高等院校嵌入式人才培养规划教材

Gaodeng Yuanxiao Qianrushi Rencai Peiyang Guihua Jiaocai

嵌入式 Linux C语言开发

华清远见嵌入式学院 曾宏安 主编

Qianrushi
Linux C Yuyan Kaifa

精选实用编程

重视实际应用

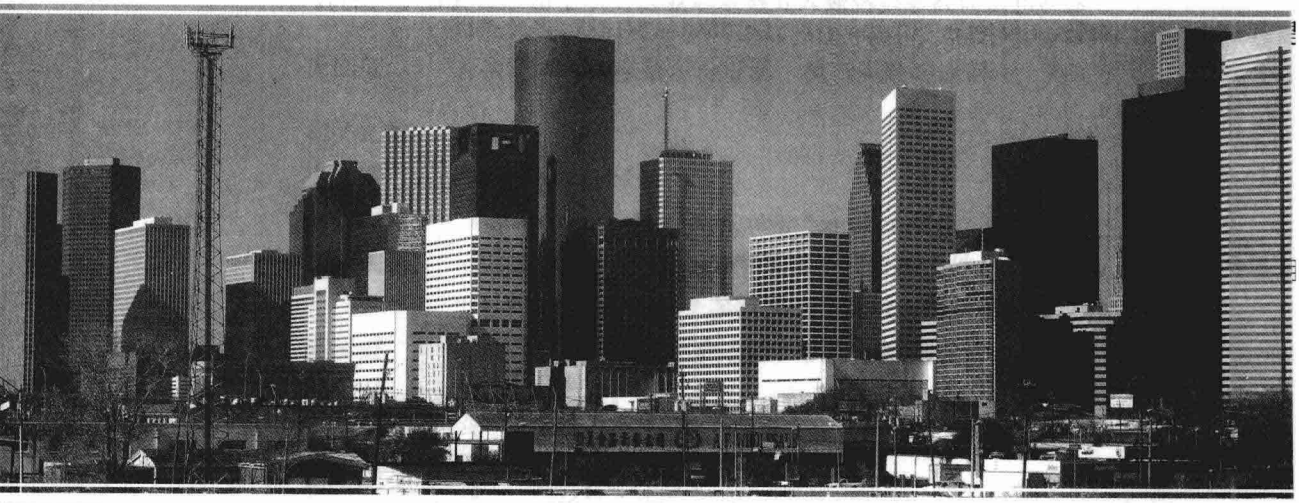
全部代码示例

人民邮电出版社
POSTS & TELECOM PRESS

高等院校嵌入式人才培养规划教材
Gaodeng Yuanxiao Qianrushi Rencai Peiyang Guihua Jiaocai

嵌入式 Linux C语言开发

华清远见嵌入式学院 曾宏安 主编



Qianrushi
Linux C Yuyan Kaifa

人 民 邮 电 出 版 社
北 京

图书在版编目（CIP）数据

嵌入式Linux C语言开发 / 曾宏安主编. —北京：人民邮电出版社，2009.8
高等院校嵌入式人才培养规划教材
ISBN 978-7-115-21115-6

Ⅰ. 嵌… Ⅱ. 曾… Ⅲ. ①Linux操作系统－程序设计－高等学校－教材②C语言－程序设计－高等学校－教材
Ⅳ. TP316.89 TP312

中国版本图书馆CIP数据核字（2009）第121492号

内 容 提 要

本书介绍开发工具和 Linux C 语言基础、嵌入式 Linux C 语言高级用法、内核常见数据结构的解析与应用、嵌入式 Linux 编程基础、文件 I/O 操作相关的 C 语言应用、网络通信相关的 C 语言应用等，并设置了嵌入式 Linux C 函数参考附录。学习本书前应掌握 C 语言程序设计的基本知识。

本书可作为高等院校嵌入式技术专业以及电子信息类其他专业的教材。

高等院校嵌入式人才培养规划教材

嵌入式 Linux C 语言开发

◆ 主　　编　华清远见嵌入式学院　曾宏安
责任编辑　潘春燕
执行编辑　郭　晶

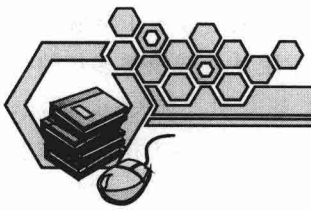
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京昌平百善印刷厂印刷

◆ 开本：787×1092　1/16
印张：17
字数：434 千字　　2009 年 8 月第 1 版
印数：1－3 000 册　　2009 年 8 月北京第 1 次印刷

ISBN 978-7-115-21115-6

定价：29.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154



前　言

随着消费群体对产品要求的日益提高，嵌入式技术在机械器具制造、电子产品制造、通信、信息服务等行业领域得到了大显身手的机会，应用日益广泛，相应地企业对嵌入式人才的需求也越来越多。因此近几年来，各高等院校开始纷纷开设嵌入式专业或课程。但是，各院校在嵌入式专业教学建设的过程中几乎都面临教材难觅的困境。虽然目前市场上的嵌入式开发相关书籍比较多，但几乎都是针对有一定基础的行业内研发人员而编写的，并不完全符合学校的教学要求。学校教学需要一套充分考虑学生现有知识基础和接受度的，明确各门课程教学目标的，便于学校安排课时的嵌入式专业教材。

针对教材缺乏的问题，我们以多年来在嵌入式工程技术领域内人才培养、项目研发的经验为基础，汇总了近几年积累的数百家企业对嵌入式研发相关岗位的真实需求，调研了数十所开设“嵌入式工程技术”专业的高等院校的课程设置情况、学生特点和教学用书现状。通过细致的整理和分析，对专业技能和基本知识进行合理划分，我们编写了这套高等院校嵌入式人才培养规划教材，包括以下 5 本：

《ARM 嵌入式体系结构与接口技术》

《μC/OS II 嵌入式操作系统》

《嵌入式 Linux 操作系统》

《嵌入式 Linux C 语言开发》

《嵌入式应用程序设计》

本套教材按照专业整体教学要求组织编写，各自对应的主干课程之间既相对独立又有机衔接，整套教材具有系统性。《ARM 嵌入式体系结构与接口技术》侧重介绍接口技术；在操作系统教材方面，考虑到各院校不同的教学侧重点，编写了 μC/OS II 和 Linux 两个版本；考虑到本专业对学生 C 语言能力要求较高，编写了《嵌入式 Linux C 语言开发》这本少课时的教材，可供“C 语言基础”课程的后续提高课程使用；《嵌入式应用程序设计》介绍了贯穿前面所学知识的实训内容，供“Linux 应用开发”课程使用。

本书是其中之一。全书共 7 章，第 1 章介绍了嵌入式 Linux 下常用的 C 语言开发工具，为后面的学习打下基础。第 2 章和第 3 章讲解了嵌入式 Linux C 语言的基础和高级用法。第 4 章介绍了嵌入式 Linux 内核中常见的数据结构。第 5 章为文件操作，主要讲述了 Linux 系统调用、Linux 文件 I/O 系统、底层文件 I/O 操作、嵌入式 Linux 串口应用编程、标准 I/O 编程等内容。第 6 章为进程/线程编程，主要讲解了 Linux 系统下进程的基本概念、与进程管理相关的系统调用、进程间通信的方法和多线程编程的知识。第 7 章为网络通信相关的 C 语言应用，主要讲解了 Linux 环境下网络编程方法。涉及网络的非阻塞访问、异步处理、多路复用等具体实现。

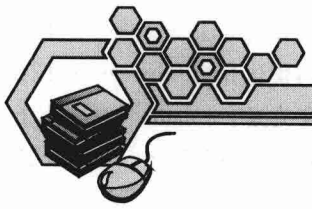
本书由曾宏安主编。本书的完成需要感谢华清远见嵌入式学院，教材内容参考了学院与嵌入式企业需求无缝对接的、科学的专业人才培养体系。同时，嵌入式学院从业或执教多年的行业专家团队也对教材的编写工作做出了贡献，孙天泽、刘洪涛、穆煜、赵苍明、季久峰、胡波、贾燕枫等老师在书稿的编写过程中认真阅读了所有章节，提供了大量在实际教学中积累的重要素材，对教材结构、内容提出了中肯的建议，并在后期审校工作中提供了很多帮助，在此表示衷心的感谢。

本书所有源代码、PPT 课件、教学素材等辅助教学资料，请到 www.ptpedu.com.cn 下载。

由于作者水平所限，书中不妥之处在所难免，恳请读者批评指正。对于本书的批评和建议，可以发到 www.embedu.org 技术论坛。

编　者

2009 年 7 月



目 录

第1章 嵌入式 Linux C 语言开发工具

任何应用程序的开发都离不开编辑器、编译器及调试器，嵌入式 Linux 的 C 语言开发也一样，它也有一套优秀的编辑、编译及调试工具。

掌握这些工具的使用是至关重要的，它直接影响到程序开发的效率。希望读者通过自己的实践，熟练掌握这些工具的使用。通过本章的学习，读者将会掌握如下内容：

- C 语言产生的历史背景
- 嵌入式 Linux 下 C 语言的开发环境
- 嵌入式 Linux 下的编辑器 vi
- 嵌入式 Linux 下的编译器 GCC
- 嵌入式 Linux 下的调试器 GDB
- 嵌入式 Linux 下的工程管理器 make
- eclipse 集成开发环境

1.1 嵌入式 Linux 下 C 语言概述

在嵌入式系统中，应用程序的主体是在宿主机中开发完成的。就嵌入式 Linux 而言，此过程则一般是在安装有 Linux 的宿主机中完成的。

本章中介绍的是嵌入式 Linux 下 C 语言的开发工具，用户在开发时往往是在 Linux 宿主机中对程序进行调试，然后再进行交叉编译。

1.1.1 C 语言简史

C 语言于 20 世纪 70 年代诞生于美国的贝尔实验室。在此之前，人们编写系统软件

主要使用汇编语言。汇编语言编写的程序依赖于计算机硬件，其可读性和可移植性都比较差。而高级语言的可读性和可移植性虽然较汇编语言好，但一般又不具备低级语言能够直观地对硬件实现控制和操作而且执行速度快等特点。

在这种情况下，人们迫切需要一种既具有一般高级语言特性，又具有低级语言特性的语言，于是 C 语言就应运而生了。由于 C 语言既具有高级语言的特点又具有低级语言的特点，因此迅速普及，成为当今最有发展前途的计算机高级语言之一。C 语言既可以用来编写系统软件，也可以用来编写应用软件。现在，C 语言已经被广泛地应用在除计算机行业外的机械、建筑和电子等各个行业中。

C 语言的发展历程如下。

① C 语言最初是美国贝尔实验室的 D.M.Ritchie 在 B 语言的基础上设计出来的，此时的 C 语言只是为了描述和实现 UNIX 操作系统的一种工作语言。在一段时间里，C 语言还只在贝尔实验室内部使用。

② 1975 年，UNIX 第 6 版公布后，C 语言突出的优点引起人们的普遍注意。

③ 1977 年出现了可移植的 C 语言。

④ 1978 年 UNIX 第 7 版的 C 语言成为后来被广泛使用的 C 语言版本的基础，被称为标准 C 语言。

⑤ 1983 年，美国国家标准协会（ANSI）根据 C 语言问世以来的各种版本，对 C 语言进行发展和扩充，并制定了新的标准，称为 ANSI C。

⑥ 1990 年，国际标准化组织（ISO）制定了 ISO C 标准，目前流行的 C 语言编译系统都是以它为标准的。

1.1.2 C 语言特点

C 语言兼有汇编语言和高级语言的优点，既适合于开发系统软件，又适合于编写应用程序，被广泛应用于事务处理、科学计算、工业控制、数据库技术等领域。

C 语言之所以能存在和发展，并具有强大的生命力，都要归功于其鲜明的特点。这些特点是多方面的，归纳如下。

1. C 语言是结构化的语言

C 语言采用代码及数据分隔的方式，使程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰，便于使用、维护以及调试。

C 语言是以函数形式提供给用户的，这些函数可被方便地调用，并具有多种循环、条件语句控制程序流向，从而使程序完全结构化。

2. C 语言是模块化的语言

C 语言主要用于编写系统软件和应用软件。一个系统软件的开发需要很多人经过几年的时间才能完成。一般来说，一个较大的系统程序往往被分为若干个模块，每一个模块用来实现特定的功能。

在 C 语言中，用函数作为程序的模块单位，便于实现程序的模块化。在程序设计时，将一些常用的功能模块编写成函数，放在函数库中供其他函数调用。模块化的特点可以大大减少重复编程。程序设计时，只要善于利用函数，就可减少劳动量、提高编程效率。

3. 程序可移植性好

C 语言程序便于移植。目前 C 语言在许多计算机上的实现大都是由 C 语言编译移植得到的，

不同计算机上的编译程序大约有 80%的代码是公共的。程序不做任何修改就可用于各种型号的计算机和各种操作系统。因此，特别适合在嵌入式开发中使用。

4. C 语言运算符丰富、代码效率高

C 语言共有 34 种运算符，使用各种运算符可以实现在其他高级语言中难以实现的运算。在代码质量上，C 语言可与汇编语言媲美，其代码效率仅比用汇编语言编写的程序低 10%～20%。

1.1.3 嵌入式 Linux C 语言编程环境

嵌入式 Linux C 语言程序设计与在其他环境中的 C 程序设计很类似，也涉及编辑器、编译链接器、调试器及项目管理工具的使用。现在我们先对这 4 种工具进行简单介绍，后面会一一进行讲解。

1. 编辑器

嵌入式 Linux 下的编辑器就如 Windows 下的 Word、记事本等一样，完成对所录入字符的编辑功能，最常用的编辑器有 vi（vim）和 Emacs，它们功能强大，使用方便，本书重点介绍 vi。

2. 编译链接器

编译过程包括词法、语法和语义的分析、中间代码的生成和优化、符号表的管理和出错处理等。在嵌入式 Linux 中，最常用的编译器是 GCC 编译器。它是 GNU 推出的功能强大、性能优越的多平台编译器，其执行效率与一般的编译器相比平均效率要高 20%～30%。

3. 调试器

调试器可以方便程序员在程序运行时进行源代码级的调试，但不是代码执行的必备工具。在程序开发的过程当中，调试所消耗的时间远远大于编写代码的时间。因此，有一个功能强大、使用方便的调试器是必不可少的。GDB 可以方便地设置断点、单步跟踪等，足以满足开发人员的需要。

4. 项目管理器

嵌入式 Linux 中的项目管理器“make”类似于 Windows 中 Visual C++里的“工程”管理，它是一种控制编译或者重复编译代码的工具。另外，它还能自动管理软件编译的内容、方式和时机，使程序员能够把精力集中在代码的编写上而不是在源代码的组织上。

1.2 嵌入式 Linux 编辑器 vi 的使用

vi 是 Linux 系统的第一个全屏幕交互式编辑工具。它从诞生至今一直得到广大用户的青睐，历经数十年后仍然是人们主要使用的文本编辑工具，足见其生命力之强，其强大的编辑功能可以同任何一个最新的编辑器相媲美。

虽然用惯了 Windows 中的 Word 等编辑器的读者在刚刚接触 vi 时或多或少会有些不适应，但使用过一段时间后，就能感受到它的方便与快捷。

Linux 系统提供了一个完整的编辑器家族系列，如 Ed、Ex、vi 和 Emacs 等，按功能它们可以分为两大类：行编辑器（Ed、Ex）和全屏幕编辑器（vi、Emacs）。行编辑器每次只能对一行进行操作，使用起来很不方便。而全屏幕编辑器可以对整个屏幕进行编辑，用户编辑的文件直接显示在屏幕上，从而克服了行编辑的那种不直观的操作方式，便于用户学习和使用，具有强大的功能。

1.2.1 vi 的基本模式

vi 编辑器具有 3 种工作模式，分别是命令行模式（command mode）、插入模式（insert mode）和底行模式（last line mode），各模式的功能区分如下。

1. 命令行模式（command mode）

在该模式下用户可以输入命令来控制屏幕光标的移动，删除字符、单词或行，移动复制某区段，也可以进入到底行模式或者插入模式下。

2. 插入模式（insert mode）

用户只有在插入模式下才可以进行字符输入，用户按［Esc］键可回到命令行模式下。

3. 底行模式（last line mode）

在该模式下，用户可以将文件保存或退出 vi，也可以设置编辑环境，如寻找字符串、显示行号等。这一模式下的命令都是以“:”开始。

不过在一般使用时，人们通常把 vi 简化成两个模式，即将底行模式（last line mode）也归入命令行模式中。

1.2.2 vi 的基本操作

1. 进入与离开 vi

进入 vi 可以直接在系统提示符下键入“vi < 文档名称 >”，vi 可以自动载入所要编辑的文档或是创建一个新的文档。如在 shell 中键入“vi hello.c”（新建文档）即可进入 vi 画面。如图 1.1 所示。

图 1.1 在 vi 中打开/新建文档

进入 vi 后，屏幕最左边会出现波浪符号，凡是有该符号就代表该行目前是空的。此时进入的是命令行模式。

要离开 vi 可以在底行模式下键入“:q”（不保存离开），而“:wq”（保存离开）则是存档后再离开（注意冒号）。如图 1.2 所示。

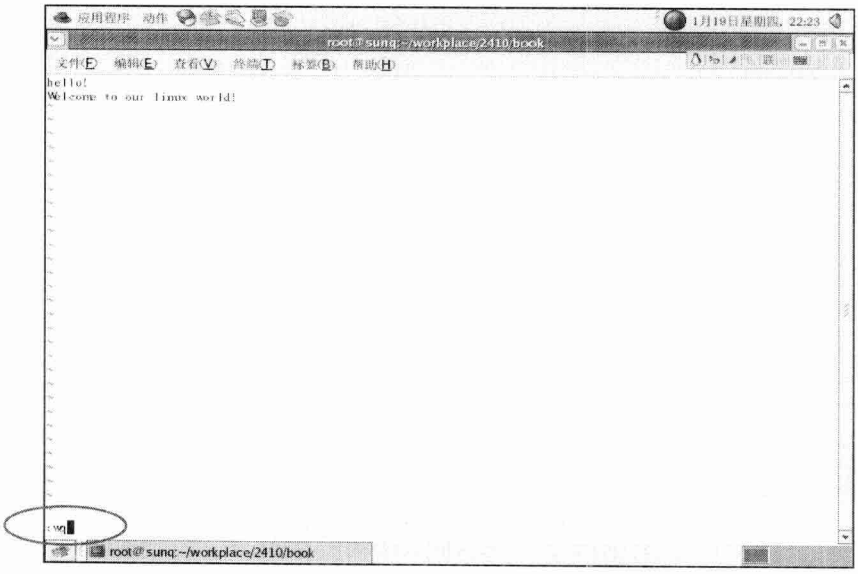


图 1.2　在 vi 中退出文档

2. vi 中 3 种模式的切换

在 vi 的使用中，3 种模式的切换是最为常用的。在处理的过程中，读者要时刻注意屏幕左下方的提示。在插入模式下，左下方会有“插入”字样，而在命令行或底行模式下则无提示。

（1）命令行模式、底行模式转为插入模式

在命令行模式或底行模式下转入到插入模式有 3 种方法，如表 1.1 所示。

表 1.1　　命令行模式、底行模式转为插入模式

特　征	指　令	作　用
新增	a	从光标所在位置后面开始新增资料，光标后的资料随新增资料向后移动
	A	从光标所在列最后面的地方开始新增资料
插入	i	从光标所在位置前面开始插入资料，光标后的资料随新增资料向后移动
	I	从光标所在列的第一个非空白字符前面开始插入资料
开始	o	在光标所在列下新增一列，并进入插入模式
	O	在光标所在列上方新增一列，并进入插入模式

在这里，最常用的是“i”，在转入插入模式后如图 1.3 所示。

（2）插入模式转为命令行模式、底行模式

从插入模式转为命令行模式、底行模式比较简单，只需使用［Esc］键即可。

（3）命令行模式与底行模式转换

命令行模式与底行模式间的转换不需要其他特别的命令，只需要直接键入相应模式中的命令

键即可。

图 1.3 命令模式转入插入模式

3. vi 的删除、修改与复制

在 vi 中进行删除、修改都可以在插入模式下使用键盘上的方向键及 Delete 键，另外，vi 还提供了一系列的操作指令，用以大大简化操作。

这些指令记忆起来比较复杂，希望读者能够配合操作进行实验。以下命令都是在命令行模式下使用的。

表 1.2 所示为 vi 的删除、修改与复制命令。

表 1.2　　vi 的删除、修改与复制命令

特　征	指　令	作　用
删除	x	删除光标所在的字符
	dd	删除光标所在的行
	s	删除光标所在的字符，并进入输入模式
	S	删除光标所在的行，并进入输入模式
修改	r 待修改字符	修改光标所在的字符，键入 r 后直接键入待修改字符
	R	进入取代状态，可移动光标键入所指位置的修改字符，该取代状态直到按［Esc］才结束
复制	yy	复制光标所在的行
	nyy	复制光标所在的行向下的 *n* 行
	p	将缓冲区内的字符粘贴到光标所在位置
	u	取消上一次的文本编辑操作

4. vi 的光标移动

由于许多编辑功能都是通过光标的定位来实现的，因此，掌握 vi 中光标移动的方法很重要。虽然使用方向键也可以实现 vi 的操作，但 vi 的指令可以实现复杂的光标移动，只要熟悉以后都非常方便，希望读者能切实掌握。

表 1.3 所示为 vi 中的光标移动指令，这些指令都是在命令行模式下使用的。

表 1.3　　vi 中光标移动的指令

指　令	作　用	指　令	作　用
0	移动到光标所在行的最前面	b	移动到上一个字的第一个字母
$	移动到光标所在行的最后面	w	移动到下一个字的第一个字母
Ctrl+d	光标向下移动半页	e	移动到下一个字的最后一个字母
Ctrl+f	光标向下移动一页	^	移动到光标所在行的第一个非空白字符
H	光标移动到当前屏幕的第一行第一列	n-	向上移动 *n* 行
M	光标移动到当前屏幕的中间行第一列	n+	向下移动 *n* 行
L	光标移动到当前屏幕的最后行第一列	nG	移动到第 *n* 行

5. vi 的查找与替换

vi 中的查找与替换也非常简单，其操作有些类似在 Telnet 中的使用。其中，查找的命令在命令行模式下，而替换的命令则在底行模式下（以“:”开头），其命令如表 1.4 所示。

表 1.4　　vi 的查找与替换指令

特　征	指　令	作　用
查找	/<要查找的字符>	向下查找要查找的字符
	?<要查找的字符>	向上查找要查找的字符
替换	:0,$s/string1/string2/g	0，$：替换范围从第 0 行到最后一行 s：转入替换模式 string1/string2:把所有 string1 替换为 string2 g：强制替换而不提示

6. vi 的文件操作指令

vi 中的文件操作指令都是在底行模式下进行的，所有的指令都是以“:”开头，其命令如表 1.5 所示。

表 1.5　　vi 的文件操作指令

指　令	作　用	指　令	作　用
: q	结束编辑，退出 vi	: wq	保存文档并退出
: q!	不保存编辑过的文档	: zz	功能与“: wq”相同
: w	保存文档，其后可加要保存的文件名	: x	功能与“: wq”相同

1.2.3　vi 的使用实例分析

本节给出了一个 vi 使用的完整实例，通过这个实例，读者一方面可以熟悉 vi 的使用流程，

另一方面也可以熟悉 Linux 的操作，希望读者能够先自己思考每一步的操作，再看后面的实例解析答案。

1. vi 使用实例内容

① 在“/root”目录下建一个名为 vi 的目录。

② 进入 vi 目录。

③ 将文件“/etc/inittab”复制到当前目录下。

④ 使用 vi 编辑当前目录下的 inittab。inittab 是/etc 下的系统配置文件.Linux 启动时会读取其内容。里面定义了默认的运行级别和要执行的程序。

⑤ 将光标移到该行。

⑥ 复制该行内容。

⑦ 将光标移到最后一行行首。

⑧ 粘贴复制行的内容。

⑨ 撤销第（8）步的动作。

⑩ 将光标移动到最后一行的行尾。

⑪ 粘贴复制行的内容。

⑫ 光标移到“si::sysinit:/etc/rc.d/rc.sysinit”。

⑬ 删除该行。

⑭ 存盘但不退出。

⑮ 将光标移到首行。

⑯ 插入模式下输入“Hello,this is vi world!”。

⑰ 返回命令行模式。

⑱ 向下查找字符串“0:wait”。

⑲ 再向上查找字符串“halt”。

⑳ 强制退出 vi，不存盘。

2. vi 使用实例解析

在该实例中，每一步的使用命令如下所示。

① mkdir /root/vi

② cd /root/vi

③ cp /etc/inittab ./

④ vi ./inittab

⑤ 17<enter>（命令行模式）

⑥ yy

⑦ G

⑧ p

⑨ u

⑩ $

⑪ p

⑫ 21G

⑬ dd

⑭ :w（底行模式）

⑮ 1G

⑯ i 并输入“Hello,this is vi world!”（插入模式）

⑰ Esc

⑱ /0:wait（命令行模式）

⑲ ?halt

⑳ :q!（底行模式）

1.3 嵌入式 Linux 编译器 GCC 的使用

1.3.1 GCC 概述

作为自由软件的旗舰项目，Richard Stallman 在刚开始编写 GCC 的时候，只是把它当作一个 C 程序的编译器，GCC 的意思也只是 GNU C Compiler 而已。

经过多年的发展，GCC 除了能支持 C 语言，目前还支持 Ada 语言、C++语言、Java 语言、Objective C 语言、PASCAL 语言、COBOL 语言，以及支持函数式编程和逻辑编程的 Mercury 语言等。GCC 也不再单指 GNU C 语言编译器，而是变成了 GNU 编译器家族。

正如前文中所述，GCC 的编译流程分为 4 个步骤，分别为以下内容。

① 预处理（pre-processing）。

② 编译（compiling）。

③ 汇编（assembling）。

④ 链接（linking）。

编译器通过程序的扩展名来分辨编写源程序所用的语言，由于不同的程序所需要执行编译的步骤是不同的，因此 GCC 根据不同的后缀名对它们进行相应的处理，表 1.6 指出了不同后缀名的处理方式。

表 1.6　　GCC 所支持不同后缀名的处理方式

后　缀　名	所对应的语言	编 译 流 程
.c	C 原始程序	预处理、编译、汇编
.C/.cc/.cxx	C++原始程序	预处理、编译、汇编
.m	Objective C 原始程序	预处理、编译、汇编
.i	已经过预处理的 C 原始程序	编译、汇编
.ii	已经过预处理的 C++原始程序	编译、汇编
.s/.S	汇编语言原始程序	汇编
.h	预处理文件（头文件）	（不常出现在指令行）
.o	目标文件	链接
.a/.so	编译后的库文件	链接