

150

7P312C
H436

UNIX C 程序设计 入门与应用

洪锦魁 编著



A1004336

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书主要介绍 C 语言入门与应用的基础知识,适用于 UNIX 系统。本书共分 3 篇,分别是 C 语言程序设计基础知识、C 语言数据结构和 UNIX 系统程序设计。第 I 篇介绍 C 语言的基础知识,包括基本概念、基本的输入与输出、程序的流程控制、数组和指针、函数变量等;第 II 篇介绍 C 语言数据结构,包括基本表结构、堆栈与队列、二叉树等;第 III 篇介绍 UNIX 系统的文件管理、程序管理。

本书内容编排合理、由浅入深,程序示例丰富,非常适合对 C 语言感兴趣的初、中级读者使用。

本书中文繁体字版名为《UNIX C 语言入门与应用彻底剖析》,由文魁资讯股份有限公司出版,版权属洪锦魁所有。本书中文简体字版由文魁资讯股份有限公司授权清华大学出版社独家出版。未经本书原版出版者和本书出版者书面许可,任何单位和个人均不得以任何形式或任何手段复制或传播本书的部分或全部内容。

北京市版权局著作权合同登记号:图字 01-2002-1741 号

版权所有,翻印必究。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

书 名: UNIX C 程序设计入门与应用

作 者: 洪锦魁 编著

责任编辑: 杨志娟

出 版 者: 清华大学出版社(北京清华大学学研大厦,邮编 100084)

<http://www.tup.tsinghua.edu.cn>

印 刷 者: 北京市清华园胶印厂

发 行 者: 新华书店总店北京发行所

开 本: 787×1092 1/16 印张: 20.25 字数: 462 千字

版 次: 2002 年 8 月第 1 版 2002 年 8 月第 1 次印刷

书 号: ISBN 7-302-05713-3/TP·3370

印 数: 0001~4000

定 价: 28.00 元

前 言

随着计算机科学的发展，C 语言的普及和应用面越来越广，要求学习和掌握 C 语言的人越来越多。因为 C 语言同时具备低级汇编语言的功能和高级语言的亲和性，自然也就成了计算机语言的主流。

本书主要介绍如何在 UNIX 系统上使用 C 语言。为了方便读者学习，本书用了将近 220 个程序范例，一步一步对读者进行引导。相信读者只要花时间去学习这些程序，必定可在短时间内精通 C 语言编程。

全书分为 3 篇，共 21 章。第 I 篇为 C 语言程序设计基础，主要介绍 C 语言和一般 C 语言函数的用法；第 II 篇为 C 语言数据结构，通过程序范例说明基本数据结构的应用；第 III 篇为 UNIX 系统程序设计，列出了基本 UNIX 系统调用，同时用程序范例说明如何使用这些系统调用来设计有用的系统程序。

第 I 篇共 13 章，具体内容如下：

第 1 章简要介绍 C 语言的发展和 C 语言程序的开发过程。

第 2 章介绍使用 C 语言处理数据的基本概念，包括变量、基本的数据类型、常量的表达方式等。

第 3 章讲述基本的输入与输出，包括 `printf()`、`scanf()`、`putchar()` 和 `getchar()` 等函数。

第 4 章介绍 C 语言程序的流程控制，包括程序流程使用的关系运算符和流程控制指令。

第 5 章介绍数组的指针，包括一维和二维数组的使用方式、指针的基本概念，以及数组与指针配合使用的基本常识。

第 6 章介绍函数的应用，包括函数的声明、调用、使用以及程序间参数的传递方式。

第 7 章介绍变量的等级，包括变量的类别和应用。

第 8 章介绍 C 语言的预处理程序，包括预处理程序的定义和用法。

第 9 章介绍自定义数据类型，帮助读者建立自定义数据类型的概念，并通过实例讲解它的使用方法。

第 10 章介绍字符串，包括字符数组和字符串的差异、基本字符串的输入与输出、字符串函数的用法。

第 11 章介绍基本位运算，包括 C 语言的位运算指令及其用法。

第 12 章介绍测试符号与符号转换函数，包括这些函数的用法与应用。

第 13 章介绍文件的输入与输出，包括文件指针的基本知识，以及如何使用文件指针进行一般的文件输入与输出。

第 II 篇共 3 章，具体内容如下：

第 14 章介绍基本的表结构，包括表结构的基本知识，如何配置和释放存储器空间，以及如何将指针的结构数据合成表结构。

第 15 章介绍堆栈与队列，包括堆栈与队列的基本概念及其应用。

第 16 章介绍二叉树，包括树状数据结构的基本定义、二叉树的创建方法，以及打印二叉树的基本函数。

第III篇共 5 章，具体内容如下：

第 17 章介绍 UNIX 系统的文件管理，包括文件管理的基本知识、文件的基本操作。

第 18 章介绍程序的管理，包括进程的定义、建立及应用。

第 19 章介绍进程间的信息传递，包括进程通信的基本定义，以及进程间通信程序的设计。

第 20 章介绍屏幕控制，包括屏幕方位的定义，以及如何利用 `curses` 函数实现动态画面。

第 21 章是其他相关函数的总结，介绍了屏幕与文件之间的关系、时间函数的使用原则。由于我们水平有限，错误之处在所难免，恳请广大读者不吝指正。

目 录

第 I 篇 C 语言程序设计基础	
第 1 章 UNIX 系统 C 语言简介	3
1.1 C 语言简介	3
1.2 C 语言程序的开发过程	3
1.3 简单的 C 语言程序范例	5
第 2 章 C 语言数据处理的基本概念	9
2.1 变量名称的使用	9
2.2 变量的声明方式	10
2.3 基本的数据类型	10
2.3.1 整型数据	11
2.3.2 字符型数据	11
2.3.3 浮点数	12
2.3.4 双精度浮点数	12
2.3.5 字符串数据类型	13
2.4 常量的表达方式	13
2.4.1 整型常量	13
2.4.2 浮点常量	14
2.4.3 字符常量	14
2.4.4 字符串常量	14
2.5 基本的算术运算	15
2.6 不同类型的转换	17
2.7 C 语言的特殊运算	18
2.7.1 递增和递减运算	19
2.7.2 设置特殊运算符	20
2.7.3 sizeof	21
第 3 章 基本的输入与输出	23
3.1 printf() 函数	23
3.1.1 C 语言的控制符 \n	24
3.1.2 十进制整数的打印	25
3.1.3 浮点数的打印	27
3.1.4 字符的打印	28
3.1.5 其他格式化数据的打印原则	29
3.2 scanf() 函数	30
3.3 字符的输入和输出函数	32
第 4 章 程序的流程控制	37
4.1 关系运算符与逻辑运算符	37
4.2 for 语句	38
4.3 while 语句	43
4.4 do...while 语句	47
4.5 if 语句	48
4.6 else...if 语句	51
4.7 e1?e2:e3 语句	54
4.8 switch 语句	55
4.9 break 语句	58
4.10 continue 语句	59
4.11 goto 语句	60
第 5 章 数组和指针	65
5.1 一维数组	65
5.2 二维数组	67
5.3 指针的基本概念	70
5.4 指针和数组的混合应用	72
第 6 章 函数的应用	79
6.1 函数的定义	80
6.2 主程序和函数的位置	82
6.3 函数的返回值	83
6.4 数组数据的传递	86
6.5 指针的传递	88
6.6 递归式函数和新式 ANSI 函数	91
6.7 简单数学函数的使用	92

第 7 章 变量的等级.....97	第 11 章 基本位运算..... 157
7.1 auto 变量.....97	11.1 & 运算符..... 157
7.2 static 变量.....99	11.2 运算符..... 159
7.3 外部变量和静态外部变量.....102	11.3 ^ 运算符..... 160
7.4 register 变量.....105	11.4 ~ 运算符..... 161
第 8 章 C 语言的预处理程序.....107	11.5 << 运算符..... 163
8.1 宏指令#define.....107	11.6 >> 运算符..... 164
8.2 #include 指令.....111	11.7 位字段..... 165
8.3 条件式的编译.....113	第 12 章 测试符号与符号转换函数..... 167
第 9 章 自定义数据类型.....117	12.1 isalnum(c)函数..... 167
9.1 typedef 指令.....117	12.2 isalpha(c)函数..... 168
9.2 结构数据类型.....118	12.3 isascii(c)函数..... 168
9.3 结构数据与数组.....121	12.4 iscntrl(c)函数..... 169
9.4 结构数据与函数.....122	12.5 isdigit(c)函数..... 170
9.5 union 类型.....126	12.6 islower(c)函数..... 171
9.6 enum 类型.....128	12.7 isprint(c)函数..... 172
第 10 章 字符串.....131	12.8 isspace(c)函数..... 172
10.1 字符组成的一维数组.....131	12.9 tolower(c)函数..... 173
10.2 字符串的应用.....134	12.10 toupper(c)函数..... 173
10.2.1 构成字符串的基本要素.....134	第 13 章 文件的输入与输出..... 175
10.2.2 标准字符串的输出.....137	13.1 文件的打开和关闭..... 175
10.2.3 标准字符串的输入.....139	13.1.1 fopen()函数..... 175
10.3 字符串与指针.....140	13.1.2 fclose()函数..... 176
10.4 gets()和 puts()函数.....141	13.2 getc()和 putc()函数..... 176
10.4.1 gets()函数.....141	13.2.1 getc()函数..... 176
10.4.2 puts()函数.....144	13.2.2 putc()函数..... 177
10.5 系统字符串处理的函数.....145	13.3 fscanf()和 fprintf()函数..... 180
10.5.1 strcat().....146	13.4 fgets()和 fputs()函数..... 181
10.5.2 strcmp().....147	13.5 ungetc()函数..... 182
10.5.3 strcpy().....148	
10.5.4 strlen().....148	第 II 篇 C 语言数据结构
10.5.5 strncat().....149	第 14 章 基本的表结构..... 187
10.5.6 strncmp().....150	14.1 动态数据结构的声明..... 187
10.5.7 strncpy().....151	14.2 存储器的配置..... 187
10.6 字符串数组.....151	14.3 链表的基本操作..... 190
10.7 字符数组的指针.....153	14.4 表的连接..... 193
10.8 命令行参数.....154	

14.5	表中节点的删除	197
14.6	释放存储器空间	200
14.7	节点的插入	203
14.8	双向链表	208
第 15 章	堆栈与队列	211
15.1	堆栈	211
15.2	队列	215
第 16 章	二叉树	221
16.1	二叉树的节点结构	221
16.2	二叉树的建立	222
16.3	二叉树的输出	223
16.3.1	中序输出方式	224
16.3.2	前序输出方式	226
16.3.3	后序输出方式	229

第Ⅲ篇 UNIX 系统程序设计

第 17 章	UNIX 系统的文件管理	235
17.1	文件管理的基本知识	235
17.2	打开一个文件	237
17.3	新建一个文件	238
17.4	关闭一个文件	239
17.5	文件的读取	240
17.6	文件的写入	241
17.7	随机访问文件	243
17.8	管理文件	245
第 18 章	程序的管理	251
18.1	fork()指令	251
18.2	利用 exec 执行新的进程	252

18.2.1	execl 指令	252
18.2.2	execv 指令	253
18.2.3	execlp 指令	254
18.2.4	execvp 指令	255
18.3	暂时等待功能	257
18.4	获得程序 pid	260
第 19 章	进程间的信息传递	267
19.1	sleep 的使用	267
19.2	使用 SIGNAL 函数	268
19.3	使用 kill 传递信号	271
19.4	暂时中止函数 pause	271
19.5	闹钟函数 alarm	272
19.6	setjmp 和 longjmp 函数	274
19.7	使用 pipe 实现程序间通信	275
第 20 章	屏幕控制	279
20.1	程序的基本结构	279
20.2	字符的输出	281
20.3	字符串的输出	282
20.4	以格式化的方式输出数据	283
20.5	光标的移动	284
20.6	屏幕数据的删除	285
20.7	数据的输入	287
20.8	多窗口的应用	288
第 21 章	其他函数总结	303
21.1	终端特性	303
21.2	时间的处理	305
附录 A	ASCII 字符集	309
附录 B	简易 Makefile 使用原则	311

第 I 篇 C 语言程序设计基础

学习 C 语言的首要任务是了解 C 语言数据处理的基本概念。因此本书特将这一部分安排在第 1 章和第 2 章中来说明。

至于 C 语言的其他概念，则分门别类在本篇的其他章节内进行叙述。对于 C 语言而言，另一个很重要的概念是函数的应用，本篇也将一一介绍，以便读者在阅读完本书之后，能对 C 语言有一个综合性的了解。

学习 C 语言最终的目标是能以有效的方式充分地利用系统资源，以实现软件取代硬件的目标。所以，读完本篇后，应该继续阅读第 II 篇 C 语言的数据结构，以了解应该如何有效地设计 C 语言程序。然后学习第 III 篇 UNIX 的系统资源，以了解应该如何利用 UNIX 的系统资源。

本篇内容：

- 第 1 章：UNIX 系统 C 语言简介
- 第 2 章：C 语言数据处理的基本概念
- 第 3 章：基本的输入与输出
- 第 4 章：程序的流程控制
- 第 5 章：数组和指针
- 第 6 章：函数的应用
- 第 7 章：变量的等级
- 第 8 章：C 语言的预处理程序
- 第 9 章：自定义数据类型
- 第 10 章：字符串
- 第 11 章：基本位运算
- 第 12 章：测试符号与符号转换函数
- 第 13 章：文件的输入与输出

第 1 章 UNIX 系统 C 语言简介

由于 C 语言同时具备低级汇编语言的功能和高级语言的亲和性，自然就成为未来计算机语言的主流。

本章将对 C 语言的发展做一个简单的介绍。同时，用程序示例说明 C 语言的基本结构、编译以及执行方式。至于 C 语言的其他细节，在以后的章节中将分别介绍。

本章学习目标

- 了解 C 语言的发展
- 熟悉 C 语言的基本构架、编译以及程序执行方法

1.1 C 语言简介

C 语言是 1972 年 Rennis Ritchie 和 Ken Thompson 两人在一起设计 UNIX 操作系统时开发出来的，由于 C 的起源就是为了要设计 UNIX 操作系统，如果只叙述 C 语言本身，而不涉及系统设计，那么读者将无法深入了解 C 语言的实际好处。因此，在本书的第Ⅲ篇，会详细地说明 UNIX 系统程序设计。

由于 UNIX 已经成为微机或者中型机的主要操作系统，C 语言也就成为这些计算机的主要程序语言。另外，IBM 公司的大型机原本没有 C 编译程序，由于大势所趋也不得不安装这个功能强大的编译程序。在个人计算机中，Turbo C 和 Microsoft C 也成为了学生以及小软件公司的主要程序语言。

1.2 C 语言程序的开发过程

用 C 语言来编写程序，从设计到最后的执行，一般按照下列步骤进行：

- (1) 程序规划。
- (2) 利用编辑程序编写源程序。
- (3) 编译和连接程序，此时系统将产生可执行模块。
- (4) 执行此程序。

图 1.1 是上述步骤的流程说明。

1. 规划程序

初学计算机的人常会对程序操作感到头疼。一般而言，要克服困难只有一个方法，那就是多做练习。

惟有多练习才可精益求精。当然，在规划程序过程中，也有简单的基本原则可循。那就是按照程序操作，首先检查有哪些输入的数据，输出的要求是什么。可以将程序操作分成几个区段，然后再一个一个组合起来。图 1.2 是程序操作的流程图。

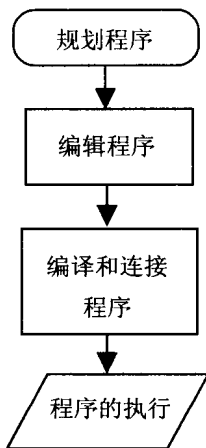


图1.1 C语言程序的流程图

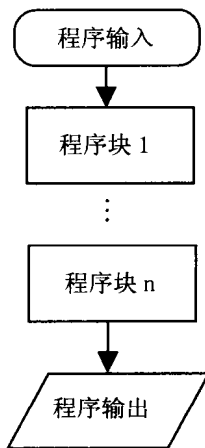


图1.2 规划过程流程图

2. 程序的编辑

在 UNIX 系统中，不像用于个人计算机的 Turbo C 那样有属于自己的全屏幕编辑程序。不过 UNIX 系统上可用的编辑程序有很多，如 `ed`、`ex`、`edit` 和 `vi`。本书中的程序大多是利用 `vi` 全屏幕编辑程序完成的。

C 语言程序的文件名称，一般有两部分组成：一个是主文件名，另一个是扩展文件名。其中主文件名可自行命名，扩展文件名是 `.c`。例如，有一个程序主文件名是 `chl`，则这个程序的全名为 `chl.c`。

注意： 主文件名和扩展文件名使用点分隔。

一般程序设计，大多是选用能够表示计算机功能的名字来做文件名称。例如，如果设计一个时钟程序，可用 `clock.c`。

如果设计一个排序程序，则用 `sort.c`。

`ch` 是 `chapter`(章节)的缩写，为了使本书所有程序保持一致性，本书中所有程序都以 `ch` 开头。

3. 编译和连接程序

UNIX 系统内，C 语言的编译和连接程序称为 `cc`，要编译时只要输入下列格式即可：

```
cc 文件名
```

假如现在要编译的程序是 `chl.c`，则需要输入下列指令：

```
cc chl.c
```

注意： 在 UNIX 系统中，`cc` 不仅具有编译的功能，同时也有连接的功能。因此，执

行完 cc 之后，只要程序没有错，即可执行。

4. 程序的执行

在上一节中，程序被编译和连接完之后，会产生一个可执行文件，称为 a.out。要执行此文件，只要输入下列指令即可：

```
a.out
```

如果不特别设置，所有的 C 语言程序，编译完后的可执行文件都为 a.out。当然也可以更改这个可执行文件，下一节将用程序示例来说明这个概念。

1.3 简单的 C 语言程序范例

本节将介绍基本 C 程序的结构，让读者了解 C 语言结构后，再继续说明各种 C 程序的语法。这样，就可以循序渐进地进入 C 语言世界了。

程序范例ch1.c

一个简单的输出程序，其代码如下。

```
01 /*===== program description===== */
02 /* program name:ch1.c */
03 /* A first program in c */
04 /*===== */
05
06 main ()
07 {
08     int i;
09
10     i=1;
11     printf("An Introduction in UNIX C\n");
12     printf("program exercise %\n",i);
13     printf("C is fantastic \n");
14 }
```

要执行上面的程序，可以先输入下列指令，进行编译和连接。

```
cc ch1.c
```

当完成编译和连接之后，系统会自动产生一个文件，文件名为 a.out。要执行此程序，只要输入下列指令就可以了。

```
a.out
```

下面是本程序的执行结果：

```
An Introduction in UNIX C
Program exercise 1
C is fantastic
```

1. 程序的行号

在程序范例 `chl.c` 中，在程序的左边有程序的行号。其实 C 语言程序是没有行号的，在此之所以加行号，主要是为了阅读方便。所以在输入程序的时候，可自行将行号删除。

2. 程序的注释

程序范例中，第 1~4 行是注释，如下所示：

```
01 /*===== program description =====*/
02 /*   program name:chl.c                      */
03 /*A first program in c                      */
04 /* =====*/
```

凡是介于 `/*` 和 `*/` 之间的文字，C 的编译程序均会忽略，不予编译。设计程序时，最好养成写注释的习惯，便于以后阅读。

3. 主程序 `main()` 声明

所有的 C 语言程序均由 `main()` 开头。C 语言程序会执行 `{` 和 `}` 之间的内容。在此例子中，此程序会执行第 7~14 行之间的内容。

C 语言的基本结构是：

```
main( )
{
}
```

4. 程序的内容

在程序范例中，第 8~13 行都属于程序内容，如下所示：

```
08   int i;
09
10   i=1;
11   printf("An Introduction in UNIX C\n");
12   printf("Program exercise %d\n",i);
13   printf("C is fantastic  \n");
```

 **注意：** 在每一个完整的语句后面，一定要加上分号(;)，表示每一个语句的结束。

5. 变量的声明

在 C 语言中，所有的变量在使用前，一定要加以声明，以便 C 编译程序为这个变量安排一个存储空间。以后使用此变量时，C 编译程序就会自行到这个存储空间存取数据。

范例：在程序代码的第 8 行中，声明了 `[int i;]` 以后，整个存储器如图 1.3 所示。

范例：在程序代码的第 10 行中定义 `[i=1;]` 时，整个存储器将如图 1.4 所示。

至于其他更详细的变量声明的原则，下一章再做讨论。

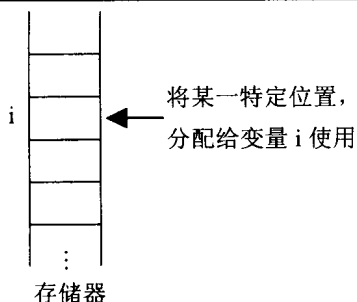


图1.3 变量声明与内存空间

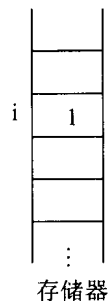


图1.4 变量使用与内存空间

6. 程序范例ch1.c的解释说明

在程序范例中，预留了很多空白行，例如：第5行及第9行。这样是为了使程序看起来不拥挤。C编译程序在看到这些行时，会将其忽略。

以下是整个程序的说明：

- 第1~4行，是程序的注释。
- 第6行说明这是主程序。
- 第7行“{”（左大括号），表示主程序从这里开始。
- 第8行是把变量*i*声明为整数(int)。
- 第10行是将变量*i*的值设置为1。
- 第11行是打印下列字符串：

```
An Introduction in UNIX c
```

- 第12行是打印下列字符串：

```
Program Exercise 1
```

- 第13行是打印下列字符串：

```
C is fantastic
```

- 第14行“}”（右大括号），在此表示主程序结束。

有关程序输入与输出的原则，将留待第3章再进行说明。

7. 附注

如果有下列指令编译程序：

```
cc -o outchl ch1.c
```

↑ 可以自行设置任何可执行文件名

则编译完本程序之后，程序的可执行文件将被存放在 `outchl` 内，此时可以直接输入 `outchl` 执行此文件。

第 2 章 C 语言数据处理的基本概念

程序语言最基本的数据处理对象就是变量和常量，本章将对所有的变量和常量进行解释说明。另外，C 语言程序拥有的很多不同于其他高级语言的运算符，本章也将一一说明。

本章学习目标

- 了解变量的使用和声明方式
- 了解 C 语言的各种数据类型和基本运算
- 了解 C 语言的一些特殊运算

2.1 变量名称的使用

C 语言对变量名称的使用是有一些限制的，变量名必须以 3 种字符作为开头即大写字母、小写字母和下划线(_)；至于变量名称的组成则是由 4 种字符所构成：大写字母、小写字母、下划线(_)和阿拉伯数字 0~9。

范例：下列均是合法的变量名称：

```
SUM
hung
sum_1
_fg
x5
y6l
```

范例：下列均是不合法的变量名称：

```
sum,1    ← 变量名称不可有逗号(,)
3y       ← 变量名称不可由阿拉伯数字开头
x$2      ← 变量名称不可含有$符号
```

另外，需要注意的是，在 C 语言中大写字母和小写字母代表不同的变量。

范例：下列 3 个字串，分别代表 3 个不同的变量。

```
sum
Sum
SUM
```

有关变量使用另一点需要注意的是，有些字为系统保留字(也称为关键字)，在 C 语言

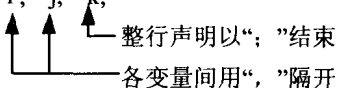
编译程序中代表特别的含义，所以不能作为变量名称。以下是 C 语言的保留字。

auto	break	case	char	continue	default
do	double	else	enum	extern	float
for	goto	if	int	long	register
return	short	sizeof	static	struct	switch
typedef	union	unsigned	void	while	

2.2 变量的声明方式

已经讲过，任何变量在使用前一定要先加以声明。

范例：如果想将 i、j、k 声明为整数，则下列的声明方式均是合法的。

1. int i, j, k;


2. int i,
 j,k;

由于 i 和 j 之间仍是用 “,” 隔开，所以这是合法的声明方式。

3.
 int i;
 int j;
 int k;

由于每一次声明都是由 “;” 结束的，所以这仍是合法的声明。

另外，也可以在声明变量的同时，设置变量的值。

范例：将 i 声明成整数，并将其设置成 7。

```
int       i = 7
```

2.3 基本的数据类型

C 语言的基本数据类型有以下 4 种：

- int: 整型数据
- char: 字符型数据
- float: 单精度浮点数