

原创经典，程序员典藏

全面解读Linux C程序设计的开发环境、开发工具和典型应用
清晰把握Linux环境编程的精髓，彻底攻克开发的重点和难点

Linux C 程序设计大全

吴岳 等编著

- ◎ 对C语言的本质进行剖析，讲解实际开发中可能遇到的陷阱
- ◎ 详细讲解Linux进程操作，并分析进程环境对进程的影响
- ◎ 通过实际案例说明Linux线程的特性、属性和应用
- ◎ 深入讲解Linux文件系统，并且以实验的方式验证结论
- ◎ 用实用的案例剖析Linux环境下网络应用程序的开发流程
- ◎ 提供643个示例、2个综合实例、412个技巧，可作为案头必备的查询手册

清华大学出版社



原创经典，程序员典藏



全面解读Linux C程序设计的开发环境、开发工具和典型应用
清晰把握Linux环境编程的精髓，彻底攻克开发的重点和难点

Linux C 程序设计大全

吴岳 等编著



清华大学出版社



北 京

内 容 简 介

Linux 是一个开放源代码的操作系统,其稳定性与低廉的价格使其在服务器、嵌入式领域以及桌面应用中逐渐占有越来越大的市场份额。因此,对 Linux 环境下的程序开发人员的需求也就越来越大了。C 语言是 Linux 操作系统中的核心语言,掌握 Linux 环境下的 C 语言开发是至关重要的。

本书共分为 6 篇,循序渐进地讲述了 Linux 环境下的 C 语言编程技术,从基本概念到具体实践、从系统函数接口的认识到具体操作都进行了详细的阐述,并对具体知识点进行了详细的实例讲解。

本书的特点是详细介绍了 Linux 的系统接口函数、Linux 的编程环境以及 C 语言程序开发的辅助技术。本书通过大量实例,详细描述了 Linux 系统提供的系统函数接口,以及代码编写技巧,以方便读者实践。本书适合想全面学习 Linux 环境下 C 语言编程的读者,并可作为开发人员的参考手册。

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 13701121933

图书在版编目(CIP)数据

Linux C 程序设计大全 / 吴岳等编著. —北京:清华大学出版社, 2009.2
ISBN 978-7-302-19211-4

I. L… II. 吴… III. ①Linux 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(2009)第 000182 号

责任编辑:冯志强 赖 晓

责任校对:徐俊伟

责任印制:杨 艳

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

社 总 机:010-62770175

邮 购:010-62786544

投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈:010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者:清华大学印刷厂

装 订 者:北京市密云县京文制本装订厂

经 销:全国新华书店

开 本:185×260 印 张:56 字 数:1397 千字

版 次:2009 年 2 月第 1 版 印 次:2009 年 2 月第 1 次印刷

印 数:1~5000

定 价:89.00 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:030201-01

前 言

随着计算机技术的高速发展，Linux 系统在服务器解决方案中的优势越来越被开发者认同。在计算机就业市场上，基于 Linux 系统开发的人员需求量也不断增加。其中，对 Linux 系统环境下 C 语言开发人员的需求尤为明显。由于 C 语言本身在语法结构、语言风格和编程思想上的特点，使 C 语言一直被公认为难以学习、轻松掌握。

尽管 Linux 环境下 C 语言开发越来越被企业和开发人员认同，关注的人数也越来越多，但实际上，真正了解和掌握 C 语言开发的人却很少。因此，笔者总结亲身学习 Linux 系统函数接口的经验，并结合自己多年从事 Linux 环境下 C 语言应用程序开发的经验，编写出这本能够真正让开发者掌握 Linux 环境下 C 语言编程技巧的书。在这本书中，笔者将通过 28 章的学习规划，让读者快速掌握 Linux 环境下 C 语言编程的基本知识和编程技巧。

本书特色

相比同类图书，本书具有以下明显特色。

1. 技术翔实，内容突出

本书从实际开发者的角度全面地介绍了 Linux 环境下 C 语言编程的基础知识。本书涉及了 Linux 系统函数的各个方面，选择了当前最热门、应用最广的技术进行了深入的讨论。

2. 概念准确，理解深刻

在本书中，对每个核心的概念都使用通俗形象的语言进行解释，对于很多关键概念，还配有相关插图，让读者更加直观地掌握概念的含义。同时，鉴于广大的开发者对 Windows 比较熟悉，本书在多处概念讲解中，将其和 Windows 系统开发中相关的概念进行对比讲解，来加深读者对 Linux 开发的理解。

3. 实例丰富，强调实践

为了让读者易于掌握 Linux 环境下 C 语言编程的技巧，本书列举了大量实例进行讲解，通过这些实例，读者可以更加深入地理解相关概念，从而达到灵活使用 Linux 系统函数接口编写程序的目的。另外，本书重点强调实践性，本书中的很多例子都来源于作者的实际开发，大多数实例都是一些实际项目中截取的一部分子功能。通过对这些例子的学习，可以增强读者的动手实践能力。

4. 代码规范, 注释详细

为了让读者了解 Linux 环境下开发的要求, 本书在讲解代码时, 十分注意代码规范。同时, 所有的代码都取自实际开发经验, 符合规范。为了帮助读者理解代码含义, 本书对代码进行了详细的注释, 读者可以通过注释十分便利地了解代码的结构和流程。本书所涉及的源代码可到清华大学出版社的网站上下载, 网址: www.tup.cm.cn。

本书内容

本书共分为 6 篇, 循序渐进地讲述了 Linux 环境下 C 语言的编程技术, 从基本概念到具体实践、从系统函数接口的认识到具体操作等全方位的知识。

第 1 篇(第 1~5 章)简单介绍了 Linux 的发展历史、发展方向以及 Linux 环境下的一些常用的开发语言。同时还介绍了 C 语言中的重点和难点。在本篇跨过 C 语言的基础部分, 对 C 语言的难点和在现实开发过程中容易出错的部分进行了深入讲解, 并且配以大量的实例。

第 2 篇(第 6~9 章)讲述了 Linux 操作系统中的 C 语言开发环境。在本篇中详细介绍了 Linux 环境下的开发工具, 同时为了使读者理解透彻, 还举了对应的实例, 供读者参考、模拟、实践。

第 3 篇(第 10~16 章)讲述了 Linux 环境下和进程有关的操作, 包括 Linux 操作系统中进程运行的环境、Linux 操作系统中对进程的控制、Linux 环境下进程之间的通信方法以及线程的相关操作。

第 4 篇(第 17~21 章)介绍了 Linux 环境下和文件有关的操作, 包括 Linux 操作系统中读写文件的 I/O、文件系统的结构与组织、Linux 中的特殊文件的使用以及 Linux 环境下基于流的 I/O。

第 5 篇(第 22~26 章)专门介绍 Linux 环境下的网络编程。通过对网络基础协议、Linux 网络的编程基础操作以及两个网络应用程序实例的讲解, 使读者理清 Linux 环境下网络的流程。

第 6 篇(第 27~28 章)介绍 Linux 环境下的 shell 脚本。通过对编写 shell 脚本基础知识的阐述, 使读者基本掌握这门和 C 语言配合的最好的脚本语言。

阅读本书建议

为了提高读者学习效率, 增强学习效果, 特别提出以下学习建议。

- ❑ 在本书中, 为了帮助读者理解概念, 多次使用了图示的方法来讲解概念。请读者认真查看这些示意图, 这样可以帮助读者更加深刻地理解概念。
- ❑ 本书的所有实例都有实际开发背景, 所以, 请读者亲自完成书中的实例, 这样才能身临其境地感受到实际项目对开发者的要求。
- ❑ 注重技术交流。Linux 本身就是开源的系统, 从诞生之日起, 就和技术交流密切相

关。根据笔者的经验，技术交流和网络资源对开发者而言，是至关重要的。希望读者一开始就能注意到技术交流的重要性。

本书读者对象

- ❑ Linux 环境下 C 语言编程的初学者。
- ❑ Linux 系统函数接口的研究人员。
- ❑ Linux 服务器程序的开发人员。
- ❑ 嵌入式 Linux 程序的开发人员。
- ❑ Linux 桌面应用的开发人员。
- ❑ 需要在 Linux 环境下进行毕业的计算机学员。
- ❑ 想了解 Linux 环境下 C 语言编程的其他人员。

编者

目 录

第 1 篇 Linux 下 C 语言基础

第 1 章	Linux 简介	2
1.1	GNU 简介	2
1.2	Linux 简介	2
1.2.1	Linux 发展史	2
1.2.2	Linux 发行版	4
1.2.3	Linux 内核版本	5
1.2.4	Linux 与 UNIX 的关系	5
1.2.5	Linux 在服务器方面的发展	5
1.2.6	Linux 在嵌入式系统方面的发展	6
1.2.7	Linux 在桌面系统方面的发展	7
1.3	Linux 环境下的其他编程语言	7
1.3.1	C++	7
1.3.2	Java	10
1.3.3	Perl	12
1.3.4	Python	13
1.3.5	Ruby	14
1.3.6	PHP	14
第 2 章	控制结构	17
2.1	goto 语句	17
2.1.1	C 语言中的无条件跳转	17
2.1.2	使用 goto 语句进行出错处理	18
2.1.3	出错处理的一般模型	20
2.2	C 语言中的分支结构	21
2.2.1	分支结构的翻译	22
2.2.2	使用 goto 语句实现分支结构	23
2.3	短路计算	24
2.3.1	短路计算	24
2.3.2	&&运算的短路计算	25
2.3.3	运算的短路计算	26

2.4	C 语言中的循环结构	28
2.4.1	while 循环	28
2.4.2	do...while 循环	30
2.4.3	for 循环	32
2.5	switch 语句	34
2.5.1	switch 语句的应用	34
2.5.2	使用 goto 语句实现 switch 语句	35
2.6	优化控制结构	37
2.6.1	表达式优化——使用替换程序中的乘除法	37
2.6.2	表达式优化——常量折叠	38
2.6.3	表达式优化——使用数学公式	38
2.6.4	表达式优化——存储问题	40
2.6.5	分支优化——改变判断顺序	40
2.6.6	分支优化——使用 switch 语句	44
2.6.7	循环优化——一次性计算	46
第 3 章	C 语言中的函数	49
3.1	函数的本质	49
3.2	变量的作用域和生命期	50
3.2.1	全局变量	50
3.2.2	局部变量	51
3.3	变量的初始值	53
3.3.1	全局变量的初始值	53
3.3.2	局部变量的初始值	54
3.4	与函数有关的优化	55
3.4.1	函数调用与程序优化	55
3.4.2	变量存储优化	58
3.5	编写多文件程序——变量的存储类别	60
3.5.1	存储类别	60
3.5.2	static 变量的作用——改变变量的生命期	60
3.5.3	static 变量的作用——实现封装和模块化设计	63
3.6	编写多文件的程序——链接的作用	66
3.6.1	链接多个文件	66
3.6.2	链接时符号解析规则	68
3.6.3	链接规则的应用	68
3.7	可变参数	72
3.7.1	可变参数的概念	72
3.7.2	实现一个简单的可变参数的函数	73
3.7.3	可变参数实例	75

3.7.4	关于 printf 函数的疑问——缺少整型参数	80
3.7.5	关于 printf 函数的疑问——缺少字符串地址参数	81
第 4 章	C 语言中的指针与字符串	84
4.1	sizeof 运算符	84
4.1.1	sizeof 运算符的应用——得到内置类型的大小	84
4.1.2	sizeof 运算符的应用——得到复合类型的大小	85
4.2	指针的应用	86
4.2.1	指针与别名陷阱	86
4.2.2	数组的指针	88
4.2.3	指针的指针	90
4.2.4	指针与参数传递	91
4.2.5	指针类型的意义	98
4.2.6	void* 型指针	100
4.3	函数的指针	103
4.3.1	C 语言中的函数指针	103
4.3.2	函数指针的应用——回调函数	106
4.3.3	函数指针数组	111
4.4	字符串	113
4.4.1	字符串与字符数组	113
4.4.2	字符串与指针	114
4.4.3	限定修饰符 const	116
4.4.4	const 关键字修饰指针——在指针定义之前	117
4.4.5	const 关键字修饰指针——在指针定义之中	117
4.4.6	const 关键字修饰指针——在指针定义之前和定义之中	118
4.4.7	使用 const 关键字的意义	119
第 5 章	C 语言的高级技术	122
5.1	结构体与共同体	122
5.1.1	结构体中成员变量的存储分布	122
5.1.2	内存对齐	123
5.2	位运算	126
5.2.1	掩码运算	127
5.2.2	不安全的位运算	129
5.2.3	异或运算的特性	130
5.2.4	移位运算的陷阱	133
5.2.5	移位运算的实例	134
5.3	预处理	136
5.3.1	常用的代码组织形式	136

5.3.2	调试开关	142
5.4	C99 新标准关键字详解	145
5.4.1	inline 关键字的概念	145
5.4.2	inline 关键字实例	145
5.4.3	inline 关键字使用总结	146
5.4.4	restrict 关键字的概念	147
5.4.5	restrict 关键字的应用	148

第 2 篇 C 语言开发环境

第 6 章	vi 与 vim 编辑器	152
6.1	vi 编辑器入门	152
6.1.1	vi 简介	152
6.1.2	vi 的工作模式	152
6.2	vi 一般操作	153
6.2.1	进入 vi	153
6.2.2	文本插入操作	156
6.2.3	文本删除操作	158
6.2.4	文本复制操作	159
6.2.5	撤销命令	161
6.2.6	重复命令	162
6.2.7	退出 vi	162
6.3	vi 的增强操作	164
6.3.1	替换命令	164
6.3.2	光标移动	165
6.3.3	按行移动光标	166
6.3.4	按字移动光标	167
6.3.5	按句移动光标	168
6.3.6	按段移动光标	169
6.3.7	文本行的移动	169
6.3.8	文本的异行移动	170
6.3.9	屏幕滚动	171
6.3.10	查找命令	171
6.3.11	替换命令	173
6.3.12	窗口的切分	175
6.3.13	设置环境	176

第 7 章	gcc 编译器	177
7.1	初探 gcc 编译器	177
7.1.1	从经典的 hello world 开始	177
7.1.2	gcc 编译流程	178
7.2	gcc 常用选项	179
7.2.1	gcc 常用选项汇总	179
7.2.2	-c 选项	179
7.2.3	-S 选项	180
7.2.4	-E 选项	182
7.2.5	-o 选项	182
7.2.6	-I 选项	183
7.2.7	-g 选项	183
7.3	链接原理	183
7.3.1	链接器的任务	183
7.3.2	目标文件	184
7.3.3	ELF 格式的可重定位目标文件	185
7.3.4	目标文件中的符号表	187
7.3.5	重定位的概念	191
7.3.6	符号的重定位信息	192
7.4	关于库	192
7.4.1	使用库的优势	192
7.4.2	静态库的概念	194
7.4.3	创建静态库	194
7.4.4	使用静态库	195
7.4.5	动态库的概念	197
7.4.6	创建动态库	197
7.4.7	使用动态库	199
7.5	gcc 工具链	201
第 8 章	makefile	203
8.1	makefile 文件入门	203
8.1.1	makefile 文件的组成内容	203
8.1.2	makefile 文件包含	203
8.1.3	make 工具的退出码	204
8.2	书写 makefile 规则	205
8.2.1	使用基本规则	205
8.2.2	使用隐式规则	206
8.2.3	使用伪目标	207
8.2.4	使用通配符	209

8.2.5	搜索源文件	209
8.3	使用命令	210
8.3.1	显示命令	211
8.3.2	执行命令	211
8.3.3	命令出错	212
8.4	使用变量	213
8.4.1	使用普通变量	213
8.4.2	变量中的变量	215
8.4.3	追加变量的值	217
8.4.4	自动化变量	218
8.5	使用条件判断	219
8.5.1	条件表达式	219
8.5.2	表达式实例	221
8.6	使用函数	221
8.6.1	函数调用的语法	222
8.6.2	字符串处理函数	222
8.6.3	文件名操作函数	229
8.6.4	foreach 函数	233
8.6.5	if 函数	233
8.6.6	call 函数	234
8.6.7	origin 函数	235
8.6.8	shell 函数	235
8.7	makefile 实例	236
8.7.1	makefile 实例——项目中的总 makefile	236
8.7.2	makefile 实例——makefile 模板	238
第 9 章	gdb	240
9.1	列出源程序	240
9.1.1	不带参数的 list 命令	240
9.1.2	带一个参数的 list 命令	241
9.1.3	带两个参数的 list 命令	242
9.2	运行程序的命令	243
9.3	操作断点的命令	244
9.3.1	设置断点	244
9.3.2	显示当前 gdb 的断点信息	247
9.3.3	删除指定的断点	247
9.3.4	禁止或启用断点	248
9.3.5	清除断点	248
9.3.6	观察点	249

9.3.7 设置断点实例	250
9.4 查看运行时数据	254
9.4.1 数据观察命令	254
9.4.2 对程序中函数的调用	256
9.4.3 查看表达式的值	256
9.4.4 查看数组的值	257
9.4.5 变量的输出格式	257
9.4.6 查看内存	258
9.4.7 自动显示变量	259
9.4.8 设置显示选项	263
9.4.9 显示变量的历史记录	265
9.4.10 查看寄存器	266
9.4.11 查看使用 gdb 环境变量	267
9.4.12 查看数据实例	268
9.5 改变程序的执行	273
9.5.1 修改变量的值	273
9.5.2 跳转执行	274
9.5.3 信号的产生及处理	275
9.5.4 强制调用函数	276
9.5.5 强制函数返回	276
9.6 gdb 高级应用	278
9.6.1 产生 core 文件	278
9.6.2 跟踪栈上数据	278
9.6.3 绑定运行进程	281
9.6.4 源文件搜索	283
9.6.5 机器语言工具	283
9.6.6 其他有用的调试命令	284

第 3 篇 Linux 进程操作

第 10 章 进程环境	288
10.1 程序的启动和退出	288
10.1.1 在 shell 中启动一个程序	288
10.1.2 加载一个程序	289
10.1.3 加载地址	290
10.1.4 退出程序	291
10.1.5 进程终止处理函数	292

10.2	Linux 进程内存管理	294
10.2.1	数据的内部存储	294
10.2.2	C 程序的存储布局——代码段	296
10.2.3	C 程序的存储布局——数据段和缓冲段	298
10.2.4	C 程序的存储布局——栈	299
10.2.5	C 程序的存储布局——堆	301
10.2.6	常量的存储	302
10.2.7	动态内存管理	302
10.2.8	动态内存分配的深入研究	305
10.3	shell 环境	309
10.3.1	命令行参数	310
10.3.2	得到程序文件名	311
10.3.3	环境变量	312
10.3.4	得到指定的环境变量	314
10.3.5	设置一个环境变量	315
10.3.6	得到进程结束状态	320
10.3.7	使用 <code>errno</code> 调试程序	322
10.3.8	输出错误原因	324
10.4	全局跳转	325
10.4.1	局部跳转的局限性	326
10.4.2	使用全局跳转	327
10.4.3	使用全局跳转代替局部跳转	329
第 11 章	进程控制	332
11.1	进程标识符	332
11.1.1	进程 ID	332
11.1.2	进程中重要的 ID 值	333
11.2	进程操作	335
11.2.1	创建一个进程	335
11.2.2	父子进程的共享资源	337
11.2.3	<code>fork</code> 函数的出错情况	340
11.2.4	创建一个共享空间的子进程	341
11.2.5	在函数内部调用 <code>vfork</code> 函数	342
11.2.6	退出进程	344
11.2.7	使用 <code>exit</code> 函数检查进程出错信息	345
11.2.8	<code>exit</code> 函数与内核函数的关系	346
11.2.9	设置进程所有者	346
11.2.10	调试多进程——设置跟踪流	348
11.2.11	调试多进程——使用 <code>gdb</code> 的 <code>attach</code> 命令	348

11.3	执行程序	349
11.3.1	执行一个新程序	350
11.3.2	执行解释器文件	352
11.3.3	在程序中执行 shell 命令	355
11.3.4	实现 system 函数	356
11.4	关系操作	358
11.4.1	等待进程退出	358
11.4.2	等待指定的进程	362
11.4.3	僵尸进程的概念	364
11.4.4	产生一个僵尸进程	364
11.4.5	避免僵尸进程的产生	366
11.4.6	输出进程统计信息	369
第 12 章	时间和日历历程	372
12.1	系统时间	372
12.2	日历时间	374
第 13 章	信号及信号处理	378
13.1	信号的基础	378
13.1.1	信号的基本概念	378
13.1.2	产生信号	379
13.1.3	处理信号	379
13.1.4	常用信号的使用方法	380
13.2	信号的影响	382
13.2.1	重入	382
13.2.2	原子操作	385
13.2.3	中断系统调用	387
13.3	信号处理函数	387
13.3.1	设置信号处理函数	387
13.3.2	发送信号	391
13.3.3	向进程本身发送信号	392
13.3.4	设置 Linux 定时器	393
13.3.5	定时等待 I/O	395
13.3.6	挂起进程	397
13.3.7	进程休眠	399
13.4	信号集与屏蔽信号	402
13.4.1	信号集和信号集处理函数	402
13.4.2	屏蔽信号	404
13.4.3	处理未决信号	406

13.4.4	高级信号注册函数	409
13.4.5	信号选项实例——SA_NOCLDWAIT 选项	410
13.4.6	信号选项实例——SA_NODEFFER 选项	412
13.4.7	信号选项实例——SA_RESETHAND 选项	414
第 14 章	进程间通信	416
14.1	进程间通信概述	416
14.1.1	进程间通信简介	416
14.1.2	进程间通信的难点	417
14.1.3	IPC 的多种方式	417
14.2	管道	418
14.2.1	管道的概念	418
14.2.2	匿名半双工管道	418
14.2.3	匿名半双工管道的读写操作	420
14.2.4	创建管道的标准库函数	423
14.3	FIFO 管道	426
14.3.1	FIFO 的概念	426
14.3.2	创建 FIFO	427
14.3.3	FIFO 的读写操作	428
14.3.4	FIFO 的缺点	432
14.4	System V IPC/POSIX IPC	433
14.4.1	IPC 对象的概念	433
14.4.2	IPC 对象的问题	435
14.4.3	IPC 对象系统命令	435
14.5	共享内存	436
14.5.1	共享内存的概念	436
14.5.2	共享内存的创建	437
14.5.3	共享内存的操作	439
14.5.4	共享内存使用注意事项	442
14.6	信号量	442
14.6.1	信号量的概念	442
14.6.2	信号量的创建	443
14.6.3	信号量集的操作	446
14.7	消息队列	448
14.7.1	消息队列的概念	448
14.7.2	创建消息队列	449
14.7.3	读写消息队列	452
第 15 章	线程	456
15.1	线程与进程	456

15.1.1	线程的概念	456
15.1.2	线程的优势	457
15.2	线程标识符	458
15.3	线程基本操作	458
15.3.1	创建线程	458
15.3.2	向线程体函数传递参数	460
15.3.3	线程访问资源的限制	462
15.3.4	终止线程	466
15.3.5	正确得到线程退出信息的方法	470
15.3.6	取消一个线程的运行	473
15.3.7	线程退出函数	476
第 16 章	线程高级操作	480
16.1	线程同步——使用互斥量	480
16.1.1	初始化与销毁互斥量	480
16.1.2	得到与释放互斥量	481
16.1.3	使用互斥量的实例	482
16.2	线程同步——使用读写锁	489
16.2.1	初始化与销毁读写锁	489
16.2.2	得到与释放互斥锁	490
16.2.3	使用互斥量与读写锁的比较	491
16.3	线程属性	499
16.3.1	创建和销毁属性结构	499
16.3.2	线程的分离状态	500
16.3.3	分离一个已经创建的线程	502
16.3.4	线程栈的属性	503

第 4 篇 Linux 文件操作

第 17 章	文件 I/O	508
17.1	文件描述符的概念	508
17.2	文件 I/O 操作	508
17.2.1	打开一个文件	508
17.2.2	打开文件的出错情况	512
17.2.3	关闭一个文件	513
17.2.4	创建一个新文件	514
17.2.5	文件定位	515
17.2.6	文件截短	517