

开源软件丛书

第二版

LINUX 上的C编程



怀石工作室 编著



中国电力出版社
www.infopower.com.cn

开源软件丛书

LINUX

上的 C 编程

(第二版)



怀石工作室 编著

中国电力出版社

内 容 提 要

本书着重讲解如何在 Linux 系统中使用 C 语言进行编程。全书共 13 章, 分四个部分。第一部分简要介绍了 Linux 系统和 C 语言; 第二部分讲述了 Linux 系统的 C 编程环境, 详细讲述了 Linux 下 C 语言编译器 (如 GCC)、调试工具 (如 GDB) 和程序自动维护工具的使用方法。第三部分详细介绍 Linux 的系统调用。最后, 是一个实现 shell 简单功能的命令解释器的实例。

本书适合有一定 C 语言基础, 有意在 Linux 系统上一试身手的程序设计人员阅读。

图书在版编目 (CIP) 数据

Linux 上的 C 编程/怀石工作室 编著. -北京: 中国电力出版社,
2001

ISBN 7-5083-0601-5

I. L... II. 怀... III. C 语言-程序设计
IV. TP312

中国版本图书馆 CIP 数据核字 (2001) 第 021846 号

中国电力出版社出版、发行

(北京三里河路 6 号 100044 <http://www.cepp.com.cn>)

三河市实验小学印刷厂印刷

各地新华书店经售

*

2001 年 5 月第二版 2001 年 5 月北京第三次印刷

787 毫米×1092 毫米 16 开本 25 印张 557 千字

定价 45.00 元

版 权 所 有 翻 印 必 究

(本书如有印装质量问题, 我社发行部负责退换)

第二版前言

《Linux 上的 C 编程》一书出版之后，在比较短的时间内就印了两次。其间，我们收到了不少读者的热情来信，也看到了一些相关网站对本书的介绍和评论，这些信息都肯定了本书的价值，也对书中存在的一些问题提出了善意的批评。在此，对广大读者的支持表示衷心的感谢。

根据读者的意见和建议，也为了适应 Linux 本身的发展，我们在 2000 年底对本书又进行了一次修订，主要是对原书中的程序进行了一次全面的检查，保证这些程序能够在最新的 Linux 系统下正确运行（主要是在流行的 RedHat Linux 环境下进行的，对在不同环境可能引起的不同结果一般也进行了说明）。另外，对原书中存在的一些错误和不准确的地方也进行了修正。不过，由于我们水平有限，因此本书不可避免地还会存在一些问题，希望读者能不吝指教。

怀石工作室

2001 年 2 月 14 日

前 言

Linux 是一个奇迹。它由一个学生的业余爱好发展而来，现在已经成为最为流行的免费操作系统。它的独特之处在于它的建立不受任何商品化软件的版权的制约，全世界都能免费、自由使用。世界各地有几十万自愿者为这个充满魅力的系统贡献着自己的才能，从初学者到计算机专业人士、还有经验丰富的黑客们，一起不断改进和维护着这个系统。Linux 是具有专业水平的操作系统，它的爱好者和用户遍及世界，许多大学与研究机构、公司、家用 PC，都在使用 Linux。Linux 年轻而富有朝气，它从诞生到现在不过 8 年时间，已经在市场上确立自己的地位和广泛的影响。

C 语言是国际上广泛使用的计算机高级语言。C 语言最初用于描述和实现 UNIX 系统，后来逐渐被广大程序员所接受，成为最受欢迎的编程语言。在其后的发展过程中，C 语言不断吸收计算机方面新的成果，引入了面向对象的概念，近年来又出现了可视化编程语言工具 Visual C/C++，使这个古老的语言又散发出新的魅力。

Linux 作为一个操作系统，一个重要的功能就是要支持用户编程；C 语言作为当前使用最广泛的编程语言，又具有多平台、移植性好的特点，因此，它们很快形成了完美的结合，为用户提供了—个强大的编程环境。本书正是从这样的结合点出发，介绍 Linux 系统中使用 C 语言编程的有关知识。

本书主要针对已有一定 C 语言编程基础，但未在 Linux 系统中使用过 C 的读者，着重讲解 Linux 系统中使用 C 语言编程特殊的地方，其它环境中 C 编程共有的知识，例如基本语法、算法等，本书不作重点，只在第一章的概述中简要地作—提示。

全书从逻辑上可以分为四部分：

第一部分即第一章，简要介绍了 Linux 系统和 C 语言，其中扼要地回顾了 C 语言的语法，为后面的阅读提供—定的基础。

第二至四章是第二部分，带领读者熟悉 Linux 系统的 C 编程环境，详细讲述了 Linux 下 C 语言编译器(如 gcc)、调试工具(如 gdb)和程序自动维护工具 make 的使用方法。

第三部分包括第五至十二章，详细介绍 Linux 的系统调用。Linux 为用户程序提供了丰富的系统调用，而且这些就是 Linux 系统作为编程环境特有的东西，因此本书的主要篇幅都用来讲解这些系统调用和它们的使用方法。首先按使用领域分别介绍了基本的系统调用，依次是文件系统、标准输入输出、进程、信号。后几章提高了一—定层次，由浅入深的讲解 IPC (进程间通信)、Daemon 进程、网络 Socket 编程、设备编程等内容。由于是编程介绍，本书中会附带较多的实例和源代码，既方便说明和理解，也可以使读者通过阅读实例获得—定的感性认识。

最后，我们举—个综合性的例子作为全书的结束———个实现 shell 简单功能的命令解释器。在这个例子中，会涉及到第 5 章至第 12 章间各章的内容，帮助读者复习总结学

过的系统调用。

附录中将系统调用按字母顺序列出，并注明了所涉及的章节，以便于读者的查阅。

希望读者学完本书之后，都能有所收获，成为 Linux 系统的编程高手。

限于时间和水平，书中必然存在着不少缺点，请广大读者指正。

作者

1999.10.14

目 录

第二版前言

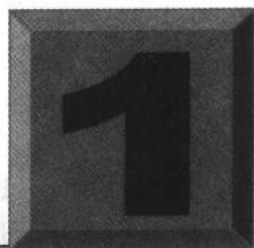
前 言

第一章 C 语言与 Linux 系统	1
1.1 Linux 系统简介	1
1.2 C 语言概述	5
1.3 Linux 系统中的 C 语言	26
第二章 GCC 编译器.....	29
2.1 GCC 的安装	29
2.2 GCC 的使用	36
2.3 其他编译调试工具	44
第三章 C 语言的调试工具	47
3.1 gdb 符号调试器	47
3.2 gdb 命令详解及简单应用举例	48
3.3 其他调试工具	74
第四章 使用 make	76
4.1 make 的简单使用	76
4.2 控制 make 的属性	81
4.3 使用宏 (macro)	91
4.4 内部规则	96
4.5 使用库	101
第五章 文件系统的操作	107
5.1 文件系统简介	107
5.2 顺序文件操作	113
5.3 随机文件操作	122
5.4 文件共享	126
5.5 索引节点	129
5.6 文件层次结构	133

5.7	改变文件属性	134
5.8	文件链接	138
5.9	设备文件	142
第六章	标准输入输出	145
6.1	简介	145
6.2	标准输入输出的基本操作	146
6.3	非格式化输入输出操作	151
6.4	格式化输入输出操作	160
6.5	临时文件的有关操作	162
第七章	进程的控制	167
7.1	进程概述	167
7.2	进程的基本操作	169
7.3	进程之间的关系	191
第八章	信号及其处理	202
8.1	简介	202
8.2	信号与信号处理函数	206
8.3	信号集及其处理	215
8.4	作业控制信号	238
第九章	进程间的通信	241
9.1	简介	241
9.2	管道	241
9.3	命名管道	253
9.4	System V 的进程间通信机制	258
第十章	守护进程的编制	280
10.1	守护进程简介	280
10.2	守护进程的编码	282
10.3	守护进程的输出	286
第十一章	网络编程	290
11.1	预备知识	290
11.2	流式套接字的基本操作	305
11.3	客户/服务器机制	323
11.4	数据报套接字的操作	330
11.5	高级技巧: select()和 poll()	337

第十二章 设备编程	348
12.1 对声音设备编程	348
12.2 对鼠标端口编程	349
12.3 对调制解调器编程	351
12.4 对打印机端口编程	354
12.5 对控制台终端编程	356
第 13 章 一个实例——自己的 Shell	361
13.1 头文件 head.h	361
13.2 主过程 main.c	363
13.3 初始化模块 init.c	363
13.4 语法分析 get_simcom.c	364
13.5 得到下一标识符 get_word.c	365
13.6 得到当前命令行 get_comln.c	366
13.7 执行简单命令 Run_com.c	366
13.8 执行输入的命令 Execute.c	367
13.9 分析简单命令 Get_simarg.c	368
13.10 字符串匹配 Check.c	369
附录 Linux 系统调用一览表	371

C 语言与 Linux 系统



欢迎进入本书的学习。本书主要针对已有一定 C 语言编程基础，但未在 Linux 系统中使用过 C 的读者，着重讲解 Linux 环境中 C 编程的独特之处。第一章简要介绍 Linux 系统和 C 语言的基本知识，为后面的学习提供了一定的基础。在之后的章节中，我们将熟悉 Linux 系统的 C 编程环境，并逐一学习 Linux 各种独特的系统调用。希望读者学完本书之后，都能有所收获，成为 Linux 系统的编程高手。

1.1 Linux 系统简介

在奥斯卡的颁奖晚会上，获得最佳导演奖的《泰坦尼克号》导演领奖时说了影片中男主角的一句台词：“我是世界之王！”他的确值得骄傲，《泰坦尼克号》风靡了整个世界，并一举夺走 11 项奥斯卡奖项。然而，泰坦尼克的成功有很大一部分要归功于 Linux。影片制作过程中用的 160 台 Alpha 工作站，有 105 台使用的是 Linux。没有 Linux，就没有影片逼真的视觉效果、精致的画面。可以说没有 Linux，就没有泰坦尼克。

本节将介绍 Linux 的背景知识，包括它的历史和特点等。希望读者阅读之后能对 Linux 有一个大概的了解。

1.1.1 什么是 Linux?

简单地说，Linux 就是一种主要适用于个人计算机的类似于 UNIX 的操作系统。它的独特之处在于不受任何商品化软件的版权制约，全世界都能免费、自由使用。

Linux 是互联网上的独特现象。虽然它是由学生的业余爱好发展而来，但是现在它已经成为最为流行的免费操作系统。没有版权和允许自由使用并不意味着这个系统不专业。事实上，这个系统是在科研和学术的环境下开发出来的，无数计算机专业人士和经验丰富的黑客（HACKER）们不断改进和维护着这个系统。也许很多人对此心存疑虑：免费的东西怎么会变得如此有价值？在一个由少数软件公司统治的世界，由一帮黑客们编写的东西是怎样与那些公司的产品竞争的？然而，事实证明 Linux 的确是稳定而富有竞争力的。相对于 Windows 系统令人头疼的漏洞百出，Linux 系统稳定而有效；相对于 UNIX 系统的庞大，Linux 又显得精致小巧。而且 Linux 对硬件配置的要求相对较低，显得更加“平民化”。

另外, Linux 的源代码全部公开, 任何人都能按自己的意愿对其进行修改, 使它成为“自己”的操作系统, 这一富有挑战性的工作更令无数电脑发烧友如痴如狂。

现在, 许多大学与研究机构都使用 Linux 完成日常计算任务, 人们在家用 PC 上使用 Linux, 许多公司也在使用它。Linux 绝对不是玩具, 而是具有专业水平的操作系统, 它的爱好者遍及世界。虽然现在 Linux 所占的市场份额还无法与微软的 Windows 相比, ——与 Windows 相比还是个小娃娃——但 Linux 在市场上确立自己的地位和产生广泛的影响, 所花的时间却只有 Windows 的一半。在这个商业化的社会中, Linux 实在是一个奇迹。

1.1.2 Linux 的历史

Linux 的源头要追溯到最古老的 UNIX。1969 年, Bell 实验室的 Ken Thompson 和 Dennis Richie 开始利用一台 PDP-7 计算机开发一种多用户、多任务操作系统, 在他们的共同努力下诞生了最早的 UNIX——这个名字来源于一个更古老的系统 MULTICS。早期 UNIX 是用汇编语言编写的, 但其第三个版本用崭新的编程语言 C 重新设计了。通过这次重新编写, UNIX 得以移植到更为强大的 DEC PDP-11/45 与 11/70 计算机上运行。从此, UNIX 从实验室走出来并成为操作系统的主流, 现在几乎每个主要的计算机厂商都有其自有版本的 UNIX。现在比较流行的 UNIX 版本有 AT&T 发布的 System V 和美国加州大学 Berkeley 分校的 BSD 版 UNIX。这些版本繁多、形态各异的 UNIX 系统有一些基本的共同特征, 如树形的文件结构、设备文件、shell 用户界面、以 fork 为代表的系统调用和以 ls 为代表的命令等等。这些特征在后来的 Linux 中也都继承下来了。

Linux 起源于一个学生的业余爱好, 他就是芬兰赫尔辛基的 Linus Torvalds——Linux 的创始人与主要维护者。Linus 上大学时开始学习 Minix, 这是一个功能简单的 PC 平台上的类 UNIX 操作系统。Linus 对 Minix 不是很满意, 于是决定自己编写一个保护模式下的操作系统软件。他以学生时代熟悉的 UNIX 作为原型, 在一台 Intel 386 PC 上开始了他的工作。他的进展很快, 很快得到了一个虽然不那么完善、却已经可以工作的系统。他自己这样描述当时的情况:

“……在那以后是一帆风顺的: 仍然是烦琐的编程工作, 但我有了一些设备, 调试工作容易些了。在这个阶段我开始使用 C 语言编程, 明显加快了开发速度。也就在此时, 我开始认真考虑一个大胆的设想: 做一个比 Minix 更好的操作系统。我梦想有一天我能在 Linux 下重新编译 GCC……”

“我花了两个月时间基本搭出了框架, 然后很快又有了一个磁盘驱动程序(虽然还有问题, 但在我的机器上可以工作了)和一个小型文件系统。大约在 1991 年 8 月下旬, 我完成了 0.01 版本, 但还很不完善: 没有软盘驱动程序, 也干不了什么事; 但我已经被它吸引住了, 除非我完全放弃使用 Minix, 我不会停止改进我的系统。”

受工作成绩的鼓舞, 他将这项成果通过互联网与其他同学共享。1991 年 10 月, Linux 首次放到 FTP 服务器上供自由下载。有人看到了这个软件并开始分发, 每当出现新问题时, 有人会立刻找到解决办法并加入其中。最初的几个月中, 知道 Linux 的人还很少, 主

要是几十名黑客，但正是这些人修补了系统中的错误，完善了Linux系统，为Linux后来风靡全球奠定了基础。Linux正式发布那天，Linus这样说：“用户可曾想像过Minix有这样美好的一天：人们可以自己编写驱动程序？用户是否已经发现这样一个美好的计划：人们可以自己修改操作系统以适应自己的需要？”Linux正是凭着这样的挑战性和自由精神，终于成为风靡全世界的操作系统。

说到Linux，不能不提GNU。GNU是Stallman在1984年发起的自由软件基金会，其目标就是将UNIX加以改进，写出一个新的操作系统，使所有用户都能免费获得该系统以及系统的源代码。GNU先开发了UNIX的一些工具软件，如Emacs、GCC等，再开发核心Hurd。这时，Linux异军突起，逐渐取代了Hurd的地位。Linux和GNU的关系十分密切。两者的精神和目的是一致的，而且Linux使用了该基金会的版权声明和大量GNU软件，甚至Linux自身也是用它们构造而成的。图1-1是Linux的标志物。

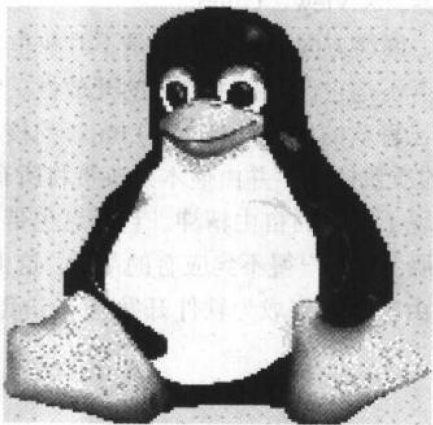


图 1-1 Linux 的标志物

1.1.3 Linux 的特点

Linux继承了UNIX系统的许多优良特性，同时又有自己的长处：

- ◆ 多任务。这是现代操作系统的基本特点，允许多个作业、多个进程、多个任务在一个时间段中同时并存，并发运行。例如，用户可以一边编译一个程序，一边在等待编译完成的过程中玩玩游戏。Linux就支持这样的多任务。它提供了后台作业、子进程、子shell等概念，并且有其独特的一套调度机制。
- ◆ 多用户。多个用户能同时从一台或多台终端上登录，使用同一应用程序的副本进行工作。多用户是网络时代分布式操作系统的特点，Linux继承了UNIX这一方面的长处。
- ◆ 多平台。Linux系统主要运行于Intel系列的个人计算机，但也支持Alpha、Sparc等平台。C语言的硬件无关性使得Linux有很好的可移植性。
- ◆ 可编程shell。Linux除了提供包括awk在内的强大命令集，还允许用户自己编

写脚本 (shell script)。在脚本中, 用户可以使用变量、控制执行的流程、使用正则表达式... 几乎你可以想像到的一切功能。

- ◆ 强大的通信和联网功能。网络连接在核心中完成, 支持多种网络协议, 常用的有 TCP、IPv4、IPX、DDP 等, 还支持其他一些协议。提供的命令有 `cu`、`uucp`、`mail`、`write`、`call`、`ftp`、`telnet` 等, 功能强大。
- ◆ 丰富的软件。经过多年的发展, 目前 UNIX 的许多配套软件都能在 Linux 上使用了; 同时, 又开发了许多为 Linux 度身定制的软件。常用的软件有:

编程工具: `gcc`、`gdb`、`make`、`perl`、`prof`

用户图形界面: X Window.

编辑器: `emacs`、`vi`、`jove`、`epock`

shell: `bash`、`zsh`、`csh`、`ash`、`rc`、`es`

图形软件编辑器: X View、Gimp

还有文字处理、多媒体、网络方面的软件, 更是不胜枚举。

然而, Linux 最大的特点还是它作为自由软件的开放性。它的代码是完全开放的, 任何人都可以将系统改造成自己喜欢的样子 (当然, Linux 的核心不是任何人都能随便乱改的, Linux 进行了非常严格的质量控制, 并由他本人负责将所有的新代码加入核心)。它摆脱了商品化软件供应商, 体现了真正的自由精神。当然这也带来了一定的问题——没有供应商意味着没有完善的售后服务, 用户得不到应有的帮助。这也正是自由软件最大的缺点。但是, 我们有理由相信, 自由软件必将成为软件开发的一个重要发展方向, Linux 会有更加蓬勃的发展。

1.1.4 Linux 的版本

Linux 的版本分为内核版本和发行套件版本。内核版本号形如 A.BB.CC, A 可以是 0、1、2, 后两位是 0~99 之间的整数。通常要选择的是发行套件的版本, 现将常见的版本列举如下:

- ◆ Slackware, Walnut Creek CDROM 公司发行。这是最早出现的 Linux 套件之一。可从 <ftp.cdrom.com> 获得。
- ◆ Red Hat Linux (小红帽 Linux), Red Hat Software 公司发行。这是目前最为流行的 Linux 套件。它支持的硬件平台多, 安装界面优秀, 安全性好, 在线文档详尽 (包括 HowTo、LDP、FAQ 等), 还有功能强大的管理程序 RPM 和丰富的软件包。可从 <ftp.redhat.com> 获得。
- ◆ Debian Linux, GNU 发行, 完全由网络上的计算机爱好者维护, 是开放式的开发环境。下载地址是 <ftp.debian.org/debian/>。

另外还有 Craftworks、Linux Pro、Trans_Ameritech Linuxware 等 Linux 套件版本, 影响不如上面几种广泛, 这里不再介绍。

1.2 C语言概述

本节中我们简单回顾一下 C 语言的语法，为后面章节的具体编程作一些准备。对 C 语言十分熟悉的读者可以直接进入下一章的学习。

1.2.1 C语言的历史和特点

C 语言是国际上广泛使用的、很有前途的计算机高级语言。它是在 B 语言基础上发展起来的。最初 C 语言用于描述和实现 UNIX 系统，后来逐渐被广大程序员所接受，成为最受欢迎的编程语言。发展到今天，C 语言已经有了很多种版本，常见的有 Microsoft C、Turbo C、Quick C 等。面向对象的概念出现之后，C 语言吸收了这个新概念，形成 C++ 语言。近年来，出现了可视化编程语言工具 Visual C/C++，使这个古老的语言又散发出新的魅力。

C 语言之所以具有如此巨大的生命力，与它的以下特点息息相关：

- ◆ 语言紧凑简洁、使用灵活。有人做过试验，同一程序用 pascal、basic、C 这几种语言书写，以 C 语言的代码总字符量为最少。
- ◆ 运算符、数据结构丰富。以赋值符号为例，除了一般语言都提供的赋值符“=”，C 还提供扩展赋值运算符。在后几小节中我们将具体介绍这些运算符和数据结构。
- ◆ 允许位操作和直接对硬件的操作。C 语言支持位操作运算，提供位的与、或、非运算。它还支持对内存地址等硬件的直接操作。也就是说，C 语言在某种意义上综合了高级语言和低级语言的优点。
- ◆ 可移植性好。一个用 C 语言书写的程序可在不同平台上运行。因为 C 语言具有这个特点，许多操作系统用 C 语言书写，以获得较好的可移植性。

也正是因为 C 语言有这样强大的功能且使用时灵活，它对程序员提出了很高的要求。C 语言语法的限制少，编译时不易检查出问题，但运行时一些隐患就会发作，因此更需要程序员具有较高的素质和丰富的经验。

1.2.2 C语言的数据类型

有一种流行的说法：程序 = 数据结构 + 算法，可见数据结构的重要性。要评价一种语言的优劣，很重要的一点就是要看它所支持的数据类型。C 语言为用户提供了极为丰富的数据类型，除了一些基本的数据类型，还允许用户自定义一些复杂的数据类型。

图 1-2 给出了 C 语言支持的所有数据类型。



图 1-2 C 语言的数据类型

常量 and 变量

在正式介绍 C 语言的各种数据类型之前，我们先简单给出常量和变量的概念。

常量是程序中其值保持不变的数据。它可以属于不同的基本数据类型。例如 4 是一个整型的常量，“A”是一个字符型的常量。常量的表示形式可以是数值本身，也可以是一个标识符（C 中的标识符是以字母开始，由字母、数字和下划线的任意组合组成的字符串，并且区分大小写）。用标识符定义常量的格式为：

```
#define 常量名 常量值
```

例如 #define pi 3.14，将 pi 定义为浮点型常量，值为 3.14，在整个程序中都不允许再赋值。但一个常量值可以对应多个常量名。

变量是指程序中值可以改变，在内存中占一定存储单元的量。使用一个变量，要先说明它的类型，由系统（编译工具）分配给它一定的内存空间，然后才能在程序中引用。变量说明的一般形式为：

数据类型 变量名；

变量名是一个标识符。例如，“int i;”说明了一个整型变量，“char ch1,ch2;”说明了两个字符型变量。数据类型可以是简单类型，也可以是数组、结构之类的复杂类型。

C 语言中要求变量都在过程开头集中说明（这与后来的 C++ 不同，习惯于 C++ 的读者要注意这些细微的差别）。

变量的作用域是一个复杂的概念，有许多规则。最基本的一条是在定义它的过程块中及其内层可用。本书对此不展开论述，请感兴趣的读者自己查阅有关书籍。

有了这些基本概念，我们下面来逐一解释图 1-2 的各种类型。

📖 整型

整型表示计算机可以表示的整数的集合，具体的集合大小与实现的计算机有关。C语言的整型又分为有符号数和无符号数两类，有符号数的最高位表示正负，无符号数都是正数，所有位都用来表示数的大小。这两类数的表示和运算在硬件层次上都有很大区别，但C语言为程序员屏蔽了这些硬件细节。每一类数又分为三种：int、short 和 long。它们的主要差别在于位数的不同，因此能表示的数的范围也不同，具体位数因机器而异。

从上面的描述我们可以看到，整型又可细分为六种情形。这六种类型之间可以互相转化，不同类的数据也可以一起运算。下面这一段程序：

```
int a = 3;
short b = 7;
long c;
c = a+b;
printf ("a+b = %d ", c);
```

最后一个语句调用了标准函数 printf，我们会在后面具体介绍。%d 指示以十进制数形式打印参数，计算机中常用的还有八进制和十六进制，分别以 0 和 0x 开头。

程序的运行结果是：a+b=10。a, b, c 是不同类型的变量，但它们可以一起运算，并互相赋值。不同类型的数据参与运算并互相转化的具体规则我们会在下一小节“表达式”中详细阐述。

这里再提一下，C语言中不提供布尔类型。如果需要表示布尔量的概念，也用整数表示——C中认为 0 是“假”，其他非零的整数是“真”。在表达式中将布尔量也作为整数处理，允许布尔量参与整数的运算，也允许整数出现在布尔表达式中。例如：

```
while (3)
{ 循环体 }
```

while 语句是一种循环语句，当括号中的条件为真时执行循环体，否则跳过循环。这里的 3 是非零整数，可以作为布尔量使用，认为是“真”，因此条件永远成立，一直执行这个循环。这是一个死循环。

📖 浮点型

浮点型表示实数的集合。因为计算机只能保存有限位的数字，所以浮点型一般都是近似值。

计算机中的浮点型数有两种表示方法：一般表示法和科学表示法。一般表示法由整数部分、小数点、小数部分组成，如 3.14、123.456、-0.00032 等；科学表示法由尾数和指数两部分组成，如 -3.2e-4，适合表示很大或很小的数。

浮点型也可再细分为两种：单精度（float）与双精度（double）。它们的具体实现牵涉

到 IEEE 的浮点数标准，但主要的差别就是位数的不同，从而导致表示范围和精度的不同。一般的系统中，一个单精度浮点数在内存中占 4 个字节，有效范围约为 10^{-38} 到 10^{38} 之间，一个双精度数则占 8 个字节。

□ 字符型

字符型表示字符集中的可打印字符。常用的字符集是 ASCII 字符集，有 95 个可打印字符。某些特殊符号可以用多个符号的组合来表示，称为转义字符。常用的转义字符有：

<code>\n</code> 换行；	<code>\t</code> 制表符 (tab)；
<code>\b</code> 退格；	<code>\\</code> 反斜杠 (\)；
<code>\'</code> 单引号；	<code>\r</code> 回车；

字符型的变量在对应的内存单元中存放的其实是该字符相应的 ASCII 码，使用时再依据该 ASCII 码转换成相应的字符。因此，在 C 语言中，`char` 和 `int` 两种类型可以通用。字符型数据可以以整数形式输出，也可以作为字符输出。而且字符型和整型的数据可以一起参与算术运算。如：

```
printf ("%d", 97- 'a');
printf (" %c", 97- 'a');
```

字符常量 `'a'` 的 ASCII 值是 97，字符型数据参与运算时使用的是与它对应的 ASCII 码，因此 `97- 'a'` 即 `97-97`，等于 0，对应的字符是一个空字符。所以，第一个语句的输出结果是 0，第二个语句的输出是空。

字符型和整型变量还可以互相赋值。如：

```
char c = 65;
int a = 'a';
```

这两条说明语句的结果是：字符型变量 `c` 的初值是 `'A'`，整型变量 `a` 的初值是 97。

在其他高级语言中，也提供了一定的机制使整型和字符型可以互相转化，如 BASIC 语言中的 `ASC` 和 `CHR` 函数，PASCAL 中的 `CHR` 和 `ORD` 函数。这些语言的运算对数据类型同一的要求很高，上述转换都要通过显式的标准函数调用实现，显然 C 语言要灵活而简洁得多。

需要注意的是字符与字符串的区别。字符常量用单引号引用，而字符串常量用双引号引用。因此，形如 `char c = "a"` 的语句是错误的，因为它试图将一个字符串赋值给一个字符变量。这与它们的硬件实现有关。如前所述，字符变量的内存地址中放的是它的 ASCII 码整数值，占用一个字节；而字符串除了其中的每个字符占用一个字节，还要有一个额外的单元存放空操作字符 `\0`，作为结束标志。因此，字符串 `"a"` 在内存中占用两个字节：32 `\0`，显然它不能赋值给一个字符变量。