

嵌入式

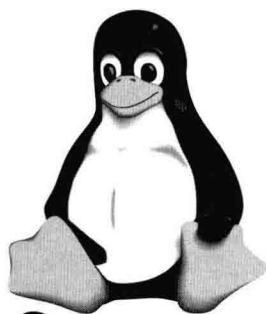
Linux

系统开发全程解析

韩超 等著



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>



嵌入式

Linux

系统开发全程解析

韩超 等著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书是一本全面介绍嵌入式 Linux 开发的专著，书中涵盖了程序生成工具、调试工具、引导加载器、Linux 系统结构、Linux 内核、驱动程序、用户空间编程、用户空间中间件等方面的内容。本书内容前后照应、贴近实践，且有较强的延伸型，有利于读者建立嵌入式 Linux 开发系统化的知识结构和理念。

本书不仅适用于嵌入式 Linux 的工程师增强能力，也适用于其他领域的技术人员了解嵌入式 Linux。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目 (CIP) 数据

嵌入式 Linux 系统开发全程解析 / 韩超等著. —北京：电子工业出版社，2014.5

ISBN 978-7-121-22888-9

I. ①嵌… II. ①韩… III. ①Linux 操作系统—程序设计 IV. ①TP316.89

中国版本图书馆 CIP 数据核字 (2014) 第 066337 号

策划编辑：李 冰

责任编辑：李云静

印 刷：中国电影出版社印刷厂

装 订：三河市鹏成印业有限公司

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：27.25 字数：646 千字

印 次：2014 年 5 月第 1 次印刷

定 价：59.00 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 zits@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

本书写作目的

嵌入式 Linux 开发已经不算很新的技术。本书作者从 2003 年开始从事 Linux 方面的研究、开发和科普工作。在 10 年多的时间里，Linux 内核已经从 2.4 版本发展到了 3.x 版本，广泛使用的嵌入式处理器也从 ARMv4 的 ARM7 核心发展成为 ARMv6 的 Cortex 核心。这 10 年多嵌入式 Linux 技术逐步成熟，产品已经数不胜数。

作者在几年前出版过一些有关嵌入式系统、Linux 和 C 语言编程的书籍。目前面对技术的发展，感觉有必要出版一本全面介绍嵌入式 Linux 的书籍，以帮助相关行业的学习者和开发者更高效地了解嵌入式 Linux，更好地从事相关工作。

希望将本书打造成简明、高效的工具书，成为快速开发 Linux 软件的指导书、硬件移植的工具书，以及计算机专业从业者理论联系实际的桥梁。

本书特点

本书结合了作者多年的开发经验和知识技术的传播经验，主要有下面一些特点。

- 内容来自工程实践，实用性强。
- 覆盖面更全面、知识系统完整。
- 使用框架图+代码路径+关键代码的方式，一目了然。
- 内容紧凑，读者可以结合手头代码对照学习。
- 将工程技巧蕴含于理论知识的网络之中。
- 包含 Linux 软件编程开发的常用技巧：查找代码、运行时看信息等。
- 结合硬件和操作系统的知识。
- 帮助读者深入理解 Linux 系统的关键结构，具有完备的开发调试能力。
- 重点关注目前的主要应用场景：用户空间开发和驱动开发。

本书以 Linux 尤其是嵌入式 Linux 中最常用的内容为主，这些内容大部分是 Linux 开发不同方面的工程师均需要掌握的。通过对本书的学习，可以让有 C 语言基础的工程师在 Linux 环境中开发用户空间软件；让有硬件基础的工程师可以在嵌入式 Linux 平台中具有适

配硬件的能力；让具有一定经验的嵌入式 Linux 工程师具有更广泛的视野、更强的开发能力。

本书主要内容

本书包含了嵌入式 Linux 系统的主要内容，按照知识结构分成四个方面。

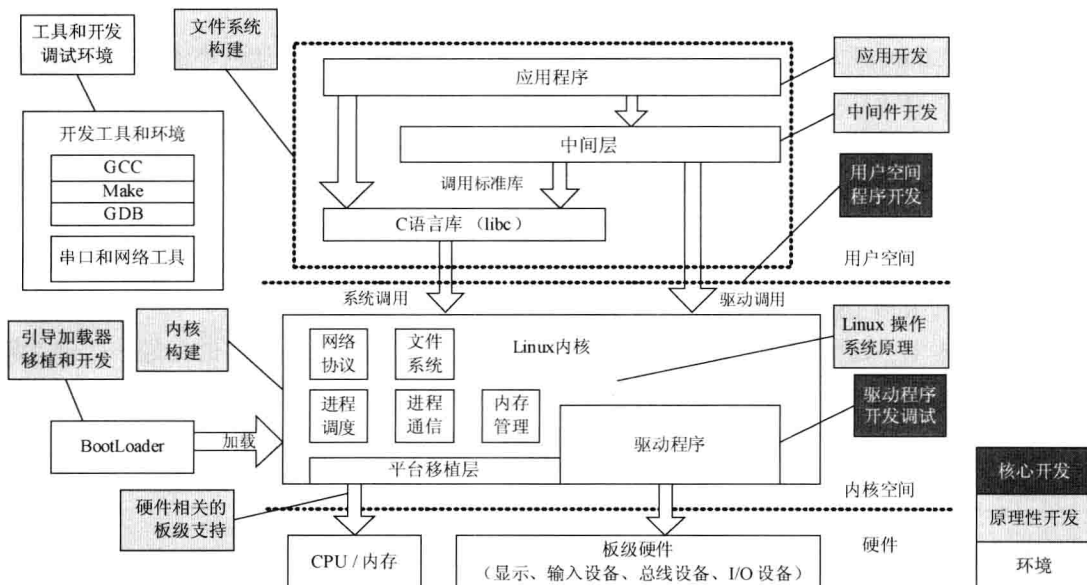
第一个方面：开发环境和编程基础（第 1 章到第 4 章）。

第二个方面：Linux 系统结构（第 5 章到第 8 章）。

第三个方面：嵌入式 Linux 的用户空间（第 9 章到第 13 章）。

第四个方面：嵌入式 Linux 的驱动开发（第 14 章到第 24 章）。

按照内容的侧重点，本书具有环境、原理性开发和核心开发几个方面。嵌入式 Linux 系统和本书知识结构如下图所示。



对读者的话

目前的 IT 技术领域有很多热点，除了嵌入式设备的开发外，还有移动开发、互联网开发等。嵌入式 Linux 是很多产品的技术基础。在实践过程中，很多问题都是嵌入式 Linux 最基础的问题，只是由于工程师可能来自其他领域，不熟悉嵌入式 Linux，从而小问题就成了大问题。因此，对于嵌入式 Linux，是目前从事 IT 技术领域的各类工程师都至少需要了解的。

本书不仅适用于嵌入式 Linux 的工程师增强能力，也适用于其他领域的技术人员了解嵌入式 Linux。本书尤其注重和高校计算机专业的互补关系，基于高校计算机专业知识基础，书中大量补充了在实践中的应用，帮助读者将知识“落地扎根”，引领读者进一步在工作中

让知识和技能“生根发芽”，直到在工作中“结出丰硕的果实”。

本书的几个基础方面是 C 语言编程、嵌入式处理器、操作系统，这也是计算机行业的基础。通过对本书的学习，读者得到的不仅仅是各个方面的知识和技巧，更有它们之间的有机结合。

本书作者

韩超是中国大陆长期工作于一线的知名工程师、架构师，也是嵌入式 Linux 相关技术在大陆发展 10 年的技术领航人之一，同时也是畅销书作者。其主要从事相关技术研发方向，包括嵌入式 Linux 板级平台、GUI 系统和应用、移动多媒体。韩超对嵌入式 Linux 的技术把握以实用技术为主，以操作系统本身为辅，重视在系统使用特定硬件的技术，重视内核与用户空间的交互的要点，适用于嵌入式 Linux 的软件工程等方面。

韩超完成了本书主要部分的编写工作，此外，众多不同规模的企业开发成果也为本书的编写提供了重要的素材。参与本书编写的还有康硕、于仕林、张超等人，以及清华大学计算机系操作系统研究兴趣小组的肖奇学、徐永健、王欢、何嘉权、范文良、茅俊杰等人。

目 录

第 1 章 Linux 的开发环境	1	3.2.3 Makefile 中的函数	36
1.1 开发环境概述	1	3.3 Makefile 使用示例	39
1.2 串口终端工具	2	3.3.1 简单的 Makefile	39
1.3 TFTP	6	3.3.2 依赖关系实例	39
1.4 NFS	7	3.3.3 隐含规则的编译实例	41
1.5 SAMBA 共享	8	3.3.4 指定依赖的编译实例	44
1.6 Linux 系统的软件发布协议	9	3.4 自动生成 Makefile	46
第 2 章 程序生成和 GCC	11	3.4.1 autoconf 工具介绍	46
2.1 程序生成工具概述	11	3.4.2 automake 工具介绍	46
2.1.1 GUN 的 GCC 工具	11	3.4.3 其他工具	47
2.1.2 ELF 文件格式	14	3.4.4 自动生成 Makefile 的流程	47
2.2 GCC 工具的使用	16	第 4 章 调试和 GDB	49
2.2.1 示例工程	16	4.1 嵌入式系统的调试技术	49
2.2.2 编译、汇编和连接	18	4.1.1 调试技术	49
2.2.3 预处理和汇编	20	4.1.2 硬件调试	50
2.2.4 归档工具 (ar) 和静态库	20	4.1.3 代码调试	51
2.2.5 动态库	22	4.2 Linux 的基本信息	51
2.2.6 ELF 格式文件信息读取		4.3 GDB 调试和远程调试	52
(readelf)	22	4.4 GDB 的安装与使用	57
2.2.7 符号信息工具 (nm)	25	4.4.1 使用 gdbstub 实现调试用户	
2.2.8 字符串工具 (strings)	26	程序	57
2.2.9 去除符号 (strip)	27	4.4.2 GDB 和 GDB Server 的编译	59
2.2.10 目标文件复制 (objcopy)	28	4.5 使用 gdbserver 调试	61
2.2.11 目标文件信息 (objdump)	28	第 5 章 Linux 系统的结构	65
第 3 章 工程管理和 make 机制	33	5.1 Linux 操作系统基本概念	65
3.1 make 工具	33	5.1.1 Linux 的进程信息	65
3.2 Makefile 的基本原则	34	5.1.2 Linux 的文件系统和文件	
3.2.1 Makefile 的变量	34	信息	70
3.2.2 Makefile 的条件执行	36	5.1.3 文件的另外三位属性	71
		5.2 Linux 系统的组成和构建	72

5.2.1 Linux 系统的组成.....	72	第 8 章 文件系统及其构建	123
5.2.2 嵌入式 Linux 的构建.....	73	8.1 Linux 文件系统特性.....	123
5.3 Linux 系统的启动流程.....	74	8.2 Linux 文件系统的结构.....	125
第 6 章 BootLoader 及其构建	76	8.2.1 文件系统的主要接口.....	125
6.1 嵌入式 Linux 的 BootLoader.....	76	8.2.2 文件系统的实现.....	130
6.1.1 BootLoader 的开发要点.....	76	8.2.3 默认的公共实现.....	134
6.1.2 BootLoader 的结构.....	78	8.3 几种 Linux 使用的文件系统.....	136
6.2 U-Boot 的使用.....	80	8.3.1 EXT2/3 (扩展文件系统2/3)	136
6.2.1 U-Boot 概述.....	80	8.3.2 NFS (网络文件系统)	136
6.2.2 U-Boot 的结构.....	81	8.3.3 ROMFS (只读文件系统)	137
6.2.3 U-Boot 的生成.....	83	8.3.4 CRAMFS (压缩 ROM 文件	
6.2.4 U-Boot 的启动流程.....	84	系统)	137
6.3 U-Boot 的命令.....	86	8.3.5 JFFS2 (日志 Flash 文件	
6.3.1 U-Boot 命令概述.....	86	系统)	138
6.3.2 增加命令.....	88	8.3.6 YAFFS (另一种 Flash 文件	
6.4 U-Boot 的移植.....	91	系统)	138
6.4.1 U-Boot 的移植概述.....	92	8.3.7 UBIFS (非排序块映像文件	
6.4.2 U-Boot 的扩展.....	92	系统)	139
6.4.3 板级支持.....	94	8.4 Linux 文件系统的构建.....	140
第 7 章 Linux 内核及其构建	97	8.4.1 根文件系统的结构.....	140
7.1 Linux 内核概述.....	97	8.4.2 制作根文件系统映像.....	141
7.1.1 Linux 内核结构.....	97	8.4.3 内核启动中根文件系统的	
7.1.2 Linux 源文件结构.....	98	参数.....	142
7.2 嵌入式 Linux 的配置和编译.....	99	第 9 章 Linux 用户空间的核心	143
7.2.1 Linux 内核配置结构.....	99	9.1 嵌入式系统中的操作系统和系统关系...	143
7.2.2 Linux 内核的配置.....	99	9.2 C 语言库.....	144
7.2.3 Linux 内核的生成.....	107	9.3 Shell 工具 Busybox.....	147
7.3 Linux 内核的启动过程.....	108	9.3.1 Busybox 配置和编译.....	148
7.4 特定系统的 Linux 的构建.....	114	9.3.2 Busybox 的源代码结构.....	150
7.4.1 Linux 内核的移植.....	114	第 10 章 Linux 用户空间的编程	152
7.4.2 ARM 处理器上运行的 Linux		10.1 Linux 用户空间编程概述.....	152
系统.....	115	10.2 文件的相关内容.....	154
7.4.3 S3C6410 Linux 内核的构建....	117	10.2.1 文件的打开、关闭和	
7.4.4 S3C6410 Linux 内核的移植		读写等.....	155
内容.....	118	10.2.2 文件的控制、映射和	
		查询等.....	157

10.2.3	文件的其他操作	158	12.1.1	驱动的理念和结构	228
10.3	进程相关的内容	159	12.1.2	驱动程序对用户空间的 接口	230
10.3.1	fork 和 exec	159	12.2	设备文件和相关文件系统	230
10.3.2	管道	161	12.2.1	设备文件	230
10.3.3	System V IPC	162	12.2.2	sys 文件系统	231
10.3.4	POSIX IPC	165	12.2.3	proc 文件系统	233
10.4	信号相关的内容	166	第 13 章	Linux 的内核编程	237
10.5	pthread 线程	168	13.1	Linux 内核编程概述	237
10.5.1	线程的基本使用	169	13.2	内核模块的编写	237
10.5.2	线程的属性	171	13.2.1	Linux 内核中的模块	237
10.5.3	线程互斥量	172	13.2.2	内核模块的编译结构	239
10.5.4	线程条件量	173	13.3	内核编程接口	241
10.5.5	线程取消	175	13.3.1	Linux 编程风格	241
10.6	dlopen 机制	176	13.3.2	Linux 编程主要接口	242
10.6.1	dlopen 的结构和意义	176	第 14 章	Linux 的驱动核心架构	248
10.6.2	在 C 语言中使用 dlopen	178	14.1	用户空间的接口	248
10.6.3	在 C++ 中使用 dlopen	180	14.1.1	用户空间的驱动调用接口	248
第 11 章	Linux 用户空间的中间件	185	14.1.2	系统调用	248
11.1	基于嵌入式 Linux 的系统与中间件	185	14.1.3	驱动的主要调用函数	249
11.2	网络协议相关	186	14.2	字符设备和块设备的框架	250
11.2.1	Linux 套接字编程的基础	186	14.2.1	文件操作 file_operations	250
11.2.2	TCP 和 UDP 协议的流程	189	14.2.2	字符设备的基本框架	251
11.2.3	TCP 编程实例	189	14.2.3	块设备的框架	252
11.2.4	UDP 编程实例	193	14.2.4	字符设备和块设备的默认 file_operations 实现	254
11.2.5	深入网络编程	196	14.3	网络协议和网络设备的框架	258
11.2.6	用作 IPC 的 UNIX Socket	198	14.3.1	网络系统的核心	259
11.3	GUI 应用开发	201	14.3.2	网络协议的实现	261
11.3.1	Qt 系统	203	14.3.3	网络设备的框架	263
11.3.2	MiniGUI 应用程序	209	14.4	proc 文件系统的框架	264
11.3.3	MicroWindows (Nano-X Window)	216	14.4.1	proc 文件系统的编程接口	264
11.4	数据库	217	14.4.2	proc 文件系统的实现	266
11.4.1	关于嵌入式数据库	217	14.5	sys 文件系统的框架	266
11.4.2	SQLite	218	14.5.1	sys 文件系统的编程接口	266
第 12 章	Linux 驱动基础	228	14.5.2	sys 文件系统的实现	267
12.1	Linux 驱动概述	228			

第 15 章 Linux 驱动的要 点	269	17.4.1 GPIO 驱动框架	310
15.1 驱动程序的核心实现	269	17.4.2 GPIO 具体硬件的驱动	312
15.2 设备、驱动和资源	273	17.5 Power Supply 驱动框架和具体驱动 ...	312
15.3 中断的处理	276	17.5.1 Power Supply 驱动框架	312
15.4 中断的下半部	277	17.5.2 Power Supply 驱动	313
15.4.1 软中断	277	17.6 TTY 驱动框架和驱动	314
15.4.2 软中断之 tasklet	278	17.6.1 TTY 驱动框架	314
15.4.3 软中断之定时器	279	17.6.2 伪 TTY 驱动	316
15.5 竞态处理	280	17.6.3 串口 TTY 和虚拟 TTY	316
15.5.1 自旋锁	280	第 18 章 MTD 系统和驱动	318
15.5.2 信号量	280	18.1 MTD 概述	318
15.6 阻塞处理	281	18.2 MTD 的核心	319
15.7 异步操作	282	18.2.1 MTD 的接口部分	320
第 16 章 几个典型的简单驱动	283	18.2.2 MTD 的核心实现部分	322
16.1 设备驱动概述	283	18.3 MTD 的设备层	322
16.2 内存设备驱动	284	18.3.1 MTD 字符设备	322
16.2.1 内存设备驱动的公共内容	284	18.3.2 MTD 块设备	323
16.2.2 空设备	286	18.4 CFI 硬件实现层	324
16.2.3 零设备	287	18.4.1 公用部分	324
16.2.4 满设备	288	18.4.2 ROM 的 MTD 实现	325
16.3 内存块设备驱动	288	18.4.3 RAM 的 MTD 实现	325
16.4 回环块设备驱动	291	18.4.4 Nor Flash 的 MTD 实现	326
16.5 回环网络设备驱动	294	18.5 Nand Flash 的硬件实现层	326
第 17 章 几个典型的驱动框架和相应 的驱动	296	18.5.1 公用部分	326
17.1 Misc 驱动框架	296	18.5.2 GPIO 的 Nand Flash 实现	327
17.2 帧缓冲驱动框架和具体驱动	297	18.5.3 处理器芯片上的 Nand Flash 实现	330
17.2.1 帧缓冲驱动框架	297	第 19 章 USB 系统和驱动	331
17.2.2 虚拟帧缓冲驱动	300	19.1 USB 概述	331
17.2.3 针对硬件实现的帧缓冲 驱动	302	19.1.1 USB 规范	331
17.3 输入-事件驱动框架	305	19.1.2 USB 的软件系统	333
17.3.1 输入-事件驱动框架概述	305	19.2 Linux 的 USB 主机端支持	334
17.3.2 针对硬件的事件驱动	307	19.2.1 USB 主机端的软件结构	334
17.4 GPIO 驱动框架和具体驱动	310	19.2.2 USB 主机端的核心部分	335
		19.2.3 USB 驱动的实现	337
		19.2.4 HCI 的实现	339

19.3	Linux 的 USB 设备端支持	340	22.1.4	PCI 的总线配置	379
19.3.1	USB 设备端的软件结构	340	22.1.5	PCI 的发展和衍生标准	381
19.3.2	Gadget 的核心部分	340	22.2	PCI 总线的驱动框架	381
19.3.3	Gadget 驱动	342	22.3	PCI 设备的驱动	384
19.3.4	UDC 驱动的实现	345	22.3.1	PCI 的桩实现	384
第 20 章	SPI 总线和驱动	348	22.3.2	网卡的 PCI 实现	385
20.1	SPI 概述	348	第 23 章	音频系统和驱动	389
20.2	SPI 总线驱动的框架	349	23.1	音频系统概述	389
20.3	简单字符设备 spidev	353	23.2	OSS 架构	389
20.4	SPI 主控制器的实现	355	23.2.1	OSS 系统的结构	390
20.4.1	GPIO 实现的 SPI 主控制器	355	23.2.2	OSS 系统的核心	391
20.4.2	S3C64xx 的 SPI 主控制器	356	23.2.3	OSS 系统的实现	392
20.5	SPI 从设备的驱动	358	23.3	ALSA 架构	393
第 21 章	I2C 总线和驱动	361	23.3.1	ALSA 系统的结构	393
21.1	I2C 概述	361	23.3.2	ALSA 系统的核心	395
21.1.1	基本概念	361	23.3.3	ALSA 系统芯片层	395
21.1.2	SMBus	362	23.3.4	ALSA 的用户空间	400
21.2	I2C 总线驱动的框架	362	第 24 章	视频系统和驱动	403
21.2.1	I2C 核心框架	362	24.1	视频系统概述	403
21.2.2	I2C 总线接口	367	24.2	Video for Linux 系统	403
21.2.3	I2C 设备和驱动	368	24.2.1	基本结构	404
21.3	具体的 I2C 主控制器	370	24.2.2	Video for Linux 的核心 结构	405
21.4	I2C 从设备的驱动	372	24.2.3	Video for Linux 的其他 方面	410
第 22 章	PCI 总线和驱动	375	24.2.4	Video for Linux 驱动的 接口	413
22.1	PCI 概述	375	24.2.5	Video for Linux 驱动的实 现层	417
22.1.1	PCI 的基本结构	375			
22.1.2	PCI 的总线信号	377			
22.1.3	PCI 的总线操作	378			

第1章

Linux 的开发环境

1.1 开发环境概述

在 Linux 和嵌入式 Linux 的开发中，一般将生成代码的机器称为“主机”，而将运行代码的机器称为“目标机”。主机和目标机不是同一个机器的开发，则被称为交叉开发。

主机和目标机代码生成的过程如图 1-1 所示。

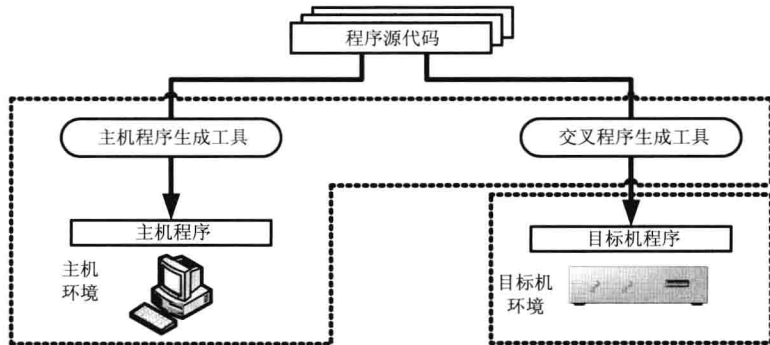


图 1-1 主机和目标机的代码生成

如果是单纯桌面（x86）上的程序开发，主机和目标机的结构类似，生成的程序一般可以在开发机和目标机同时运行。如果在嵌入式系统中开发，通常在主机环境中开发，通过交叉编译工具生成目标机的程序，然后通过某种手段放入目标机系统中运行。

对于嵌入式的开发，一般分成主机和目标机通常是两个不同的设备。在嵌入式 Linux 的开发中，主要是指目标机使用 Linux 作为操作系统，并在操作系统上构造不同应用。嵌入式 Linux 开发的主机环境，主要是指在主机上生成运行于目标机的代码。主机一般是运行 Linux 操作系统的 x86 主机，也可以在 Windows 上开发嵌入式 Linux。

Linux 系统最基本的程序开发是 C 语言的程序开发。C 语言程序的生成分成编译、汇

编、连接等几个步骤。最终的目标文件的主要部分是处理器可执行的机器代码组合。根据系统的不同，生成的目标包括了可执行的二进制机器代码，也包括一定的头信息。

Linux 系统的开发主要涉及了如下内容：

- GCC 和交叉 GCC 均运行于开发机，但程序生成的部分存在区别。
- Make 和 Makefile 运行于开发机。
- GDB 可分为运行于主机的部分和运行于目标机的部分，也适用于本机开发。
- 主机-目标机的连接方式有多种，也适用于本机开发。

在嵌入式系统的开发中，除了程序生成工具之外，还需要使用主机到目标机的通信工具。主要的工具包括目标机的终端模拟工具和文件传输工具，前者一般使用从主机到目标机的串口连接完成，后者一般使用网络协议来实现。

嵌入式系统主机和目标机的连接如图 1-2 所示。

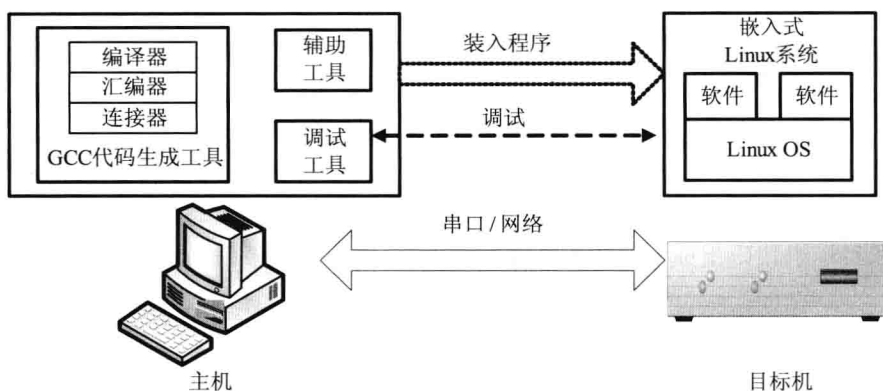


图 1-2 主机-目标机的开发结构

在嵌入式系统的开发中，需要使用主机到目标机的通信工具。主要的工具包括目标机的终端模拟工具和文件传输工具，前者一般使用从主机到目标机的串口连接完成，后者一般使用网络协议来实现。

主机到目标机通常有两条通道：一条是目标机 UART 到主机串口的连接，可以提供打印信息和控制台支持；另一条是网络连接，可以建立于网络协议 TCP/IP 之上，这样可以屏蔽数据链路层以下的细节，并且有多种基于上层协议的软件可以使用。基于网络协议的通信不仅限于在网络芯片硬件上，也可以通过 USB 或其他通信端口作为网络协议的硬件基础。

1.2 串口终端工具

串口是嵌入式开发过程中，主机和目标机最基本的连接方式，可以用于字符形式的输出和输入。例如：在 C 语言的程序开发中，使用 `printf()` 作为调试语句，主机的终端常使用显示器作为屏幕输出。在嵌入式系统的目标机调试过程中，由于一般不具有屏幕等输出设

备，因此通常使用目标机处理器的串口作为输出，在主机端使用串口工具接收这些信息。同样，在目标机需要交互调试的时候，串口的连接方式也可以提供目标机的输入，例如：getchar()、scanf()等功能。在开发过程中，终端虽然硬件基础不一样，但是其表现形式是类似的。

使用 UART 连接的几个主要参数。

- 波特率 (Speed, Baud rate)，串口数据的速率，一般可能有 1200 b/s、38 400 b/s、115 200 b/s 等。
- 数据位 (Data)：通常有 5、6、7、8 几种。
- 奇偶校验方式 (Parity)：一般有无校验 (None)、偶校验 (Even)、奇校验 (Odd)、标记 (Mark)、空格 (Space) 等。
- 停止位 (Stopbits)：一般有 1 位、2 位两种方式。
- 流控制 (Flow Control)：使用硬件流控制或者软件流控制。

提示：如果主机的串口终端的输出为乱码，则通常是波特率设置的问题；如果主机串口的控制台不能输入，通常是流控制的设置问题。

对于主机上使用的串口终端工具，在 Windows 系统中可使用超级终端，在 Linux 系统中可以使用 minicom 工具。

1. Windows 的超级终端

超级终端是 Windows 自带的工具，在使用的过程中很简单。在超级终端中，选择“文件”→“新建连接”，在建立的过程中，需要设置连接方式为 com (串口)，根据实际连接的情况选择使用 com1 或 com2，设置的对话框如图 1-3 所示。

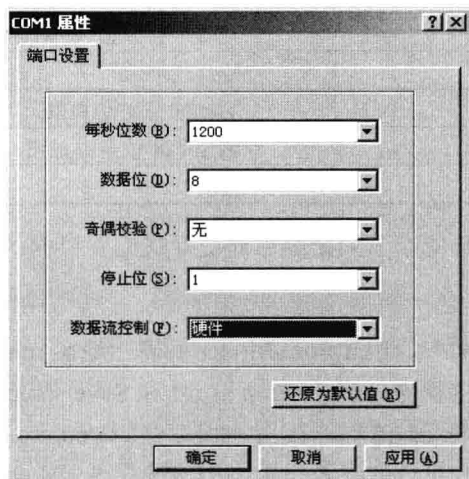


图 1-3 主机-目标机的连接

以上设置的参数，需要与目标机处理器的 UART 端口的设置一致，目标机的设置一般在 UART 的初始化过程中通过配置寄存器完成。

2. Linux 的 minicom

minicom 是 Linux 下的串口终端工具,使用 minicom 可以在 Linux 下实现目标机和主机的连接。minicom 可以建立于 Linux 的串口设备 (ttyS0) 或者调制解调器设备 (modem) 上,其使用起来没有 Windows 下的工具友好,需要从命令行启动。

minicom 命令行启动的参数如下所示:

```
minicom [OPTION]... [configuration]
```

minicom 命令主要参数的含义如表 1-1 所示。

表 1-1 minicom 的参数

参数	含义
-s, --setup	输入建立模式, 仅在 root 上
-o, --noinit	不要初始化调制解调器
-m, --metakey	使用 Alt 或 meta 键作为命令键
-M, --metakey8	使用 8 位的 meta 键作为命令键
-l, --ansi	逐字翻译, 假设屏幕使用 IBM-PC 字符集
-L, --iso	同 L, 但是假设屏幕使用 SO8859
-w, --wrap	换行
-z, --statline	使用终端状态行
-8, --8bit	8 位干净模式, 例如为日本字符所使用的
-c, --color=on/off	ANSI 样式的颜色使用开关
-a, --attrib=on/off	用户反转高亮的开关
-t, --term=TERM	重载 TERM 环境变量
-S, --script=SCRIPT	使用启动的脚本文件
-d, --dial=ENTRY	从拨号目录拨入 ENTRY
-p, --ptty=TTY	连接伪终端
-C, --capturefile=FILE	启动捕获文件 FILE
-v, --version	输出版本信息并退出
configuration	配置文件

一种较为方便的方式是使用下列命令启动交互式配置界面:

```
# minicom -s
```

在 minicom 主配置界面中, Exit from Minicom 表示退出 minicom 应用, Exit 表示退出配置程序, 进入 minicom 主程序。Save setup as dfl 和 Save setup as 可以选择保存当前的配置。minicom 选择 Save setup as dfl 时, 配置将保存在/etc/minirc.dfl 文件中。

配置的主要工作是针对串口的参数设置, 因此选择 Serial port setup 选项。

minicom 主配置界面和串口配置界面如图 1-4 所示。

```

+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup           |
| Modem and dialing           |
| Screen and keyboard         |
| Save setup as dfl           |
| Save setup as..            |
| Exit                        |
| Exit from Minicom          |
+-----+
| A - Serial Device           : /dev/ttyS0
| B - Lockfile Location       : /var/lock
| C - Callin Program          :
| D - Callout Program         :
| E - Bps/Par/Bits           : 115200 8N1
| F - Hardware Flow Control   : Yes
| G - Software Flow Control   : No
+-----+
| Change which setting? [ ]
+-----+
| Screen and keyboard         |
| Save setup as dfl           |
| Save setup as..            |
| Exit                        |
| Exit from Minicom          |
+-----+

```

图 1-4 minicom 主配置界面（左）和串口配置界面（右）

在 minicom 串口界面中，可以使用 A~G 等字母选择相应的配置内容，A 用于选择串行的设备，使用 Linux 下的设备名称，如/dev/modem、/dev/ttyS0 等。

在串口终端的配置中，主要需要设置的内容是波特率、数据位等内容，使用键 E 启动串口的配置界面，如图 1-5 所示。

```

文件(E) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
+-----[Comm Parameters]-----+
| A - Serial Device           : Current: 115200 8N1
| B - Lockfile Location       :
| C - Callin Program         : Speed      Parity      Data
| D - Callout Program        :
| E - Bps/Par/Bits           : A: 300      L: None      S: 5
| F - Hardware Flow Control   : B: 1200     M: Even      T: 6
| G - Software Flow Control   : C: 2400     N: Odd       U: 7
|                             : D: 4800     O: Mark      V: 8
| Change which setting?     : E: 9600     P: Space
|                             : F: 19200
|                             : G: 38400   Stopbits
|                             : H: 57600   W: 1
|                             : I: 115200  X: 2
|                             : J: 230400
|                             : Q: 8-N-1
|                             : R: 7-E-1
+-----+
| Choice, or <Enter> to exit? █
+-----+

```

图 1-5 串口波特率、数据位、奇偶校验、停止位界面

本界面用于配置，串口波特率、数据位、奇偶校验、停止位使用按键选择实现。在配置完成后，可以使用没有命令行参数的方式启动 minicom。

在 minicom 启动主界面中如果有来自串口的数据，将在屏幕上显示。在主界面下可以继续使用 Ctrl+A 组合键，然后再按 Z 键实现进行配置。minicom 启动主界面的帮助界面，如图 1-6 所示。

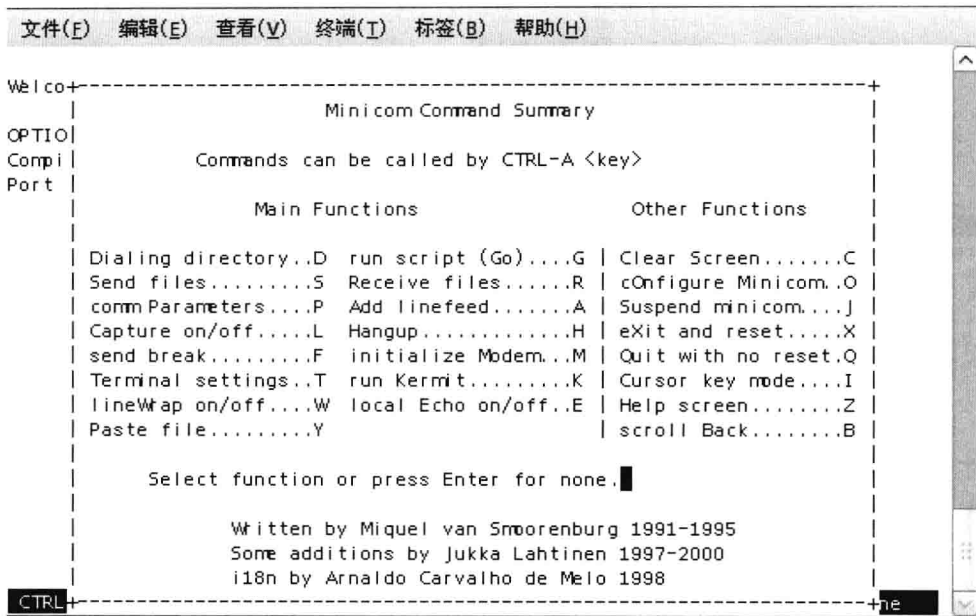


图 1-6 minicom 启动主界面的帮助界面

在主界面的帮助界面中，可以完成一些使用过程中的配置和辅助功能。

1.3 TFTP

TFTP 的全称为 Trivial File Transfer Protocol（简单文件传输协议）。此协议设计时的目标是进行小文件传输的。与基于 TCP 的 FTP（File Transfer Protocol，文件传输协议）不同，TFTP 使用 UDP 的端口 69。TFTP 只能从文件服务器上获得或写入文件，不能列出目录，不进行认证，按照 8 位来传输数据。

TFTP 使用以下 3 种传输模式。

- Netascii: 这是 8 位的 ASCII 码形式。
- octet: 这是 8 位源数据类型。
- mail: 它将返回的数据直接返回给用户而不是保存为文件。

在嵌入式系统的开发过程中，TFTP 常常被使用作为主机-目标机文件传输的协议。一个典型的用法是在主机端开启 TFTP 服务器，在目标机端由是支持 TFTP 的 Bootloader 或者运行于嵌入式 Linux 操作系统的 TFTP 客户端。

例如：在 Ubuntu 环境中设置 TFTP 服务器，需要下载以下的包：

```
$ sudo apt-get install tftpd-hpa
```

进一步，修改 etc 目录中的配置文件/etc/default/tftp-hpa:

```
#Defaults for tftpd-hpa
RUN_DAEMON="no"
OPTIONS="-l -s /var/lib/tftpboot"
```