(或鼠标)来让计算机执行人们所要求的任务。

由于计算机主要通过键盘(或鼠标)接受外部的命令,所以键盘和鼠标在功能上属于计算机的输入设备(Input Device)。任务的完成情况通常会显示在显示器或打印机上,因而显示器或打印机都是输出设备(Output Device)。计算机的整体外观与运行情况如图 1-1 所示。

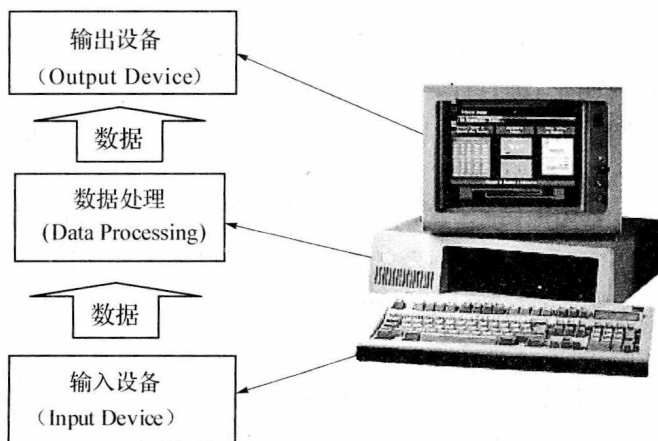


图 1-1 计算机外观与运行情况

从外部宏观地考察整个计算机的运行情况,可知计算机无外乎是从输入设备接受命令、获得数据,在内部按照命令的要求对数据进行处理之后,再把结果输出到输出设备上的一种电子设备。当然,输入设备不一定非得是键盘和鼠标,输出设备也不一定非得是

显示器或打印机。

计算机对数据的处理工作主要是在机箱内部的主机中完成的。

在 PC 机箱内部一般有一块很大的电子线路板叫做主板(Main Board), 主板上面的所谓的总线(Bus)连接着计算机所有的设备, 计算机的主要工作都是在这里完成的。在主板上有几个关键部件: CPU、ROM、RAM (参见图 1-2)。

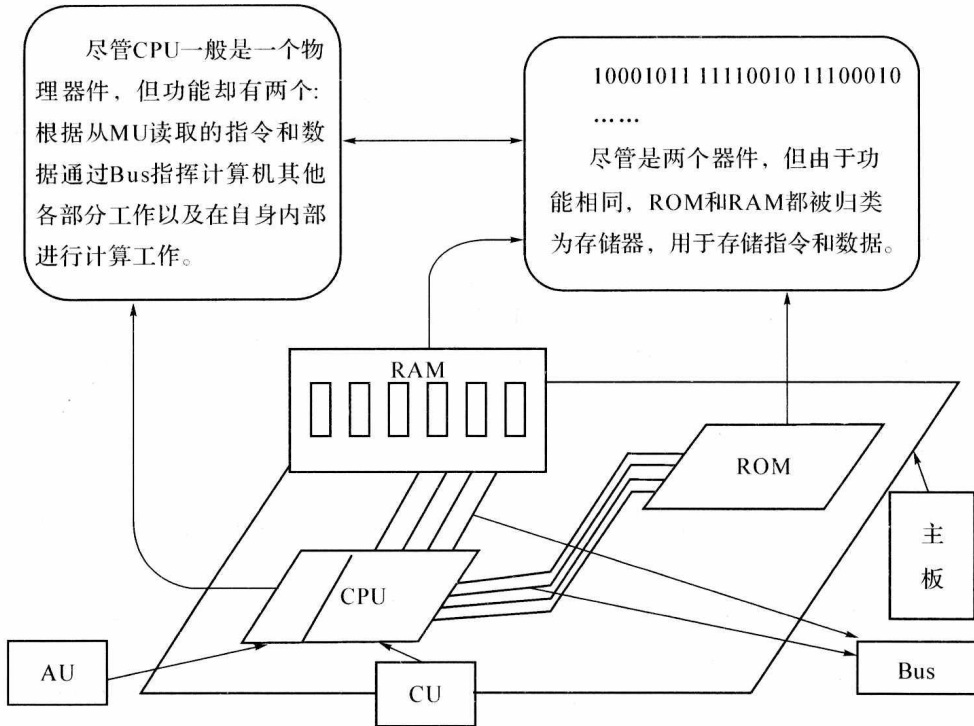


图 1-2 计算机内部工作原理示意图

2. CPU

CPU 是中心控制器(Central Process Unit)的简称。尽管 CPU 一般是一个单独的物理器件,但其功能却有两个:进行计算和指挥计算机其他各部分工作。CPU 进行计算工作的那部分叫做运算器(Arithmetic Unit,简称 AU),指挥计算机其他各部分工作的那部分被称为控制器(Control Unit,简称 CU)。

CPU 的计算工作是由 CPU 内部的运算器部分完成的;指挥计算机其他各个部分完成指定动作则是由 CPU 内部的控制器部分通过总线(Bus)发出电信号实现的。这里的计算只是针对有限位数二进制数的算术运算,或者判断某个二进制数是否为 0 这样简单的逻辑运算。

我们要求计算机执行的任务都是被分解成这样极小的计算和极其简单的“小动作”的有序组合来完成的。尽管这种“小动作”极其微小而琐碎,但由于计算机完成得很快,所以平时我们感觉不到这一点。

为什么 CPU 需要从内存中读取工作命令和数据呢? 因为如果 CPU 直接从键盘接受一个一个的“小动作”的命令再转发给计算机的各个部件的话,那么 CPU 会因为人的动作相对极慢而处于长时间等待的状态(这种情况和使用计算器类似),这样就无法发挥

CPU 高速、高效的特性。

此外大多数人既可能不了解那么多计算机的“小动作”，也可能不懂得如何把让计算机执行的任务分解成这样的“小动作”，所以使用计算机时一般只是通过键盘或其他输入设备发出一个关于任务的总的命令，而这个总的命令或任务所分解成的各个“小动作”或运算的命令则是存储记录在 RAM 和 ROM 中的。

3. 内存

尽管通常是两个物理器件，但由于功能相同，ROM 和 RAM 这两种元件都被归类为存储器(Memory Unit, 简称 MU)。由于计算机的工作还需要另外一些辅助的存储设备，所以 RAM、ROM 被合称为主存或内存(计算机中其他的存储设备如磁盘、光盘等则叫做辅存或外存)。要求计算机执行的任务被分解成一个个的“小动作”，在任务被执行时存储在内存之中。

目前计算机的内存由半导体材料构成，其基本元件有两种状态，分别表示 0 和 1。大量的这种基本元件的状态组成了类似下面的序列：

000000110010100101110010101010101010101011010101011010101011 这就是 CPU 执行动作和计算的依据。

如图 1-3 所示，内存中的每个元件的状态(0 或 1)被称为一个 bit 或“位”(和平时使用十进制数时的“位”的概念相似，只是这里每一个位只能写一个 0 或一个 1，而十进制数的每一个位可以写 0~9 间的任意一个数字)，每若干个相邻的元件被划分为一组(通常是 8 个 bit 一组)，一个这样的元件组被称为一个字节(Byte)，就如同下面这样：

00000011 00101001 01110010 10101010 10101010 10101101 01010110 10101011

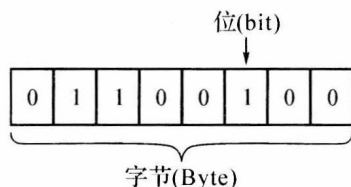


图 1-3 位与字节

每若干个连续的字节可能表示让计算机执行的一个动作，也可能表示一个需要被处理的数据，控制器每次读取若干字节，通过总线将其变成电信号，让计算机执行各种对应的操作或计算，完成之后再读取后面的若干字节继续执行。让计算机完成一个动作的一组二进制数叫做一条指令(Instruction)。

由此可见，所谓指令就是能被计算机 CPU 识别并执行的二进制代码，它确定了计算机的某一个具体操作。计算机的运行是在控制器的指挥、协调下完成的。

然而控制器本身并不了解为完成某个任务究竟应该让计算机各个部分执行哪些动作，控制器是根据存储器(这里指的是内存，也叫主存)中的内容来指挥各个部件应该做什么样的动作的。如果把计算机看成录音机的话，内存就相当于磁带，而 CPU 则相当于磁头和电磁—声音信号的转换元件。磁带上录制的是什么声音，录音机就会根据录制好的电磁信号播出什么样的声音。

用于解决某个具体问题或完成一项特定任务的许多指令的有序集合就叫程序。这种把计算机要执行的动作写成一系列二进制数形式的指令，并在执行前将之存储在内存

中,再由控制器自动读取执行的思想就是所谓“存储程序控制原理”。

1.1.2 内存中的程序来自何处

现在我们已经知道计算机是如何运行的了。随之而来的问题是,程序是如何存储到内存中以及程序是如何编写出来的呢?

第一个问题的答案是,大多数程序平时都是以文件的方式存储在外部存储设备中的,这些文件的扩展名通常是“.EXE”,叫做可执行文件。

在需要执行这些可执行文件所表示的程序时,可以通过双击代表其文件的图标,或在命令行键入要执行的可执行文件的名称后按回车键,如图 1-4 所示,之后这个文件的内容就会被操作系统复制到内存,然后由控制器逐条读取并执行。

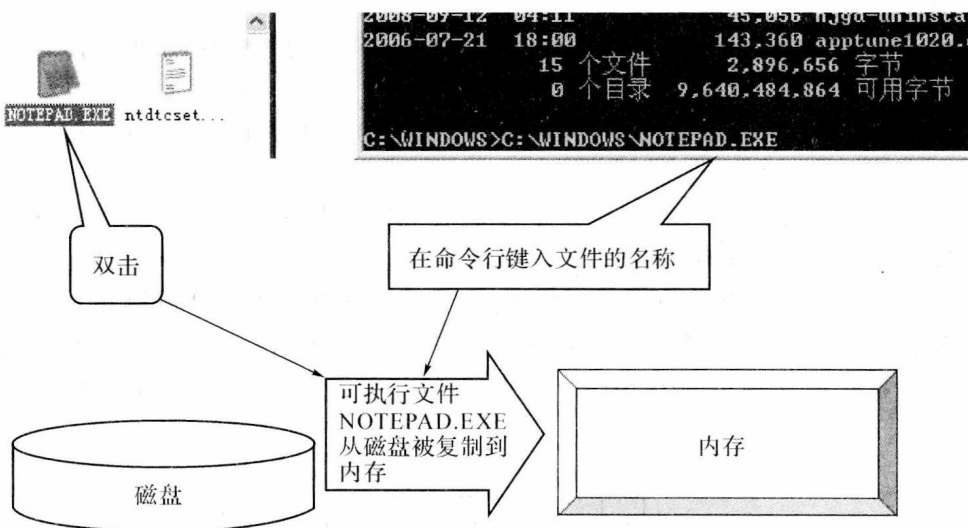


图 1-4 可执行文件的装载

把可执行文件复制到内存的工作一般是由操作系统完成的,内存中的程序是可执行文件的一个“映像”。因此只要有了可执行文件,就相当于完成了程序的制作。

那么,这些文件是怎么生成的呢?如何才能把我们的要求变成一条条的 01101110……那样的指令,并“制造”出由这样的指令构成的可执行文件?下面的内容将回答这个问题。

1.1.3 如何制作可执行文件

如前所述,在可执行文件中,所有指令和数据都用二进制代码形式表示。

1. 机器语言

直接用二进制代码表示给计算机的命令的编程语言叫做机器语言(Machine Language),这是计算机(CPU)唯一能够识别的语言。一般来说,不同型号的计算机(CPU)的机器语言是互不相同的。由于机器语言是 CPU 直接使用的语言,与人类的语言相距甚远,因此它被称为“低级语言”。

毫无疑问的是,完成下面这样用机器语言写出的程序

```
00000011 00101001 01110010 11101010
10101010 10101101 01110110 10001011
```