Broadview
www.broadview.com.cn

亚嵌教育
AKA www.akaedu.org
作品系列

嵌入式
技术丛书

嵌入式Linux上的C语言编程实践

北京亚嵌教育研究中心 组编
韩超 魏治宇 廖文江 等著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

亚嵌教育
AKA www.akaedu.org
作品系列

嵌入式
技术丛书

嵌入式Linux上的C语言编程实践

北京亚嵌教育研究中心 组编
韩超 魏治宇 廖文江

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

目前嵌入式技术和 Linux 程序开发技术成为计算机界比较流行的两大技术。作为一种非常基础和常用的编程语言，C 语言在嵌入式系统开发中体现了其强大的软硬件操控能力。本书重点关注嵌入式 Linux 中的 C 语言编程，目的在于帮助读者在基本掌握 C 语言的基础上，逐渐掌握嵌入式 Linux 中的 C 语言开发技术。本书包含程序环境搭建、调试技术、库函数、高级编程、程序优化等方面的知识。本书的讲解由浅入深，知识点突出，其中的一些示例取自常见技术和项目中的精华，工程应用性强。

本书适合高校学生阅读，帮助其向具有研发能力的工程技术人员过渡，同样也适用于嵌入式软件开发人员补充知识、开阔眼界。

图书在版编目（CIP）数据

嵌入式 Linux 上的 C 语言编程实践 / 韩超等著；北京亚嵌教育研究中心组编. —北京：电子工业出版社，2009.2
（嵌入式技术丛书）
ISBN 978-7-121-07999-3

Ⅰ. 嵌… Ⅱ. ①韩… ②北… 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（2008）第 199118 号

责任编辑：许　艳
印　　刷：北京智力达印刷有限公司
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1092　1/16　印张：24.5　字数：476.3 千字
印　　次：2009 年 2 月第 1 次印刷
印　　数：4000 册　定价：49.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

《嵌入式 Linux 上的 C 语言编程实践》是一本提高嵌入式开发基本功的图书。本书的关键词为“嵌入式”和“C 语言”。C 语言是当前嵌入式领域开发中使用的主要语言，也是嵌入式开发最重要的基本功所在。同时，嵌入式领域的发展也为 C 语言提供了广阔的应用场景。

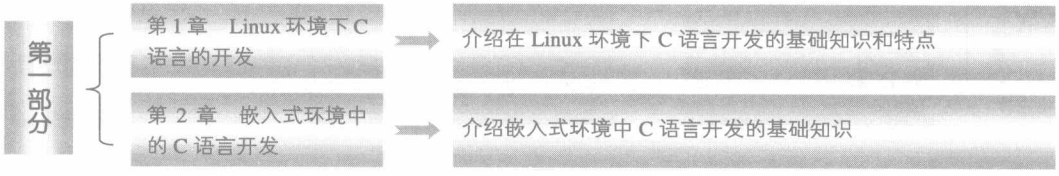
本书特点

- 虽然以介绍 C 语言为主，但更重要的是体现了 C 语言、“嵌入式”和“Linux”的关系。
- 虽然是基于 Linux 开发环境来介绍 C 语言编程的知识和技巧，但是书中的开发理念和技巧也适用于其他操作系统。
- 深入地挖掘了 C 语言与硬件的关系，突出嵌入式系统中编程的基本理念。
- 目前单纯讲解 C 语言和嵌入式开发的书籍较多，但是将二者相结合的书籍还不多见，本书的目标就是将二者相结合，立志于提高读者在嵌入式开发中最重要的基本功。

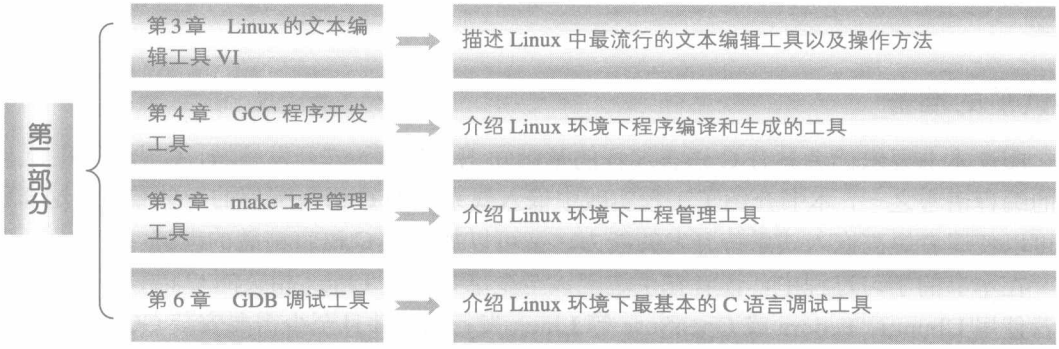
本书结构

本书由五个部分组成，即按照知识的组织结构，分成基础知识、Linux 环境中 C 语言的开发环境和工具、库函数、C 语言高级编程和在嵌入式环境下的 C 语言编程。

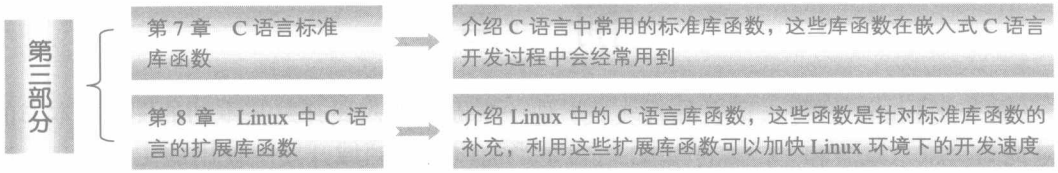
第一部分（第 1~2 章）：基础知识，体现本书 Linux 和嵌入式的特点。



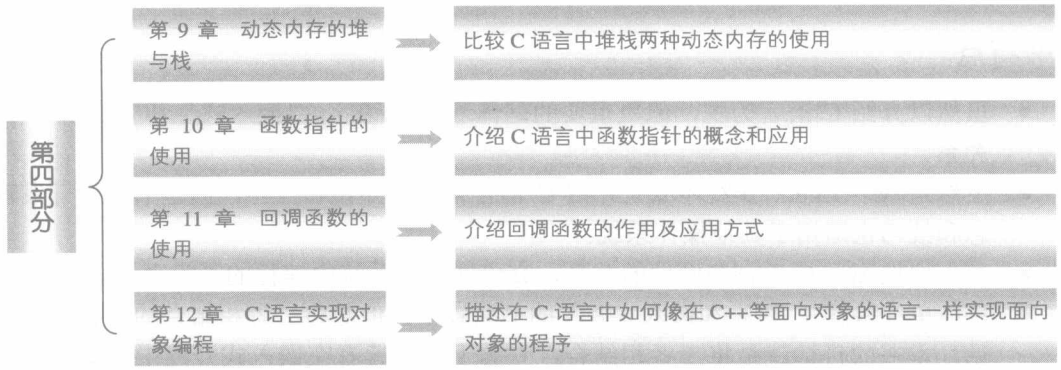
第二部分（第 3~6 章）：Linux 环境中 C 语言的开发环境和工具，介绍最流行的文本编辑、程序开发、工程管理及调试工具。



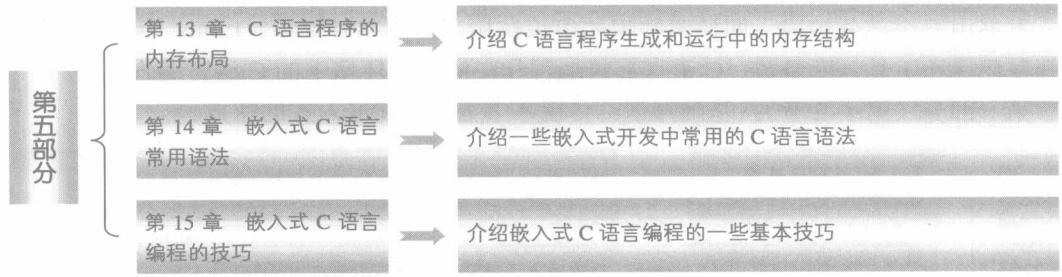
第三部分（第 7~8 章）：库函数，分两章、有重点地介绍 C 语言库函数的使用。



第四部分（第 9~12 章）：C 语言高级编程，介绍一些 C 语言常用的语法特性。



第五部分（第 13~15 章）：在嵌入式环境下的 C 语言编程，主要关注 C 语言在嵌入式系统中的理念和方法。



通过阅读本书，读者可以提高 C 语言高级编程的能力；学习 Linux 环境中 C 语言编程的方法；深入地理解 C 语言与底层硬件控制的关系；理解 C 语言在嵌入式领域内的应用。读者可以在深入学习 C 语言的精华内容的同时，对嵌入式领域的开发有一个基本的了解。

阅读前的准备

阅读本书要求读者具有 C 语言编程的基础知识。在嵌入式开发领域，C 语言也是最重要的编程语言之一，本书介绍了工程实践中嵌入式系统的 C 语言编程的一些要点。这样不仅可以使读者提高嵌入式系统开发的能力，又可以加深对 C 语言精髓的理解。

在本书的学习过程中，读者在手边应该至少准备一套 Linux 相关的 C 语言开发环境，推荐使用 Ubuntu、Fedora 或 OpenSuse 等 Linux 发行版，并且至少需要安装 GCC 工具。如

果不方便使用 Linux 环境，也可以使用 Windows 中的 Cygwin 或 DevC++。在文档方面，除了参考相关书籍之外，最好的方式是使用 Linux 中的 man 文档。推荐采用边学边练的方式，对于书中点到为止的知识，应深入学习、举一反三，以求达到事半功倍的效果。

本书由北京亚嵌教育研究中心组织编写，嵌入式系统资深工程师韩超结合该领域工程经验、知识技术传播经验、教学经验以及教材编写经验对全书进行规划，力求全书符合大陆读者阅读习惯并且具有实用价值。本书的第 1 章、第 2 章、第 4 至 6 章、第 11 至 15 章由韩超编写，第 3 章、第 7 章、第 8 章由魏治宇编写，第 9 章由廖文江编写，全书由韩超统稿。全书的审阅和修订工作由韩超及亚嵌教育研究中心完成。此外参与本书编写和审校工作的还有何晓龙、孙剑、夏鹏万等。

本书的编写结合了亚嵌教育研究中心的多位技术专家思想和教学思路，同时本书出版也是亚嵌多年开发和技术推广的努力成果。

目　录

第一部分　基础知识

第 1 章　Linux 环境下 C 语言的开发 ……2

1.1　Linux 下的 C 语言开发环境 ……2
1.2　在 Linux 中使用 C 语言开发 ……3
1.2.1　开发流程和开发工具 ……3
1.2.2　Linux 中程序的运行原理 ……4

第 2 章　嵌入式环境中的 C 语言开发 ……7

2.1　嵌入式 C 语言的开发环境 ……7
2.2　嵌入式开发中 C 语言编程要点 ……9

第二部分　Linux 环境中 C 语言的开发环境和工具

第 3 章　Linux 的文本编辑工具 VI ……12

3.1　VI 编辑器概述 ……12
3.1.1　VI 简介 ……12
3.1.2　VI 的工作模式和使用之前的准备 ……12
3.1.3　进入和退出 VI ……13
3.2　VI 的增强版 VIM ……16
3.3　VI 编辑器的基本使用方法 ……17
3.3.1　在屏幕上移动光标 ……17
3.3.2　插入文本 ……20
3.3.3　删除文本 ……22
3.3.4　修改文本内容 ……25
3.3.5　替换文本内容 ……27
3.3.6　合并文本内容 ……30
3.3.7　移动文本内容 ……30
3.4　VI 编辑器的命令和高级操作 ……32
3.4.1　VI 常用命令的列表 ……32
3.4.2　VI 的一些高级的操作和使用技巧 ……35

第 4 章　GCC 程序开发工具 ……39

4.1　GNU 工具综述 ……39
4.2　GCC 的编译和连接 ……43
4.2.1　工程示例 ……43
4.2.2　编译、汇编和连接 ……46
4.2.3　动态库 ……48
4.3　GCC 的二进制工具 ……49
4.3.1　ar（归档工具） ……49
4.3.2　readelf（读取 ELF 格式文件信息） ……51
4.3.3　strings（查看字符串） ……54
4.3.4　nm（显示符号信息） ……55
4.3.5　strip（删除符号） ……57
4.3.6　objdump（显示目标文件信息） ……58
4.3.7　objcopy（复制目标文件） ……63

第 5 章　make 工程管理工具 ……67

5.1　make 和 Makefile ……67
5.1.1　make 机制概述 ……67
5.1.2　make 和 Makefile 的使用 ……68
5.2　Makefile 使用示例 ……69
5.2.1　简单的 Makefile ……69
5.2.2　Makefile 中的依赖关系 ……71
5.2.3　Makefile 中使用隐含规则来编译程序 ……73
5.2.4　Makefile 中指定依赖关系的编译 ……76
5.3　自动生成 Makefile ……78
5.3.1　自动生成 Makefile 的意义和相关工具 ……78
5.3.2　自动生成 Makefile 的流程 ……79

第 6 章　GDB 调试工具……85
6.1　GDB 简介……85
6.2　使用 GDB 调试程序……86
6.2.1　基本操作……88
6.2.2　查看命令……90
6.2.3　高级命令……92
6.2.4　attach 命令的使用……94
6.3　远程 GDB 调试……95
6.3.1　本地 GDB 调试和远程 GDB 调试的比较……95
6.3.2　远程 GDB 调试流程……97
6.3.3　远程 GDB 调试示例……98

第三部分　库函数

第 7 章　C 语言标准库函数……106
7.1　ISO 的 C 语言标准库函数分类……106
7.2　标准格式化输入/输出类函数……107
7.2.1　scanf 函数：格式化输入字符串……107
7.2.2　printf 函数：格式化输出字符串……109
7.2.3　putchar 函数：输出字符到标准输出……111
7.2.4　getchar 函数：从标准输入获取字符……111
7.2.5　putc 函数：向文件输出字符……112
7.2.6　getc 函数：从文件输入字符……112
7.2.7　gets 函数：获得字符串……112
7.2.8　puts 函数：输出指定字符串……113
7.2.9　ungetc 函数：把字符写回流中……113
7.3　字符处理类函数……114
7.4　字符串处理及转换函数……116
7.4.1　sprintf 函数：格式化输出字符串到一个缓冲区……116
7.4.2　strcat 和 strncat 函数：字符串连接……119
7.4.3　strcpy 和 strncpy 函数：字符串复制……120
7.4.4　strcmp 和 strncmp 函数：字符串比较……121
7.4.5　strlen 函数：获取字符串长度……122
7.4.6　strchr 和 strrchr 函数：字符/字符串定位……122
7.4.7　strstr 函数：字符串查找……123
7.4.8　strrev 函数：字符串逆序……124
7.4.9　strupr 和 strlwr 函数：字母形式转换……125
7.4.10　strdup 和 strndup 函数：字符串复制……125
7.4.11　memset 函数：内存设置……126
7.4.12　memmove 函数：内存移动……126
7.4.13　memcmp 函数：内存比较……127
7.4.14　memcpy 函数：内存复制……128
7.5　数学计算类函数……128
7.6　数据结构和算法类函数……133
7.6.1　bsearch 函数：二元搜索……133
7.6.2　lfind 函数：线性搜索……134
7.6.3　lsearch 函数：线性搜索……135
7.6.4　qsort 函数：利用快速排序法排列数组……136
7.6.5　rand 函数：产生随机数……136
7.6.6　srand 函数：设置随机数种子……137
7.7　文件 I/O 操作类相关函数……137
7.7.1　fopen 函数：打开文件……138
7.7.2　fclose 函数：关闭文件……139
7.7.3　fgetc 函数：从文件中读取一个字符……139
7.7.4　fputc 函数：将一指定字符写入文件流中……139
7.7.5　fgets 函数：从文件中读取一字符串……140

7.7.6 fputs 函数：将一指定的字符串写入文件内……140
7.7.7 rewind 函数：重设文件流的读写位置为文件开头……141
7.7.8 ftell 函数：取得文件流的读取位置……141
7.7.9 fseek 函数：移动文件流的读写位置……141
7.7.10 fwrite 函数：将数据写至文件流……142
7.7.11 fread 函数：从文件流读取数据……142
7.7.12 remove 函数：删除文件……143
7.7.13 rename 函数：更改文件名称或位置……143
7.7.14 freopen 函数：重新打开文件……144
7.7.15 fflush 函数：同步缓冲区……144
7.7.16 fgetpos 函数：获得文件位置……145
7.7.17 fsetpos 函数：设置文件位置……145
7.7.18 mktemp 函数：建立临时文件……146
7.7.19 tmpfile 函数：临时文件……146
7.7.20 tmpnam：得到临时文件名……147
7.8 日期时间类函数……147
7.8.1 clock 函数：获得 CPU 时间……148
7.8.2 time 函数：获得当前日历时间……148
7.8.3 difftime 函数：获得时间差值……148
7.8.4 gmtime 函数：将日历时间转换成 UTC 时间……149
7.8.5 mktime 函数：将 UTC 时间转换成日历时间……149
7.8.6 asctime 函数：将 UTC 时间转换成字符串……149
7.8.7 ctime 函数：将日历时间转换成当地时间的字符串……150
7.8.8 localtime 函数：将日历时间转换成本地时间……150
7.8.9 strftime 函数：转换日期和时间格式……151
7.9 国际化和本地化函数……152
7.9.1 setlocale 函数：本地化控制函数……153
7.9.2 localeconv 函数：本地化转换……154
7.10 错误处理类函数……155
7.10.1 clearerr 函数：清除流中的结束指示符和错误指示符……155
7.10.2 feof 函数：指示文件结束……155
7.10.3 ferror 函数：指示文件出错……156
7.10.4 perror 函数：输出出错信息……156
7.10.5 errno 函数：错误编号记录……156
7.11 其他一些工具函数……157
7.11.1 assert 函数：程序诊断……157
7.11.2 长跳转函数……157
7.11.3 可变长的参数控制函数……160
7.11.4 获取结构体成员函数（宏）……161
7.12 一些标准库中有用的宏……161
第 8 章 Linux 中 C 语言的扩展库函数……163
8.1 文件 I/O 操作函数……163
8.1.1 open 函数：打开文件……163
8.1.2 close 函数：关闭文件……164
8.1.3 read 函数：读文件……165
8.1.4 write 函数：写文件……165
8.1.5 lseek 函数：文件定位……167
8.1.6 ioctl 函数：文件控制……167
8.1.7 flock 函数：锁定文件……167
8.1.8 mmap 函数和 munmap 函数：内存映射……168

8.1.9 create 函数：创建新文件 ……170
8.1.10 dup 函数和 dup2 函数：复制文件描述符……171
8.1.11 fcntl 函数：改变已打开的文件的属性……171
8.2 文件权限相关的操作函数……172
8.2.1 access 函数：判断是否具有存取文件的权限……172
8.2.2 chown 函数和 fchown 函数：改变文件的所有者……173
8.2.3 chmod 函数和 fchmod 函数：改变权限……173
8.2.4 unlink 函数：删除文件……173
8.2.5 utime 函数和 utimes 函数：改变文件时间……174
8.2.6 umask 函数：设置建立新文件时的权限掩码……175
8.2.7 link 函数：建立文件连接……175
8.2.8 stat 函数、fstat 函数和 lstat 函数：获取文件信息……175
8.3 用户组操作函数 ……176
8.3.1 getgid 函数和 setgid 函数：获得/设置组识别码……176
8.3.2 getegid 函数和 setegid 函数：获得/设置有效的组识别码……177
8.3.3 getuid 函数和 setuid 函数：获得/设置真实的用户识别码……177
8.3.4 geteuid 函数和 seteuid 函数：获得/设置有效的用户识别码……178
8.3.5 getgroups 函数和 setgroups 函数：获得/设置组代码……178
8.4 信号类函数……179
8.4.1 kill 函数：传送信号给指定的进程……181
8.4.2 raise 函数：信号发送函数 ……181
8.4.3 alarm 函数：设置超时函数……182
8.4.4 signal 函数：信号安装函数 ……182
8.5 进程处理函数……183
8.5.1 getpid 函数和 getppid 函数：获得进程 ID 和父进程 ID……183
8.5.2 fork 函数：建立子进程 ……183
8.5.3 sleep 函数和 usleep 函数：让进程暂停执行一段时间 ……185
8.5.4 exec 函数族：找到可执行文件……185
8.5.5 _exit 函数和_Exit 函数：结束进程执行……188

第四部分 C 语言高级编程

第 9 章 动态内存的堆与栈……190
9.1 程序内存区域的使用 ……190
9.1.1 静态内存与动态内存……190
9.1.2 C 语言中的动态内存 ……191
9.2 C 程序中栈空间的使用 ……196
9.2.1 参数使用栈空间……196
9.2.2 自动变量使用栈空间 ……199
9.2.3 程序中较大的栈……201
9.2.4 栈空间的特性……202
9.3 C 程序中的堆空间使用 ……203
9.3.1 分配和释放堆内存的库函数…203
9.3.2 库函数使用……204
9.3.3 堆内存的特性……218
9.4 堆内存和栈内存使用的比较……222
9.4.1 利用返回值传递信息 ……222
9.4.2 利用参数传递信息 ……226
9.4.3 堆与栈内存管理的区别 ……231
第 10 章 函数指针的使用……232
10.1 函数指针的概念 ……232
10.1.1 C 语言函数的本质 ……232
10.1.2 函数指针在 C 语言中的意义……234
10.2 函数指针的使用 ……237
10.2.1 函数指针使用初步……237
10.2.2 函数指针的类型定义 ……240

10.2.3 函数指针作为结构体成员……242
10.2.4 函数指针作为函数的参数……243
10.2.5 函数指针作为函数的返回值……244
10.2.6 函数指针数组……246
10.3 函数指针使用示例……248
第 11 章 回调函数的使用……252
11.1 回调函数的概念与作用……252
11.1.1 程序调用的方式……252
11.1.2 回调函数的作用……254
11.2 回调函数的语法……254
11.2.1 简单的回调函数……254
11.2.2 完全形式的回调函数……256
11.3 回调函数的使用……259
11.3.1 qsort 中的回调函数……259
11.3.2 atexit 和 on_exit 函数的注册退出函数……263
第 12 章 C 语言实现对象编程……268
12.1 C 语言实现基于对象编程的概念与作用……268
12.2 C 语言基于对象编程实现封装……269
12.2.1 简单的程序示例……269
12.2.2 C 语言基于对象编程的详解……272
12.2.3 C 语言基于对象编程与 C++面向对象编程的对比……275
12.3 C 语言基于对象编程实现部分继承功能……278
12.3.1 利用数据结构的包含实现继承功能……279
12.3.2 利用私有指针实现继承功能……282
12.3.3 C 语言实现继承的总结……287
12.4 C 语言基于对象编程实现部分多态功能……288
12.4.1 利用操作指针组的包含实现多态功能……288
12.4.2 C 语言实现多态功能的总结…292
12.5 对 C 语言实现基于对象编程的思考……292
12.5.1 C 语言基于对象编程的特性……292
12.5.2 C 语言基于对象编程中接口、实现和调用者的关系……293

第五部分 在嵌入式环境下的 C 语言编程

第 13 章 C 语言程序的内存布局……295
13.1 C 语言程序的存储区域……295
13.2 C 语言程序的段……297
13.2.1 段的分类……297
13.2.2 程序中段的使用……298
13.3 可执行程序的连接……301
13.3.1 可执行程序的组成……301
13.3.2 各个目标文件的关系……303
13.3.3 连接错误示例……304
13.4 C 语言程序的运行……309
13.4.1 RAM 调试运行……311
13.4.2 固化程序的 XIP 运行……312
13.4.3 固化程序的加载运行……313
13.4.4 C 语言程序的运行总结……315
第 14 章 嵌入式 C 语言常用语法……317
14.1 内存指针操作……317
14.1.1 内存操作的意义……317
14.1.2 使用指针操作内存……319
14.1.3 volatile 的使用……324
14.1.4 嵌入式系统指针的实际应用……325
14.2 位操作……327
14.2.1 位操作的意义……327

14.2.2 位操作的语法……328
14.3 大小端与对齐问题……330
14.3.1 大小端问题……331
14.3.2 内存对齐问题……335
14.3.3 结构体成员的对齐问题……338
14.4 程序的跳转……344
14.4.1 嵌入式系统程序跳转的类型……344
14.4.2 C 语言中实现程序的跳转……345
第 15 章 嵌入式 C 语言编程的技巧……348
15.1 程序的优化技巧……348
15.1.1 循环缓冲区……348
15.1.2 查表法……350
15.1.3 针对循环执行效率的优化……353
15.2 关于小数运算……355
15.3 函数参数和返回值的传递……357
15.4 变量的初始化技巧……360
15.4.1 数组的初始化……360
15.4.2 结构体的初始化……362
15.4.3 变量的初始化总结……362
15.5 程序的调试和宏使用的技巧……363
15.5.1 打印文件、函数和程序行……363
15.5.2 #：字符串化操作符……364
15.5.3 ##：连接操作符……366
15.5.4 调试宏的第一种定义方式……367
15.5.5 调试宏的第二种定义方式……368
15.5.6 对调试语句进行分级审查……369
15.5.7 条件编译调试语句……370
15.5.8 使用 do…while 的宏定义……372
15.6 代码剖析……373
参考文献……378

第一部分

基础知识

- 第 1 章　Linux 环境下 C 语言的开发
- 第 2 章　嵌入式环境中的 C 语言开发

第 1 章

Linux 环境下 C 语言的开发

本章介绍 Linux 操作系统环境下 C 语言开发的基本概念和程序运行的原理。

在本章的学习中，读者应重点关注以下内容：

- Linux 中 C 语言开发的流程和工具
- Linux 中 C 语言程序的运行机制

1.1 Linux 下的 C 语言开发环境

Linux 和 C 语言有很深的渊源，因为 Linux 本身就是用 C 语言编写的。同时，在 Linux 操作系统中也提供了 C 语言的开发环境。这些开发环境一般包括程序生成工具、程序调试工具、工程管理工具等。

1. 程序生成工具

在 Linux 中，一般使用 GCC（GNU Compiler Collection）作为程序生成工具。GCC 提供了 C 语言的编译器、汇编器、连接器以及一系列辅助工具。GCC 可以用于生成 Linux 中的应用程序，也可以用于编译 Linux 内核和内核模块，是 Linux 中 C 语言开发的核心工具。

2. 程序调试工具

GDB 是 Linux 中一个强大的命令行调试工具，使用 GDB 调试 C 语言的时候，可以使用设置断点、单步运行、查看变量等功能。

3. 工程管理工具

在 Linux 操作系统下的程序开发中，一般使用 make 和 Makefile 作为工程管理工具。在工程管理方面，有效地使用它们可以统筹工程中的各个文件，并在编译过程中根据时间戳，有选择地进行编译，减少程序生成时间。

1.2 在Linux中使用C语言开发

在Linux操作系统中，C语言程序的开发和其他环境类似，程序生成主要分成编译、汇编、连接等几个步骤。在Linux中使用文本编辑工具编辑程序源代码也是程序开发的重要步骤。

1.2.1 开发流程和开发工具

C语言程序的开发过程是：使用编辑工具编写文本形式的C语言源文件，然后编译生成以机器代码为主的二进制可执行程序的过程。由源文件生成可执行程序的开发过程如图1-1所示。

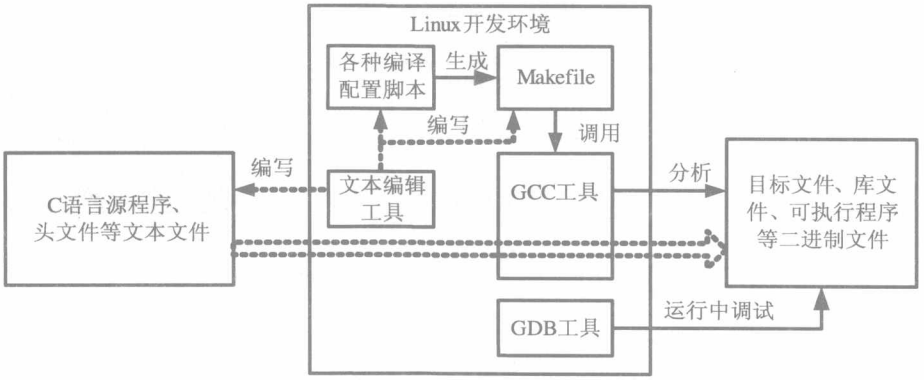


图1-1 Linux中C语言程序的开发流程

编译是指把用高级语言编写的程序转换成相应处理器的汇编语言程序的过程。从本质上讲，编译是一个文本转换的过程。对嵌入式系统而言，一般要把用C语言编写的程序转换成处理器的汇编代码。编译过程包含了C语言的语法解析和汇编语言的生成两个步骤。汇编一般是逐个文件进行的，对于每一个C语言编写的文件，可能还需要进行预处理。

汇编是从汇编语言程序生成目标系统的二进制代码（机器代码）的过程。机器代码的生成和处理器有密切的联系。相对于编译过程的语法解析，汇编的过程相对简单。这是因为对于一款特定的处理器，其汇编语言和二进制的机器代码是一一对应的。汇编过程的输入是汇编代码，这个汇编代码可能来源于编译过程的输出，也可以是直接用汇编语言书写的程序。

连接过程是指将汇编生成的多段机器代码组合成一个可执行程序。一般来说，通过编译和汇编过程，每一个源文件将生成一个目标文件。连接器的作用就是将这些目标文件组合起来，组合的过程包括了代码段、数据段等部分的合并，以及添加相应的文件头。

在Linux的C语言程序生成过程中，源代码经过编译-汇编-连接生成可执行程序。GCC是Linux下主要的程序生成工具，它除了编译器、汇编器、连接器外，还包括一些辅助工具。

调试是程序开发一个很重要的环节。在 Linux 的程序开发中，最主要的调试工具是 GDB。GDB 是一个命令行调试工具，可以实现在程序中设置断点、单步执行、查看对应源代码等功能。

虽然 Linux 中基本的开发工具 GCC 和 GDB 都是命令行工具，但是它们也可以和 IDE（集成开发环境）结合使用。

Linux 下程序的开发过程及相关工具的使用如图 1-2 所示。

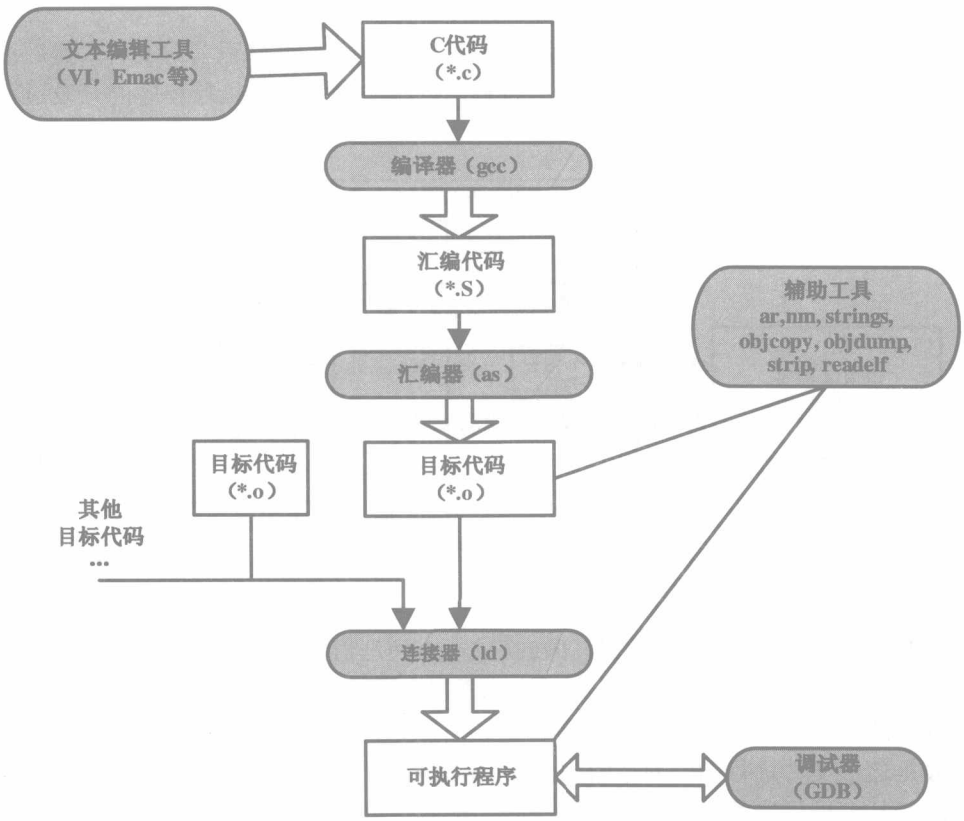


图 1-2　Linux 程序开发过程及相关工具

1.2.2　Linux 中程序的运行原理

在 Linux 的开发环境中，C 语言程序的运行环境如图 1-3 所示。

作为 UNIX 操作系统的一种，Linux 的操作系统提供了一系列的接口，这些接口被称为系统调用（System Call）。在 UNIX 的理念中，系统调用“提供的是机制，而不是策略”。C 语言的库函数通过调用系统调用来实现，库函数对上层提供了 C 语言库文件的接口。在应用程序层，通过调用 C 语言库函数和系统调用来实现功能。一般来说，应用程序大多使用 C 语言库函数实现其功能，较少使用系统调用。

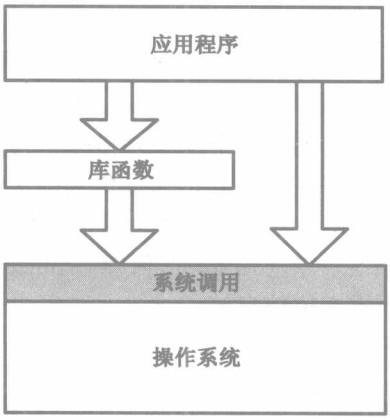


图 1-3 Linux 下 C 语言程序的结构

在 Linux 等系统的环境中，C 语言库及其头文件都是系统的一部分，只要安装了编译工具即可以完成 C 语言程序的开发。这点与 Windows 中程序的开发有所不同，Windows 中一般需要安装开发包才能进行程序开发。

C 语言程序经过编译-汇编-连接，最终生成可执行程序格式。可执行程序中包含两个部分的内容：

- 程序头
- 程序主体（二进制机器代码）

在程序头中包含了供操作系统加载的信息，操作系统根据这些信息加载可执行程序。而可执行程序的主体依然是二进制的机器代码。程序在运行的时候，正是靠逐条地执行这些机器代码，形成程序运行的序列。

在 Linux 操作系统中，普遍使用 ELF 格式来作为可执行程序或者程序生成过程中的中间格式。ELF（Executable and Linking Format，可执行连接格式）是 UNIX 系统实验室（USL）作为应用程序二进制接口（Application Binary Interface，ABI）而开发和发布的。工具接口标准委员会（TIS）选择了正在发展中的 ELF 标准作为工作在 32 位 Intel 体系上不同操作系统之间可移植的二进制文件格式。

如果开发者定义了一个二进制接口集合，ELF 标准允许用它来支持流线型的软件运行。通过它可以减少不同执行接口的数量，也可以减少重新编程和重新编译的代码。

ELF 文件格式包括三种主要的类型：可执行文件、可重定向文件、共享库。

1. 可执行文件（应用程序）

可执行文件包含了代码和数据，是可以直接运行的程序。

2. 可重定向文件（*.o）

可重定向文件又称为目标文件，它包含了代码和数据（这些数据是和其他重定位文件和共享的 object 文件一起连接时使用的）。