

Linux

环境下 C编程指南

杨树青 王欢 编 著

- ◎ 通过大量程序实例介绍Linux平台下进行C程序开发的方法和技巧。
- ◎ 全面介绍C语言编译器、调试工具和自动维护工具的使用方法。
- ◎ 详细介绍进程、文件的相关操作以及输入/输出操作和内存管理。
- ◎ 重点介绍C语言网络编程基本原理和编程技术。



清华大学出版社

Linux

环境下C编程指南

LINUX HUAN JING XIA C BIAN CHENG ZHI NAN

杨树青 王欢 编 著

清华大学出版社
北 京

内 容 简 介

本书系统地介绍了在 Linux 平台下用 C 语言进行程序开发的过程,通过列举大量的程序实例,使读者很快掌握在 Linux 平台下进行 C 程序开发的方法和技巧,并具备开发大型应用程序的能力。本书内容翔实,主要包括:Linux 平台下 C 语言及其编程环境的介绍,C 语言编译器、调试工具和自动维护工具的使用方法,进程、文件的相关操作,输入输出操作和内存管理,C 语言网络编程方法等。

本书是作者根据多年来的开发和教学经验并融合大量的编程实例而著成此书。读者通过本书的学习能够快速学会 Linux 下 C 语言编程,掌握其中的方法和编程技巧,并能从一开始就养成良好的编程习惯,以便于读者低起点、高效率地掌握 Linux 环境下的编程知识。

本书结构合理、概念清晰、实例丰富,并具有很强的启发性和实用性,适合有一定的 C 语言基础,需要在 Linux 系统上编程的程序设计人员阅读,也可作为本、专科学生的教材或参考书,还可供广大计算机爱好者学习 C 语言使用。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13501256678 13801310933

图书在版编目(CIP)数据

Linux 环境下 C 编程指南/杨树青,王欢编著. —北京:清华大学出版社,2007.5

ISBN 978-7-302-15102-9

I. L… II. ①杨…②王… III. Linux 操作系统—程序设计②C 语言—程序设计 IV. TP316.89 TP312

中国版本图书馆 CIP 数据核字(2007)第 058180 号

责任编辑:邹 杰

装帧设计:Z2 工作室

责任校对:马素伟 周剑云

责任印制:李红英

出版发行:清华大学出版社 地 址:北京清华大学学研大厦 A 座

<http://www.tup.com.cn> 邮 编:100084

c-service@tup.tsinghua.edu.cn

社总机:010-62770175 邮购热线:010-62786544

投稿咨询:010-62772015 客户服务:010-62776969

印刷者:清华大学印刷厂

装订者:三河市新茂装订有限公司

经 销:全国新华书店

开 本:190×260 印 张:24.75 字 数:586 千字

版 次:2007 年 5 月第 1 版 印 次:2007 年 5 月第 1 次印刷

印 数:1~5000

定 价:42.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系
调换。联系电话:(010)62770177 转 3103 产品编号:026048-01

前 言

Linux 是由 UNIX 发展而来，最初由一个芬兰大学生开发维护，现在 Linux 已经成为最为流行的免费操作系统。Linux 的独特之处在于它的建立不受任何商品化软件的版权制约，全世界都能免费、自由地使用。世界各地有几十万计算机自愿者为这个充满魅力的操作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，一直在不断地改进和维护着这个系统。许多大学与研究机构、公司及家用 PC 机都在使用 Linux。Linux 年轻而富有朝气，它从诞生到现在不过 15 年时间，但已经确立了自己的地位并产生了广泛的影响。

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，成为倍受欢迎的编程语言。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，使该语言逐渐完美起来。作为 Linux 系统的开发语言，C 语言在 Linux 编程开发中扮演着重要的角色。

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，具有多平台性、移植性好的特点，因此它们很快形成了完美的结合，为用户提供了一个强大的编程环境。本书正是从这样的结合点出发，介绍在 Linux 系统中使用 C 语言编程的有关知识。国内 Linux 的发展方兴未艾，由于国内计算机教育体系等因素，Linux 的发展进入瓶颈状态，很多 Linux 爱好者仅仅停留在 Linux 系统管理的层次，而更多的 Linux 的系统管理员或者 Windows 的程序员要转向做 Linux 的开发却无从下手，除 Linux 开发界面本身不是很友好之外，更重要的是苦于找不到合适的学习参考资料入门。

在这种情况下，作者根据多年来的开发和教学经验并融合大量的编程实例而著成此书。读者通过本书的学习能够快速学会 Linux 下 C 语言编程，掌握其中的编程方法和技巧，并能从一开始就养成良好的编程习惯，从而实现 Linux 环境下的编程知识入门和提高。

本书主要针对具有一定的 C 语言编程基础，但未在 Linux 系统中使用过 C 语言的读者，着重讲解在 Linux 系统中使用 C 语言编程的方法。

全书共分 13 章。介绍的内容包括：Linux 平台下 C 语言及其编程环境的介绍，C 语言编译器、调试工具和自动维护工具的使用方法，进程、文件的相关操作，输入输出操作和内存管理，C 语言网络编程方法等。

本书深入浅出、通俗易懂，很适合初学者学习，条理性较强，不会让人感到迷失方向。内容的可读性和技术含量都很高，所讲的知识点清楚而且容易上手，但也不乏一些主题的深入和扩展。通过阅读本书，会让读者没有心理负担，不必担心自己才疏识浅，也不会遇到晦涩难懂的问题，能使读者可以尽快克服对 Linux 下编程的神秘感，真正进入 Linux 编程世界。

本书编写过程中得到了张文女士的关心和帮助，期间张辉先生也提出了很多宝贵的意见。另外，张涛、徐强、赵磊、周鸣、范丽娜、刘伟、陈艳华、谢振华、唐兵、张俊岭、尹建民等人也参加了本书部分内容的编写及素材整理工作，在此一并表示感谢。

由于作者水平有限，书中难免有不足和疏忽之处，恳请各位专家和广大读者批评指正。

目 录

第 1 章 C 语言基础和 Linux 系统概述..... 1

1.1 C 语言基础.....	2
1.1.1 C 语言概述.....	2
1.1.2 数据类型.....	2
1.1.3 运算符和表达式.....	9
1.1.4 C 程序语句.....	10
1.1.5 函数.....	15
1.1.6 编译预处理.....	16
1.2 Linux 系统概述.....	18
1.2.1 Linux 系统的发展历史.....	18
1.2.2 Linux 系统特点及主要功能.....	19
1.2.3 Linux 系统的主要产品.....	20
1.3 本章小结.....	20

第 2 章 vim 与 Emacs 编辑器..... 21

2.1 vim 简介.....	22
2.1.1 启动与退出 vim.....	22
2.1.2 命令行模式的操作.....	23
2.1.3 命令行模式切换到输入模式.....	25
2.1.4 最后一行模式的操作.....	26
2.2 Emacs 简介.....	26
2.2.1 Emacs 编辑器的运行和结束.....	26
2.2.2 基本操作.....	27
2.3 Emacs 的 C 模式.....	29
2.3.1 自动缩进.....	29
2.3.2 注释.....	29
2.3.3 预处理扩展.....	29
2.3.4 自动状态.....	30
2.3.5 使用 Emacs 进行编译和调试.....	30
2.4 本章小结.....	30

第 3 章 gcc 编译器..... 31

3.1 gcc 编译器简介.....	32
--------------------	----

3.1.1	Hello World 程序	32
3.1.2	gcc 选项概述	33
3.1.3	警告	35
3.1.4	优化 gcc	36
3.1.5	调试标记	40
3.1.6	使用高级 gcc 选项	42
3.2	gcc 编译流程简介	44
3.2.1	C 预处理器 cpp	44
3.2.2	GUN 连接器 ld	44
3.2.3	GUN 汇编器 as	45
3.2.4	文件处理器 ar	45
3.2.5	库显示 ldd	45
3.3	其他编译调试工具	45
3.4	本章小结	46

第 4 章 调试工具 gdb 47

4.1	gdb 符号调试器简介	48
4.2	gdb 功能详解及其应用	48
4.2.1	调试步骤	49
4.2.2	显示数据命令	57
4.2.3	使用断点	62
4.2.4	使用观察窗口	65
4.2.5	查看栈信息	68
4.2.6	查看源程序	70
4.2.7	查看运行时数据	73
4.2.8	改变程序的执行	81
4.2.9	core dump 分析	84
4.3	其他调试工具	90
4.4	本章小结	90

第 5 章 使用 make 91

5.1	makefile 文件简介	92
5.2	make 书写规则	94
5.2.1	规则举例	94
5.2.2	在规则中使用通配符	94
5.2.3	文件搜寻	95
5.2.4	伪目标	96
5.2.5	多目标	97

5.2.6	静态模式	97
5.2.7	自动生成依赖性	99
5.3	使用命令	100
5.3.1	显示命令	100
5.3.2	执行命令	101
5.3.3	命令出错	101
5.3.4	嵌套执行 make	102
5.3.5	定义命令包	103
5.4	使用变量	104
5.4.1	变量的基础	104
5.4.2	赋值变量	105
5.4.3	变量的高级用法	107
5.4.4	追加变量值	109
5.4.5	override 指示符	110
5.4.6	多行变量	110
5.4.7	环境变量	110
5.4.8	目标变量	111
5.4.9	模式变量	112
5.5	使用条件判断	112
5.5.1	示例	112
5.5.2	语法	113
5.6	使用函数	115
5.6.1	函数的调用语法	115
5.6.2	字符串处理函数	116
5.6.3	文件名操作函数	119
5.6.4	foreach 函数	120
5.6.5	if 函数	121
5.6.6	call 函数	122
5.6.7	origin 函数	122
5.6.8	shell 函数	123
5.6.9	控制 make 的函数	124
5.7	make 的运行	124
5.7.1	make 的退出码	124
5.7.2	指定 makefile 文件	124
5.7.3	指定目标	125
5.7.4	检查规则	126
5.7.5	make 的参数	127
5.8	隐含规则	130

5.8.1	使用隐含规则	131
5.8.2	隐含规则一览	131
5.8.3	隐含规则使用的变量	133
5.8.4	隐含规则链	135
5.8.5	定义模式规则	136
5.8.6	隐含规则搜索算法	139
5.9	使用 make 更新函数库文件	140
5.9.1	函数库文件的成员	140
5.9.2	函数库成员的隐含规则	141
5.9.3	函数库文件的后缀规则	141
5.9.4	注意事项	142
5.10	高级使用	142
5.10.1	宏的使用	142
5.10.2	内部规则	147
5.10.3	make 递归	150
5.10.4	依赖性的计算	150
5.11	库的使用	153
5.11.1	创建库和维护库	154
5.11.2	库的链接	155
5.12	本章小结	156

第 6 章 进程控制 157

6.1	进程的基本概念	158
6.1.1	进程基本介绍	158
6.1.2	进程的属性	159
6.2	进程控制的相关函数	160
6.2.1	进程的创建	160
6.2.2	进程等待	164
6.2.3	进程的终止	168
6.2.4	进程 ID 和进程组 ID	171
6.2.5	system 函数	176
6.3	多个进程间的关系	177
6.3.1	进程组	177
6.3.2	时间片的分配	178
6.3.3	进程的同步	180
6.4	线程	180
6.4.1	线程的创建	180
6.4.2	线程属性的设置	181

6.4.3	结束线程	182
6.4.4	线程的挂起	182
6.4.5	取消线程	183
6.4.6	互斥	184
6.5	本章小结	184
第 7 章 文件操作		187
7.1	文件系统简介	188
7.1.1	文件	188
7.1.2	文件的相关信息	190
7.1.3	文件系统	191
7.2	基于文件描述符的 I/O 操作	191
7.2.1	文件的创建、打开与关闭	192
7.2.2	文件的读写操作	194
7.2.3	文件的定位	198
7.3	文件的其他操作	200
7.3.1	文件属性的修改	200
7.3.2	文件的其他操作	203
7.4	特殊文件的操作	206
7.4.1	目录文件的操作	206
7.4.2	链接文件的操作	207
7.4.3	管道文件的操作	210
7.4.4	设备文件	210
7.5	本章小结	211
第 8 章 输入输出——基于流的操作		213
8.1	流简介	214
8.2	基于流的 I/O 操作	215
8.2.1	流的打开和关闭	216
8.2.2	缓冲区的操作	217
8.2.3	直接输入输出	219
8.2.4	格式化输入输出	221
8.2.5	基于字符和行的输入输出	224
8.3	临时文件	227
8.4	本章小结	231
第 9 章 内存管理		233
9.1	静态内存与动态内存	234

9.1.1	静态内存	234
9.1.2	动态内存	236
9.2	安全性问题	236
9.3	内存管理操作	237
9.3.1	动态内存的分配	237
9.3.2	动态内存的释放	238
9.3.3	调整动态内存的大小	239
9.3.4	分配堆栈	240
9.3.5	内存锁定	241
9.4	使用链表	241
9.5	内存映像 I/O	244
9.5.1	创建内存映像文件	245
9.5.2	撤销内存映像文件	245
9.5.3	将内存映像写入外存	246
9.5.4	改变内存映像文件的属性	248
9.6	本章小结	249

第 10 章 信号及信号处理 251

10.1	信号及其使用简介	252
10.1.1	信号简介	252
10.1.2	信号的使用	254
10.2	信号操作的相关系统调用	255
10.2.1	信号处理	255
10.2.2	信号的阻塞	263
10.2.3	发送信号	268
10.3	信号处理的潜在危险	276
10.4	本章小结	277

第 11 章 进程间通信 279

11.1	进程间通信简介	280
11.2	共享内存和信号量	280
11.2.1	SYSV 子系统的基本概念	280
11.2.2	共享内存	282
11.2.3	信号量	288
11.3	管道通信	298
11.3.1	管道的创建和关闭	299
11.3.2	管道的读写操作	300
11.4	命名管道	301

11.4.1	命名管道的创建	302
11.4.2	命名管道的使用	302
11.5	消息队列	306
11.5.1	消息队列的创建与打开	308
11.5.2	向消息队列中发送消息	308
11.5.3	从消息队列中接收消息	308
11.5.4	消息队列的控制	309
11.6	本章小结	311

第 12 章 网络编程 313

12.1	网络编程基本原理	314
12.1.1	计算机网络体系结构模式	314
12.1.2	TCP/IP 协议	315
12.1.3	客户机/服务器模式	316
12.1.4	套接口编程基础	320
12.1.5	IP 地址转换	332
12.2	TCP 套接口编程	336
12.2.1	基于 TCP 的客户机/服务器模式	337
12.2.2	信号处理	343
12.2.3	高级技术	345
12.3	UDP 套接口编程	354
12.3.1	基于 UDP 的客户机/服务器模式	354
12.3.2	主要系统调用函数	355
12.3.3	基于 UDP 套接口编程实例	355
12.3.4	可靠性问题	358
12.3.5	UDP 套接口的连接	360
12.4	原始套接口编程	361
12.4.1	基本形式和操作	361
12.4.2	原始套接口编程实例	363
12.5	网络编程实例	368
12.6	本章小结	372

第 13 章 底层终端编程 373

13.1	底层终端编程	374
13.1.1	属性控制	374
13.1.2	使用 terminfo	377
13.2	伪终端	379
13.3	本章小结	380



本章重点

- ◆ C 语言基础
- ◆ Linux 系统概述

第

1

章

C 语言基础和 Linux 系统概述

本章将简单介绍 C 语言编程基础和 Linux 操作系统的基本知识，以帮助读者对所学习的知识建立起清晰的认识，为后续学习打下坚实的基础。



1.1 C 语言基础

1.1.1 C 语言概述

C 语言是国际上广泛使用的，且很有发展前途的计算机高级语言，时下流行的 C++ 语言和 C#(用于网络编程)都是从 C 语言发展而来的。C 语言适合用来进行系统描述，既可用于编写系统软件，也可用来编写应用软件。C 语言是一种与 UNIX 密切相关的程序设计语言，它最初用于 DECPDP-11 计算机 UNIX。20 世纪 70 年代以来，操作系统中的大部分内容和应用程序都是用 C 语言编写的。C 语言之所以能长期存在和发展，并具有强大的生命力，与它的以下优点是分不开的。

- ◆ 语言简洁、紧凑(32 个关键字)，使用方便、自由(书写形式自由)，与 Pascal 和 Basic 语言比较起来，C 语言程序显得非常简练。
- ◆ 运算符丰富，共有 34 种，C 语言把括号、赋值、强制类型转换等都作为运算符处理。表达式类型多样化，灵活使用各种运算符可以在其他高级语言上难以实现的运算。
- ◆ 数据结构合理，具有现代语言的丰富数据结构，能用来实现各种复杂的数据结构(如链表、树、栈等)的运算。
- ◆ 具有结构化的控制语句，是结构化的理想语言，符合现代编程风格。
- ◆ 语法限制不太严格，程序设计自由度较大。
- ◆ 允许位操作和对硬件进行编程。
- ◆ 生成目标代码质量高，程序执行效率高。
- ◆ 程序可移植性较好。

1.1.2 数据类型

一个程序应包括数据的描述和动作的描述两方面内容。著名计算机科学家沃思曾提出一个公式：数据结构+算法=程序，可见数据结构在程序中的地位。C 语言为用户提供了丰富的数据结构，还允许用户自定义复杂的数据结构。

C 语言提供的数据结构是以数据类型形式出现的，C 的数据类型划分如图 1-1 所示。

1. 常量与变量

C 语言中数据有常量与变量之分。在程序运行过程中，其值不能改变的量称为常量，其值可以改变的量称为变量。常量可分为不同类型，如整型常量 3，实型常量-1.23，字符型常量‘a’。常量一般能从字面形式判别，也可用一个标识符代表，可用下面形式声明。

```
#define 常量名 常量值
```



例如:

```
#define PI 3.14          /*定义PI代表常量3.14*/
```

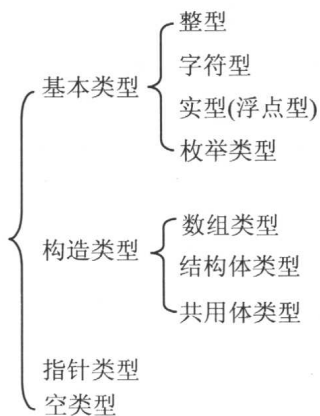


图 1-1 C 语言数据类型划分



注意

“/*”和“*/”之间表示这行符号中为注释内容，不会被编译。

变量可用下面的形式声明。

```
数据类型 变量名
```

如:

```
int i;          /*定义 i 为整型变量*/
char name;     /*定义 name 为字符型变量*/
```

在 C 语言中，习惯上用大写字符代表常量，用小写字符代表变量。对于变量，要求“先定义，后使用”。以下分别介绍整型、实型等数据类型。

2. 整型数据

(1) 整型常量

整型变量可分为 int、short int、long int 和 unsigned int、unsigned short、unsigned long 等类型，它们表示的数的范围不同。

(2) 整型变量

整型变量可分为基本型、短整型、长整型和无符号型 4 种，下面给出一个实例。

【代码 1-1】

```
main()
{
```



```

int a,b,c;
unsigned d;
a=6;b=-7;u=78;
c=a+b+u;
printf("%d\n",c);/*printf()为格式化输出函数*/
/* "%d"表输出格式为十进制整数*/
}

```

运行结果为:

```
77
```

printf()是标准输出函数，在后面我们将要详细说明。

3. 实型数据

(1) 实型常量

实型常量有两种表示方式，一种为十进制形式，如 0.12、36.2；另一种为指数形式，如 123e3 表示 123000。要注意 e 字母之前必须要有数字，e 字母后面必须为整数。

(2) 实型变量

实型变量分为单精度(float)和双精度(double)两种，如：

```

float x ; /*指定 x 为单精度实数*/
double y ; /*指定 y 为双精度实数*/

```

单精度实数在内存中占 4 个字节，而双精度实数在内存中占 8 个字节。

4. 字符型

(1) 字符常量

C 的字符常量是用单引号括起来的一个字符，如 ‘a’、‘A’ 等，注意以上两个字符是不同的。另外，C 语言还允许以一个 “\” 开头的特殊字符常量，具体规定如表 1-1 所示。

表 1-1 特殊字符常量表

字符类型	功 能
\n	换行
\t	横向跳格
\v	竖向跳格
\b	退格
\r	按 Enter 键
\f	走纸换页

【代码 1-2】

```

main()
{

```




```
printf(" ab c\t de\rf\tg\n");  
printf("h\ti\b\bjd");  
}
```

读者可以自己思考一下程序的输出结果。

(2) 字符变量

用来存放字符常量，且只能放一个字符，定义规则如下：

```
char d; /*表示d为字符型变量*/
```

(3) 字符数据在内存中的存储形式和使用方式

字符变量在对应的内存中存放的是该字符相应的 ASCII 码，例如字符 ‘a’ 的 ASCII 代码为 97，‘b’ 的 ASCII 代码为 98，而字符型与整型变量还可以互相赋值。

【代码 1-3】

```
main()  
{  
    char c1,c2;  
    c1=97,c2=98;  
    printf("%c %c",c1,c2);/* "%c"表示输出格式为字符类型*/  
}
```

输出结果是：

```
a b
```

(4) 字符串常量

字符串常量与字符常量的区别在于前者是双引号括起来的字符序列，而后者是单引号括起来的单个字符。如在内存中 ‘a’ 的长度是一个字符，而 “a” 则占有两个字符，后者包括结束标志 “\0”。

5. 枚举型

所谓“枚举”，是指将变量的值一一列举出来，变量的值只限于列举出来的值，可用如下定义形式：

```
enum weekday {sun,mon,tue,wed,thu,fri,sat};
```

它定义了一个名为 weekday 的枚举类型，能取 7 个值。

然后，我们可以用此类型来定义变量，如：

```
enum weekday date;
```

则 date 被定义成枚举变量，它能取 sun 到 sat 这 7 个值之一，如：

```
date=sun;
```

也可直接定义枚举变量，如：